Universidad de Las Palmas de Gran Canaria
Departamento de Telemática

Protocolos de Transporte en Redes de Comunicaciones

Domingo Marrero
Departamento de Telemática
Noviembre 2000

Título: Protocolos de Transporte en Redes de Comunicaciones

Editado e impreso por el Servicio de Reprografía de la Universidad de Las Palmas de Gran Canaria, Enero2001. Campus Universitario de Tafira. Las Palmas de Gran Canaria (35017).

Fecha de impresión: Enero 2001

Fotocopiadora:

- Marca: Rank Xerox
- Modelo: 5090
- Número de Serie: 1104236487

Dpto. Legal: **GC–1759–2000**
ISBN : **84-87526-91-8**

PROLOGO

El presente documento pretende ser una referencia sobre los grandes olvidados dentro de las arquitecturas de comunicaciones, como son los protocolos de transporte. Si bien representan el "cerebro" de un sistema de comunicaciones en red, la manifestación más relevante de cara a los usuarios de los protocolos de red, como pueden ser X.25, IP, etc.. los releva a un segundo plano.

Este mayor protagonismo de las capas inferiores se debe a su acceso y configuración por parte de administradores y usuarios, pero los resultados no son óptimos si no se dispone con los adecuados protocolos de transporte. Dada la importancia que tienen, ya que son los encargados de detectar pérdidas, duplicidades o desordenes, para destacar estos protocolos y darles el adecuado valor dentro de una red, se presenta esta documentación que abarca los principales protocolos de transporte actuales y que predominan en la mayoría de las redes.

Especial mención se presta a la capa de transporte propuesta por OSI, si bien no ha llegado a ser lo que se pretendía hace unos años, si permite ser un marco de referencia ideal para cualquier implementación de un protocolo de transporte. Asimismo destacan sobremanera, TCP (Transmission Control Protocol) y UDP(User Datagram Protocol), principales protocolos de transporte en internet y que son analizados minuciosamente. Por último, se dan breves referencias sobre los protocolos próximos a transporte Netware IPX (Internetwork Packet eXchange) y SPX (Sequenced Packet eXchange). Finalmente se dan idea de la utilización de los TLI (Transport Layer Interface) para servicio de comunicaciones sobre una capa de transporte.

Este documento puede ser utilizado por cualquier alumno con conocimientos generales de arquitecturas de redes, especialmente TCP/IP o el modelo OSI, para asimilar mejor los diferentes aspectos mostrados en los diferentes temas. Especialmente esta dirigido a alumnos de la titulación de Telemática de la Escuela Universitaria de Ingeniería Técnica de Telecomunicación de la ULPGC.

El Autor

Protocolos de Transporte en Redes de Comunicaciones

INDICE

Capa de Transporte OSI

1. CAPA DE TRANSPORTE (ISO8072/8073)

1.1 Introducción.-

Con esta documentación se pretende describir la especificación de la capa de transporte en el contexto del Modelo de Referencia OSI de la ISO.

Si partimos de un servicio de red inseguro que envía continuamente N_RESET todo el tiempo, habrá que aplicar algún tipo de solución. De entre ellas tenemos a) Mejorar los conmutadores o IMP incrementando el tratamiento de errores en nivel de enlace o b) Colocar una capa por encima de la capa de red que mejore la calidad del servicio.

Según esta última solución, si ha esta entidad, superior a la capa de red, se le avisa, a la mitad de una larga transmisión, que se ha interrumpido súbitamente su conexión de red, sin ninguna indicación respecto a lo que incidió con los datos que se encontraban en tráfico, ella puede establecer una nueva conexión de red con la entidad remota. Mediante ella puede enviar una pregunta para averiguar que datos llegaron y cuáles no, y después reiniciar la transmisión a partir del momento en el cual se perdieron. De esta manera, los paquetes extraviados, los datos dañados, e incluso los N_RESET de la red pueden ser detectados y compensados por esta capa. Ante esta situación se define la **capa de transporte.**

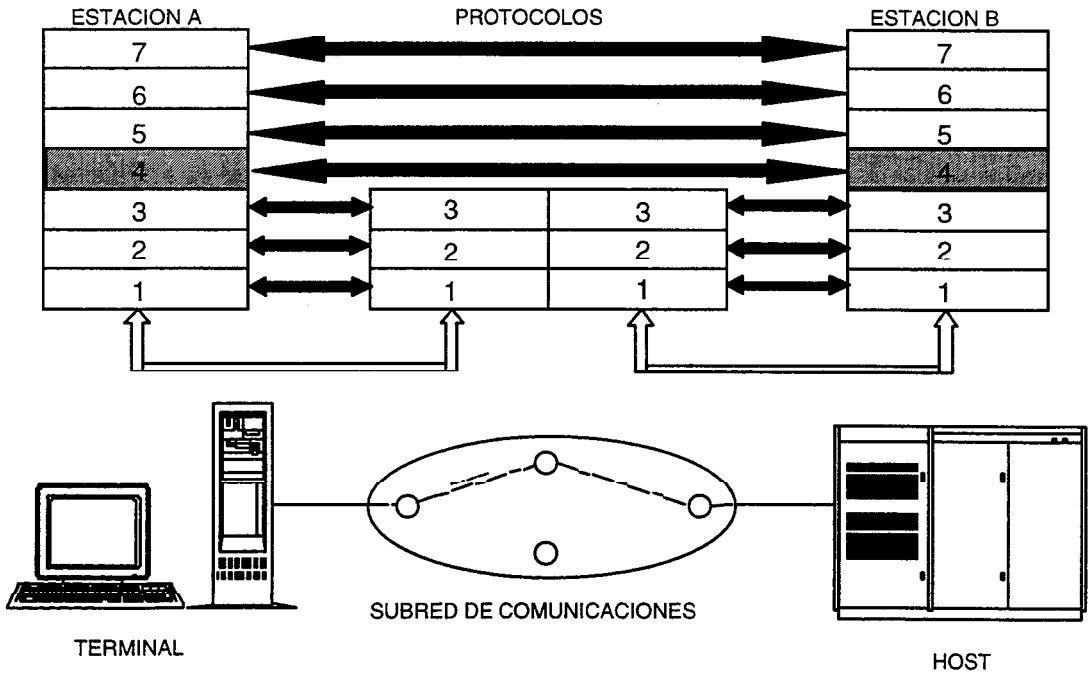


Fig. 1.1 RM-OSI (capas en nodos y hosts)

La capa de transporte es la cuarta capa del modelo de referencia OSI, vista en orden creciente desde la capa física (1) y es la primera de las denominadas orientadas a la aplicación. Es la primera, en orden ascendente, cuya comunicación con su **entidad par** es **extremo a extremo** (ver figura 1.1) lo cual le da una funcionalidad muy importante,

al estar ubicada adyacente a la **capa de red.** Tanto la capa de red como las inferiores intercambian unidades de datos en modo **"punto a punto"** con el siguiente elemento de la red (nodo intermedio o estación) a diferencia del resto de las capas superiores (extremo a extremo). Desarrolla un papel muy importante y viene a ser realmente el *corazón de la jerarquía de protocolos.*

A su vez se considera la primera que forma parte de las capas **proveedoras del servicio de transporte de datos** (capas 1, 2, 3, y 4) entre sistemas abiertos (ver figura 1.2) y las capas **usuarias del servicio de transporte** (capas 5, 6 y 7).

La capa de transporte tiene como función ofrecer a la capa de sesión un recurso de transporte de mensajes transparente y fiable (libre de errores, en orden y sin duplicidades) o garantizar la integridad extremo a extremo de los paquetes de protocolos de las capas altas.

El concepto de integridad extremo a extremo de los datos transferidos, como ya se indicó, incluye asegurar que las PDUs no dañadas sean entregadas y que las PDUs no se pierdan, dupliquen o lleguen fuera de orden. **ACKs (reconocimientos)** son entregados para indicar que las PDUs han sido recibidas satisfactoriamente. No obstante, estas ACKs no necesariamente significan que el contenido de los paquetes sean almacenados correctamente. ACKs de capas altas o **mecanismos de recuperación** son necesarios cuando una recuperación se necesita después de una caída de uno de los sistemas.

Estas acciones deber ser independientes de la calidad de la red subyacente que se utilice, parámetro denonimado como **Calidad del Servicio (QOS : Quality of Service).** Debe asegurar que la calidad de la transmisión de datos extremo a extremo demandada por la capa de sesión sea posible, a pesar de la naturaleza de la red inferior (potencialmente constituída por múltiples subredes con diferentes grados de calidad).

El tipo de servicio que ofrece el nivel de red disponible (normalmente viene impuesto, como ocurre en el caso de utilizar una conexión X.25) no tiene por qué coincidir con el tipo de servicio que queremos prestar en el nivel de transporte. Es posible, que tengamos una red que nos ofrezca un servicio no excesivamente fiable de envío de datagramas (no orientado a conexión) y, sin embargo, queramos disponer de unos servicios de transporte orientados a conexión. Es misión de este nivel, por tanto, salvar la distancia entre los servicios de red y los servicios de transporte que se quieren ofrecer. De esta forma será posible desarrollar en los niveles superiores software independiente de la tecnología de la red.

Como se observa en la figura 1.1, el sistema intermedio tiene dos diferentes "pilas" de entidades de las tres capas inferiores, donde cada una refleja la naturaleza de las subredes que representan. Con ello, la realización de una transmisión de datos extremo a extremo es manejada por la capa de red; cualquier encaminamiento que se necesite a través de los sistemas intermedios es responsabilidad de la cada de red.

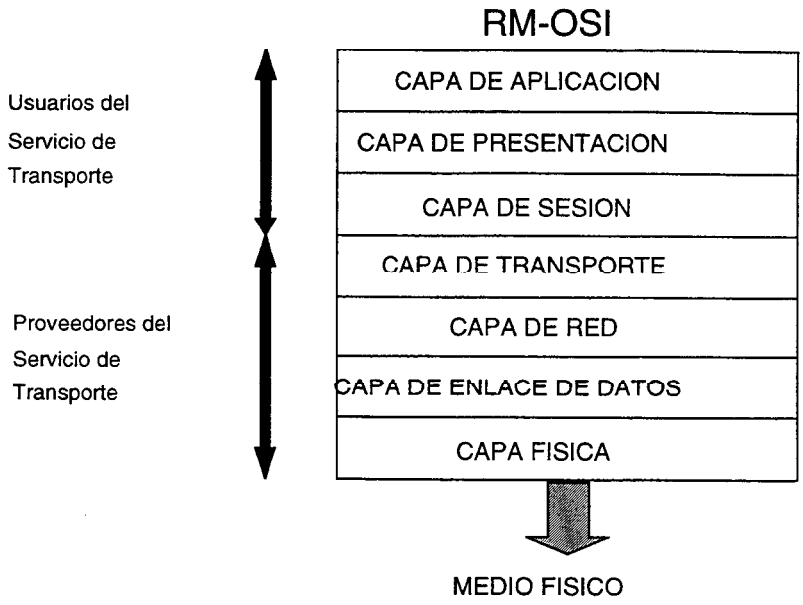


Fig. 1.2 Ubicación de capa de transporte

Como la mayor parte de los protocolos orientados a conexión, la conexión de transporte debe distinguirse por proveer seguridad, secuenciado, entrega mediante control de flujo y mantenimiento de información de estado sobre los datos transferidos.

La información de estado incluye el actual estado de los paquetes salientes, el número de secuencia para la siguiente transmisión, el siguiente número de secuencia esperado en recepción o entrante y la asignación de actual control de flujo.

Como en cada una de las capas del Modelo de Referencia se especifican los conceptos de:

- **Servicio de Transporte** provisto a los usuarios y solicitado mediante las correspondientes primitivas y parámetros a través de los puntos de acceso al servicio de transporte (TSAP). (ISO8072).

- **Protocolo de Transporte** y sus diferentes unidades de datos de protocolo. (ISO8073).

Un usuario del servicio de transporte (**TS_user**) se comunica con la entidad de transporte subyacente (o proveedor de servicio) a través de un **punto de acceso al servicio de transporte (TSAP)** empleando un conjunto de primitivas de servicio de usuario. El TSAP es el que esta asociado a la entidad iniciadora. Mediante las unidades de datos del servicio de transporte (**TSDU**) denominadas también mensajes se realizan las acciones encaminadas a generar una o varias TPDUs. Las primitivas del servicio son la causa, o el resultado, del intercambio de **unidades de datos de protocolo de transporte (TPDU)** entre las dos entidades de transporte correspondientes (pares) que intervienen en una **conexión de transporte (TC).** Estas TPDUs a su vez resultarán separadas en una o varias NPDUs en la capa de red. Las TPDUs que resultan se intercambian mediante los servicios proporcionados por la capa de red subyacente, a

través de un **punto de acceso al servicio de red (NSAP)** asociado. El conjunto, las direcciones de TSAP y NSAP ayudan a identificar de manera única la entidad de aplicación (y por tanto el AP (Proceso de Aplicación) vinculado) que participa en la conexión. Se requiere por tanto atribuir números a cada conexión e incluir dicho número en cada TPDU.

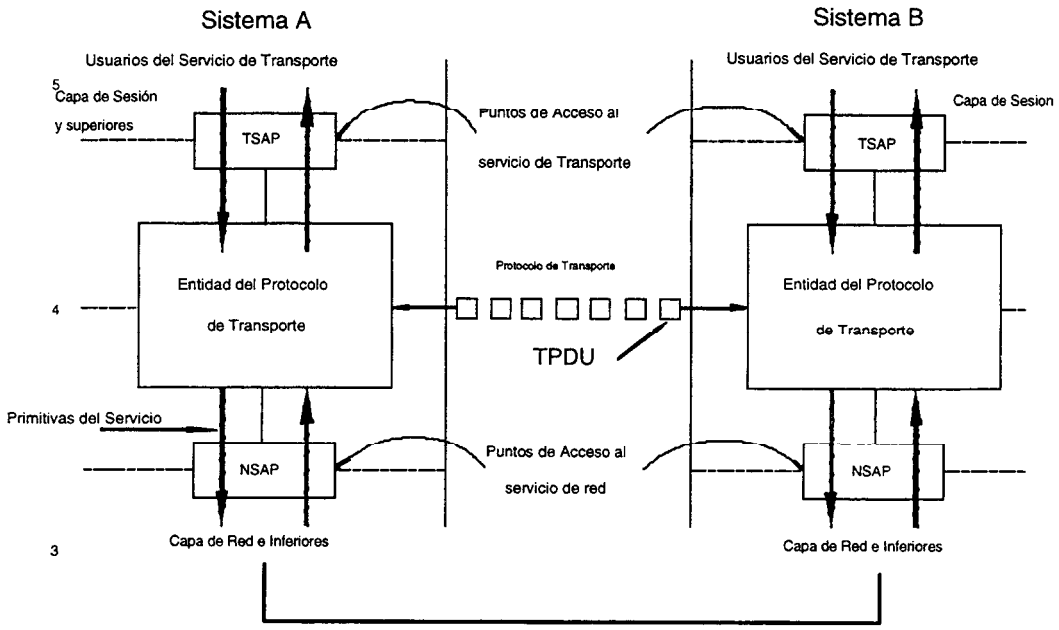


Fig. 1.3 Conceptos de capa de transporte

1.2 Servicio de Transporte.-

Aunque el servicio de red se parece al servicio de transporte hay una diferencia importante que es que el servicio de red esta diseñado para modelar el servicio ofrecido por las redes reales (p.e. X.25) defectos y todo. Estas redes pueden perder paquetes y, en forma espontánea, emitir N_RESET, debido a problemas internos de la red. De este modo, el servicio de red proporciona una manera para que sus usuarios traten con los asentimientos y N_RESET.

El servicio de transporte por el contrario, no hace ninguna indicación de los asentimientos o N_RESET. Desde el punto de vista del usuario de transporte, el servicio está libre de errores. Como es obvio, las redes reales no están libres de errores, pero ésto es precisamente el propósito de la capa de transporte, el de proporcionar un servicio fiable por encima de una red insegura, como ya se comentó. Los asentimientos y los N_RESET que vienen del servicio de red son interceptados por las entidades de transporte, y los errores se recuperan por medio del protocolo de transporte.

La entidad de transporte forma parte del Sistema Operativo o se encuentra en una tarjeta especial de hardware o en un chip, es la que utiliza el servicio de red. Muy pocos usuarios se encargan de escribir sus propias entidades de transporte y, por consiguiente, muy pocos usuarios o programas llegan a ver el servicio de red desnudo. Los programas que utilizan la red interactuan con las primitivas de transporte.

Los servicios provistos por la capa de transporte se dividen en dos tipos :

- ***orientados a conexión (connection-oriented)***
- ***no orientados a conexión (connection-less)***

Esta diferencia atiende a la forma en la cual se realiza la comunicación. Así mientras en el primer caso se cuentan con tres etapas:

- **1ª) Establecer conexión de transporte (TC).**
- **2ª) Transferencia de unidades de datos.**
- **3ª) Liberación de la conexión de transporte.**

en el caso sin conexión solo existe una etapa:

- **Transferencia de unidades de datos.**

Mientras en el caso orientado a conexión, se requiere gestionar conexiones lógicas de un usuario del servicio de transporte (TS-user) mediante su establecimiento, mantenimiento y desconexión, en el caso no orientado a conexión no se requieren gestionar conexiones lógicas pues no existen. Esta especificación en un entorno OSI es poco utilizada por su baja fiabilidad y eficiencia en la comunicación.

Para especificar cada una de las etapas se especifican una serie de primitivas para especificar cada acción, así como cuales de las primitivas básicas intervienen.

1.2.1 Orientado a conexión:

1.2.1.1 Fase de Establecimiento de Conexión.-

Para establecer una conexión de transporte, la capa de sesión especifica el **elemento de servicio** mediante la primitiva **T_CONNECT** donde **"connect"** representa establecer conexión y **"T"** indica de capa correspondiente. Para esta fase se requieren todas las 4 primitivas básicas pues es un **servicio confirmado**, o sea:

- **request**
- **indication**
- **response**
- **confirm**

de tal forma tendremos :

- **T_CONNECT.request** => petición de conexión de transporte por parte de TS_user (capa de sesión en el entorno OSI).
- **T_CONNECT.indication** => indicación de conexión de transporte a TS-user remoto.
- **T_CONNECT.response** => respuesta a conexión de transporte de TS-user remoto originador.
- **T_CONNECT.confirm** => confirmación de conexión de transporte desde sistema remoto.

que se puede ser como un diagrama lógico entre capas en la figura 1.4:

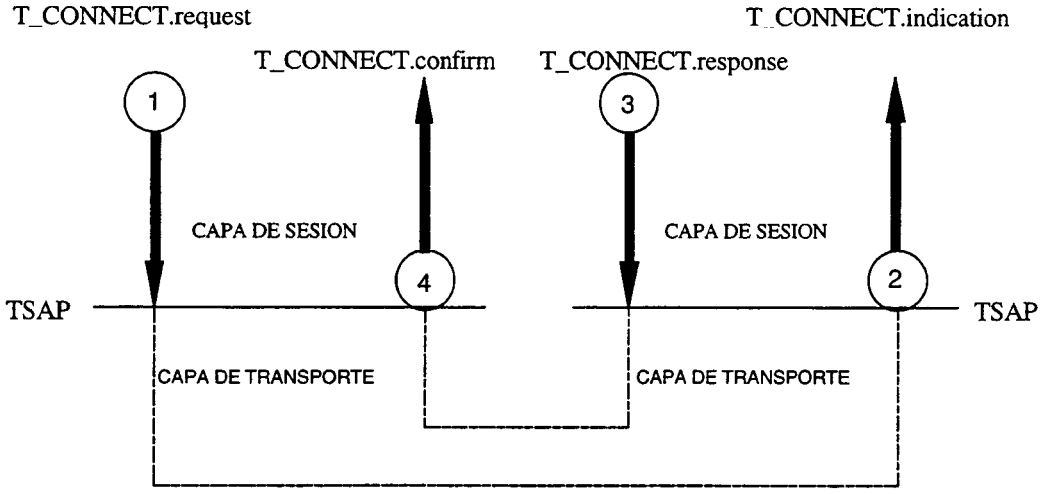


Fig. 1.4 Fase de conexión de transporte

así como mediante un diagrama de secuencia temporal tenemos (ver figura 1.5):

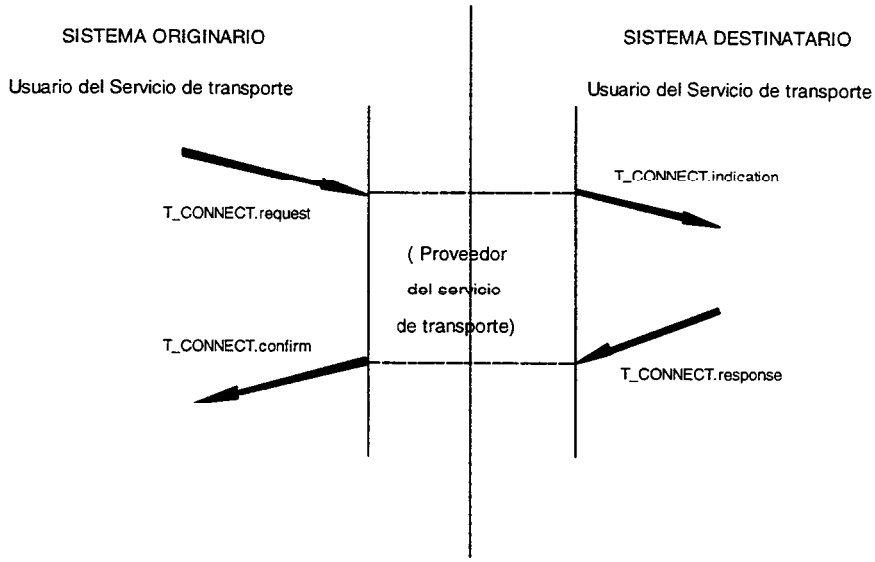


Fig. 1.5.- Fase de Establecimiento de conexión

Los parámetros que incluyen cada una de las anteriores primitivas son :

PRIMITIVA	PARAMETRO
T_CONNECT.request .indication	Dirección que llama Dirección llamada Opción de datos urgentes Calidad de servicio Datos de TS_user
T_CONNECT.response .confirm	Dirección que responde Calidad de servicio Opción de datos urgentes Datos de TS_user

Donde :

Dirección que llama : es una ***concatenación de las direcciones TSAP y NSAP*** asociada a la entidad de aplicación que establece la conexión.

Dirección llamada : es una concatenación de las direcciones TSAP y NSAP asociada a la entidad de aplicación que recibe la llamada.
(Nota : Algunos autores indican solo TSAPs no NSAPs (OSI E. John Henshal Wiley, OSI Implementation Gerald D. Cole Wiley).

Dirección que responde : Generalmente este parámetro debería ser el mismo que la dirección llamada. No obstante, el protocolo OSI es lo suficiente flexible para permitir una dirección de respuesta diferente a la dirección llamada para usar en

posibles direcciones genéricas en el futuro. Ante esta respuesta, para el resto de fases se deberá usar esta nueva dirección como referencia del destino.

Opción de datos urgentes : permite negociar si esta disponible o si se puede seleccionar el uso en la fase de transferencia de datos de la opción de enviar **datos acelerados o urgentes,** también llamados **fuera de banda.** Esta opción es un servicio opcional. Si en la petición de conexión se solicitó la opción de datos acelerados, la respuesta puede ser **seleccionado** o **no seleccionado**. La respuesta no puede solicitar la opción de datos acelerados si la petición no lo solicitó.

Calidad de Servicio : representa una lista de características que se espera tenga la conexión de transporte, como throughput (volumen de transmisión), retardo, errores permitidos, fallos, etc.. relacionados con la calidad de la red utilizada. "Ver apartado 3.5". La calidad del servicio indicado en la respuesta no puede ser mayor que la de la petición de conexión.

Datos de TS_user : Corresponde con los datos del usuario del servicio de transporte. En esta fase solo puede ocupar un máximo de 32 bytes. En esta fase habilita a una cantidad limitada de datos de usuario que de manera transparente serán pasados entre los usuarios del TS pudiendo con ello cualificar el servicio.

1.2.1.2. Fase de Transferencia de Datos

Para transferir datos de usuario del servicio de transporte, o sea la capa de sesión, se especifica el **elemento de servicio** mediante la primitiva **T_DATA,** donde **"data"** representa transferencia de datos y **"T"** indica la capa correspondiente- Permite utilizar un canal normal. Esta fase es un **servicio no confirmado**, por lo que solo se requieren las siguientes primitivas básicas:

- **request**
- **indication**

De igual forma se especifica la primitiva **T_EXPEDITED_DATA** para utilizar el **canal urgente** para el envío de datos urgentes o acelerados, si la red lo soporta, y en el caso de que ambos sistemas hayan acordado utilizar esta opción en la fase de establecimiento de conexión.

Combinando las primitivas tenemos:

- **T_DATA.request =>** Petición de envío de datos de usuario de TS (TS_user).
- **T_DATA.indication =>** Indicación de envío de datos de usuario de TS remoto.

Y para el caso de **transferencia de datos urgentes** :

- **T_EXPEDITED_DATA.request =>** Petición de envío de datos urgentes de usuario de TS.

- **T_EXPEDITED_DATA.indication** => Indicación de envío de datos urgentes de usuario de TS remoto.

que como un diagrama lógico entre capas se muestra en la figura 1.6:

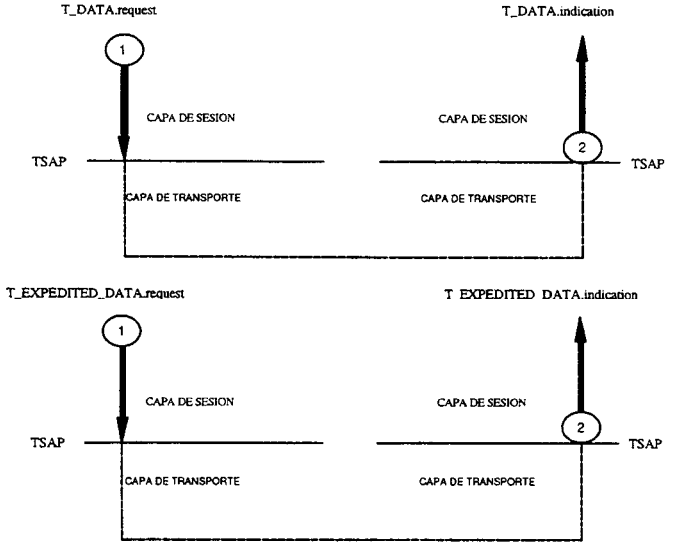


Fig. 1.6.- Fase de Transferencia de Datos (Normales y Urgentes)

mediante un diagrama de secuencia temporal tenemos (ver figura 1.7):

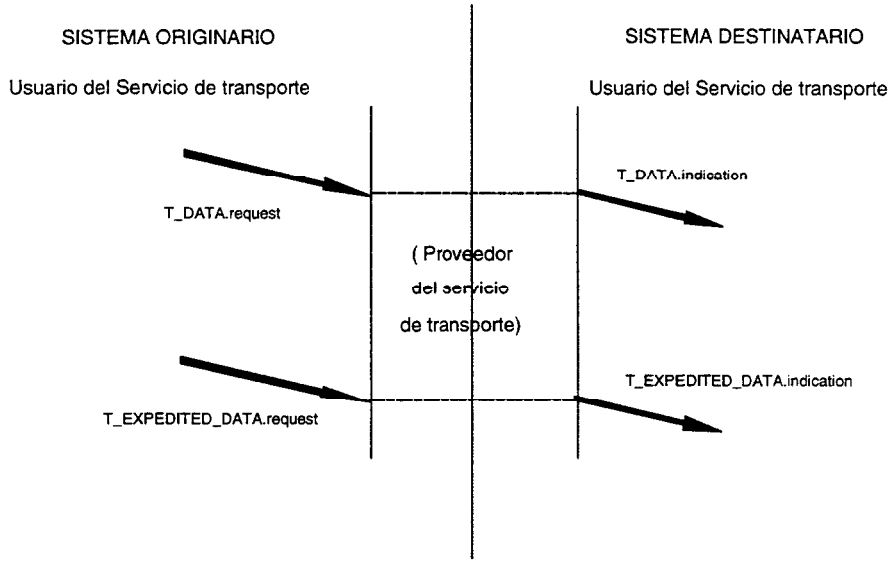


Fig. 1.7.- Diagrama temporal de transferencia de Datos

Los parámetros que incluyen cada una de las anteriores primitivas son :

PRIMITIVA	PARAMETRO

T_DATA.request .indication	Datos de TS_user
T_EXPEDITED_DATA.request .indication	Datos de TS_user

Donde :

Datos de TS_user: Son las unidades de datos del servicio de transporte intercambiadas a través de una conexión de transporte entre usuarios del servicio de transporte pares. Este campo no tiene ninguna restricción en cuanto a tamaño, salvo en el caso de T_EXPEDITED_DATA que esta **limitado a 16 bytes.**

Al igual que en el servicio de red orientado a conexión, en una situación normal de trabajo, los datos normales alcanzan su destino en el mismo orden en el que el usuario los envía. Los datos urgentes llegarán siempre en orden y antes que cualquier dato normal que se encuentre en la cola o que haya sido enviado con posterioridad.

T_EXPEDITED_DATA, normalmente, solo se utiliza para transmitir los comandos **BREAK,DEL o de interrupción**.

1.2.1.3. Fase de Liberación de Conexión.-

Para liberar una conexión de transporte ya establecida o rechazar el intento de realizar una conexión se especifica el **elemento de servicio** mediante la primitiva **T_DISCONNECT** donde **"disconnect"** representa liberación de conexión y **"T"** indica la capa correspondiente. Esta fase es un **servicio no confirmado** y abrupto (se pueden perder los datos pendientes de recepción), por lo que solo se requieren las siguientes primitivas básicas :

- **request**
- **indication**

La conexión de transporte tras el uso de estas primitivas se da por **cerrada de forma inmediata.**

Combinado con las primitivas tenemos :

- **T_DISCONNECT.request** => Petición de liberación de conexión de transporte (TC).
- **T_DISCONNECT.indication** => Indicación de liberación de conexión de transporte (TC).

que como un diagrama lógico se muestra en la figura 1.8:

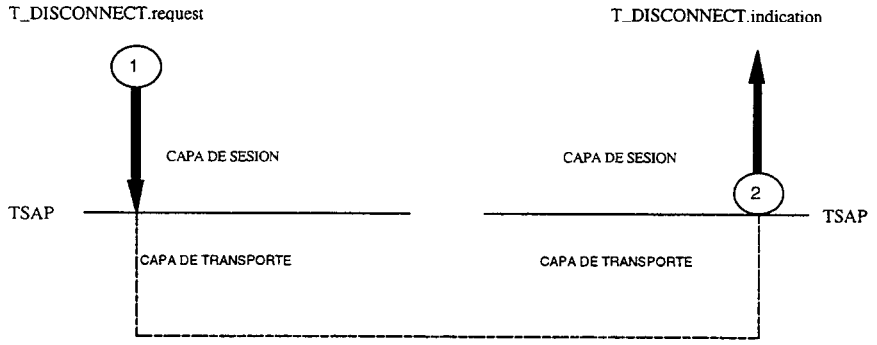


Fig. 1.8.- Fase de Liberación de Conexión

mediante un diagrama de secuencia temporal tenemos (ver figura 1.9):

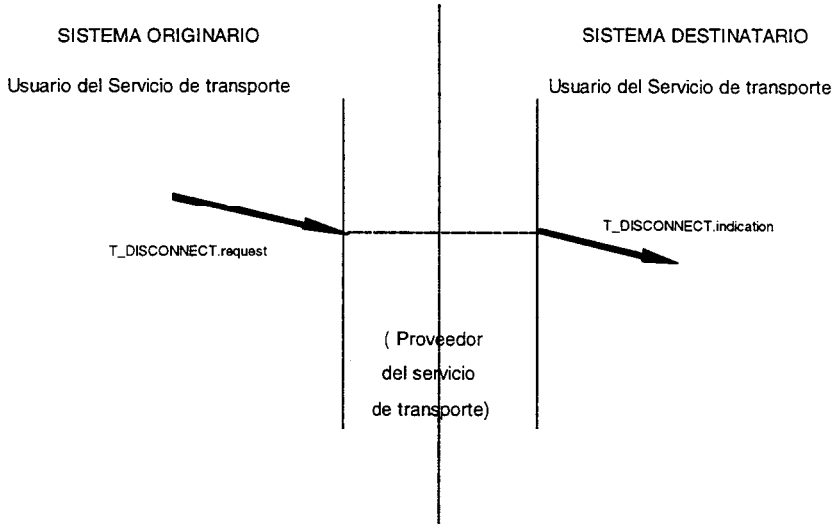


Fig. 1.9.- Diagrama temporal de Liberación de Conexión

Los parámetros que incluyen cada una de las anteriores primitivas son:

PRIMITIVA	**PARAMETRO**
T_DISCONNECT.request	Datos de TS_user
T_DISCONNECT.indication	Razón de Desconexión Datos de TS_user

Donde :

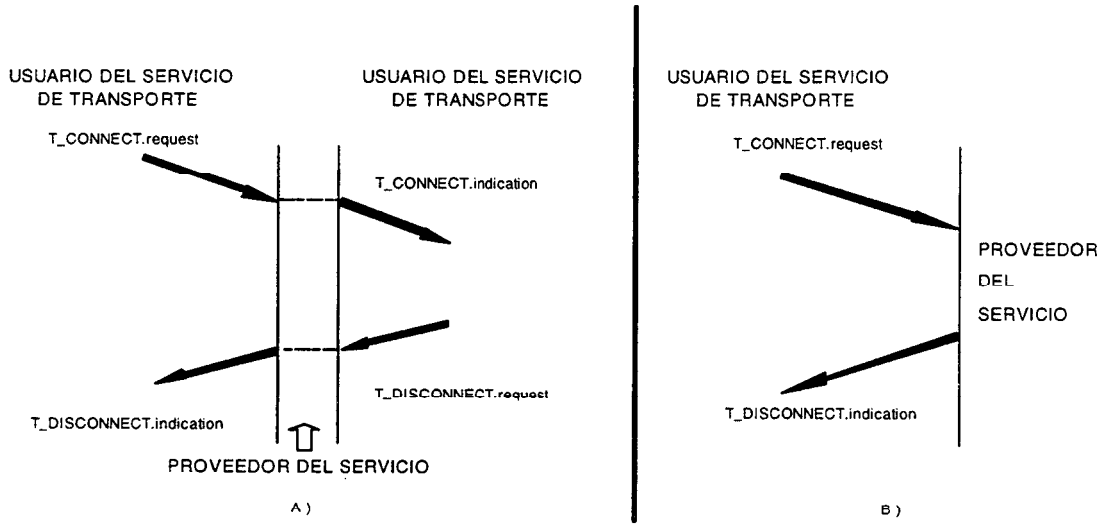
Datos de TS_user: Se corresponde con datos del usuario del servicio de transporte. En esta fase solo puede ocupar un máximo de 64 bytes. Se habilita por tanto, una cantidad limitada de datos de usuario que de manera transparente serán pasados entre los usuarios del TS.

Razón de Liberación (desconexión) : Este parámetro se utiliza para cualificar los datos de usuario o para indicar la causa de la desconexión. Posibles valores son:

- Calidad del Servicio ha caído por debajo del nivel acordado en TC.
- Congestión o fallo del proveedor del TS local o remoto. (imposibilidad de dar un servicio de la calidad solicitada).
- Razón desconocida.
- Dirección TSAP llamada no válida. (usuario remoto desconocido)
- Dirección TSAP no disponible. (usuario remoto no dispuesto a establecer TC).

Liberación/Rechazo de Conexiones

Como se puede apreciar, la desconexión no solo se produce al final de un proceso normal de transmisión tras las tres fases indicadas, sino que puede producirse por diversos motivos. En el siguiente diagrama (ver figura 1.10) se indican diferentes posibles usos de las solicitudes de desconexión para situaciones que se suelen denominar **rechazos de conexión** generados por el usuario del servicio de transporte o por el proveedor del servicio de transporte.



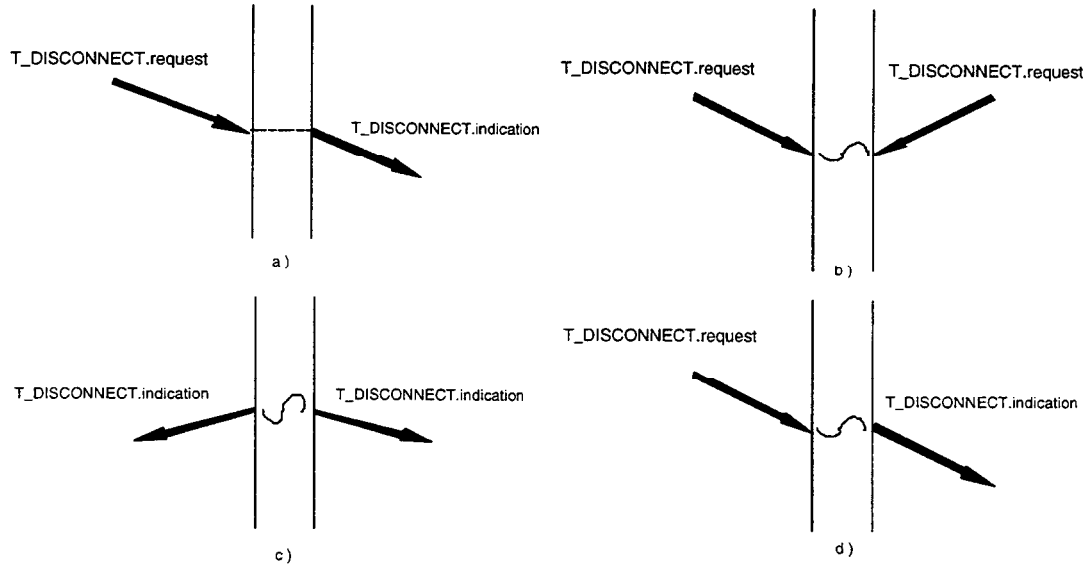
A) Petición de conexión rechazada por el usuario del servicio.

B) Petición de conexión rechazada por el proveedor del servicio.

Fig. 1.10.- Tipos de desconexión

El rechazo mostrado como b) en la figura 1.0, realizado por el proveedor del servicio de transporte puede ser por culpa del usuario de transporte (p.e. un parámetro erróneo en la primitiva T_CONNECT.request o por fallo de la entidad de transporte local (p.e. falta de espacio en las tablas internas)).

Si ambas entidades emiten T_DISCONNECT.request, la conexión se libera sin que ninguna de las dos partes obtenga una indicación al respecto. El mismo proveedor del servicio puede terminar una conexión al emitir primitivas T_DISCONNECT.indication en ambos extremos, y es un poco como la capa de red emitiendo un N_RESET.indication (ver figura 1.11).



a) Liberación de la conexión iniciada por el usuario del servicio de Transporte

b) Liberación iniciada por ambos usuarios.

c) Liberación iniciada por el proveedor del servicio de transporte.

d) Liberación iniciada por el usuario y el proveedor.

Fig. 1.11 Diferentes situaciones de rechazo

1.2.1.4 Máquina de Estados

El establecimiento, uso y liberación de conexiones de transporte para modo orientado a conexión puede ser representada en forma de máquina (autómata) de estados finitos que consta de **4** estados y **8** sucesos (eventos que provocan la transición de estados) y las salidas que serán las acciones del protocolo de transporte, como se muestra en la figura 1.12.

La máquina comienza en el estado **inactivo.** En este estado no hay conexión, ni se intenta abrir una. Cuando una petición T_CONNECT.request es enviada, se pasa al estado **pendiente de conexión saliente** (lado izquierdo) Aún no ha llegado respuesta del otro extremo. Si se recibe una confirmación a T_CONNECT.request (T_CONNECT.confirm) se pasa al estado de **transferencia de datos**. En este estado se ha establecido la conexión de transporte y pueden producirse un intercambio de datos, o puede cerrarse. Las consiguientes primitivas T_DATA.request y T_DATA.indication mantendrán la máquina en el estado actual. Liberaciones de conexión y retorno al estado inactivo pueden ser generados por peticiones e indicaciones T_DISCONNECT.request y T_DISCONNECT.iindication.

Si por el contrario se recibe una indicación T_CONNECT.indication produce una transición al estado **pendiente de conexión entrante** (lado derecho). Aún no se ha aceptado ni rechazado. Si recibe una respuesta del usuario del servicio (T_CONNECT.response) produce la transición al estado de **transferencia de datos.**

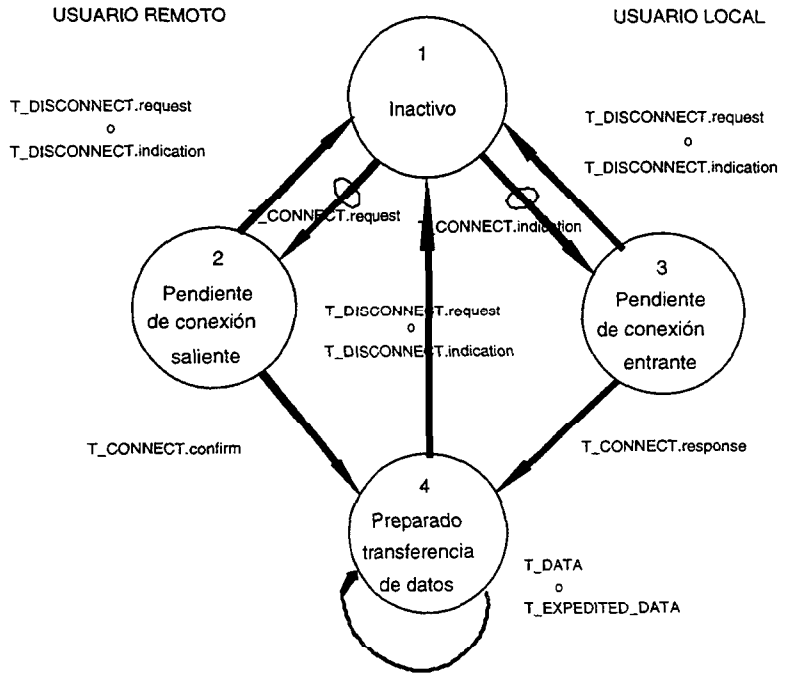


Fig. 1.12. Diagrama de transición de estados de servicio de Transporte

1.2.2 No Orientado a Conexión

En este caso, el servicio de transporte solo cuenta con la fase de transferencia de datos por lo cual solo dispone del elemento de servicio representado mediante una primitiva y un servicio **no confirmado.**

Con este servicio, sin embargo, no hay ninguna garantía de que la transferencia se haya realizado con éxito y se deja a las capas superiores orientadas a la aplicación la tarea de recuperarse de tales situaciones.

1.2.2.1 Fase de Transferencia de Datos

Para transferir datos de usuario del servicio de transporte, o sea la capa de sesión especifica el **elemento de servicio** mediante la primitiva **T_UNITDATA** donde "**unitdata**" representa transferencia de datos sin conexión y "**T**" indica de capa correspondiente. Esta fase es un **servicio no confirmado.** por lo que solo se requieren las siguientes primitivas básicas :

- **request**
- **indication**

Combinando las primitivas tenemos :

- **T_UNITDATA.request =>** Petición de envío de datos de usuario de TS (TS_user).
- **T_UNITDATA.indication =>** Indicación de envío de datos de usuario de TS remoto.

mediante un diagrama de secuencia temporal tenemos (ver figura 1.13):

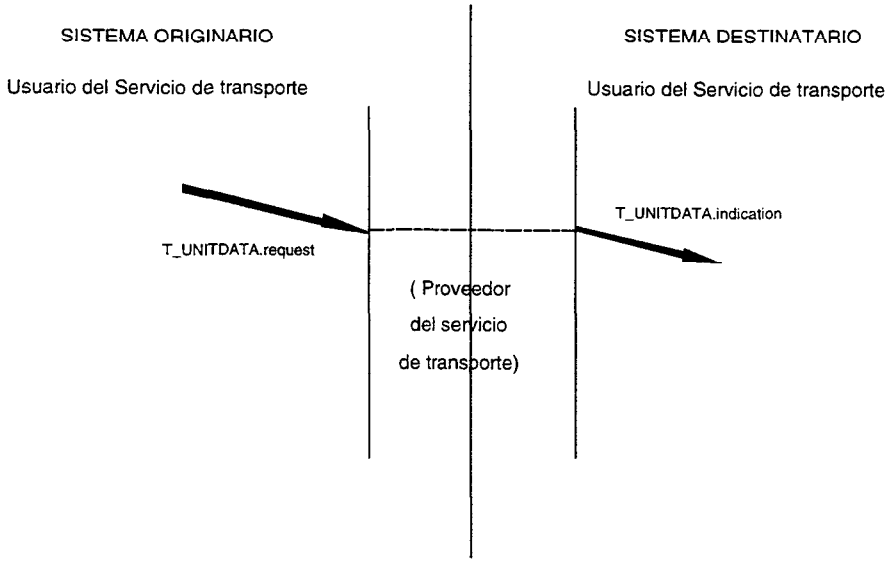


Fig. 1.13. Primitivas de servicio no orientado a conexión

Los parámetros que incluyen cada una de las anteriores primitivas son:

PRIMITIVA	**PARAMETRO**
T_UNITDATA.request .indication	Dirección que llama Dirección llamada Calidad de servicio Datos de TS_user

Donde :

Dirección que llama: es una concatenación de las direcciones TSAP y NSAP asociada a la entidad de aplicación que establece la conexión.

Dirección llamada : es una concatenación de las direcciones TSAP y NSAP asociada a la entidad de aplicación que recibe la llamada.
(Nota : Algunos autores indican solo TSAPs no NSAPs (OSI E. John Henshal Wiley, OSI Implentation Gerald D. Cole Wiley).

Calidad de Servicio : representa una lista de características que se espera tenga la transferencia de datos de transporte. Ver apartado 1.3.5.

Datos de TS_user : Son los unidades de datos del servicio de transporte intercambiadas a través de una comunicación de transporte entre usuarios del servicio de transporte pares. Este campo no tiene restricción en cuanto a tamaño.

Como resumen se ilustran a continuación todas las primitivas y sus parámetros :

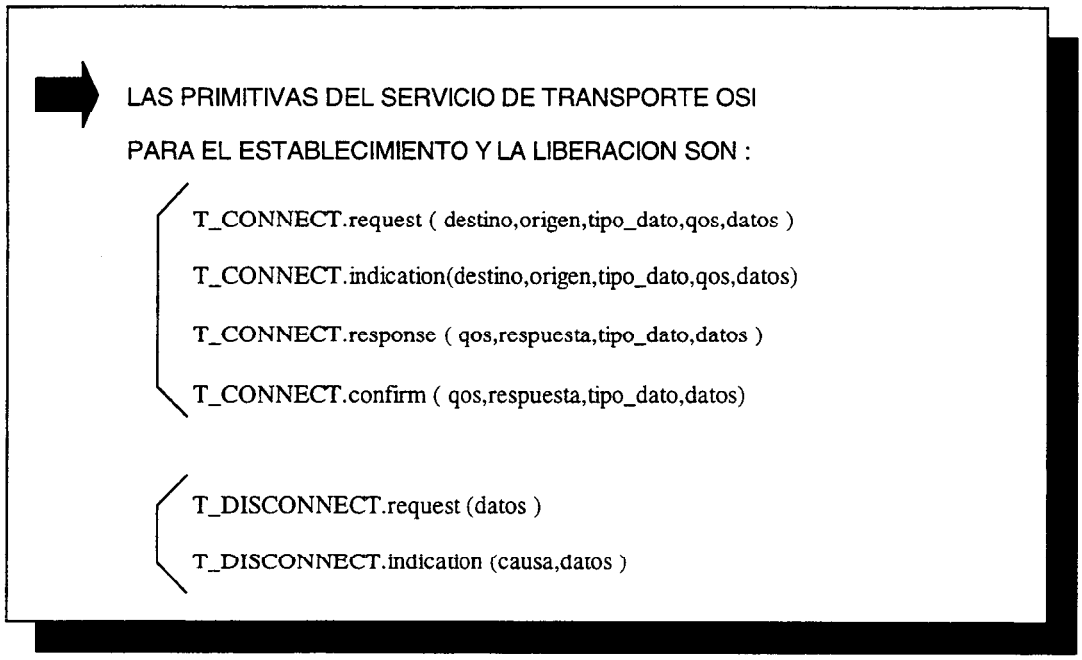


Fig. 1.14 Primitivas de Conexión y Desconexión

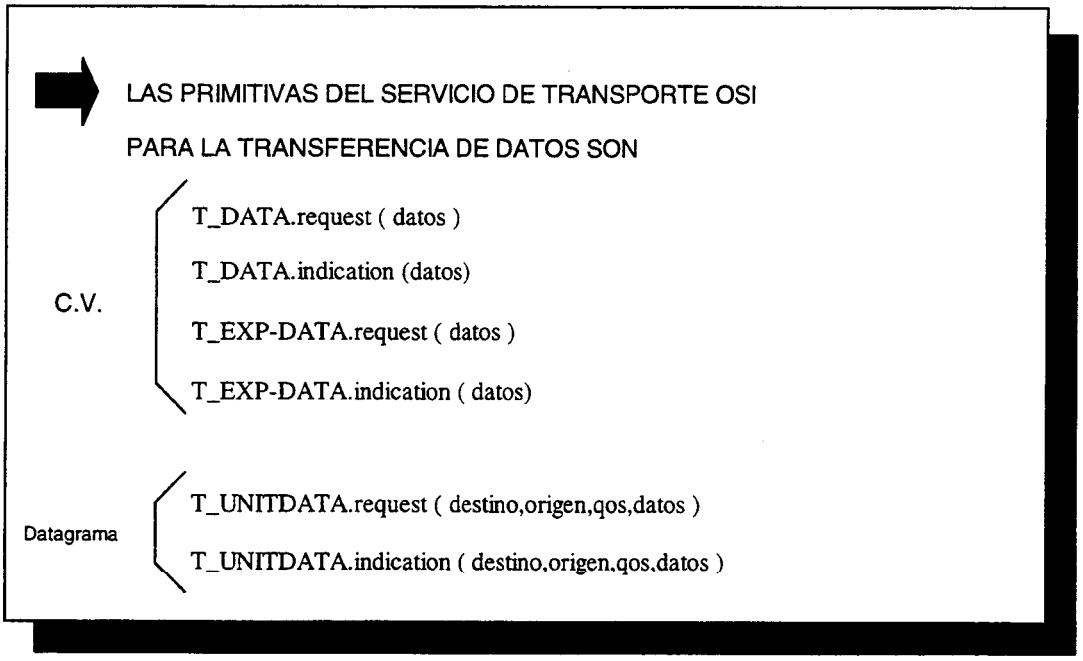


Fig. 1.15 Primitivas de Datos

Toda la secuencia de primitivas del servicio en un diagrama temporal se puede ver en la figura 1.16:

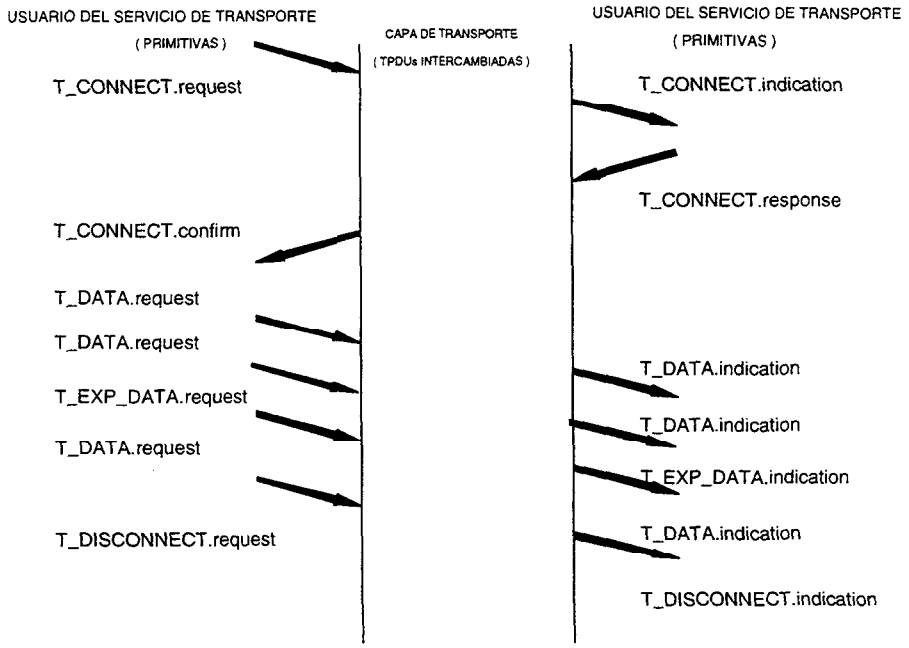


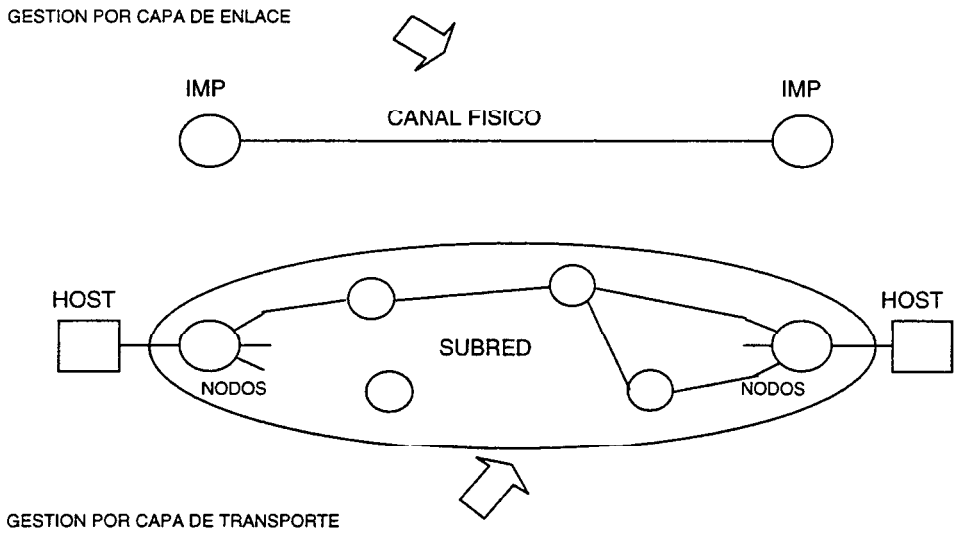
Fig. 1.16.- Secuencia de Primitivas en modo orientado a conexión

1.3. Protocolo de Transporte.-

El protocolo de transporte orientado a conexión se define el documento de especificación de protocolo ISO 8073.

El servicio de transporte se realiza por medio de un protocolo de transporte que se utiliza entre las dos entidades de transporte, y es similar al de la capa de enlace de datos, pues los dos tienen que ver con el control de errores, secuenciamiento y control de flujo, entre otros asuntos.

Las diferencias entre ambos son: En el caso de la capa de enlace, hay dos IMP o conmutadores que se comunican directamente a través del canal físico, en tanto que en la capa de transporte, este canal se sustituye por la subred completa.



En este caso, no es necesario que un IMP especifique con que IMP desea comunicarse -cada línea de salida especifica unívocamente un IMP particular-. En la capa de transporte se necesita un direccionamiento explícito de los destinos. Por otro lado el proceso para llegar a establecer una conexión en el cable es simple: el otro extremo siempre esta ahí (a menos que haya fallado, en cuyo caso, ya no estará ahí). De cualquier manera, no hay mucho que hacer al respecto. Otra diferencia muy molesta entre ambas capas es la existencia potencial de capacidad de almacenamiento en la subred. Cuando un IMP transmite una trama, esta puede llegar a perderse, pero no puede saltar de un lugar a otro durante algún tiempo, ni tampoco esconderse en un extremo alejado del mundo y después súbitamente emerger en un momento inoportuno, 30 segundos más tarde. Si en el interior de la subred se utilizan datagramas y encaminamientos adaptables, existe una probabilidad poco despreciable de que un paquete pueda almacenarse durante varios segundos y posteriormente entregarse. Las consecuencias, de esta facultad que tiene las subred de almacenar paquetes, puede ser catastróficas y, por lo tanto, requerir el uso de protocolos especiales.

Para especificar un protocolo de transporte es necesario tener conocimiento de la calidad del servicio soportado por la capa de red inferior y subredes asociadas. Para ello existen dos factores principales. El primero es la limitación física de una subred, o de sus componentes; por ejemplo, si un conmutador de paquetes en una subred particular X.25 tiene una restringida capacidad de conmutación de **n** paquetes por segundo, entonces el **throughput** de datos de esa subred esta limitado por el valor **n**. El segundo aspecto son los **errores**; el nivel de errores que una subred tiene, limita el nivel de fiabilidad de servicio que puede ser ofrecido por ella.

1.3.1 Errores/Tipos de Redes.-

Los errores originados en una subred y consecuentemente los observados por la capa de transporte, que deteminarán en gran medida la calidad de servicio de dicha subred, pueden ser :

- **señalizados ó**
- **residuales**

Un ***error señalizado*** es un error detectado por la capa de red, pero ante él ninguna acción es tomada dentro de esa capa para su recuperación. Este suceso solamente se notifica a la capa de transporte para que esta realice las acciones oportunas. Dos ejemplos son **desconexión de red** (la conexión de red se ha perdido) y **reset de red** (la conexión de red se resetea por un estado desconocido, posiblemente con pérdida de datos en tránsito, pero la conexión permanece disponible para su uso).

Los ***errores residuales*** son el resto de los no incluídos en los señalizados. Son aquellos que no han sido detectados por la capa de red. Algunos ejemplos son pérdida, corrupción, duplicación y entrega fuera de secuencia de unidades de datos.

En función de la clase de servicio que nos ofrezca la red, y de la calidad del mismo, las tareas que realizan los protocolos de transporte pueden variar enormemente. Evidentemente, no será igual un protocolo para ofrecer un servicio de transporte orientado a conexión utilizando una red fiable y orientada a conexión, que un protocolo que ofrezca el mismo servicio sobre una red no fiable de intercambio de datagramas.

Atendiendo a un análisis de los tipos de errores que provocan las diferentes subredes, se realiza una clasificación en tres tipos o categorías de redes :

TIPO A : Aceptable tasa de errores señalizados y residuales.

La fracción de paquetes perdidos, duplicados o dañados, viene a ser despreciable. Los N_RESET son tan raros que pueden ser ignorados. Sin errores ni N_RESET. Las redes públicas tipo WAN que ofrecen un servicio tipo A son tan escasas como los "dientes de una gallina", pero las redes tipo LAN se acercan bastante. Se usan protocolos directos-

TIPO B : Aceptable tasa de errores residuales pero inaceptable tasa de errores señalizados.

Más comunes WAN. Los paquetes individuales rara vez, si es que alguna, llegan a perderse (porque los protocolos de la capa de red y enlace se recuperan de estas pérdidas en forma transparente), sin embargo de cuando en cuando la capa de red emite N_RESET, como consecuencia de una congestión interna, por problemas de hardware, o bien, por irregularidades del software. Depende, por consiguiente, del protocolo de transporte recolectar los pedazos, establecer una nueva conexión de red, resincronizarse y continuar, de tal manera que el N_RESET quede perfectamente oculto para el usuario de transporte. La mayoría de las redes X.25 son tipo B. (Entrega perfectamente de paquetes pero con N_RESET).

TIPO C : Inaceptable tasa de errores residuales.

En estas redes no se puede confiar. Incluyen las WAN que ofrecen un servicio puro sin conexión (datagrama), las redes radio-paquetes y varias interredes. Son muy inseguras pues provocan pérdidas de paquetes, duplicidades y probablemente N_RESETs.

O sea podemos ver esta subdivisión de la siguiente forma :

Tipo de Red	Características
A	Aceptable. Redes Seguras. Bajo número de errores. Bajo rechazos y desconexiones.
B	Ocurren errores pero son aceptables. Elevadas desconexiones y rechazos de conexión.
C	Tasa de errores inaceptable.

A continuación se muestra otra gráfica función de la calidad de la red (ver figura 1.17):

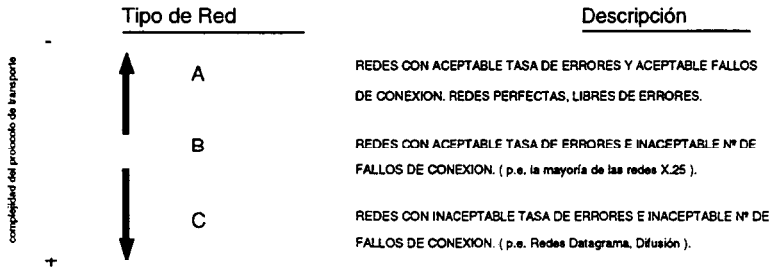


Fig. 1.17. Tipos de Redes

Para clasificar que tipo de red se esta utilizando, si esta está formada por varias subredes, se asignará aquella que tenga la peor calidad de servicio ofrecido. Como parte de la fase de establecimiento de la conexión las entidades de transporte pares deben establecer el nivel de servicio de red que deberá ser provisto para una calidad de servicio especificada para esta conexión. Esto incluye la selección del conjunto de procedimientos que serán utilizados durante la conexión que se hará de manera paralela con la **negociación de la calidad de servicio**.

1.3.2 Clases de Protocolo de Transporte.-

Hay un conjunto de cinco niveles o clases de protocolos de transporte disponibles para la capa de transporte. Cada clase está, de una forma u otra, relacionada con los tres tipos de redes. Las entidades de transporte, durante la fase de establecimiento de conexión de transporte (TC), negocian el procedimiento que permite concretar que clase de protocolo de transporte será utilizada sobre la red para esta conexión particular, y con ello definir los procedimientos a utilizar en la entidad de transporte.

Dichas clases son:

Clase 0:

Clase Simple. Provee la conexión de transporte más básica y esta diseñada para usar con redes Tipo A por la calidad de servicio ofrecida por estas redes. Dado que este tipo de servicio de red provee transmisión de datos segura, solo se requiere el nivel básico de actividad de transporte. Concretamente, el TP0 no hace otra cosa que **segmentación y reensamblado**. No realiza secuenciamiento ni control de flujo, basándose en que la capa de red subyacente lo haga todo correctamente. Existen conexiones de transporte que se apoyan directamente sobre conexiones de red. En caso de fallo en la red, el protocolo no tiene ningún mecanismo de recuperación, por lo que la conexión de transporte se cierra inmediatamente. TP0 no ofrece sercicio de transporte urgente.

Clase 1:

Clase con recuperación básica de error. Provee, con un mínimo sobretrafico (overhead), una conexión de transporte básica para ser usada con redes Tipo B. Maneja errores señalizados tales como desconexión de red (N_DISCONNECT) y reset de red (N_RESET) sin que le efecte al usuario del servicio de transporte. Las unidades de datos se numeran y, en caso de que se produzca un cierre no esperado de la conexión de red, o bien un N_RESET, se abre o se restablece una conexión de red y se produce un diálogo entre las entidades de transporte para ponerse de acuerdo sobre qué unidades de datos se han recibido correctamente y cuales no. Estas últimas son retransmitidas. El proceso de restablecimiento de una conexión es completamente transparente a los usuarios, aunque, debido al tiempo que toma, se notará un descenso en las prestaciones de la conexión. No

proporciona control de error ni control de flujo adicicional al que la misma capa de red proporciona.

Clase 2

Clase con Multiplexación. Es una clase 0 pero con mecanismos adicionales para soportar la multiplexación de conexiones de transporte dentro de una simple conexión de red, siempre que sea posible. Esto exige en ocasiones la incorporación de mecanismos de **control de flujo** en cada conexión de transporte, lo que precisa a su vez que las unidades de datos vayan numeradas. Esta clase resulta muy útil cuando hay varias conexiones de transporte abiertas, cada una con relativamente poco tráfico, y el operador aplica una tarifa muy elevada por tiempo de conexión, por cada conexión de red abierta. Por ejemplo terminal de reservas aérea, donde existen muchos terminales que acceden a un mismo equipo central.

Clase 2 = Clase 0 + Multiplexación

Clase 3:

Clase con recuperación básica de errores y multiplexación. Es una clase 1 con mecanismos adicionales de multiplexación. Utiliza control de flujo.

Clase 3 = Clase 1 + Multiplexación

Clase 4:

Clase con Detección y Recuperación de error. Provee toda la capacidad de la clase 3 junto con mecanismos requeridos para detectar y recuperarse de errores no señalizados por el proveedor del servicio de red. Esta clase genera un incremento de throughput para añadir una adicional resistencia a fallos producidos por el proveedor del servicio de red. Esta diseñada para usarse con redes tipo C. Su diseño y uso parte de usar la Ley de Murphy como base "Si algo puede fallar, seguro que fallará". Es una clase completamente diferente a las anteriores. Incorpora todo tipo de mecanismos que permitan ofrecer un servicio correcto. TP4 puede operar sobre redes fiables, pero supone una sobrecarga innecesaria de proceso en los sistemas finales, al repetirse funciones que ya realiza el nivel de red en los sistemas intermedios. Deberá ser capaz de manejar paquetes que se hayan perdido, duplicados o dañados, N_RESET y cualquier otra información que la red pueda enviarle.

Clase 4 = Clase 3 + Detección de errores

CLASE	CARACTERISTICA
0	Simple
1	Multiplexación
2	Recuperación Básica de Errores
3	Recuperación Básica y Multiplexación
4	Detección y Recuperación B. de errores. Multiplexación

Combinado ambas clasificaciones tenemos :

CLASE	TIPO
0	A
1	B
2	A
3	B
4	C

Fig 1.18. Combinación de Clases y Tipos

Por ejemplo para aplicaciones como el correo electrónico puede ser válida una clase 0 pero para una aplicación de transacciones bancarias se parte de clase 4, aunque la red no sea tipo C. La negociación de la clase a utilizar se realiza de forma que, la entidad iniciadora propone una clase ***preferida*** junto con, opcionalmente, una clase ***alternativa***. La clase preferida será aceptada o modificada a otra clase, numericamente más baja, por la entidad que responde. Si ninguna de las opciones ofrecidas es aceptable, la conexión se rechazará. Las diferentes combinaciones ante la negociación de la clase de protocolo a utilizar se muestran en la tabla siguiente:

CLASE PREFERIDA	CLASE ALTERNATIVA PROPUESTA					
	0	1	2	3	4	Ninguna
0	-	-	-	-	-	0
1	1, 0	1, 0	-	-	-	1, 0
2	2, 0	-	2	-	-	2
3	3,2,0	3,2,1,0	3, 2	3, 2	-	3, 2
4	4,2,0	4,2,1,0	4, 2	4,3,2	4, 2	4, 2

La entidad que responde puede escoger aceptar cualquier clase indicada en la apropiada intersección en la tabla.

1.3.3 Unidades de Datos del Protocolo de Transporte.-

Como se sabe, las entidades de transporte se comunican mediante el protocolo de transporte usando unidades de datos de protocolo, que se generan a partir de las primitivas correspondientes. O sea que al recibir una primitiva de servicio entrante válida (del TS_user o del proveedor de la red), la entidad de protocolo de transporte

genera una **TPDU** asociada. Por lo general, y suponiendo que la primitiva entrante provenga de un TS_user, la TPDU comprende los datos de usuario asociados a la primitiva e información de control de protocolo adicional que agrega la entidad de transporte. La TPDU generada se transfiere a la entidad de transporte correspondiente mediante los servicios que provee la capa de red subyacente (ver caso general en figura 1.19).

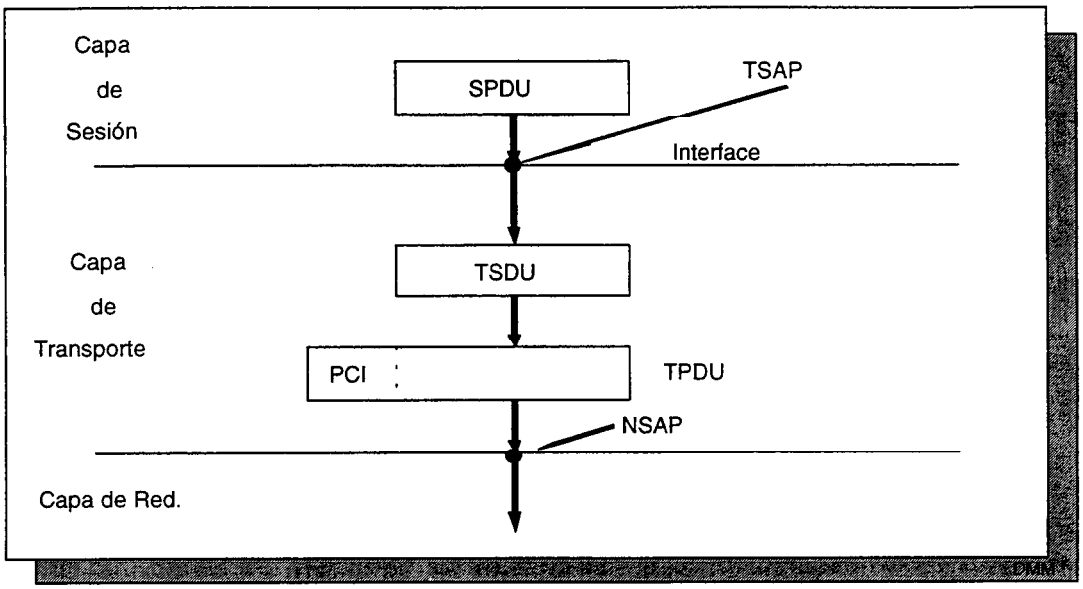
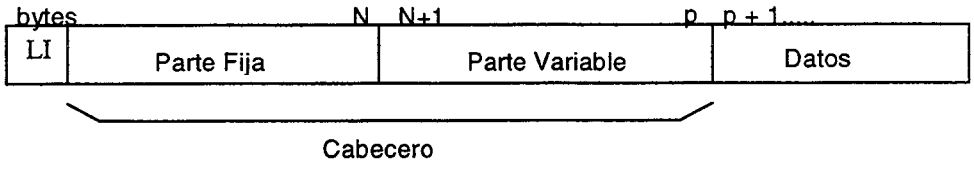


Fig. 1.19. Transformación de TSDUs en TPDUs

El protocolo de transporte define 10 TPDUs cuyo formato sigue una estructura basada en cuatro campos (ver figura 1.20) :

- **Campo longitud de Cabecero (LI).**
- **Cabecero :**
  - **Parte Fija**
  - **Parte Variable**
- **Datos de usuario**



FORMATO GENERAL DE LAS TPDU's

Fig. 20 Formato general de TPDUs

A partir de ello hagamos un análisis atendiendo a las diferentes fases de la comunicación. Estas 10 TPDUs estaá codificadas con 4 bits y representadas con dos letras identificando la acción que realizan. Cada una de ellas se analizan en los siguientes apartados. (ver siguiente figura 1.21):

TABLA DE TPDU Y CODIGOS ASOCIADOS

		Cantidad de datos
CR,petición de conexión	1110	<= 32
CC,confirmación de conexión	1101	<= 32
DR,petición de desconexión	1000	<= 64
DC,confirmación de desconexión	1100	ninguno
DT,transferencia de datos	1111	negociado
ED, datos expeditos	0001	<= 16
AK, reconocimiento	0110	ninguno
EA, reconocimiento datos expeditos	0010	ninguno
RJ, rechazo	0101	ninguno
ER, error	0111	ninguno

Fig. 1.21. Codificación de diferentes TPDUs

1.3.3.1 Fase de Establecimiento de Conexión.-

El establecimiento de una conexión de transporte se inicia cuando un TS_user emite una primitiva T_CONNECT.request. La entidad de protocolo de transporte local responde creando una TPDU que se representa como **CR** , que aquella envía a su entidad de protocolo de transporte par en el sistema al que se llama. Al recibir esta TPDU, la entidad de protocolo de transporte par notifica al usuario designado en su propio sistema que se ha presentado esta solicitud de conexión mediante una primitiva T_CONNECT.indication.

A continuación, el usuario correspondiente (suponiendo que está preparado para aceptar la llamada) responderá con un primitiva T_CONNECT.response, respondiendo a la entidad emisora con una TPDU denominada **CC.** Por último, la entidad de protocolo iniciadora comunicará esta respuesta al usuario mediante la primitiva T_CONNECT.confirm.

En la figura 1.22 se ilustran las primitivas que intervienen y las TDPUs asociadas en cada sistema:

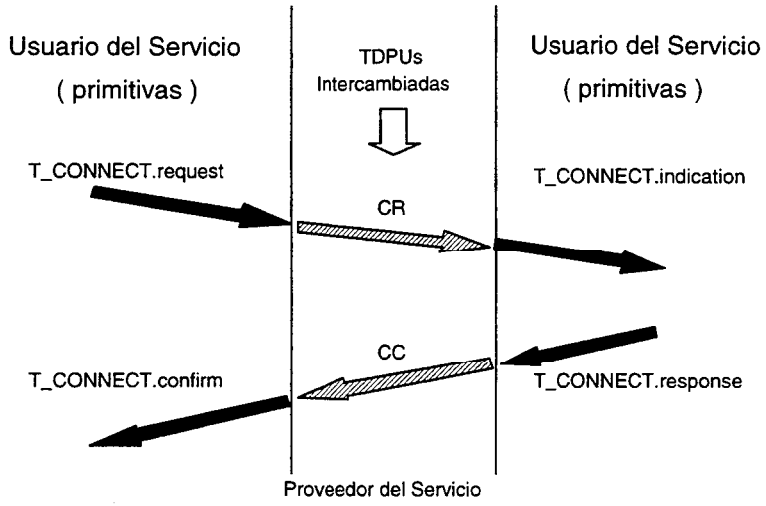


Fig. 1.22. TPDU intercambiadas

Sólo se realiza un intercambio bidireccional. Si al mismo tiempo dos usuarios intentan establecer una conexión entre sí, se establecerán dos conexiones independientes, aunque es poco probable pues en OSI la mayor parte de las aplicaciones son cliente-servidor.

El formato de la TPDU CR tiene la forma mostrada en la figura 1.23 :

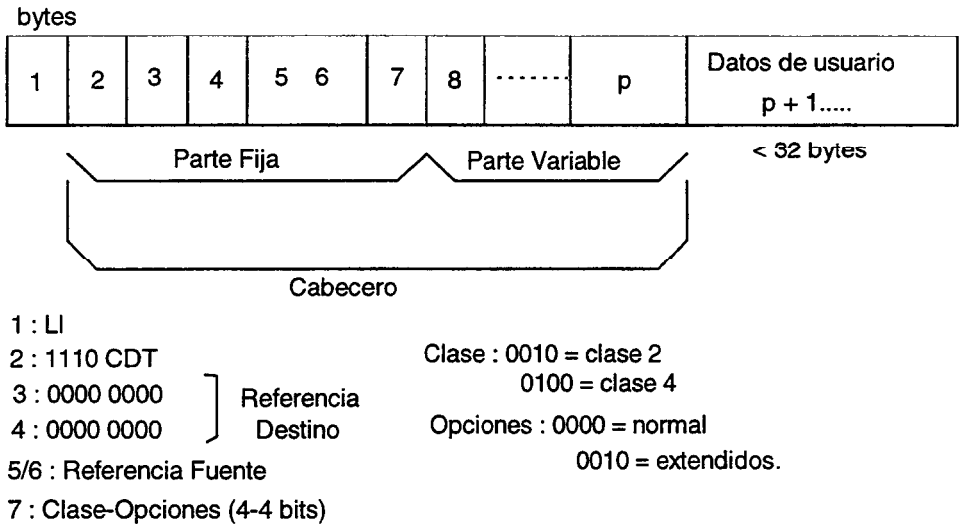


Fig. 1.23. TPDU para establecer conexión

Los parámetros contenidos en las TPDUs CR y CC comunican información relacionada con la conexión que se está estableciendo y que ambas entidades de protocolo de transporte deben conocer para poder gestionar transferencias de datos siguientes para la conexión establecida.

En el caso de TPDU CR tenemos :

- **LI :** Longitud del cabecero (parte fija+parte variable, no incluido este byte).

- **Código de TPDU :** 4 primeros bits del 2° byte indica código de TPDU.

- **CDT :** (Credit Allocation). Variable de control de flujo inicial. Permite negociar el tamaño de ventana o número de TPDUs a enviar antes de recibir un reconocimiento que valide la anteriores TPDUs. Este tamaño de ventana es continuamente variable, salvo el inicial. Ver apartado 3.3.2.4.

- **N° de Referencias Destino/Fuente:** números de referencia destino (en este caso no ha sido asignado aún, por lo que será cero) y origen que asociarán a la TC tanto la entidad de transporte que llama (origen) como la que es llamada (destinatario),

- **Clase :** Clase de protocolo de transporte OSI. Clase de servicio requerido (p.e. Clase 2 o Clase 4).

- **Opción :** Negociación de campos de secuencia y de crédito (CDT) **normales** (números de secuencia de 7 bits y valores de crédito de cuatro bits) o **extendidos** (secuencia de 31 bits y crédito de 16 bits).

- **Parte Variable :** En esta parte se incluyen ternas de campos indicando :

①tipo de campo de 8 bits,
②longitud de campo de 8 bits y
③valor de campo

representando parámetros (ver apartado 1.3.3.5) como :

⇨ **N° de version.**

⇨ **Negociación de Tamaño Máximo de TPDU:**

Longitud máxima de las TPDUs de datos subsecuentes. Por lo general, la longitud depende del tipo de red(es) subyacente(s) que se use(n) y puede variar entre ***128*** *y* ***8192*** *bytes, en incrementos de potencias de 2.*

⇨**Punto de Acceso al Servicio.**

Esta TPDU CR contiene Números de Referencia y puntos de acceso al servicio (oculto en el campo de parámetros dentro de la parte variable). Ambos valores tienen significación extremo-extremo. En este caso el número de referencia destino no esta predeterminado (puesto a cero) ya que el otro extremo puede seleccionar su propio número para usar.

- **Datos de Usuario :** Un máximo de 32 bytes pueden ser enviados en una TPDU CR.

La **TPDU CC (Confirmación de Conexión)** tiene el mismo formato anterior, salvo en el campo de código (primeros 4 bits del segundo byte) que llevarán el número **1101** y como referencia destino incluirá ya la referencia adecuada para utilizar en las TPDUs de datos.

1.3.3.2 Fase de Transferencia de Datos.-

Un TS_user inicia la transferencia de datos a un usuario correspondiente por una conexión previamente establecida emitiendo la primitiva T_DATA.request. Luego, la entidad de transporte local transfiere los datos de usuario (TSDU) dentro de una o más TPDU **DT**, dependiendo de la cantidad de datos de usuario contenida en la TSDU y del tamaño de TPDU máximo especificado para la conexión. Cada TPDU DT contiene un marcador (**EOT**) que se pone a 1 si se trata de la última TPDU de una secuencia que constituye una sola TSDU. Así mismo, cada TPDU DT contiene un número de secuencia así como la TPDU **AK**, para la confirmación y el control de flujo. Una vez que la entidad de transporte del destino ha recibido y confirmado todas las TPDU DT que constituyen una TSDU, pasa el bloque de datos reensamblado (es decir, la TSDU) al usuario correspondiente por medio de una primitiva T_DATA.indication.

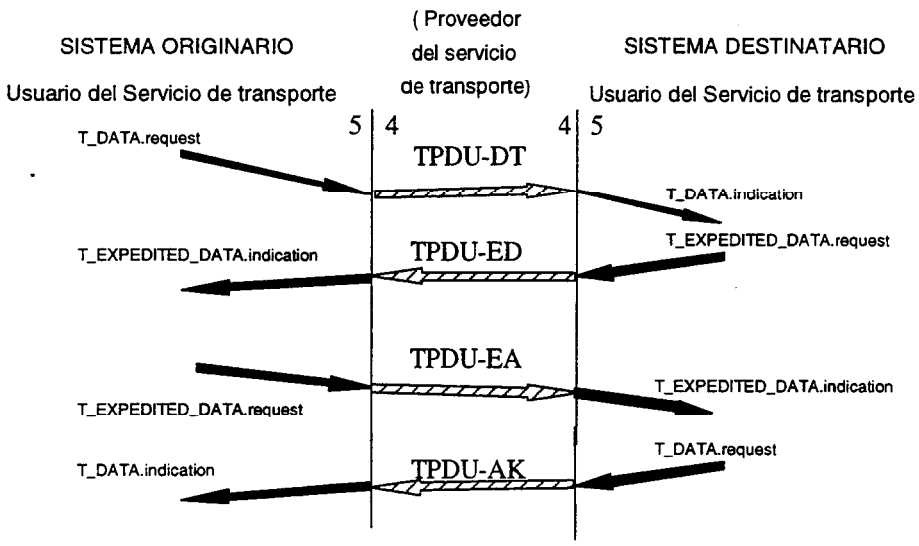


Fig. 1.24 TPDU de Datos intercambiadas (AK confirmación recepción).

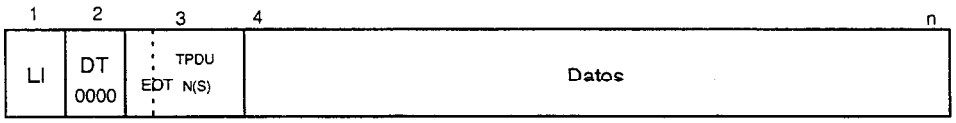
En la figura 1.24 se observa una TPDU de datos normales, una urgente en sentido contrario, la confirmación dc la acelerada y finalmente la confirmación de la normal.

Las entidades de transporte pueden aceptar datos para transferirlos en cualquiera de los dos sentidos por la conexión lógica establecida.

El formato de la TDPU **DT** y TDPU **ED** se muestra en la figura 1.25 con los siguientes campos

- **LI :** Longitud del cabecero (parte fija+parte variable, no incluído este byte).
- **Código de TPDU :** 4 primeros bits del 2° byte indica código de TPDU (DT o ED).
- **DST-REF :** números de referencia destino que ha sido asociado a la TC tanto la entidad de transporte que llama (origen) como la que es llamada (destinatario),
- **Datos de Usuario :** Un cantidad de datos de TSDU negociados en la fase de conexión.

FORMATO DE LA PRIMITIVA TPDU-DT CLASE 0 o 1

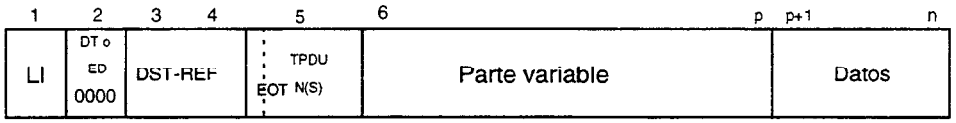


EOT : Bit 8 del byte 3 si es 1 indica que la actual DT es la última TPDU de una secuencia, o sea el final de una TSDU

2º Byte : 1111 0000 => DT

(NS) : Numero se secuencia de TPDU (tamaño normal (7 bits), extendido (31 bits)).

FORMATO DE LA PRIMITIVA DT o ED-TPDU CLASE 2,3 o 4



EOT : Bit 8 del byte 5 si es 1 indica que la actual DT es la última TPDU de una secuencia, o sea el final de una TSDU

2º Byte : 1111 0000 => DT
0001 0000 => ED

(NS) : Numero se secuencia de TPDU(tamaño normal (7 bits), extendido (31 bits)).

Fig. 1.25 Formatos de TDPU de Datos

En estas TPDUs se destaca para clase 0 y 1 :

- Formato más sencillo de todas las TPDUs.
- No lleva referencia destino.
- Se indica 1 byte para número de secuencia TPDU N(S) para el caso de secuencia normal o 3 bytes para número de secuencia extendido.
- 8 bits de número de secuencia si es 1 => Ultima de TDPU de una misma TSDU.
- No tiene Parte variable.

Para las clases 2, 3 y 4, por sus características y sobre el tipo de red sobre el que suelen actuar tendrá :

- ***Solo referencia destino.***
- ***Incluye parte variable(Parámetros de Checksum (si se usa)).***
- ***Tamaño de datos negociado en fase de conexión.***
- ***➢ En TPDU-ED máximo de 16 bytes y EOT = 1.***

A diferencia de los datos normales, los datos urgentes o acelerados, por estar limitados por control de flujo, solo un paquete de datos acelerados puede ser transmitido en cualquier momento durante la conexión. Después de transmitir una TPDU ED, el reconocimiento debe ser recibido antes de enviar otra TPDU ED o cualquier otra TPDU DT. Esta asegura que ninguna otra TPDU normal puede adelantar a la TPDU ED anterior (ver figura 24).

En el caso de clases que requieran una confirmación o reconocimiento de la entrega correcta de las TPDU, se definen las TPDU **AK** para confirmar las TPDU DT y TPDU **EA** para confirmar la TPDU ED (necesario en las clases 3 y 4 y negociable en la clase 1). A diferencia de la mayor parte de los protocolos de transmisión fiables, el TP no usa una retransmisión de reconocimientos sobre PDU de datos independientes. En lugar de ello, se utiliza la técnica que consiste en aprovechar las mismas unidades de datos concatenando **ACK** PDU con **DT** PDU para formar una Unidad de Datos del Servicio de Red que vayan para el mismo destinatario.(ver figura 1.26). Esta técnica es bastante interesante porque permite ahorrar tiempo de cálculo y el overhead que se produciría de otra manera. Por otro lado puede provocar un aumento en el tamaño del paquete que requerirá ser segmentado posteriormente.

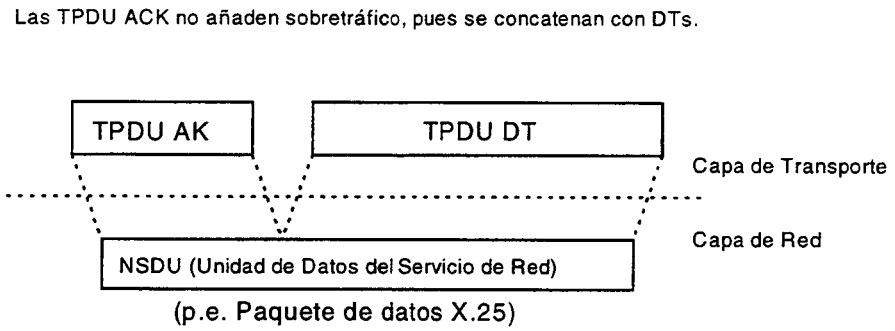


Fig. 1.26. Concatenación de ACK con TPDU de Datos.

El formato de estas dos TPDU se muestran en la figura 1.27:

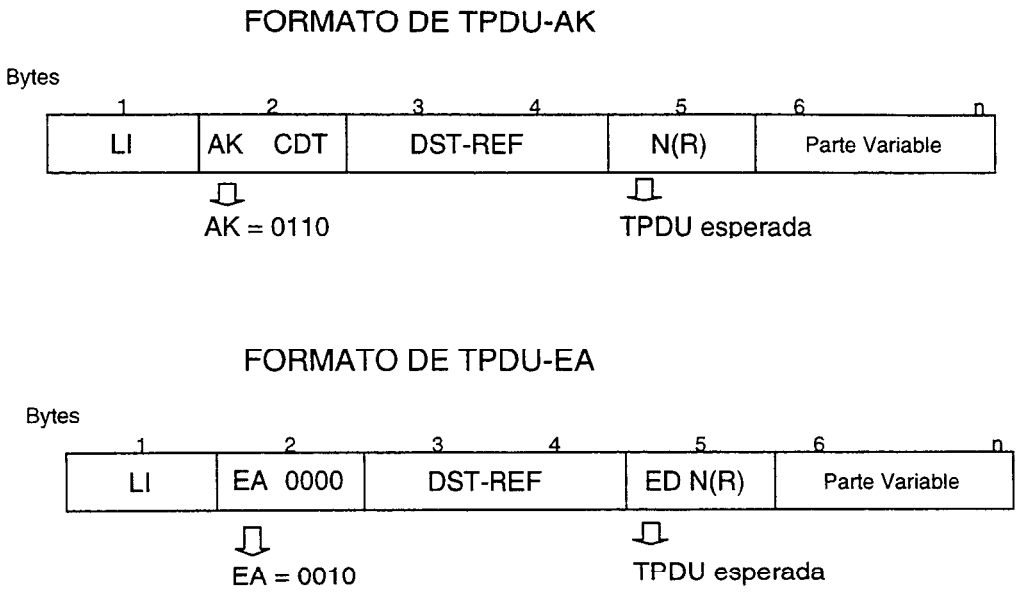


Fig. 1.27. Formatos de ACKs

donde se aprecia la existencia de solamente la referencia destino y los números de secuencia. Estos son los valores que el extremo receptor selecciona para esta conexión. No obstante, la palabra DST-REF debe ser interpretado en el contexto del sistema extremo transmisor. N(R) ocupa tres bytes si estamos operando en modo extendido.

Además incluye la parte variable que indica los campos :

- **Checksum,**
- **Confirmación de control de flujo,**
- **Número de sub-secuencia (resuelve cambios de ventana fuera de banda),**
- **Tamaño de ventana (crédito), otros.**

La característica de un AK es similar al paquete de supervisor de protocolos de enlace de datos. El AK provee el flanco de bajada de la ventana, y el campo **credit** provee el tamaño de ventana (valor de **control de flujo** (apartado 1.3.3.2.4)). Si se indica que el tamaño de ventana es cero es el equivalente de **parar transmisión**.

Dos campos muy importantes se incluyen en la parte variable. El primero de ellos es el campo **número sub-secuencia** que permite poner números de secuencia sobre TPDU-AK. Supongamos que se envia una TPDU AK con un campo de crédito de 10. y más tarde otra TPDU AK con un campo **crédito** de 20, de esta forma se esta intentando incrementar el tamaño de la ventana fuera de banda. El receptor puede obtenerlos en cualquier orden y no sabría cuál utilizar. El número de sub-secuencia resuelve este problema porque la TPDU-AK de 20 tiene un número de sub-secuencia mayor. El

segundo campo es el campo de confirmación de control de flujo. Permite a los extremos compaginar sus respectivos valores para llevar el adecuado control de flujo.

1.3.3.2.1 Procedimiento Control de Flujo

El funcionamiento del procedimiento de confirmación de TPDU recibidas usado con la clase 4, se basa en la técnica de **retroceder N** (ver anexo). Cuando el receptor recibe la siguiente TPDU DT en orden (o la que completa una secuencia contigua de TPDU) devuelve una TPDU AK con un número de secuencia de recepción , **N(R)**, que confirma positivamente la recepción correcta de todas las TPDU DT hasta la que tiene un número de secuencia de transmisión igual a **N(R)-1**, inclusive.

Si la entidad de transporte receptora recibe una TPDU DT **fuera de secuencia** (esto es, con un número N(S) mayor que el de la siguiente TPDU DT en orden esperada), devolverá una TPDU **RJ** (confirmación negativa) con un N(R) que indica el N(S) de la siguiente TPDU DT en orden esperada. Con las TPDU AK y RJ se utiliza un mecanismo de tiempo máximo para solucionar la pérdida de estas TPDU.

Si la capa de red no garantiza que las TPDU DT siempre lleguen en orden, la entidad de transporte receptora se basará en los números de secuencia contenidos en cada TPDU DT para reensamblarlas en el orden correcto. En tales casos, el receptor sólo devolverá una TPDU AK que indique la recepción correcta si ha recibido la siguiente TPDU DT en orden o la TPDU DT que completa una secuencia contigua de TPDU pendientes.

A continuación (figura 1.28) se ilustra el proceso de confirmación de TPDU recibidas mediante la combinación de N(S) en TPDU DT y N(R) en TPDU AK, o en sus equivalentes de unidades de datos urgentes o acelerados.

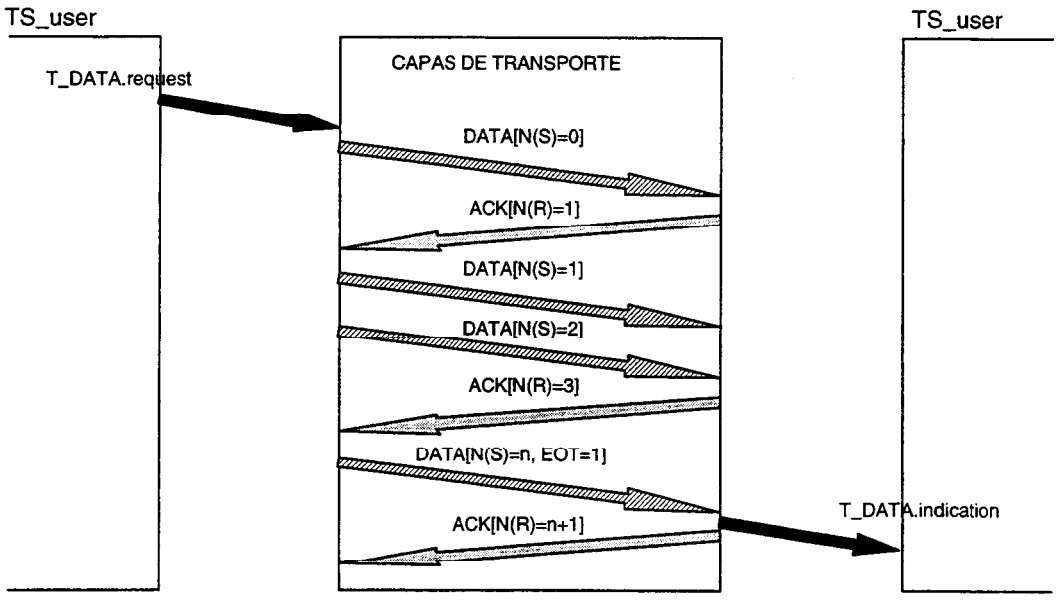


Fig. 1.28 Control de Flujo por ACKs

Como se observa en la figura 1.28, es una situación de transferencia sin ningún problema de una TSDU de datos que se ha segmentado en **n TPDUs**. Con la primera TPDU AK con N(R)=1 se está validando la primera TPDU DT con N(S)=0. Con la TPDU AK con N(3)=3 se están validando con una sola TPDU AK de las N(R)-1 anteriores, por lo tanto las TPDU DT con N(S)=1 y N(S)=2, y finalmente la última TPDU DT lleva EOT a 1 indicando última TPDU de determinada TSDU, y por lo tanto habrá la confirmación final con N(R)=n+1, para validar las n anteriores.

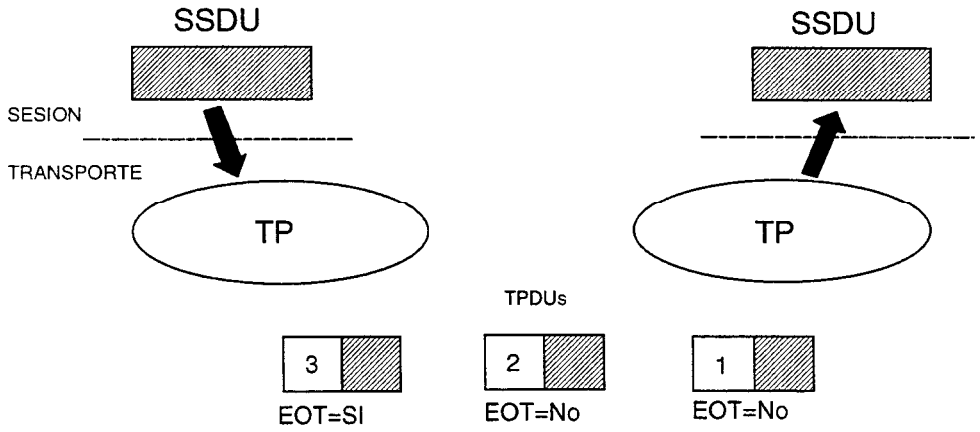


Fig. 1.29 Control de Secuencia con EOT

Otro factor importante es que si el servicio suministrado por la capa de red es de baja calidad, podrían perderse TPDU sin que el remitente ni el destinatario tengan noticia de ello; también podría entregar varias TPDU con errores de transmisión. Por ello, mientras se establece la conexión, el usuario puede especificar una clase de servicio que aplique un **procedimiento de tiempo máximo y retransmisión**, a fin de manejar la posibilidad de pérdida de TPDU, y que también calcule **sumas de verificación** y aplique **mecanismos de detección de errores** para asegurar la integridad de todas las TPDU transmitidas.

1.3.3.2.2 Procedimiento de Tiempo Máximo.-

⌦ Este procedimiento de tiempo máximo se basa en que cuando una entidad de transporte envía una TPDU que requiere una respuesta, pone en marcha un temporizador. Si el temporizador expira antes de que se reciba la respuesta apropiada, se retransmitirá la TPDU y se restablecerá el temporizador. Este ciclo se repite varias veces. Si todavía no se recibe la respuesta correcta, la entidad de transporte supondrá que ya no esta en comunicación con su par, de lo cual informará en seguida al usuario

por medio de una primitiva T_DISCONNECT.indication con la razón de la desconexión como parámetro.

El empleo de tiempos máximos implica la posibilidad de que se generen duplicados, por ejemplo si se recibe correctamente una TPDU pero se pierde su confirmación. Si con base en el número de secuencia se determina, que una TPDU DT es un duplicado de una TPDU previamente recibida, se devolverá una TPDU AK, pero el duplicado será desechado.

1.3.3.2.3 Suma de Verificación.-

⌦ La entidad de protocolo de transporte mantiene la integridad de los datos generando una suma de verificación de 16 bits e incluyéndola como parámetro en la cabecera de cada TPDU que transmite, concretamente en la parte variable. Con un algoritmo similar, el receptor calcula una suma de verificación para la TPDU completa (incluido el parámetro de suma de verificación), que será igual a cero si la TPDU no contiene errores. Si la suma de verificación calculada es distinta de cero, se desechará la TPDU, pero los mecanismos de tiempo máximo y retransmisión asegurarán el envío de otra copia.

Todas las TPDU CC y CR cuentan con una suma de verificación y, si se escogen los servicios de la clase 4, también todas las demás TPDU. El propósito de esta suma es detectar las TPDU que contengan errores residuales (no detectados) después de ser transferidos a través de la red. Este algoritmo se especifica en la norma denominado **Checksum de Fletcher**.

Este algoritmo calcula dos bytes de suma de verificación, X e Y, tales que :

$$X = -C1 + C0$$

$$Y = C1 - 2C0$$

donde

$$C0 = \sum a_i \ (0 \text{ módulo } 255)$$

$$C1 = \sum (L + 1 - i)\, a_i \ (0 \text{ módulo } 255)$$

ambos sumatorios desde **i=1** hasta **i=L**. C0 es la suma en módulo 255 de los bytes en la TPDU (L) y C1 es la suma módulo 255 de los bytes de la TPDU multiplicados por su posición relativa dentro de ella (i).

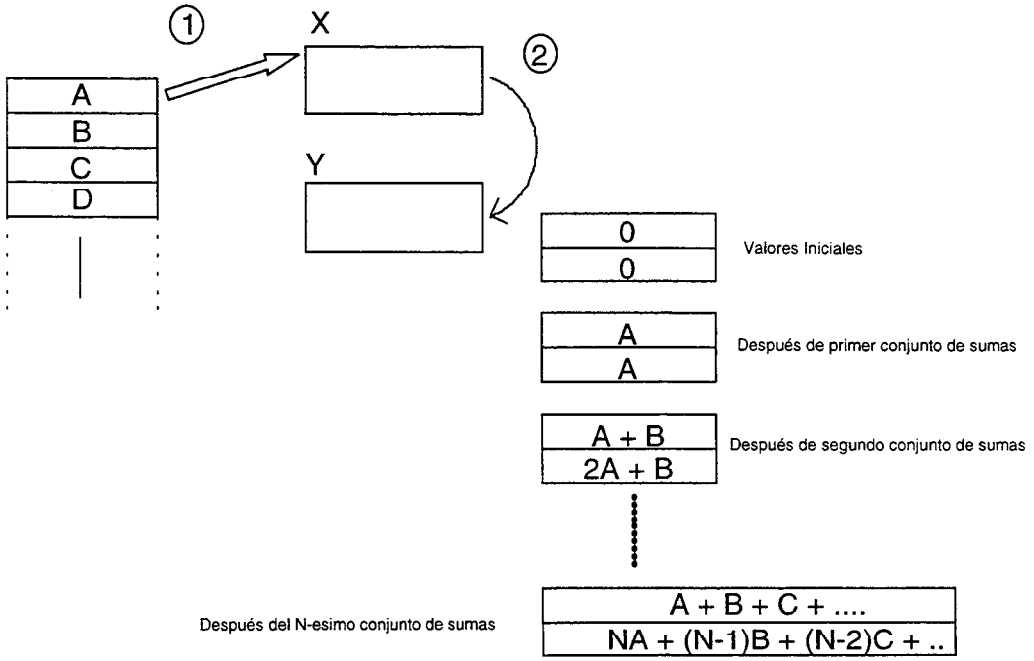


Fig. 1.30 Secuencia de cálculo de checksum

L es el número de bytes de la TPDU completa y $\mathbf{a_i}$ es el valor del i-ésimo byte de la TPDU. Se supone que los contenidos de la TPDU son una cadena de enteros sin signo de ocho bits, en la que los dos bytes de suma de verificación (X e Y) son inicialmente cero.

El procedimiento se basa en los siguientes pasos :

① Asignar cero a C0 y C1 como valor inicial.
② Procesar cada byte en orden desde i = 1 hasta L
③ En cada etapa :
a) sumar el valor del byte a C0;
b) sumar el nuevo C0 a C1.
④ Calcular X e Y de modo que :
X = -C1 + C0; Y = C1 - 2C0.

A continuación se muestra un ejemplo :

Ej. Se supone que el contenido de la TPDU es de 3 bytes a la que se añade X e Y:

i=	1	2	3	4	5
	5	9	6	X	Y

L = 5

Generación de la suma de verificación :

i=0	C0 = C1 = 0	X = Y = 0
i=1	C0 = 0 + 5 = 5	C1 = 0 + 5 = 5
i=2	C0 = 5 + 9 = 14	C1 = 5 + 14 = 19
i=3	C0 = 14 + 6 = 20	C1 = 19 + 20 = 39
i=4	C0 = 20 + 0 = 20	C1 = 39 + 20 = 59
i=5	C0 = 20 + 0 = 20	C1 = 59 + 20 = 79

X = -79 + 1.20 = -59 (196)
Y = +79 - 2.20 = +39 (39)

Comprobación de la suma de verificación :

i=0	C0 = C1 = 0	X = -59 (196) Y=+39(39)
i=1	C0 = 0 + 5 = 5	C1 = 0 + 5 = 5
i=2	C0 = 5 + 9 = 14	C1 = 5 + 14 = 19
i=3	C0 = 14 + 6 = 20	C1 = 19 + 20 = 39
i=4	C0 = 20 - 59 = -39 (216)	C1 = 39 -39 = 0 (255)
i=5	C0 = -39+39 = 0 (255)	C1 = 0 + 0 = 0 (255)

Observe en el ejemplo que este procedimiento produce el mismo C1 que el obtenido al hacer la sumatoria de $(L+1-i)a_i$. Una vez calculados, los dos bytes de la suma de verificación (X e Y) se insertan en la TPDU antes de su transmisión; en el receptor se sigue una secuencia de pasos similar durante la fase de comprobación. Entonces, si cualquiera de C0 o C1 es cero, se supone que no hay error, pero si tanto C0 como C1 son distintos de cero, se supone que hubo un error y se ignora la TPDU. Es por esta razón que se incorpora un mecanismo de tiempo máximo en el protocolo.

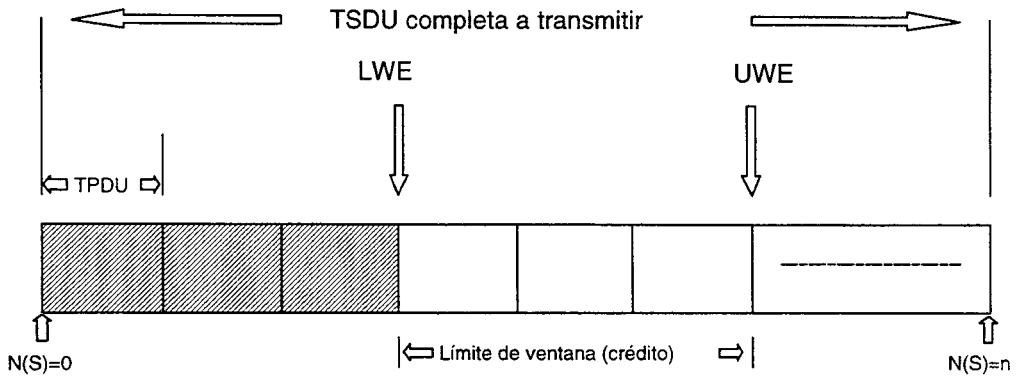
En el cálculo de C0 y C1 se emplea aritmética módulo 255, esto es, aritmética sin signo (no hay desborde y el acarreo se ignora), y se supone que el resultado cae dentro del intervalo [0..255]. Los dos bytes de la suma de verificación (X e Y) se calculan con aritmética en complemento a uno. Ello implica que se usa un acarreo cíclico, y que un resultado de 255 se considera como cero.

El aspecto negativo de este chequeo de error es que requiere un elevado recurso de computación para su cálculo, pero compensa los posibles errores que puede detectar.

1.3.3.2.4 Control de Flujo mediante créditos.-

Este mecanismo de control de flujo que se utiliza con la clase 4 se basa en una variación del protocolo de ventana deslizante. En el campo CDT de cada TPDU CR y TPDU CC intercambiadas durante el establecimiento de una conexión se especifica, para cada dirección de transmisión, un valor de crédito inicial igual al número TPDU DT pendientes (no confirmadas). En el momento de establecerse la conexión se pone en cero el número de secuencia inicial para cada dirección de la transmisión, y este número se convierte en el **borde inferior de la ventana (LWE)**. El transmisor puede calcular

continuamente el **borde superior de la ventana (UWE)** sumando en aritmética módulo T (donde T es el tamaño del campo de secuencia de recepción) el valor del crédito de la conexión al LWE. El flujo de TPDU DT se detiene si N(S) llega al valor UWE. El LWE se incrementa continuamente conforme se reciben las TPDU AK de las TPDU DT pendientes. Ver figura 1.31 y la breve explicación que se incluye.



-> El valor inicial de LWE es cero y se incrementa cada vez que se recibe una TPDU AK.

-> El valor inicial de UWE es el valor CDT convenido cuando se estableció la conexión.

Posteriormente se incrementa sumándosele el valor CDT contenido en cada TPDU AK recibida.

-> El flujo se detiene si N(S) es igual a UWE.

Fig. 1.31 Tamaño de Ventana variable por créditos.

El número real de TPDU DT nuevas que el transmisor puede enviar es variable durante el tiempo de vida de la conexión, pues esta completamente bajo el control del receptor. Cada TPDU AK contiene, además de un número de secuencia de recepción, un nuevo valor de crédito que especifica el número de TPDU nuevas que el receptor esta en condiciones de aceptar después de la que esta confirmando. No obstante, el valor de crédito suele utilizarse cuando el receptor asigna un número fijo de **buffers** a una conexión. Cada vez que se recibe una TPDU, el número de TPDU nuevas que el receptor está preparado para aceptar (el UWE) se reduce porque los buffers comienzan a llenarse.

1.3.3.3 Fase de Liberación.-

La terminación (o liberación) de una conexión se inicia cuando cualquiera de los TS_user emite una primitiva T_DISCONNECT.request a su entidad de transporte local, con la razón de la liberación como parámetro. **En la clase 0, la terminación de la TC implica también la terminación de la conexión de red (NC) asociada**, pero en las demás clases la TC puede terminarse independientemente de la NC. Al recibir la

primitiva T_DISCONNECT, la entidad de transporte envía una TPDU **DR** e ignora todas las TPDU que recibe después hasta recibir una TPDU **DC**. La entidad par, al recibir la TPDU DR, devolverá una TPDU DC y emitirá una T_DISCONNECT.indication al TS_user corresponsal. En este momento se considera que la TC está cerrada.

En la figura 1.32 se ilustran las primitivas que intervienen y las TDPU asociadas en cada sistema:

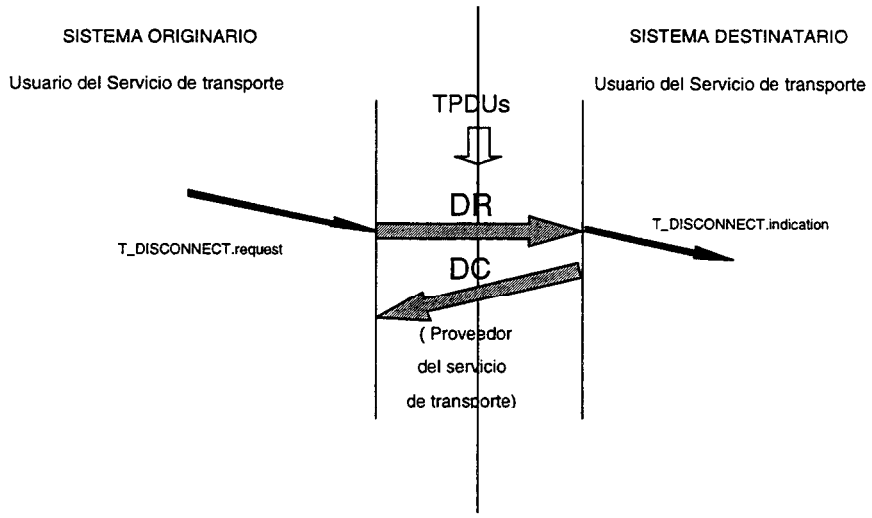


Fig. 1.32 TPDU de desconexión intercambiadas.

Como se observa en la figura, la liberación de la conexión es un servicio no confirmado o **abrupto**, o sea no requiere primitivas de respuesta y confirmación.

El formato de la TPDU DR se muestra a continuación (figura 1.33):

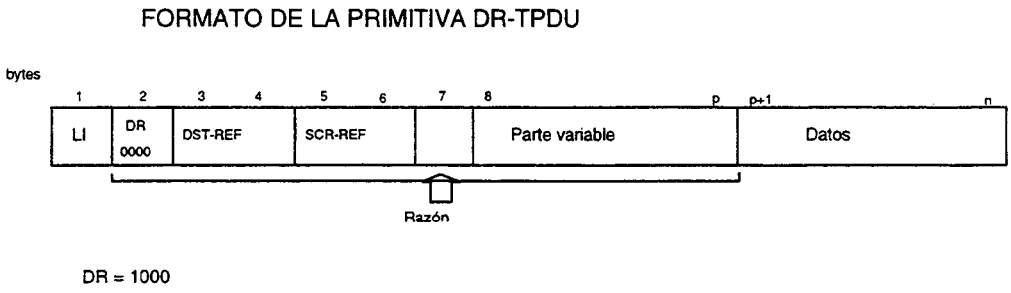


Fig. 1.33 Formato de TPDU de desconexión

En el caso de TPDU DR tenemos:

- **LI :** Longitud del cabecero (parte fija+parte variable, no incluído este byte).

- **Código de TPDU :** 4 primeros bits del 2° byte indica código de TPDU.

- **DST-REF y SCR_REF N° de Referencias Destino/Fuente:** números de referencia destino y origen que asociado a la TC que se va a liberar tanto la entidad de transporte que llama (origen) como la que es llamada (destinatario),

- **Razón.:** La causa o motivo de la desconexión. Parámetros correspondiente al indicado en la primitiva T_DISCONNECT (ver aptdo. 2.1.3).

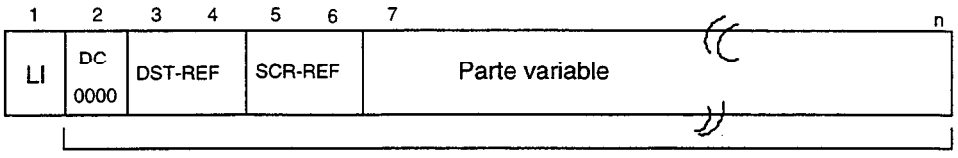
- **Parte Variable :** En esta parte se incluyen ternas de campos indicando :

①tipo de campo de 8 bits,
②longitud de campo de 8 bits y
③valor de campo

- **Datos de Usuario :** Un máximo de 64 bytes pueden ser enviados en una TPDU DR.

La **TPDU DC (Confirmación de Desconexión)** tiene el mismo formato que la anterior, salvo en el campo de código (primeros 4 bits del segundo byte) que llevarán el número **1100** y no llevará campo de **Razón** ni **Datos de Usuario**. (Ver figura 1.34).

FORMATO DE LA PRIMITIVA DC-TPDU



DC = 1100

Fig. 1.34 Formato de TPDU de confirmación de desconexión

Como resumen, el proceso completo de primitivas y TPDUs que intervienen en una conexión completa sencilla se muestra en la figura 1.35 (**para clases 0 y 2 => no existen TPDU AK (ACKs), en clase 1 es opcional):**

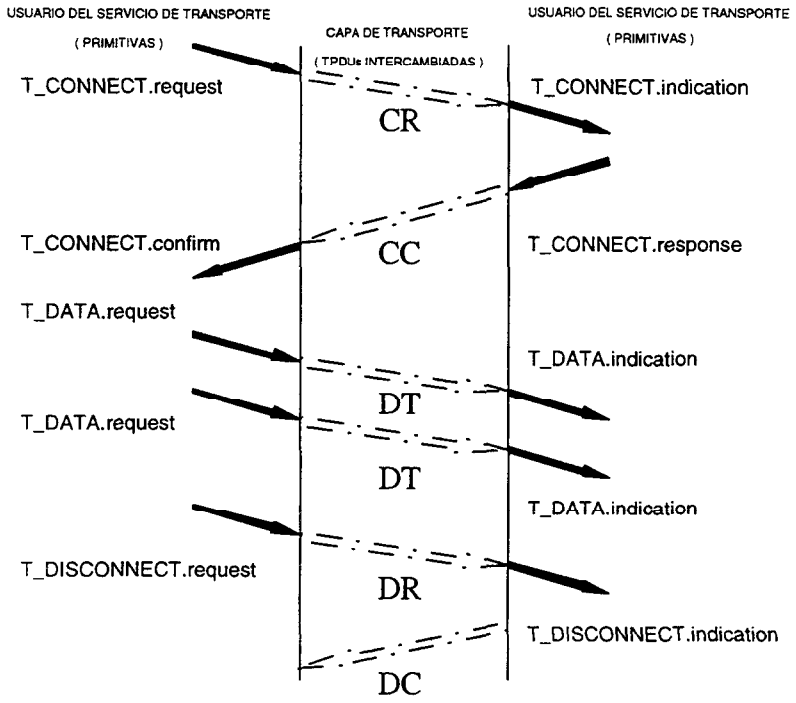


Fig. 1.35 TPDUs intercambiadas y Primitivas.

1.3.3.4 Otras TPDUs.

Como se observa en la lista de TPDU (ver figura 1.21) se definen las Unidades de Datos de Protocolo **Reject (RJ)** y **Error (ER)**.

El uso de **Reject o Error** se utiliza en situaciones excepcionales durante la transferencia de unidades de datos, concretamente como se verá más adelante, utilizadas para el tratamiento y recuperación de errores y en el caso de problemas o errores del protocolo (ver apartado 1.3.4.22 Errores de Protocolo).

El formato de ambas TPDU se muestra en la figura 1.36 :

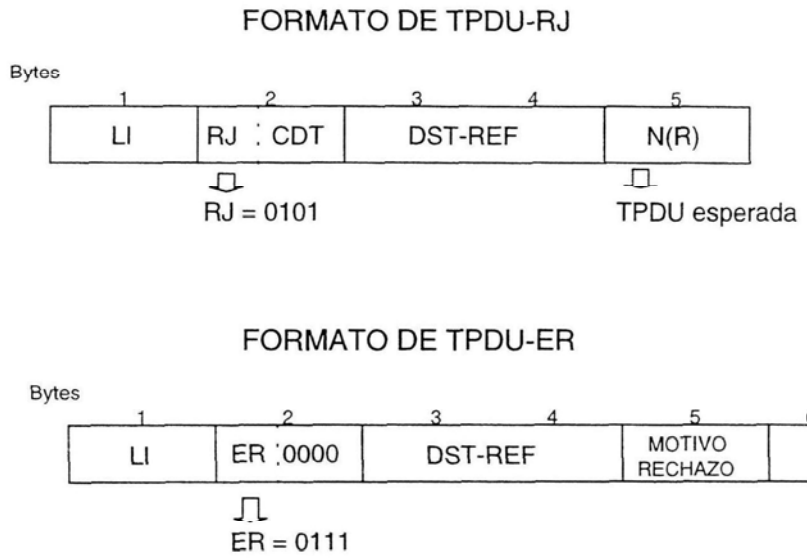


Fig. 1.36 Formato de TPDU RJ y TPDU ER

1.3.3.5 Campo de Parte Variable

La siguiente tabla muestra los diferentes parámetros que pueden aparecer en la parte variable del cabecero de las TPDUs. La mayoría de estos campos sólo aparecen en las TPDUs CR y CC, es decir, sólo tienen utilidad en el proceso de apertura de la conexión.

Campo	TPDUs	Notas
Dirección TSAP llamante	CR,CC	
Dirección TSAP llamado	CR,CC	
Tamaño TPDU	CR,CC	
Número de versión	CR,CC	
Parámetro de seguridad	CR,CC	
Checksum	Todas	(1)
Selección de opciones adicionales	CR,CC	
Clase de protocolo alternativo	CR,CC	
Tiempo para acuses de recibo	CR,CC	(2)
Velocidad de transferencia	CR,CC	
Proporción de errores residuales	CR,CC	
Prioridad	CR,CC	
Retraso de tránsito	CR,CC	
Tiempo de reasignación	CR,CC	(3)
Información adicional	DR	
Número de subsecuencia	AK	(2)
Confirmación de control de flujo	AK	(2)
TPDU no válida	ER	

Notas:
(1) : Todas las TPDUs exepto RJ y ER. Sólo para clase 4.
(2) : Sólo para clase 4.
(3) : Sólo para clase 1 y 3.

Alguno de estos parámetros representa :

Tamaño de TPDU : Las entidades de transporte negocian entre sí el tamaño máximo que podrán tener las TPDUs. Este parámetro es importante para posteriores procesos de segmentación.

Número de versión : Se utiliza para las entidades de transporte se pongan de acuerdo en la versión del protocolo que se va a utilizar. Esto se hace en previsión de posibles cambios.

Parámetro de seguridad : Definido por el usuario.

Checksum : Aplicando el algoritmo estandarizado a la TPDU se obtiene un valor, que se inserta en este campo. Si, en recepción, no se obtiene el mismo valor al hacer el mismo cálculo, se asume que la TPDU recibida se ha visto afectada por algún error en la transmisión. (ver apartado 1.3.2.2.3).

Selección de opciones adicionales : Se utiliza para negociar el uso o no de algunas características de la conexión, como : (1) uso del canal urgente de datos de la red en la clase 1, (2) uso de acuses de recibo a nivel de transporte o a nivel de red en la clase 1 , (3) uso de checksums en la clase 4, y (4) uso o no de un canal de transporte urgente.

Clase de Protocolo Alternativo : En la cabecera fija se propone una clase de protocolo. En este campo se ofrece otra clase que también puede ser aceptable.

Tiempo para acuses de recibo : Tiempo máximo de espera al acuse de recibo de una TPDU, antes de darla por pérdida y proceder a retransmitirla.

Tiempo de reasignación : Intervalo de tiempo durante el cual la entidad de transporte intentará realizar una reconexión tras un fallo en la red, antes de dar la conexión de transporte por pérdida.

Información Adicional : Más información sobre el motivo de una desconexión

Número de Subsecuencia : Utilizado para numerar secuencialmente los acuses de recibo.

Confirmación de control de flujo : Copia de los valores Credit, TPDU esperada y Número de subsecuencia de la última TPDU AK recibida.

TPDU no válida : Copia de la TPDU que causó el error.

La codificación de estos parámetros, como ya se comentó, consta de tres partes : ***un tipo***, que dice de qué parámetro se trata, una ***longitud*** del valor del parámetro, y el ***valor*** correspondiente.

31..4 Funciones realizadas por el Protocolo de Transporte (Procedimientos).-

Una vez analizadas todas las TPDUs se puede integrar en la siguiente tabla que TPDU se utilizan en cada una de las clases de protocolo :

TPDU		*Clase 0*	*Clase 1*	*Clase 2*	*Clase 3*	*Clase 4*
CR	Connect Request	SI	SI	SI	SI	SI
CC	Connect Confirm	SI	SI	SI	SI	SI
DR	Disconnect Request	SI	SI	SI	SI	SI
DC	Disconnect Confirm	NO	SI	SI	SI	SI
DT	DaTa	SI	SI	SI	SI	SI
ED	Expedited Data	NO	SI	1	SI	SI
AK	data AcKnowledge	NO	2	1	SI	SI
EA	Expedited data Ack.	NO	SI	1	SI	SI
RJ	ReJect	NO	SI	NO	SI	NO
ER	TPDU ERror	SI	SI	SI	SI	SI

SI : TPDU siempre disponible, NO : No disponible en clase, 1 : No disponible si no esta seleccionado control de flujo explícito, 2 : No disponible cuando la variante de confirmación de recibo esta seleccionada.

Por otro lado podemos analizar, mediante la siguiente tabla donde se muestran las diferentes funciones del protocolo y en que clases se aplica :

FUNCION	**CLASE**				
Procedimiento (Opciones)	0	1	2	3	4
Asignación a una conexión de red (1)	X	X	X	X	X
Establecimiento de Conexión (5)	X	X	X	X	X
Transferencia de TPDUs (2)	X	X	X	X	X
Segmentación/Reensamblado (3)	X	X	X	X	X
Concatenación/Separación (4)	-	X	X	X	X
Rechazo de Conexión (6)	X	X	X	X	X
Liberación Normal de Conexiones (7) (Implícita)	X	-	-	-	-
(Explícita)	-	X	X	X	X
Liberación por error(8)	X	-	X	-	-
Asociación de TPDU a Conexión de Transporte (9)	X	X	X	X	X
Numeración de las TPDUs de datos (10) (Normal)	-	X	m	m	m
Tanenbaum	-	X	o	X	X
G. Cole Wiley "	-	X	X	X	X
(Acelerados)	-	-	o	o	o

	G. Cole Wiley "	-	-	X	X	X	
Transferencia de Datos acelerados(11)(Normal Red)		-	m	X	X	X	←
	Tanenbaum	-	o	o	X	X	
	G. Cole Wiley	-	X	X	X	X	
	(Acelerados Red)	-	ao	-	-	-	←
Reasignación en caso de fallo (12)		-	X	-	X	X	
Retención hasta acuse de recibo (13)J. add. (Red)		-	ao	-	-	-	↔
	(TPDU AK)	-	m	-	X	X	↔
	Tanenbaum y G. Cole Wiley	-	X	-	X	X	
Resincronización (14)		-	X	-	X	X	
Multiplexación ascendente (15)		-	-	X	X	X	
Control de Flujo Explícito (16) Tanenb.Con) (J.H.)		-	-	m	X	X	
	G.Cole Wiley	-	-	X	X	X	
	(Sin)**	X	X	o	-	-	
Checksum (17)	(Usar)**	-	-	-	m	X	
	G. Cole Wiley	-	-	-	-	X	
	(No usar)**	X	X	X	X	o	
(J. H. Ellis Horwood) y Tanenbaum		-	-	-	-	m	
Referencias congeladas (18)		-	X	-	X	X	
Retransmisión con Temporizador (19)		-	-	-	-	X	
Resecuenciamiento (20)		-	-	-	-	X	
Control de inactividad (21)		-	-	-	-	X	
Tratamiento de errores de protocolo (22)		X	X	X	X	X	
Multiplexación descendente (23) (splitting y recombinación)		-	-	-	-	X	

Donde : **X** : Siempre incluído en el protocolo.

- : Nunca disponible en esta clase.

m : Negociable cuya implementación es obligatoria en la entidad de transporte.

o : Negociable cuya implementación es opcional en la entidad de transporte.

ao : Negociable cuya implementación es opcional y depende de su disponibilidad en el servicio de red.

← : No aplicable en clase 2 si el control de flujo explícito no esta seleccionado. Normales Red se refiere a que el servicio de transporte urgente se implementa sobre el servicio de datos normales de red. Acelerados Red se refiere al uso del canal urgente de red (si existe) para los datos acelerados de transporte.

↔ : Red indica que las TPDUs se almacenan hasta que se recibe notificación de entrega por parte del servicio de red. TPDU AK se refiere a que se necesita acuse de recibo explícito por parte de la entidad par.

1.3.4.1 Asignación a una conexión de red.-

Este procedimiento es común a todas las clases, por supuesto, para que sea hecha una asignación inicial a una conexión de transporte (TC), sino no puede ser establecida. Cuando la red ofrece un servicio orientado a conexión, una conexión de transporte está soportado por al menos una conexión de red. La asignación es la asociación de una (potencial) TC con una conexión de red (NC); "**potencial**" porque, en la etapa de la TC, el establecimiento no puede proceder hasta que una asignación sea hecha. En el caso más simple, de relación uno a uno, cuando el usuario solicita el servicio T_CONNECT.request, la entidad de transporte debe comprobar si tiene ya abierta una conexión de red que le sirva para establecer sobre ella la conexión solicitada.

Si no es así, la entidad de transporte debe abrir una conexión de red (N_CONNECT) con su entidad par. No obstante, una vez hecha y establecida la TC, y en el caso de que la NC se pierda, la TC puede ser retenida y asignada a una diferente NC (conexión de red). En cualquier caso la entidad de transporte puede escoger entre establecer una nueva NC (por el uso del elemento de servicio N_CONNECT) o usar una adecuada NC existente.

Una conexión de Red no tiene por qué estar dedicada en exclusiva a una conexión de transporte, sino que puede ser compartida por varias.

A su vez se verá más adelante que una simple TC puede ser asignada a mas de una NC en cualquier momento (**splitting : Multiplexación Descendente ver 3.4.23**) y mas de una TC pueden ser asignadas a una simple NC(**multiplexing : Multiplezacón ascendente 1.3.4.15**).

En la figura 1.37 se muestra la secuencia de primitivas relatada para poder establecer la conexión de transporte sobre una conexión de red.

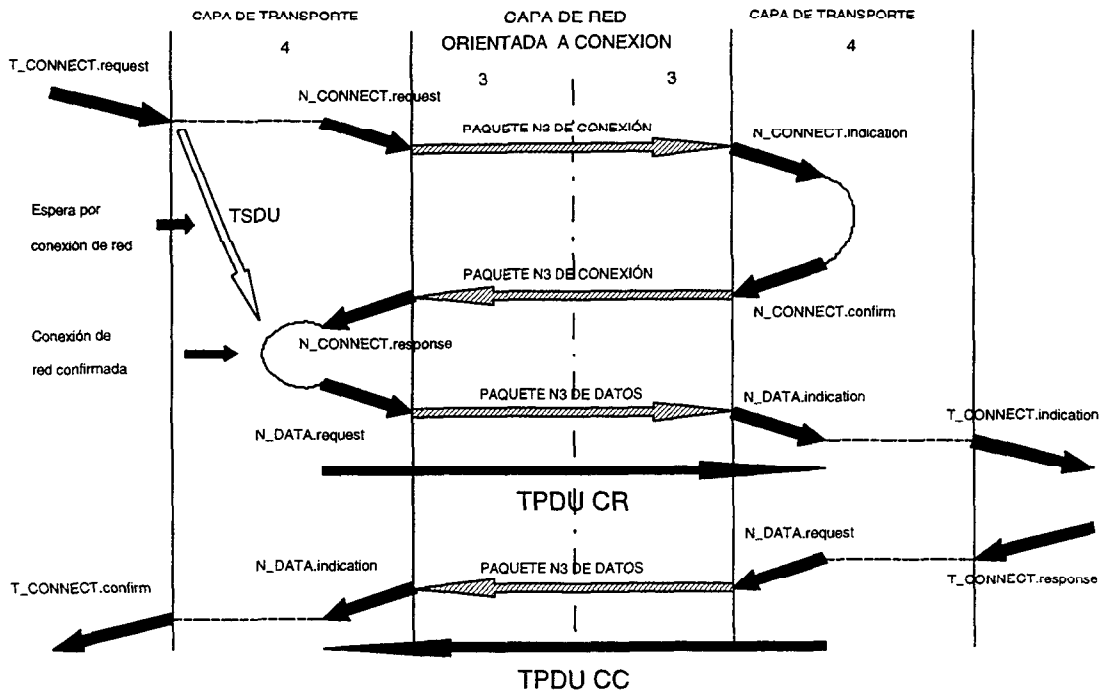


Fig. 1.37. Establecimiento de conexión de red que soporte conexión de transporte.

1.3.4.2 Establecimiento de conexión.-

Este procedimiento esta disponible en todas las clases de protocolo de transporte para establecer una TC después de ser asignada correctamente a una conexión de red. Una vez que se tiene asignada una conexión de red (si es que **el servicio de red es orientado a conexión**) se puede iniciar el establecimiento de la conexión de transporte. Las TPDUs involucradas en el intercambio confirmado son **Connect Request CR** y **Connect Confirm CC**. (Ver figura 1.22).

Una conexión de transporte se establece por negociación entre entidades de transporte pares, por el intercambio de las apropiadas PDUs, las cuales son transportados por el uso de datos normales de red, N_DATA (ver figura 1.37). Como resultado de negociación se determina una QOS para ser utilizada así como una determinada clase de transporte para ser usada sobre la red. Hay procedimientos asociados con clases particulares que son opcionales en si mismos dentro de la clase, y así la negociación de estas características opcionales son también transportadas en este momento.

Durante el transcurso del establecimiento de la conexión, la información es intercambiada entre entidades de transporte en forma de parámetros (sobre las TPDUs CR y CC). Esta información contenida en CR y CC, como ya se detallo en el apartado 2, que definen y configuran el resultado de la TC se compone de :

- **Referencias Origen y Destino** son identificadores arbitrarios escogidos por las entidades de transporte para identificar la TC en cada extremo. La conexión queda identificada por este par de referencias, en vez de por el par <dirección TSAP origen, dirección TSAP destino>.

• **Direcciones Origen y destino** que definen los TSAPs llamantes y llamados para establecer la conexión, posteriormente se usarán las referencias.

• **Crédito inicial** que esta relacionado y es necesario para el control de flujo explícito.

• **Datos de Usuario** que son una serie de bytes de datos que son tratados de manera transparente y suministrados por los usuarios de TS.

• **Checksum** que es un parámetro para negociar si se añade un checksum a las TPDUs (en TP4) o no.

• **Temporizadores para reconocimiento** que solo están relacionados con la clase 4.

• **Parámetro de Seguridad** que es un parámetro general de seguridad cuya semántica es definida por el usuario del TS, que lo son por el **entorno** de la actividad OSI para la que está TC se define.

Los aspectos negociados entre las entidades de transporte y los usuarios durante el establecimiento de TC son :

✹ **Clase de Protocolo.** en función del servicio de red disponible.

✹ **Tamaño Máximo de la TPDU.** El iniciador puede proponer un límite; el receptor puede aceptarlo o proponer un límite más bajo. Dependerá del tamaño de paquete de la red.

✹ **Uso de checksums sobre las TPDU,** solo aplicable a la clase 4 y opcional en esa clase. Aspecto muy importante para aumentar la fiabilidad de la transferencia.

✹ **Calidad de Servicio** que son aspectos del servicio de transporte que son ajustados mutuamente por la negociación a niveles aceptables. Estas incluyen: **Throughput, retardo de tránsito, prioridad, tasa de errores residuales.** (Ver apartado 1.3.5).

✹ **Uso de control de flujo explícito.** Es obligatorio en las clases 3 y 4 y opcional en la clase 2. En la clase 2 luego puede ser excluído durante la negociación. (Ver apartado 3.4.16).

✸ **Uso de confirmación de recepción basado en la red y de intercambio de datos urgentes o acelerados en la red.** Estos dos están relacionados solo con la clase 1 y opcional dentro de esa clase. (Ver apartado 1.3.4.13).

✸ **Uso de servicio de datos de transporte acelerados (urgentes),** disponibles en todas las clases salvo la clase 0.

1.3.4.3 Transferencia de TPDUs.-

Este procedimiento coordina el transporte de TPDUs entre entidades de transporte pares. Usa los elementos del servicio de datos de red normales o urgentes (N_DATA y N_EXPEDITED_DATA) o sobre una red modo datagrama N_UNIT_DATA. Este procedimiento es común a todas las clases de protocolo (ver figura 1.38).

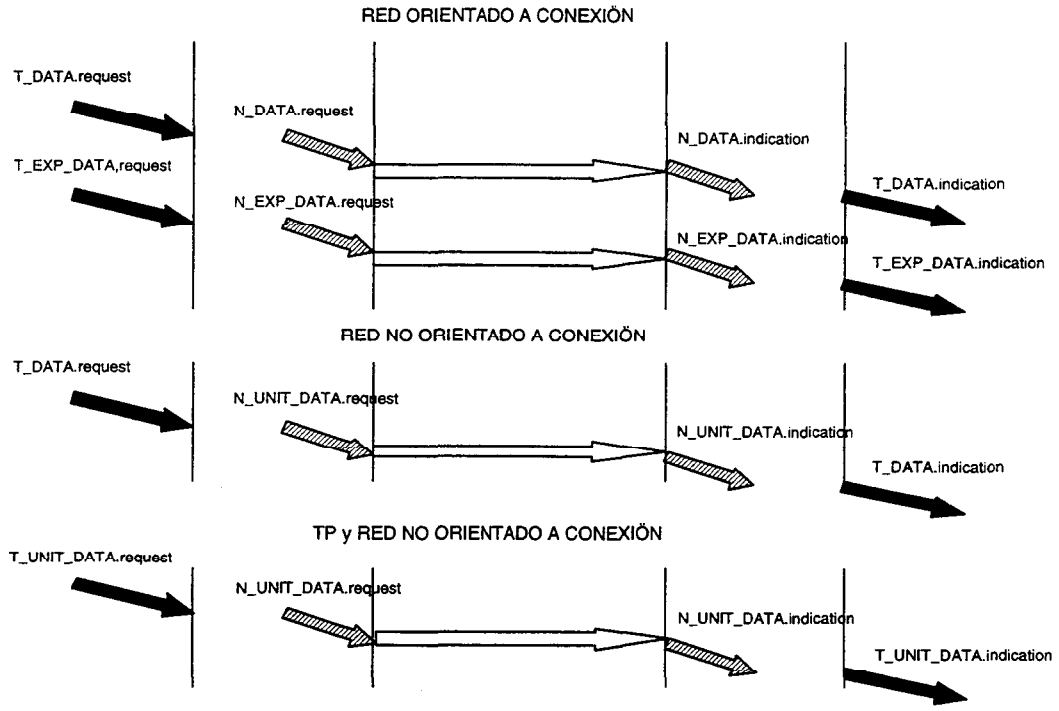


Fig. 1.38. Transferencia de datos según tipo de red.

En la TPDUs de datos DT y ED, la estructura de control de la PDU o sea el PCI, consta de un identificador junto con un parámetro de longitud dando la longitud de la PCI dentro de la PDU. No obstante, no hay indicador de longitud para el campo de datos (ilimitado en el caso de DT) de la PDU. Esta se pasa en su totalidad al proveedor del NS como una NSDU y es a partir de la longitud total de esta NSDU como la entidad de transporte receptora puede determinar el tamaño del campo de datos: p.e. la más pequeña longitud de la NSDU es el campo PCI.

1.3.4.4 Segmentación y Reensamblado.-

Una solicitud para transferir una TSDU por parte de un TS_user puede exceder el límite del tamaño permitido para los datos que pueden ser transportados entre entidades de transporte pares en una simple TPDU de datos. Tal limitación es un reflejo del servicio de red sobre la NSDU asociada con el elemento del servicio N_DATA. Un buen servicio de transporte debe aceptar de sus usuarios TSDUs (mensajes) de cualquier tamaño, independiente del tipo de red. En esta situación la **segmentación** es invocada para romper la TSDU en un serie de TDPUs DT de tamaño apropiado. En otros casos una simple TPDU DT será suficiente (ver figura 1.39).

En la entidad receptora, la secuencia de TPDUs DT representando una TSDU segmentada será **reensamblada** en un simple TSDU y solo cuando esta TSDU en su totalidad se haya recibido, una indicación del servicio de datos será notificada al TS_user receptor con la TSDU completa. Para garantizar que una TSDU pueda ser reconocida por una entidad de transporte receptora, se utiliza el parámetro **EOT,** en cada TPDU DT, y que solamente será activado cuando una TSDU entera haya sido transferida. Como se observa en la figura 39 si una TSDU se segmenta en varias TPDUs, la última de las cuales llevará el parámetro EOT activado (ver figura 1.40).

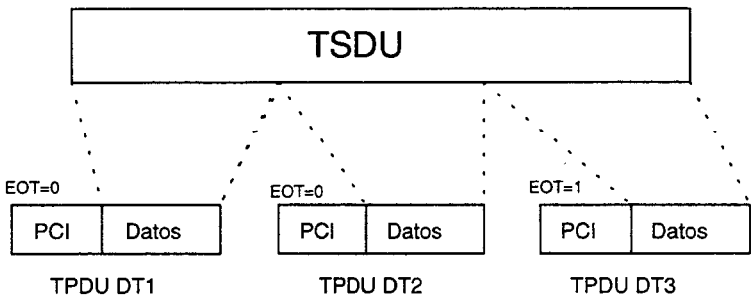


Fig.1. 39 Proceso de Segmentación en varias TPDU

En el caso de que la TSDU quepa enteramente en una TPDU DT, este será activado en todas las TPDU DT. Este procedimiento esta disponible en todas las cinco clases.

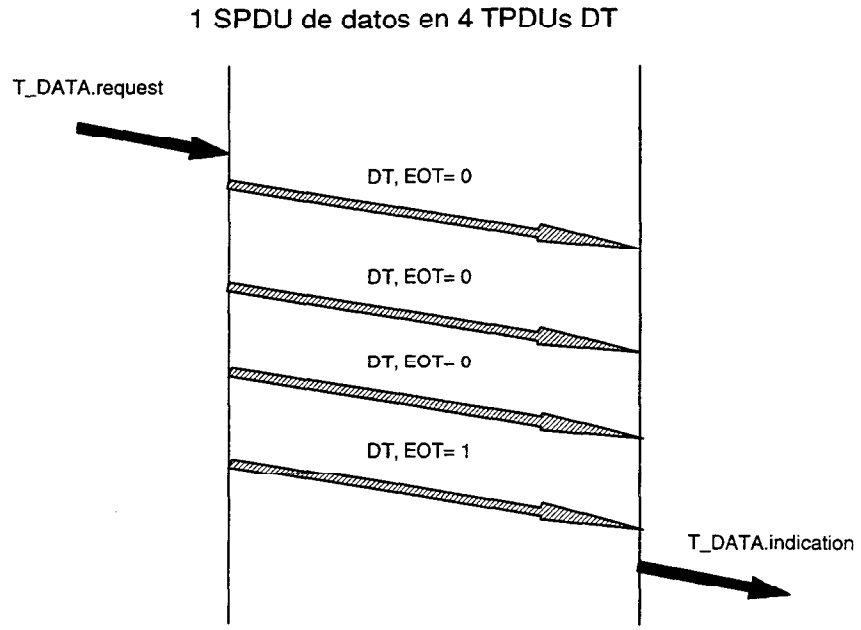


Fig 1.40. Ejemplo de uso de parámetro EOT.

1.3.4.5 Concatenación/Separación.-

En este procedimiento, sujeto a ciertas reglas, un número determinado de TPDUs pueden ser **concatenadas** en una simple NSDU ("paquete" de datos nivel 3) para la transmisión, reduciendo con ello el número de llamadas a la capa de red, y **separadas** por la entidad de transporte receptora en el receptor. Esta disponible en todas las clases, excepto en la clase 0 (ver figura 1.41).

Como la longitud de las TPDUs no se conoce a priori, si una NSDU de datos esta constituida por un grupo de TPDUs concatenadas entonces solo puede haber una DT entre ellas y deberá ser la última TPDU de la concatenación. Simplemente esto es porque una TPDU concatenada sería invisible para el receptor en la entidad de transporte receptora, así aparecería como parte del campo de datos (longitud de datos = (longitud del resto de la NSDU) - (longitud de PCI de TPDU DT)).

El mismo argumento se aplica a concatenación extendida dentro de la capa de sesión.

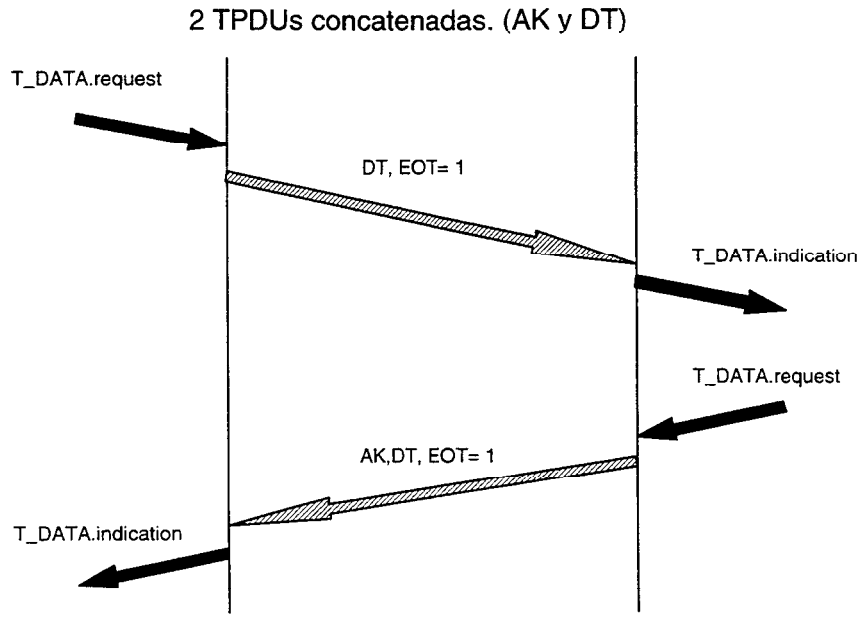


Fig. 1.41 Ejemplo de uso de concatenación de varias TPDUs.

1.3.4.6 Rechazo de Conexión.-

Este procedimiento es iniciado por la entidad de transporte receptora devolviendo T_DISCONNECT.indication ante la imposibilidad de llegar a un acuerdo a los requerimientos en cuanto a calidad del servicio de la entidad de transporte durante el establecimiento de conexión mediante la TPDU CR, devolviendo un T_DISCONNECT.request en lugar de T_CONNECT.response.

Es común a todas las clases de protocolo y se consigue por medio del envío de una petición de desconexión (TPDU DR) al iniciador de la conexión usando los datos de red normales. (ver figura 1.10).

1.3.4.7 Liberación Normal de Conexiones.-

Este procedimiento es común a todas las clases y usado para terminar una conexión de transporte.

En clase 0 hay una relación **uno-a-uno** entre el tiempo de vida de una conexión de transporte y la conexión de red que le es asignada. Esta clase de liberación normal se consigue, por la entidad de transporte o por la desconexión de la conexión de red (usando T_DISCONNECT.request). Una T_DISCONNECT.request genera una N_DISCONNECT.request consecuente y el receptor al recibir N_DISCONNECT.indication dará por cerrada la conexión de red y avisará al TS_user

con T_DISCONNECT.indication. El receptor de una T_DISCONNECT.indication estará habilitado, en clase 0, para generar la liberación de la TC asociada. Esto se conoce como variante **implícita** de liberación normal. También podemos verlo como que las liberaciones normales en todas las clases, salvo en la clase 0, son correlados uno a uno; o sea que la conexión de transporte se libera **implícitamente,** solo con liberar la conexión de red subyacente.

La otra variante, **explícita**, esta asociada con todas las otras clases y aquí la TC es liberada por una actividad confirmada en la que interviene el intercambio entre pares de las TPDUs de Petición de Desconexión (DR) y Confirmación de Desconexión (DC), usando datos de red normales. La necesidad de usar liberación explícita llega a ser clara cuando se considera la multiplexación. En este caso, una vez cerrada la conexión de transporte, el proveedor del servicio de transporte puede optar por **cerrar la conexión de red** que sostenía a la de transporte, o bien **mantenerla abierta** para abrir sobre ella una nueva conexión de transporte.

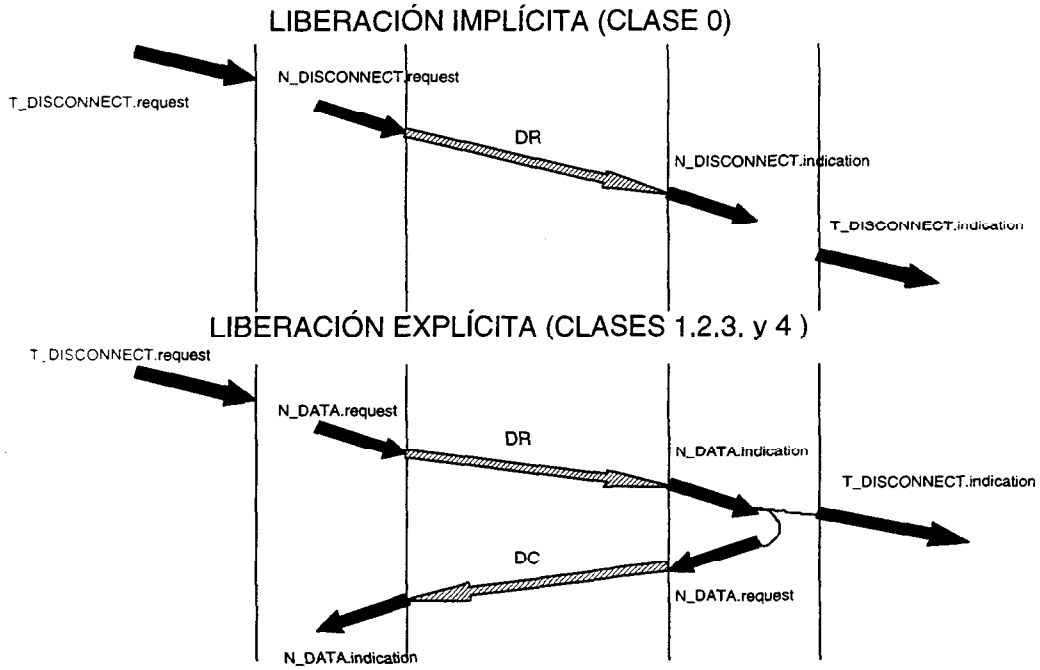


Fig. 1.42 Liberaciones de Conexiones Normales (Implícita y Explícita)

1.3.4.8 Liberación por error.-

Este es un mecanismo usado solamente en las clases 0 y 2 para liberar la conexión de transporte después de que un **error señalizado** (N_RESET o N_DISCONNECT) haya sido recibido desde el proveedor del servicio de red. En estas clases simples (diseñadas para redes tipo A ("seguras")) esta es la única posible acción a realizar, ya que no están preparadas para recuperarse de un error en la red, así que deben liberar la conexión de red o conexiones de transporte implicadas. Los usuarios del TS son, por supuesto, informados de la liberación por una T_DISCONNECT.indication. En las otras clases, los procedimientos estan disponibles para recuperarse de los errores señalizados.

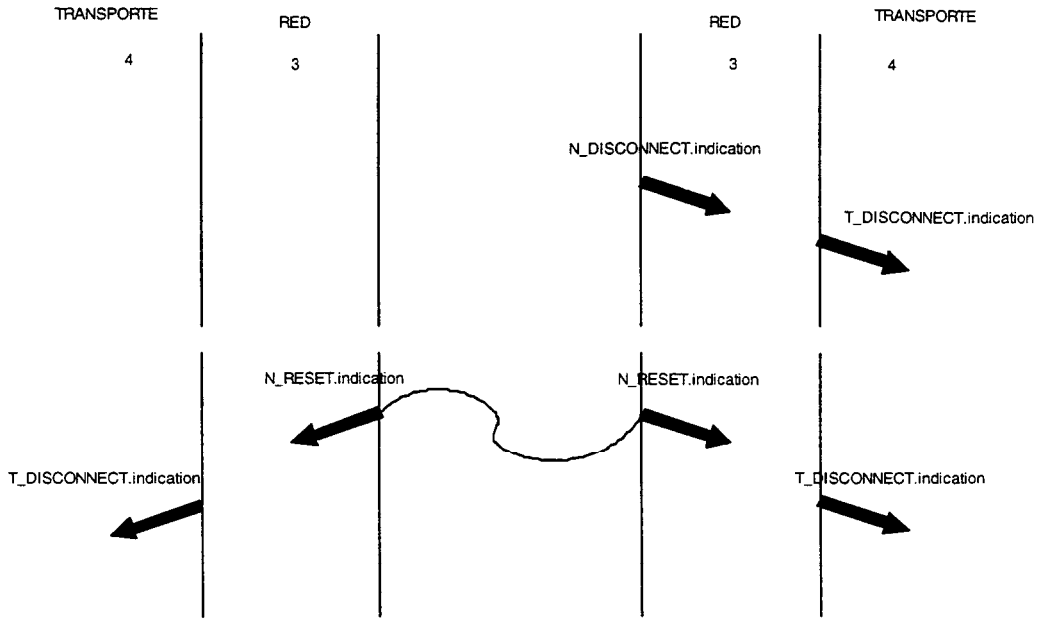


Fig. 1.43 Liberación por error de red.

1.3.4.9 Asociación de TPDUs a conexión de Transporte.-

Este es un procedimiento usado en todas las clases mientras los datos estan siendo recibidos. Cuando una entidad de transporte reciba una NSDU desde el proveedor del servicio de red debe realizar los siguientes pasos :

1. Chequear que la NSDU puede estar codificada en una o más TPDUs **concatenadas**.
2. Si la concatenación es detectada luego se invoca el procedimiento de **separación.**
3. Si la conexión de red sobre la que esta NSDU es recibida tiene asociada múltiples TCs (p.e. **multiplexación**), entonces asegura que las TPDUs sean asociadas con la apropiada TC.

En este tercer paso, se observa que implícitamente la entidad de transporte puede **"gestionar" más de una simple conexión** de transporte en cada momento. De manera general se puede indicar que una entidad n puede manejar cualquier número de n conexiones en cada momento. Cualquier restricción sobre esto será debido a la implementación de la entidad. El único punto de contacto con las capas altas es el TSAP y es el único límite para que la relación entre TCs y NCs sea permitida de manera una-a-muchas o muchas-a-una y estas estén correctamente asociadas.

Obviamente, estos pasos dependerán de la clase utilizada. Así por ejemplo, en clase 0 tendremos :

- Paso 1. En esta clase sólo una única TPDU puede estar presente (concatenación no esta permitida). Si esta NSDU contiene TPDUs concatenadas entonces se invoca un procedimiento de protocolo de error.

- El Paso 2 nunca es invocado en esta clase.
- El Paso 3 tampoco es invocado en esta clase - la TPDU es automáticamente asignada a la única TC.

O sea, de manera básica, se puede decir que se tendrá que determinar si existe concatenación o multiplexación para que cada TPDU se asocie a la correspondiente conexión de transporte en cada caso.

1.3.4.10. Numeración de las TPDUs DT.-

Para garantizar que ciertos procedimientos puedan actuar satisfactoriamente es necesario que para transportar cada TPDU DT contenga un **número de secuencia**, como un parámetro en el PCI. Serán números consecutivos comenzando al principio de la conexión con el cero. Se utilizan para tener un seguimiento de las mismas. Con la asignación de números de secuencia consecutivamente mayores a las TPDUs sucesivas en una conexión, es posible tener un control de flujo y de asentimientos, y habrá una forma de determinar, después de que ocurre un N_RESET, cuál fue la última TPDU que se recibió. La clase 0 (y opcionalmente, la clase 2) no utilizan números de secuencia. Los procedimientos en cuestión esta relacionados con la **recuperación, control de flujo y resecuenciamiento**. La numeración de TPDU se usa en las clases 1, 3 y 4, y en la clase 2 cuando el uso el control de flujo explícito no fue negociado durante la fase de establecimiento de conexión.

1.3.4.11. Transferencia de datos urgentes.-

Este procedimiento debe garantizar que los datos urgentes adelanten al flujo de datos normales por la red para una determinada conexión, para ello se consigue colocando las ED delante de todas las DT en la cola de envío. Nótese en las clases 2 y 3, puesto que la red garantiza entrega ordenada, esto es suficiente para que la TPDU urgente llegue lo antes posible. En la clase 4 se requiere detener el envío de DT hasta la recepción del acuse de recibo pertinente.

El procedimiento coloca los datos provistos por TS mediante la primitiva T_EXPEDITED_DATA.request en el campo de datos de una TPDU ED (Expedited Data). Aunque **el servicio de transporte de datos urgentes es no confirmado**, el protocolo de transporte demanda que el procedimiento de la entidad par sea confirmado, y así cada TPDU ED deba ser confirmada (acuse de recibo) por la entidad de transporte receptora par mediante el uso de la TPDU EA (Expedited Data Acknowlegde). Realmente solo un reconocimiento puede ser enviado (sobre la red) para cada dirección de control de flujo de la TC en cada momento.

En las clases 2, 3 y 4 las TPDUs EA y ED seran transportadas entre pares mediante el uso de datos normales de red, eso es, mediante la primitiva N_DATA.request o N_UNIT_DATA.request. En este caso, sólo se permite que haya circulando por la red una ED pendiente de confirmación de recepción. En la clase 1, o los datos de red normales pueden ser usados o, si se negocia durante el establecimiento de la conexión, datos de red urgentes (N_EXPEDITED_DATA.request).

Este procedimiento no esta disponible en clase 0.

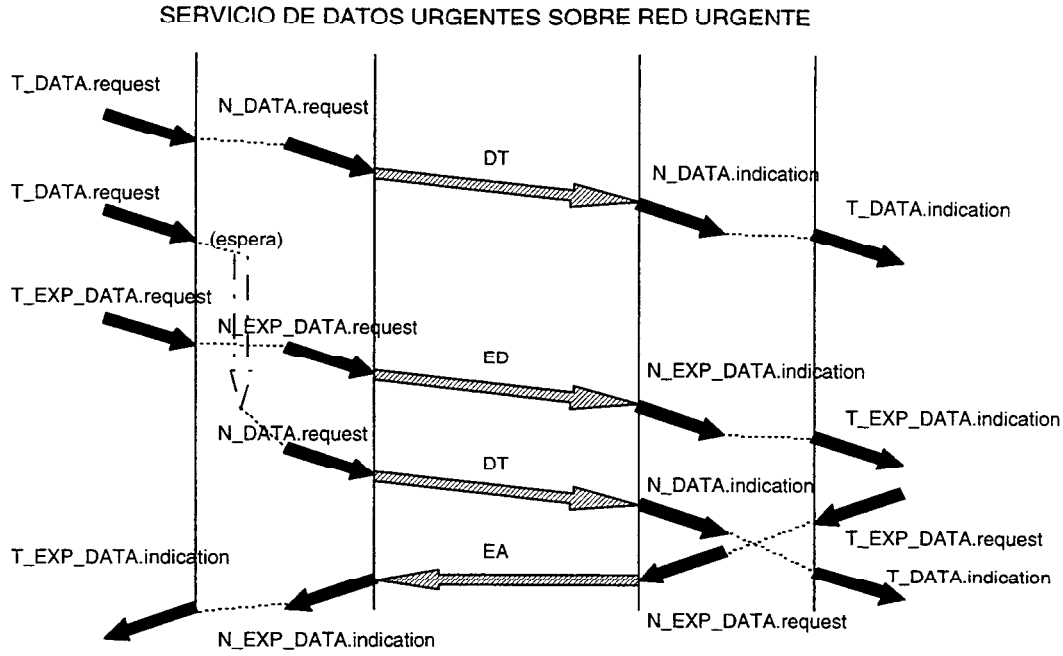


Fig. 1.44 Transferencia de TPDU ED y reconocimiento EA

Para analizar esta situación, partimos de que bajo la dirección de control de flujo, alcanzada por la entidad de transporte receptora, la entidad de transporte transmisora será capaz de transmitir datos (estado a) o bloqueado (estado b)para la transmisión de datos a través de restricciones de control de flujo.

Consideremos una entidad de transporte en el estado b. Este estado será liberado al estado a por una actividad de control de flujo iniciada por la entidad receptora. Un uso eficiente de la red para la transmisión de datos "bulk" solo será alcanzado si una entidad de transporte continua en estado b para aceptar datos pedidos desde el usuario del servicio de transporte transmisor aunque no pueda servirlos. En esta situación, cuando se entra en el estado la entidad transmisora tendrá una **cola** de peticiones de datos listos para ser enviados, así eliminando cualquier retardo inherente a la gestión de un mecanismo de control de flujo restrictivo sobre una TSAP en el sistema extremo transmisor. Algún control de flujo es necesario aquí ya que la entidad de transporte no tendrá recursos (p.e. buffers) para manejar una cola de peticiones de datos infinita; pero tal control de flujo ocurrirá en paralelo con esta entre las entidades de transporte pares en lugar de secuencialmente lo cual es ineficiente inherentemente.

Analizando la figura 1.43, una vez que ocurre el suceso (1), el sistema transmisor A, esta en el estado b. Este diagrama demuestra el simple aspecto de gestión de cola. Si ahora suponemos que justo antes del suceso (4), la liberación de control de flujo, había una petición de T_EXPEDITED_DATA.request, entonces podemos ver que, por el uso de las siguientes reglas para la cola de TSDU, el efecto urgente sería alcanzado una vez (4) ocurre :

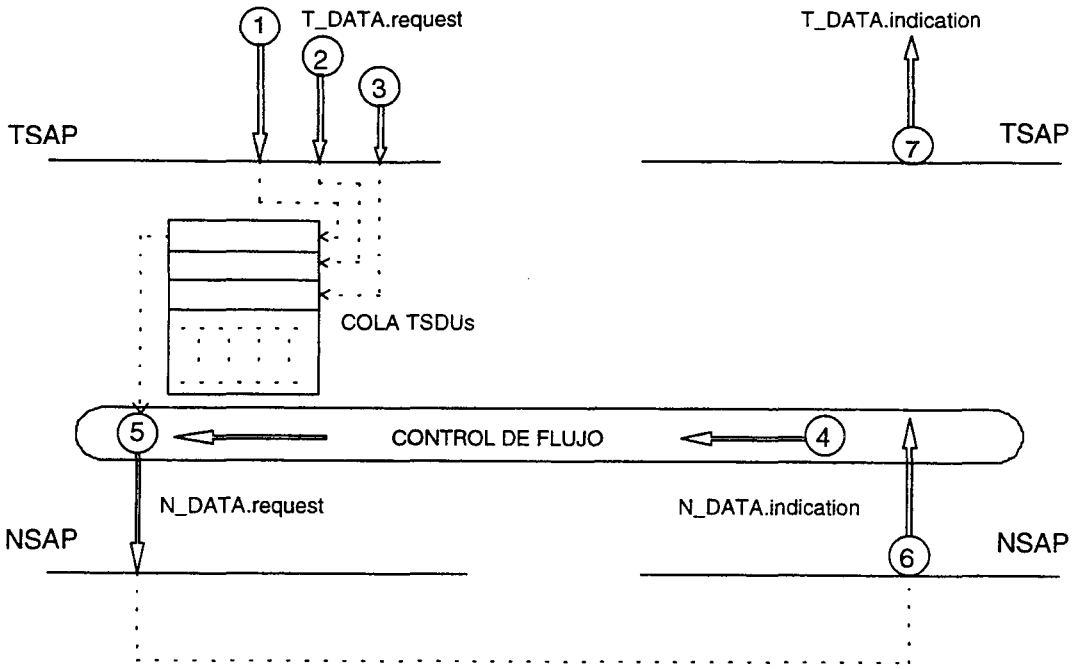


Fig. 1.43 Esquema de cola de TSDUs.

Si no hay nada sobre la cola de TSDU entonces se coloca la TSDU urgente en la cabeza. Si cualquier TSDU urgente esta ya lista sobre la cola entonces se coloca la nueva TSDU urgente después de todas las TSDUs pendientes de envio. De otra forma se coloca en la cabeza de la cola.

Esto asegura que la definición de urgencia sea atendida.

1.3.4.12. Reasignación en caso de fallo.-

Este procedimiento esta disponible en clase 1, 3 y 4. Este procedimiento actçuacuando un error señalizado de red se recibe, indicando la pérdida de la conexión de red (N_DISCONNECT.indication) a la cual una conexión de transporte esta asignada. El resultado será que la TC es asignada a otra conexión de red que ya exista y este conectada a la apropiada entidad de transporte destinataria o es nuevamente creada para este propósito. Cuando la reasignación es tratada el procedimiento de **resincronización** es invocado. Si transcurrido un cierto tiempo (parámetro de la cabecera variable), no se consigue abrir esa conexión de red, se cierra la conexión de transporte No obstante si una reasignación no fuera conseguida entonces la TC sería liberada y las **referencias de transporte congeladas**; que se analiza en el apartado 1.3.4.18. Si se esta usando multiplexación hacia abajo, o sea la asociación de esta TC con muchas NC, y la calidad del servicio requerida puede ser mantenida por las restantes NCs, entonces la reasignación no será tratada y la TC se mantendrá.

Si en lugar de reiniciar una nueva conexión, la red la rompe por completo, entonces dependerá de la capa de transporte el establecer una nueva conexión sobre la que pueda trabajar (**reasignación**)..

1.3.4.13. Retención hasta reconocimiento de TPDUs.-

Este mecanismo, soportado por las clases 1, 3 y 4 , provee a la entidad de transporte para que retenga "copias" de las TDPUs después de enviadas hasta que un **reconocimiento explícito** desde el receptor sea recibido desde la entidad par. Si los reconocimientos no se reciben al cabo de cierto tiempo (solo clase 4) o aparece un error señalizado (clases 1,3 y 4) entonces las TPDUs pueden ser retransmitidos.

El lector puede detectar ahora que estas clases, incluso aunque el servicio de transporte de datos es no confirmado, el protocolo dicta que la transferencia debe ser confirmada entre entidades de transporte pares.

En las clases 3 y 4 el reconocimiento de TPDUs es efectuado mediante la transmisión por parte de la entidad de transporte receptora de una TPDU AK (acknowledge) conocida como la **variante AK** o sea una confirmación a nivel de transporte. En esta variante la TPDU AK realiza la confirmación de la recepción de todas las TPDUs con números anteriores a la TPDU esperada, y se actualiza la concesión de crédito (ver figura 1.45).

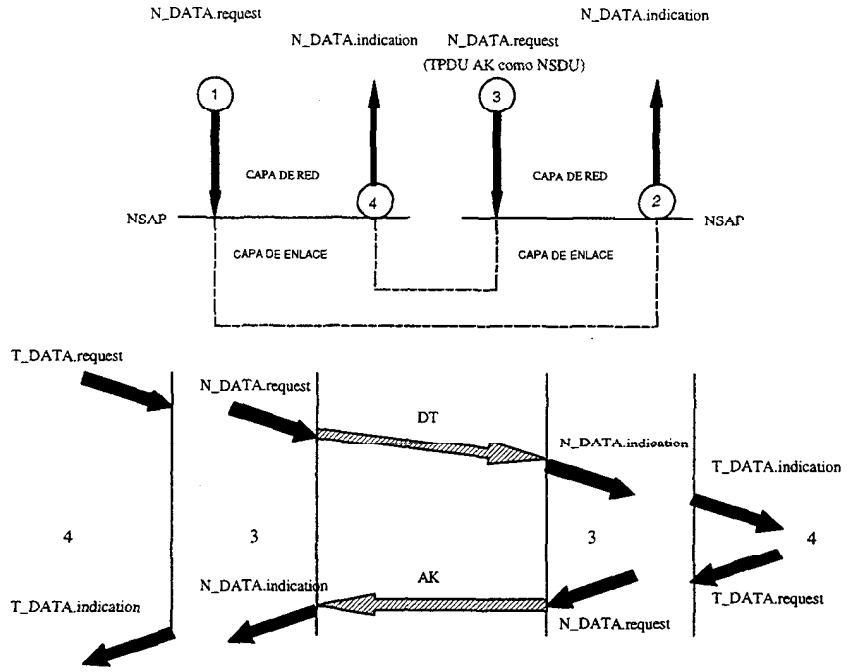


Fig. 1.45 Confirmación mediante TPDU AK

En clase 1 puede ser usada la variante AK, negociado durante el establecimiento de la conexión, mediante el elemento de servicio de red de reconocimiento de datos; este puede ser invocado por la entidad de transporte receptora (por el uso de N_DATA_ACKNOWLEDGE.request). El uso del servicio de red en este extremo es claramente dependiente de la disponibilidad de este servicio en la red inferior. Este se conoce como la **confirmación de la variante receptor**. En la figura 1.46 se muestra la variante de confirmación basada en la red.

Por supuesto la pérdida de TPDUs causará una bajada de la calidad de servicio por debajo del nivel aceptable negociado, y potencialmente, la conexión será terminada y los usuarios del servicio de transporte informados.

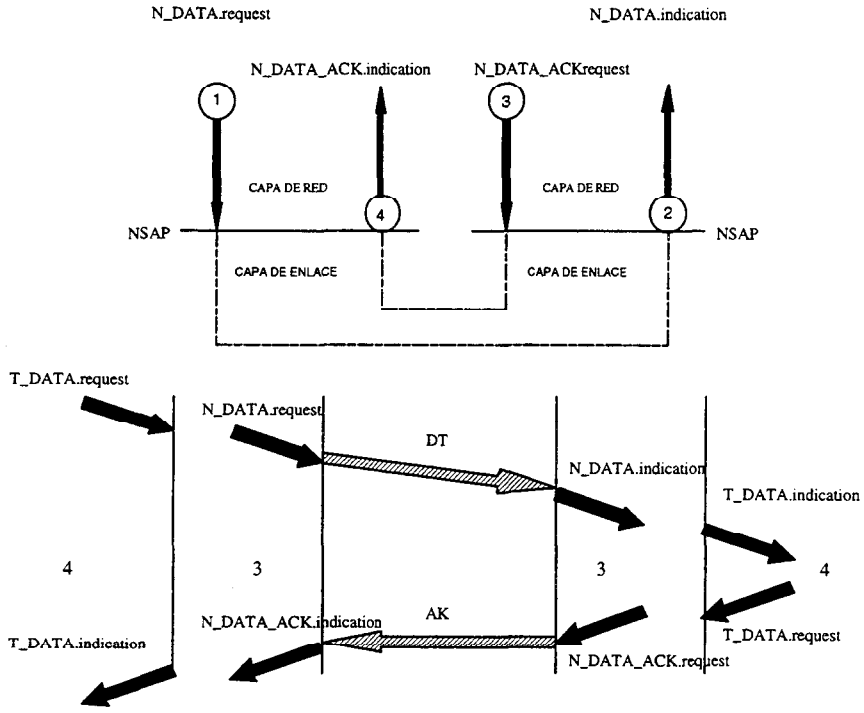


Fig. 1.46. Reconocimiento basado en la red (N_DATA_ACKNOWLEDGE).

1.3.4.14. Resincronización.-

Este es un procedimiento disponible para las clases 1, 3 y 4. Después de un suceso señalizado desde el proveedor del servicio de red que indica un problema en la conexión de red (un N_RESET.indication), o después de una reasignación de una conexión de transporte después de un fallo de conexión de red, este procedimiento se usa para restaurar la conexión de transporte a modo **"normal".**

Para ello es necesario que las entidades de transporte retengan copias de las TPDUs enviadas, hasta que se hayan asentido, de tal forma que se puedan retransmitir en el caso de que se presente un N_RESET. Dado que en la clase 0 y 2 dan una liberación por error después de la ocurrencia de un N_RESET, en vez de intentar efectuar una resincronización, éstas no tienen que retener las TPDUs hasta que sean asentidas. Ver 1.3.4.8 que expresa que en el caso de clase 0 y 2 se usa liberación por error frente a las clases 1, 3 y 4 que se usa resincronización.

La resincronización es solo tratado por la entidad de transporte que fue la iniciadora de la conexión de transporte, su entidad par solo toma una aptitud pasiva en el proceso de resincronización. El propósito de la resincronización en la entidad de transporte es

restaurar la actividad en la conexión de transporte que estaba saliente en el momento de un suceso de sincronización. Una de las entidades de transporte pares deben tomar un papel pasivo en esto, aunque ambas entidades serán conscientes de la necesidad de la resincronización , dado la naturaleza de la causa, y para ambos casi simultaneamente, y si hubiesen intentando iniciarlo ambos, resultaría una innecesaria resolución de colisión. La entidad pasiva simplemente activa un timer y espera las TPDUs de resincronización relatadas que lleguen desde el iniciador de la TC. Si expira el timer la entidad considera la TC liberada y la referencia congelada.

El protocolo define las acciones que deben ser tomadas para alcanzar una TC resincronizada en todos los casos. Este puede incluir, por ejemplo, la retransmisión de TPDUs de reconocimiento. Si la resincronización falla la TC se considera liberada y las referencias de transporte congeladas.

Como parte de resincronización la **TPDU RJ (Reject)** es transportada entre pares, como orden para resincronizar y como valor transportado de un parámetro, el valor del siguiente número de secuencia de TPDU DT esperado. Todas las anteriores se dan por recibidas.

1.3.4.15. Multiplexación y demultiplexación.-

Este procedimiento esta disponible para las clases 2, 3 y 4. Permite que mas de una TC pueda compartir una única NC. Una entidad de transporte recibe TPDUs pertenecientes a diferentes TCs sobre la misma NC. Una entidad de transporte que recibe TPDUs, en este entorno, debe desarrollar demultiplexación; la TC para la cual pertenecen las TPDUs debe invocar al procedimiento de **asociacion de TPDUs**.

Claramente, el uso de la multiplexación y concatenación, y ambos juntos donde una simple NSDU es transferida conteniendo TPDUs concatenadas para diferentes TC, hacen un uso más económico y eficiente de una red.

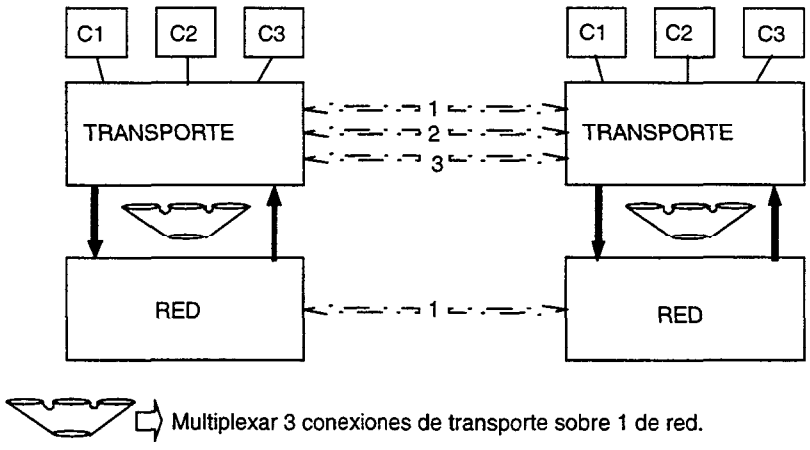


Fig. 1.47. Multiplexación ascendente.

1.3.4.16.- *Control de Flujo explícito.-*

Este es un procedimiento disponible para las clases 2, 3 y 4. El control de flujo que es controlado por la capa de transporte actua independientemente de cualquier otro control de flujo que pueda estar disponible en la red. **Regula el flujo de TPDUs DT entre entidades de transporte** par sobre una conexión de transporte. Consiste en tener una parte explícita del protocolo de transporte que se ocupe del número de TPDUs que se pueden transmitir en cualquier instante. Se puede utilizar el esquema de ventana deslizante u otros. Si no se usa un control de flujo en la capa de transporte se utilizará el control de flujo subyacente de la conexión de red o cuando la relación entre una TC y una NC es uno a uno, no hace falta que las entidades de transporte incorporen mecanismos para el control de flujo de la conexión. Si una entidad de transporte se ve desbordada porque un usuario acepta las TSDUs a un ritmo menor que el de llegada de datos del otro extremo, se limita a no aceptar paquetes de la red, o a tomarlos a un ritmo reducido. La red incorpora mecanismos para control de flujo que hacen que a la entidad de transporte remota no le sean aceptadas las entregas de nuevos paquetes. A su vez, esa entidad de transporte remota puede limitar el número de TSDUs que le entrega su usuario. Así, si uno de los elementos de la comunicación es lento, gracias a mecanismos de control de flujo de la red se consigue que todo el sistema disminuya su ritmo hasta adaptarse, sin que se produzcan problemas de falta de espacio en buffers que puedan suponer pérdida de datos.

Sin embargo, la técnica anterior no es posible cuando la red no está orientada a conexión, ni cuando existe multiplexación. En el primer caso, si no se toman paquetes de la red no se consigue nada más que desbordar los buffers internos de la red, con lo que se corre el riesgo de perder paquetes. Si hay multiplexación ascendente de varias conexiones de transporte sobre una conexión de red, y un usuario de una conexión de transporte es muy lento tomando las TSDUs que su entidad de transporte le entrega, la entidad de transporte no puede dejar de aceptar paquetes de la red, porque esos paquetes pueden pertenecer a otras conexiones de transporte en absoluto relacionadas con la que tiene problemas. La entidad de transporte puede encontrarse con un número grande de mensajes pendientes de entrega, que pueden llegar a desbordar el espacio de buffers. Es evidente que hay que encontrar una solución a este problema, y ésta pasa por hacer que disminuya el flujo de datos en la TC afectada, no en la NC que le da soporte.

Para realizar control de flujo en el nivel de transporte se pueden utilizar **técnicas similares a las que se utilizan en el nivel de enlace de datos**. Sólo similares, porque los problemas en el nivel de transporte son de una magnitud mucho mayor: mientras que en el **nivel de enlace de datos se controla el flujo** de bloques pequeños de datos a través **de una línea física**, en el de **transporte se controla el flujo** de mensajes, bastante grandes, **a través de toda una red** (que puede a su vez estar compuesta de varias redes). La principal diferencia está en los retrasos que tiene la transferencia de datos en uno y otro caso. Así, técnicas tipo **stop/wait**[1] resultan prohibitivas por el bajísimo rendimiento que se obtiene del canal de comunicaciones.

La técnica que se utiliza en los protocolos de transporte de la ISO combina el uso de **acuses de recibo** para dar más fiabilidad a la conexión con el control de flujo, aunque

[1] Esta técnica consiste en, por cada dato enviado, esperar a recibir un acuse de recibo del otro extremo.

ambos conceptos están claramente diferenciados. Para realizar esta técnica es preciso que las **TPDUs DT vayan numeradas**.

Una de la entidades de transporte (emisora) envía al otro extremo TPDU DTs numeradas secuencialmente. Inicialmente dispone de un cierto **crédito,** es decir, de un límite en el número de TPDU DTs que puede enviar. Periódicamente irá recibiendo de su par (receptora) acuses de recibo (TPDU AK), que confirman la recepción de algunas DTs (campo TPDU esperada) y renuevan la concesión de crédito (campo credit). Si la entidad de transporte receptora se ve desbordada, no tienen más que limitar el crédito que concede a la emisora, sin por ello dejar de informar de la llegada (correcta o no) de los datos.

El control de flujo explícito es opcional en la clase 2 (su uso es negociado como una parte del establecimiento de TC) y obligatorio en clases 3 y 4. El parámetro de crédito inicial se activa durante el establecimiento de la TC actua como el valor inicial para la ventana que gobierna la transmisión de TPDU DT. Esta ventana es consecuentemente manipulada por la entidad de transporte receptora para controlar el flujo desde su par. En términos generales la ventana impone un número máximo de TPDUs DT que pueden ser transmitidas sin reconocimiento explícito. Esta es una TPDU AK (acknowledge) que el receptor incluye como parámetro adelantando la ventana en el transmisor. Una entidad de transporte puede, en cualquier momento, enviar una TPDU RJ a su par para renovar la ventana.

Claramente la imposición de control de flujo de transporte y la eficiencia con la que la entidad de transporte transmisora responde para ajustar la ventana puede tener un efecto significativo sobre throughput.

1.3.4.17. Checksum.-

Este procedimiento es solo usado para clase 4 y es opcional.

El checksum es un valor calculado de acuerdo a un algoritmo definido en la especificación del protocolo el cual toma los bytes de la TPDU a los cuales se le asocia como su argumento. Se usa un código de redundancia por software basado en Fletcher 1982. Este checksum es transportado en la TPDU en un parámetro checksum. Después de la transmisión sobre la red el checksum es recalculado (por la entidad de transporte receptora) y comparado con el valor en el parámetro de la TPDU. Si los valores difieren entonces la **TPDU esta deteriorada y será descartada** y **no se devuelve un reconocimiento**. Esto provocará en la entidad de transporte transmisora retransmitir la TPDU, después de un tiempo especial. (ver apartado 1.3.4.19).

Aunque el checksum es opcional en esta clase, siempre se usa sobre la TPDU CR.

Descripción mucho más detallada se realizó en el apartado 1.3.3.2.3.

1.3.4.18. Referencias Congeladas.-

Este es un procedimiento usado por las clases 1, 3 y 4 para asegurar que **una referencia** - la información relacionada para identificar una TC - **no sea reasignado a otra TC** después de ser **"congelada"**. Una referencia es congelada porque las secuencias, como la siguiente, puede aparecer.

Una NC tiene muchas TCs asignadas (multiplexadas) cuando una TC de clase 4 es liberada por el receptor. Al mismo tiempo el emisor retransmite TPDUs de reconocimiento (usando el procedimiento de retransmisión desde de temporizador). Estas son entregados por el proveedor del NS a la entidad de transporte par para liberación. Si en este momento, la entidad reusará la referencia entre la liberación y la recepción de estas TPDUs entonces serían incorrectamente asociados con la nueva TC y así causa problemas.

Referencias son congeladas por un intervalo de tiempo considerable (determinado por la implementación) para evitar tales problemas y evitar de que se confundan viejas TPDUs con nuevas.

1.3.4.19. Retransmisión con Temporizador.-

Este procedimiento usado solo por la clase 4 para proveer retransmisión por el emisor de TPDUs para que estas, en el caso de haberse perdido, puedan alcanzar el destino sin necesidad de señalización. Solo se especifica en esta clase porque es la única en la que la pérdida de paquetes es lo suficientemente común como para requerir un control de error en la capa de transporte.

La entidad de transporte transmisora detecta la pérdida de TPDUs (p.e. no entregada) cuando **no recibe un reconocimiento durante el intervalo de tiempo** predeterminado y cuando reconocimientos son conocidos como salientes. En tal caso, la primera TPDU en la secuencia (ver el procedimiento de retención hasta reconocimiento) de TPDUs de reconocimiento es retransmitido y el temporizador se resetea y deja pasar. Si los reconocimientos son luego recibidos, entonces, quizás todo ha ido bien. No obstante, después de un cierto número de retransmisión sin reconocimientos la entidad de transporte transmisora invocará el procedimiento de liberación e informa al usuario del TS del fallo (ver figura 1.48).

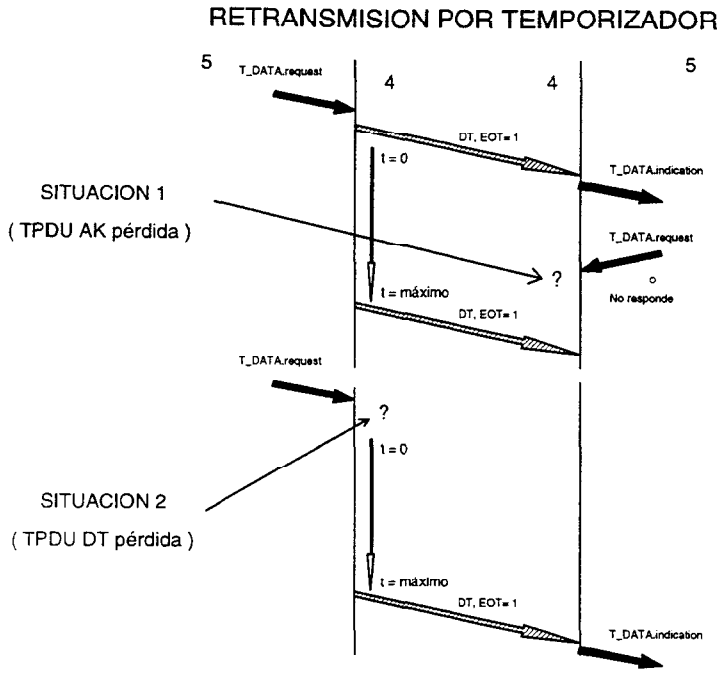


Fig. 1.48. Uso de temporizadores para retransmisión ante pérdidas.

1.3.4.20. Resecuenciamento.-

Usado solamente por la clase 4, este procedimiento se usa para **ordenar** cualquier cantidad de TPDUs DT que lleguen desordenadas por el proveedor NS. Esto se realiza para garantizar que cada TSDU entregada al TS_user tendrá todos los bytes correctamente ordenados a pesar de que la red inferior puede causar TPDUs fuera de orden. Tal desorden puede ocurrir cuando, la TSDU es segmentada por la entidad de transporte transmisora dentro de muchas TPDUs o donde como resultado de multiplexación descendente de estas TPDUs que viajan entre sistemas-finales se despliegan sobre un determinado número de conexiones de red. La posibilidad de que una TPDU llegue en otro orden al de su transmisión en este caso es clara.

Este procedimiento también **detecta cualquier TPDU DT duplicada**, y descarta tales TPDUs duplicadas. La duplicación puede aparecer, por ejemplo, cuando hay una elevada congestión sobre la red posibilitando que tras el vencimiento de temporizadores se transmitan las TPDUs por las entidades de transporte. Esto provoca que se retransmitiría una TPDU DT aunque todavía la TPDU DT original este en tránsito y pueden ser ambas entregadas.

Relacionado con este mecanismo está también el de los **números de subsecuencia**. Estos números se utilizan para numerar secuencialmente las TPDUs AK en el protocolo de clase 4, con el objetivo de que el otro extremo las vaya procesando en orden.

Veamos una situación : Supongamos que una entidad de transporte, a la que llamaremos receptora, envía a su par, o emisora, una AK(8,9), donde 8 es la próxima DT que se

espera y 9 es la concesión de crédito. Muy poco después, envía otra AK(8,3): ha reducido drásticamente el crédito porque, por cualquier motivo, la primera previsión ha sido demasiado optimista.

Puesto que en general, la red no garantiza entrega ordenada, puede ocurrir que llegue primero la AK(8,3). El entidad de transporte emisora de datos tendrá en cuenta esta asignación de crédito, luego no habrá problemas por parte de la receptora. Sí, posteriormente, llega AK(8,9), la entidad de transporte emisora de datos entiende que se le ha ampliado el crédito, por lo que envía 6 DTs más que, probablemente, no podrán ser aceptadas por el entidad de transporte receptora, puesto que no dispone de espacio suficiente (por eso se redujo el crédito).

Si se decide numerar secuencialmente las AK, y procesarlas en orden, no existirá este problema: aunque se reciba AK(8,3), no se procesará hasta que no se haya recibido el anterior AK(8,9). Las modificaciones de crédito se producen en orden, y quedará un crédito de 3, sin dar lugar a confusiones.

Conjuntamente con este mecanismo, cada vez que se envía una AK al otro extremo, puede incluir como parámetro la confirmación de control de flujo, copia de parte de la información recibida en la última AK del otro extremo. Así, cada entidad de transporte puede saber si su par le está controlando el flujo de manera correcta.

1.3.4.21. Control de inactividad.-

Usado solamente en la clase 4, este procedimiento negocia una terminación no señalizada de una conexión de red (no envíe N_DISCONNECT.indication). Es invocado cuando **acaba el temporizador de inactividad** mantenido por la entidad de transporte. Este tiempo vendrá determinado la **ausencia de recepción de ninguna TPDU**. Expira después de un intervalo lo suficientemente largo y luego invoca el procedimiento de liberación normal. Este temporizador debe tener un valor calculado de forma adecuada, ya que si fija un valor demasiado pequeño, es posible que se cierren conexiones de transporte activas, ya que puede existir congestión. (ver figura 1.49).

Cuando no hay actividad se puede prevenir esta situación mediante la transmisión de ocasionales o regulares TPDU AK.

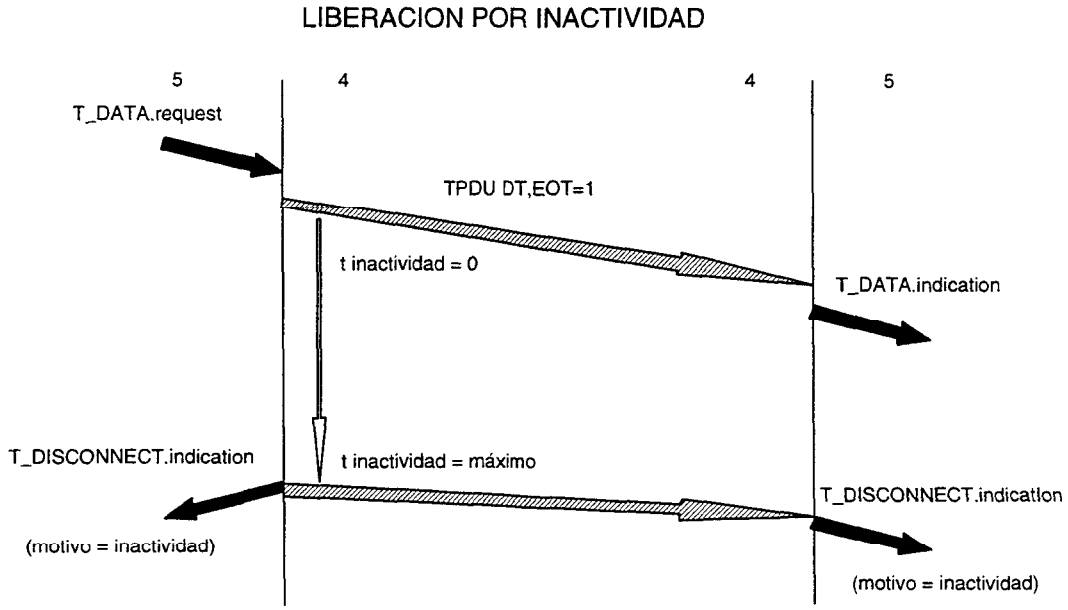


Fig. 1.49. Uso de Temporizador para liberar por inactividad.

1.3.4.22. Tratamiento de errores de protocolo.-

Este procedimiento, usado en todas las clases, se utiliza cuando una TPDU se recibe y no puede ser interpretada según las reglas definidas en el estándar cuando no son errores señalizados o cuando el checksum no es correcto. La entidad de transporte que detecta la TPDU errónea tomará una de las siguientes acciones :

a) Usar una **TPDU de Error** (TPDU ER) que provocará la activación, en la entidad de transporte transmisora, del apropiado procedimiento de liberación de la conexión de transporte para la clase,

b) Inmediatamente, invocar al apropiado procedimiento de liberación para la clase,

c) Resetear (Resincronizar) o cerrar la conexión de red asociada.

La apropiada acción se selecciona por las entidades según sus modos de operación. Por ejemplo, si la conexión de red es compartida por otras conexiones de transporte entonces sería inadecuado bloquear las otras conexiones de transporte. No obstante, si no puede asociar la TPDU con una determinada conexión de transporte entonces no queda otro remedio que tomar esta decisión. Bajo ciertas circunstancias es posible ignorar la TPDU. No deberían haber errores de protocolo.

1.3.4.23. Multiplezación Descendente y Recombinación.-

Este es un procedimiento usado solo por la clase 4 para habilitar a un conexión de transporte poder usar **múltiples conexiones de red**. Como resultado puede incrementarse el throughput (para encontrar el apropiado nivel de servicio requerido) o

puede haber una mayor elasticidad (resistencia) frente a fallos en redes inseguras, o ambas. Una vez una asosiación existe entre una conexión de transporte y muchas conexiones de red, las TPDUs de esa conexión de transporte puede ser transmitida sobre cualquiera de las conexiones de red, y así las TPDUs pueden llegar a la entidad de transporte par fuera de secuencia.

El establecimiento de conexión de transporte inicial sobre una simple conexión de red tendrá que haberse establecido un servicio de transporte de clase 4 - sin la cual la multiplexación descendente no es posible. Después de esto, cada entidad de transporte puede asignar la conexión de transporte a una nueva, o una conexión de red existente, en el caso de la detección de un fallo en el throughput. Una vez que se tienen varias conexiones de red asignadas, la entidad de transporte puede enviar las TPDUs DT por cualquiera de ellas. En estas circunstancias la red no garantiza que las TPDUs lleguen en el mismo orden en que fueron emitidas, por lo que se hace necesario el uso del mecanismo de resecuenciamiento.

1.3.5 Calidad de Servicio.-

Este parámetro permite negociar una serie de características relacionados con la calidad de la conexión de transporte, que servirán para toda la vida de la misma. El proceso de negociación tiene una forma descendente.

La calidad de servicio de una entidad de transporte puede ser expresada como un conjunto de **criterios de prestaciones**. Estos criterios se pueden dividir :

- criterios de velocidad y
- criterios de seguridad/fiabilidad.

Cada uno de las parámetros se detallan a continuación :

Retardo en el establecimiento de conexión de transporte.-

Este parámetro pone un límite al máximo tiempo permitido entre la correspondiente petición de conexión de sesión y la correspondiente confirmación. Si el valor excede este límite la entidad de sesión iniciadora de la conexión fallará para ese intento de conexión, informando al usuario del servicio de sesión del fallo, y "limpiarla"- lo cual puede involucrar el usar la T_DISCONNECT.request si la conexión de transporte ha sido establecida.

Probabilidad de fallo de establecimiento de conexión.-

Este parámetro es un límite máximo para la tasa de fallos (o el tipo de retardo) para establecer una conexión sobre el número total de peticiones sobre una muestra realizada. Si este "ratio" es mayor que el valor del parámetro entonces al entidad de sesión no intentará establecer conexión. La liberación de la conexión de sesión tiene un significado similar.

Throughput (Velocidad de Transferencia).-

Da la tasa mínima aceptable de transferencia de bytes (SSDUs) entre usuarios del servicio de sesión sobre la conexión de sesión. Se especifica para cada dirección del flujo, iniciador a receptor y viceversa, y se relaciona solo con datos normales y tipados (ver capa de sesión). Esta figura es negociada entre los usuarios del servicio de sesión y quizás, modificado por el proveedor del servicio de sesión durante el establecimiento de conexión.

Retardo de Tránsito.-

Da el retardo máximo aceptable, durante la fase de transferencia de datos, entre el atender el servicio por parte de la entidad de sesión de una petición de servicio y el uso de la correspondiente indicación de la entidad par. Se especifica en ambas direcciones de la actividad, el cálculo esta basado sobre algún tamaño promedio de datos de usuario de sesión asociado con cada par de primitivas del servicio.

Tasa o Proporción de errores residuales.-

Da el aceptable "ratio" total de perdidas, incorrectas y duplicadas unidades de datos del usuario del servicio de sesión sobre el total de unidades recibidas sobre el SSAP sobre un período medido en la fase de transferencia de datos.

El retardo de tránsito, throughput o tasa de errores residuales caen por debajo de niveles aceptables dan como resultado un fallo de la transferencia y una indicación de fallo será dada al usuario del servicio de sesión (un S_P_ABORT o S_P_EXCEPTION_REPORT.indication).

Probabilidad de fallo en la transferencia.-

Da la probabilidad de máximo aceptable del total de fallo del proveedor del servicio de sesión. Si la tasa actual de fallos observada sobre una muestra medidas mayor que el valor de este parámetro de calidad de servicio entonces la petición de establecimiento de conexión por el usuario del servicio de sesión no se intentará. Una transferencia, para propositos de muestra, puede ser considerado generalmente como la duración de una conexión de sesión individual, y un fallo de una transferencia como una S_P_ABORT o S_P_EXCEPTION_REPORT.indication que resulta desde uno de los últimos tres criterios estan siendo infringidos.

Elasticidad(resitencia) de conexión de transporte.-

Es una probabilidad de un máximo aceptable de fallo total del proveedor del servicio de sesión por eventos iniciados no por el proveedor del servicio de sesión, eso es, resultando indicaciones S_P_ABORT o S_P_EXCEPTION_REPORT, incluyendo los anteriores fallos de transferencia relatados. Otra vez, si una muestra medida indica una mayor probabilidad entonces la coneción de sesión no se intenta.

Retardo en la liberación de conexión de transporte.-

Da el retardo máximo aceptable durante la fase de liberación de la conexión.

Probabilidad de fallo de liberación de la conexión.-

Este parámetro es un límite máximo para la tasa de fallos (o el tipo de retardo) para liberar una conexión sobre el número total de peticiones sobre una muestra realizada.

Prioridad de Conexión.-

Esta involucrada con la importancia de una conexión frente a otra y es una medida de su importancia relativa. Si, por alguna necesidad, la QOS general de una conexión se degrada, por ejemplo, por una pérdida de una conexión de red, entonces serán las conexiones de menor prioridad las que sufran los efectos primero. Si la conexión debe ser abortada para mantener el nivel del servicio general, según esta prioridad que determina cual será rota.

Protección de Conexión.-

Es un parámetro de seguridad que permita que una conexión tenga un cierto nivel de protección contra monitorización o manipulación. Hay cuatro niveles : sin protección; protección contra monitorización pasiva, protección contra monitorización, repetición, adición o borrado; y combinación de las dos últimas.

La entidad de transporte determina parámetros de QOS equivalentes por el proveedor del servicio de red basado en su experiencia y capacidad, junto con los valores de los parámetros incluidos en la petición de conexión de transporte. Lo mismo sucederá en la N_CONNECT (redes orientadas a conexión) o con N_UNIT_DATA.request (redes con orientadas a conexión) en sus parámetros de calidad de servicio.

A continuación se muestra la fase donde se gestiona cada uno de estos parámetros :

FASE	VELOCIDAD	SEGURIDAD/FIABILIDAD
Establec. de Conexión	Retardo de Establecimiento	Probab. de fallo de establec.
Liberación de Conexión	Retardo de Liberación	Probab. de fallo de liberación
Transferencia de Datos	Throughput Retardo de Tránsito	Tasa de errores residuales Elasticidad de conexión Probab.de fallo en transferen.

1.4. Gestión de Conexiones.-

1.4.1 Direccionamiento

Como ya se comento anteriormente, las conexiones de transporte se establecen mediante la definición y uso de los puntos de acceso al servicio de transporte TSAPs, a los cuales puedan unirse los procesos de los usuarios y esperar que llegue alguna solicitud de conexión. Estos TSAP son análogos a los NSAP de la capa de red, por lo que la interrelación entre TSAPs y NSAPs se definen los puntos finales de conexión de cada capa (ver figura 1.50). En ella vemos que un proceso servidor entrega un servicio en la máquina B a través del TSAP número 555 esperando peticiones mediante T_CONNECT.indication. Este número y como se enlaza con los procesos no se define en OSI y se deberá gestionar por el S.O. local. Si un proceso en la máquina A solicita el servicio de la máquina B deberá especificar un TSAP origen, p.e. el número 222 mediante la primitiva T_CONNECT.request con el TSAP 222 de origen y TSAP 555 destino (recuérdese parte variable de TPDU CR y CC y no confundir con referencias fuente y destino de parte fija). La entidad de transporte local selecciona un NSAP local y otro remoto para establecer la conexión de red subyacente, necesaria para soportar la conexión de transporte. Una vez establecida esta primera fase, el servidor aceptará o rechazará la conexión, con las correspondientes primitivas del servicio.

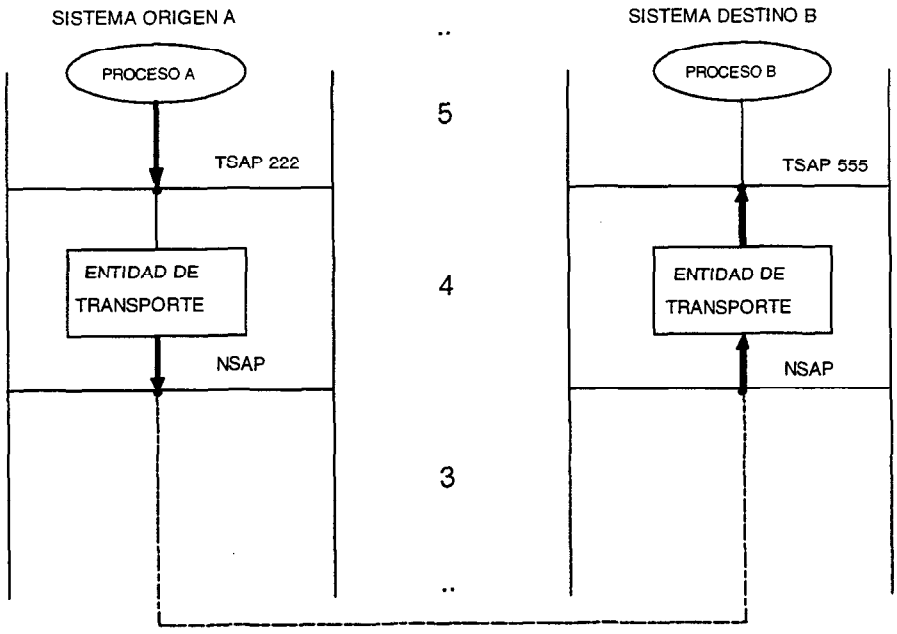


Fig. 1.50 Uso de TSAP y NSAP para establecer conexiones de transporte

Resulta importante destacar un hecho, y es que hemos supuesto que la máquina cliente conoce el TSAP destino. O bien es un número predefinido e invariable durante el uso de los sistemas o este mecanismo tiene cuando menos una premisa no definida. Como determinar los TSAP, que no sean fijos para optimizar su uso y que los posibles clientes los conozcan y puedan actualizarlos es un cuestión muy interesante.

Para gestionar esta situación, se plantea un mecanismo definido por **arpanet** denominado **protocolo de conexión inicial**. Consiste en que en lugar de tener cada servicio escuchando por un TSAP específico y bien conocido e invariable para cada uno de ellos, se dispondrá de un **servidor de procesos** a través del cual se hacen las solicitudes a cada uno de los servicios deseados. Este servidor estará escuchando por un TSAP bien conocido. Los usuarios de cualquier servicio deberán comenzar por hacer un T_CONNECT.request, especificando la dirección del TSAP del servidor de procesos. Una vez que la conexión quedó establecida, el usuario transmite un mensaje al servidor de procesos, indicándole el programa que desea usar. Entonces el servidor de procesos selecciona un TSAP inactivo y crea un nuevo proceso, indicándole a éste que escuche en el TSAP seleccionado. Por último, el servidor de procesos envía al usuario remoto la dirección del TSAP seleccionado, después termina la conexión, y vuelve a seguir escuchando nuevamente en su bien conocido TSAP.

Llegados a este punto, el proceso nuevo estará escuchando en un TSAP que ya conoce el usuario, por lo que éste puede liberar la conexión que lo une al servidor de procesos y conectarse al proceso nuevo. Una vez que se llega a establecer esta conexión, el proceso nuevo ejecuta el programa deseado cuyo nombre recibió del servidor de procesos, junto con la dirección TSAP que tenía que escuchar. Cuando el servidor termina de ejecutar su trabajo, liberará la conexión y terminará también. Todo este proceso se muestra en la figura 1.51.

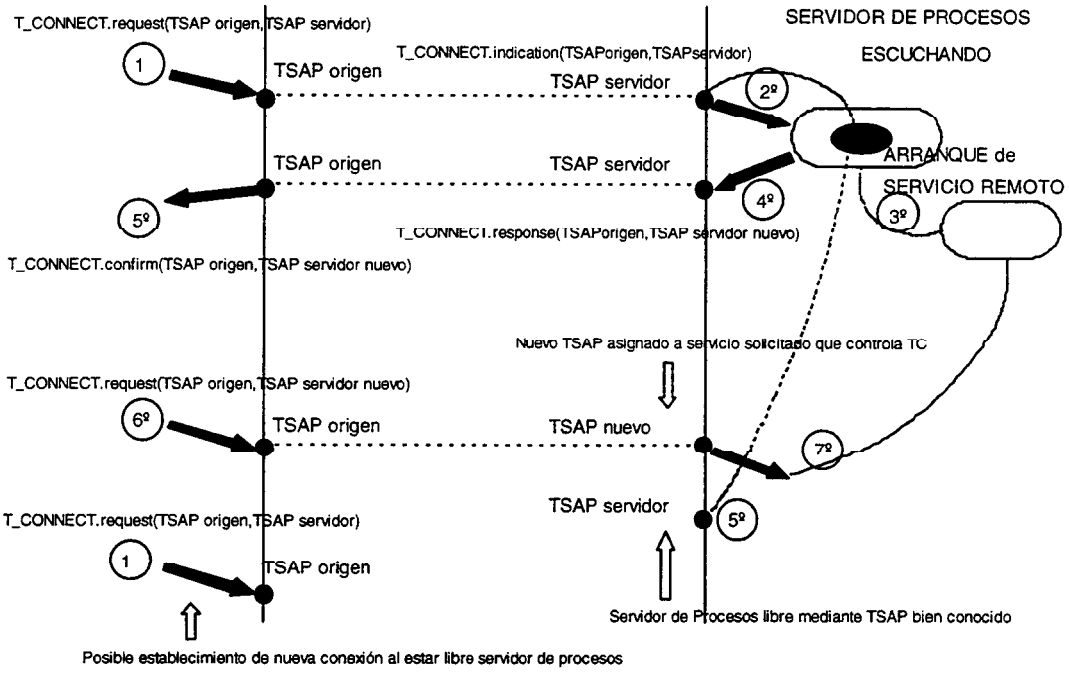


Fig. 1.51 Secuencia de acceso a servidor de procesos y servicio solicitado

De manera más general se suelen especificar **servidores de nombres o servidores de directorio** para localizar (los clientes accediendo a los TSAPs de estos servidores bien conocidos) y asignar (los servidores devolviendo los nuevos TSAPs y procesos asociados) los TSAP de los diferentes servicios disponibles en los diferentes servidores. Otra pregunta interesante es como sabrá la entidad de transporte local que NSAP debe usar para acceder a un determinado servicio o en que máquina esta ese servicio. La respuesta depende de la estructura de las direcciones TSAP. Una posible estructura es que las direcciones de los TSAPs sean **direcciones jerárquicas**. En las direcciones

jerárquicas, la dirección consta de una serie de campos utilizados para dividir separadamente el espacio de direcciones. Por ejemplo, una dirección jerárquica universal tendría una estructura de forma :

dirección : <galaxia><estrella><planeta><país><red><host><puerto>

De esta forma si tendremos la asignación exacta y universal. De la misma forma, si la dirección de un TSAP es una concatenación de una dirección NSAP y un puerto (un identificador local especificando uno de los TSAP locales), entonces, cuando una entidad de transporte se le da una dirección de TSAP a la cual se debe conectar, ésta utiliza la dirección NSAP que está contenida en la dirección TSAP, con objeto de alcanzar la entidad de transporte remota apropiada. Un ejemplo de esta clase de direcciones es el número telefónico común, p.e. 928123456 especifica provincia (928) y abonado (123456).

La otra posibilidad es utilizar un espacio de **direcciones horizontal o plano**, en cuyo caso es necesario establecer un segundo nivel de correlación para localizar a la máquina apropiada. Ahí tendría que encontrarse un servidor de nombres que tomara las direcciones TSAP como entradas y devolviera direcciones NSAP como salidas. De manera alternativa, en algunas situaciones, podría caber la posibilidad de **difundir una pregunta**, pidiéndoles atentamente a la máquina destino que por favor se identifique.

1.4.2 Establecimiento de Conexión.-

Para controlar las duplicidades, pérdidas o retardos en las TPDUs se puede hacer de la siguiente forma :

1) Usar TSAP desechables. Cada vez que se necesita un TSAP, se genera una nueva dirección única, que típicamente se basa en el tiempo actual. En el momento en que se libera una conexión, las direcciones se desechan definitivamente. Esta estrategia hace imposible el modelo de servidor de procesos.

2) Consiste en darle a cada conexión un identificador de conexión (por ejemplo, un número de secuencia incrementado durante cada establecimiento de conexión), seleccionado por la parte iniciadora, y puesto en cada una de las TPDU incluyendo la que está solicitando la conexión. Después de liberar cada conexión, cada entidad de transporte podría actualizar una tabla listando las conexiones obsoletas por parejas (la entidad de transporte corresponsal y el identificador de conexión). Cuando una solicitud de conexión llegará, podría ser corroborada en la tabla, para saber si pertenece a una conexión previamente liberada. El **problema** en este caso que cada entidad de transporte tiene que mantener cierta cantidad de información histórica en forma indefinida. Si una máquina fallara y perdiera su memoria, jamás podría saber que identificadores de conexión fueron utilizados ya previamente.

Plan de Acción : En lugar de permitir que los paquetes vivan indefinidamente dentro de la subred, deberemos diseñar un mecanismo que aniquile los paquetes más antiguos que todavía se encuentren dando vueltas. Si pudiéramos asegurar que un paquete no viviera más allá de un tiempo establecido, el problema llegaría a ser, de alguna manera, más manejable.

El tiempo de vida de un paquete puede restringirse a un máximo conocido por medio de las siguientes técnicas:

a) Diseño restringido de la subred
b) Colocación de un contador de saltos en cada paquete.
c) Sellado de cada paquete con un estampilla de tiempo.

El primer método a) incluye a cualquiera de los métodos que evitan que los paquetes entren en un ciclo, combinado con alguna manera de limitar el retardo por congestión sobre la trayectoria más larga posible (ahora conocida).
El segundo método b) consiste en tener un contador de saltos que se incremente cada vez que un IMP reexpide un paquete. El protocolo de enlace, simplemente descarta cualquier paquete cuya cuenta de saltos haya excedido cierto valor.

En el tercer método c) se necesita que cada paquete lleve consigo el tiempo en que fue creado, con las IMP aceptando descartar cualquier paquete más antiguo que cierto tiempo acordado previamente.

Este último método requiere que los relojes de las IMP estén sincronizados, lo cual viene a ser una tarea nada trivial, a menos que la sincronización se logre fuera de la red, por ejemplo, escuchando la difusión periódica de la hora exacta de alguna estación de radio.

En la práctica, no sólo se necesitará garantizar que el paquete esté efectivamente muerto, sino que también los asentimientos están muertos, así que introducimos T, que es múltiplo pequeño del verdadero tiempo de vida máximo del paquete. El múltiplo depende del protocolo y simplemente tiene el efecto de hacer más grande el valor de T. Si esperásemos un tiempo T, después del envio de un paquete, podríamos asegurar que todas sus huellas se habrán desvanecido y, que, ni él ni sus asentimientos, surgirán súbitamente de la nada para complicar la situación.

En 1975 Tomlinson diseño un método que resolvía el problema y mejorados por SunSHine y Dalal en 1978. Para darle la vuelta al problema de una máquina que pierde la memoria se encontraba después del fallo, Tomlinson propuso equipar a cada host, con un reloj indicando la hora del día. Los relojes ubicados en diferentes hosts no necesitan estar sincronizados. Se supone que cada reloj toma la forma de un contador binario que se incrementa a intervalos uniformes. Además, el número de bits en el contador deberá ser igual o mayor que el número de bits de los números de secuencia. Por último, y muy importante, se supone que el reloj continuá funcionando aún cuando el host se desactive.

La idea básica es la asegurar que dos TPDU idénticamente numeradas nunca estén presentes al mismo tiempo. Cuando es establece una conexión, los k bits de menor orden del reloj se utilizan como número de secuencia inicial (que también es de k bits). Por lo tanto, a diferencia de otros protocolos, aquí cada conexión comienza a numerar a sus TPDU con un número de secuencia diferente. El espacio de la secuencia deberá tener un longitud tal (por ejemplo 32 bits p.e TCP) que, para cuando los números de secuencia se repitan, ya las antiguas TPDU con el mismo número de secuencia habrán

desaparecido desde hace tiempo. Esta relación lineal entre el tiempo y los números de secuencia inicial se observa en la figura 1.52.

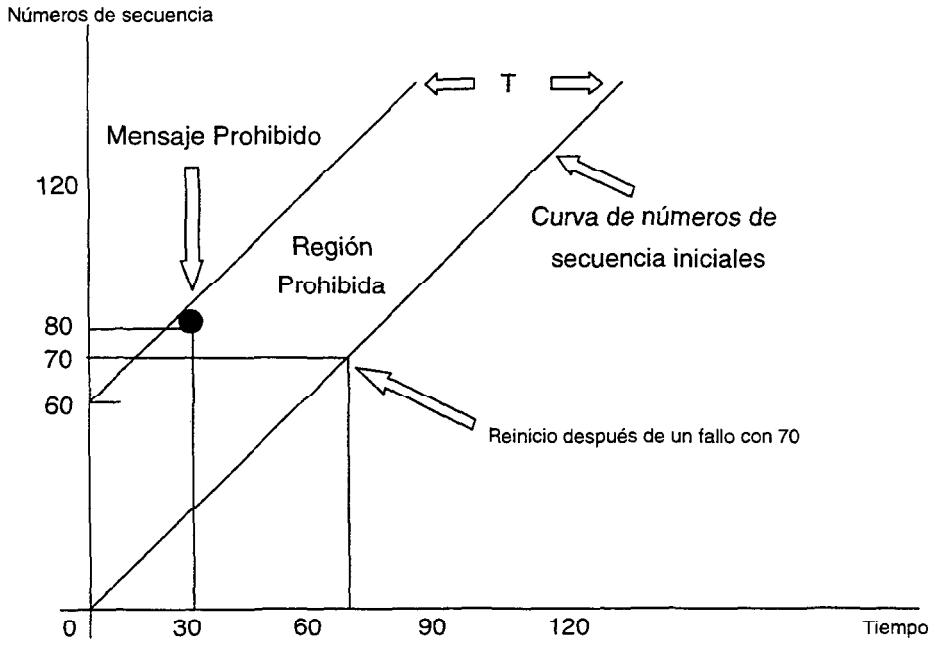


Fig. 1.52 Región Prohibida

Una vez que las entidades de transporte hayan llegado a un acuerdo sobre el número de secuencia inicial, cualquier protocolo de ventana deslizante puede utilizarse para el control de flujo de datos. Ocurre un **problema** cuando un host falla. Cuando vuelve a funcionar, una entidad de transporte no sabrá en que lugar se encontraba dentro del espacio de la secuencia. Una posible solución es la de exigir que las entidades de transporte estén inactivas durante un período de T segundos posteriores a la recuperación, para permitir que las TPDUs antiguas se mueran. Sin embargo, en el caso de una interconexión de redes complicada, el valor de T puede ser muy grande, así que esta estrategia no es muy atractiva.

Para evitar la necesidad de requerir un tiempo muerto de T segundos después de un fallo, será necesario introducir una nueva restricción en el uso de números de secuencia. Por medio de un ejemplo será más fácil ver la necesidad de esta restricción. Considérese que T, el tiempo de vida máximo de los paquetes es de 60 segundos, y que el tic-tac del reloj se presenta una vez por segundo. Como se mostró en la figura anterior, el número de secuencia inicial para una conexión abierta en el tiempo x, será x. Imagínese que en el momento t=30 segundos, a una TPDU de datos ordinaria enviada sobre la conexión 5 (que previamente se abrió), se le asignó el número de secuencia 80. Llámase a ella TPDU X. Después de enviar la TPDU X, al host falla e, inmediatamente, se reestablece. En t=60 segundos, comienza a reabrir las conexiones 0 a 4. En t=70seg, reabre la conexión 5, utilizando el número de secuencia inicial 70, tal y como se requería. Dentro de los siguientes 15 segundos, envía las TPDU de datos del 70 al 80. Entonces, en t=85seg se introduce a la subred una nueva TPDU con el número de secuencia 80 a través de la conexión 5. Desafortunadamente, la TPDU X todavía existe. Si pudiese llegar al receptor antes de que lo haga el TPDU 80, sería aceptada, y la TPDU correcta se rechazaría.

Con objeto de evitar que sucedan estos problemas, se deberá impedir que los números de secuencia sean utilizados (es decir, asignados a nuevos TPDU) durante un tiempo T, antes de poder volver a ser usados como números de secuencia inicial. En la figura se representan las combinaciones ilegales de tiempos y números de secuencia, como **región prohibida.** Antes de que se envíe alguna TPDU a cualquier conexión, la entidad de transporte deberá leer el reloj y verificar si no se encuentra en la región prohibida.

El protocolo se puede meter en problemas de dos maneras diferentes. Si un host envía demasiados datos, y demasiado rápido, sobre una conexión recién creada, la curva real de números de secuencia contra el tiempo puede tener una pendiente más pronunciada que la curva del número de secuencia inicial contra el tiempo. Esto significa que la máxima velocidad de datos en cualquier conexión es de una TPDU por tic-tac de reloj. También significa que, la entidad de transporte deberá esperar hasta que el reloj haga tic-tac, antes de abrir una nueva conexión después de un restablecimiento tras caída, de tal manera que no se utilice el mismo número dos veces. Estos dos puntos conducen a la necesidad de tener tic-tacs de reloj de corta duración (es decir, de algunos milisegundos). Esto resuelve el problema para TPDUs de datos pero habra que establecer la conexión primero.

⇨ El protocolo **ida-vuelta-ida (Tomlinson 1975)** resuelve el problema de establecer la conexión. Este método no requiere sincronizarse con el mismo método. La técnica consiste en que la entidad A selecciona un número de secuencia x por ejemplo, y lo envía a B; el cual contesta con una TPDU CC asintiendo a x y anunciando su propio número de secuencia inicial, y (el cual, por supuesto, puede ser igual a x). Por último, A asiente la elección que hizo B, del número de secuencia inicial en su primera TPDU de datos.

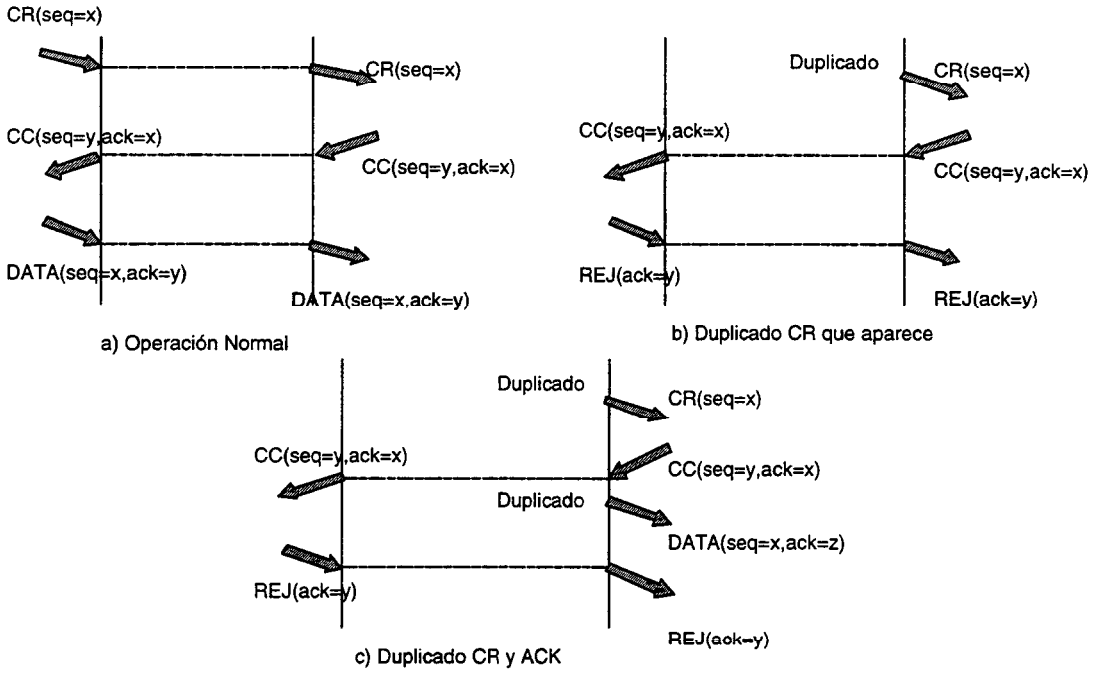


Fig. 1.53 Situaciones ida-vuelta-ida

En el caso de TPDU duplicadas o retardadas vemos en la figura que en el caso b) al llegar a B una duplicado retardado de CR, sin que A se llegue a enterar, B reacciona mediante la transmisión de una TPDU CC para A, a través del cual pide que efectivamente se verifique que estaba en verdad tratando de establecer una nueva conexión. En el momento en que A rechaza el intento de establecimiento de B, éste comprende que fue engañado por un duplicado retardado y abandona la conexión.

El peor caso, se tiene cuando tanto un CR retardado como un asentimiento para un CC se encuentran flotando dentro de la subred, ver sección c de figura 1.53. Al responder B intenta utilizar a y como el número de secuencia inicial para el tráfico que va de B hacia A, sabiendo muy bien que todavía no existe ninguna TPDU que contenga un número de secuencia y, o un asentimiento para y. Cuando llega B la segunda TPDU retardada, el hecho de que z haya sido asentido en lugar de y, le indica claramente a B que éste también, es un duplicado antiguo.

1.4.3 Liberaciones de Conexiones.-

Para la liberación de una conexión la situación es mucho más compleja pues puede darse una situación como la siguiente si no se garantiza la correcta petición y respuesta de una desconexión de transporte. En la figura 1.54 se observa la pérdida de información por desconexión abrupta sin control.

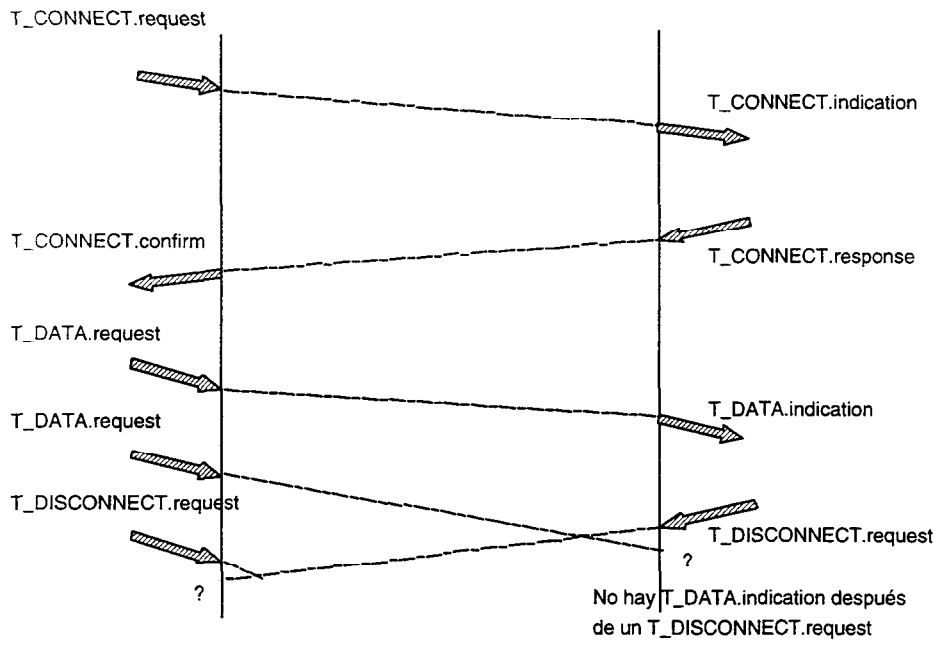


Fig. 1.54 Problemas de desconexión abrupta

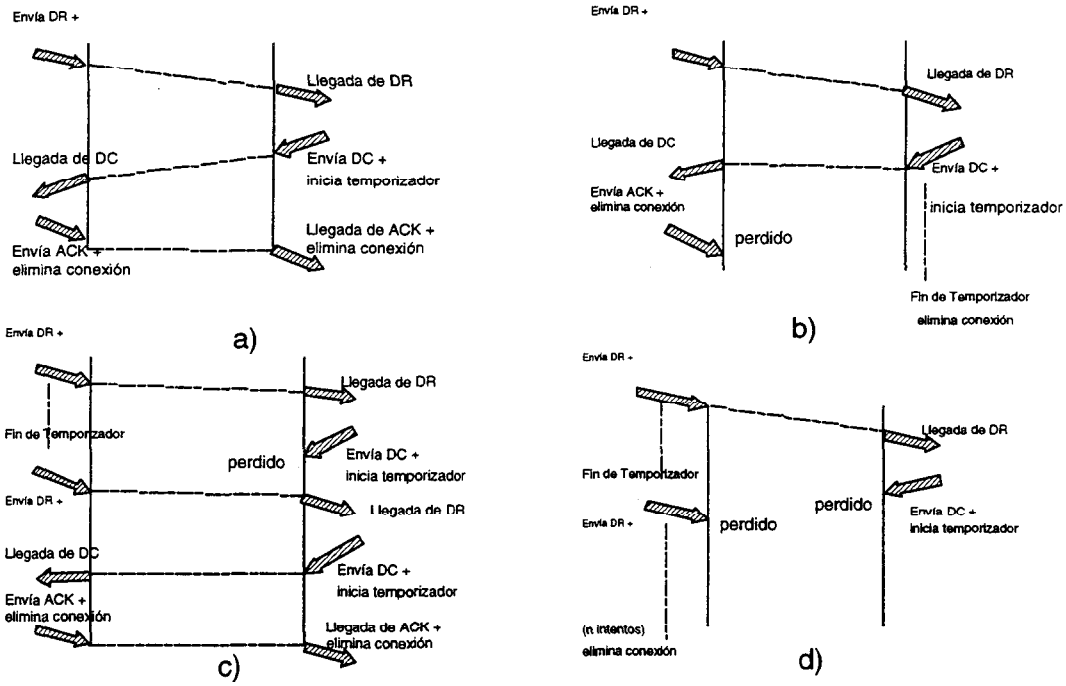
Se requiere de igual manera un **protocolo de ida-vuelta-ida**, que si bien no es infalible si, por lo general, es adecuado.

En la figura 1.55 se muestran cuatro situaciones de liberación utilizando dicho protocolo.

En la marcada con a) se presenta el caso normal, en el que uno de los usuarios transmite una TPDU DR (solicitud de desconexión), para iniciar la liberación de la conexión. En el momento en que ésa llega, el receptor devuelve una TPDU DC (confirmación de desconexión) y arranca un temporizador, por si acaso se pierde la TPDU DC. En el momento en que esta DC llega, el emisor original devuelve una TPDU ACK y elimina la conexión. Si se llegará a perder el último ACK, como se muestra en la figura b), la situación queda protegida por el temporizador. Cuando el plazo de éste término, la conexión, de todos modos, se elimina.

Ahora consideramos el caso en el que se pierde un DC (o DR). El usuario que inicia la desconexión no recibirá la DC, pero al término de una temporización comenzará todo de nuevo. En la figura representada con c) se observa cómo funciona esto, suponiendo que en esta segunda ocasión no se llegan a perder las TPDUs.

En el último caso, el d), es el mismo que en el c) salvo que ahora se supone que todos los intentos de repetición, para transmitir la DR, también son infructuosos, como consecuencia de la pérdida de las TPDU. Después de intentar hacerlo n veces, el transmisor se da por vencido y elimina la conexión. Mientras tanto, el receptor, al término de una temporización, también la deja.



a) Caso normal de una comunicación ida-vuelta-ida
b) Pérdida del último ACK.
c) DC perdido y
d) DC perdido y pérdida de los subsiguientes DR

Fig. 1.55 Protocolo ida-vuelta-ida con temporizadores

Aunque este protocolo, por lo general es suficiente, en teoría puede fallar si tanto la DR inicial como las n retransmisiones se llegan a perder. El transmisor se dará por vencido y eliminará la conexión, mientras que el otro extremo no tiene conocimiento alguno sobre los intentos de desconexión, y sigue todavía activo. Habrá por tanto una conexion

abierta en uno de los extremos. Se puede evitar este problema mediante que el transmisor no desista tras los n intentos, hasta obtener respuesta. Sin embargo, si al otro extremo se le permite salir después de un tiempo, entonces el emisor efectivamente continuará transmititendo por siempre, porque no habrá respuesta que llegue jamás. Si al extremo receptor no se le permitiese salir después de un tiempo, entonces el protocolo se bloqueará. (Ver figura 1.55 apartado b)).

Esto se soluciona mediante el uso de **una regla** indicando que, en caso de que no hayan llegado TPDUs durante cierto número de segundos, la conexión se desconectará automáticamente. De esa forma, si alguno de los lados se desconecta alguna vez, el otro extremo detectará la falta de actividad y también se desconectará. Para ello es necesario que exista un temporizador en cada entidad de transporte que se detenga y después se reinicie cuando se envíe alguna TPDU. Si el período de temporizador expira, se transmitirá una TPDU de relleno, sólo con el objeto de evitar que el otro extremo se desconecte. Por otra parte, si la regla de desconexión automática se utiliza y se llegan a perder demasiadas TPDU de relleno en una conexión que de otra forma estaría inactiva, se desconectarán en forma automática, primero un lado y después el otro.

1.4.4 Administración de conexiones basadas en temporizadores.-

Volviendo a la situación de TPDUs flotando por la red, vimos que se podría solucionar en parte con el uso de identificador de conexión único a cada conexión, para que pueda llegar a asentir aquellas TPDUs procedentes de conexiones anteriores. Otra segunda manera fue utilizar el protocolo ida-vuelta-ida para el establecimiento de conexiones. Fletcher y Watson (1978) y Watson (1981) propusieron una tercera manera basada en temporizadores.

La idea es controlar la falta de actividad como para el caso de la liberación anterior. Según FyW hicieron que la entidad de transporte no eliminase información con respecto a una conexión, hasta que todas las TPDUs relacionadas con ella dejaran de existir. Así las entradas en la tabla de conexiones no se eliminan tras que la conexión sea liberada, sino cuando haya expirado un intervalo de tiempo cuidadosamente seleccionado.

El proceso es el siguiente :

Cuando un emisor desea enviar a un receptor un flujo consecutivo de TPDU crea internamente un registro de conexión. Por medio de este registro de conexión se mantiene un seguimiento de aquellas TPDUs que se han enviado, de qué asentimiento se han recibido, etc. Cada vez que se crea un registro de conexión, se arranca un temporizador. Cada vez que una TPDU se envía, utilizando un registro de conexión creado previamente, el temporizador se arranca de nuevo. Si el período del temporizador expira (lo cual significa que no se ha transmitido nada durante cierto intervalo de tiempo), se elimina el registro de conexión. Los intervalos de tiempo para el emisor y el receptor son diferentes, y se seleccionan cuidadosamente, para que el receptor siempre elimine su registro de conexión, mucho antes que el emisor.

La primera TPDU en el flujo contiene una bandera de un bit, llamado DRF (Bandera de serie de datos), que lo identifica como el primer elemento de una serie de TPDU. En el momento en que se transmite cualquier TPDU, se arranca un temporizador; si se recibe

asentimiento de TPDU, el temporizador se detiene; si el temporizador vence, se retransmite la TPDU. Si, por otra parte, después de que se efectúan n retransmisiones, no existe todavía un asentimiento, el emisor se da por vencido. El tiempo de abandono tiene un papel muy importante en el protocolo.

Cuando llega al receptor una TPDU con la bandera DRF puesta, el receptor anota su número de secuencia y crea un registro de conexión. Las TPDUs subsiguientes sólo se aceptarán se están en secuencia. Si la primera TPDU que llega al receptor no tiene la bandera DRF puesta, se descartará. Eventualmente, la primera TPDU (ya sea original o correspondiente a un retransmisión), llegará y el registro de conexión podrá crearse. En otras palabras, solamente se puede crear un registro de conexión en el momento en que llega una TPDU con la bandera DRF puesta.

Cada vez que una TPDU llega en secuencia, se pasa al usuario de transporte y se devuelve un asentimiento al emisor. Si llega una TPDU fuera de secuencia, cuando exista un registro de conexión, podrá ser almacenada. También podrá ser asentida. Sin embargo, el hecho de que se reciba un asentimiento no implica que todas las TPDU anteriores también hayan sido recibidas a menos que la bandera, ARF(bandera de serie asentimientos), haya sido puesta.

Veamos varios casos :

Una secuencia de TPDU se transmite y todas se reciben ordenadamente y , a su vez, son asentidas. Cuando el emisor obtiene el asentimiento de la última TPDU que se transmitió, y ve la bandera ARF, detiene todos los temporizadores de retransmisión. Si no hay más datos próximos a llegar procedentes del usuario, del transporte, el temporizador del registro de conexión del receptor expira eventualmente, para que posteriormente también lo haga el del emisor. No hay una necesidad explícita de TPDU de establecimiento o liberación (aunque la bandera DRF es de alguna manera análoga a una TPDU CR).

Ahora, considere qué sucede si la TPDU que lleva la bandera DRF se llega a perder. El receptor no creará un registro de conexión. El emisor, eventualmente, retransmitirá la TPDU al término de una temporización en forma repetida si es necesario, hasta que sea asentida. Una vez que el receptor haya creado un registro de conexión, se puede llegar a detectar con facilidad una discontinuidad en el flujo de TPDU, y se puede reparar mediante las temporizaciones y retransmisiones del emisor.

Supóngase que se transmite y recibe correctamente un flujo de TPDU, pero que algunos de los asentimientos se llegan a perder. Las retransmisiones subsiguientes también se pierden, por lo que el registro de conexión del receptor se eliminará la término de una temporización. ¿ Como puede impedirse ahora que una antigua TPDU duplicada que tenía la bandera DRF puesta, aparezca en el receptor y active un nuevo registro de conexión ?. El truco consiste en hacer que el temporizador del registro de conexión del receptor sea mucho mayor que el tiempo de abandono del emisor, más el tiempo de vida máximo de las TPDUs. Por lo tanto, una vez que la TPDU inicial es aceptada, el registro de conexión mantendrá su existencia hasta que el receptor este seguro de que el emisor detuvo efectivamente el envío de TPDU con la bandera DRF (ya sea porque obtuvo un asentimiento o se dio por vencido). De esta manera, una vez que fue aceptada la TPDU inicial, no habrá ningún peligro de que pueda aparecer, después de que el

registro de conexión se haya eliminado - todas las copias de la misma habrán desaparecido por completo - . Las otras TPDU de la serie son inofensivas, porque el receptor no las aceptará (solamente las TPDUs con la bandera DRF puesta serán aceptables cuando el receptor no tenga un registro de conexión).

Además, si algunas TPDU permanecen sin asentimiento, el emisor continuará retransmitiéndolas, manteniendo activo su registro de conexión. Mientras el registro de conexión muestre que existen TPDU sin asentimiento pendientes no se enviarán nuevos TPDUs con la bandera DRF puesta.

Este protocolo es una combinación de propiedades de protocolos sin conexión y orientados a conexión.

1.4.5 Control de Flujo.-

En redes inseguras tipo B y C la capa de transporte debe almacenar copia de las TPDUs transmitidas por si necesitase retransmitirlas al no recibir asentimientos. Para ello la memoria o tampones se deberán reservar de una de las tres formas siguientes :

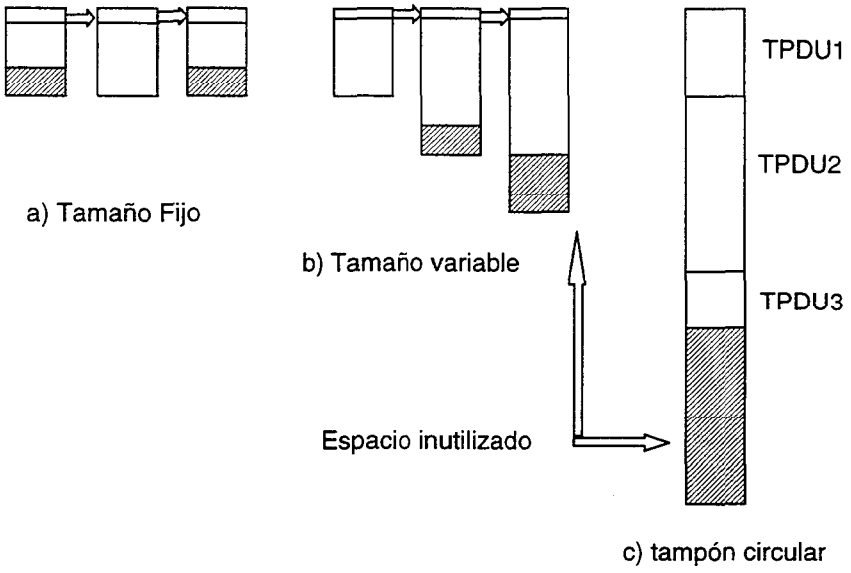


Fig. 1.56 Posibles configuraciones de buffers

El primero a) que se reserva tantos tampones como TPDUs enviadas de tamaño fijo es poco optimo pues no todas las TPDUs tienen el mismo tamaño. Otra solución es el tamaño variable. Por ultimo el más optimo es el tampón circular que hace un buen uso de la memoria pero es menos ventajoso para conexiones con poco tráfico. La solución idónea depende del tráfico acarreado. Así por ejemplo para tráfico a ráfagas con un ancho de banda pequeño, como el producido en un terminal interactivo, resulta más conveniente no dedicarle ningún tampón, sino más bien adquirirlos dinámicamente en los dos extremos. Por otro lado para transferencias de ficheros y otro tipo de tráfico con ancho de banda grande, es mejor que el receptor pueda dedicar una ventana completa de tampones. Por lo general la capa de transporte va a negociar el número de tampones que

puede necesitar en el receptor y a la inversa. Este tipo de asignación de ventana dinámica presenta más problemas que en el caso de ventana fija como en la capa de enlace.

Se puede ver un ejemplo y analizar que puede suceder :

	Host A	Mensaje	Host B	Observaciones
1	>	<request 8 buffers>	>	A desea 8 tampones
2	<	<ack = 15, buf = 4>	<	B solamente autoriza mensajes de 0 a 3
3	>	<seq = 0, data = m0>	>	A solo se queda con 3 tampones
4	>	<seq = 1, data = m1>	>	A solo se queda con 2 tampones
5	>	<seq = 2, data = m2>	....	El mensaje se pierde, pero A piensa que todavía le queda 1
6	<	<ack = 1, buf = 3>	<	B asiente 0 y 1, permite 2-4
7	>	<seq = 3, data = m3>	>	A solamente tiene un tampón
8	>	<seq = 4, data = m4>	>	A tiene 0 tampones, y deberá detenerse
9	>	<seq = 2, data = m2>	>	A retransmite después de una temporización
10	<	<ack = 4, buf = 0>	<	Todo se asiente, pero A permanece bloqueado
11	<	<ack = 4, buf = 1>	<	A puede ahora transmitir 5
12	<	<ack = 4, buf = 2>	<	B encontró un nuevo tampón en alguna parte
13	>	<seq = 5, data = m5>	>	A tiene un tampón
14	>	<seq = 6. data = m6>	>	A queda bloqueado nuevamente
15	<	<ack = 6, buf = 0>	<	A esta todavía bloqueado
16	.....	<ack = 6, buf = 4>	<	Bloque potencial

Supongamos que se esta administrando dinámicamente la ventana en una subred que utiliza datagrama con número de secuencia de 4 bits. Supóngase que la información de asignación de tampones viaja en TPDU separadas y no se superponen. Al principio A desea 8 memorias, pero sólo se le otorgan cuatro. Entonces, envía tres TPDUs de las cuales la tercera se pierde. La TPDU 6 asiente la recepción de todas las TPDUs hasta el número de secuencia 1 inclusive, permitiéndole a A enviar otras tres TPDUs adicionales, comenzando a continuación del número 1 (es decir, las TPDUs 2, 3 y 4). El extremo A sabe que ya envió el número 2, por lo que piensa que puede enviar las TPDUs números 3, 4, lo que procede a hacer. En ese momento queda bloqueado y

deberá esperar a que se le asigne más memoria. Sin embargo, las retransmisiones inducidas por temporización (línea 9), pueden llegar a presentarse mientras esté bloqueado, dado que utilizan tampones que ya se han asignado previamente. En la línea 10, B asiente la recepción de todas las TPDU hasta, e inclusive el número 4, pero se deja que A continue. La siguiente TPDU desde B hasta A asigna otro tampón y permite que A continúe.

Problemas potenciales aparecen como el de la línea 16, donde B ha asignado más memoria para A, pero se perdió la TPDU de asignación. Dado que las TPDU de control no van en secuencia, ni están protegidas por una temporización, A se encuentra ahora bloqueada. Para evitar esto es necesario transmitir periodicamente TPDU de control para dar el estado de los asentimientos y memoria de cada una de las conexiones.

Si bien como límite del control de flujo hemos supuesto la capacidad de memoria del receptor, hay que recordar, que hay una limitación debida al canal de transmisión y el número de ellos, ya que si tengo un subred con canal de X tramas por segundo y hay k trayectorias posibles entre un par de hosts, no hay manera de intercambiar más de kx TPDU por segundo (si corresponde una TPDU por trama).

Capa de Transporte Internet

2. Capas de Transporte Internet

2.1 Introducción

Como se sabe, la arquitectura TCP/IP, que es la utilizada en Internet esta formada por cinco capas:

Capa de Aplicación que proporciona la comunicación entre procesos o aplicaciones de diferentes sistemas. Esta formada por los diferentes protocolos de aplicación para redes (p. e. transferencia de archivos, terminal virtual, correo electrónico, servicio de news, acceso a la world wide web, etc.)

Capa de Transporte que proporciona un servicio de transferencia de datos extremo a extremo. Su principal actividad es independizar los detalles de la red de las diferentes aplicaciones o sea hacer transparente el acceso a la red de transporte

Capa Internet que se encarga de encaminar las unidades de datos (paquetes) desde el origen al destino. Su principal misión es direccionar las estaciones para posibilitar su acceso mediante un protocolo determinado (p.e. IP: Internet Protocol).

Capa de acceso a la red que esta relacionada con la interfaz lógica entre los sistemas y gestiona el acceso al medio físico.

Capa física que especifica las características eléctricas, funcionales y procedurales del medio físico y las señales que se transmiten por él (p.e. Capa física Ethernet, X.21, V24)

En cada una de las capas se implementan una serie de protocolos de lo que se conoce como la familia de protocolos TCP/IP. Para ello si se observa la figura 2.0 se muestra la ubicación de los mismos contrastado con el RM-OSI en primer término, arquitectura TCP/IP y la familia de protocolos principal.

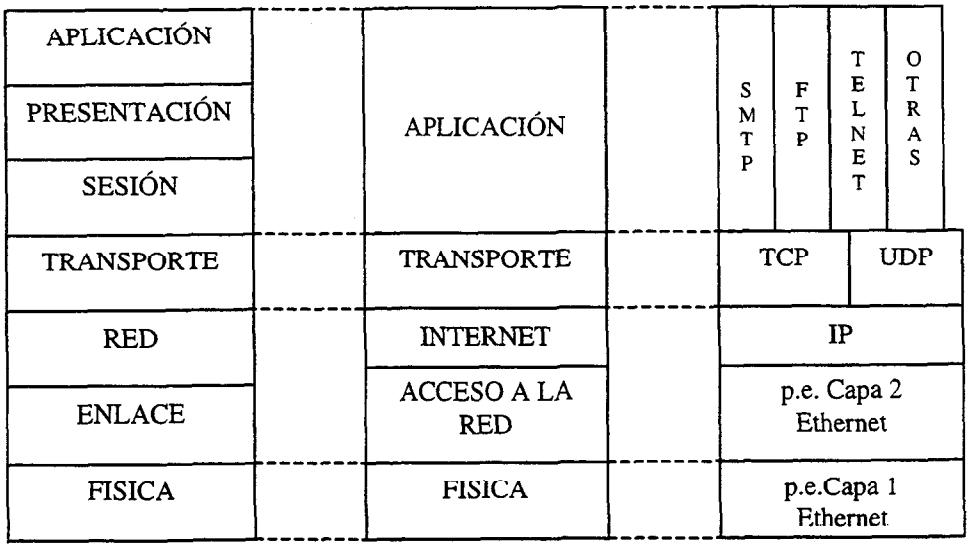


Fig. 2.0 Arquitectura TCP/IP

Como cada capa interacciona con sus adyacentes, la capa de aplicación utilizará los servicios de la capa de transporte pasándole los datos y a su vez, esta utilizará los servicios de la capa Internet y así sucesivamente.

Es de destacar, que mientras las capas inferiores utilizan un determinado protocolo en función de medio físico y direccionamiento en red (Capas 1, 2 y 3), para la capa de transporte se especifican dos protocolos; uno orientado a conexión (TCP: Transmisssion Control Protocol) y otro no orientado a conexión (UDP: User Datagram Protocol). Uno de estos dos protocolos será utilizado e implementado en los sistemas finales para garantizar la entrega segura a las diferentes aplicaciones.

Cualquiera que sea el protocolo de transporte utilizado, TCP o UDP, la capa IP transporta los datos entregados para alcanzar la máquina destino mediante el uso del número IP (32 bits o mediante notación decimal de puntos) que identifica dicha máquina destino.

Finalmente la capa 2 encapsula los paquetes IP en función de la tecnología de red utilizada para progresar por los diferentes medios y nodos, si existiesen.

TCP
(Transmission Control Protocol)

CAPA DE TRANSPORTE INTERNET.

2.2 TCP (Transmission Control Protocol).-

2.2.0. Introducción.-

Este protocolo esta definido en la RFC793 de las especificaciones para Internet. En este estándar se dan recomendaciones de cómo debe ser un interfaz entre TCP y los usuarios de dicho servicio, en forma de llamadas a funciones y el protocolo correspondiente TCP.

TCP es un protocolo equivalente al protocolo de transporte OSI diseñado específicamente para trabajar sobre redes inseguras (de tipo C, en términos del modelo OSI). Acepta mensajes de tamaño variable, e incluso excesivamente grande, procedentes de los procesos de usuario, los separa en pedazos que no excedan de 64k bytes, y transmite cada pedazo como si fuera un datagrama separado. La capa de red, no garantiza que los datagramas se entreguen apropiadamente, por lo que TCP deberá utilizar temporizadores y retransmitir los datagramas si es necesario. Los datagramas que consiguen llegar, pueden hacerlo en desorden; y dependerá de TCP el hecho de reensamblarlos en mensajes, con la secuencia correcta.

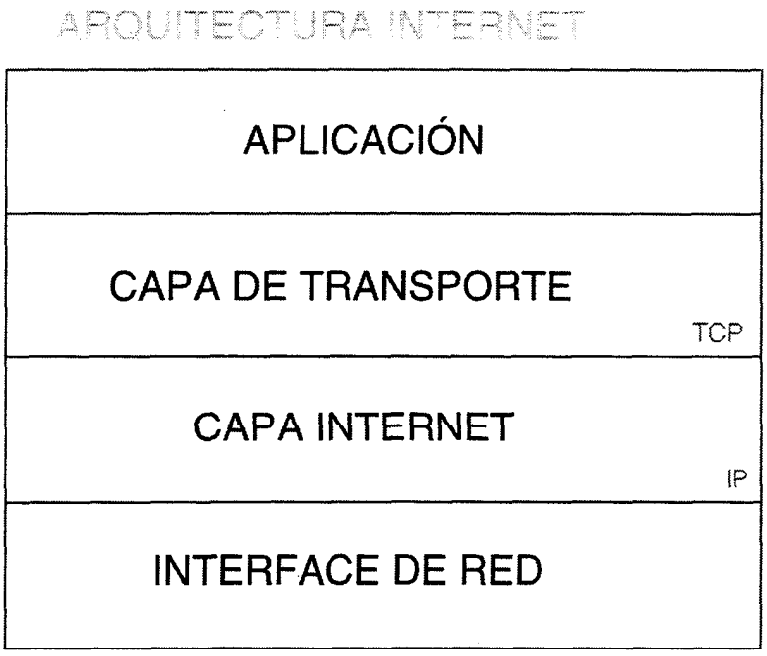


Fig. 2.1 Arquitectura Internet

2.2.1. Servicio de Transporte Internet.-

Internet especifica dos implementaciones del servicio de transporte, un servicio no orientado a conexión y otro servicio orientado a conexión. Ambas implementaciones se realizan mediante sus dos protocolos asociados, o sea **UDP (User Datagram Protocol)** para el primer caso y **TCP (Transmision Control Protocol)** para el segundo caso, que es el que nos ocupa.

El servicio orientado a conexión, proporciona un servicio de transporte seguro, orientado a conexión y extremo-a-extremo sobre una red insegura que puede perder, dañar, almacenar y duplicar paquetes, como el caso del TP4 del modelo OSI.

Es un servicio confirmado, lo cual nos indica que posee las fases de establecimiento de conexión, transferencia de datos y liberación de conexiones.

Las definiciones del servicio entre TCP y su capa inferior no se especifican en el standard TCP. Aunque se asume que las operaciones entre TCP y la capa inferior pueden hacerse pasando información de una a otra asíncronamente.

Para solicitar su servicio se definen una serie de llamadas con sus parámetros asociados. Las principales llamadas al sistema son :

➔ ESTABLECIMIENTO DE CONEXIÓN.-

OPEN(local,remoto,act/pas[,temporizador][,prioridad][,seguridad][,opciones]-> nombre de conexión)

Los parámetros entre corchetes son opcionales y los que siguen a -> son los valores devueltos por la función.

Llamar a OPEN supone solicitar a la entidad de transporte que inicie una conexión de transporte, donde **local** es el socket[1] (dirección del TSAP origen en la terminología OSI), **remoto** es el socket remoto y **act/pas** especificará si dicha conexión se inicia por el usuario local (conexión ***activa***) o se pretende esperar por otra iniciada por la entidad extremo (conexión ***pasiva).***

En el caso de ser pasiva se puede especificar el socket extremo, con lo cual se indica de quién se espera conexión o no indicarlo, en cuyo caso cualquier interlocutor podría conectarse. Para saber con quién se establece la conexión se hace uso de la llamada al servicio **STATUS.**

Si se especifica el parámetro **temporizador**, se indica al proveedor del servicio de transporte (capa de transporte Internet) que establezca un temporizador para la

[1] Socket : Es la concatenación de un puerto y una dirección de red. Las conexiones se identifican por un par de sockets que se comunican.

conexión, o sea que todo dato que se envíe por la conexión se destruirá si no alcanza su destino en un tiempo inferior al indicado por este parámetro.

Los parámetros **prioridad y seguridad** permiten especificar algunas características de calidad de servicio de la conexión. TCP sólo permitirá al usuario establecer una conexión de forma pasiva con otro usuario activo que haya solicitido el mismo valor de seguridad y un valor mayor o igual prioridad.

OPEN devuelve un **nombre** de la conexión, que después podrá ser utilizado en vez del nombre completo de la conexión, definido por el par <socket local, socket remoto>.

→ TRANSFERENCIA DE DATOS.-

SEND (nombre de conexión, buffer_datos, long_datos, señal_PUSH, señal_URGENT [,acción en tiempo máximo])

Mediante SEND se solicita al proveedor del servicio que envíe por la conexión, identificada con **nombre de conexión**, la cantidad de **long_datos** bytes ubicados en la zona de memoria especificada por **buffer_datos.**

Si el parámetro **señal_PUSH** está activo se obliga a la entidad TCP a que envíe los datos al otro extremo inmediatamente, sin esperar a que este lleno el buffer. Si no lo está, el proveedor TCP puede almacenar datos de diferentes SEND para concatenarlos entre sí antes de enviarlos, haciendo así un uso más eficiente de los servicios de red.

Si el parámetro **señal_URGENT** está activo los datos se consideran como urgentes, y así se le indicará al otro extremo.

Si el parámetro **acción en tiempo máximo** está activo, se cambia el valor previamente establecido como temporizador de la conexión por el que aparece en SEND.

El estándar sugiere que la implementación de SEND sea **asíncrona,** esto es, el usuario invoca al servicio SEND y la entidad TCP receptora le devuelve el control inmediatamente, antes de que se haya realizado la entrega de datos en el otro extremo. Esto permite tener pendientes varios SENDs, que TCP debe poner en cola y procesar en orden FIFO (First In-First Out) o FCFS (First Come-First Serve). Este sistema exige, sin embargo, que exista un mecanismo para que la entidad TCP comunique a su usuario si su petición ha tenido éxito o no.

Corresponde a enviar como solicitud en el lado del cliente y el servidor.

RECEIVE (nombre de conexión, buffer_datos, long_datos, -> long_datos, señal_PUSH, señal_URGENT)

Este servicio sirve para recibir datos por la conexión de transporte abierta cuyo **nombre de conexión** local se indica. Una implementación simple de RECEIVE obligaría al usuario a bloquearse hasta que la petición fuese satisfecha. Más aconsejable sería una

implementación en la que el usuario retomase el control inmediatamente, de forma que un proceso pudiese tener pendientes diferentes peticiones de recepción, que irían satisfaciendo conforme llegasen datos. Esto añade la complejidad de implementar un sistema para avisar a los procesos, de forma asíncrona, que sus peticiones han sido atendidas.

Si TCP tiene pendientes de entrega datos suficientes para llenar el **buffer_datos**, **señal_PUSH** estará inactiva y los dos **long_datos** serán iguales. Si se tienen menos, **señal_PUSH** estará activa y el buffer sólo se llenará parcialmente.

En el caso de que se reciban datos urgentes del otro usuario, la entidad TCP informa a su usuario de este suceso. El usuario pasa a un ***modo urgente*** y recibe los datos urgentes. Si no los lee todos de una vez, **señal_URGENT** permanece activa. Cuando se han leído todos los datos urgentes, esta señal se inactiva y el usuario puede abandonar el modo urgente.

Una particularidad del servicio ofrecido por TCP es que el número de RECEIVEs solicitados por el receptor no tiene por qué ser igual que el número de SENDs del emisor. Se dice que este servicio está **orientado a bytes** porque el emisor entrega grupos de bytes a su servidor TCP en el otro extremo, donde el usuario receptor los va tomando de la forma que a él le convenga (quizá byte a byte). TCP ofrece un **canal para el intercambio de bytes**, frente al **canal de intercambio de mensajes** (TSDUs) del servicio de transporte OSI.

STATUS(nombre de la conexión,-> info_estado)

Se puede invocar esta función en cualquier momento para obtener un bloque de información acerca del estado de la conexión, que incluye : socket local, socket destino, nombre de la conexión en el sistema local, valor en curso del temporizador, etc..

Corresponde a estado y es una solicitud en el lado del cliente y del servidor.

➔ LIBERACIÓN DE CONEXIÓN.-

CLOSE(nombre de la conexión)

Inicia el cierre de una conexión. Se intenta que el cierre sea ordenado, en el sentido de que todos los datos que se encuentren haciendo cola sean entregados. Después de realizar la función CLOSE, el usuario debe seguir haciendo RECEIVEs para aceptar los posibles datos pendientes. Hay que entender CLOSE como "**No tengo nada más que enviar**", en contraposición con el "**No quiero recibir nada más**" del cierre de conexiones de transporte en OSI. El tiempo máximo que habrá que esperar para que la conexión se dé definitivamente por cerrada estará relacionado con el último valor establecido para el temporizador de la conexión.

Corresponde a cerrar y es una solicitud en el lado del cliente y del servidor.

ABORT(nombre de la conexión)

Cierre abrupto de la conexión. Tiene como efecto lateral que todos los SEND y RECEIVE pendientes sean cancelados. Similar al cierre de conexiones OSI.

Corresponde a abortar y es una solicitud en el lado del cliente y servidor.

La comunicación de sucesos a los usuarios del servicio tales como la finalización de un SEND o un RECEIVE, o la aparición de un error, o el paso a modo urgente se llevan a cabo a través de comunicación entre procesos gestionados por el sistema operativo de las máquinas donde las entidades de transporte estan implementadas. Esta información incluiría :

- Nombre local de la conexión.
- Un mensaje indicando la naturaleza del suceso.
- Una dirección del buffer en el que se encuentran los datos, para el caso de SEND y RECEIVE.
- El número de bytes recibidos (RECEIVE).
- El valor de las señales señal_PUSH y señal_URGENT.

Esta información se correspondería con las primitivas indication y confirm en terminología OSI.

En la figura 2.2 se ilustra el uso de estas solicitudes de servicio y donde actuán :

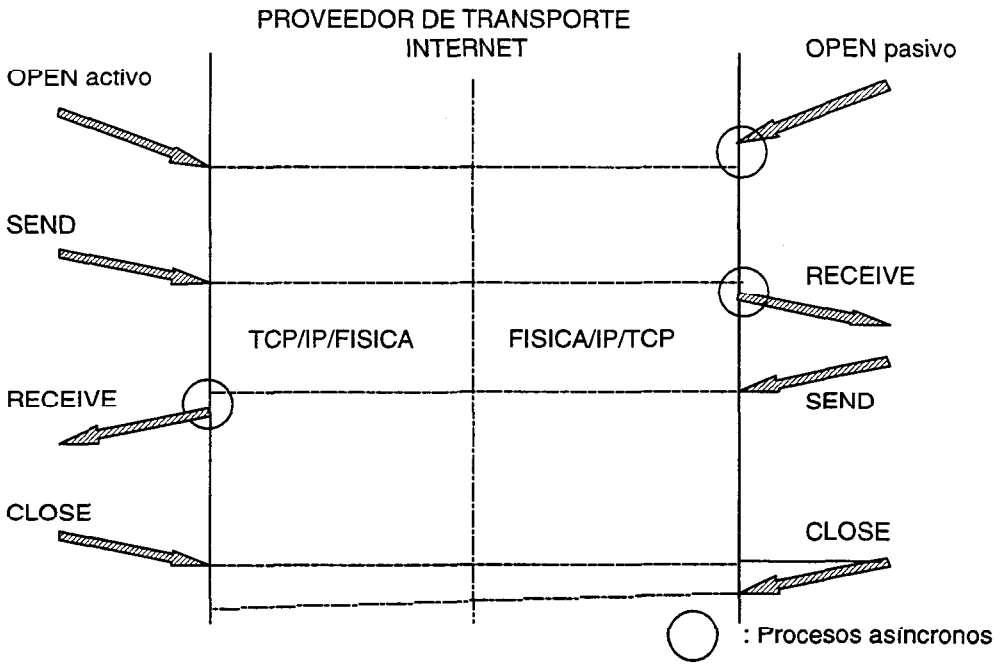


Fig. 2.2 Solicitudes del Servicio de Transporte Normal

Además de las anteriores primitivas de solicitud de servicio desde el usuario del servicio al proveedor (TCP), en las RFC793 se especifican las siguientes, aunque las diferentes implementaciones pueden variar:

Primitivas de Petición:

UNSPECIFIED_PASSIVE_OPEN (puerto local, ULP timeout(*), acción en timepo máximo(*), precedencia(*),intervalo de seguridad)

En esta clase de conexión, no se asocian direcciones, pero los parámetros de seguridad estarían dentro de un rango aceptable. Corresponde a una apertura no especificada. Esta primitiva es una solicitud en el servidor.

FULL_PASSIVE_OPEN (puerto local, puerto destino, dirección destino, ULP timeout (*), acción en tiempo máximo(*), precedencia(*), intervalo de seguridad(*), opciones(*)).

En esta clase de conexión la dirección destino es la misma en el lado activo y pasivo. No obstante, la operación de apertura pasiva local tiene totalmente especificado el socket remoto. Los parametros de seguridad en el lado activo estan dentro del rango de parámetros de seguridad del lado pasivo. Corresponde con una apertura completa y es una solicitud en el servidor.

ACTIVE_OPEN (puerto local, puerto destino, dirección destino, ULP timeout(*), acción en tiempo máximo(*), precedencia(*), intervalo de seguridad(*), opciones(*))

Corresponde con una apertura activa de solicitud en el lado del cliente.

ACTIVE_OPEN_WITH_DATA(puerto fuente, puerto destino, dirección destino, ULP timeout(*), acción en tiempo máximo(*), precedencia(*), intervalo de seguridad(*), datos, longitud de datos, flag push, flag urgente(*))

Corresponde con una apertura activa incluyendo datos de solicitud en el lado del cliente.

ALLOCATE (nombre de conexión local, longitud de datos).

Corresponde a asignar y es una solicitud en el lado del cliente y del servidor.

Nota : (*) indica opcional.

Otras primitivas de respuesta desde TCP al usuario del servicio son:

OPEN_ID (nombre de conexión local, buffer_datos, long_datos, -> long_datos, señal_PUSH, señal_URGENT)

Corresponde a la identificación de apertura y es una respuesta local en el lado del cliente.

OPEN_FAILURE (nombre de conexión local)

Corresponde a un fracaso de apertura y es una confirmación en el lado del cliente.

OPEN_SUCCESS (nombre de conexión local)

Corresponde a la apertura con éxito y es una confirmación en el lado del cliente.

DELIVER (nombre de conexión local, dirección de buffer, contador de bytes, flag de urgente)

Corresponde a la entrega y es una indicación en el lado del cliente y del servidor.

CLOSING (nombre de conexión local)

Corresponde a cierre y es una indicación en el lado del cliente y del servidor..

TERMINATE (nombre de conexión local, descripción(razón))

Corresponde a terminar y es una confirmación en el lado del cliente y del servidor.

STATUS_RESPONSE (nombre de conexión local, dirección y puerto local, dirección y puerto remoto, estado de conexión, ventana de transmisión y recepción, cantidad de ACK esperados y recibidos, modo urgente, precedencia, timeout,acción por tiempo máximo, seguridad).

Corresponde a respuesta de estado y es una respuesta en el lado del cliente y del servidor.

ERROR (nombre de conexión local, descripción del error).

Corresponde a un error y es una indicación en el lado del cliente y del servidor.

Los diferentes parámetros representan :

- **Dirección** : es la dirección IP origen o destino, según se indique en cada caso.
- **Puerto** : es la dirección de puerto intercapas asociadas a los protocolos de origen y destino.
- **Temporizador o Tiempo Máximo ULP** : Permite a un protocolo de aplicación de usuario especificar el tiempo máximo que el TCP de origen debe esperar la confirmación relacionada con un segmento que se transmitió.
- **Acción en tiempo Máximo** : Especifica que debe hacerse si expira el tiempo máximo. Normalmente lo que se hace es cerrar la conexión.
- **Precedencia** : Conjunto de parámetros con los que el usuario puede especificar el contenido del campo de tipo de servicio de la cabecera del datagrama IP que se usará para transportar el segmento TCP.
- **Intervalo de Seguridad**: Permite a un protocolo de aplicación servidor especificar un nivel de seguridad que debe aplicarse a los usuarios potenciales.

- **Flag de empujar y urgente** : Permite al usuario indicar al TCP como debe tratar los datos del parámetro **datos**: Empujar significa transmitir de inmediato y urgente significa transmitir fuera del flujo normal.

➢ A continuación (en la figura 2.3) se muestra la secuencia de primitivas para las diferentes fases de una conexión TCP.

① Fase de Establecimiento de Conexión.-

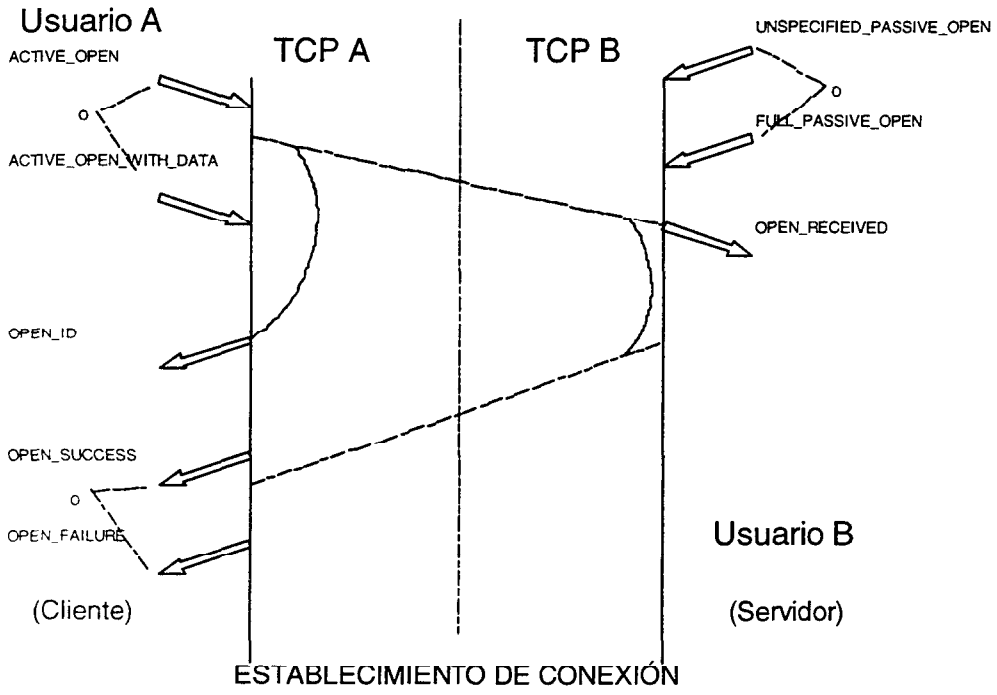


Figura 2.3. Primitivas de gestión de establecimiento de conexión

La primera se corresponde con el establecimiento de la conexión. Un servidor se vale de las primitivas **UNSPECIFIED_PASSIVE_OPEN** o **FULL_PASSIVE_ OPEN**. En el primer caso indica que esta preparado para recibir una solicitud de conexión, más que para iniciar (activamente) el establecimiento de la conexión y además no especificada, o sea, permite aceptar la conexión de cualquier proceso que satisfaga el nivel de seguridad especificado si existe. En el segundo caso, contiene también parámetros de puerto de destino y dirección IP para indicar que protocolo/proceso de aplicación especifico del cual el servidor esta esperando una conexión. En el lado del cliente, se puede establecer la conexión mediante **ACTIVE_OPEN** especificando las direcciones de puerto y de IP del destino o con **ACTIVE_OPEN_WITH_DATA** que además incluye un corto mensaje de datos de usuario en un parámetro, que se entrega al usuario remoto mientras se establece la conexión. En ambos casos el TCP local responde con **OPEN_ID** devolviendo un nombre de identificación de conexión que lo ha asignado a esta solicitud de conexión y procede a iniciarla con el módulo TCP remoto. Si hay éxito, en el lado de cliente se indicará con **OPEN_SUCCESS** y en el lado de servidor **OPEN_RECEIVED** los módulos TCP podrán intercambiar datos en durante la fase de transferencia de datos correspondiente, en caso contrario el cliente recibe del TCP local **OPEN_FAILURE** con la causa indicada en sus parámetros.

② Fase de Transferencia de Datos.-

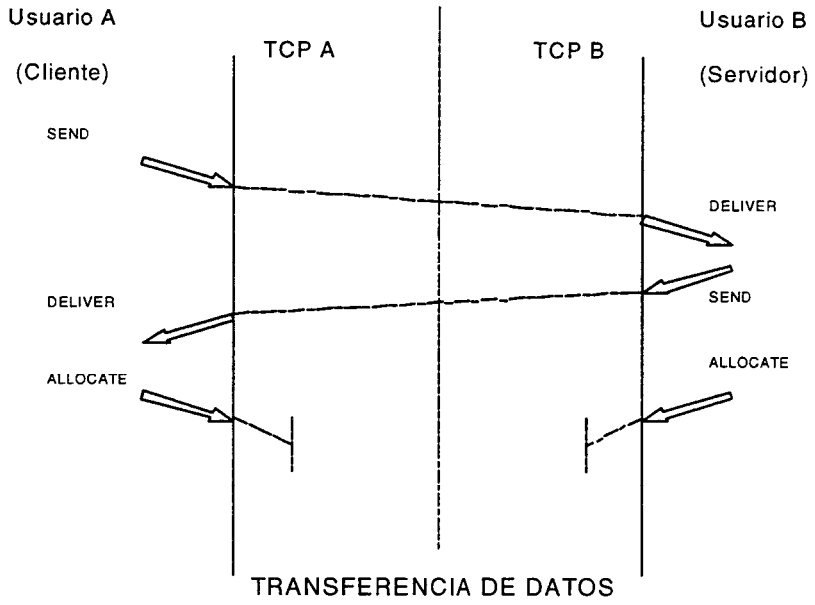


Figura 2.4. Primitivas de gestión de transferencia de datos.

Los datos se transfieren en ambos sentidos mediante las primitivas **SEND** y **DELIVER** junto con los flags de empujar (PUSH) y de urgente (URG), si es necesario. Con la primitiva **ALLOCATE,** un usuario puede incrementar la cantidad de almacenamiento temporal para mensajes de cada conexión en el lado local (ver figura 2.4).

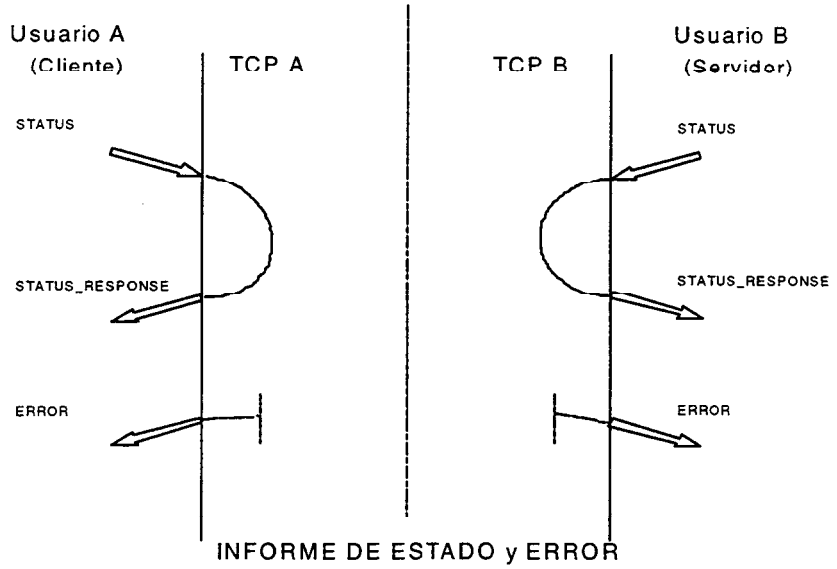


Figura 2.5. Primitivas de gestión del estado y errores de las conexioens

Además, mientras se esta en la fase de transferencia de datos, el usuario puede solicitar que se le informe del estado de la conexión emitiendo una primitiva **STATUS**, a la cual el TCP local responderá con una primitiva **STATUS_RESPONSE**. Si ocurre una condición de error -por ejemplo, si las dos entidades de protocolo de TCP se

desincronizan- se informará de esto al usuario con una primitiva **ERROR** (ver figura 2.5).

③ Fase de Liberación de Conexión.-

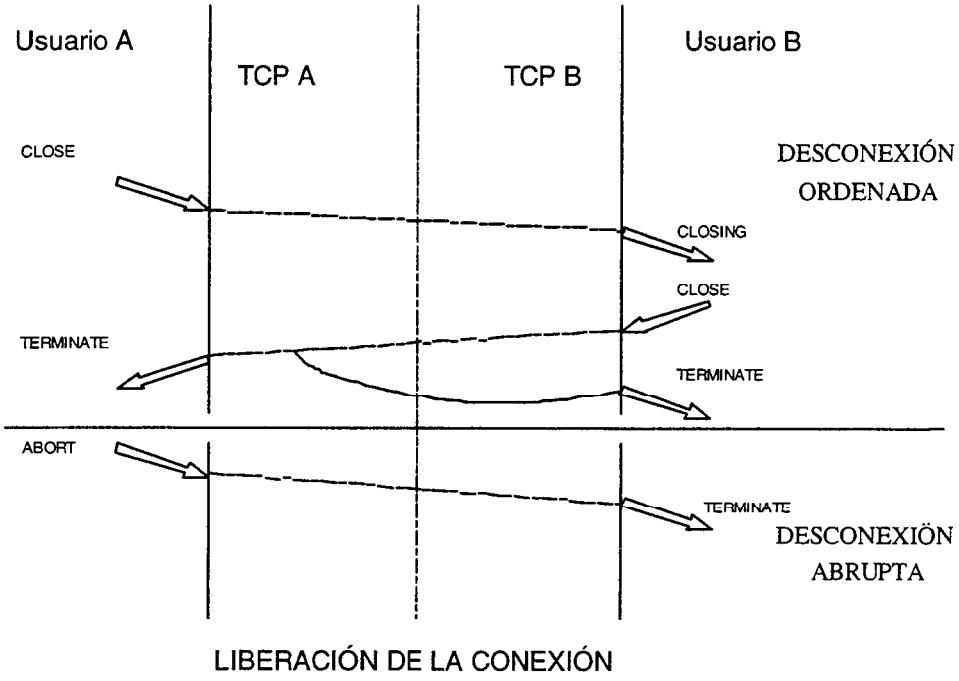


Figura 2.6. Primitivas de gestión de Liberación de conexiones.

Hay dos formas de liberar una conexión. Puesto que el flujo de datos de mensaje en cada sentido se controla por separado, lo usual es que la conexión se libere en cada sentido en forma independiente. Esto se conoce como **desconexión ordenada, negociada o elegante** y requiere que ambos usuarios emitan una primitiva de solicitud **CLOSE** después de haber presentado todos sus datos al TCP local y se manifiesta en el otro extremo tal acción mediante la primitiva **TERMINATE**.

En el modo alternativo, un usuario emite la primitiva **ABORT** que hace que ambas partes desechen los datos que tienen pendientes y terminen la conexión. A este último caso se le conoce como **desconexión abrupta** (ver figura 2.6).

2.2.2. Protocolo de Transporte Internet.-

El principal protocolo definido por la capa de transporte es TCP (Transmission Control Protocol). Especifica el formato de los datos y los reconocimientos que dos computadores intercambian para manejar una transferencia fiable. Para ello, igual que X.25, provee un circuito virtual para los programas llamado conexión, pero a diferencia de X.25, una conexión puede ser establecida simultaneamente por ambos extremos.

El propósito de TCP es ofrecer conexiones entre una pareja de procesos utilizando una red basada en IP. Para especificar estas conexiones se definen los puntos finales, uno en la máquina origen y otro en la máquina destino, constituídos por las parejas formadas por los **números IP** (número que identifica la máquina dentro de la red, gestionados por la capa inferior) y **puerto** (número que identifica el destino último dentro de la máquina).

N° de conexión = (número IP, número de puerto)

N° de Conexión = (153.122.135.222 , 1050)

Para implementar este protocolo se penso, para garantizar la fiabilidad, en usar la **técnica de envio y espera de reconocimiento**, como se observa en la figura 2.7, pero en caso de excesivo tiempo sin recibir la respuesta, se asume por el protocolo que algo ha sucedido y al vencer un temporizador se retransmite la misma unidad de datos (ver figura 2.8).

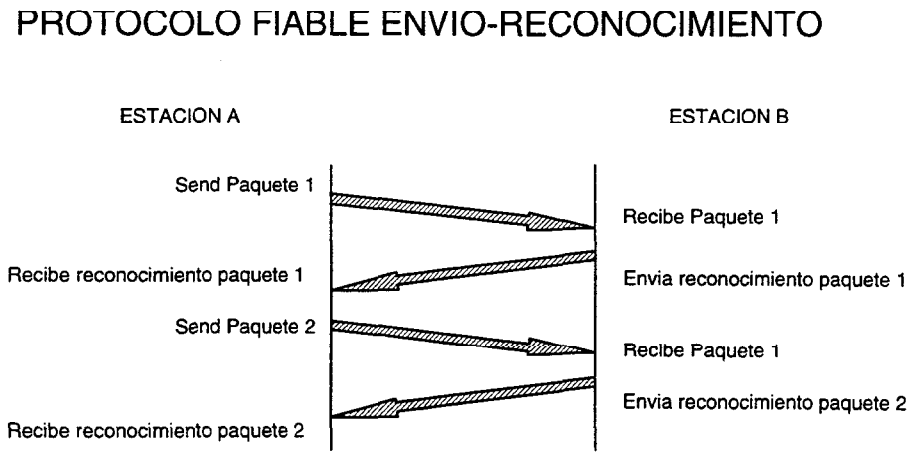


Fig. 2.7. Protocolo Seguro pero poco aprovechado el ancho de banda

al tener que esperar por cada reconocimiento.

PROTOCOLO FIABLE ENVIO-RECONOCIMIENTO
CON TEMPORIZACION PARA RETRANSMISIONES

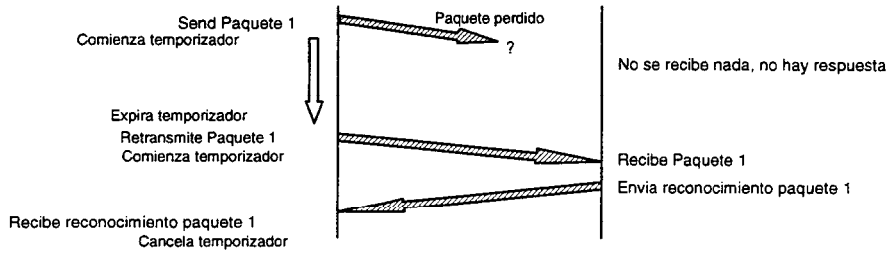


Fig. 2.8. Protocolo Fiable con temporizadores, control de perdidas y duplicidades

Además deberán enumerarse cada una de las unidades de datos para poder descartar duplicados. Como principal defecto de esta técnica, se tiene la pérdida de ancho de banda de la red, ya que mientras no se reciba el reconocimiento, el transmisor no podría seguir transmitiendo, durante los períodos de espera de los reconocimientos. Para aprovechar mejor la red, se introduce el uso de ventana deslizante, posibilitando transmitir varias unidades de datos dentro del margen definido por **la ventana** antes de recibir la confirmación de cada una de ellas (ver figura 2-9 y ver apartado 2.2.4). Finalmente se llega a implementar, como en TP4, la técnica de **ida-vuelta-ida** para eliminar las dificultades potenciales ocasionadas por paquetes antiguos que aparecieran súbitamente y pudiesen causar problemas.

PROTOCOLO FIABLE ENVIO-RECONOCIMIENTO
CON VENTANA DESLIZANTE.

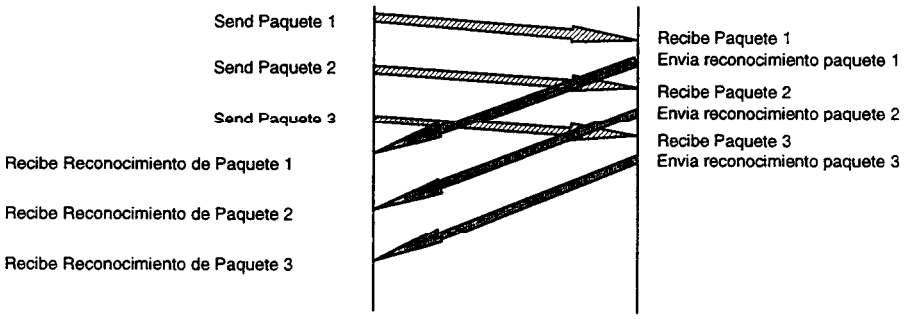


Fig. 2.9. Protocolo seguro con técnica de ventana deslizante. (Se envian

tres paquetes antes de recibir reconocimientos de ninguno de ellos)

⌦ Las funciones que deben implementarse son :

➔ ***Transferencia básica de datos:***

Una conexión TCP se comporta como un **canal bidireccional de bytes, no de mensajes**, es decir que los mensajes de solicitud/respuesta son flujos de datos independientes (**flujos de bytes**). Las entidades TCP deciden cuándo construir una TPDU, formar con ella un paquete y enviarlo por la red, a su conveniencia. Si el usuario desea el envio inmediato de un bloque de datos, se lo indica a su entidad TCP activando en el servicio SEND la señal **señal_PUSH**. El resultado es que se envía inmediatamente una TPDU. Esta circunstancia es invisible en el otro extremo.

➔ ***Fiabilidad:***

TCP debe recuperar los efectos producidos por datos dañados, perdidos, duplicados o entregados fuera de orden por IP. Lo consigue asignando un **número de orden** a cada byte que se transmite, y exigiendo un acuse de recibo (ACK) por parte de las entidades TCP receptoras. Si no se recibe un ACK en un período de tiempo prefijado, se produce una **retransmisión**. En recepción, los números de secuencia sirven para reordenar los datos entregados en desorden. Para detectar paquetes alterados durante la transmisión, se añade un checksum a cada TPDU.

➔ ***Control de Flujo:***

TCP ofrece mecanismos para que una entidad receptora gobierne el flujo de información enviado por una emisora. Se consigue con un **sistema de créditos (o ventana deslizante)**, que se conceden por bytes, no por TPDUs y que es dinámicamente ajustado durante la conexión.

➔ ***Multiplexación:***

El concepto de multiplexación tiene aquí un significado diferente al definido en OSI. Se refiere a que una entidad TCP puede dar servicio a varios usuarios simúltaneamente. Para ello, cada entidad TCP tiene una serie de TSAPs que, en este caso, se denominan **puertos**. Si concatenamos un **puerto** con una **dirección de red,** tenemos **un socket**. Cada conexión TCP viene identificada por la pareja de sockets que se comunican. La asignación de un puerto a un proceso depende del sistema operativo sobre el que se implemente TCP, aunque en todas las máquinas de la Internet se reservan ciertos puertos, bien conocidos, para aplicaciones concretas (FTP, Telnet, SMTP,e tc..). El sistema debe ofrecer mecanismos para que un proceso pueda conocer el puerto correspondiente a una aplicación dada.

➔ ***Conexiones:***

TCP ofrece un servicio orientado a conexión. Esto supone que las entidades TCP deben mantener cierta información de estado para cada conexión, entre las que se encuentran :

- los números de sockets local y remoto,
- los punteros a buffer de transmisión y recepción
- punteros a cola de retransmisión,
- valores de seguridad y precedencia para las conexiones
- los números de secuencia
- y los créditos en cada dirección.

Toda esa información se almacena en un **TCB (Transmision Control Block).** El TCB también contiene una serie de variables asociadas con los números de secuencia tx y rx.(ver tabla 1) Cuando dos procesos quieren comunicarse, sus entidades TCP deben establecer una conexión, lo que supone asignar un TCB en cada extremo. Cuando la comunicación se ha completado, la conexión se cierra y se liberan los recursos (TCBs) asignados. Puesto que debe establecerse una conexión utilizando una servicio no fiable, se utiliza un mecanismo bastante complejo de establecimiento y liberación de conexiones.

Variable	Propósito
	Variables de Secuencia de Transmisión
SND.UNA	**Envio reconocido**
SND.NXT	**Siguiente envio**
SND.WND	**Ventana de TX**
SND.UP	**Número de secuencia del último byte de datos urgentes**
SND.WL1	**Nº de secuencia usado por la última ventana actualizada**
SND.WL2	**Nº de ACK usado por última ventana actualizada**
SND.PUSH	**Nº de secuencia de último byte tx. por push**
ISS	**Nº de secuencia de transmisión inicial**
	Variables de Secuencia de Recepción
RCV.NXT	**Nº de secuencia de siguiente byte a ser recibido**
RCV.WND	**Nº de bytes que pueden ser recibidos**
RCV.UP	**Nº de secuencia de último byte de datos urgentes recibidos**
RCV.IRS	**Nº de secuencia de recepción inicial**

Tabla 1. Variables de Control de Ventana en TCB.

➔ ***Prioridad y Seguridad :***

Los usuarios de TCP pueden indicar la prioridad y la necesidad de seguridad de su comunicación. Estas características se implementan utilizando las posibilidades que da IP de seleccionar el **tipo_servicio** deseado.

Cada byte de datos transmitido por TCP tiene su propio número de secuencia privado. El espacio de números de secuencia tiene una extensión de 32 bits, para asegurar que los duplicados antiguos hayan desaparecido, desde hace tiempo, en el momento en que los números de secuencia tengan que volver a repetirse. TCP, sin embargo, si se ocupa en forma explícita del problema de los duplicados retardados cuando intenta establecer una conexión, utilizando el protocolo ida-vuelta-ida (ver apartado 2.2.2.3) para este propósito.

2.2.2.1 Formato de Unidad de Datos de Protocolo TCP.-

En la figura 2.10 se muestra el formato de la TPDU en terminología OSI que en Internet se le denomina **segmento**. Esta formado por la cabecera que se utiliza en TCP y el campo de datos (capa superior).

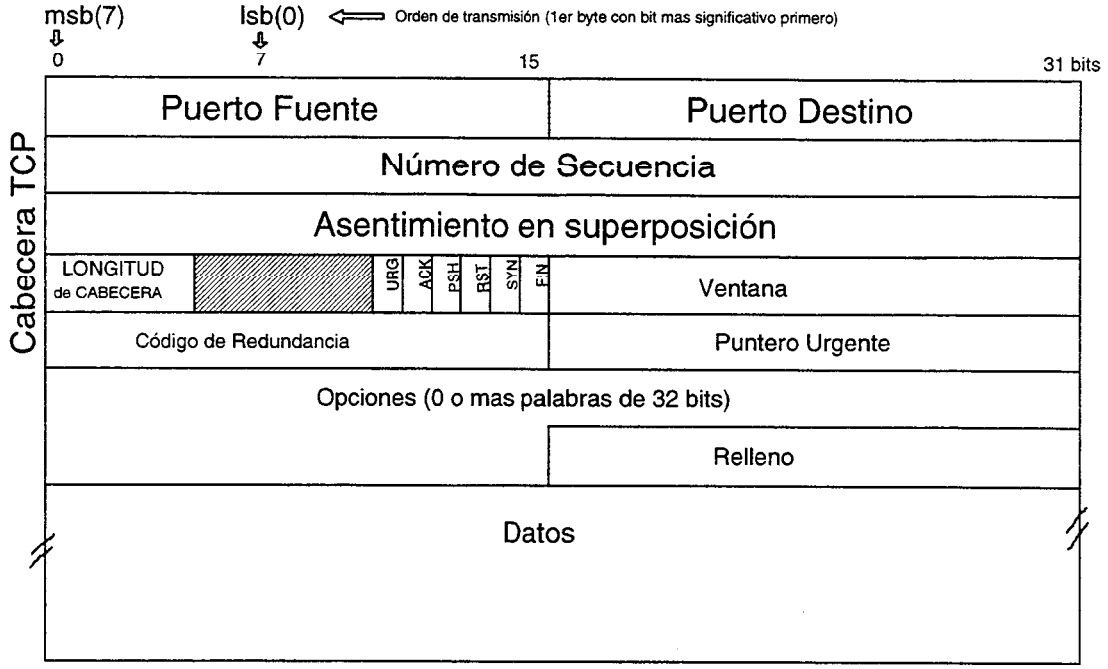


Fig 2.10. Unidad de Datos de Capa de Transporte Internet (segmento TCP)

El primer aspecto que llama la atención es que la cabecera mínima de TCP es de 20 bytes. A diferencia de la clase 4 del modelo OSI, con la cual se puede comparar a grandes rasgos, TCP sólo tiene un formato de cabecera de TPDU. Los campos representan :

Puerto Fuente y Destino : Identifican los puntos terminales de la conexión (las direcciones TSAP de acuerdo con la terminología del modelo OSI), llamados **puertos** en terminología Internet. Cada host deberá decidir por sí mismo como asignar sus puertos (16 bits). Identifican los programas de aplicación de los puntos finales de la conexión.

La dirección TCP completa se obtiene añadiendo a esta información la dirección IP, presente en el paquete IP dentro del cual va encapsulado el segmento TCP.

Número de Secuencia y Asentimiento : Realizan la función de controlar el orden de las TPDUs y los correspondientes asentimientos asociados. Representa el **número del primer byte de datos de la TPDU** (o lo que es lo mismo la posición del byte del flujo de datos del transmisor), excepto cuando el bit SYN está activo. En este caso (ver SYN) indica el número de secuencia inicial para los datos (no se empiezan a numerar los bytes desde 0), y el primer byte tiene como número el que aparece en este campo más uno. Son de 32 bits debido a que cada byte de datos esta numerado en TCP (recuérdese que es un contador de bytes y no de unidades de datos).

O sea en TCP, los números de secuencia de transmisión y de confirmación se relacionan con la posición de un byte en el flujo de mensajes completo, más que con la posición de un bloque de mensajes en la secuencia.

Concretamente, el número de secuencia indica la posición del primer byte del campo de datos del segmento respecto al principio del mensaje completo, y el número de confirmación indica el byte del flujo de datos en el sentido opuesto que el TCP transmisor espera recibir a continuación.

El número de asentimiento, si ACK está activo, indica el número de secuencia del siguiente byte que el emisor de la TPDU está esperando recibir del otro extremo.

Longitud de Cabecera TCP : Indica el número de palabras de 32 bits (n° de grupos de 4 bytes) que están contenidas en la cabecera de TCP. Esta información es necesaria porque el campo **Opciones** tiene una longitud variable, y por tanto la cabecera también, o sea sirve para identificar dónde empiezan los datos. Su tamaño es de 4 bits.

Reservado : 6 bits, normalmente a cero, sin ningún significado. Se reservan para uso futuro.

Bits de Control : Algunos segmentos transportan solo un reconocimiento mientras otros transportan datos. Otros transportan peticiones de conexión o cerrar conexiones. Este campo determina el propósito y contenido de cada segmento.

URG : Bit que indica que el **puntero acelerado (urgente)** se esta utilizando y este se emplea para indicar un desplazamiento en bytes a partir del número de secuencia actual en el que se encuentran los datos acelerados dentro de la misma unidad de datos. Esta facilidad se brinda en lugar de los mensajes de interrupción (ver figura 2.12).

ACK : Bit que activa el uso del Número de asentimiento (acuse de recibo). Siempre especifica como número de secuencia el del siguiente byte que el receptor espera recibir.

PSH : Función PUSH. Esta TPDU se ha enviado, normalmente a petición del usuario, antes de completar una TPDU del tamaño máximo acordado. Indica a la entidad transmisora que transmite la información de este segmento sin esperar a llenar sus buffers. Indica una entrega inmediata.

RST : Bit que se utiliza para reiniciar una conexión que se ha vuelto confusa debido a SYN duplicados y retardados, o a caídas de los hosts. Permite forzar una resincronización de la conexión. Todos los buffers asociados con la conexión son liberados y las entradas al TCB se borran.

SYN : Bit que se utiliza para el establecimiento de conexiones. La solicitud de conexión tiene SYN = 1 y **ACK** = 0, para indicar que el campo de asentimiento en superposición no se está utilizando. La respuesta a la solicitud de conexión si lleva un asentimiento, por lo que tiene SYN = 1 y ACK = 1. En esencia, el bit SYN se utiliza para denotar las TPDU CONNECTION request y CONNECTION confirm, con el bit

ACK utilizado para destinguir entre estas dos posibilidades. Permite la sincronización de los números de secuencia.

FIN : Bit que se utiliza para liberar la conexión; especifica que el emisor ya no tiene más datos. Después de cerrar una conexión, el proceso iniciador puede seguir recibiendo datos indefinidamente, ya que el otro extremo puede tener datos pendientes de envío.

El control de flujo en TCP se trata mediante el uso de una ventana deslizante (**créditos** en bytes) de tamaño variable que se concede al otro extremo. Es necesario tener un campo de 16 bits, porque el campo **Ventana** indica el número de bytes que se pueden transmitir por encima de los bytes asentidos por el campo ventana y no cuántas TPDU.

Código de Redundancia (checksum) : Se incluye como un factor de seguridad añadido, ya que IP no incluye checksum de datos, sino solo de cabecero. El algoritmo del código de redundancia consiste en sumar simplemente todos los datos (cabecero y campo de datos, y , si existe, el pseudocabecero), considerados como palabras de 16 bits, y después tomar el complemento a 1 de la suma. A veces es necesario añadir los suficientes ceros de relleno para conseguir la aritmética 16 bits. A veces se añade un pseudocabecero formado por las direcciones IP fuente y destino para asegurar mejor la transmisión. (ver figura 2.11).

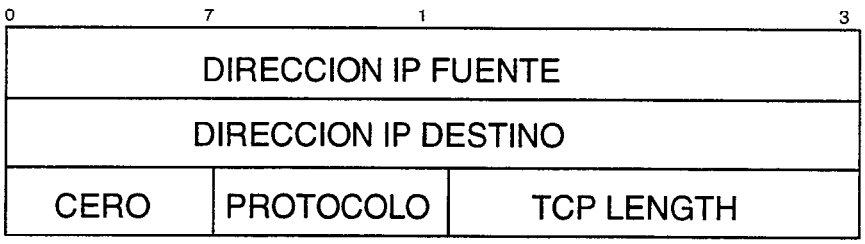


Fig. 2.11 PseudoCabecero TCP para usar en el checksum.

El **pseudocabecero** tiene los campos **TCP LENGTH** que representa la longitud total del segmento incluyendo el cabecero y **PROTOCOLO** que identifica el protocolo utilizado (por ejemplo IP transportando TCP será el valor 6).

Puntero Urgente : Si la TPDU contiene datos urgentes (indicado con el bit URG activo) este campo indica un desplazamiento desde el número de secuencia al último byte donde estan los bytes urgentes en la parte de datos (datos urgentes en-banda). El resto de los bytes de la TPDU corresponden a datos normales (ver figura 2.12).

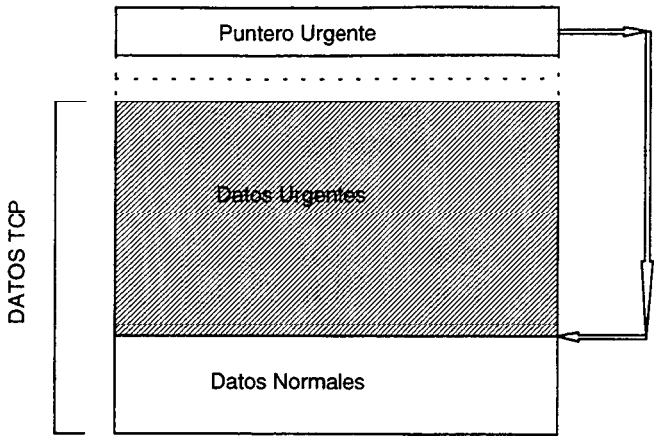


Fig. 2.12 Ubicación de los datos urgentes en el campo de datos.

Opciones : Se utiliza para diferentes cosas, por ejemplo para comunicar tamaños de tampones durante el procedimiento de establecimiento, el tamaño máximo del segmento, etc.. Su uso requiere que se forme mediante las ternas, nº de opción, longitud de opción y valor de opción (ver tabla siguiente). Estas definidos los valores:

- **0** : fin de lista de opciones,
- **1** : no operación y
- **2** : tamaño máximo de segmento.

Este campo puede aparecer o no, si aparece y su número de bits no es múltiplo de 32, se necesita añadir el campo de relleno, con tantos bits hasta alcanzar el múltiplo de 32. Podrá tener otros usos en versiones futuras de TCP (tamaño de ventana escalable, marcas de tiempo o reconocimientos selectivos), probablemente relacionados con el aumento de la velocidad de los medios de comunicación y permitir mejorar la fiabilidad de las comunicaciones a nivel de transporte.

Opción : Tipo=2	Longitud : 4	Tamaño Máximo de Segmento

Relleno : Una serie de bits de relleno para conseguir que la parte de opciones, sumada a este campo, tenga un tamaño múltiplo de palabra de 32 bits.

Datos : Bytes con datos de usuario. La longitud de este campo se sabrán a partir de la longitud del paquete de datos de la red que contiene la TPDU. El número máximo, por omisión, de bytes del campo de datos de un segmento es 536 (para el uso de Redes de Area extensa (WAN) donde la tasa de errores es elevada) frente a las Redes de Area Local, donde la tasa de error es menor, permitirá un tamaño de datos mayor.

2.2.2.2 Máquina de Estados TCP.-

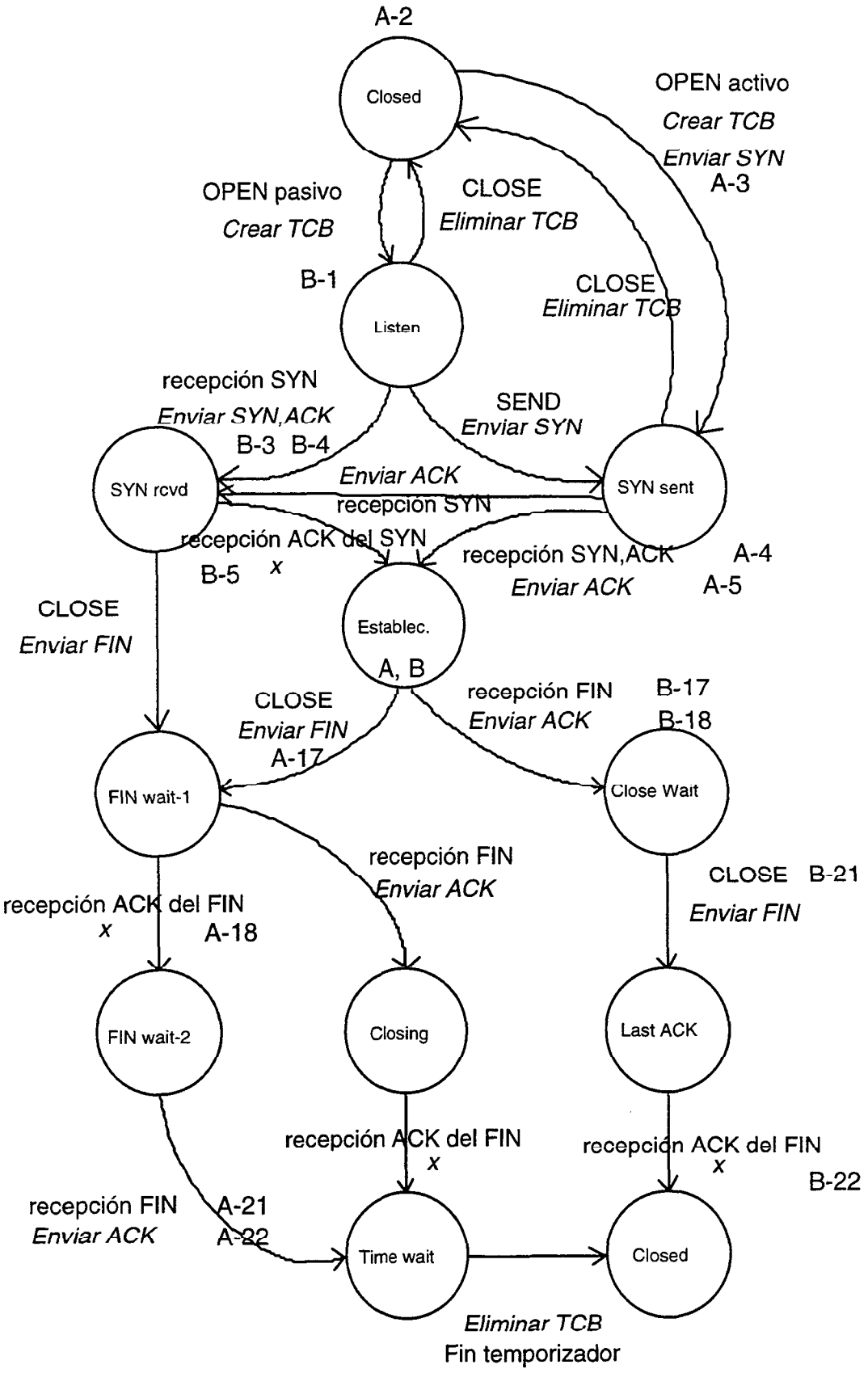


Fig. 2.13. Máquina de Estados de entidad TCP.

2.2.2.2.1 Establecimiento y Liberación de Conexiones.-

El establecimiento y la liberación de conexiones de transporte, al estar basada en una red que ofrece un servicio de baja calidad no es nada trivial, por lo cual se requiere una máquina de estado bastante depurada ya que muchas cosas pueden ir mal.

En la figura anterior (fig. 2.13) se muestra la máquina de estados para establecer y liberar conexiones, donde los círculos son los diferentes estados posibles, las flechas indican la transición entre ellos, el texto normal es el suceso que provoca la transición y el texto en cursiva representa la acciones realizadas junto con la transición del estado.

Los cuatro estados de la parte superior corresponden al proceso de establecimiento de conexión, mientras que los siete de la parte inferior corresponden al cierre. Los sucesos OPEN, CLOSE y SEND se refieren a la invocación, por parte del usuario, del servicio correspondiente. Los sucesos **recepción** corresponden a la recepción de una TPDU desde la entidad TCP par. Las acciones de **enviar** SYN, ACK y FIN indican que se envían TPDUs con los flags correspondientes activados.

Los estados válidos y sus significados son :

Listen : el usuario ha invocado el servicio OPEN pasivo, por lo que se está a la espera de una solicitud de conexión desde un socket remoto.

SYN sent : el usuario ha solicitado un OPEN activo. Se ha iniciado el proceso de establecimiento de conexión y se está a la espera de contestación.

SYN rcvd : se ha recibido una TPDU petición de establecimiento y se ha enviado un acuse de recibo y otra TPDU de establecimiento. Ahora se está a la espera del acuse de recibo de la petición de establecimiento de conexión.

Estab : representa una conexión abierta. Se pueden enviar datos por ella. Es el estado normal durante la fase de transferencia de datos.

FIN wait-1 : el usuario ha solicitado cerrar la conexión, se ha enviado a la entidad TCP para una TPDU de petición de cierre y se está a la espera de una petición de fin de conexión desde el otro extremo, o de un acuse de recibo de la TPDU de terminación enviada.

FIN wait-2 : se ha recibido el acuse de recibo de una petición de cierre de conexión, y se está esperando a que la entidad TCP remota envíe una TPDU solicitando el cierre de la conexión.

Closing : tras enviar una solicitud de cierre se ha recibido otra del otro extremo. Se continúa a la espera del acuse de recibo de la TPDU de fin de conexión que se envió a la entidad TCP remota.

Time wait : se está esperando el tiempo suficiente para que la entidad TCP esté segura de que su acuse de recibo a la solicitud de cierre ha llegado a su destino.

Close wait : se está esperando a que el usuario local pida el cierre de la conexión.

Last ack : se espera el acuse de recibo de la petición de cierre de conexión que se envió previamente a la entidad TCP remota.

Closed : este estado representa que no hay conexión.

➔ ***Establecimiento de Conexión.-***

De manera genérica el establecimiento de conexión se realiza mediante el intercambio de SYN como se muestra en la figura 2.14:

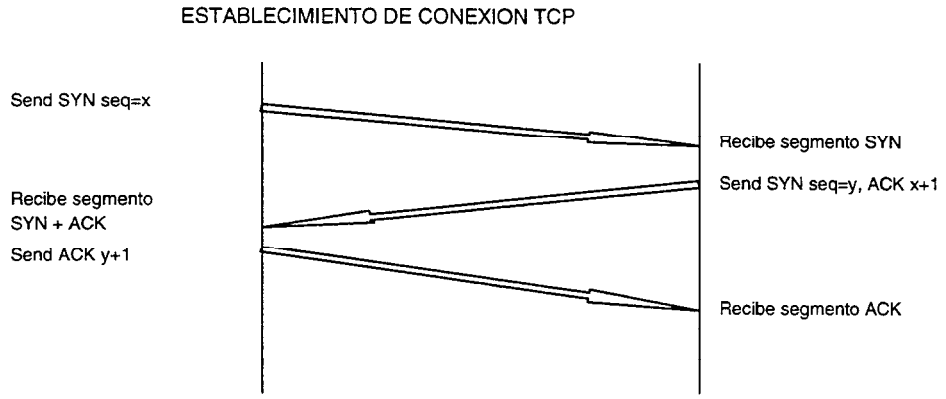


Figura 2.14. Secuencia de acciones desde el usuario del servicio para establecer conexión.

De igual forma en la figura 2.15 se muestra el intercambio de segmentos para el establecimiento de conexión indicando las primitivas del servicio y su segmento correspondiente.

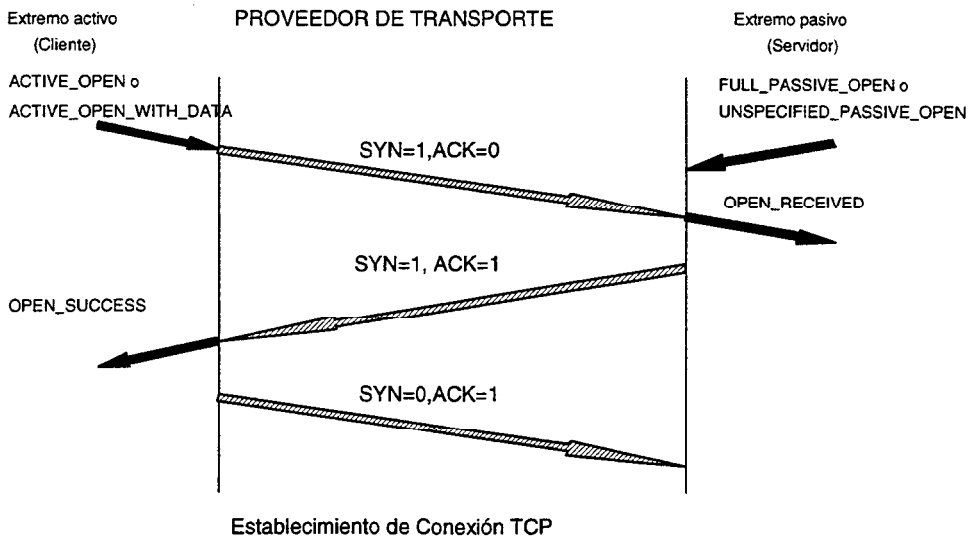


Fig.2.15. Intercambio de flags para establecer conexión de transporte (TCP).

Esta técnica de establecimiento de conexión se basa en la técnica de **saludo a tres pasos (three-way handshake).** En ella se ve que el intercambio de **SYN - SYN** inicial permite a los dos extremos ponerse de acuerdo en cuanto a los números de secuencias iniciales (uno en cada dirección) para los datos. No ocurre como en OSI, donde los protocolos de transporte empiezan la numeración siempre desde 0. La explicación de por qué no empiezan siempre desde el mismo número se encuentra en el uso de **referencias congeladas**, que se usan para evitar que TPDUs de una conexión se confundan con las de otra anterior. En OSI se conseguía no repitiendo, en un tiempo prudencial, los números de referencia. En TCP se consigue no reutilizando, en un tiempo prudencial, los números de secuencia. Veamos la secuencia que se sigue en el siguiente gráfico (figura 2.16), donde se incluyen las unidades intercambiadas, los estados, las primitivas y el número de orden de las diferentes acciones:

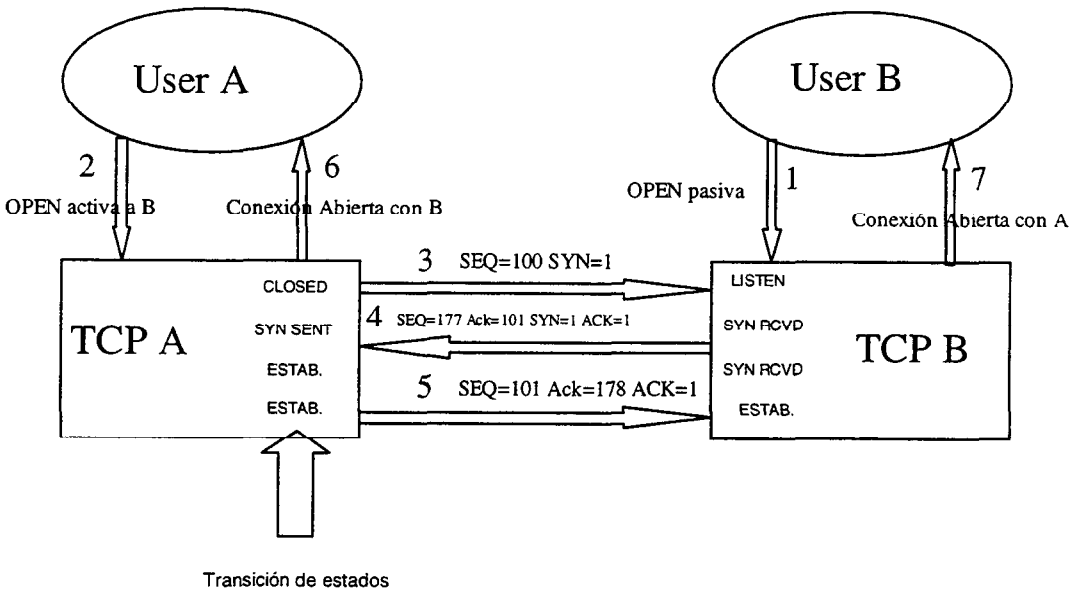


Fig. 2.16. Secuencia de sucesos y estados para establecer conexión TCP.

El usuario de TCP A ha enviado una primitiva de conexión TCP activa **(OPEN activa).** El usuario remoto ha enviado una equivalente de conexión pasiva **(OPEN pasiva).** Estas operaciones se listan como sucesos 2 y 1, respectivamente aunque cualquiera de los dos sucesos pueden aparecer primero.

Al invocar una primitiva de conexión activa requiere al TCP A para preparar un segmento con el bit **SYN puesto a 1.** El segmento es enviado a TCP B y se representa como 3 en la figura 2.16 (se corresponde con B-3 en la máquina de estados de la figura 2.13) y codificado como **SEQ=100 SYN=1.** En este ejemplo, el campo secuencia con valor 100 se usa como el número **ISS (Initial Send Sequence=**N° de secuencia inicial), aunque cualquier número pudo ser escogido dentro de unas reglas predefinidas (lo mas común es poner este valor a cero).

Una vez se recibe el segmento **SYN**, TCP B retorna un reconocimiento con número de secuencia de reconocimiento **Ack=101**. También envía su número ISS **SEQ=177**. Este suceso se etiqueta con el número 4. Una vez recibido este segmento, TCP A lo reconoce con un segmento con número de secuencia 101 (siguiente al anterior SEQ=100, nótese que al ser establecimiento de conexión solo incrementa el número de secuencia en una unidad Ack=SEQ+DATA+1 al ser DATA=0 Ack=101) conteniendo el número de reconocimiento **Ack=178**, mostrado con el número 5.

Una vez estas operaciones de **handshaking** (tras pasos) han ocurrido con los sucesos 3, 4 y 5 (denominadas **handshake tres vías**) las dos entidades TCP envían las primitivas **OPEN** (received, success o failure según sea el caso (ver figura 2.3)) a sus respectivos usuarios, como se muestra con 6 y 7. En la figura 2.13 (máquina de estados TCP) se representan con una letra A o B la entidad TCP A o B respectivamente, y al lado un número que se corresponde con el suceso que provoca la transición de estados, también indicados en la figura anterior (p.e. B-3 representa que la entidad TCP B transiciona al estado **SYN rcvd** debido al suceso 3 al recibir **SEQ=100 SYN=1**.

Una situación especial se puede dar en TCP, justamente cuando maneja simultaneamente dos conexiones activas TCP. Desde el punto de vista de la implementaciòn se requiere que la llamada contenga tanto el identificador del socket local como del socket remoto. También puede contener **precedencia, seguridad e información de temporización de usuario**. En la figura 2.17 se ilustra la secuencia de sucesos para esta situación (aperturas en ambos sentidos) :

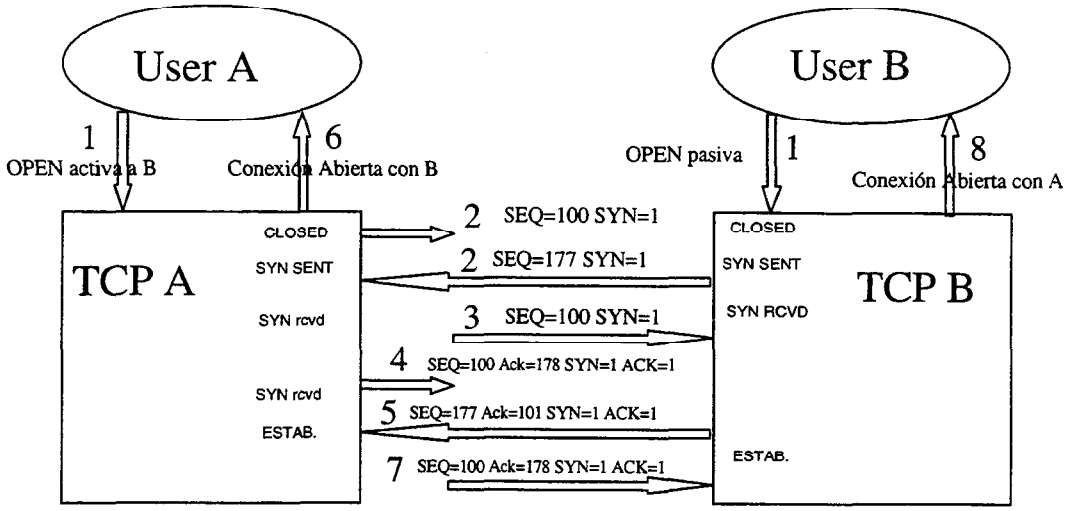


Fig.. 2.17. Aperturas simultaneas de conexión TCP.

En el caso mostrado en la figura 2.17, las llamadas son enviadas desde A y B aproximadamente al mismo tiempo. En este caso tenemos los siguientes sucesos :

Suceso 1 : Después de recibir estas llamadas, los modulos TCP crean nuevos bloques de control de transmision para contener la información de la conexión.
Suceso 2 : Los segmentos SYN son enviados desde TCP A y B aproximadamente al mismo tiempo. La posición de las flechas se usan en la figura para indicar la secuencia de tiempo relativa del tráfico. Consecuentemente, el segmento SYN desde el TCP A aún no ha llegado al TCP B cuando ha llegado el de TCP B al TCP A.
Suceso 3 : El segmento SYN desde TCP A finalmente llega a TCP B. El resultado de los segmentos en el suceso 2 mueven a los dos módulos TCP de CLOSED a SYN sent y SYN-rcvd.
Sucesos 4 y 5 : Ambos módulos TCP usan un segmento ACK, el cual, es el reconocimiento a los segmentos SYN. El segmento de TCP B en el suceso 5 llega antes que el segmento de TCP A en el suceso 4. Este es una aspecto asíncrono de TCP que es resultado de los retardos variables en internet. El retardo varia en ambas direcciones.
Suceso 6 : Una vez se recibe el reconocimiento en TCP A (de suceso 5), TCP A envia una señal de apertura a su usuario.
Suceso 7 : El segmento ACK desde TCP A finalmente llega a TCP B.
Suceso 8 : Para completar la conexión, TCP B envia una apertura de conexión a su usuario.

➔ ***Liberación de Conexión.-***

A continuación se ilustra la secuencia de acciones para la liberación de conexión (figura 2.18). La liberación se basa en una **técnica three-way handshake modificada**, para garantizar cerrar ambos lados de la conexión. Lo más peligroso en una comunicación es tener conexiones medio abiertas : un extremo cree que la conexión está abierta y el otro no lo sabe.

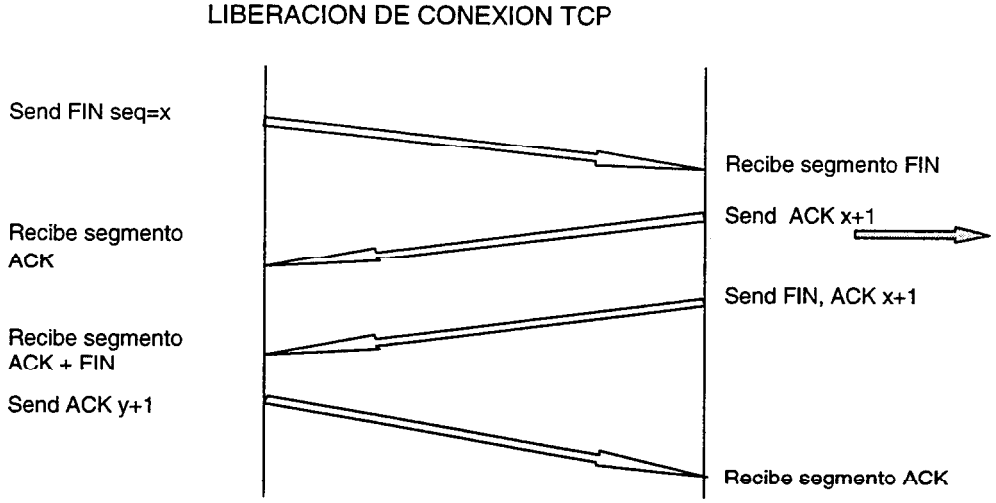


Figura 2.18 Secuencia de acciones desde el usuario del servicio para liberar conexión (TCP).

Este tipo de conexiones supone un desperdicio de recursos (TCBs ocupados, uso inútil de la red). Por este motivo, el cierre de conexiones TCP es muy cuidadoso. Aún así, algunas de las TPDUs pueden perderse y se sigue corriendo el riesgo de dejar conexiones sin cerrar completamente. En estos casos se utilizan temporizadores: si se lleva mucho tiempo sin recibir TPDUs del otro extremo, la conexión se da por cerrada

de forma unilateral (ver figura 2.19 donde se muestran las primitivas del servicio y los flags implicados en cada segmento transmitido).

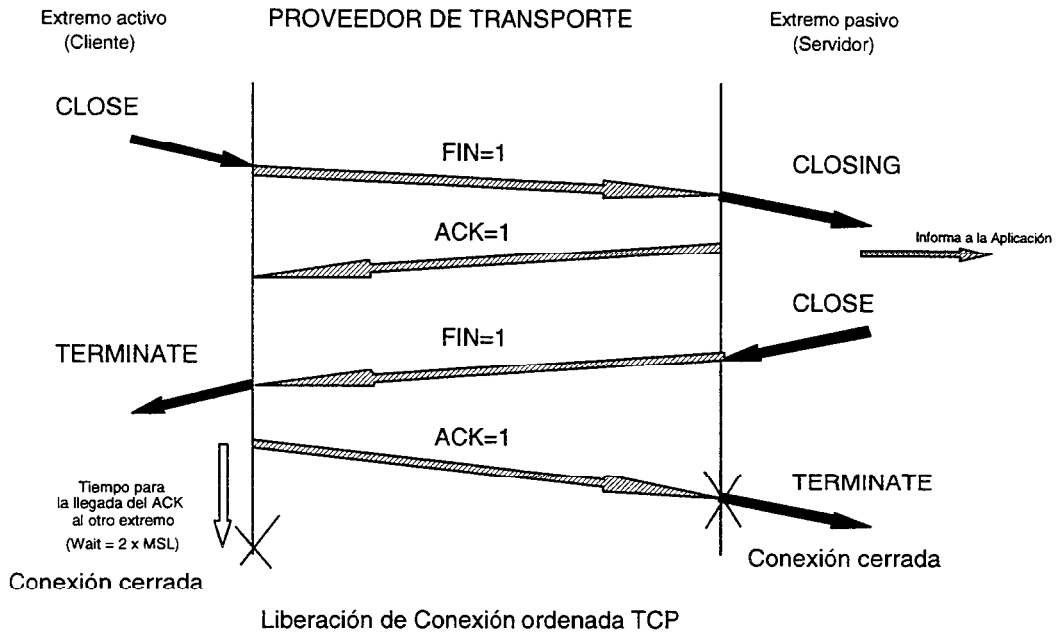


Fig. 2.19. Secuencia de segmentos de liberación de conexión de transporte (TCP).

A continuación (figura 2.20) se muestra la secuencia de estados para la liberación de conexiones TCP (Nótese que continuamos la numeración de sucesos que se harían después de la transferencia (ver apartado 2.2.2.2.2 y figura 2.22).

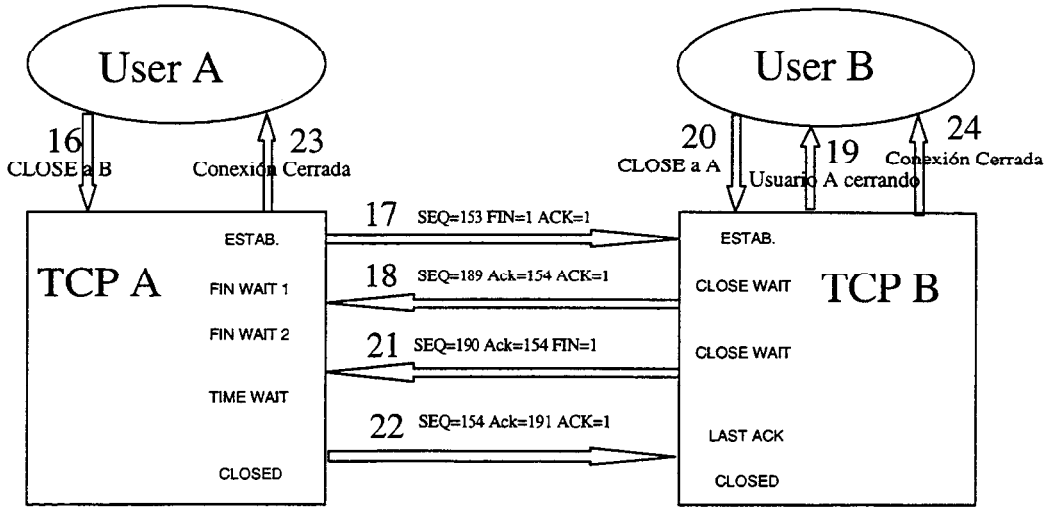


Fig. 2.20. Secuencia de sucesos para liberar conexiones TCP negociadas

En la figura 2.20 vemos que el suceso 16 ilustra que el usuario TCP A desea cerrar sus operaciones con su capa de protocolo par TCP B, mediante **CLOSE**. El efecto de esta

primitiva se muestra en el suceso 17 donde TCP A envia un segmento con el bit FIN puesto a 1 (**FIN = 1**). El número de secuencia de 153 es una continuación de la operación durante la transferencia de datos (ver figura 2.22) y es el siguiente número de secuencia del módulo TCP se requiere enviar.

El efecto de este segmento se muestra en el suceso 18 desde TCP B. TCP B reconoce el segmento **SEQ=153 FIN=1 ACK=1** de TCP A. Su segmento tiene **SEQ = 189** y **Ack = 154 ACK=1**. A continuación, se usa una primitiva de cierre a su usuario, lo cual se realiza mediante el suceso 19 **CLOSING**.

En este ejemplo, la aplicación de usuario reconoce y acuerda el cierre mediante el suceso 20 (**CLOSE a A**). La aplicación puede o no puede escoger cerrarla, dependiendo del estado de sus operaciones. Por simplicidad, se asume que el cierre se acepta mediante el suceso que se produce con el n° 20. Esta primitiva es mapeada en el suceso 21, que es el ultimo segmento de TCP B. Notese, que en el suceso 21, el flag FIN esta puesto a 1, **SEQ=190 Ack=154 FIN=1 ACK=1**. TCP A reconoce este segmento final con el suceso 22 mediante **SEQ=154 Ack=191 ACK=1** y tras una espera aproximada de 2 veces el MSL (Maximum Segment Life, generalmente 120 segundos), o sea tras 4 minutos cierra la conexión. El resultado de todas estas operaciones se muestran en los sucesos 23 y 24, donde las señales de cierre de conexión son enviadas a las aplicaciones de usuario. En el caso de que la entidad receptora quisiera enviar otro segmento y posponer el cierre, transmitiría los siguientes segmentos, antes del 21, en los sucesos indicados a continuación (nótese en 20.1, al poner FIN=0 no acepta cierre, e incluye 5 datos):

20.1 <- SEQ=190 Ack=154 ACK=1 FIN=0 DATA=5

respondiendo el iniciador del cierre, reconociendo los 5 bytes:

20.2 -> SEQ=190 Ack=195 ACK=1

luego podría cerrar el B de la misma forma que la mostrada, pero con los números de secuencia actualizados (nótese FIN=1 aceptando cierre):

21 <- SEQ=195 Ack=155 FIN=1 ACK=1
22 -> SEQ=155 Ack=196 ACK=1

Existe una segunda posibilidad de liberar una conexión, denominada **desconexión abrupta**, en cuyo caso se pierden los datos pendientes de entregar, y cuya utilidad es necesario en caso de pérdidas de sincronización o errores de temporización. Esta acción se activa poniendo el flag RST a 1 (**RST=1**) (ver figura 2.21).

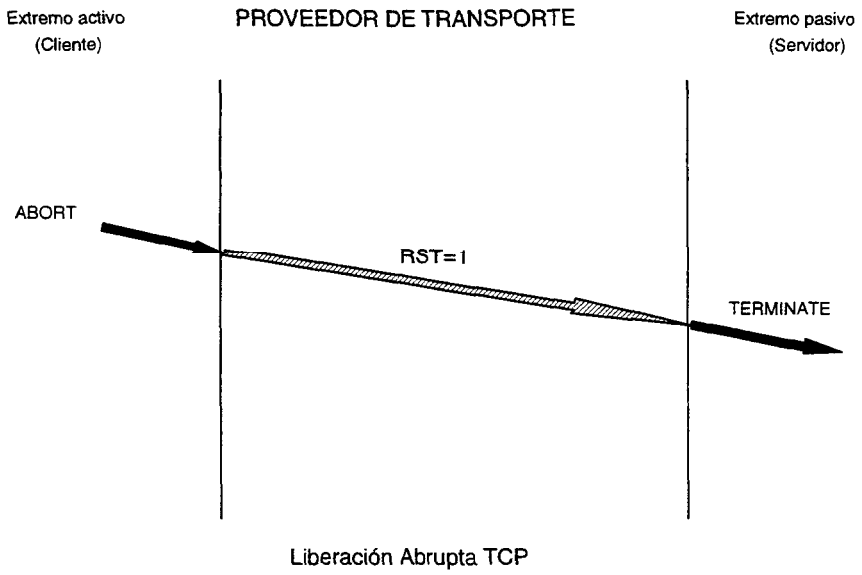


Figura 2.21 Primitivas y segmento para liberación abrupta de conexión.

2.2.2.2.2 Transferencia de Datos.-

Una vez establecida una conexión TCP puede empezar el intercambio de TPDUs (segmentos, en terminología TCP). Dada la falta de fiabilidad del servicio IP, se utilizan mecanismos de retransmisión para la recuperación de errores debidos a la llegada de segmentos dañados o perdidos. También se comprueba la posible llegada de segmentos duplicados.

Tras enviar una cierta cantidad de bytes en una o varias TPDUs, cantidad limitada por el crédito concedido por el otro extremo, se espera la llegada de una acuse de recibo, que se realiza por ráfagas: un acuse de recibo del byte N supone dar por recibidos todos los bytes anteriores (**Técnica de Retroceder N**). Si, después de transcurrir un tiempo fijado de antemano, no se recibe acuse de recibo de los datos de un segmento, se produce una retransmisión.

Los datos urgentes se transmiten de la misma forma que los normales, e incluso pueden mezclarse en un mismo segmento. Cuando un usuario entrega un bloque de datos urgentes, la entidad TCP genera una o varias TPDUs con datos urgentes. Cada una de ellas lleva un bit indicando la presencia de esta clase de datos. Las TPDUs con este bit activo llevan un campo que indica donde se encuentran los bytes urgentes que hay en el segmento; todos los demás son normales.

Con respecto al **control de flujo**, se utiliza un **sistema de créditos** como el de OSI, pero basado en bytes en vez de en TPDUs. Sin embargo, no se utiliza el mecanismo de los números de subsecuencia. Inicialmente el receptor de un segmento contiene una ventana igual a cero (cerrada) no pudiendo transmitir segmentos al transmisor, excepto ACKs y ***segmentos de prueba*** (segmento que contiene un simple byte que se usa para detectar si un host o una red no es alcanzable).

A continuación (figura 2.22) se muestra la secuencia de estados para el intercambio de datos TCP (Nótese que continuamos la numeración de sucesos usada en la figura 2.16).

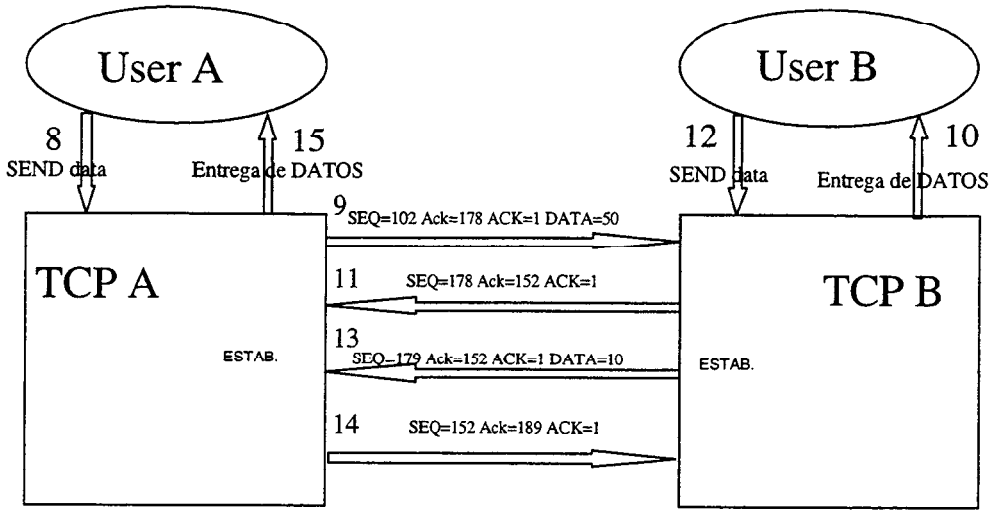


Fig. 2.22. Secuencia de sucesos durante transferencia de datos TCP

En el suceso 8, el usuario A solicita el envio de datos a TCP A para la transmisión con la primitiva **SEND**. Asumimos que 50 bytes se enviarán. TCP A encapsula estos datos dentro de un segmento y lo envia a TCP B con el número de secuencia 102 (**SEQ=102**), mediante el suceso 9 y los 50 bytes de datos en el campo de datos (**DATA=50**). Recordar que **este número de secuencia identifica el primer byte del flujo de datos de usuario**.

En el módulo TCP remoto, los datos son entregados al usuario en el suceso 10, y TCP B reconoce los datos con un segmento de reconocimiento con el número 152 (**Ack=152**), mediante el suceso 11. El número de reconocimiento 152 (Ack=SEQ+DATA) reconoce los 50 bytes transmitidos en el segmento del suceso 9. Este mismo segmento, transmitido en el suceso 11, pudo incluir datos implícitamente, por ejemplo el campo DATA=10, a diferencia de hacerlo en otro segmento como se ve en el suceso 13, el 14 sería exactamente igual.

A continuación, el usuario conectado a TCP B envía datos mediante el suceso 12. Estos datos se encapsulan en un segmento de datos y se transmiten mediante el suceso 13. El número de secuencia inicial desde TCP B fue 177 en **SYN=1**, por lo tanto TCP comienza su secuencia con 179 (**SEQ = 179**). En este ejemplo se transmiten 10 bytes.

TCP A reconoce los 10 segmentos de TCP B en el suceso 14 retornando un segmento de reconocimiento con el número 189 (**Ack=SEQ+DATA=178+10=189**). En el suceso 15, estos datos son entregados al usuario de TCP A.

A continuación se muestra la utilización del flag **PUSH** para permitir la transmisión inmediata del segmento y el uso de créditos pasados entre procesos TCP. En la siguiente tabla se muestra que al tener activado el flag PSH, se indica a la entidad TCP que

transmita inmediatamente el segmento, sin esperar a que se llene (solo se transmiten los datos inmediatos y la zona sombreada indica que no lleva datos de usuario).

CABECERO TCP	DATOS TCP
PSH=1	N bytes

Como se observa en la figura 2.23, inicialmente el módulo izquierdo al poner PSH a 1 especifica que desea que se transmitan los N bytes, que son menos del máximo permitido para el campo de datos (el segmento irá semivacío). Observar que se devuelve dicho tamaño para volver a la situación inicial. Luego el módulo de la derecha desea transmitir un mensaje de 3W bytes. Si 2W es el tamaño de la ventana y si, por ejemplo,W es el máximo tamaño del campo de datos, solo podrá transmitir dos segmentos con W bytes de datos en cada uno y esperar el nuevo crédito con las confirmaciones pertinentes, antes de proseguir transmitiendo el último segmento que le falta con los W últimos bytes del mensaje.

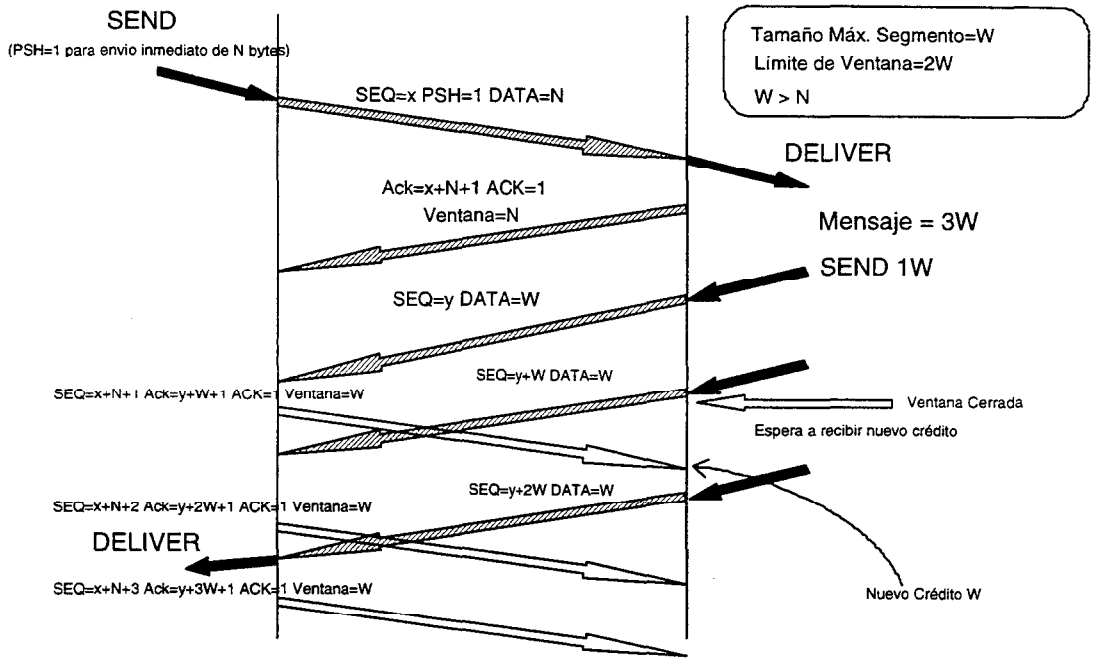


Figura 2.23. Entrega inmediata y gestión de n° de ventana.

A nivel de variables utilizadas por TCB para gestionar los números de secuencia y el tamaño de ventana se muestra un ejemplo en la figura 2.24.

Se observa secuencialmente el valor de cada una de las variables implicadas en el proceso.

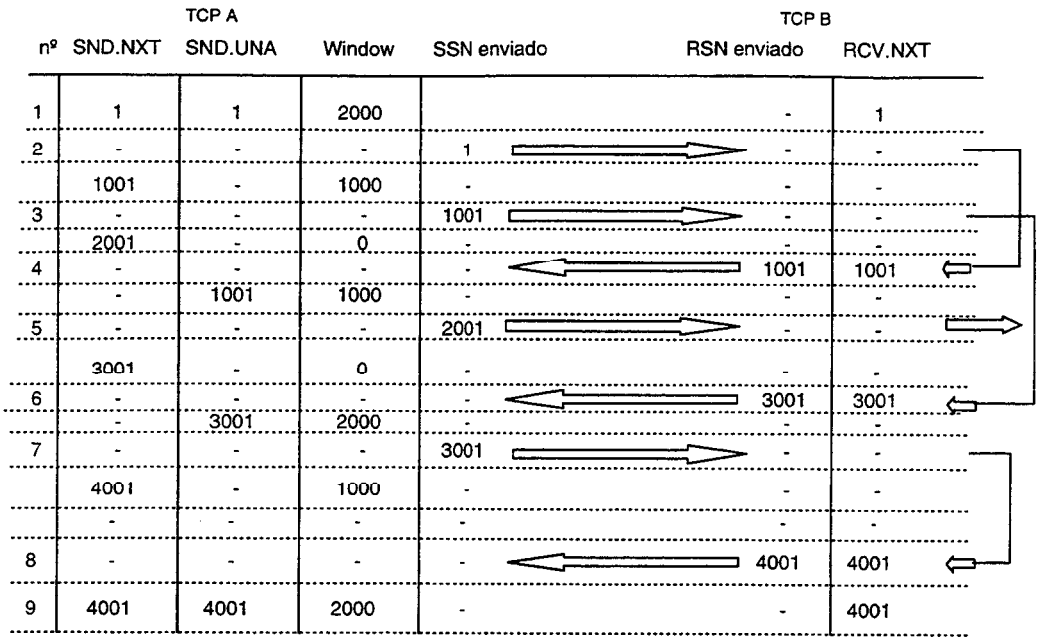


Figura 2.24. Secuencia de valores de las variables de gestión de ventana.

La variable SND.NXT se usa para asignar el número del siguiente número de envio (SSN:Send Sequence Number). La variable RCV.NXT se usa para guardar el número del siguiente SSN esperado. La variable SND.UNA se usa para guardar el reconocimiento de número de secuencia más antiguo. La variable SND.WND se usa para determinar el rango de nuevas SSNs que puede ser enviadas -entre SND.NXT y (SND.UNA + SND.WND - 1). Analizando la tabla 1 y en el apartado 2, podemos dar como ejemplo que en este caso :

- Solamente el lado A tiene datos a enviar. Los segmentos ACK son usados para reconocer SSNs. Como es normal, los reconocimientos serán devueltos con los segmentos de datos.
- El SSN seleccionado aleatoriamente es igual a 1.
- El tamaño de ventana anunciado por el lado B es 2000.
- El lado A tiene un flujo de 4000 bytes para enviar y divide el flujo en segmentos de 1000 bytes.

Cuando un segmento es enviado, el transmisor avanza SND.NXT por la longitud del segmento. Cuando el segmento es recibido, el receptor avanza RCVb.NXT por la longitud del segmento y envía un reconocimiento. Una vez se recibe el reconocimiento, el transmisor avanza el SNDa.UNA por la longitud del segmento. Las diferentes situaciones numeradas (1ª columna con indicación nº) representan :

1. Situaciones iniciales.
2. El lado A envía los números de secuencia 1 hasta 1000 (primer segmento). El SSN del segmento se obtiene desde la variable SND.NXT. La variable SND.NXT se actualiza añadiendo el actual valor de SND.NXT la longitud del segmento enviado. La ventana efectiva es reducida por la longitud del segmento enviada (2000 - 1000 = 1000).

3. Ya que la ventana esta todavía abierta, el lado A envía un segundo segmento con SSN igual a 1001. El SND.NXT se actualiza a 2001 (por la longitud del segmento) y la ventana efectiva es puesta a cero (cerrada).
4. El lado B remueve el primer segmento (con SSN=1) desde su cola. Añade el número de número de secuencia en el segmento a la variable RCV.NXT y envía un segmento ACK con el RSN = RCV.NXT (1001). El lado A actualiza su número de secuencia de reconocimiento más viejo (SND.UNA) a la primera secuencia del segundo segmento y añade el número de secuencia de reconocimiento a la ventana efectiva (0 + 1000).
5. Con la ventana efectiva ahora abierta, el lado A envía el tercer segmento con el SSN = SND.NXT (2001). La variable SND.NXT es actualizada a 3001.
6. El lado B procesa ambos segmentos 2 y 3 y actualiza la variable RCV.NXT por el número de números de secuencia en ambos segmentos (añade 2000). Envía un segmento de ACK con el RSN = RCV.NXT (3001). El lado A actualiza su SND.UNA y la ventana efectiva por el número de secuencia de reconocimiento (2000).
7. Con la ventana completa abierta otra vez, el lado A envía el último segmento con el SSN=SND.NXT (3001) y actualiza la variable SND.NXT (4001) y la ventana efectiva (reducida por 1000).
8. El lado B procesa el segmento final. Actualiza la RCV.NXT y envía un segmento ACK con RCN=RCV.NXT (4001).

El lado A actualiza su SND.UNA y la ventana efectiva (otra vez totalmente completa). Las variables SNDa.NXT, SNDa.UNA y RCVb.NXT son iguales otra vez.

Operaciones de Retransmisión.-

A continuación se ilustra otra situación donde podemos analizar las operaciones de retransmisiones.

TCP, como ya se comento, tiene una única forma de **contador de tráfico** en cada conexión. A diferencia de otros protocolos, no tiene reconocimiento negativos explícitos (NAK). Mejor que esto, se procesa sobre la entidad transmisora el uso de un temporizador y retransmisión de datos ante el no haber recibido el reconocimiento positivo (ACK).

Para entender este concepto, en la figura 2.25 se muestran 8 operaciones etiquetadas con números del 1 al 8. Cada una de estas operaciones se describen en orden.

Suceso 1. TCP de la máquina A envía un segmento de 300 bytes a la máquina B. Este ejemplo asume una ventana de 900 bytes y un tamaño de segmento de 300 bytes. El número de secuencia (SEQ) contiene el valor 3 (**número de secuencia del primer byte de los datos**).

Suceso 2. TCP de B chequea los errores en el tráfico y envía hacia atrás un reconocimiento con un valor de 303. Recuérdese, que este valor es un reconocimiento inclusivo que reconoce todo el tráfico hasta este incluyendo 302, o números SEQ 3 a través de 302. La flecha en el suceso 2 indica que el segmento de ese tráfico aún no ha

llegado a TCP A cuando el suceso 3 ocurre. (La punta de la flecha no ha conectado con A).

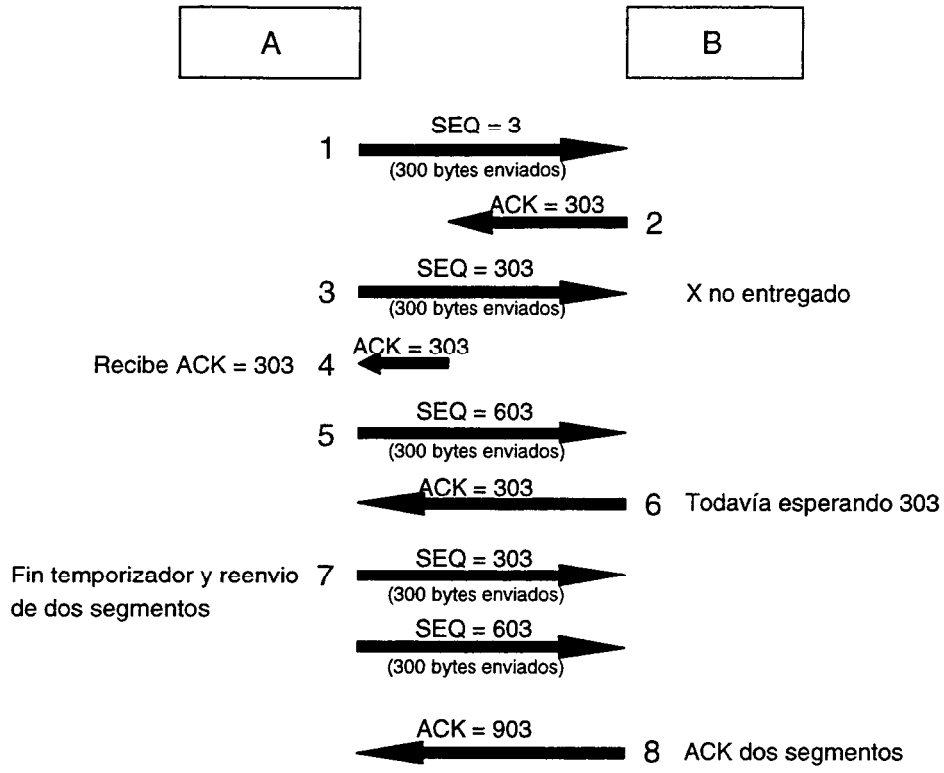


Fig.2.25. Esquemas de retransmisión TCP

Suceso 3. Como TCP A todavía tiene su ventana abierta, envía otro segmento de datos comenzando con el número 303. Por cualquier razón, este segmento de tráfico no se entrega a TCP B.

Suceso 4. El segmento de reconocimiento transmitido en el suceso 2 llega a TCP A indicando que el TCP B esta esperando un segmento comenzando con el número 303. En este momento, TCP A no sabe si el tráfico transmitido en el suceso 3 no fue entregado o simplemente aún no ha llegado debido a diferentes retardos en internet. Consecuentemente, procesa el suceso 5.

Suceso 5. TCP A envía el siguiente segmento comenzando con el número 603. Llega libre de errores a TCP B.

Suceso 6. TCP B recibe satisfactoriamente el número de segmento 603, el cual fue transmitido en el suceso 5. TCP B luego devuelve un segmento con ACK 303 porque todavía esta esperando el número de segmento 303.

Suceso 7. Eventualmente, TCP A acaba el temporizador y reenvia los segmentos para los que no haya recibido el reconocimiento. En este ejemplo, debe reenviar los segmentos comenzando con los números 303 y 603.

La operación indicada en el suceso 7 tiene sus ventajas e inconvenientes. Hace el protocolo bastante simple, ya que TCP simplemente retorna el último número de segmento sin reconocer y transmite todos los segmentos satisfactorios. Por otro lado, retransmite los segmentos si no hubo error, tales como los segmentos iniciados en 603, los cuales había sido recibidos libres de error a TCP B. Aún así, TCP opera de esta forma por la simplicidad incluso con el riesgo de degradar el throughput.

Suceso 8. Todo el tráfico es contado después de que TCP B reciba y chequee los errores de los segmentos 303 y 603 y retorne un ACK con un valor igual a 903.

2.2.3. Tabla de Conexiones TCP.-

La Base de Información de Gestión de Internet (MIB) define la tabla de conexiones TCP, que contiene información acerca de cada conexión TCP existente. La tabla consiste de cinco columnas y una fila por cada conexión.

	Estado de la Conexión	*Dirección Local*	*Puerto Local*	*Dirección Remota*	*Puerto Remoto*
Conexión 1					
Conexión 2					
Conexión 3					
Conexión 4					
.....					
Conexión i					
.....					
Conexión n					

Los campos son :

- ***Estado de la conexión*** : Describe el estado de cada conexión TCP (closed, listen ,..)
- ***Dirección Local*** : Contiene la dirección IP local para cada conexión TCP. En el estado listen, este valor debe ser 0.0.0.0.
- ***Puerto Local*** : Contiene el número de puerto local para cada conexión TCP.
- ***Dirección Remota*** : Contiene la dirección IP remota para cada conexión TCP.
- ***Puerto Remoto*** : Contiene el número de puerto remoto para cada conexión TCP.

2.2.4. Mecanismo de ventana y Control de Flujo TCP.-

Para analizar como se realiza el control de flujo en una conexión TCP nos basamos en la figura 2.26, mediante el análisis de los valores de las variables de los TCBs de cada conexión.

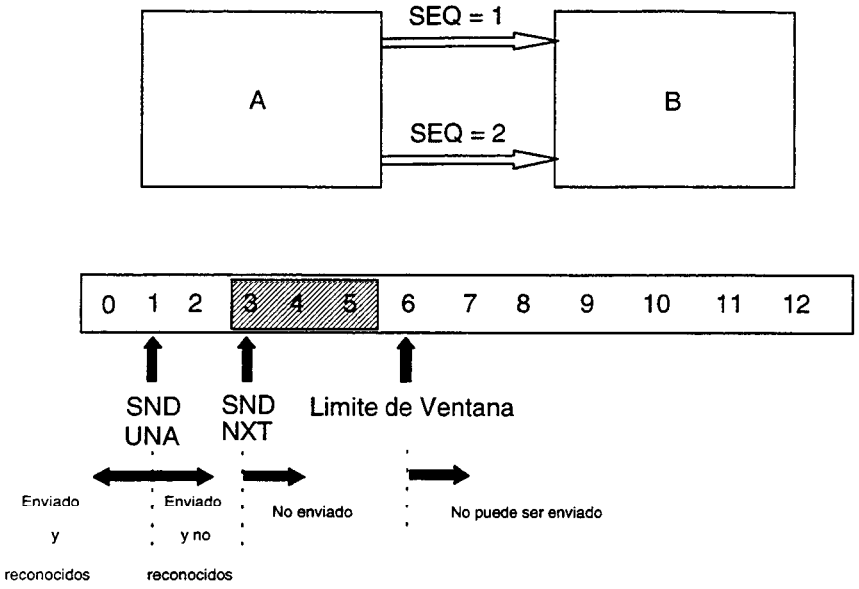


Fig. 2.26. Control de Flujo mediante ventana.

Los recuadros etiquetados como A y B representan dos módulos TCP. El módulo A esta transmitiendo dos unidades de datos, o dos bytes, al módulo B (aunque no es normal enviar solo dos bytes, en este ejemplo si, por sencillez). Estos segmentos están etiquetados como SEQ=1 y SEQ=2, ya el byte 0 se transmitió. El efecto de esta transferencia se puede ver en las variables enviadas (Ver tabla de variables de TCB) en la caja de la parte inferior de la figura. La variable **SND UNA** identifica los bytes que aún no han sido reconocidos (byte 2). También, se identifican con las flechas debajo de esta variable, los valores menores a este rango que han sido enviados y reconocidos (byte 0). Los números mayores (bytes 1 y 2) han sido enviados pero no reconocidos. La variable **SND NXT** identifica el número de secuencia del siguiente byte que puede ser enviado (byte 3). El indicador de límite de ventana es el número más grande que puede ser enviado antes de que la ventana se cierre. El valor de ventana enviado se deriva desde el valor en el campo ventana del segmento. En el recuadro de la parte inferior de la figura 23, el límite de ventana es calculado como **SND UNA + SND WND.** Este valor es 5 porque SND UNA = 2 y SND WND = 3.

Como A ha transmitido las unidades 1 y 2, su ventana de envio es 3 unidades. Eso es, A puede transmitir las unidades 3. 4 y 5. pero no la unidad 6. Esta ventana se indica en la figura 2.26 con el área subrayada.

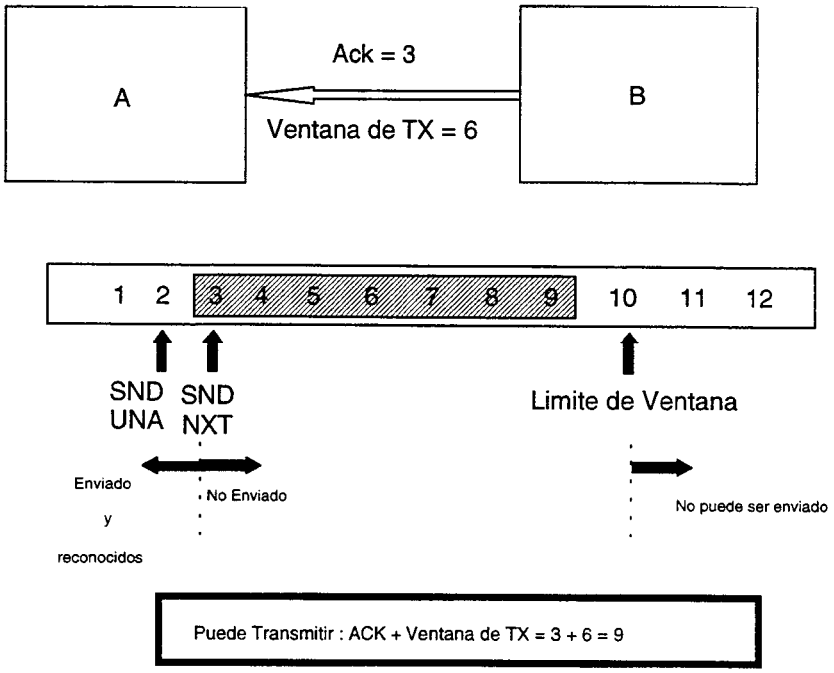


Fig. 2.27. Control de flujo mediante reconocimiento y ventana

TCP es diferente frente a otros protocolos ya que no usa el número de reconocimiento como control de ventana. Tiene un número separado transportado en el segmento TCP que incrementa o decrementa el tamaño de ventana de transmisión (ver campo de ventana en la figura 2.0). Este fenomeno se ilustra en la figura 2.27, donde B retorna un segmento a A. Este segmento contiene, además de otros campos, un campo de reconocimiento de 3 y un campo de ventana de envio con 6. El campo reconocimiento simplemente reconoce el tráfico previo (bytes 0,1 y 2). Usado solitariamente, no incrementa, decrementa, abre o cierra la ventana de A. La gestión de ventana es el trabajo del campo de ventana de transmisión. Su valor es de 6, lo que permite a A transmitir tantos bytes como sea el valor de este campo más el valor del reconocimiento. Según esto, el límite de ventana es **ACK + SND WND**. Como se observa en la parte inferior de la figura 2.27, el límite de la ventana es 9 (3 + 6), lo cual indica que se podrán enviar los bytes entre el 3 y el 9 al expandirse el tamaño de ventana a 6. El byte 10 no podrá transmitirse hasta que se reconozcan los anteriores o se amplie dicha ventana. La ventana es así expandida como se indica en el área subrayada de la figura 2.27.

El tamaño de la ventana puede ser reducido por el computador B. El campo de ventana de transmisión permite que la ventana sea expandida o reducida tanto como sea necesario para manejar el espacio de buffer y su proceso. Esta característica es más flexible que el uso del campo de reconocimiento para ambas operaciones de control de ventana y reconocimiento de tráfico.

TCP esta habilitado para enviar un segmento de datos urgente incluso si su ventana de transmisión este cerrada. Este segmento contiene el bit URG (bit de urgente) puesto a 1 si los datos urgentes necesitan ser transmitidos.

2.2.5. Temporizadores.-

Uno de los más importantes y complejos aspectos de TCP es la forma de manejar y definir los valores de los temporizadores.

Los diferentes temporizadores especificados en TCP son :

- Temporizador de Retransmisión,
- Temporizador de Persistencia,
- Temporizador de Silencio,
- Temporizador de Actividad/Inactividad

Temporizador de Retransmisión.-

Como se recordará, cada vez que se transmite un segmento, TCP arranca un temporizador y espera por reconocimiento. Si el temporizador expira antes de que los datos en el segmento hayan sido reconocidos, TCP asume que el segmento se ha perdido o estropeado y retransmite el segmento.

Para definir este temporizador, se podría pensar en una técnica de **tiempo de round-trip RTT,** la cual consiste simplemente en sustraer el tiempo en el que el segmento es enviado del tiempo en el que el reconocimiento es recibido. El problema surge que si ha habido una retransmisión de un segmento, implica transmitir el mismo segmento, no se sabe al recibir el reconociminto a cúal de los dos segmentos transmitidos se refiere, por lo que existirá ambiguedad(*).

El diseño de los temporizadores, por lo tanto no es tan trivial, ya que depende de los retardos de los equipos y de las subredes utilizadas, y en concreto debido a :

- el retardo de la recepción de reconocimientos desde el host receptor varia en una internet,
- los segmentos enviados desde el transmisor que puedan ser enviados en la internet, obviamente invalidan cualquier análisis de retardo por **round-trip** estimado por un reconocimiento espureo,
- reconocimientos desde el receptor pueden también perderse, lo cual invalida aún más la estimación de retardo round-trip.

Por estos problemas TCP no usa un temporizador de retransmisión fijo, sino **adaptativo** (algoritmo de retransmisión adaptativo) derivado del análisis de retardos encontrados en la recepción de los reconocimientos remotos.

> ***Para acomodar la variación de los retardos encontrados en un entorno de internet, TCP usa un algoritmo de retransmisión adaptativo que monitoriza los retardos sobre cada conexión y ajusta su parámetros de temporización según estos datos.***

➢ Un tiempo de round-trip se deriva de añadir el retardo de envio (SD : Send Delay), más el tiempo de proceso (PT : Processing Time) en el host remoto y el retardo de

recibo (RD: Receive Delay). Al ser variable los retardos en internet, este mecanismo es poco válido.

$$RTT = SD + PT + RD$$

La solución adoptada fue la propuesta por Phil Karn y conocido como **algoritmo de Karn**. Este evita la ambiguedad(*) anterior de los reconocimientos. Se basa en que el transmisor combine la **técnica de retransmisión por temporización** con una estrategia de **timer de backoff** en el caso de realizar retransmisiones. En el caso normal basta con round-trip. Esta técnica de back-off consiste en aplicar un temporizador inicial (**timer**) y mediante la siguiente fórmula (**fórmula de karn**):

$$nuevo\_timer = \bullet \ x \ timer$$

Si el temporizador expira y causa una retransmisión, TCP incrementa el temporizador por un factor • **(factor de multiplicación (generalmente un valor de 2 o una tabla de valores))** aplicando el nuevo temporizador (**nuevo_timer**).

Esta característica es bastante similar al algoritmo de **back-off** de Ethernet, salvo en que este se basa en un **back-off exponencial** ya que el tráfico es incrementado por las colisiones, y debido a ellas se debe introducir un retardo aleatorio en cada intento de usar el canal de comunicaciones.

Las últimas implementaciones de TCP se basan en la ventaja de la **distribución de Poisson** y la utilización de factores de Round Trip Time en la red, pero requiere un cálculo computacional mucho más elevado, lo cual provoca un retardo de proceso mayor.

Temporizador de Persistencia.-

Durante el intercambio de datos en TCP, es posible que la ventana receptora sea exactamente 0 y un segmento que reabra la ventana se pierda, en tal caso, ambos TCPs entraran en un buble infinito dc espera de uno por el otro. Para evitar esto, se define el **Temporizador de Persistencia**, que provoca enviar pequeños segmentos TCPs (1 byte) en función de este timer para chequear si la parte receptora esta lista otra vez. Si, como antes, el tamaño de ventana es 0, un reconocimiento negativo se retorna; si el tamaño de ventana es mayor, mayor número de datos podía ser enviado después del reconocimiento positivo.

Temporizador de Silencio.-

Los diseñadores de TCP fueron muy cuidadosos con las personas que deseaban prevenir cualquier posible confusion de conexiones debidas a segmentos TCP fuera de banda deambulando por la red. Así, después de que la conexión TCP se cierre, los números de puerto son únicamente liberados después de que un intervalo de tiempo fijo, justo del doble del **tiempo máximo de vida del segmento** (**MSL**) haya transcurrido. El MSL corresponde del tiempo en el campo **TTL(Time To Live)** usado por IP. En entornos

UNIX, una nueva conexión puede ser configurada después de aproximadamente 30 segundos.

Temporizador de Keep_Alive e Inactividad.-

Estos timers no fueron previsto en las especificaciones iniciales de TCP. El primero de ellos tiene el efecto de que un paquete vacío es enviado a intervalos regulares para chequear que la conexión a otro módulo TCP todavía esta activa. Si el módulo TCP par no responde, la conexión se reseteada después de que este temporizador expire. Una aplicación activa este timer con la opción KEEP_ALIVE a través del interface socket.

Algunos valores comunes de estos temporizadores son :

Retransmisión	Dinámico
Persistencia	5 segundos
Silencio	30 segundos
Keep_alive	45 segundos
Inactividad	360 segundos

2.2.6. Comparativa con capa de transporte de RM-OSI.-

En la siguiente gráfica se muestran algunas diferencias entre TP4 y TCP :

Característica	**OSI TP4**	**TCP**
Número de tipos de TPDU	9	1
Conexión	2 conexiones	1 conexión
Formato de direcciones	No está definido	32 bits
Calidad de Servicio	Extremo Abierto	Opciones específicas
Datos del usuario en CR	Permitido	No permitido
Flujo	Mensajes	Bytes
Datos importantes	Acelerados	Acelerados
Superposición	No	Si
Control de Flujo Explícito	Algunas veces	Siempre
Números de subsecuencia	Permitidos	No permitido
Liberación	Abrupta	Ordenada

① Se destaca que TP4 utiliza **9 tipos diferentes de TPDU**, en tanto que **TCP sólo tiene 1 tipo**. Esta diferencia trae como resultado que TCP sea más sencillo, pero al mismo tiempo también necesita una cabecera más grande. porque todos los campos deben estar presentes en todas las TPDU. El mínimo tamaño de la cabecera TCP es de 20 bytes; el mínimo tamaño de la cabecera TP4 es de 5 bytes. Los dos protocolos permiten campos opcionales, que pueden incrementar el tamaño de las cabeceras por encima del mínimo permitido.

② Una segunda diferencia es con respecto a lo que sucede cuando los dos procesos, en forma simultanea, intentan **establecer conexiones** entre los mismos dos TSAP (es decir, una colisión de conexiones). Con TP4 se establecen dos conexiones duplex

independientes; en tanto que con TCP, una conexión se identifica mediante un par de TSAP, por lo que solamente se establece una conexión.

③ Una tercera diferencia es con respecto al **formato de direcciones** que se utiliza. TP4 no especifica el formato exacto de una dirección TSAP; mientras que TCP utiliza números de 32 bits.

④ El concepto de **calidad de servicio** también se trata en forma diferente en los dos protocolos, constituyendo la cuarta diferencia. TP4 tiene un mecanismo de extremo abierto, bastante elaborado, para una negociación a tres bandas sobre la calidad de servicio. Esta negociación incluye al proceso que hace la llamada, al proceso que es llamado y al mismo servicio de transporte. Se pueden especificar muchos parámetros, y pueden proporcionarse los valores: deseado y mínimo aceptable. A diferencia de esto, TCP no tiene ningún campo de calidad de servicio, sino que el servicio subyacente IP tiene una campo de 8 bits, el cual permite que se haga una elección a partir de un número limitado de combinaciones de velocidad y seguridad.

⑤ Una quinta diferencia es que TP4 permite que los **datos del usuario** sean transportados en la TPDU **CR**, pero TCP no permite que los datos del usuario aparezcan en la TPDU inicial. El dato inicial (como por ejemplo, una contraseña), podría ser necesario para decidir si se debe, o no, establecer una conexión. Con TCP no es posible hacer que el establecimiento dependa de los datos del usuarios.

Estas cinco diferencias se relacionan con la fase de **establecimiento de la conexiòn**. Las siguientes se relacionan con la **fase de transferencia.**

⑥ Una diferencia básica es el modelo de transporte de datos. El modelo TP4 es el de una serie de **mensajes ordenados** (correspondientes a las TSDU en la terminología OSI). El modelo TCP es el de **flujo continuo de bytes**, sin que haya ningún límite explícito entre mensajes. En la práctica, sin embargo, el modelo TCP no es realmente un flujo puro de bytes, porque el procedimiento de librería denominado *push* puede llamarse para enviar todos los datos que estén almacenados, pero que todavía no se hayan transmitido, sin esperar a llenar al segmento Cuando el usuario remoto lleva a cabo una operación de lectura, los datos anteriores y posteriores al *push* no se combinarán, por lo que, en cierta forma un *push* podría pensarse como si definiera una frontera entre mensajes.

⑦ La séptima diferencia se ocupa de cómo son tratados los datos importantes que necesitan de un procesamiento especial (como los caracteres BREAK). TP4 tiene dos flujos de mensajes independientes, los datos normales y los **acelerados multiplexados** de manera conjunta. En cualquier caso, únicamente un mensaje acelerado puede estar activo. TCP utiliza el campo *Acelerado* para indicar que cierta cantidad de bytes, dentro de la TPDU actualmente en uso, es especial y debería procesarse fuera de orden.

⑧ La octava diferencia es la ausencia del concepto de **superposición** en TP4 y su presencia en TCP. Esta diferencia no es tan significativa como al principio podría parecer, dado que es posible que una entidad de transporte ponga dos TPDU, por ejemplo, DT y AK en un único paquete de red.

⑨ La novena diferencia se relaciona con la forma como se trata el **control de flujo**. TP4 puede utilizar un esquema de crédito, pero también se puede basar en el esquema de ventana de la capa de red para regular el flujo. TCP siempre utiliza un mecanismo de control de flujo explícito con el tamaño de la ventana especificado en cada TPDU.

①⓪La décima diferencia se relaciona con este **esquema de ventana**. En ambos protocolos el receptor tiene la capacidad de reducir la ventana en forma voluntaria. Esta posibilidad genera problemas potenciales problemas, por ejemplo, si se permite una ventana grande y su respuesta subsiguiente llegan en un orden incorrecto. En TCP no hay ninguna solución para este problema; en tanto que en TP4 este problema se resuelve por medio del **número de subsecuencia** que está incluido en la respuesta, permitiendo de esta manera que el emisor determine si la ventana pequeña siguió, o precedió, a la más grande.

①① En cuanto a la **fase de liberación** las diferencias son : TP4 utiliza una desconexión abrupta en la que una serie de TPDU de datos pueden ser seguidos directamente por una TPDU DR. Si las TPDU de datos se llegarán a perder, el protocolo no los podría recuperar y la información, al final se perdería. TCP utiliza una comunicación de **ida-vuelta-ida** para evitar la pérdida de datos en el momento de la desconexión. El modelo OSI trata este problema en la capa de sesión. Las versiones de Estados Unidos de TP4 y las internacionales son diferentes, porque la Oficina de Normalización USA no estaba de acuerdo con la existencia de TPDU extras para controlar la desconexión sin que hubiera pérdida de datos.

2.2.7. Resumen.-

- TCP (Transmission Control Protocol)
- Protocolo seguro orientado a conexión
(establecimiento, transferencia y liberación de conexión).
- Conexión full-duplex entre dos máquinas única extremo-extremo.
- Unidades intercambiadas llamadas segmentos.
- Formato único de unidades de datos. Bits de control especifican servicio.
- Puntos Finales definen conexiones = (nº IP , Puerto).
- Soporta datos fuera de banda (puntero urgente) dentro del segmento normal..
- Soporta mecanismo de push de transmisión (volcado completo).
- Conexión mediante saludo a tres pasos (ida-vuelta-ida).(three-way Handshake).
- Sistema de transferencia con numeración de secuencia a nivel de bytes.
- Sistema piggyback de reconocimientos a nivel de bytes.
- Temporizaciones mediante round-trip y back-off ante retransmisiones.
- Control de Flujo mediante Ventana deslizante y gestión de buffer implícita por el receptor.
- Control de errores mediante técnica de retroceder N.

2.2.8. Referencias.-

Para completar este análisis de TCP es necesario saber donde recurrir para profundizar en el conocimiento de este protocolo entre los diferentes RFC relacionados con el.

A continuación se indican cuales :

RFC	**AUTOR**	**Aspectos Definidos**
793	Postel	Servicio y Protocolo
1122	Braden	Servicio y Protocolo
813	Clark	Gestión de ventana TCP
816	Clark	Aislamiento ante fallos y recuperación.
879	Postel	Tamaño max. de segmento TCP
896	Nagle	Congestión en redes TCP/IP
1987	Karn & Partridge	Estimación tiempo round-trip y algoritmo de Karn
1988	Jacobson	Control de congestión
1975	Tomlinson	Handshake tres vías
889	Mills	Retardos round-trip en internet

UDP
(User Datagram Protocol)

CAPA DE TRANSPORTE INTERNET.

2.3 UDP (User DatagramProtocol)

2.3.1.Conceptos Generales.-

Este protocolo esta definido en la RFC768 de las especificaciones para Internet. Proporciona un servicio no orientado a conexión para los procedimientos de la capa de aplicación. Según esto último, es un servicio no seguro, lo cual implica que la entrega y la protección frente a duplicados no están garantizadas. Por el contrario el volumen de protocolo adicional se ve reducida en gran medida así como existen protocolos de aplicación en los que la información adicional de establecimiento y mantenimiento de datos relacionados con la conexión no están justificados o son contraproducentes. Ejemplos de este tipo de tareas son:

- **Recolección de datos** que supongan una actividad periódica o muestreo de datos pasivo. Por ejemplo en la toma de datos en tiempo real (p.e. desde sensores) la pérdida de uno de los datos no causaría ningún desastre ya que la siguiente muestra llegaría un momento después.
- **Difusión de datos** a múltiples usuarios de la red, cambios de configuración general, etc.
- **Petición-Respuesta** donde las aplicaciones proporcionan un servicio de transacción a usuarios distribuidos y se atienden simplemente tras la petición y la respuesta asociada.
- **Aplicaciones en tiempo real** como sonido y telemedida requieren utilizar redundancias y su transmisión instantánea, no requiriendo servicios como los de retransmisión.

Al ser un protocolo no orientado a conexión tiene muy pocas cosas que hacer. Es equivalente al protocolo de transporte OSI no orientado a conexión diseñado específicamente para trabajar sobre redes seguras (de tipo A, en términos del modelo OSI) y se basa en el uso de simples unidades de datos. Simplemente incorpora a las capacidades IP un direccionamiento a puerto (identificativo de aplicación).

El formato de la cabecera UDP se muestra en la figura 2.3.1. En ella se observan los siguientes campos:

- **Puerto Origen y Puerto Destino**: Dos campos de 16 bits cada uno que identifican exactamente las dos aplicaciones comunicantes.
- **Longitud** : Indica el tamaño del segmento UDP entero, es decir cabecero más datos. Esta campo se necesita al ser de tamaño variable el campo de datos.

- **Suma de Verificación**: Implementa el mismo algoritmo de TCP como medida de seguridad adicional. Representa la suma del segmento UDP más un pseudo-cabecero (mismo que TCP) que se incorpora a la cabecera UDP.

Si se detecta un error, el segmento se descarta sin tomar ninguna medida adicional. Si bien su uso es opcional, en cuyo caso, se pone todo a cero, es necesario indicar que la suma de verificación utilizada por el protocolo IP solo se aplica al cabecero IP, por lo que de no utilizarse, el campo de datos no tendrá ninguna comprobación.

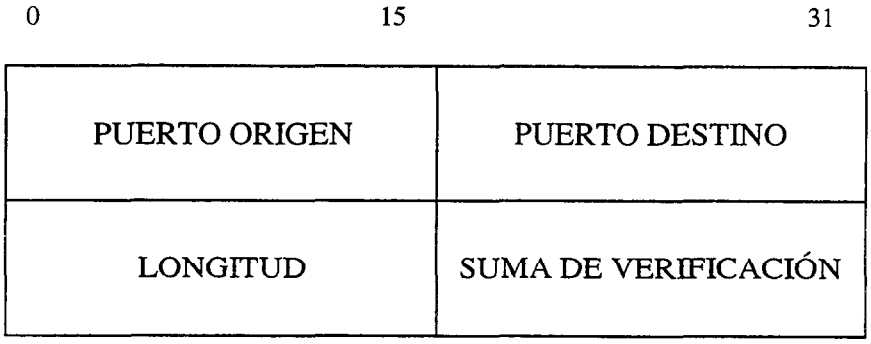


Fig. 2.3.1 Cabecera UDP

Capa de Transporte Novell

3.Protocolos de Transporte Netware.

3.0 Introducción

Dada la enorme variedad de protocolos e interfaces de programación asociados a Netware, existe una gran correspondencia entre Netware y sus tecnologías relacionadas, y el Modelo de Referencia OSI.

Aunque los protocolos de Netware son modulares y estratificados, los diseñadores de la arquitectura de Netware se centraron en suministrar un alto nivel de funcionalidad sin una estricta adhesión a ningún modelo estándar existente. Por tanto, la serie de protocolos no encaja plenamente en los siete niveles de ISO.

Sin embargo, dado que todas las redes de computadores deben hacer frente a temas y métodos similares, los protocolos de Netware deben asignarse o agruparse como se muestra en la siguiente figura 3.0:

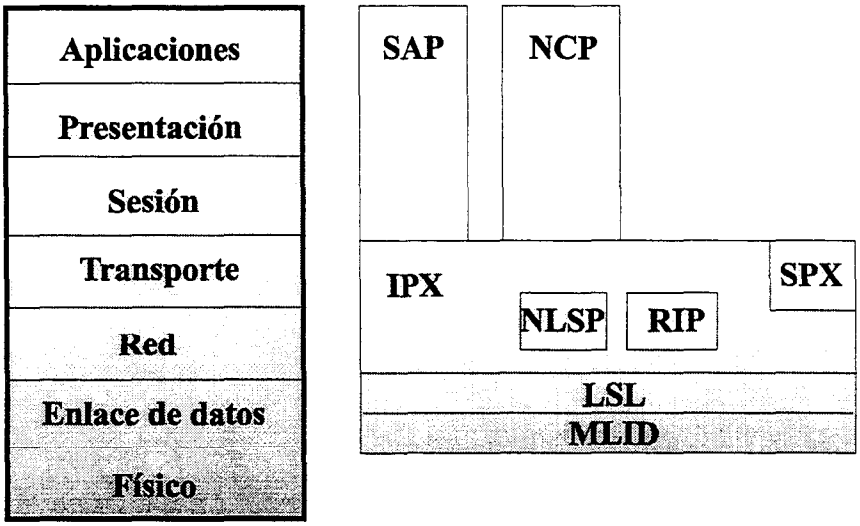


Fig. 3: Netware-OSI. Comparación.

A continuación, se resumen brevemente los protocolos clave de Netware:

- **Controlador de interfaz de enlaces múltiple (MLID)** es el nombre que utiliza Novell para un controlador de tarjeta de interfaz de red.

En concreto, cada MLID es un componente de software (generalmente creado por el fabricante de la tarjeta) que se ajusta a la arquitectura Interfaz de enlace de datos abierta (ODI) de Novell.

Los MLID no están vinculados directamente con un stack de protocolos individual, por lo que pueden estarlo a varios stacks simultáneamente.

- **Nivel de soporte de enlace (LSL)** es la interfaz entre un MLID y los distintos stacks de protocolos del nivel superior.

La LSL interpreta el campo de identificación de protocolo de cada paquete y pasa el paquete al stack de protocolos correspondiente.

- **Protocolo de Intercambio de paquetes de Interred (IPX)** es un protocolo de nivel de red sin conexión (datagrama) que se basa en el Internetwork Datagram Protocol (IDP) de XNS.

Como protocolo de nivel de red, IPX direcciona y encamina paquetes de una ubicación a otra de una interred IPX. Para ello, IPX lleva a cabo funciones de direccionamiento de red lógica y encaminamiento de interred.

Se basa en la dirección del dispositivo físico del hardware subyacente para realizar direccionamiento de nodo y en direcciones de servicios de nivel superior (denominados 'sockets') para direccionar el destino último de los paquetes dentro del nodo.

Además, IPX realiza selecciones de ruta basándose en la información de accesibilidad de red compilada por **RIP** (Protocolo de información del router).

- **Protocolo de Información del router (RIP)** también se ha derivado de XNS. RIP en un protocolo de descubrimiento de ruta de vector de distancia que utiliza un recuento de saltos para determinar el coste.

Aunque RIP se implementa como servicio (de hecho, dispone de su propia dirección de servicio), se basa directamente en IPX y realiza funciones de nivel de red.

- **Protocolo de servicios de enlace de Netware (NLSP)** es un protocolo de descubrimiento de ruta de estado de enlace diseñado a partir de un protocolo de encaminamiento, desarrollado por una organización de estándares ISO llamada Sistema intermedio a sistema intermedio (IS-IS), Dadas sus características inherentes de estado de enlace, NLSP admite las redes de gran tolerancia de malla y malla mixta.

- **Protocolo de intercambio de paquetes secuenciales (SPX)** es una ampliación de IPX que permite la transmisión de paquetes de nivel de transporte orientada a la conexión. Se basa en el Protocolo de paquetes secuenciales Xerox y mejora el protocolo proporcionando una transmisión fiable.

Los circuitos virtuales de SPX se denominan **conexiones** y cuentan con identificadores de conexión específicos (ID de conexión) que se sitúan en la cabecera de SPX. Pueden incluirse varios ID de conexión a un solo proceso de nivel superior. SPX asegura una transmisión fiable mediante la retransmisión de la información que no se recibe correctamente.

- **Protocolos centrales Netware (NCP):** Muchas funciones de aplicación de Netware (por ejemplo, servicios de archivos, servicios de impresión, gestión de nombres, bloqueo de archivos y sincronización) están disponibles para los usuarios a través de los Protocolos centrales Netware (NCP).

NCP se compone de numerosas llamadas de funciones que admiten servicios de red. El software de cliente de Netware se ejecuta en la estación de trabajo para proporcionar acceso transparente a archivos e impresoras a los clientes mediante llamadas de funciones NCP.

- **Protocolo de notificación de servicios (SAP).** Los servidores Netware utilizan **SAP** para notificar sus servicios. Cada servidor en el que se ejecuta SAP puede identificarse asimismo y a sus servicios cada minuto mediante la emisión de un Paquete de identificación de servicio.

Los usuarios de SAP también pueden determinar la identidad y los servicios de los servidores de la interred utilizando un Paquete de consulta de servicio. Consultando la información de cabecera de IPX, las implementaciones de SAP identifican las direcciones de servicio que se necesitan para iniciar una sesión.

Como se observa en la figura 3.0 y por las descripciones básicas realizadas, la frontera entre la capa de red y la capa de transporte no esta clara en el modelo netware, es por lo que se pueden analizar en el ámbito de esta publicación los protocolos IPX y SPX como formando parte de la capa de red y de la capa de transporte.

IPX
(Internetwork Packet eXchange)

4. Conceptos Generales IPX (Novell Netware).

El primer protocolo de comunicaciones que se implementó dentro del entorno Netware fue el **IPX** (Internetwork Packet Exchange, intercambio de paquetes en redes múltiples).

Se empleaba exclusivamente para comunicar estaciones de trabajo y servidores entre sí. IPX es un protocolo orientado a no conexión derivado del protocolo **IDP** (Internetwork Datagram Protocol, protocolo de datagramas en redes múltiples) del sistema de red de Xerox (XNS).

IPX se utiliza para enviar y recibir paquetes de información entre las estaciones de trabajo y los servidores participantes en la comunicación.

Esta comunicación es **no garantizada**, ya que no se prevé el envío de un acuse de recibo desde el destino del paquete. Sin embargo, IPX indica si el paquete fue enviado realmente o no.

IPX trabaja con **datagramas**, es decir, paquetes independientes de datos que no requieren confirmación. Los datagramas reducen el tráfico y dinamizan el funcionamiento de la red.

En el caso de IPX, el tanto por ciento de los datagramas distribuidos con éxito ronda sobre el 95%. Consecuentemente, si se utiliza IPX, se necesitará desarrollar una estrategia que confirme la distribución de los datos y posiblemente, se tendrá que guardar los paquetes en orden.

El interfaz de IPX con la red es un controlador de LAN (red de área local). Para que IPX funcione es necesario que se cargue en memoria con un controlador LAN específicamente programado para funcionar con IPX. Esto se realiza de manera diferente según se trate de una estación de trabajo o de un servidor Netware.

Para usar estos protocolos se debe pensar en términos de **paquetes**. Las aplicaciones que utilicen IPX deben preparar los paquetes de una manera específica y deben seguir un procedimiento para iniciar la comunicación con IPX.

4.1. Paquete IPX.

Un **paquete** es un conjunto de datos formateado de una manera especial para ser transmitido de una estación de trabajo a otra. El paquete propiamente dicho consta de una **cabecera** de 30 bytes y un máximo recomendado de 546 bytes de **datos**.

La aplicación (cliente) determina el contenido y la estructura de la 'porción' de datos del paquete:

- La **cabecera** contiene los datos necesarios para transmitir el paquete desde su origen hasta su destino.
- El mensaje son los **datos** de la aplicación.

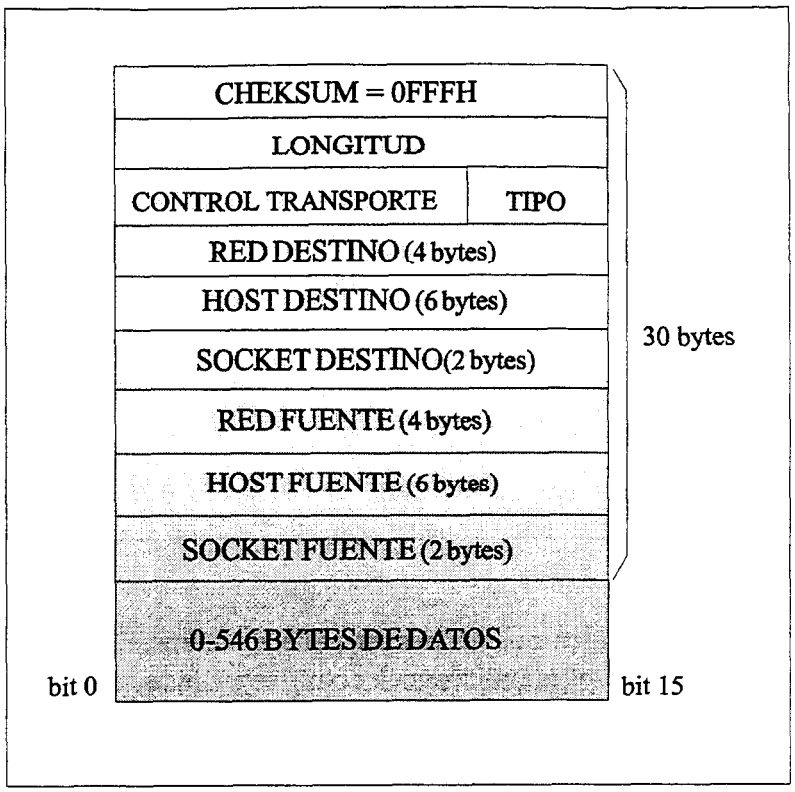


Fig. 4-1. Paquete IPX.

4.1.1. Cabecera IPX.

El **tamaño máximo** de un paquete IPX es 576 bytes. Los primeros 30 bytes de cada paquete forman la cabecera de un paquete IPX. Los bytes restantes pueden ser usados por la aplicación para transmitir datos. La cabecera incluye los siguientes campos:

- **Cheksum** (2 bytes): Este campo contiene un cheksum de los 30 primeros bytes de la cabecera de un paquete IPX.

Desde que los 'drivers' de Netware añaden protección eficaz contra los errores en los paquetes, la mayoría de los programas no necesitan este campo.

- **Longitud de paquete** (2 bytes): Contiene la longitud total en bytes del paquete, incluyendo la cabecera y los datos. Una aplicación NLM$_{(tm)}$ puede examinar el ECB de recepción (ver apartado 4.2) para determinar el valor de este campo.

- **Control de transporte** (1 byte): Este campo es utilizado por los 'routers' de red. IPX siempre coloca este campo a 0 antes de enviar un paquete.

- **Tipo de paquete** (1 byte): Indica el tipo de servicio que el paquete ofrece o requiere. Xerox ha definido varios valores, pero para las aplicaciones IPX, este campo debe valer 0 ó 4.

La tabla siguiente (tabla 4-1) muestra los distintos valores definidos por Xerox:

Valor	Tipo de paquete
0	Tipo de paquete desconocido
1	Routing Information Packet (RIP)
2	Echo packet
3	Error packet
4	Pacte Exchange packet (PEP)
5	Protocolo de Secuencia de Paquete (SPP)
16-31	Protocolos experimentales
17	Netware Core Protocol (NCP)

Tabla 4-1: Tipos de paquetes.

Netware utiliza para los siguientes protocolos:

- **IPX:** el tipo de paquetes del 0 al 4.
- **SPX:** el tipo de paquetes 5.
- **NCP:** el tipo 17.

➢**Direcciones Network** (Direcciones de red):

Las direcciones de red de origen y destino ocupan la mayor parte del espacio en las cabeceras IPX. Una **dirección de red** incluye lo siguiente:

- 4 bytes de dirección de **red**.
- 6 bytes de dirección de **nodo**.
- 2 bytes de número de **socket**.

La figura 4-2 muestra un ejemplo de dirección de red:

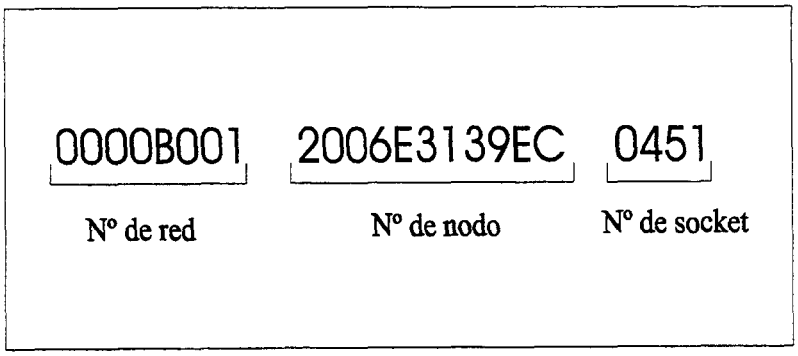


Fig. 4-2: Ejemplo de dirección de red.

Los routers de la red cuentan con las direcciones de red para distribuir un paquete cuando el destino es una red distinta de la red origen.

Cuando el paquete llega a la red de destino, el driver de LAN de la estación de trabajo reconoce el paquete por su dirección de nodo.

El driver de LAN maneja el paquete sobre IPX, el cual usa el número de 'socket' para distribuir los datos a un proceso en la estación de trabajo.

Mediante la asignación de un número de socket a la transmisión, IPX puede soportar múltiples procesos en la estación de trabajo.

- **Dirección destino:** Esta dirección está compuesta por los siguientes tres campos:

  - **Red destino** (4 bytes): Contiene el número de red que el administrador de red asigna, es decir, indica la dirección de la red destino en el caso de hacer routing entre redes.

Para un nodo de destino en la misma red que el nodo de origen, este campo puede contener la dirección de red o el valor '0'.

  - **Nodo destino** (6 bytes): Este campo contiene la dirección física del nodo destino, es decir, especifica la dirección hardware física de la tarjeta, o sea, el **NIC** (Network Interface Card).

No todas las topologías LAN usan el mismo tamaño para el campo de dirección. Por ejemplo, un nodo en una red Ethernet requiere los seis bytes completos para especificar su dirección, mientras que un nodo en una red Omninet requiere sólo un byte.

Si se necesitan menos de 6 bytes para especificar la dirección de un nodo, esta ocupará los bytes menos significantes, quedando los más significantes a '0'.

Si la dirección de nodo toma el valor '0x FF FF FF FF FF FF' (dirección Broadcast), se difunde el paquete a todos los nodos de la red de destino.

  - **Socket destino** (2 bytes)**:** Es un valor que asocia una aplicación o un proceso con la transmisión de datos. En este caso, especifica el proceso a ejecutar dentro del nodo destino.

Algunos valores definidos por Xerox para este campo son:

Nº de socket	
0x01	Routing information packet (RIP).
0x02	Echo protocol packet.
0x03	Paquete controlador de errores.
0x20-0x3F	Experimental.

Tabla 4-2: Números de sockets.

Los valores de socket comprendidos en el rango '0x01' y '0xBB8' se consideran 'bien conocidos', esto quiere decir que están registrados por Xerox. Los números de socket mayores que '0xBB8' son asignables dinámicamente.

El servidor Netware utiliza el número '0x0451' (valor hexadecimal). Novell asigna dinámicamente números de socket comenzando en '4000H' para paquetes Netware de valor añadido, y otros muy utilizados comienzan en '8000H'.

Los valores de socket asignados a Netware son:

Nº de socket	
0x451	Paquete de servicio de ficheros
0x452	Paquete de anuncio de servicio
0x453	RIP
0x455	Paquete NetBIOS
0x456	Paquete de diagnóstico.

Tabla 4-3: Sockets asignados a Netware.

- **Dirección fuente:** IPX rellena automáticamente la dirección de origen de los paquetes que se envían.
    - **Red fuente:** Contiene los 4 bytes que indican el número de red que IPX asigna.
    - **Nodo fuente:** Equivalente al caso del nodo destino.
    - **Socket fuente:** Contiene la dirección de socket del proceso que transmite el paquete.

Hay que tener en cuenta que las **estructuras** (cabeceras IPX, ECB's, etc...) que se van a utilizar en una comunicación cliente/servidor varían dependiendo si el ejecutable va a estar en la **estación cliente** (.EXE) o en el **servidor** (.NLM).

Además, dichas estructuras se encuentran definidas en ficheros distintos.

4.1.2. Datos.

La **zona de datos** (546 bytes) corresponde a la información procedente de los niveles superiores. El buffer de datos es simplemente una secuencia de bytes.

La memoria asignada a la cabecera debe ser contigua, pero la parte de datos puede estar formada por fragmentos de memoria no contiguos.

El **tamaño máximo** de un **paquete IPX** es de 576 bytes.

4.2. ECBs (Bloque de control de eventos).

Otra **estructura** muy importante que interviene en la comunicación a nivel IPX es el **ECB** (Event Control Block, bloque de control de eventos).

Los ECBs son el enlace entre la aplicación e IPX. Cada ECB contiene la información que IPX necesita para enviar o recibir paquetes a la medida de las necesidades de la aplicación.

Controlan los eventos relacionados con la transmisión y recepción de paquetes IPX y SPX; también controlan el establecimiento y finalización de las sesiones SPX.

El ECB se utiliza, entre otras cosas, para indicar la localización de los fragmentos de memoria que forman el paquete que se va a enviar.

En el caso de recepción de paquetes, el ECB se emplea para indicar dónde colocar la información del paquete cuando éste se reciba.

4.2.1. Estructura del ECB

A continuación se muestra la estructura del ECB definida en nwipxspx.h":

```
typedef struct IPX_ECBStruct
  {
  unsigned long          semHandleSave;      /* R */
  struct IPX_ECBStruct   **queueHead;        /* sr */
  struct IPX_ECBStruct   *next;              /* A */
  struct IPX_ECBStruct   *prev;              /* A */
  unsigned short         status;             /* q */
  unsigned long          semHandle;          /* sr */
  unsigned short         IProtID;            /* R */
  unsigned char          protID [6];         /* R */
  unsigned long          boardNumber;/* R */
  unsigned char          immediateAddress [6]; /* s*/
  unsigned char          driverWS [4];       /* R */
  unsigned long          ESREBXValue;        /* R */
  unsigned short         socket;        /* sr */
  unsigned short         protocolWorkspace;  /* R */
  unsigned long          dataLen;            /* q */
  unsigned long          fragCount;          /* sr */
  ECBFrag                fragList [2];       /* sr */
  } IPX_ECB;
```

donde:

- **R**: Indica que el campo es reservado.

- **s**: Indica que el campo debe ser usado por la aplicación cuando utilice el ECB para enviar un paquete.

- **r**: Equivalente al anterior, pero cuando se use el ECB para la recepción de un paquete.

- **A**: Indica que el campo puede ser utilizado cuando el ECB no esté siendo usado por IPX/SPX.

- **q**: Son campos que la aplicación puede leer.

El ECB incluye los siguientes **campos**:

- **semHandleSave:** IPX y SPX utilizan este campo para propósitos internos y no debe modificarse.

- **queueHead:** La aplicación debe colocar en este campo un puntero a otro ECB si se quiere mantener una lista encadenada de ECBs, siendo éste la cabeza de la lista; si no, se debe rellenar este campo con el valor NULL.

- **next, prev:** IPX/SPX usan estos campos para propósitos internos, mientras el ECB esté en uso.

Cuando no se esté utilizando, la aplicación lo podrá usar para mantener el ECB en una lista enlazada.

- **status:** Este campo indica el estado del evento (por ejemplo, SPX está 'escuchando' en un socket).

Si el valor de este campo es positivo, el evento todavía no ha ocurrido; si es cero, el evento se completó de forma satisfactoria; si es negativo, el evento se completó con un error.

Los valores que suele tomar son:

0x11 - AES (asynchronous event service) waiting
0x12 - Holding
0x13 - Session listen
0x14 - Processing
0x15 - Receiving
0x16 - Sending
0x17 - Waiting

- **semHandle:** Debe rellenarse por la aplicación con un manejador de semáforos (valor que nos devuelve la función **OpenLocalSemaphore**), o bien con el valor NULL.

El propósito del manejador de semáforos es permitir que la aplicación se bloquee dependiendo de que uno o más eventos IPX/SPX ocurran o terminen.

Si el evento (asociado con alguna de las funciones IPX/SPX que toma un ECB como parámetro) ocurre, y este campo tiene un valor no nulo, IPX/SPX llama a la función **SignalLocalSemaphore** con el manejador de semáforos especificado.

Con esto, se desbloquea la aplicación si, anteriormente, estaba bloqueada en el semáforo (es decir esperando a que el evento ocurriera).

- Los campos **IProtID, boardNumber, driveWS, ESREBXValue, protocolWorkSpace** y **dataLen** son usados internamente por IPX/SPX y su valor no debe ser modificado.

- **immediateAddress:** Este campo debe contener la dirección del último nodo que transmitió el paquete (bien, un nodo de la red, o bien, un router).
- Sólo debe inicializarse para IPX (no es necesario para una comunicación SPX).

- **socket:** Identifica el socket de envío o recepción que está asociado al ECB. Este campo puede usarse de dos formas diferentes:

  - Se coloca en este campo el número de socket deseado y se iguala a cero en aquellas funciones que tengan al ECB y al socket como parámetros (**openSocket**, por ejemplo).
  - La aplicación ignora este campo y sí lo tiene en cuenta en el tipo de funciones antes mencionadas.

- **fragCount:** Indica el número de fragmentos de buffers (uno o más) que forman el paquete.

- **fragList:** La aplicación provee una **lista** (el ECB siempre define dos fragmentos) de identificadores de los distintos fragmentos. Esta tiene la siguiente estructura:

```
typedef struct tagECBFrag
  {
  void              *fragAddress;
  unsigned long     fragSize;
  } ECBFrag;
```

El campo **fragAddress** indica la dirección del fragmento y **fragSize** el tamaño. El primer **fragList** debe identificar un buffer lo suficientemente largo para contener la cabecera del paquete. Por eso, para IPX el buffer debe ser por lo menos de 30 bytes y para SPX, al menos de 42 bytes.

4.2.2. Estructura del ECB (para estaciones cliente).

La **estructura del ECB para el cliente** viene definida en el fichero de cabecera "**nxtw.h**":

```
typedef struct ECB
        {
        void far        *linkAddress;
        void (far       *ESRAddress)();
        BYTE            inUseFlag;
        BYTE            completionCode;
        WORDsocketNumber;
        BYTE            IPXWorkspace[4];
        BYTE            driverWorkspace[12];
        BYTE            immediateAddress[6];
        WORDfragmentCount;
        ECBFragment fragmentDescriptor[2];
        }ECB;
```

Esta estructura se divide en los siguientes **campos**:

- **linkAddress:** IPX utiliza este campo cuando el ECB está en uso. La aplicación lo puede utilizar mientras no se haya pasado el ECB a IPX (por ejemplo, para mantener una lista enlazada de ECBs).

- **ESRAddress:** Contiene la dirección de una **ESR** (Event Service Routine, rutina de servicio de evento) definida por la aplicación.

IPX ejecutará la ESR cuando IPX use el ECB para enviar o recibir un paquete. Sólo en el caso de utilizar un rutina de servicio de eventos hay que proporcionar una **ESRAddress** (en caso contrario, este campo debe contener el valor NULL).

- **inUseFlag:** Indica el estado actual del ECB. Si tiene un valor distinto de cero indica que el ECB está siendo usado. Después del evento, IPX resetea este campo (lo pone a cero).

La tabla siguiente (tabla 4-4) muestra los distintos valores que puede tomar el campo **inUseFlag**:

Valor	Comentario
F8h	La aplicación envió un paquete mientras IPX estaba ocupada. IPX ha puesto el paquete y el ECB en una cola para procesarlos.
FAh	IPX está procesando el ECB.
FBh	IPX ha dedicado el ECB a un evento que ya ocurrió y que está en una cola procesándose.
FCh	IPX ha dedicado el ECB a un evento controlado por el AES y que todavía no ha ocurrido.
FDh	IPX ha dedicado el ECB a controlar un evento que todavía no ha ocurrido.

FEh	IPX está esperando en un socket por la llegada de paquetes.
FFh	IPX está usando el ECB para enviar paquetes.

Tabla 4-4: Valores del campo 'InUseFlag'.

- **completionCode:** IPX le asigna un valor a este campo cuando ha liberado el ECB después de un evento. Este valor no es válido hasta que IPX resetee el campo **inUseFlag** a cero. El valor de **completionCode** depende del tipo de evento que IPX haya procesado.

Hay cuatro clases de **eventos**: enviar un paquete, esperar por él, controlar un evento asíncrono y cancelar un evento.

La siguiente tabla muestra los distintos valores de este campo para cada clase de evento:

	Send	Listen Event	Schedule	Cancel
00h	éxito	éxito	éxito	éxito
FCh	cancelado	cancelado	cancelado	indefinido
FDh	paquete erróneo	overflow	indefinido	indefinido
FEh	paquete no entregado	indefinido	indefinido	indefinido
FFh	error	socket cerrado	indefinido	indefinido
F9h	indefinido	error	indefinido	indefinido

Tabla 4-5: Valores del campo 'CompletionCode'.

- **socketNumber:** Identifica el **socket** asociado con el ECB.
- **IPXWorkSpace, driverWorkSpace:** Estos dos campos son reservados. No se deben modificar mientras IPX esté usando el ECB.
- **inmediateAddress:** Identifica la dirección del último nodo que transmitió el paquete.

Este campo debe inicializarse para un paquete IPX, no es necesario para SPX.

- **fragmentCount:** Indica el número de fragmentos de buffers que van a formar el paquete (al menos, debe ser 1).

Los fragmentos están definidos en un array de la siguiente estructura:

```
typedef struct ECBFragment
        {
        void far        *address;
        WORDsize;
        }ECBFragment;
```

El campo **address** indica la dirección del fragmento y el campo **size**, el tamaño del fragmento referenciado por el campo anterior.

El número de **fragmentDescriptors** depende de **fragmentCount.** Debe asignarse memoria para un descriptor de fragmento por cada fragmento de memoria de los que forman el paquete de información. El tamaño de un ECB varía según el número de fragmentos especificados para construir el paquete.

4.3. Utilización de rutinas de servicios de eventos.

El campo **ESRAddress** en el ECB puede contener la dirección de una **rutina de servicio de eventos** que una determinada aplicación desea ejecutar en respuesta a un evento.

El evento 'detonante' de la rutina de servicio podría ser el envío o la recepción de un paquete, o un evento planificado por la aplicación. Un uso frecuente de una rutina de servicio de eventos es la realización de una acción cuando se recibe un paquete de información.

Cuando se emplea una ESR, ésta se ejecuta automáticamente cuando la función que está utilizando el ECB termina. Una rutina de servicio de evento se comporta como una rutina de servicio de interrupción. IPX llama a esta rutina con las interrupciones desactivadas; por lo tanto, las rutinas ESR deben ser cortas y rápidas. Cuando la rutina acaba, IPX devuelve el control a la aplicación.

Una rutina de servicio no debería hacer peticiones que el programa principal de cualquier aplicación pueda efectuar adecuadamente.

Normalmente, se suelen usar para colocar los ECBs asociados en una cola; el cuerpo principal de la aplicación controlaría la cola y procesaría los datos.

SPX
(Sequenced Packet eXchange)

5. Conceptos Generales SPX.

SPX (Sequenced Packet Exchange Protocol) proporciona un **servicio de transporte orientado a conexión**, permitiendo que los datos 'viajen' sobre una conexión establecida y de forma secuencial.

SPX presenta las siguientes características:

- Proporciona una entrega garantizada usando primitivas IPX para enviar paquetes y recibir las confirmaciones de los paquetes entregados.
- Después de un número determinado de intentos fallidos en la confirmación de los paquetes, se asume que la conexión ha fallado.

Si se necesita saber con certeza que los datos se están recibiendo en el orden en que se enviaron, la aplicación debe usar SPX. SPX proporciona **secuenciamiento de paquetes** y **confirmación de la entrega**, todo esto asociado con un **servicio en modo conexión**.

Estas características añaden algo de sobrecarga a las comunicaciones de red, pero no son muy complicadas de usar.

Las **diferencias más notables entre IPX y SPX** son sus cabeceras y la sobrecarga en tiempo de ejecución de SPX, debida al mecanismo que garantiza la entrega de paquetes. Para garantizar la entrega se realizan varios intentos de envío, y se informa al NLM que hace la llamada si tras un determinado número de intentos el envío ha fallado. De este modo, el NLM que envía el paquete no tiene que comprobar si el paquete ha llegado a su destino. SPX notifica al módulo NLM el estado del envío.

5.1. Información sobre paquetes.

Tanto IPX como SPX son adaptaciones de la arquitectura Xerox* Network Systems (XNS*). SPX conforma el protocolo **SPP** (Sequenced Packet Protocol). Para usar estos protocolos se debe pensar en términos de paquetes.

Tal como se vio en el capítulo 4 un paquete contiene una cabecera y un mensaje. La cabecera contiene todos los datos necesarios para transportar el paquete de una estación a otra. El mensaje está formado por los datos de la aplicación.

IPX define el formato básico de la cabecera del paquete. SPX aumenta este formato con los datos necesarios para mantener las conexiones.

5.2. Estructura del paquete SPX.

El formato del paquete SPX es idéntico al de IPX exceptuando que la cabecera SPX tiene 12 bytes adicionales. Un paquete SPX consiste en una **cabecera** de 42 bytes además de

534 bytes de **datos.** El tamaño mínimo del paquete es 42 bytes (solamente la cabecera) y el máximo es 576 bytes. El contenido y la estructura de la parte de datos depende de la aplicación que esté usando SPX y puede tomar cualquier formato.

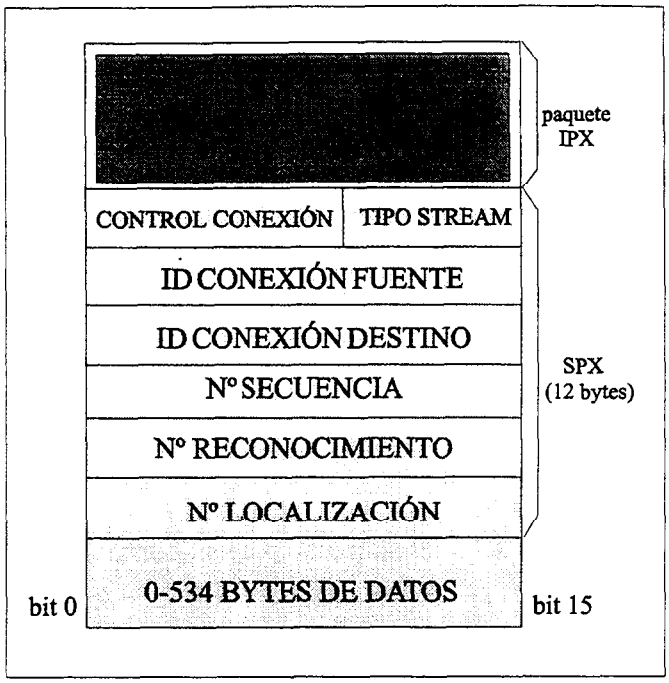


Fig. 5-1: Paquete SPX.

La **cabecera del paquete SPX** consiste en una cabecera IPX (30 bytes) más siete campos adicionales que se muestran a continuación:

```
unsigned char     connectionCtl;
unsigned char     dataStreamType;
unsigned short    sourceConnectID;
unsigned short    destConnectID;
unsigned short    sequenceNumber;
unsigned short    ackNumber;
unsigned short    allocNumber;
```

Estos campos adicionales permiten a SPX proporcionar el secuenciamiento y la distribución garantizada.

SPX rellena todos estos campos exceptuando: **destination fields** (en algunas funciones), **connectionCtl**, y **dataStreamType**.

A continuación se explica el significado de los distintos campos que forman una cabecera SPX:

- **connectionCtl** (control de conexión): Este campo controla el flujo bidireccional de los datos a través de una conexión SPX.

- **dataStreamType** (tipo de corriente de datos): Este campo indica el **tipo de datos** que va incluido en el paquete. El rango de valores comprende desde el valor hexadecimal '0x00' hasta '0xFD' y son definidos por el cliente.

Los siguientes valores los utiliza SPX para gestionar las conexiones y no deben ser usados por la aplicación:

  - **'0xFE':** Se inicializará este campo con este valor cuando sea un paquete indicador del **final de conexión**. Cuando un cliente quiere terminar una conexión activa, SPX genera un paquete de este tipo.

    Este paquete se envía al otro extremo de la conexión siendo el último mensaje que se envía sobre la conexión.

  - **'0xFF'**: Indica un paquete de 'reconocimiento' del final de la conexión. SPX genera este tipo de paquete de forma automática. Está considerado como un paquete del sistema.

- **sourceConnectID**, **destConnectID** (identificación de origen y de destino): Estos campos especifican el **número de identificación de la conexión** asignado a la conexión SPX por el nodo fuente y el nodo destino, respectivamente.

- • **sequenceNumber** (número de secuencia): Este campo mantiene la cuenta de los paquetes intercambiados en una dirección sobre la conexión. Cada extremo de la conexión mantiene su propia cuenta. Debido a que SPX controla este campo, los procesos del cliente no necesitan tenerlo en cuenta.

- • **ackNumber** (número de confirmación): Indica el siguiente paquete que una conexión SPX espera recibir. El rango de valores de este campo comprende desde el valor '0x0000' hasta el '0xFFFF'; cuando se alcanza este último se vuelve a empezar la cuenta desde el '0x0000'.

- • **allocNumber** (número de localización): Este campo, en combinación con el anterior, indica el número de buffers de recepción de paquetes disponibles para una conexión SPX determinada. SPX lo utiliza para implementar un control de flujo entre las aplicaciones que se comunican.

SPX envía paquetes solamente hasta que el número de la secuencia local sea igual al campo **allocNumber** del otro extremo de la conexión. Este campo se incrementa hasta el valor '0xFFFF' y, una vez alcanzado este valor, se reinicia la cuenta desde '0x0000'.

5.3 SPX: Servicio en modo conexión.

La comunicación bajo SPX requiere más o menos la misma preparación que se necesitaba con IPX. Se debe inicializar SPX, abrir un canal de comunicación (socket), y prepara los ECBs tanto para la transmisión como para la recepción de datos, tal y como se hacía para IPX.

Sin embargo, SPX requiere unos pasos adicionales ya que proporciona un servicio orientado a conexión.

5.3.1. Establecimiento de la conexión.

Cada evento, tales como la transmisión o recepción de un paquete, es controlado por una estructura, ya conocida, llamada **ECB** (Bloque de Control de Eventos).

Por ejemplo, antes de que una aplicación pueda mandar un paquete, debe preparar un ECB que, además de otras cosas, contiene la dirección en memoria del paquete.

Generalmente, la aplicación llama a la función de envío de datos, bien bajo IPX ó SPX, que tiene como parámetro un puntero al ECB.

De igual forma, antes de que una aplicación pueda recibir un paquete, debe preparar un ECB que contenga la dirección de un buffer en el cual se van a almacenar los datos que lleguen.

Una vez hecho esto, la aplicación llamará a la función de recepción (IPX ó SPX) con un puntero al ECB.

La explicación sobre cómo preparar los ECBs para SPX es igual que para IPX. Hay que tener en cuenta que **SPX necesita un ECB adicional** para establecer la conexión.

Generalmente, para este propósito, se tendrá que preparar otro ECB distinto al inicial, debido a que éste último no requiere rutina de servicio de eventos y solamente va a hacer referencia a la cabecera SPX.

En el **establecimiento de la conexión**, un nodo actúa como **parte activa** (el que realiza la petición de conexión) siendo el otro la **parte pasiva** (espera hasta que le llegue una petición de conexión).

Una vez se haya establecido la conexión, SPX no hace ninguna distinción entre el que 'llama' y el que 'escucha'.

5.3.2 Transferencia de Datos

Una vez la conexión ha sido establecida y se dispone del canal de comunicación, mediante las funciones SPXListenForSequencedPacket y SPXSendSequencedPacket se pueden transmitir bloques de datos.

SPX permite que las aplicaciones puedan recibir paquetes secuenciados de manera asíncrona. Por lo tanto, la primera de las funciones dedica un determinado ECB para recibir los paquetes.

Los **sockets** usados en la conexiones SPX no se pueden usar para transmitir o recibir paquetes directamente llamando a las siguientes funciones IPX: **IPXSendPacket** y **IPXListenForPacket**.

Cuando una aplicación pasa varias combinaciones ECB/paquete a SPX, este protocolo los coloca en una cola y los envía en el orden en el que se recibieron.

Cuando una aplicación cierra un socket SPX, SPX finaliza todas las conexiones asociadas con ese socket.

Una aplicación puede establecer una conexión entre dos sockets que residan en el mismo nodo.

Para recibir información sobre el estado de una conexión SPX se debe utilizar la función **SPXGetConnectionStatus**. Su sintaxis es la siguiente:

5.3.3 Liberación de Conexión

Una vez la comunicación se quiere dar por terminada, se dispone de un mecanismo de liberación ordenada y otro de liberación abrupta, según se desee. Existen dos funciones que concluyen una conexión SPX:

- **SPXTerminateConnection:** Informa al otro extremo del cierre de la conexión.
- **SPXAbortConnection:** Esta función no alerta al otro extremo de la conexión del cierre de ésta.

TLI
(Transport Layer Interface)

6. Conceptos Generales TLI (Transport Layer Interface)

TLI es un **interfaz de servicio de transporte** que permite implementar aplicaciones y protocolos de nivel superior sin conocimiento de la topología y características de la red.

TLI define un **conjunto de servicios**, incluyendo los siguientes:

- Protocolos ISO.
- Transmission Control Protocol / Internetwork Protocol (TCP/IP).
- Netware's Internetwork Packet Exchange$_{(tm)}$/ Sequenced Packet Exchange (IPX$_{(tm)}$/SPX$_{(tm)}$).
- Xerox* Network Systems* (XNS*).
- Systems Network Architecture (SNA*).

Debido a que el nivel de transporte 'oculta' los detalles del medio físico que se está utilizando, TLI ofrece una independencia del medio para las aplicaciones de red y los protocolos de niveles superiores.

Netware TLI se rige por la especificación 1988 XTI con dos pequeñas excepciones: **t_alloc** y la estructura de nombres TLI de UNIX(R).

TLI se implementa como una librería de usuario usando los mecanismos de entrada/salida del STREAM$_{(tm)}$. Por tanto, muchos servicios disponibles en las aplicaciones de STREAM$_{(tm)}$ están disponibles para usuarios de TLI.

La figura 6-1 ilustra el protocolo TLI:

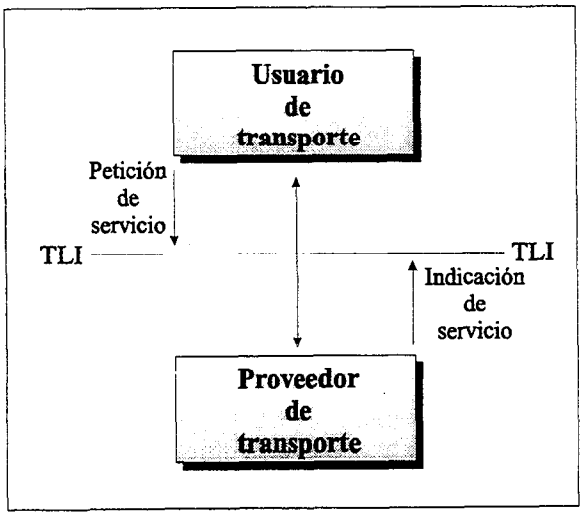


Fig. 6-1: Bases del protocolo TLI.

El **proveedor** de transporte es la entidad que provee los servicios de TLI, mientras que el **usuario** de transporte es quien requiere estos servicios. Un ejemplo de un proveedor de transporte es el protocolo de transporte ISO, mientras que un usuario de transporte puede ser una aplicación NLM, o una aplicación de red.

El usuario de transporte accede al servicio del proveedor de transporte por medio de la apropiada **petición de servicios**. Un ejemplo podría ser una petición para transferir datos en una conexión. De igual forma, el proveedor de transporte notifica al usuario de los distintos eventos, por ejemplo, la recepción de datos en una conexión.

Las funciones de TLI soportan los servicios de TLI para procesos de usuario. Estas funciones permiten al usuario hacer peticiones al proveedor y procesar los eventos que ocurran.

6.1.Tipos de servicios.

El interfaz del protocolo de transporte (TLI) proporciona dos modos o tipos de servicio:

- Modo de conexión.
- Modo de no-conexión.

El servicio en **modo conexión** está orientado a circuito y permite la transmisión de datos sobre una conexión establecida. Este servicio está pensado para aplicaciones que requieren , relativamente, larga duración.

El servicio en **modo no conexión,** por el contrario, está orientado a paquete y soporta transferencia de datos en unidades. Este servicio requiere sólo de la existencia anterior de una asociación entre los usuarios involucrados, lo cual determina las características de los datos que se van a transmitir.Toda la información que hace falta para distribuir una unidad de datos (por ejemplo, la dirección de destino) se entrega al proveedor de transporte junto con los datos a transmitir, en un acceso al servicio (no se necesita relacionar éste último con algún otro acceso al servicio).

6.1.1. Servicio en modo conexión.

El servicio de transporte en **modo conexión** está caracterizado por 4 fases:

- Gestión local.
- Establecimiento de la conexión.
- Transferencia de datos.
- Liberar la conexión (desconexión).

A continuación se explica cada una de estas fases de forma más detallada.

6.1.1.1. Gestión local.

Esta fase define **operaciones locales** entre un usuario de transporte y un proveedor de transporte.

Por ejemplo, un usuario debe establecer un canal de comunicación con el proveedor de transporte. Cada canal entre un usuario y un proveedor de transporte es un único 'punto final' (endpoint) de comunicación conocido como **extremo de conexión de transporte**.

La función **t_open** permite a un usuario elegir un determinado proveedor de transporte para proporcionar el servicio en modo conexión y establecer el extremo de conexión de transporte.

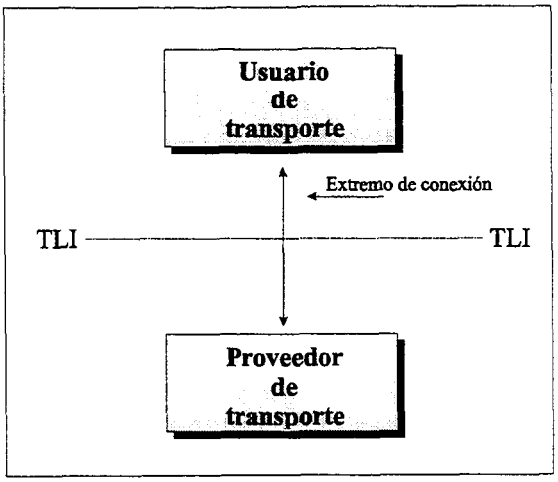


Fig. 6-2: Extremo de conexión.

Otra labor local necesaria para cada usuario es establecer una identidad con el proveedor de transporte. Cada usuario se identifica por una **dirección de transporte**; exactamente, se asocia una dirección de transporte con cada extremo de conexión. En el servicio en modo conexión, un usuario pide una conexión a otro mediante una dirección de usuario especificada.

La **estructura de una dirección de transporte** está definida por el espacio de dirección del proveedor de transporte. Una dirección puede ser tan simple como una cadena aleatoria de caracteres (por ejemplo, 'file_server'), o tan compleja como un patrón de bits codificados que especifica toda la información necesaria para enrutar los datos a través de la red.

Cada proveedor de transporte define su propio mecanismo para la identificación de los usuarios. Las direcciones se pueden asignar a cada extremo de conexión de transporte mediante la función **t_bind.**

Además de las funciones mencionadas (**t_open** y **t_bind**), existen varias funciones disponibles para apoyar las operaciones locales. Se muestran, a continuación, en la siguiente tabla (Tabla 6-1):

Función	Descripción
t_alloc	Reserva memoria para las estructuras de datos TLI.
t_bind	Asocia un extremo de conexión a una dirección.
t_close	Cierra un extremo de conexión abierto con **t_open**.
t_error	Imprime un mensaje TLI de error.
t_free	Libera el espacio reservado por la función **t_alloc**.
t_getinfo	Obtiene información sobre el proveedor de transporte.
t_getstate	Devuelve el estado de un extremo de conexión.

t_look	Devuelve el evento actual en el extremo de conexión especificado.
t_open	Abre un extremo de conexión TLI.
t_optmgmt	Permite a un usuario de transporte verificar o negociar las opciones de protocolo con el proveedor de transporte.
t_unbind	Desenlaza la dirección previamente enlazada con **t_bind**.

Tabla 6-1: Funciones TLI.

6.1.1.2.Establecimiento de la conexión.

Como se muestra en la figura 6-3, la fase de **establecimiento de una conexión** permite a dos usuarios crear una conexión o circuito virtual entre ellos.

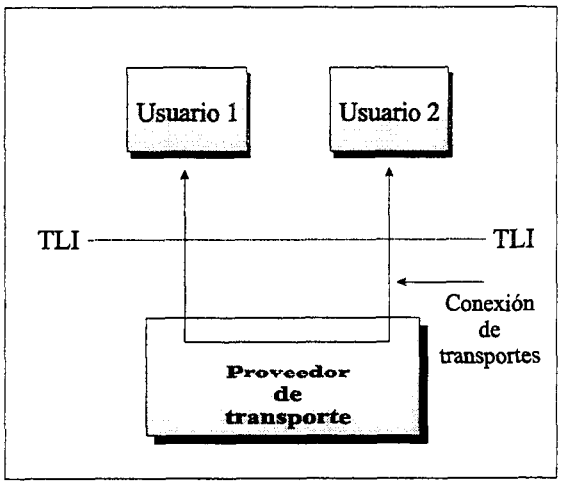


Fig. 6-3: Establecimiento de la conexión.

Esta fase se ilustra mejor mediante la relación cliente-servidor entre dos usuarios. Uno de los usuarios, el servidor, anuncia alguno de sus servicios a un grupo de usuarios; entonces, el servidor, queda a la espera de una petición por parte de ellos.

Si un cliente requiere el servicio, intentará **conectarse** al servidor utilizando la dirección de transporte antes anunciada.

La función **t_connect** inicia la petición de conexión Un argumento para esta función, la dirección de transporte, identifica a qué servidor quiere acceder el cliente.

Usando la función **t_listen**, se notifica al servidor cada petición de conexión que llega.

Si se acepta la petición del cliente de acceder al servicio, el servidor usará la función **t_accept**. Una vez aceptada la petición, se establece la conexión de transporte.

La tabla 6-2 muestra las funciones que se han de utilizar para establecer una conexión:

Función	Descripción

t_accept	Envía notificación de aceptación al extremo que solicitó la conexión.
t_connect	Establece una conexión con el usuario de transporte en un destino especificado.
t_listen	Espera una solicitud de conexión de un usuario de transporte.
t_rcvconnect	Completa el establecimiento de la conexión si se utilizó **t_connect** en modo asíncrono.

Tabla 6-2: Funciones TLI utilizadas en el establecimiento de una conexión (modo conexión).

6.1.1.3. Transferencia de datos.

La fase de transferencia de datos permite a los usuarios **transferir datos** en ambos sentidos sobre una conexión establecida. Dos funciones, **t_snd** y **t_rcv**, envían y reciben datos sobre esta conexión.

Se garantiza que todos los datos enviados por un usuario se entregarán al otro, en el otro extremo de la conexión, en el orden en que fueron enviados.

Para la transferencia de datos se usarán las siguientes funciones mostradas, a continuación, en la tabla 6-3:

Función	Descripción
t_rcv	Recibe datos sobre una conexión establecida
t_snd	Envía datos sobre una conexión establecida.

Tabla 6-3: Funciones TLI utilizadas en una transferencia de datos (modo conexión).

6.1.1.4. Desconexión.

La fase de desconexión permite **interrumpir una conexión establecida**. Cuando se decide que una conversación debe terminar, se puede pedir al proveedor que libere la conexión de transporte.

TLI soporta dos tipos de desconexión:

➢ **Abrupta** (o abortiva): El proveedor de transporte libera la conexión inmediatamente.

Cualquier dato que no haya llegado al otro usuario de transporte puede ser descartado por el proveedor de transporte.

Función	Descripción
t_snddis	Desconecta el extremo emisor de una conexión.
t_rcvdis	Indica que la conexión fue abortada, incluye la razón de la desconexión.

Tabla 6-4: Funciones TLI utilizadas en una desconexión abrupta (modo conexión).

Todos los proveedores de transporte deben soportar el procedimiento de liberación abrupta.

➢ **Ordenada:** Se facilita a los usuarios conectados que terminen la comunicación elegantemente, es decir, sin pérdida de datos.

Las funciones utilizadas, en este caso, son:

Función	Descripción
t_rcvrel	Indica que el usuario remoto solicita una desconexión ordenada.
t_sndrel	Petición de liberar de forma ordenada, una conexión.

Tabla 6-5: Funciones TLI utilizadas en una desconexión ordenada (modo conexión).

6.1.2. Servicio en modo no-conexión.

El servicio de transporte en modo **no-conexión** está caracterizado por 2 fases solamente:

- Mantenimiento o gestión local.
- Transferencia de datos.

La fase de **gestión local** define las mismas operaciones descritas anteriormente para el servicio en modo conexión.

La fase de **transferencia de datos** permite a un usuario transferir unidades de datos (**datagramas**) a otro usuario especificado.

Cada unidad de datos debe ir acompañada por la dirección de transporte del usuario destino.

Este servicio (modo no conexión) está recomendado para aplicaciones que:

- Involucran interacciones de términos cortos de petición/respuesta.
- Presentan un alto nivel de redundancia.
- Son reconfigurables dinámicamente.
- No requieren garantía en la distribución de los datos secuencialmente.

La siguiente tabla (tabla 6-6) muestra las funciones que se deben utilizar en la fase de transferencia de datos:

Función	Descripción
t_rcvudata	Recibe un datagrama enviado por otro usuario de transporte.
t_rcvuderr	Recibe un mensaje de error asociado con el anterior envío.
t_sndudata	Envía un datagrama a un usuario destino especificado.

Tabla 6-6: Funciones TLI utilizadas en una transferencia de datos (modo no-conexión).

Bibliografía

Bibliografía.-

Internetworking with TCP/IP. Volumen I:Principles, Protocols and Architecture
Douglas Comer
Prentice Hall

TCP&IP Related Protocols
Uyless Black
McGraw Hill

A guide to the TCP/IP Protocol Suite
Floyd Wilder
Artech House

Redes de Ordenadores
Andrew S. Tanenbaum
Prentice Hall

Comunicación de Datos, Redes de Computadores y Sistemas Abiertos
Fred Halsall
Addison Wesley

TCP/IP and NFS Internetworking in a UNIX environment
Michael Santifaller
Addison Wesley

Protocolos de Comunicaciones para Sistemas Abiertos
José Miguel Alonso
Addison Wesley

OSI Explained. End-to-End computer communication standards
John Henshal and Sancy Shaw
Ellis Horwood 1990

Implementing OSI Networks
Gerald D. Cole
Wiley 1990

Redes de Computadores
Andrew S. Tanenbaum
Prentice Hall 1991

Manuales del SDK de Novell Netware

LAN Protocol Handbook
Mark A. Miller
Prentice Hall