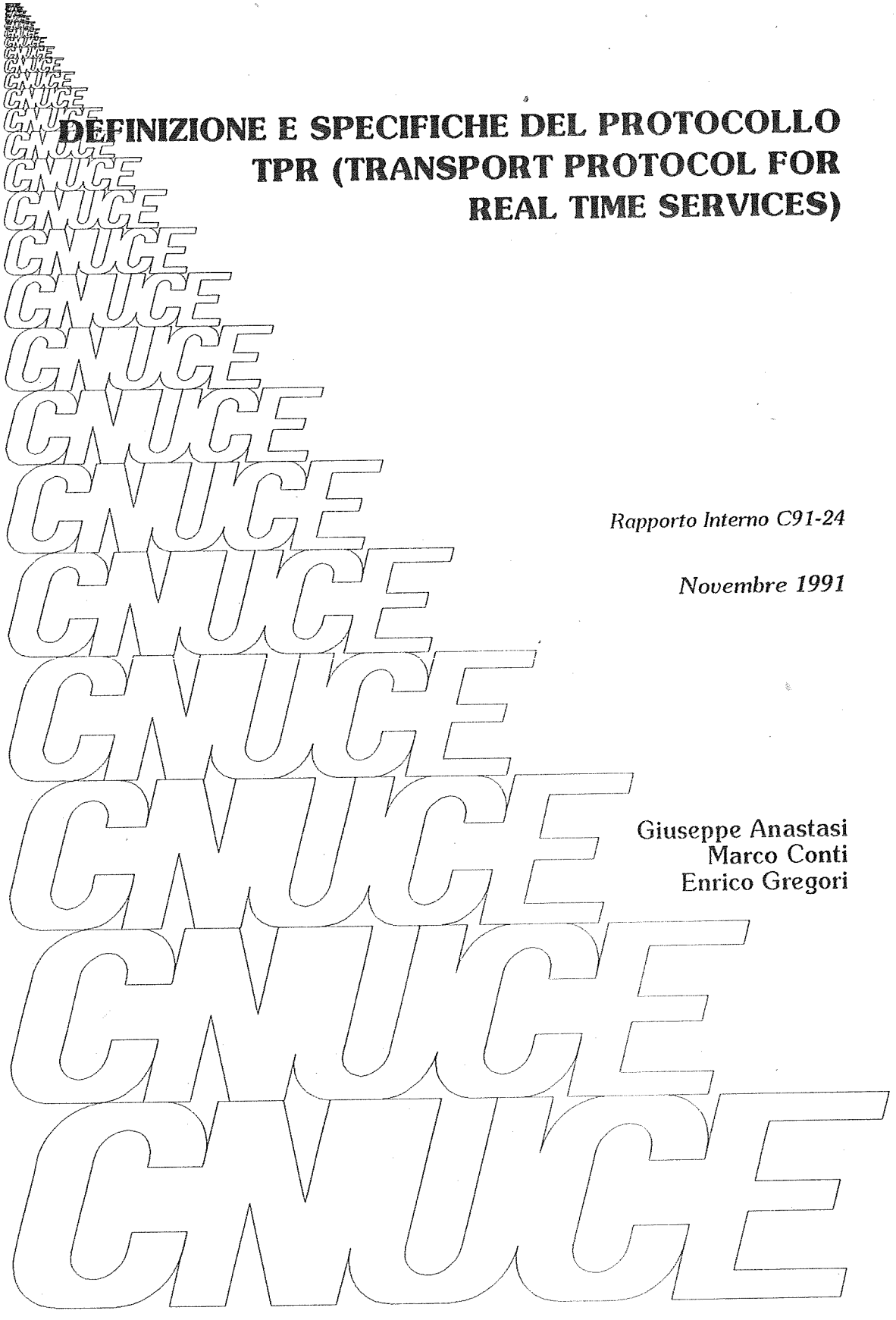

DEFINIZIONE E SPECIFICHE DEL PROTOCOLLO TPR (TRANSPORT PROTOCOL FOR REAL TIME SERVICES)

Rapporto Interno C91-24

Novembre 1991

Giuseppe Anastasi
Marco Conti
Enrico Gregori

DEFINIZIONE E SPECIFICHE DEL PROTOCOLLO TPR

(Transport Protocol for Real time services)

Versione 1.0

Giuseppe Anastasi*

Marco Conti**

Enrico Gregori**

* Laureato in Ingegneria Elettronica

** CNR - Istituto CNUCE

Indice

Introduzione

Il TPR (Transport Protocol for Real time services) e` un protocollo di trasporto per la trasmissione del traffico "real time" su una rete a commutazione di pacchetto. Questa puo` essere costituita internamente da una o piu` sottoreti anche con caratteristiche diverse fra loro ([ANAS90], [ANAS91], [CONT91]). L'unica condizione e` che la rete deve essere in grado di garantire un servizio con le seguenti caratteristiche:

- Ritardo di trasferimento dei pacchetti limitato
- Packet Loss Rate non superiore a una certa soglia
- Assenza di congestione almeno per una classe di utenti a piu` alta priorita

Il TPR prevede l'uso di un certo numero di "meccanismi di protocollo" per adeguare la Qualita` del Servizio offerta dalla rete a commutazione di pacchetto alle esigenze degli utenti. Alcuni di questi meccanismi (pacchettizzazione, depacchettizzazione) si ritrovano in tutti i protocolli di trasporto tradizionali (TCP, OSI TP, ecc.) mentre altri (assorbimento del jitter, compensazione delle informazioni perdute, correzione del drift, ecc.) sono stati inseriti allo scopo di ottenere un servizio di trasporto "real time" ([FERR90], [KIM88]). I meccanismi di protocollo vengono attivati

in base alla Qualita del Servizio richiesta dall'utente all'atto dell'apertura della connessione. In generale, pertanto, ad applicazioni di tipo diverso potra` corrispondere l'attivazione di meccanismi di protocollo diversi.

1. Meccanismi di Protocollo

La figura 1.1 raffigura il livello di trasporto di un'architettura di rete. La parte attiva del livello e` l'*entita` di trasporto* che sfruttando i servizi offerti dal livello sottostante fornisce ai suoi utenti il *servizio di trasporto.*

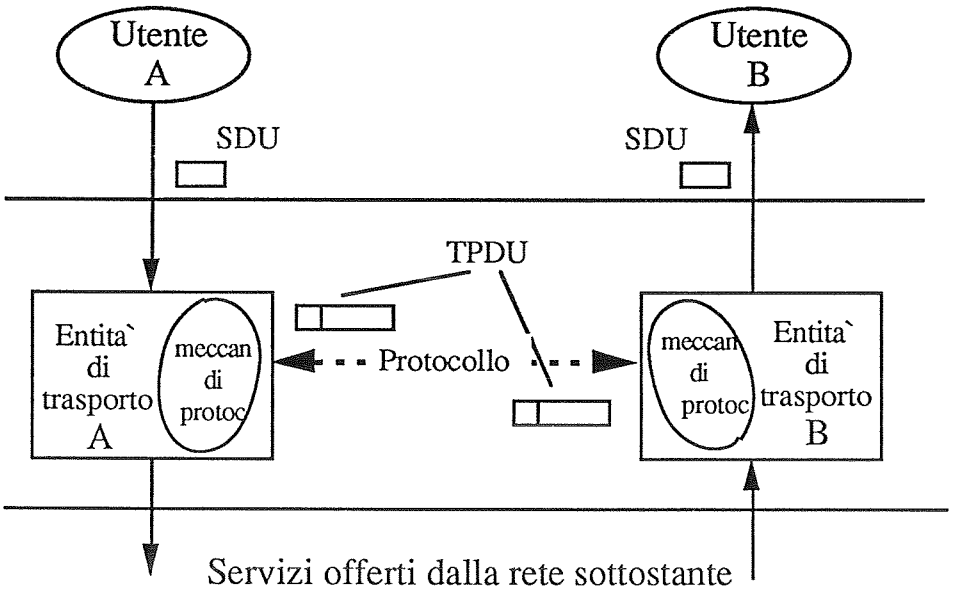

Figura 1.1 - Interazione fra gli utenti e il livello di trasporto e fra le due entita` di trasporto

Come si vede dalla figura il passaggio delle informazioni da e verso l'utente avviene sotto forma di *SDUs* (*Service Data Units*). Le entita` di trasporto comunicano invece tra di loro scambiandosi (mediante i

servizi offerti dal livello sottostante) *Transport Protocol Data Units (TPDUs)*. Il protocollo TPR prevede diversi tipi di TPDUs. Quelle usate per il trasporto delle informazioni sono chiamate DT (DATA) TPDUs. Nel seguito, quando non espressamente indicato, per TPDU si intendera` una DT TPDU. Essa consiste di un campo informazione e di una intestazione ("header") contenente informazioni di controllo.
Il servizio di trasporto previsto dal protocollo TPR e` di tipo "connection oriented". Durante la fase di apertura della connessione l'utente richiede all'entita` di trasporto la Qualita` del Servizio (QoS) di cui ha bisogno per trasmettere correttamente le proprie informazioni. Contemporaneamente egli fornisce opportune informazioni sul traffico che intende inviare attraverso la connessione. In tal modo l'entita` di trasporto puo` verificare se e` in grado, o meno, di garantire la QoS richiesta. Se le esigenze dell'utente non possono essere soddisfatte la richiesta di apertura della connessione viene respinta.
L'utente e` abilitato a specificare all'entita` di trasporto i seguenti parametri:

- Massimo ritardo di trasferimento tollerabile (D_{umax}).
- Valore massimo del jitter del ritardo che puo` essere accettato (J_{umax}).
- Soglia massima per la loss rate (L_{umax})

L'utente puo` inoltre richiedere che siano svolti i seguenti servizi:

- Compensazione delle informazioni perdute
- Ricostruzione, in ricezione, della frequenza del trasmettitore

Per soddisfare i requisiti degli utenti l'entita` di trasporto si serve di alcuni *meccanismi di protocollo* che hanno la funzione di migliorare la Qualita` del Servizio fornita dalla rete sottostante. Il meccanismo per l'*allocazione della banda* interviene durante la fase di apertura della connessione. Il meccanismo di *controllo sulla velocita` di trasmissione* evita che un utente possa inviare informazioni con una Bit rate superiore a quella dichiarata. Il meccanismo di *pacchettizzazione* ha la funzione di confezionare le TPDUs partendo dalle SDUs ricevute dall'utente. In fase di ricezione intervengono il meccanismo per la *compensazione delle informazioni perdute* , il meccanismo per l'*assorbimento del jitter* del ritardo, il meccanismo di *depacchettizzazione* e quello per la *correzione del "drift"* fra la frequenza del trasmettitore e quella del ricevitore.
Questi meccanismi vengono attivati, in modo indipendente l'uno dall'altro, sulla base dei parametri della QoS indicati dall'utente all'atto dell'apertura della connessione. Ad esempio, se i requisiti dell'utente riguardo al jitter massimo possono essere soddisfatti unicamente sulla base delle prestazioni della rete sottostante l'entita` di trasporto non attiva il meccanismo per l'assorbimento del jitter.
Nel seguito analizzeremo nel dettaglio ciascuno dei meccanismi elencati sopra.

1.1 Meccanismi di Allocazione della Banda e di Controllo del Flusso

Il meccanismo di allocazione della banda provvede ad assegnare, a ciascuna connessione, la banda necessaria a garantire la Qualita del Servizio richiesta dagli utenti. Cio` deve avvenire senza deteriorare la Qualita del Servizio che viene gia` fornita agli altri utenti.
Se a ciascun utente con esigenze di trasmissione in tempo reale viene allocata una banda maggiore o uguale alla sua bit rate massima e se la banda complessivamente allocata non supera la capacita` della rete allora e` possibile rispettare i vincoli temporali degli utenti evitando,

al contempo pericoli di congestione. Agli utenti che non presentano vincoli sui tempi di trasmissione non viene assegnata una banda garantita. Percio` essi possono trasmettere le loro informazioni solo quando la capacita` della rete non e` completamente saturata dagli utenti a banda garantita.
Per gli utenti con velocita` di trasmissione variabile l'allocazione della banda puo` essere fatta seguendo una politica probabilistica basata sulla tecnica del *multiplexing* statistico. Il protocollo di trasporto dispone di un meccanismo di controllo del flusso la cui funzione e` di bloccare alcuni utenti quando tutti vorrebbero trasmettere simultaneamente.

1.2 Meccanismo di Controllo sulla Velocita` di Trasmissione

La Qualita` del Servizio richiesta puo` essere garantita dal sistema di trasporto solo se tutti gli utenti rispettano le Caratteristiche di Traffico dichiarate all'atto dell'aperttura della connessione. Tuttavia l'entita` di trasporto dispone di meccanismo di controllo che provvede a chiudere la connesione se l'utente invia le sue informazioni con una Bit rate maggiore di quella dichiarata.

1.3 Meccanismo di Pacchettizzazione

Questo meccanismo provvede a formare le TPDUs partendo dalle SDUs ricevute dall'utente. Se quest'ultime hano una dimensione troppo grande per poter essere spedite mediante un'unica TPDU vengono segmentate, cioe` suddivise in piu` segmenti i quali vengono spediti separatamente. Giunti a destinazione i vari segmenti vengono raccolti dal meccanismo di depacchettizzazione che provvede a ricostruire le SDUs originarie. Se invece le SDUs hanno dimensioni troppo piccole vengono spedite a gruppi di N SDUs per ogni TPDU. Il numero N viene scelto in base ad un compromesso fra l'"overhead" e il ritardo di pacchettizzazione (tempo necessario per

formare una TPDU). Il primo, infatti e` tanto piu` piccolo quanto piu il campo "informazione" e` grande rispetto allo "header" in una TPDU. Al contrario il ritardo di pacchettizzazione aumenta al crescere di N.
Appena pronte le TPDUs vengono inviate dall'entita` di trasporto alla sua controparte che si trova all'altro capo della connessione.

1.4 Considerazioni sulla Loss rate

Durante la trasmissione alcune TPDUs possono andare perdute. Le ragioni per cui una TPDU non raggiunge l'esatta destinazione possono essere diverse: errori di trasmissione, buffer overflows, eccessivo ritardo ecc.
Alcune applicazioni richiedono che la percentuale delle TPDUs che possono perdersi, rispetto alla totalita` delle TPDUs trasmesse, sia inferiore a una certa soglia ([FERR91]). Per venire incontro a tali applicazioni il TPR permette all'utente di specificare la massima loss rate che puo` essere tollerata.
Chiaramente l'entita` di trasporto deve conoscere la loss rate massima che puo` essere introdotta dalla rete sottostante e deve essere in grado di valutare le eventuali perdite che si verificano al livello di trasporto. In tal modo e` possibile calcolare la massima loss rate $L_t$ con cui puo` essere fornito il servizio di trasporto. Ovviamente condizione necessaria perche` si possa aprire la connessione e` che:

$$L_t \leq L_{umax} \qquad (1.1)$$

dove L_{umax} e` la loss rate massima che puo` essere tollerata dall'utente.

1.5 Meccanismo per la Compensazione delle Perdite

Per alcune applicazioni di tipo real time, ad esempio video e/o voce, e` necessario che vi sia una sincronizzazione tra trasmettitore e ricevitore. In altre parole e` necessario che la sequenza di bit in ricezione sia identica a quella trasmessa con una semplice traslazione temporale dovuta al ritardo di trasferimento. La perdita di una TPDU non solo distrugge il sincronismo fra trasmettitore e ricevitore ma, nei casi in cui viene usata una tecnica di codifica ricorsiva, non permette neanche la corretta decodifica delle informazioni successive a quelle non pervenute. Per limitare i danni si ricorre a tecniche di compensazione che consistono nel rimpiazzare la TPDU mancante con una TPDU consegnata in precedenza o con una TPDU particolare.
La perdita di una TPDU puo` essere rilevata analizzando il numero di sequenza (contenuto in un apposito campo dello "header") man mano che le TPDUs arrivano a destinazione. Infatti, nell'ipotesi che la rete sottostante preservi l'ordinamento delle TPDUs, se due TPDUs arrivate in successione contengono numeri di sequenza non consecutivi vuol dire che almeno una TPDU non e` stata ricevuta. In tal caso il meccanismo per la compensazione delle perdite provvede a rimpiazzare la TPDU mancante con una TPDU fittizia. Quando il meccanismo di depacchettizzazione incontra la TPDU fittizia si comporta in modo differente a seconda che durante l'apertura della connessione sia stata richiesta, o meno, la compensazione delle informazioni perdute.
Se tale compensazione non e` stata richiesta, in corrispondenza della TPDU fittizia la consegna dell'informazione all'utente si interrompe per un tempo pari alla durata dell'informazione contenuta nella TPDU non ricevuta.
In caso contrario il meccanismo di depacchettizzazione rimpiazza la TPDU fittizia con la TPDU sostitutiva. Per quanto riguarda la scelta della TPDU sostitutiva i seguenti metodi sono praticabili:

- La TPDU mancante viene sostituita da una TPDU contenente tutti zeri nel campo informazione.

- La TPDU mancante viene rimpiazzata dalla TPDU immediatamente precedente.

Per le applicazioni video si ricorre generalmente ad una delle seguenti tecniche ([COCH85], [DEVA88]):

Time Masking: la TPDU non ricevuta viene sostituita con quella corrispondente nella stessa linea del quadro precedente.

Space Masking: la TPDU non ricevuta viene sostituita con quella corrispondente nella linea precedente dello stesso quadro.

1.6 Considerazioni sul Ritardo e sul Jitter

Il ritardo "end to end", cioe` il tempo necessario per trasportare le informazioni dal mittente al destinatario, e` dato dalla seguente relazione:

$$D_e = D_p + D_n \qquad (1.2)$$

dove D_p e` il ritardo di pacchettizzazione, costante, e $D_n$ rappresenta il ritardo introdotto dalla rete sottostante. Poiche` quest'ultima e` una rete a commutazione di pacchetto il ritardo De varia per ciascuna TPDU. In generale tale ritardo avra` una componente fissa e una variabile. Pertanto la (1.2) puo` essere riscritta nella seguente forma:

$$D_e = D_f + D_v \qquad (1.3)$$

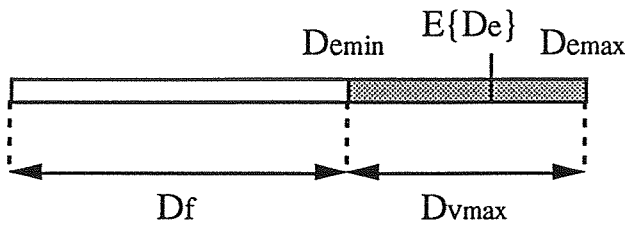

Figura 1.2 - Rappresentazione grafica del ritardo "end to end"

Ancora, osservando la figura 1.2, il ritardo D_e puo` essere caratterizzato mediante il suo valore medio $E\{D_e\}$ e il suo campo di variabilita (D_{emax} - D_{emin}) che coincide, ovviamente, col massimo di D_v.
Condizione necessaria perche` possa essere accettata una connessione e` che il ritardo D_e sia minore o uguale al massimo ritardo che puo` essere tollerato dall'utente:

$$D_{emax} \leq D_{umax} \qquad (1.4)$$

In aggiunta al limite massimo per il ritardo l'utente puo` essere interessato anche a un limite massimo per il jitter del ritardo. In altre parole l'utente puo` richiedere che oltre alla 1.4 sia verificata le seguente relazione:

$$E\{D_e\} - J_{umax} \leq D_e \leq E\{D_e\} + J_{umax} \qquad (1.5)$$

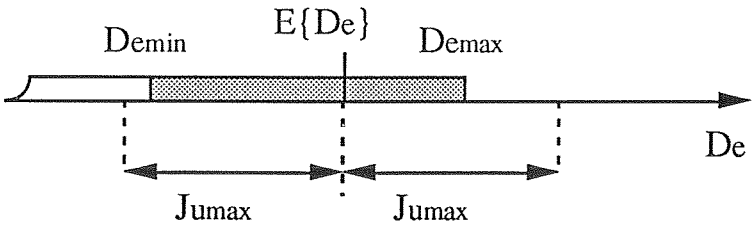

Figura 1.3a - Il requisito sul jitter puo` essere soddisfatto sfruttando unicamente le prestazioni della rete sottostante.

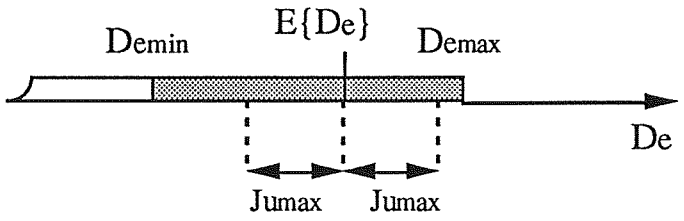

Figura 1.3b - Il requisito sul jitter non puo` essere soddisfatto sfruttando unicamente le prestazioni della rete sottostante.

Ci sono casi (figura 1.3a) in cui tale requisito puo` essere soddisfatto dal livello di trasporto sfruttando unicamente le prestazioni della rete sottostante. Se cosi` non e` (figura 1.3b) viene attivato il meccanismo per l'assorbimento del jitter la cui funzione e` di rendere costante, per ciascuna TPDU, il ritardo "end to end".
Nel prossimo paragrafo analizzeremo dettagliatamente il principio di funzionamento di tale meccanismo.

1.7 Meccanismo per l'Assorbimento del Jitter

Tale meccanismo richiede la conoscenza del massimo valore del ritardo variabile D_{vmax} (= D_{emax} - D_{emin}). Ci sono casi in cui tale limite puo` essere conosciuto con esattezza, altri in cui un valore massimo per $D_v$ non esiste. Supponendo di essere nella seconda situazione si puo` pensare di fissare un limite massimo D_{vmax} tale che la probabilita` che una TPDU arrivi con $D_v > D_{vmax}$ sia minore o uguale a un certo valore prefissato:

$$\text{Prob}\ \{D_v > D_{vmax}\} \leq p \qquad (1.6)$$

Tutte le TPDUs che arrivano con $D_v > D_{vmax}$ sono da considerarsi perdute. Di esse si dovra` tenere conto quando si calcola la massima loss rate che puo` essere garantita ([CHEN89], [FERR90]).

L'assorbimento del jitter puo` essere ottenuto ritardando la consegna delle informazioni di una quantita` D_j scelta in modo tale che il ritardo "end to end" cosi` ottenuto:

$$D_{e(j)} = D_f + D_v + D_j \qquad (1.7)$$

risulti costante per ciscuna TPDU ([MONT83]). Ovviamente il ritardo aggiuntivo D_j dovra` essere diverso per ciascuna TPDU. Ma come viene scelto D_j ?.
Se e` $D_v$ noto, anche solo per la prima TPDU, e` sufficiente ritardare la consegna delle informazioni contenute nella suddetta TPDU di una quantita pari alla differenza fra il massimo ritardo variabile e il ritardo variabile effettivamente sperimentato:

$$D_{j1} = D_{vmax} - D_{v1} \qquad (1.8)$$

Se, invece, Dv non e` noto neanche per la prima TPDU, e` necessario considerare una sua stima approsimmata di D_{v1}. L'approssimazione piu` semplice ma anche piu` grossolana consiste nel supporre D_{v1} uguale a zero. In tal modo, per la (1.8) sara`:

$$D_{j1} = D_{vmax} \qquad (1.9)$$

Per quanto riguarda le TPDUs successive se unaTPDU e` stata spedita, poniamo, α msec dopo la precedente, le informazioni in essa contenute cominceranno ad essere consegnate all'utente α msec dopo l'inizio della consegna delle informazioni della precedente TPDU.
Se e` noto D_{v1} il ritardo "end to end" risulta:

$$D'_{e(j)} = D'_{e(j)max} = D_f + D_{vmax} \qquad (1.10)$$

Se, invece, il ritardo D_j e` dato dalla (1.9) sara`:

$$D_{e(j)} = D_f + D_v + D_{vmax} \qquad (1.11)$$

Nella seconda ipotesi il ritardo massimo si ha quando $D_{v1} = D_{vmax}$ allorche` risulta:

$$D_{e(j)max} = D_f + 2 \bullet D_{vmax} \qquad (1.12)$$

La (1.10) e la (1.12) forniscono il limite massimo per il ritardo "end to end" quando e` attivo il meccanismo per l'assorbimento del jitter. Tale limite che puo` essere calcolato durante la fase di apertura della connessione dovra` essere minore o uguale al massimo ritardo tollerabile dall'utente (D_{umax}).
Per dimensionare il buffer di uscita, in cui vengono poste le informazioni in attesa della consegna all'utente, si osservi che il numero massimo di SDUs che possono arrivare nell'intervallo compreso fra l'istante di arrivo della prima TPDU (e quindi della prima SDU) e l'istante di inizio della consegna della prima SDU e` dato da:

$$N = 1 + \left\lfloor \frac{2 \bullet D_{vmax}}{T} \right\rfloor \qquad (1.13)$$

Di conseguenza la dimensione del buffer dovra` essere:

$$B = N \bullet S_M = (1 + \left\lfloor \frac{2 \bullet D_{vmax}}{T} \right\rfloor) \bullet S_M \qquad (1.14)$$

con S_M dimensione massima, in bytes, di una SDU.

1.8 Meccanismo di Depacchettizzazione

Questo meccanismo ha la funzione di rigenerare le SDUs originarie. Se una TPDU contiene piu` di una SDU il meccanismo di depacchettizzazione provvede semplicemente ad estrarre le varie SDUs dalla TPDU. Se invece una TPDU contiene solo un segmento di SDU il meccanismo raccoglie i vari segmenti riassemblando la SDU di partenza.

Una volta pronte le SDUs vengono consegnate all'utente a intervalli pari al tempo di interrarrivo T.

1.9 Meccanismo per la Correzione del Drift

La frequenza f_r con cui il meccanismo di depacchettizzazione consegna le SDUs all'utente, e la frequenza f_t con cui le SDUs vengono passate dal trasmettitore all'entita` di trasporto hanno lo stesso valore nominale:

$$f_n = \frac{1}{T} \tag{1.15}$$

Le frequenze effettive presentano pero` uno scostamento ("drift") dal valore nominale. E poiche` i due clocks da cui le frequenze sono, in generale, ricavate sono indipendenti i due scostamenti sono quasi sicuramente diversi. In altre parole la differenza fra f_t e f_r, anche se piccola, non e` nulla. Cio` puo` portare delle complicazioni notevoli soprattutto se la Bit rate dell'applicazione e` elevata. Infatti se la f_t e` maggiore della $f_r$ le informazioni tendono ad accumularsi nel buffer di uscita del TPR il quale puo` andare in overflow con conseguente perdita di informazioni. Se viceversa le informazioni giungono piu` lentamente di quanto escano il buffer tende a svuotarsi e puo` non essere piu` garantita la continuita` del flusso di informazioni verso l'utente ricevitore.
Pertanto il TPR prevede, in ricezione, l'uso di un meccanismo per la ricostruzione della frequenza del trasmettitore f_t.
Tale meccanismo usa un metodo per la stima del drift fra le due frequenze che oltre ad essere applicabile anche in situazioni di jitter molto ampio presenta le caratteristiche di semplicita` e di robustezza ([ANAS90]). Questo metodo si basa sull'osservazione del livello di riempimento del buffer espresso come numero di SDUs presenti Tale livello viene registrato ogni volta che viene consegnata

all'utente una SDU[1]. Indichiamo con X_i il livello di riempimento del buffer all'istante in cui viene consegnata la i-esima SDU. Se consideriamo i valori di X_i all'interno di un certo intervallo di tempo essi possono variare (anche notevolmente) per effetto del jitter. Possiamo pero` considerare il livello medio del buffer, $X^*$, durante tale intervallo che e` dato dalla seguente espressione:

$$X^* = \frac{\sum_{i=n+1}^{n+F} X_i}{F} \qquad (1.16)$$

Nella precedente relazione F indica la finestra di osservazione su cui e` stata effettuata la misura. Gli estremi della sommatoria denotano il numero di sequenza delle SDUs che vengono consegnate al ricevitore rispettivamente per prima e per ultima durante l'intervallo di osservazione. La durata di quest'ultimo e` data da:

$$T_O = F \bullet T \qquad (1.17)$$

In questo modo ad ogni finestra temporale di durata T_O viene associato un livello medio del buffer X^*. Confrontando l'ultimo valore calcolato di X^* con quello relativo alla finesta precedente si puo` rilevare la presenza di un eventuale drift fra $f_t$ e $f_r$. Se $\Delta X^*$ e` la variazione del livello medio rispetto all'intervallo precedente lo scostamento fra le frequenze risulta:

$$\Delta = \frac{f_r - f_t}{f_n} = - \Delta X^* \qquad (1.18)$$

In questo modo e` possibile ricostruire la frequenza del trasmettitore mediante la relazione:

$$f_t' = f_r + \Delta X^* \bullet f_n \qquad (1.19)$$

[1]Se le SDUs hanno dimensioni troppo piccole si puo` misurare il livello del buffer dopo la consegna di m SDUs con m scelto opportunamente

Nella precedente abbiamo posto f_t' e non f_t in quanto a causa degli inevitabili errori la frequenza ricostruita non corrisponde esattamente alla frequenza del trasmettitore. Tuttavia poiche` il meccanismo e` costantemente attivo eventuali errori possono essere determinati al passo successivo.
La precisione del metodo dipende da quanto grande viene scelto il valore di F ovvero la durata dell'intervallo di osservazione T_O.
A seconda del tipo di applicazione la variazione istantanea della frequenza del ricevitore potrebbe comportare dei disturbi[2] molto evidenti. In tali casi e` conveniente eseguire una sola correzione molto precisa per cui il valore di $T_O$ va` scelto sufficientemente grande. In casi in cui la correzione della f_r non comporta disturbi apprezzabili si puo` scegliere un valore piu` piccolo per T_O ed eseguire piu` correzioni.
Bisogna tenere presente che la prima correzione viene effettuata un tempo pari a $2 \cdot T_O$ dopo la consegna della prima SDU. Durante questo intervallo, supponendo che lo scostamento fra le due frequenze sia uguale, in valore assoluto, al massimo possibile ($\Delta = \Delta max$) e con segno tale da svuotare il buffer ($f_r \geq f_t$), il numero massimo di SDUs che posono mancare e` dato da:

$$\Delta N = 1 + \left\lfloor (1 + \left\lfloor \frac{2 \cdot T_O}{T} \right\rfloor) \cdot \Delta_{max} \right\rfloor \qquad (1.20)$$

Pertanto occorre ritardare la consegna della prima SDU di un ulteriore intervallo pari a:

$$Dd = \Delta N \cdot T \qquad (1.21)$$

---

2 Nel caso del video conviene differire la correzione della frequenza del ricevitore all'inizio del nuovo quadro.

Di conseguenza il massimo ritardo "end to end", quando sono attivi sia il meccanismo per l'assorbimento del jitter sia quello per la correzione del drift, e` dato dalla seguente relazione:

$$D_{e(j)max} = D_f + D_{vmax} + Dd \tag{1.22}$$

Inutile dire, a questo punto, che tale ritardo dovra` soddisfare i requisiti degli utenti sul ritardo massimo.
Facendo un passo indietro, se e` ancora $\Delta = \Delta max$ ma $f_r \leq f_t$ il numero massimo di SDUs che potrebbero non trovare posto nel buffer e` dato ancora dalla (1.20). Di conseguenza e` necessario allungare il buffer di una quantita` pari a ΔN. In presenza di drift, pertanto, la dimensione del buffer di uscita e` data dalla seguente relazione:

$$B_d = (1 + \left\lfloor \frac{2 \bullet D_{vmax}}{T} \right\rfloor + 2 \bullet \Delta N) \bullet S_M \tag{1.23}$$

Il metodo ora descritto e` estremamente semplice e puo` essere reso robusto quanto si vuole aumentando l'intervallo di osservazione. E` di applicabilita` piu` generale di altri metodi analoghi descritti in letteratura ([COCH85], [DEPR87], [VAKI89]) in quanto non pone limitazioni sull'ampiezza del jitter. Inoltre puo` essere usato sia con applicazioni con velocita` di trasmissione costante sia con applicazioni a Bit rate variabile. L'unica limitazione e` che il passaggio delle informazioni da e verso l'utente avviene per mezzo di SDU di dimensione variabile con tempo di interarrivo costante.

2. Servizi

2.1 Primitive di Servizio

I *servizi* che il livello di trasporto mette a disposizione dei suoi utenti sono definiti mediante le *primitive di servizio*. Nella tabella 2.1 e` riportata la lista di tali primitive con i relativi parametri.
Le primitive OPEN.request, CONNECT.request, B_OPEN.request, B_CONNECT.request e CONNECT.indication intervengono durante la fase di apertura della connessione. La DATA.request e la DATA.indication operano il passaggio delle informazione da e verso l'utente, rispettivamente. Infine la DISCONNECT.request e la DISCONNECT.indication vengono generate durante la fase di chiusura della connessione.
Come si puo` vedere dalla tabella 2.1 quasi tutte le primitive ora introdotte presentano dei parametri.
I *filter addresses* sono gli indirizzi degli utenti con i quali l'utente che ha inoltrato la OPEN.request (o la B_OPEN.request) e` disposto a stabilire una connessione. L'*opening address* e` invece l'indirizzo dell'utente che ha inoltrato la OPEN.request (o B_OPEN.request). I parametri della Qualita` del Servizio (*QoS parameters*) e i parametri delle Caratteristiche del Traffico (*CoT parameters*) saranno analizzati separatamente, rispettivamente nei paragrafi 2.2 e 2.3.

Primitiva	Parametri
OPEN.request	**(filter addresses opening address QoS parameters CoT parameters)**
B_OPEN.request	**(filter addresses opening address QoS parameters**
CONNECT.request	**(called address calling address QoS parameters CoT parameters)**
B_CONNECT.request	**(calling address QoS parameters CoT parameters)**
CONNECT.indication	**(called address calling address)**
DATA.request	**(SDU)**
DATA.indication	**(SDU)**
DISCONNECT.request	**()**
DISCONNECT.indication	**(reason)**

Tabella 2.1 - Primitive di servizio con relativi parametri.

Il *called address* e` l'indirizzo dell'utente al quale si richiede di stabilire la connessione mentre il *calling address* e` l'indirizzo dell'utente che ha inoltrato la CONNECT.request.(o B_CONNECT.request), cioe` l'utente che richiede l'apertura della connessione.
Le *SDUs* (*Service Data Units*) sono i blocchi di informazione passati dall'utente all'entita` di trasporto in fase di trasmissione e in direzione opposta in fase di ricezione. Infine il parametro *reason*,

nella DISCONNECT.indication riporta il motivo che ha condotto alla chiusura della connessione.

2.2 I parametri della Qualita` di Servizio

Quando l'utente richiede un servizio di trasporto esige che gli venga garantita una certa Qualita` di Servizio. Quest'ultima dipende fortemente dall'applicazione cui il servizio e` destinato. Si capisce, infatti, che un file server ha delle esigenze completamente diverse da quelle di una sorgente video o voce ([FERR90]).
La Qualita` del Servizio viene richiesta dall'utente all'entita` di trasporto durante la fase di apertura della connessione mediante i parametri della QoS. Nella tabella 2.2 sono mostrati i parametri della QoS previsti dal protocollo TPR.
Il parametro Dumax serve a specificare il massimo ritardo "end to end" che puo` essere tollerato. E` possibile anche indicare il limite massimo sul "jitter" (variabilita`) del ritardo usando il parametro Jumax . In modo del tutto analogo, il parametro Lumax permette all'utente di fissare una soglia massima per la Loss rate.
L'utente ha anche la possibilita` di richiedere che venga effettuata la compensazione delle informazioni perdute mediante il parametro booleano Lc. L'altro parametro booleano, Dc, permette invece di attivare il meccanismo per la correzione del drift.
Dal momento che e` possibile specificare un limite massimo o richiedere la compensazione delle TPDUs perdute o la correzione del drift sia per le informazioni che devono essere ricevute sia per quelle che devono essere inviate, l'utente e` tenuto a indicare due valori per ciascuno dei parametri QoS sopra menzionati. A meno che egli non intenda solo trasmettere o solo ricevere, nel qual caso indichera`, ovviamente, solo i valori che sono significativi.

Parametro	Simbolo
Ritardo massimo	**Dumax**
Jitter massimo	**Jumax**
Loss rate massima	**Lumax**
Compensazione perdite	**Lc**
Correziione drift	**Dc**
Bit rate massima in Rx	**BmaxR**
Dimensione massima SDU	**SmaxR**
Tempo di interarrivo min	**TminR**

Tabella 2.2 - Parametri della Qualita` del Servizio.

Gli ultimi tre parametri riportati nella tabella 2.2 riguardano, invece, solo la ricezione e percio` devono essere presi in considerazione solo da quegli utenti che intendano ricevere. Tali paarmetri permettono di specificare la massima Bit rate con cui le informazioni possono essere ricevute (BmaxR), la dimensione massima che una SDU deve avere perche` possa essere ricevuta dall'utente (SmaxR) e infine il Tempo di interrarivo minimo (o la frequenza massima) con cui le SDUs devono essere consegnate dall'entita` di trasporto per potere essere ricevute correttamente dall'utente (TminR).

2.3 I parametri delle Caratteristiche di Traffico

Affinche` il livello di trasporto possa stabilire se e` in grado di soddisfare le esigenze degli utenti (rese note mediante i parametri QoS) gli stessi utenti devono fornire alcune indicazioni sul tipo di traffico che intendono trasmettere. Cio` avviene mediante i parametri delle Caratteristiche di Traffico (CoT) che devono essere specificati da tutti gli utenti che hanno informazioni da trasmettere o all'atto dell'apertura della connessione o all'atto della generazione della OPEN.request nel caso di apertura di connessione passiva.
Nella tabella 2.3 sono riporatti i parametri CoT previsti dal protocollo TPR.

Parametro	Simbolo
Bit rate media (I)	Bav
Bit rate massima	BM
Dimensione massima SDU	SM
Tempo di interrarivo	T

Tabella 2.3 - Parametri delle Caratteristiche del Traffico

I parametri B_{av} e B_M rappresentano, rispettivamente, la Bit rate media calcolata su un opportuno intervallo temporale I e la Bit rate massima. Ovviamente tali parametri assumono lo stesso valore nel caso di applicazioni con Bit rate fissa.
Il parametro S_M indica la dimensione massima di una SDU che si vuole trasmettere. Infine il Tempo di interrarivo (T) specifica l'intervallo fra due SDUs consecutive. Tale parametro deve rimanere costante durante tutta la durata della connessione. Nel caso di

applicazioni con Bit rate fissa anche le SDU hanno lunghezza fissa mentre per applicazioni con velocita` di trasmissione variabile le SDUs hanno, in generale, dimensione diversa le une dalle altre.

2.4 Uso delle Primitive di Servizio

In questo paragrafo illustreremo l'uso delle primitive di servizio introdotte precedentemente e la relazione esistente fra di esse.
Le considerazioni riportate nei tre paragrafi seguenti non riguardano le connessioni del tipo trasmettitore broadcast-ricevitore. Questo caso verra` considerato, a parte, nel paragrafo 2.4.4.

2.4.1 Apertura di connessione

Nella figura 2.2 e` rappresentata la sequenza temporale delle primitive di servizio durante la fase di apertura di una connessione.

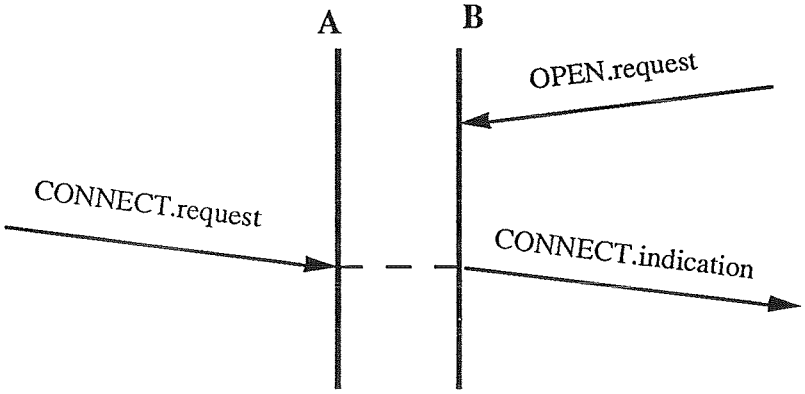

Figura 2.1 - Apertura di una connessione.

Se un utente B e` disposto ad accettare richieste di apertura di connessione da parte di alcuni utenti inoltra alla sua entita` di trasporto una OPEN.request indicando l'indirizzo di tali utenti.

Se invece un utente A intende aprire una connessione con l'utente B inoltra all'entita` di trasporto A una CONNECT.request con i parametri opportuni. Affinche` la connessione possa essere attivata e` necessario che:

1) - L'entita` di trasporto A disponga delle risorse necessarie per garantire la QoS richiesta dall'utente.

2) - L'utente B abbia precedentemente inoltrato alla sua entita` di trasporto la OPEN.request.

3) - Il campo *filter addresses* della OPEN.request inoltrata dall'utente B contenesse, eventualmente fra gli altri, l'indirizzo dell'utente A.

4) - I parametri QoS e (eventualmente) CoT contenuti nella CONNECT.request siano compatibili con i parametri QoS e (eventualmente) CoT indicati nella OPEN.request dall'utente B.

5) - L'utente B non sia impegnato contemporaneamente in connessioni con altri utenti che non gli consentono di aprire una nuova connessione.

Se tutte queste condizioni sono verificate l'entita` di trasporto B provvede ad inviare una CONNECT.indication al suo utente (figura 2.1).
Se, invece, le condizioni precedenti non sono tutte verificate, l'entita` di trasporto A inoltra un DISCONNECT.indication al suo utente (figure 2.2). Le ragioni del fallimento della richiesta di apertura di connessione sono riportate nel parametro *reason* della primitiva.

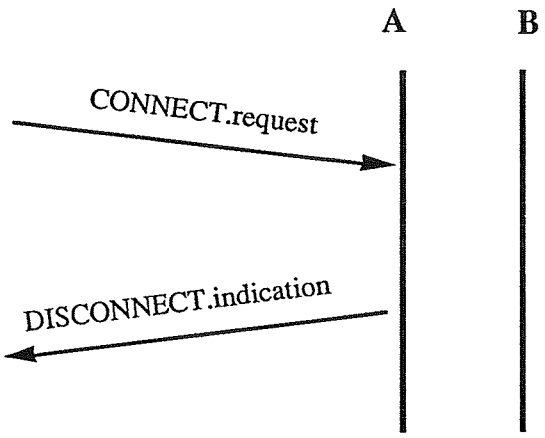

Figura 2.2 - Richiesta di apertura della connessione non soddisfatta

Un utente B che ha ricevuto una CONNECT.indication puo` rifiutare la richiesta di apertura di connessione inoltrando una DISCONNECT.request alla sua entita` di trasporto. Cio` provochera` l'invio di una DISCONNECT.indication da parte dell'entita A all'utente relativo (figura 2.3)

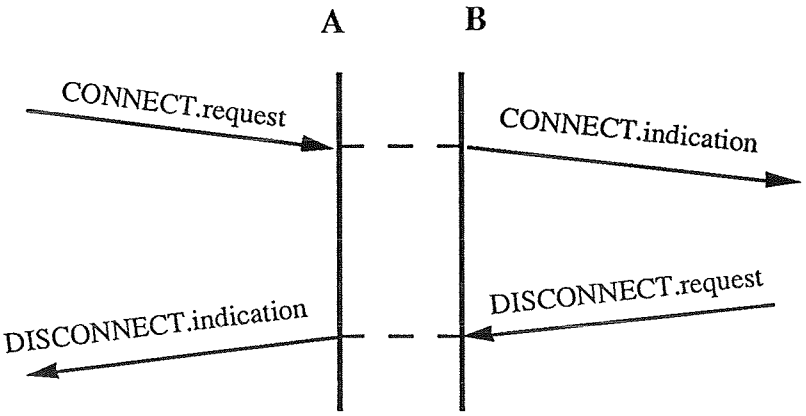

Figura 2.3 - Richiesta di connessione rigettata dall'utente interpellato.

Se, invece l'utente B accetta di aprire la connessione invia alla sua entita` di trasporto una DATA.request (eventualmente senza dati) alla quale sara` fatta corrispondere una DATA.indication all'altro capo (figura 2.4). In tal modo l'utente A capisce che la connessione

e` stata attivata e puo` cosi` iniziare a trasmettere le sue informazioni (se ne ha) o attendere l'arrivo di altre informazioni dall'utente B.

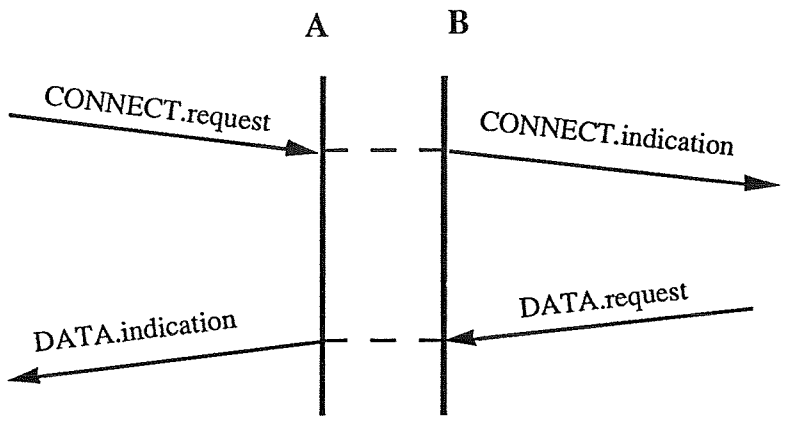

Figura 2.4 - Richiesta di connessione acettata dall'utente interpellato.

L'utente B potrebbe anche non rispondere alla CONNECT.indication. Per ovviare a tale situazione ogni volta che l'entita` di trasporto invia all'utente una richiesta di connessione fa` partire un "inactivity timer". Se l'utente non risponde prima dell'azzeramento del timer viene informata l'entita di trasporto remota la quale provvede a inoltrare una DISCONNECT.indication al suo utente. La situazione e` simile a quella rappresentata dalla figura 2.3.
Un terzo caso in cui l'utente che inizia la procedura per l'apertura di connessione riceve una DISCONNECT.indication si ha quando l'entita` di trasporto A non riceve alcuna risposta dall'entita di pari livello fino all'azzeramento di un "set up" timer. Tale timer viene attivato al momento in cui l'entita` A informa la sua controparte di avere ricevuto una richiesta di apertura di connessione da parte del proprio utente.

2.4.2 Trasferimento delle informazioni

Nella figura 2.5 e` mostrato lo scambio delle primitive di servizio durante la fase di trasferimento delle informazioni.

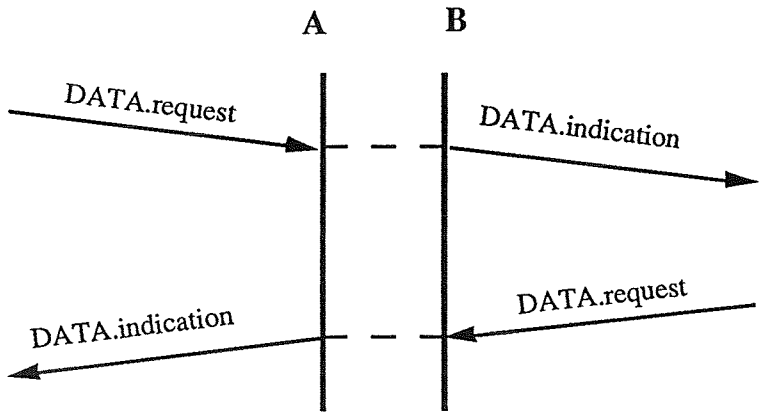

Fig. 2.5 - Uso delle primitive di servizio durante la fase di trasferimento delle informazioni.

La generazione di una DATA.request da parte di un utente causera` la sollecitazione di una DATA.indication verso l'altro utente.
La figura fa` riferimento al caso di connessione in cui il trasferimento delle informazioni avviene in entrambe le direzioni.

2.4.3 Chiusura di connessione

La chiusura di una connessione avviene sempre in modo *graceful*, cioe` senza perdita di informazioni.
L'utente A che vuole chiudere una connessione in cui e` impegnato inoltra una DISCONNECT.request (figura 2.6). Cio` generera` la sollecitazione di una DISCONNECT.indication verso l'utente B ma solo dopo che tutti i dati precedentemente inviati dall'utente A sono stati consegnati all'utente B. A questo punto anche B inoltrera` la sua DISCONNECT.request, svincolandosi cosi` dalla connessione.

Da notare che l'utente A considera chiusa la connessione subito dopo aver inoltrato la DISCONNECT.request.

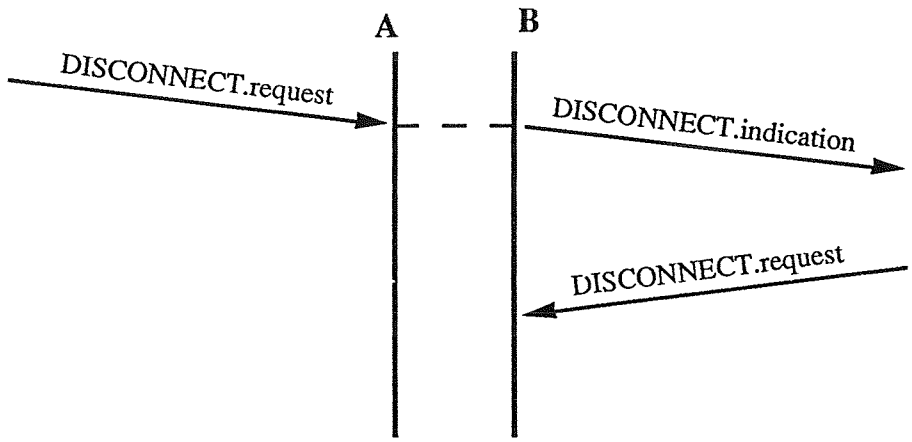

Fig. 2.6 - Uso delle primitive di servizio durante la fase di chiusura di una connessione.

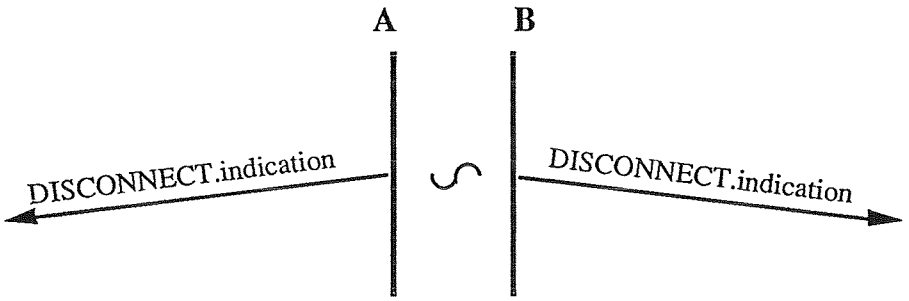

Figura 2.7 - Chiusura di connessione per motivi interni alla rete.

La figura 2.7 rappresenta il caso in cui la connessione non puo` essere mantenuta per malfunzionamenti interni alla rete. La sinusoide indica che non vi e` alcuna correlazione fra le due DISCONNECT.indication.

2.4.4 Trasmissioni broadcast

Consideriamo adesso il caso di trasmissione broadcast e supponiamo che l'utente A sia il trasmettitore e l'utente B uno dei ricevitori. In questo caso la connessione e` unidirezionale cioe` il trasferimento delle informazioni avviene solo dal trasmettitore al ricevitore.
Quando l'utente B vuole ricevere inoltra la B_OPEN.request (figura 2.8) indicando l'indirizzo del trasmettitore e i parametri QoS. Se il trasmettitore indicato e` attivo B comincera` a ricevere subito le informazioni, altrimenti non ricevera` nulla fino a quando A non avra` iniziato a trasmettere.

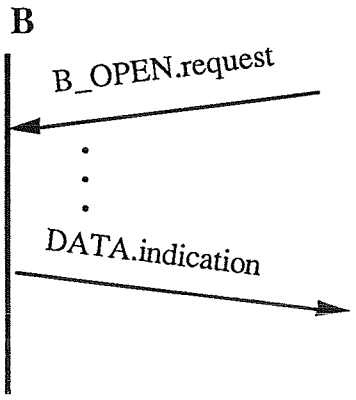

Figura 2.8 - Apertura di connessione passiva di un ricevitore per trasmissioni broadcast.

In modo completamente indipendente dal ricevitore, l'utente A (trasmettitore) quando vuole iniziare la trasmissione inoltra la B_CONNECT.request indicando solamente i parametri QoS e CoT. Se l'entita` di trasporto A e` nelle condizioni di garantire la Qualita` del Servizio richiesta la connessione viene aperta e il trasmettitore viene informato mediante un apposito segnale di "OK" (lasciato all'implementazione) che puo` iniziare a trasmettere (figura 2.9a). Altrimenti la stessa entita` di trasporto invia all'utente una DISCONNECT.indication (figura 2.9b).

Come si vede la fase di apertura di connessione nel caso di trasmissioni broadcast riguarda semplicemente il trasmettitore e la sua entita` di trasporto.
Il trasferimento delle informazioni in una connessione del tipo trasmettitore broadcast-ricevitore e` unidirezionale percio` l'utente A inoltrera` le DATA.request e l'utente B ricevera` le DATA.indication (parte superiore della figura 2.4).

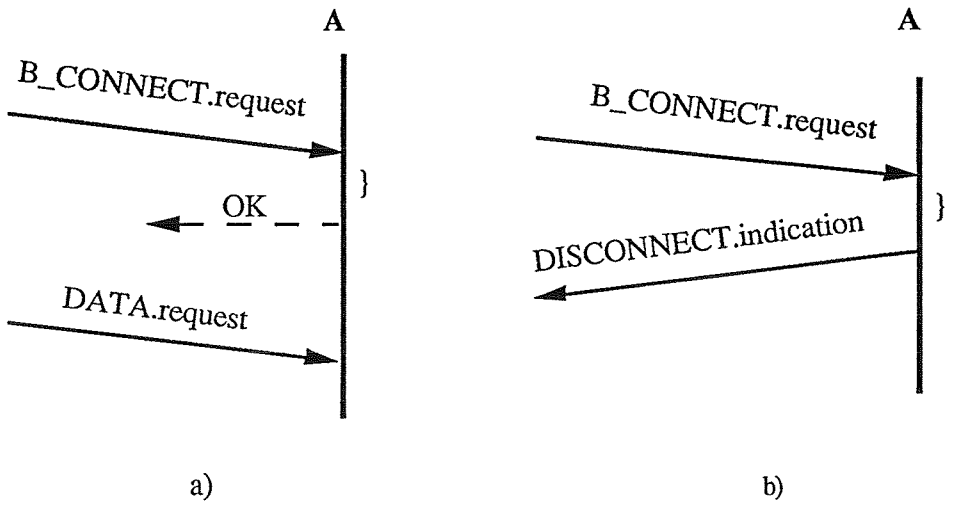

Figura 2.9 - Richiesta di apertura di connessione per trasmissione broadcast con esito positivo (a) e negativo (b).

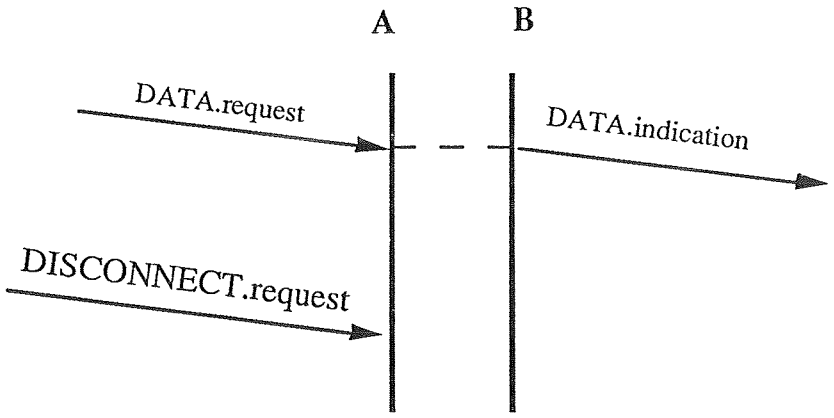

Figura 2.10 - Chusura di connessione del tipo trasmettitore broadcast-ricevitore.

Come l'apertura anche la chiusura della connessione riguarda solo il trasmettitore e puo` essere effettuata mediante la DISCONNECT.request (figura 2.10). Il ricevitore si accorge della chiusura della connessione solo dal fatto che non riceve piu` informazioni. Se pero` l'utente B ad un certo punto non vuole piu` ricevere deve inoltarre la DISCONNECT.request. Cio`, tuttavia serve solo ad informare l'entita` di trasporto la quale puo` cosi`, fra le altre cose, liberare le risorse.
Anche in questo tipo di connessioni e` possibile la chiusura forzata a causa di problemi interni alla rete (figura 2.7).

3. Macchina a Stati Finiti del Protocollo

3.1 Generalita`

Il *protocollo* di un livello generico, oltre a definire il formato delle *PDUs* (*Protocol Data Units*) scambiate fra le due entita` del livello considerato, descrive il modo in cui ciascuna entita` reagisce a determinati eventi. Questi possono essere: una richiesta di servizio inoltrata dall'utente, la ricezione di una PDU proveniente dall'entita` di pari livello, l'azzeramento di un timer, ecc. Le azioni previste dal protocollo in conseguenza di un dato evento dipendono, oltre che dall'evento stesso, dalla storia passata dell'entita` e dal valore attuale di alcune variabili locali all'entita`.
Un metodo per la descrizione formale di un protocollo e` quello della rappresentazione della sua *Macchina a Stati Finiti* o *FSM* (Finite States Machine). In generale, una FSM comprende i seguenti quattro elementi: *Stati dell'entita`*, *Predicati*, *Eventi*, *Azioni*.
Ciascuno Stato dell'entita` memorizza la storia passata dell'entita`. I Predicati sono le variabili locali all'entita`. Quando si verifica un evento il valore dei predicati insieme allo Stato attuale determina lo Stato successivo dell'entita`.
Un Evento puo` essere interno o esterno. Sono eventi esterni la ricezione di una richiesta di servizio da parte del'utente e la ricezione di una PDU dall'entita` di pari livello. Gli eventi interni

possono essere di diversa natura. Un evento interno potrebbe essere, per esempio, l'azzeramento di un timer.
Le Azioni rappresentano invece la risposta dell'entita` a un determinato Evento.
Nel prossimo paragrafo sara` analizzato il formato delle TPDU previsto dal protocollo TPR mentre i paragrafi successivi saranno dedicati alla descrizione della Macchina a Stati Finiti dello stesso protocollo.

3.2 Formato delle TPDUs

Il protocollo TPR prevede quattro tipi diversi di TPDU. Le CR (Connect Request) TPDUs vengono inviate dall'entita` di trasporto alla sua controparte quando l'utente ha inoltrato una richiesta di apertura di connessione. Le DR (Disconnect Request) servono invece a notificare alla controparte la volonta` di chiudere la connessione. Il trasferimento delle informazioni avviene mediante le DT (Data) TPDU mentre le CT (Control) TPDU vengono usate dall'entita` di trasporto per inviare informazioni di controllo alla controparte.

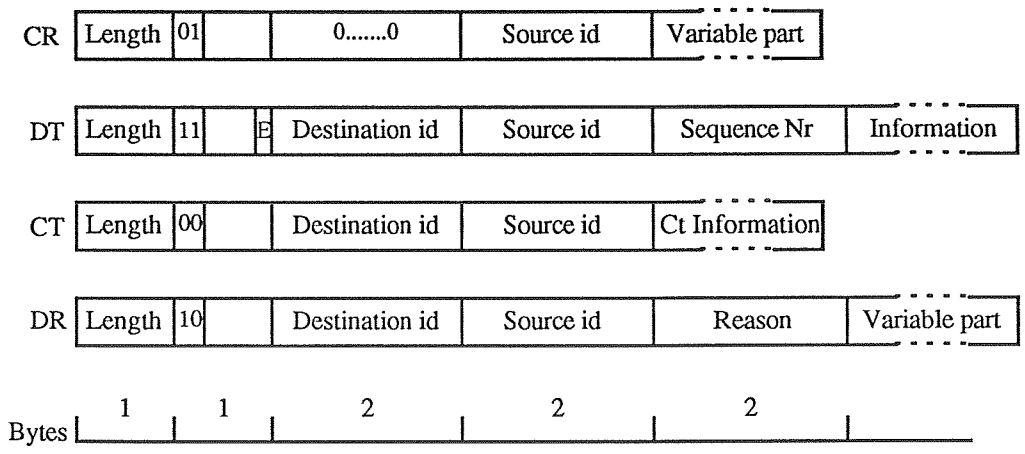

Figura 3.1 - Formato delle TPDUs

Con l'ausilio della figura 3.1 analizziamo il formato delle varie TPDUs. Poiche` le TPDUs possono avere dimensione variabile e` necessario conoscere, volta per volta, l'esatto numero di bytes contenuto in una TPDU. A cio` provvede il campo *Length.* presente in tutte le TPDUs, di qualunque tipo esse siano. I due bit dopo il campo Length specificano il tipo della TPDU. Il bit *E* nelle TPDUs di tipo DT, quando settato, indica che la presente TPDU contiene l'ultimo segmento di una SDU. I campi *Destination identifier* e *Source identifier* servono a identificare la connessione di trasporto[2]. L'entita` di trasporto che richiede l'apertura della connessione rende noto il suo identificatore riempiendo il campo *Source id* della CR TPDU. L'entita` richiesta fornisce invece il suo identificatore mediante il campo *Destination identifier* della prima DT TPDU. Il campo *Variable part* di una CR TPDU puo` contenere una qualsiasi delle seguenti opzioni:

(1) indirizzo, presso l'host remoto, dell'utente con cui si vuole stabilire la connessione.

(2) indirizzo, presso l'host locale, dell'utente che richiede l'apertura della connessione.

(3) dimensione massima proposta per le TPDUs.

(4) parametri della Qualita` di Servizio.

(5) parametri delle Caratteristiche di Traffico.

(6) altre opzioni introdotte in fase di realizzazione del protocollo.

Il campo *Sequence number* nelle DT TPDUs contiene il numero di sequenza della TPDU necessario per rilevare la perdita di qualche TPDU. Le informazioni di utente sono contenute nel campo

[2]Tali identificatori sono necessari perche` spesso piu` connessioni di trasporto vengono multiplexate sulla connessione di rete.

Information mentre il campo *Ct Information* delle CT TPDUs contiene informazioni di controllo.
Le DR TPDUs vengono inviate dall'entita` di trasporto per comunicare alla sua controparte la decisione di chiudere la connessione. Il campo *reason* della TPDU contiene le ragioni di tale decisione (DISCONNECT.request inoltrata dall'utente, comportamento illegale dell'utente rilevato dal meccanismo di controllo sulla velocita` di trasmissione, ecc). La *Variable part* e` stata introdotta per contenere eventuali opzioni non previste in fase di definizione del protocollo.

3.3 Rappresentazione della Macchina a Stati Finiti del TPR

La FSM del TPR e` rappresentata nella figura 3.2 mediante un grafico in cui compaiono delle linee verticali e una serie di freccie: le prime rappresentano gli stati attraverso cui puo` passare l'entita` di trasporto; le seconde indicano, invece, le transizioni da uno stato all'altro. Tali transizioni avvengono in corrispondenza di determinati eventi. Gli eventi sono riportati nello stesso grafico insieme all'azione che essi determinano e ad un identificatore (che serve ad individuare la transizione) secondo il seguente schema:

$$\frac{\text{<Identificatore> <Evento>}}{\text{<Azione>}}$$

L'identificatore e` formato da un numero di due cifre seguito, in alcuni casi, da una lettera. La prima cifra indica, convenzionalmente, lo stato nel quale si trova l'entita` di trasporto quando si verifica l'evento, la seconda cifra rappresenta il nuovo stato e la lettera, quando presente, serve a distinguere due transizioni diverse che pero` presentano gli stessi stati di partenza e di arrivo.
I successivi paragrafi contengono una descrizione a parole della Macchina a Stati Finiti del protocollo TPR. Per semplificare la comprensione sono state considerate separatamente le tre fasi caratteristiche in cui si articola una connessione.

3.4. Apertura di una Connessione

Inizialmente l'entita` di trasporto si trova nello stato IDLE. Quando un utente e` disponibile ad accettare richieste di apertura di connessione da parte di altri utenti inoltra la OPEN.request alla sua entita` di trasporto. Questa memorizza i parametri QoS ed eventualmente CoT indicati dall'utente, mette al valore "true" la variabile booleana locale *disponibile* (di cui esiste una copia per ciascun utente potenziale) che indica la disponibilita` dell'utente a stabilire connessioni e rimane nello stato IDLE (11a).
Se l'utente e` un ricevitore per trasmissioni broadcast che vuole ricevere da un certo trasmettitore inoltra la B_OPEN.request contenente fra i suoi parametri l'indirizzo del trasmettitore. In tal caso l'entita` di trasporto attiva il *setup timer* e passa nello stato WAIT_E (12a).

3.4.1 Apertura di Connessione richiesta dall'utente locale

Supponiamo ora che l'utente desideri aprire una connessione e pertanto inoltri una CONNECT.request con i parametri opportuni. Se l'entita` di trasporto e` in grado di garantire la QoS richiesta dall'utente allora: (1) alloca tutte le risorse necessarie; (2) modifica la variabile *disponibile* relativa all'utente, assegnandole il valore "false" (cio` e` necessario solo per quegli utenti, quali quelli telefonici, per i quali non si puo` avere piu` di una connessione attiva in un certo istante); (3) invia una CR TPDU all'entita` di trasporto destinataria; (4) fa partire il *Setup timer* (5); passa, infine, nello stato WAIT_E (12b).
Se, invece, la richiesta dell'utente non puo` essere soddisfatta l'entita` di trasporto genera una DISCONNECT.request e rimane nello stato IDLE (11b).
Se l'utente che vuole attivare la connessione e` un trasmettitore broadcast invia la richiesta di apertura di connessione mediante la

B_CONNECT.request. Se l'entita` di trasporto non e` in grado di garantire la QoS richiesta genera una DISCONNECT.request e rimane nello stato IDLE (11c). In caso contrario manda un segnale di OK all'utente (il quale e` cosi` informato che puo` iniziare la trasmissione) e passa nello stato WAIT_U (13a).
Consideriamo ora lo stato WAIT_E. In tale stato l'entita` di trasporto attende la ricezione di una TPDU dall'entita` di pari livello. Se cio` non si verifica prima che sia scattato il *Setup timer* l'entita` rilascia le risorse eventualmente allocate, genera DISCONNECT.indication e passa nello stato TERMIN (25a).
La TPDU inviata dall'entita remota puo` essere, ovviamente, di tipo CT, DT o DR.
L'arrivo di una CT TPDU informa l'entita` di trasporto che aveva inviato una CR TPDU dei progressi fatti dalla richiesta di apertura della connessione: si rimane nello stato WAIT_E (22) dopo aver informato, eventualmente, anche l'utente.
Se l'entita` di trasporto aveva inviato in precedenza una CR TPDU la ricezione di una DT TPDU indica che la richiesta di apertura di connessione e` stata accettata. Se, invece, l'entita` era pervenuta allo stato WAIT_E per effetto di una B_OPEN.request la ricezione di una DT TPDU (contenente informazioni generate dal trasmettitore richiesto) sta ad indicare che il trasmettitore e` attivo. In entrambi i casi l'entita` di trasporto puo` passare nello stato DATA_TRANFER (24) dopo aver invocato la primitiva locale DATA.mechanism. Questa trasferisce il controllo della DT TPDU appena ricevuta al primo dei meccanismi di protocollo attivi. Questi meccanismi, eccezion fatta per il meccanismo per la correzione del "drift" sono disposti a catena. Le DT TPDUs passano attraverso i meccanismi di compensazione delle informazioni perdute, di assorbimento del jitter (se attivo) per arrivare, infine, al meccanismo di depacchettizzazione (sempre attivo). Quest'ultimo provvede a rigenerare le SDUs originarie e a consegnarle all'utente destinatario mediante la generazione della primitiva DATA.indication.
Rimane da analizzare il caso in cui l'entita` di trasporto riceva una DR TPDU. Cio` si verifica se l'entita di trasporto ha inviato una richiesta di apertura di connessione che non puo` essere soddisfatta.

Pertanto essa rilascia le risorse, genera DISCONNECT.indication e passa nello stato TERMIN (25b).
Sempre nello stato WAIT_E la ricezione di una DISCONNECT.request da parte dell'utente locale provoca il rilascio delle risorse, il ritorno della variabile *disponibile* al valore "true", l'invio di una DR TPDU all'entita` di trasporto di pari livello e il ritorno nello stato IDLE (21a).

3.4.2 Apertura di Connessione richiesta dall'utente remoto

Finora abbiamo considerato il comportamento dell'entita` di trasporto quando la richiesta di apertura di una connessione sia fatta dall'utente locale. Analizziamo ora cosa succede quando, nello stato IDLE, l'entita` di trasporto riceve una CR TPDU dall'entita` di pari livello.
Anche in questo caso e` possibile che la richiesta di apertura di connessione non possa essere soddisfatta per uno dei seguenti motivi:

(1) non ci sono, al momento, risorse sufficienti per garantire la QoS richiesta

(2) *disponibile* = "false". Cio` significa che l'utente con il quale si richiede di stabilire la connessione e` attualmente occupato in altre connessioni o non ha inoltrato la OPEN.request ().

In tali casi l'entita` di trasporto provvede ad inviare una DR TPDU e rimane nello stato IDLE (11d).
Se, invece, la connessione puo` essere aperta l'entita` di trasporto alloca le risorse necessarie, invia eventualmente una CT TPDU alla sua controparte, pone *disponibile* = "false", genera la CONNECT.indication all'utente locale e passa nello stato WAIT_U (13b).
In questo stato l'entita` di trasporto attende una richiesta di servizio da parte dell'utente locale.

Se questo inoltra una primitiva di DATA.request l'entita` di trasporto passa nello stato DATA_TRANFER (34). Contemporaneamente memorizza in un buffer la SDU appena ricevuta mediante la primitiva precedente e testa la variabile booleana locale *pronto*. Se n e` il numero di SDUs che vengono spedite in un'unica DT TPDU, il predicato *pronto* viene posto al valore "true" dopo la ricezione della (n-1)esima SDU (se n=1 oppure le SDUs sono troppo grandi da dover essere segmentate *pronto* vale costantemente "true"). Pertanto se al momento della ricezione di una SDU il predicato suddetto vale "true" il meccanismo di pacchettizzazione provvede a formare la DT TPDU che viene spedita all'entita` di pari livello.
Se l'utente, al quale e` stata inoltrata la CONNECT.indication, non intende accettare la connessione, inoltra una DISCONNECT.request. Come conseguenza l'entita` di trasporto invia una DR TPDU alla sua controparte, rilascia le risorse eventualmente impegnate, pone *disponibile*="libero" e ritorna nello stato iniziale (IDLE, 31a).
L'utente potrebbe anche non rispondere affatto alla CONNECT.indication. Per ovviare a tale inconveniente nel momento in cui la primitiva viene generata viene attivato un *timer di inattivita`*. Se questo scatta senza che l'utente abbia dato alcuna risposta l'entita` di trasporto si comporta esattamente come nel caso della transizione 31a (31b).
Nello stato WAIT_U la ricezione di una DR TPDU provoca il rilascio delle risorse, la generazione di una DISCONNECT.indication e ancora il ritorno allo stato IDLE (31c). L'arrivo di una DT TPDU non implica invece nessuna azione da parte dell'entita` di trasporto che rimane anche nello stesso stato (33).

3.5 Trasferimento delle Informazioni

Allo stato DATA_TRANFER si perviene dallo stato WAIT_E attraverso la ricezione di una DT TPDU oppure dallo stato WAIT_U in seguito a una DATA.request inoltrata dall'utente locale.

Nello stato di DATA_TRANFER l'entita` rimane fino a quando riceve DATA.request o DT TPDUs (44a, 44b, 44c).
La ricezione di una DATA.request comporta il test sul predicato booleano *pronto* e, se quest vale "true", la creazione di una DT TPDU con le SDUs ricevute dal momento in cui e` stata spedita l'ultima DT TPDU e, infine, la spedizione della stessa DT TPDU(44a, 44b).
Dopo la ricezione di una DT TPDU l'entita` di trasporto invoca invece la primitiva locale DATA.mechanism con cui la stessa DT TPDU viene consegnata alla catena dei meccanismi di protocollo.

3.6 Chiusura di Connessione

La ricezione di una DR TPDU provoca il passaggio dallo stato DATA_TRANFER allo stato TERMIN (45) con conseguente rilascio delle risorse allocate e generazione di una DISCONNECT.indication. Questa viene pero` inoltrata all'utente solo dopo che tutte le informazioni arrivate fino a quel momento sono state consegnate[3].
Allo stato iniziale si ritorna, sempre dallo stato DATA_TRANFER, anche per effetto di una DISCONNECT.request inoltrata dall'utente locale (41). In questo caso l'entita` di trasporto, in primo luogo, provvede a spedire le informazioni arrivate attraverso primitive di DATA.request precedenti la richiesta di chiusura della connessione. Se tali informazioni non sono sufficienti per formare una DT TPDU di dimensioni normali vengono aggiunti dei bit di "stuffing". Successivamente l'entita` di trasporto rilascia le risorse, pone *disponibile* = "true", e invia una DR TPDU.
Dallo stato TERMIN, a seguito di una DISCONNECT.request, l'entita` di trasporto torna allo stato iniziale (51) dopo aver posto *disponibile* = "true" e dopo aver inviato una DR TPDU[4].

---

[3]Per alcune classi di utenti lo stesso evento puo` dar luogo, oltre alle azioni suddette, l'assegnamento del valore "true" al predicato *disponibile* e il passaggio allo stato IDLE anziche a quello TERMIN.

[4]Per alcune classidi utenti questa azione puo` non essere eseguita.

Sempre nello stato TERMIN la ricezione di una DR TPDU provoca la generazione di un'altra DISCONNECT.indication ma non fa cambiare lo stato (55a), mentre un'eventuale DATA.request da parte dell'utente non viene presa in considerazione (55b).
Infine una volta che si e` ritornati nello stato IDLE per effetto di una DISCONNECT.request la ricezione di un'altra DISCONNECT.request non provoca alcun effetto (11d).

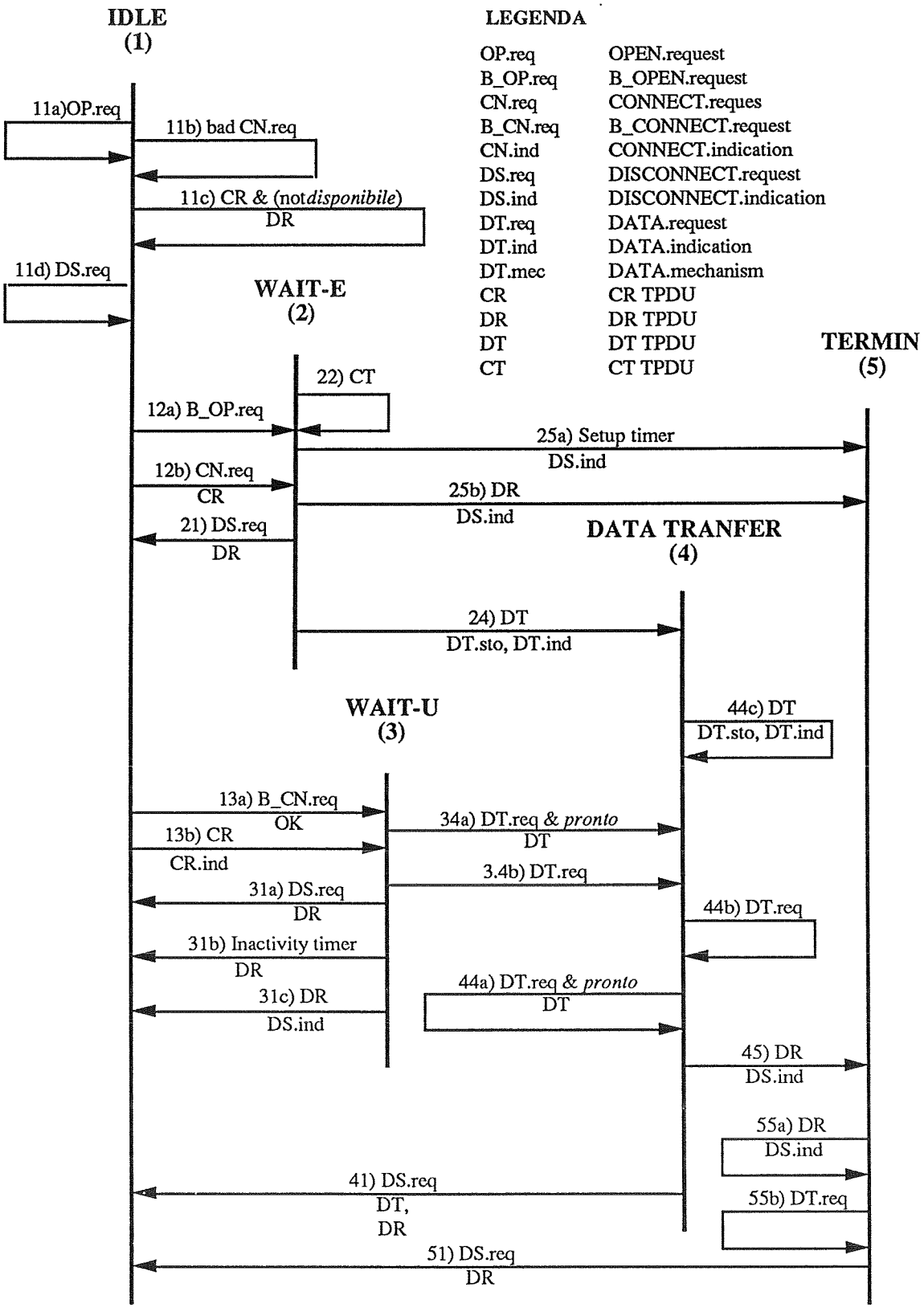

Figure 3.2 - Macchina a Stati Finiti del protocollo TPR

Bibliografia

[ANAS90] Anastasi G., Conti M., Gregori E. - "TPR: a Transport Protocol for Real Time Services in an FDDI Environment" - Proceeding of the Second IFIP WG6.1/WG6.4 International Workshop on Protocols for High Speed Networks, Novembre 1990.

[ANAS91] Anastasi G., Conti M., Gregori E. - "A Transport Protocol for Real Time Services on a Packet Switching Network" - Proceeding of EFOC/LAN '91, Londra 21-23 Giugno 1991.

[CHEN89] Chen T. M., Walrand J., Messerschmitt D. G. - "Dynamic Priority Protocols for Packet Voice" - IEEE JSAC vol 7, n. 5, Giugno 1989, pp. 632-643.

[CONT91] Conti M., Gregori E., Lenzini L. - "Design and Analysis of a Medical Communication System based on FDDI" - Comunicazione privata.

[COCH85] Cochennec J. Y., Adam P. and Houdoin T. - "Asynchronous Time Division Networks: Terminal Synchronization for Video and Sound Signals" - Proc. IEEE GLOBECOM 1985, pp. 25.6.1-25.6.4.

[DEPR87] De Prycker M., Ryckebush M., Barri P. - "Terminal Synchronization in Asynchronous Network" - Proc. ICC '87, Seattle, Giugno 1987, pp. 22.7.1-22.7.8.

[DEVA88] Devault M., Cochennec J. Y. and Servel M.- "The "Prelude" Experiment: Assessments and Future

Prospects" - IEEE JSAC vol 6, n. 9, Dicembre1988, pp. 1528-1536.

[FERR90] Ferrari D. - "Client Requirements for Real Time Communication Services" - IEEE Communication Magazine, Novembre 1990, pp. 65-72.

[KIM88] Kim C. K., Lee S.H., Wu L. T. - "Circuit Emulation" - International Journal of Digital and Analog Cabled Systems, vol. 1, 1988, pp. 245-256.

[MONT83] Montgomery W. - "Techniques for Packet Voice Synchronization" - IEEE JSAC, vol SAC-1, n. 6, December 1983, pp. 1022-1028.

[VAKI89] Vakil F. - "On Source Timing Recovery for Circuit Emulation in ATM Networks" - Proc. IEEE GLOBECOM 1989, pp. 50.4.1-50.4.8.