Capitolo 3:

Problematiche delle reti a pacchetto

In questo capitolo vengono presentate alcune problematiche generali delle reti a pacchetto. Gli argomenti presentati sono di rilievo e propedeutici anche per lo studio delle reti a commutazione di circuito, come la rete telefonica, in quanto la maggior parte dei sistemi di segnalazione e controllo per queste reti fa uso di tecniche a pacchetto.

Alcune delle problematiche presentate sono in letteratura, di solito, associate ad un particolare livello OSI. Qui l'associazione con i livelli OSI verrà fatta solo quando necessario in quanto molte aspetti sono comuni a più livelli.

1. Trame

Nella trasmissione dati, i bit che vengono trasmessi al livello fisico dal livello di linea sono organizzati in gruppi logici, detti trame, disgiunti ed identificati tramite headers. Il termine trame è usato per ragioni storiche e non va associato direttamente allo stesso termine usato nella multiplazione TDM. In realtà, anche per le trame TDM si pone il problema, analizzato nel seguito, della definizione dei metodi per la delimitazione delle trame all'interno del flusso di bit.

La necessità di dividere il flusso di dati in trame nasce da diverse ragioni che comprendono il bisogno di effettuare un controllo sulla correttezza dei bit trasmessi mediante l'aggiunta di un campo di controllo (checksum) alla fine di ogni trama.

La trama è di solito costituita dalla SDU da trasmettere con l'aggiunta di campi di servizio tipici del protocollo di linea, quali gli indirizzi di sorgente e destinazione, contatori per la numerazione delle trame e, come detto, un campo di controllo errore.

Per ragioni storiche occorre fare una differenza circa le procedure per la gestione e la delimitazione delle trame tra due tipi di protocolli di linea, quelli **orientati al carattere** (ormai in disuso) e quelli **orientati al bit**. Nei primi, l'informazione numerica è interpretata a gruppi di 8 bit secondo i caratteri di un alfabeto come l'alfabeto internazionale N.5 (ASCII). Questo, come altri alfabeti, è stato proposto proprio per funzioni di trasmissione di caratteri da vecchi terminali e, infatti, alcuni caratteri speciali sono usati per funzionalità di controllo dal livello di linea.

Tra le funzionalità di controllo che fanno uso dei caratteri speciali c'è quella di delimitazione delle trame che impiega tali caratteri per indicare l'inizio e la fine della trame. Nel protocollo a carattere BSC (Binary Synchronous Communication) dell'IBM, il carattere SYN è usato per i sincronismi di inizio trama, il carattere SOH indica l'inizio dell'header, mentre i caratteri STX ed ETX indicano l'inizio e la fine del testo. In Figura 1 è mostrata una trama del protocollo BSC.

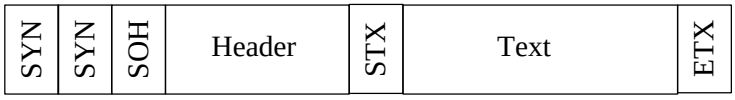

Figura 1: trama del protocollo BSC

I problemi con i protocolli a carattere nascono quando l'informazione da trasmettere è una sequenza di bit e non una sequenza di caratteri come per i vecchi terminali. Trasmettere informazione binaria con l'alfabeto

ASCII sarebbe, ovviamente, troppo dispendioso (un carattere per un bit). Conviene allora dividere la stringa di bit in sottogruppi (es. ottetti) e trasmettere i caratteri che vi corrispondono. In questo modo si aumenta l'efficienza, ma occorre risolvere il problema dovuto al fatto che, all'interno del testo, gruppi di bit possono codificare dei caratteri di controllo come quelli per la fine della trama provocando errori nel funzionamento del protocollo (di dice che il protocollo non è trasparente all'informazione trasportata).

Questi possibili malfunzionamenti possono essere evitati facendo precedere ai caratteri di controllo del protocollo il carattere DLE (Data Link Escape). Solo la sequenza DLE-carattere di controllo attiva il protocollo, mentre i gruppi singoli di bit che nel testo codificano accidentalmente un carattere di controllo vengono ignorati e trasferiti in modo trasparente. Naturalmente, i bit del testo possono anche accidentalmente codificare il carattere DLE. Per evitare problemi, in questo caso, si adotta la tecnica del *character stuffing* che consiste nella duplicazione del carattere DLE nel testo quando questo si presenta (Figura 2). In ricezione, se una coppia di caratteri DLE consecutivi viene rivelata, il primo viene soppresso.

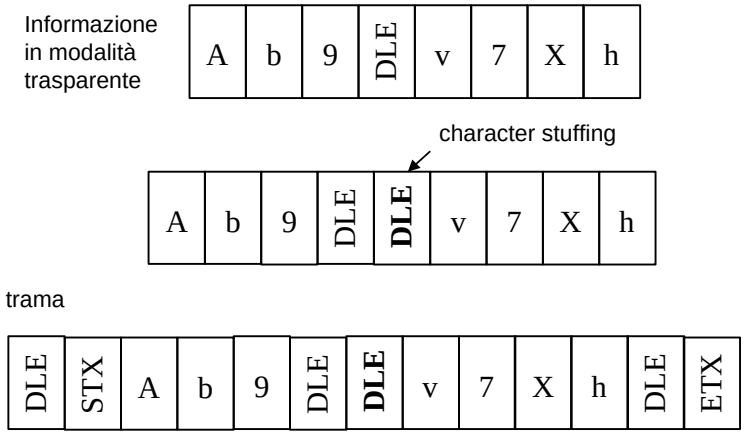

Figura 2: procedura di character stuffing

I protocolli più recenti fanno uso di trasmissione orientata al bit e non al carattere, ovvero in cui sono i bit ad avere significato diretto. Nei protocolli orientati al bit diversi metodi possono essere usati per delimitare le trame.

Un metodo è quello dell'uso di sequenze di bit particolari dette *flag* per delimitare l'inizio e la fine della trama. Esempio di tale approccio è costituito dal protocollo di linea HDLC che usa la sequenza 01111110 come sequenza di flag. Naturalmente, anche in questo caso, è possibile che la sequenza di flag si trovi accidentalmente nell'informazione trasferita. In questo caso viene usata la tecnica di *bit stuffing* che dopo 5 uno consecutivi nell'informazione aggiunge uno zero. In ricezione, dopo aver individuato i flag di inizio e fine, si depura l'informazione dai bit aggiunti eliminando il bit (sempre zero) successivo a 5 uno consecutivi (Figura 3). Naturalmente, errori in ricezione possono comunque provocare errori del protocollo che si possono propagare per più trame fino a che una coppia di flag di inizio-fine non è individuata in modo corretto.

Un altro metodo per la delimitazione delle trame si basa su una funzionalità offerta dal livello fisico, ovvero sulla violazione della codifica di linea. La codifica di linea è il metodo con il quale i bit sono identificati sul canale. Se, ad esempio, i simboli 0 e 1 corrispondono a transizioni di livello all'interno dell'intervallo di simbolo sul canale, è possibile delimitare le trame violando la regola che vuole una transizione per tempo di simbolo e trasmettendo un segnale senza transizioni. E' questo il metodo usato nelle reti locali Ethernet a 10 Mbit/s.

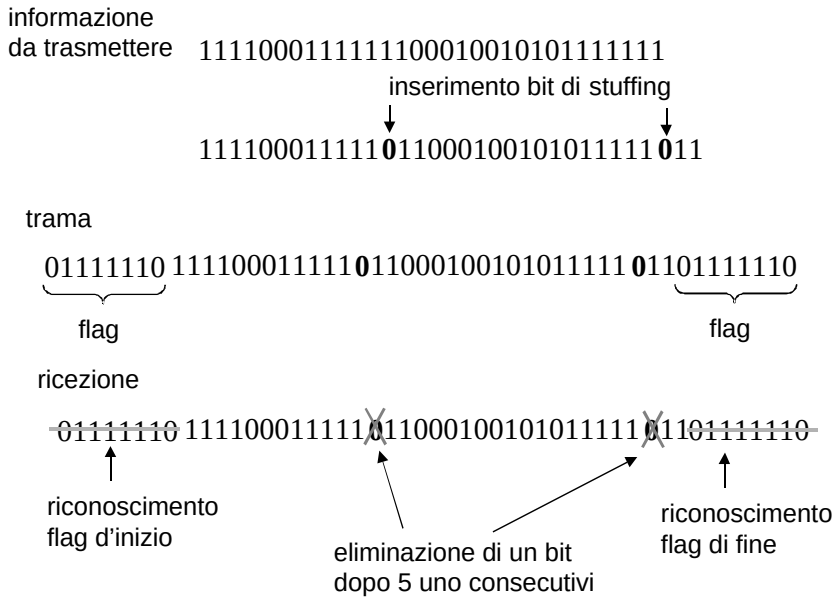

Figura 3: procedura di bit stuffing

2. Controllo d'errore

La trasmissione di informazione su un sistema di trasmissione non può prescindere dalla presenza di possibili disturbi che provocano, in ricezione, una errata decodifica dei bit ricevuti o una perdita di interi segmenti di informazione. A livello fisico i disturbi possono essere dovuti a molti fenomeni: rumore termico introdotto dai mezzi trasmissivi e dai dispositivi di trasmissione e ricezione (in molti sistemi è la fonte più importante di errori), interferenza generata da altre trasmissioni simili che usano lo stesso mezzo (come nel caso dei sistemi radiomobili), disturbi elettromagnetici di varia natura, perdita di sincronismo di bit, ecc.

Gli strati sopra il fisico, a cominciare dal livello di linea, possono offrire un servizio di comunicazione ai livelli superiori di tipo affidabile, ovvero con pochi errori, anche nel caso in cui il livello fisico sia, invece, caratterizzato da molti errori. In tal caso sono adottate opportune procedure di controllo d'errore che possono correggere in tutto o in parte gli errori introdotti dal sistema trasmissivo. Eventuali errori residui possono essere corretti da livelli superiori fino ad arrivare al livello di qualità richiesto dallo strato applicativo.

In base all'applicazione possono essere richiesti servizi di comunicazione con una tasso di errore diverso. Ad esempio, i servizi come la trasmissione della voce tollerano di solito tassi d'errore anche dell'ordine di 10^{-2}-10^{-3}, mentre servizi dati come il file transfer richiedono tassi d'errore molto più bassi.

Nei livelli superiori a quello di linea la perdita di informazione in ricezione può essere provocata non solo dagli errori introdotti dal sistema di trasmissione, ma anche alla perdita di unità informative dovuta al riempimento di qualche buffer nei nodi di rete. Infatti, se all'arrivo di un pacchetto la coda è piena il pacchetto non può che essere scartato (*overflow*). Le procedure di controllo d'errore implementate nei protocolli di trasporto sono di solito dedicate al recupero dei pacchetti persi nella rete per overflow.

I meccanismi per il controllo d'errore possono essere distinti in due classi: meccanismi **di correzione automatica degli errori** e **procedure di ritrasmissione**.

Nel primo ai bit di informazione da trasmettere vengono aggiunti degli altri bit detti di ridondanza. In ricezione, se alcuni bit risultano errati, è possibile usare i bit di ridondanza per ricostruire correttamente l'informazione se il numero di errori non è troppo grande. Le procedure per l'aggiunta della ridondanza e la ricostruzione dell'informazione sono oggetto della teoria dei codici, il cui studio è al di fuori degli scopi di questo testo. A titolo esemplificativo, però, si può considerare il codice di correzione più semplice costituito dal codice a ripetizione. All'interno dell'informazione trasmessa a ciascun bit di informazione viene

aggiunta della ridondanza costituita dalla ripetizione del bit per un numero N di volte. In ricezione per ricostruire con successo l'informazione basta che gli errori su ogni bit siamo meno di (N/2 – 1).

Anche nei meccanismi di controllo d'errore basati su ritrasmissione viene usata la teoria dei codici per aggiungere della ridondanza ai pacchetti, ma, in questo caso, non si fa ricorso alla capacità di un codice di correggere degli errori, ma semplicemente alla sua capacità di rivelarli. La ridondanza richiesta per rivelare un numero anche elevato di errori è molto più contenuta di quella richiesta per la correzione, tanto che un aggiunta di 16-32 bit ai pacchetti è, si solito, sufficiente a consentire di accorgersi se un pacchetto è corretto o meno. Il più semplice codice in grado di rivelare un errore singolo in un pacchetto è costituito dal bit di parità: alla fine del pacchetto viene aggiunto un bit pari a 1 se il numero di 1 nel pacchetto è dispari e pari a 0 se il numero di 1 è pari. Se nella trasmissione un bit risulta errato, il numero di 1 nel pacchetto risulterà dispari e potrà essere riscontrato l'errore. Anche se i codici di rivelazione sono molto più complessi del semplice bit di parità, di solito i bit di ridondanza aggiunti per la rivelazione in fondo ad un pacchetto vengono detti bit di parità o FCS (Frame Check Sequence).

In ricezione, se un pacchetto viene identificato come errato è possibile richiedere la sua ritrasmissione. La richiesta di ritrasmissione è un'informazione di servizio che deve viaggiare su un canale in direzione opposta a quella di trasmissione. Quindi, le tecniche di controllo d'errore basate su ritrasmissione, dette ARQ (Automatic Repeat reQuest), non sono utilizzabili con collegamenti monodirezionali, ma richiedono collegamenti che consentano lo scambio di informazione in entrambe le direzioni (half o full duplex).

Anche nel caso di perdita in rete di un pacchetto, può essere usata la ritrasmissione mediante meccanismi implementati a livelli superiori a quello di linea. Ovviamente, in questo caso non è un codice a rilevare che un pacchetto è errato, ma la rilevazione del pacchetto mancante avviene in modo logico, ad esempio mediante la numerazione dei pacchetti.

Affinché i meccanismi di ritrasmissione funzionino correttamente è necessario che le due entità coinvolte nello scambio di informazioni si scambino anche informazione di segnalazione. Le procedure di ritrasmissione fanno dunque parte di un protocollo di colloquio tra entità.

2.1 I protocolli di ritrasmissione

I protocolli di ritrasmissione sono regole, applicate dall'entità trasmittente e da quella ricevente, che fanno uso di segnalazione al fine di consentire il trasferimento corretto di un blocco dati, in presenza di errori di canale, mediante successive ritrasmissioni. Il blocco dati viene generalmente incapsulato in una trama che contiene campi di servizio nell'header e informazione di parità per la rivelazione d'errore FCS, normalmente posta in coda alla trama (trailer). Se il FCS rivela l'esistenza di un errore di ricezione, il ricevitore, tramite messaggi di servizio, richiede la ritrasmissione del pacchetto errato fino ad avvenuta corretta ricezione.

I messaggi di servizio comprendono messaggi detti di riscontro o ACK (da Acknowledgment), trasmessi dopo ogni ricezione corretta, i messaggi di riscontro negativo o NAK, trasmessi dopo ogni ricezione con errore, messaggi di Enquiry (ENQ) utilizzati per richiedere la ripetizione dell'ultima istruzione eseguita dal protocollo.

Il problema del riscontro delle ricezioni non è così banale come può sembrare a prima vista, perché anche i messaggi di riscontro ACK e NACK possono non essere correttamente ricevuti.

Molti protocolli di ritrasmissione sono stati sviluppati in sistemi diversi per rispondere a differenti necessità. Una classificazione molto usata li distingue, sulla base del modo con cui gli errori vengono riscontrati e le trame ritrasmesse, in protocolli di tipo: Stop and Wait, Go back N e Selective Reject.

2.1.1 Il protocollo Stop-and-Wait

E' il più semplice ed intuitivo protocollo di ritrasmissione. Il trasmettitore dopo l'invio di una trama aspetta un riscontro positivo (ACK) dal ricevitore. Se non lo riceve dopo un prefissato intervallo di tempo detto *time-out* la trama viene inviata di nuovo (Figura 4).

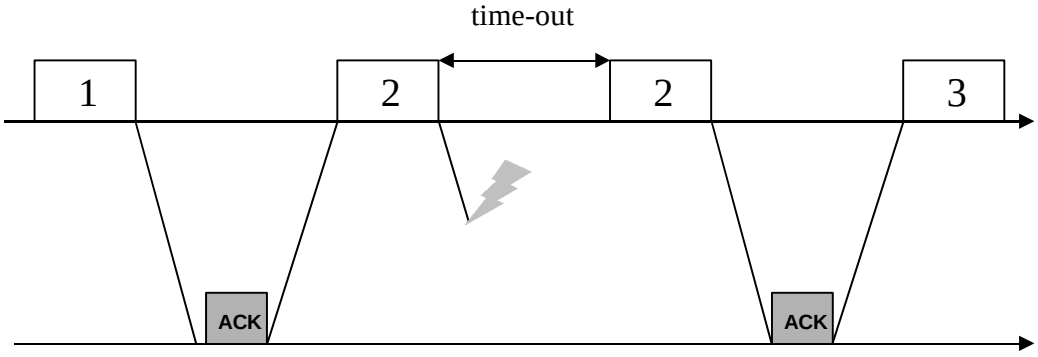

Figura 4: protocollo Stop and Wait

Come alternativa, il ricevitore può inviare un NAK (riscontro negativo) se il FCS indica una trama errata. In ogni caso però, il corretto funzionamento non può essere garantito dal solo uso del NAK perché una perdita di questo porterebbe all'impossibilità di rilevare l'errore da parte del trasmettitore.

E' facile rendersi conto che nella modalità semplice di funzionamento del protocollo sopra illustrata sono possibili malfunzionamenti che portano alla ripetizione di una stessa trama. Ciò può capitare a seguito di ritardi nell'invio dell'ACK, oppure a causa della perdita dell'ACK stesso. In questo caso, infatti, la scadenza del time out porta il trasmettitore a ripetere la trama che è già stata trasmessa correttamente. Questo comportamento non è accettabile perché, di solito, protocolli di questo tipo si occupano di garantire la consegna a livello superiore delle PDU senza errori, nell'ordine corretto e senza ripetizioni.

Per ovviare a questo problema occorre distinguere tra loro le trasmissioni delle trame mediante un opportuno identificativo inserito nell'header. Poiché però una sola trama per volta viene trasmessa dal protocollo, è sufficiente inserire una numerazione delle trame in modolo 2, ovvero costituita da un solo bit. Il campo di numerazione viene di solito indicato con SN (Sending Number) e in questo caso può assumere solo i valori 0 o 1.

Un problema più grave può nascere sempre a causa di ritardi nell'invio dell'ACK se la stazione trasmittente interpreta in maniera errata un ACK ricevuto come relativo a un'altra trama. Un comportamento di questo tipo è descritto in Figura 5.

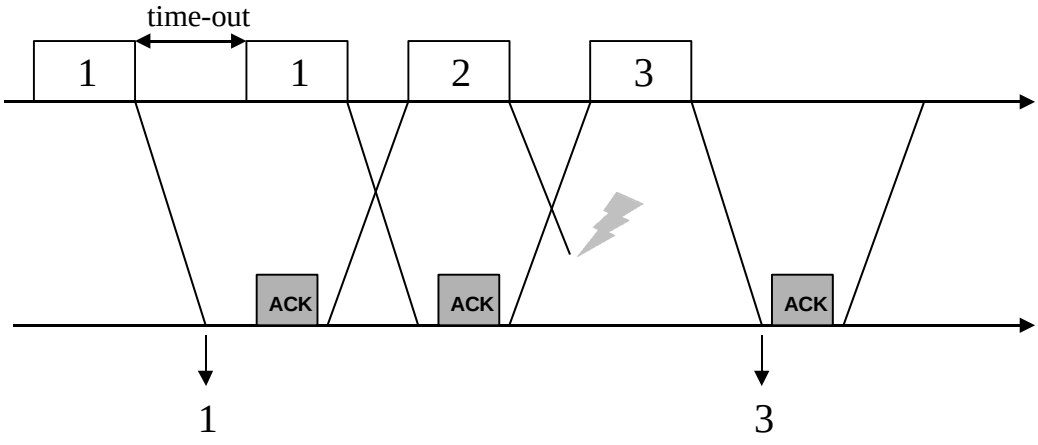

Figura 5: malfunzionamento del protocollo Stop and Wait

Per risolvere questo problema occorre, dunque, che anche il riscontro porti il numero della trama a cui si riferisce. Tuttavia è pratica comune inviare non il numero della trama ricevuta ma quello della prossima trama attesa, numero indicato con RN (Requesting Number).

Per il corretto funzionamento del trasferimento non è necessario inviare l'ACK in istanti di tempo particolari. Tuttavia, tanto più viene ritardato l'invio dell'ACK, tanto più è ritardata la successiva trasmissione, con un peggioramento del ritmo medio di trasferimento della trame del protocollo.

Nell'ipotesi che l'ACK venga inviato immediatamente dopo la ricezione corretta della trama, il trasmettitore è forzato ad attendere un tempo pari a 2τ, con τ ritardo di propagazione tra trasmettitore e ricevitore, prima di iniziare a ricevere il riscontro. Se si indica con T il tempo di trasmissione di una trama e con T_{ack} il tempo necessario per trasmettere l'ACK, l'efficienza massima raggiungibile risulta pari a:

$$h = \frac{T}{T + T_{ack} + 2t} \qquad (3.1)$$

Anche trascurando il tempo di trasmissione dell'ACK, se risulta $\tau > T$ la perdita di efficienza può essere non trascurabile. Per questo motivo, il protocollo Stop and Wait è usato per collegamenti lenti dove la condizione $\tau > T$ non si verifica. Inoltre, il suo uso è adatto a trasmissioni in modalità half duplex, dove la trasmissione di informazione vera e propria avviene sono in un senso, mentre il canale di ritorno è disponibile senza ritardi per la trasmissione degli ACK.

2.1.2 Il protocollo Go back n

Per risolvere gli inconvenienti di lunga attesa presentati dal protocollo Stop and Wait quando i tempi di propagazione sono grandi, il protocollo Go back n ammette che il trasmettitore possa trasmettere al più *n* pacchetti senza aspettare il riscontro dell'ACK. Naturalmente *n* deve essere finito e la sua scelta influenza le prestazioni del protocollo. Se il riscontro non arriva, il trasmettitore ritrasmette tutte le trame dall'ultima non riscontrata, da cui il nome "torna indietro di *n* trame".

Poiché questa variante rende i ritardi di riscontro meno critici, si ha la possibilità di funzionamento *full duplex*, ossia con informazione trasmessa nei due versi e con i due protocolli di ritrasmissione che operano indipendentemente. In questo, caso i messaggi di servizio vengono accodati insieme ai pacchetti dati nel senso di trasmissione desiderato (Figura 6).

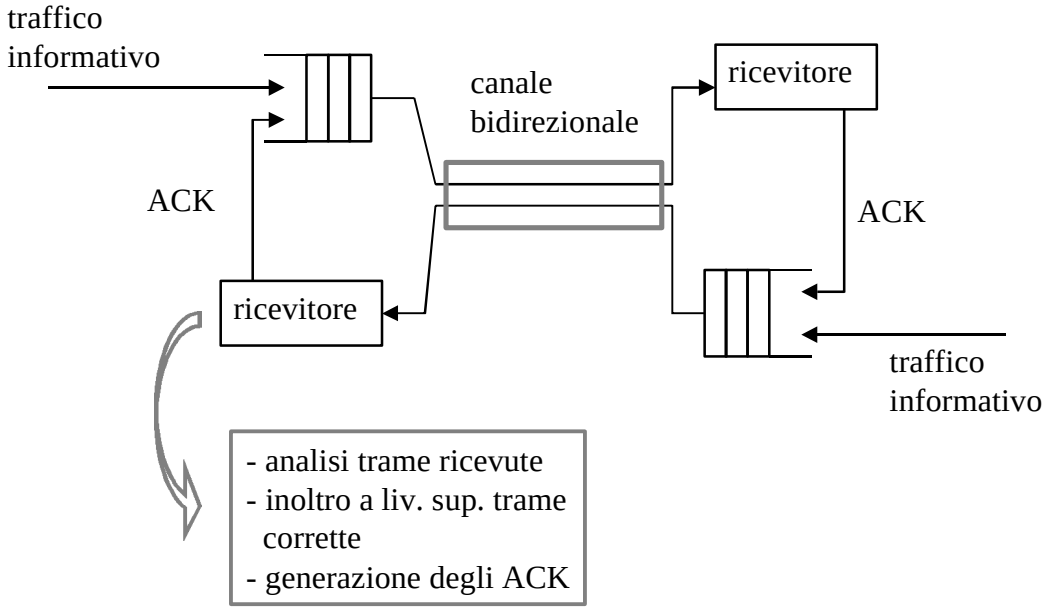

Figura 6: funzionamento full-duplex del protocollo go back n

In alternativa, si accetta la possibilità di trasmettere gli ACK inserendoli nell'header di pacchetti che vengono trasmessi nel senso opposto: questa tecnica è nota col nome di *piggy-backing*. Questo approccio impone che l'informazione di riscontro sia trasmessa solo insieme alla trame informative in direzione opposta, e ciò provoca una variabilità fra l'istante in cui si genera il riscontro e l'istante in cui viene trasmesso, perché le trame informative di ritorno possono non essere immediatamente disponibili a causa della dinamica del traffico informativo.

Il protocollo opera numerando i pacchetti da trasmettere con un campo Sending Number (SN) collocato nell'header, e con un campo Request Number (RN), sempre collocato nell'header, che indica il numero della trama che il ricevitore si aspetta di ricevere e che, quindi, funziona come ACK per tutti i pacchetti con indentificativo più piccolo (riscontro ricezione corretta pacchetti in direzione opposta con SN<RN).

Il trasmettitore, in ogni istante, mantiene memorizzato l'ultimo riscontro ricevuto memorizzando il relativo RN in N_{last} e conserva un numero identificativo corrente N_c disponibile per essere assegnato alla prossima trama in trasmissione. Le trame vengono trasmesse secondo le seguenti regole, che vengono seguite indipendentemente dall'ordine e senza vincoli di tempo:

- ✓ ogni nuova trama viene accettata per la trasmissione solo se $N_c < N_{last} + n$, altrimenti viene messa in attesa; se $N_c < N_{last} + n$ la trama viene trasmessa con SN pari a N_c e il valore di N_c viene incrementato di uno ($N_c := N_c+1$);
- ✓ ad ogni riscontro RN ricevuto, si pone $N_{last} = RN$;
- ✓ i pacchetti nella finestra vengono trasmessi senza vincoli di temporizzazione; alla fine la ritrasmissione deve ripartire dalla trama N_{last}.

Il ricevitore conserva l'identificativo della prossima trama attesa in RN e opera nel seguente modo:

- ✓ se viene ricevuta correttamente una trama con SN=RN, questa viene inoltrata ai livelli superiori e si pone RN:=RN+1;
- ✓ ad istanti arbitrari ma con ritardo finito, RN viene trasmesso al mittente utilizzando le trame in direzione opposta.

L'intervallo tra N_{last} e $(N_{last}+n-1)$ definisce una finestra di trasmissione pari a n trame che limita il numero di trame inviabili consecutivamente dal trasmettitore. Se le trasmissioni nei due sensi avvengono senza errori e se *n* è sufficientemente grande, il limite superiore della finestra non si raggiunge mai anche se sono sempre disponibili nuovi pacchetti. Un esempio di funzionamento in queste condizioni è mostrato in Figura 7, dove si è usato *n*=6 e si è indicata tra parentesi la finestra di trasmissione (N_{last}; N_{last}+n-1). Si può osservare nell'esempio in figura che le trame dal nodo B al nodo A sono più rare, ma il ritmo di invio dei riscontri consente lo stesso la trasmissione in direzione A-B senza interruzioni, ovvero senza che il trasmettitore debba interrompersi per rispettare il vincolo imposto dalla finestra di trasmissione (N_{last}; N_{last}+n-1).

Nel caso in cui si presenta errore il trasmettitore giunge fino alla fine della finestra di trasmissione e poi ricomincia a ritrasmettere dalla trama N_{last}. Un esempio di funzionamento del protocollo con errore è mostrato in Figura 8 dove si è usato n=4. Con riferimento all'esempio si può osservare che, dopo l'errore, il nodo B continua a ricevere trama corrette ma che non corrispondono alla trama attesa; queste trame non vengono accettate, in accordo alle regole descritte precedentemente, ma il valore di RN contenuto può essere usato dal protocollo per aggiornare la finestra di ricezione. Questo funzionamento obbliga il ricevitore a scartare trame corrette, ma consente di consegnare le trame in ordine crescente al livello superiore senza necessità di effettuare un riordino.

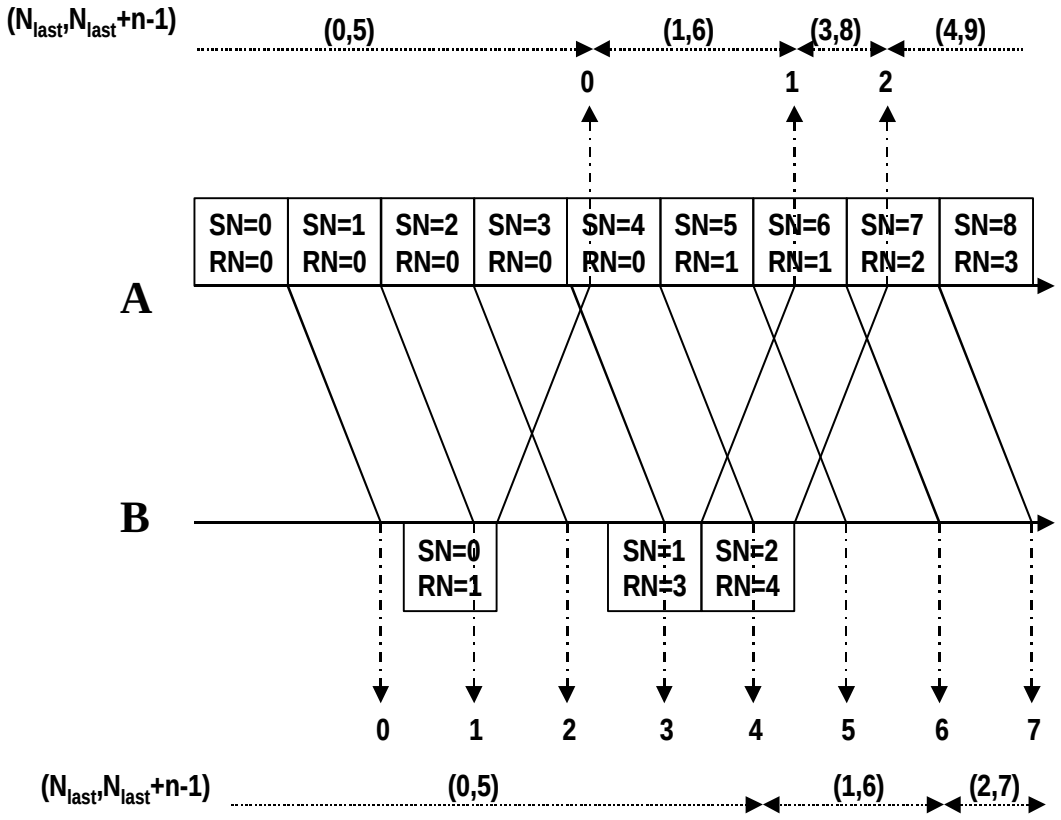

Figura 7: funzionamento senza errori del protocollo go back n

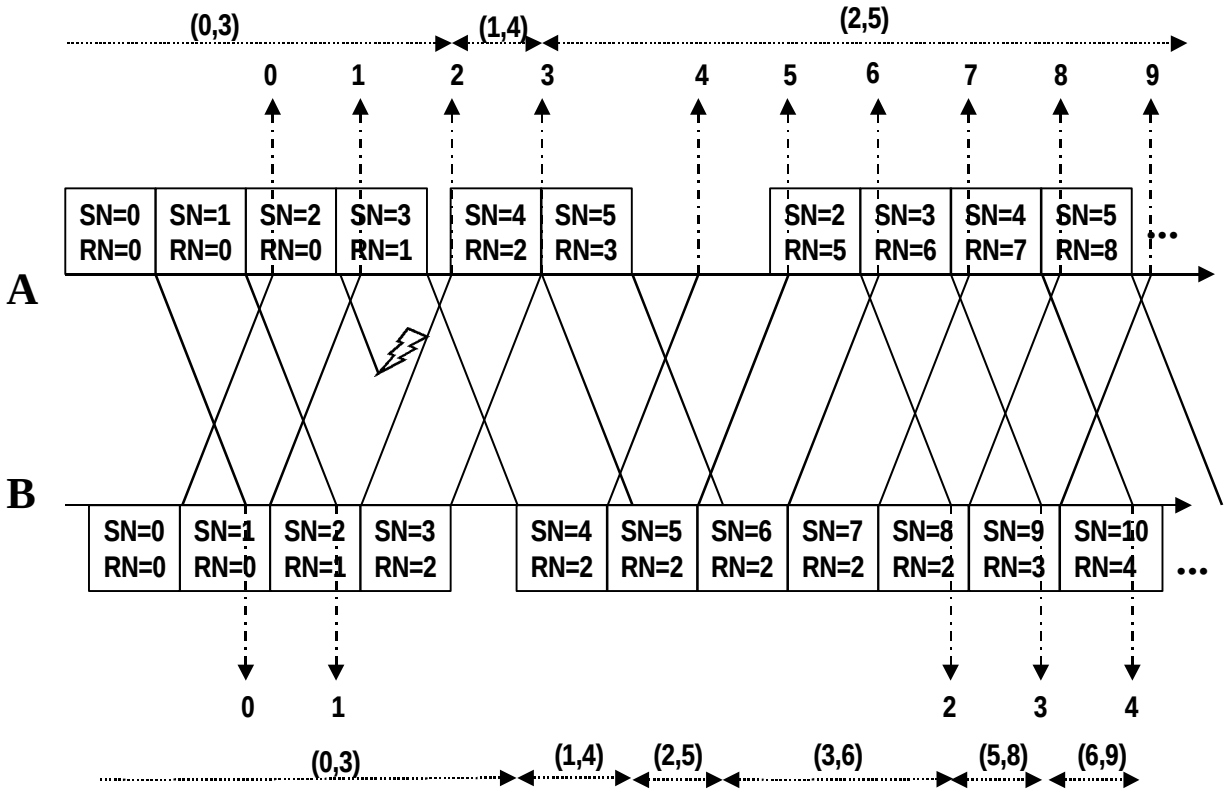

Figura 8: funzionamento del protocollo go back n con errore

Nel funzionamento del protocollo le ritrasmissioni possono essere dovute non solo ad errori, ma anche a ritardi nel riscontro delle trame dovuti alla dinamica del traffico in direzione opposta. Questo comportamento non è a rigore corretto, ma è una conseguenza del funzionamento full duplex e dell'uso del piggyback. In Figura 9 è mostrato un esempio di questo comportamento anomalo. Si può osservare che, anche se i ritardi nel riscontro provocano l'inizio della ritrasmissione, dopo l'arrivo di una trama con un riscontro valido il trasmettitore salta subito alla trama attesa indicata in RN.

Un funzionamento anomalo come quello descritto può essere causato non solo da ritardi ma anche da errori in direzione opposta. In Figura 10 è mostrato un esempio nel quale errori in un verso provocano ritardi nell'arrivo degli ACK e dunque ritrasmissioni anche in verso contrario.

Entrambe i comportamenti anomali descritti sono dovuti ai ritardi subiti dagli ACK sul canale di ritorno. Se i ritardi sono dovuti ad errori (Figura 10), ovviamente non è possibile porvi rimedio e il comportamento del protocollo, in un certo senso deve essere considerato normale. Se, invece, i ritardi sono dovuti alla dinamica del traffico è possibile limitarne l'effetto. In effetti i casi di fermo e ritrasmissione indicati nella Figura 9 si eliminano scegliendo un *n* sufficientemente grande. Un *n* troppo grande, però, peggiora la situazione perché aumenta il numero ritrasmissioni da effettuare dopo ogni errore.

Come rimedio radicale si può disaccoppiare l'invio dei riscontri dalla dinamica del traffico consentendo al protocollo di inviare delle trame senza contenuto informativo che hanno il solo scopo di far arrivare in tempo il riscontro al ricevitore. In Figura 11 si è ripreso l'esempio di Figura 9 e si è modificato il comportamento del protocollo consentendo l'invio di trame corte contenti solo il numero RN valido. Si osserva come il comportamento anomalo scompaia.

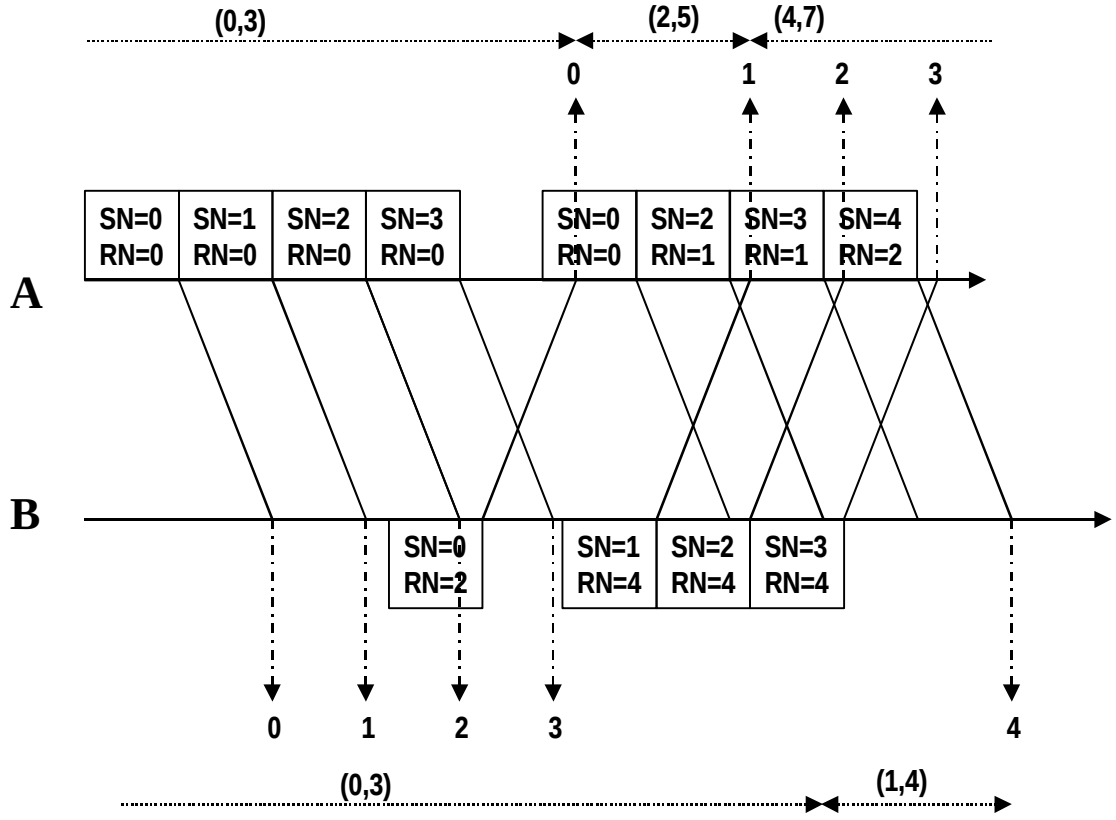

Figura 9: funzionamento anomalo del protocollo go back n

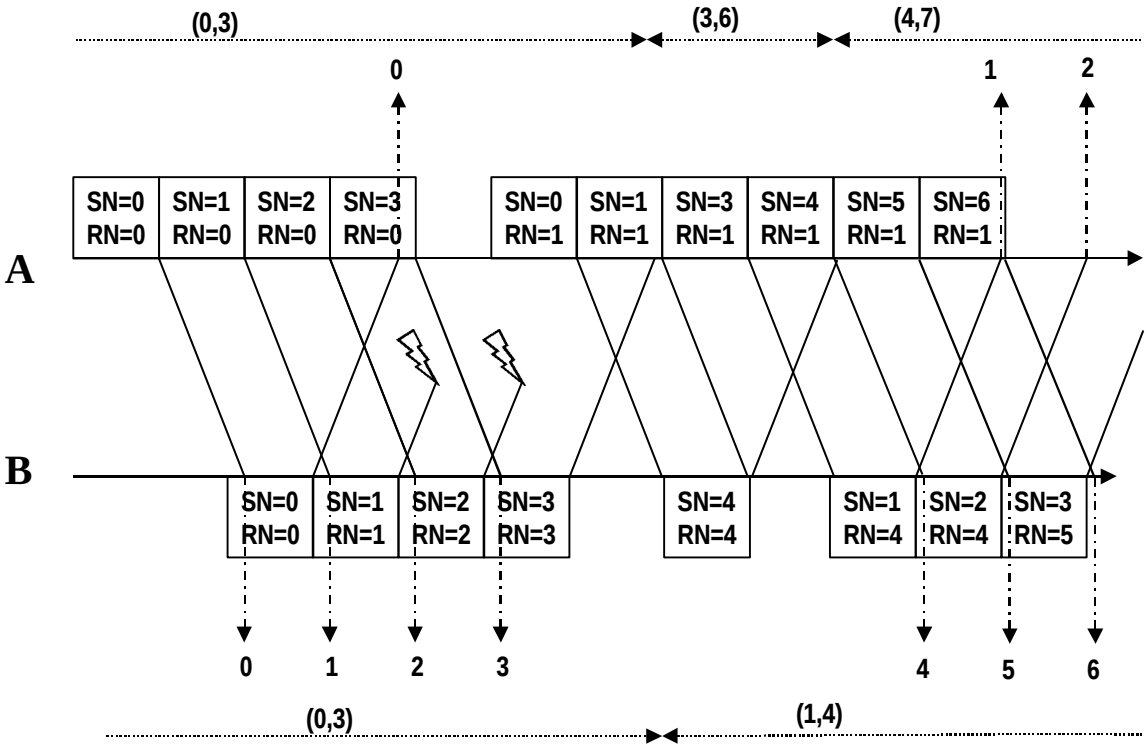

Figura 10: funzionamento anomalo del protocollo go back n con errori

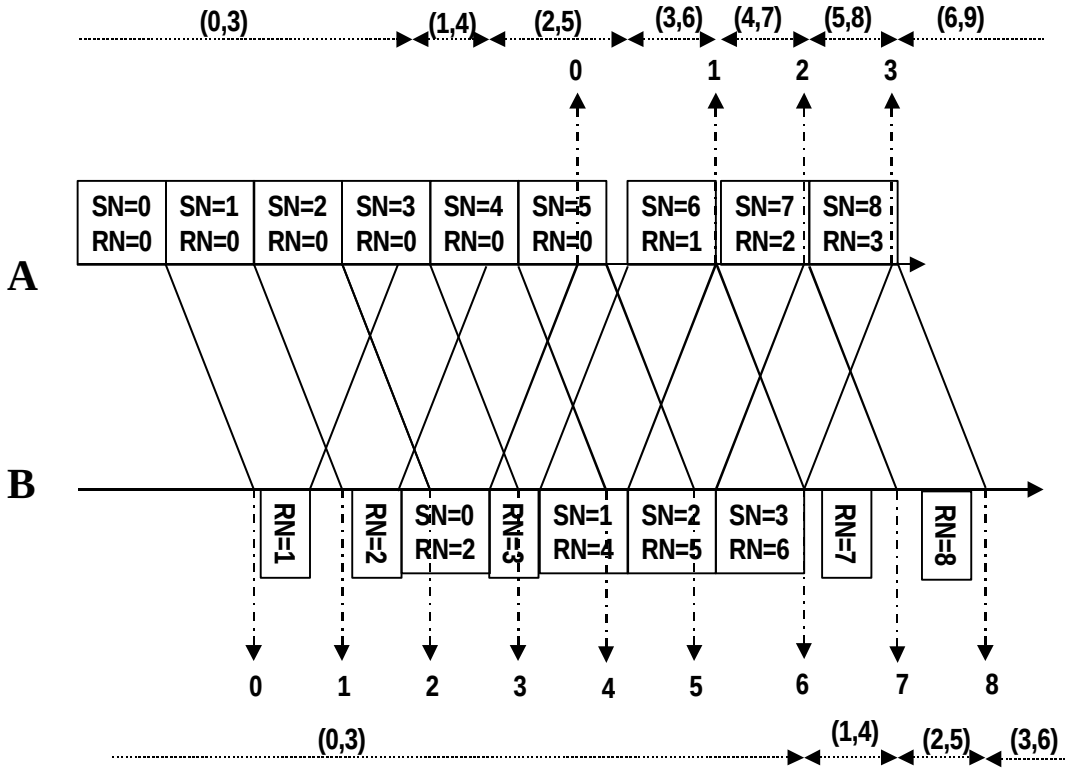

Figura 11: uso di trame di solo riscontro nel protocollo go back n

2.1.3 Il protocollo Selective Reject

Il protocollo precedente non può evitare che l'errore in un pacchetto provochi la ritrasmissione dei pacchetti successivi anche nel caso in cui questi siano in realtà stati ricevuti correttamente. In alternativa il ricevitore può tenere per buoni tutti i pacchetti correttamente ricevuti e richiedere la ritrasmissione dei soli pacchetti errati. Un protocollo che opera in questo modo è detto Selective Reject SR in quanto vengono scartati solo i pacchetti realmente errati.

Naturalmente, perché un protocollo di ritrasmissione funzioni come SR sono necessarie una serie di funzionalità aggiuntive. In primo luogo è necessaria una funzione di riordino che memorizzi le trame ricevute fuori sequenza e le riordini all'arrivo delle trame mancanti prima di procedere all'inoltro al livello superiore. In pratica, le possibilità di memorizzazione e riordino saranno limitate e quindi se gli errori sul canale si presentano in un periodo in modo molto irregolare non è possibile un comportamento reale rigorosamente di tipo SR. Come si vedrà nel seguito, questi protocolli sono sempre associati a meccanismi di controllo di flusso che impediscono di avere un numero troppo elevato di trame inviate dal trasmettitore senza alcun riscontro di ricezione, e quindi il numero di trame su cui agire per il riordino risulta limitato.

Il secondo problema relativo al SR è quello dell'invio di informazione sufficiente dal ricevitore al trasmettitore affinché questo possa correttamente capire quali trame occorre ritrasmettere e quali no. Un possibile approccio consiste nell'invio di ACK singoli, ovvero validi per un solo pacchetto. In questo caso il trasmettitore ritrasmette solo i pacchetti per i quali non ha ricevuto riscontro entro un tempo limite dalla trasmissione (time-out).

In alternativa, il ricevitore può usare ancora degli ACK cumulativi, ovvero per i quali il numero RN indica la corretta ricezione delle trame con SN inferiore a RN, ma in aggiunta può indicare quali pacchetti successivi non sono stati ricevuti correttamente mediante un campo di *bitmap* i cui bit servono da ACK/NACK per i pacchetti successivi a quello indicato in RN.

Appare evidente che la quantità di informazione di servizio richiesta per il funzionamento del SR è superiore a quella richiesta dagli altri protocolli. Questo, insieme alla maggiore complessità di implementazione fanno si che il SR sia un protocollo poco usato. Il suo utilizzo è limitato ad alcune reti con accesso radio in cui la particolare rumorosità del mezzo trasmissivo porta ad usare una tecnica altamente efficiente con il SR anche a costo di una elevata complessità.

2.1.4 Osservazioni sul controllo d'errore

Per concludere lo studio del controllo d'errore si aggiungono qui alcune considerazioni generali.

In generale e per tutti i protocolli, l'invio di un NACK richiede al ricevitore di riconoscere i pacchetti errati. Nel caso in cui il protocollo di ritrasmissione sia implementato a livello di linea, è spesso possibile riconoscere un pacchetto errato mediante il campo di frame check sequence FCS. Nel caso in cui, invece, il protocollo di ritrasmissione è implementato a livelli superiori, il protocollo deve poter far fronte a perdite in rete di pacchetti. Tali pacchetti non giungono mai a destinazione e non possono essere direttamente riconosciuti come "errati" (tipico esempio è quello del protocollo di trasporto TCP usato in Internet). In questo caso, l'invio del NACK può essere deciso in seguito all'arrivo di pacchetti corretti con numero di sequenza più elevato, ma anche in questo caso, se la rete è di tipo datagram è possibile che il pacchetto mancante sia semplicemente in ritardo perché sta percorrendo un cammino differente dagli altri. Lo studio di questi aspetti del controllo d'errore è rimandato all'analisi dell'esempio specifico offerto dal protocollo TCP.

La finestra di trasmissione operata dal go back n opera anche da freno al ritmo di invio delle trame. Esattamente questo è lo scopo dei meccanismi di controllo di flusso che devono evitare che nel colloquio tra un trasmettitore veloce ed uno lento quest'ultimo venga sommerso di trame che saturano velocemente i buffer di ricezione. Come si vedrà in seguito, proprio la finestra di trasmissione è un efficiente e molto usato meccanismo di controllo di flusso.

Tutti i protocolli visti sono basati su un meccanismo di numerazione delle trame. Dalla necessità di numerare sia le trame che gli ACK, e dunque dalla necessità di mettere d'accordo trasmettitore e ricevitore sull'inizializzazione della numerazione discende la necessità di uno scambio di informazione di servizio prima dell'inizio del trasferimento di informazione d'utente.

Questa inizializzazione dei protocolli può essere vista come la fase di instaurazione di una connessione ed è per questo che spesso questi protocolli sono detti a connessione. Qui di seguito vengono analizzate le procedure utilizzabili per l'apertura e la chiusura di una connessione.

2.2 Procedure per connessioni

A livello di linea, le procedure di connessione sono usate per attivare il canale e inizializzare i contatori SN e RN. Tali procedure sono usate dopo l'inizializzazione del livello fisico e ogni volta che il protocollo ARQ debba essere reinizializzato in seguito alla caduta della connessione fisica. A livelli superiori a quello di linea (tipicamente nel livello di trasporto), le procedure di connessione sono adoperate per gli stessi scopi con in più la necessità di una elevata affidabilità che consenta un corretto funzionamento anche nel caso di perdite di pacchetti o di ritardi elevati e variabili. Le procedure per l'apertura e la chiusura delle connessioni richiedono scambio di informazione di servizio tra le due entità colloquianti, quindi esse fanno parte, insieme ai messaggi usati e ai loro formati, di un vero e proprio protocollo di colloquio.

L'efficienza di questi protocolli non deve essere necessariamente alta perché essi sono di uso saltuario. Per garantire un certo grado di affidabilità, anche lo scambio dei messaggi di servizio avviene sotto controllo d'errore; il meccanismo usato è di tipo Stop and Wait per semplicità e, soprattutto, per usare una numerazione semplice.

I messaggi scambiati sono principalmente di due tipi, quello appunto di richiesta di connessione (INIT) con il relativo ACK e quello di rilascio della connessione (REL) con il relativo ACK.

Il problema principale da affrontare è quello di garantire che lo stato della connessione (UP o DOWN) venga interpretato in modo corretto dai nodi coinvolti. Ad evitare visioni dello stato della connessione diverse da parte dei due nodi coinvolti, si possono utilizzare inizializzazioni sbilanciate (*unbalanced setup*) in cui un solo nodo, detto primario, ha diritto ad instaurare e ad abbattere la connessione. Un esempio di questo genere è mostrato nella Figura 12 dove il nodo primario è il nodo A.

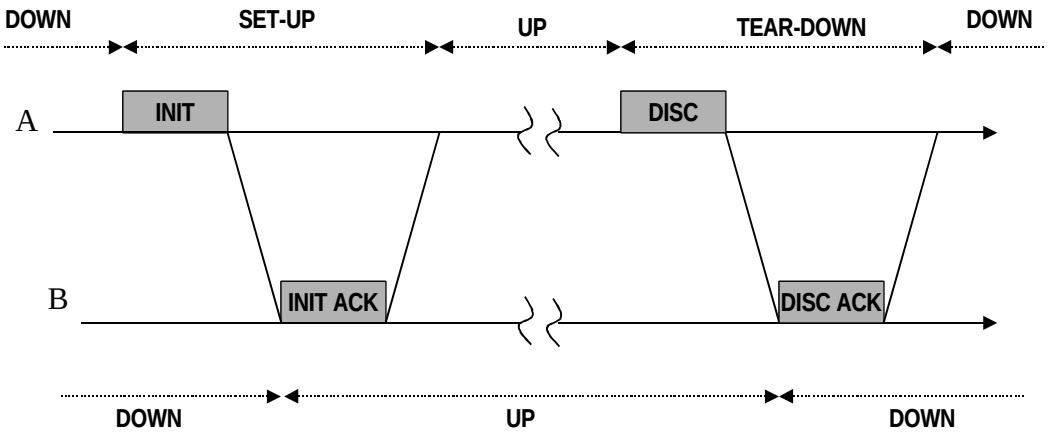

Figura 12: procedura di unbalanced setup e tear-down

Una procedura di questo tipo risulta essere un po' rigida a causa del fatto che, una volta inviato l'INIT, la procedura non può essere abortita o terminata neppure dal nodo primario prima di aver ricevuto il corrispondente ACK. Analogamente, il nodo secondario non può rifiutarsi di inizializzare la connessione, anche se, per ovviare a questo inconveniente, un ulteriore flag viene inserito nell'INIT-ACK per richiedere un immediato rilascio della connessione.

In alternativa, è possibile utilizzare un protocollo un po' più complesso che permette l'iniziativa ad entrambi i nodi. Questo protocollo, detto a struttura bilanciata (balanced setup), consiste di fatto nell'applicazione successiva da parte di tutti e due i nodi del protocollo sbilanciato, come mostrato in Figura 13. In sostanza, l'iniziatore dell'azione non aspetta solo l'ACK dell'altro nodo, ma anche un suo comando speculare a cui risponde con un suo ACK. Quando questo secondo ACK raggiunge l'altro nodo la procedura è completata. Nella figura è mostrato il caso in cui il primo ACK viene associato al comando di INIT in direzione opposta in fase di setup e al comando di DISC in direzione opposta in fase di tear-down.. Una procedura di questo genere è detta *three-way handshake*.

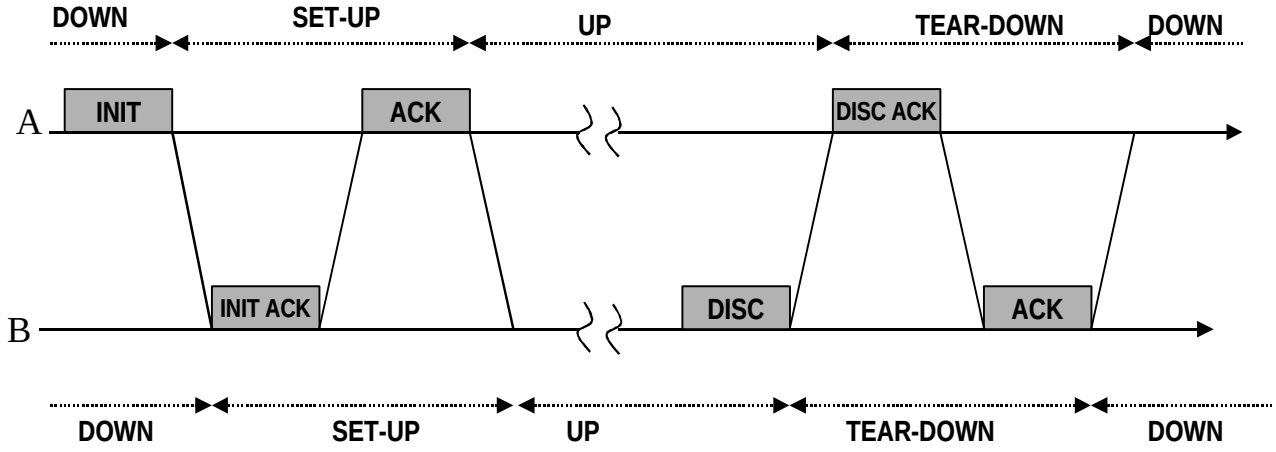

Figura 13: procedura di balanced setup e tear-down (three way handshake)

Nonostante la procedura di riscontro, possono comunque nascere problemi se uno dei nodi cade e dimentica lo stato in cui si trovava, come mostrato nella Figura 14 dove si assume che dopo ogni fallimento lo stato torni ad essere DOWN e che il formato degli ACK sia lo stesso nella fase di setup e in quella dati. Si osserva come alla fine, il pacchetto dati DATA-0 viene riscontrato positivamente, anche se non è stato mai ricevuto.

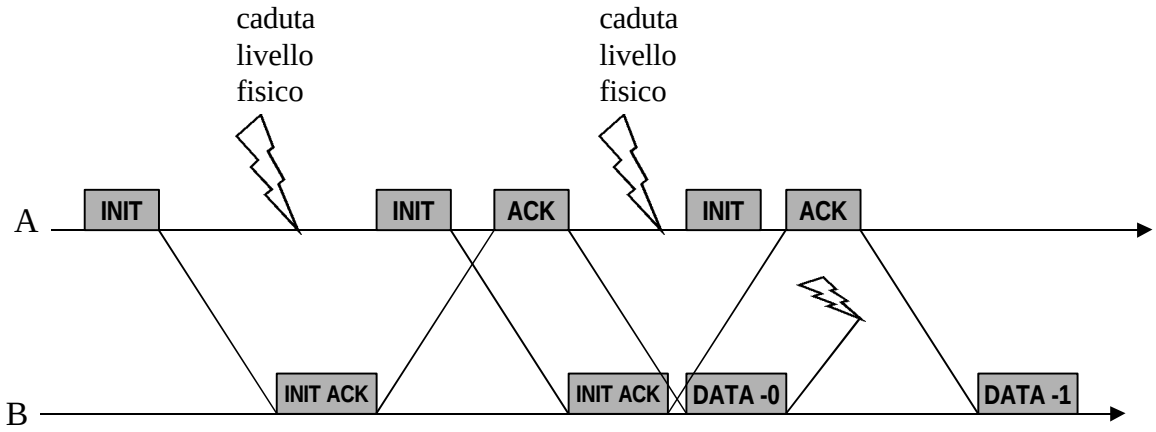

Figura 14: possibile cattivo funzionamento dovuto a cadute successivi del livello fisico

Soluzioni che evitino confusioni fra gli stati visti dai due nodi e che evitino anche possibili confusioni fra successive connessioni si possono avere introducendo dei time out stringenti se il tempo di propagazione massimo ha un limite, cosa frequente a livello di linea ma meno frequente a livello di rete o di trasporto. Un'altra soluzione è quella di identificare le sessioni di connessione con identificatori di modulo elevato e magari scelti casualmente.

3. Controllo di flusso

L'obiettivo del controllo di flusso è quello di regolare la velocità di invio delle unità informative da una sorgente ad una destinazione in modo che tale velocità non sia superiore a quella con la quale le unità informative vengono smaltite a destinazione. Il controllo di flusso può essere necessario a vari livelli della pila di protocolli, ma, di solito, viene adottano nel livello di linea (livello 2) per controllare il colloquio tra trasmettitore e ricevitore e a livello di trasporto (livello 4) per controllare il colloquio tra due applicazioni.

Per illustrare il controllo di flusso si consideri il semplice schema di Figura 15 nel quale una sorgente invia unità informative ad una destinazione mediante un canale dedicato. Alla destinazione è collegato un utente che smaltisce secondo un proprio ritmo le unità arrivate a destinazione. La destinazione è dotata di un buffer nel quale possono essere memorizzate *K* unità informative ricevute correttamente ed in attesa di essere prelevate dall'utente.

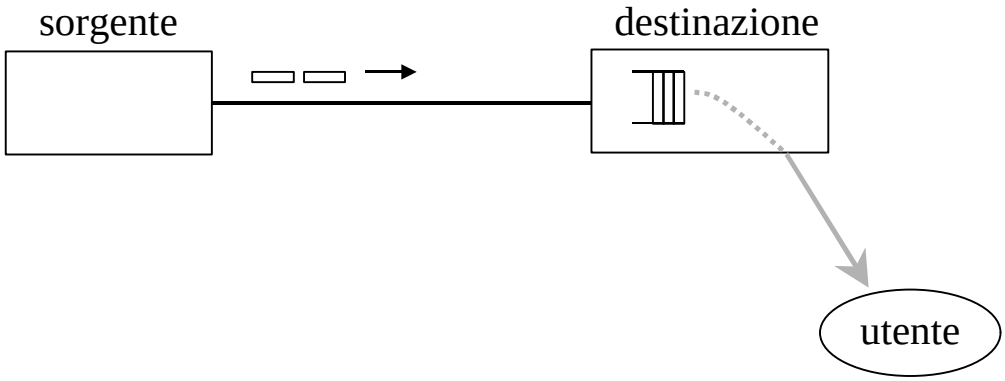

Figura 15: esempio per illustrare il controllo di flusso

In questo schema d'esempio, obiettivo del controllo di flusso è quello di evitare che il buffer di ricezione di saturi e che all'arrivo di una nuova unità informativa questa vada persa. Tale obiettivo può essere raggiunto usando un meccanismo di controllo a finestra (*sliding window flow control*) basato sullo stesso principio illustrato per il controllo d'errore go back n.

La sorgente può inviare alla destinazione un numero di unità informative pari a *W* senza attendere il riscontro (qui il parametro *W* ha la stessa funzione del parametro *n* usano nel go back n). Quando i riscontri arrivano nuove unità informative possono essere inviate dalla sorgente. Ponendo *W* uguale a *K* e facendo sì che la destinazione invii i riscontri non quando riceve correttamente una unità informativa ma solo quando questa è prelevata dall'utente, si ottiene un controllo di flusso perfetto che impedisce al buffer di ricezione di andare in saturazione e perdere pacchetti. Il funzionamento del controllo di flusso a finestra è illustrato in Figura 16 dove si è considerato il funzionamento half-duplex , si è supposto il canale privo di errori e si e posto K=W=4.

Il controllo di flusso appena descritto è strettamente legato al meccanismo di invio dei riscontri e quindi al controllo d'errore. Questo legame tra i due meccanismi può essere fonte di inconvenienti. Infatti, se la destinazione ritarda molto l'invio di un riscontro, la sorgente può erroneamente ritenere che le unità informative non siano state correttamente ricevute e iniziare la ritrasmissione. Questo comportamento può verificarsi se l'intervallo di recupero delle unità informative da parte dell'utente è più lungo del time-out di ritrasmissione adottato dalla sorgente (Figura 17). E' possibile dunque tentare di mitigare il fenomeno aumentando il time-out, ma naturalmente questo si scontra con l'esigenza opposta di tenere basso il time-out per evitare lunghe attese prima di poter rimediare ad un errore con la ritrasmissione.

Una soluzione alternativa è quella di inviare i riscontri quando le unità informative giungono a destinazione correttamente e usare, invece, un messaggio di segnalazione speciale RNR (receiver not ready) per chiedere alla sorgente di sospendere la trasmissione di nuove unità informative quando il buffer è pieno. Questo meccanismo è illustrato in Figura 18. Inoltre, è possibile mettere insieme i due approcci ritardando l'invio dei riscontri per un tempo non superiore ad un tempo massimo e usando il messaggio di RNR quando il buffer si riempie. Queste possibilità di funzionamento sono di solito adottate in molti protocolli di linea, ma

appare evidente che il principale problema è quello di avere a disposizione un solo messaggio (RNR) che è in grado di bloccare completamente la sorgente e non, invece, dei messaggi che possano servire solo a rallentare l'invio dell'informazione.

Una soluzione radicale del problema si ottiene disaccoppiando completamente il meccanismo dell'invio dei riscontri dal controllo a finestra. Ciò può essere ottenuto, ad esempio, inserendo un campo *finestra (W)* nei messaggi che viaggiano in direzione opposta. La destinazione può usare il campo *W* per indicare in modo esplicito alla sorgente lo spazio rimante nel buffer di ricezione (Figura 19). Questo approccio è quello adottato in alcuni protocolli di trasporto come ad esempio TCP (transport control protocol).

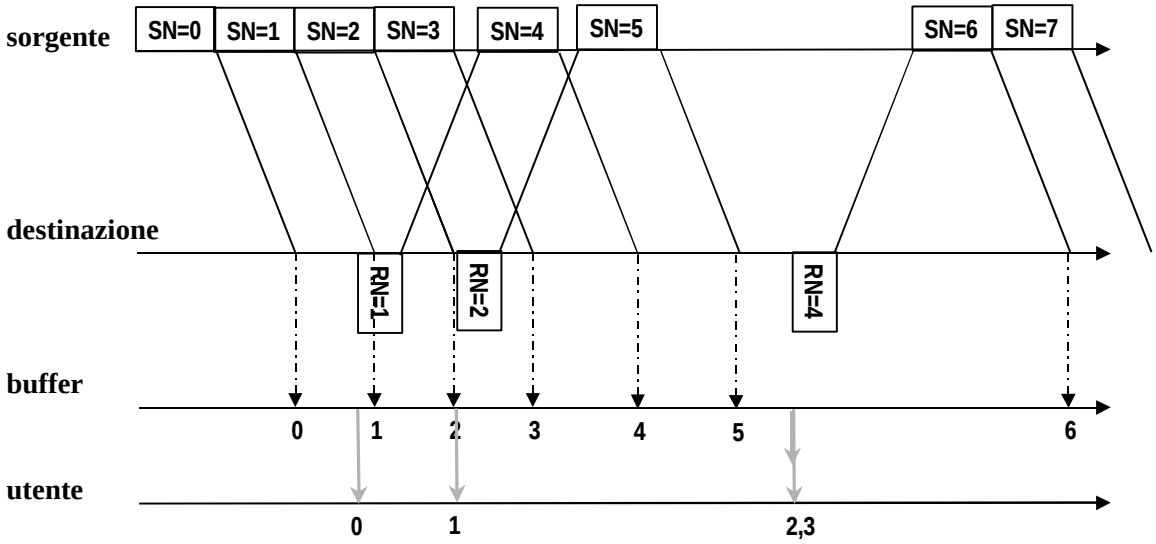

Figura 16: esempio di funzionamento del controllo di flusso a finestra

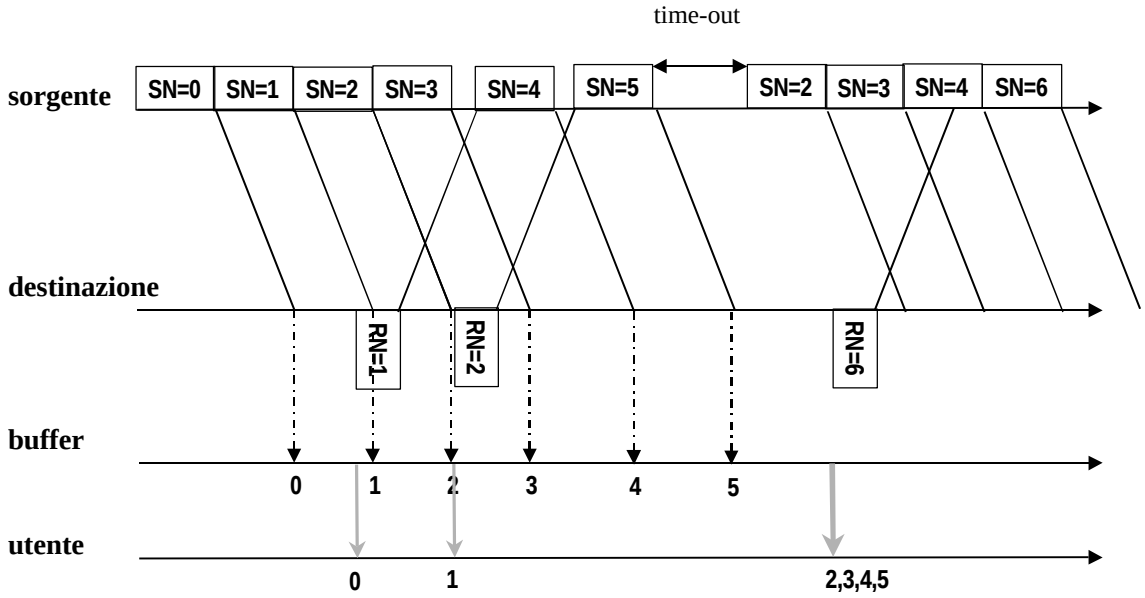

Figura 17: esempio di cattivo funzionamento del protocollo a finestra

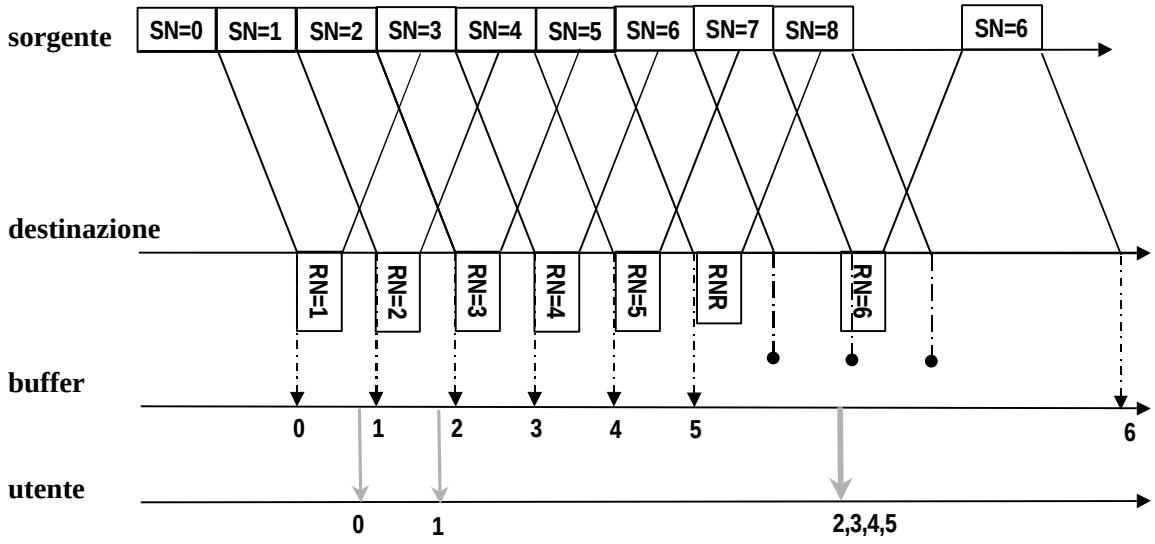

Figura 18: esempio di uso del messaggio RNR

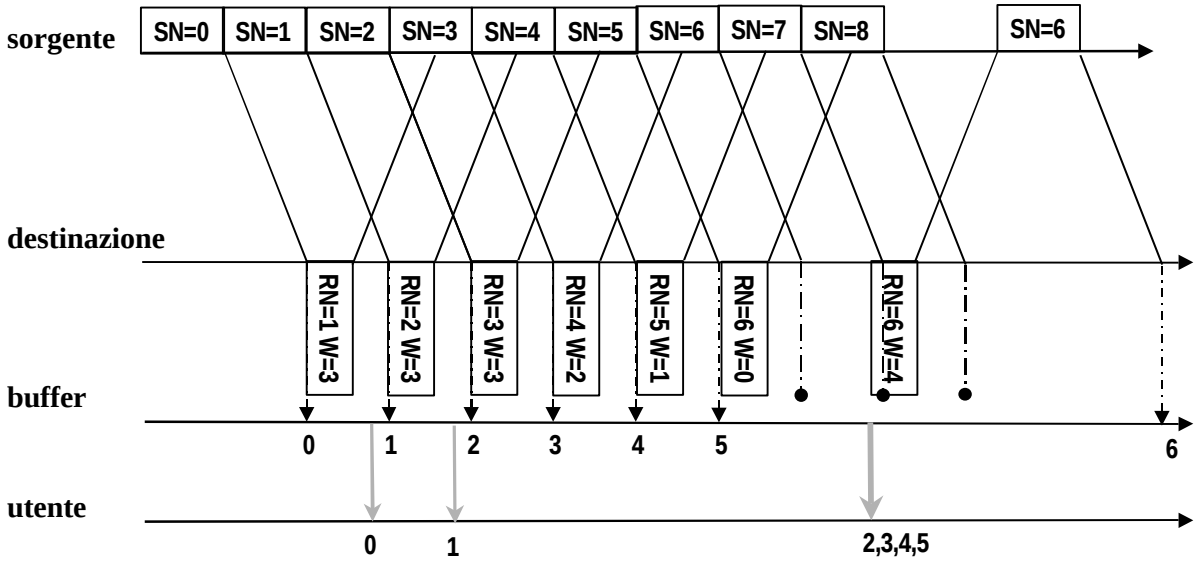

Figura 19: esempio di utilizzo del campo finestra W

4. Protocolli di linea

Come già accennato precedentemente i protocolli di linea possono essere distinti in protocolli orientati al carattere e protocolli orientati al bit. I protocolli orientati al carattere sono stati i primi ad essere introdotti, ma adesso le esigenze dei complessi sistemi di telecomunicazione attuali hanno portato ad usare i più evoluti protocolli orientati al bit. Anche questi ultimi hanno subito un'evoluzione che ha portato dai primi protocolli in grado di supportare solo il colloquio half-duplex di tipo master-slave tra una stazione primaria ed una stazione secondaria, a protocolli in grado di funzionare sia con canali punto-punto che con canali punto-multipunto e con colloquio paritario tra due stazioni.

Nei protocolli di linea usati su canali punto-punto non vi è necessità di usare alcun tipo di indirizzo nell'header della trama. Al contrario nei protocolli usati su canali punto-multipunto un campo indirizzo è sempre presente per consentire di indicare in modo esplicito le stazioni coinvolte nel colloquio. Molti protocolli sono in grado di funzionare in molti scenari differenti relativi sia a canali punto-punto sia a canali punto-multipunto, quindi il campo indirizzo risulta spesso essere presente anche quando non strettamente necessario.

In questo paragrafo verranno presentate le caratteristiche fondamentali di alcuni protocolli di linea orientati al bit. In particolare si darà una descrizione del protocollo HDLC e un cenno ad altri tipi di protocolli le cui caratteristiche derivano in maniera più o meno diretta da quelle di HDLC.

4.1.1 HDLC

Il protocollo High-level Data Link Communication (HDLC) è uno standard ISO, ma deriva dal protocollo proprietario SDLC (Synchronous Data Link Control) sviluppato da IBM per le reti SNA. Nelle intenzioni dei progettisti doveva essere un protocollo avanzato, capace di ovviare ai vincoli e agli errori di quelli esistenti. In pratica è un protocollo molto complesso, in grado di operare in diverse condizioni con moltissime varianti, ma non è privo di problemi. Da questo derivano altri usatissimi protocolli in cui si è cercato di ovviare ad alcune mancanze di HDLC.

L'HDLC può operare sia in modalità full-duplex (il colloquio tra due stazioni avviene contemporaneamente sia in un verso che nell'altro) che in modalità half-duplex (il colloquio avviene in modo alternato in un verso e nell'altro). Il colloquio può avvenire sia in modo sbilanciato, con una stazione che invia i comandi (primaria) e una che fornisce le risposte (secondaria), sia in modo bilanciato con entrambe le stazioni (stazioni combinate) che possono inviare comandi e risposte.

L'HDLC è in grado di funzionare sia su canali punto-punto che su canali punto-multipunto e per questo provvede in ogni caso all'indirizzamento delle stazioni coinvolte nel colloquio. Inoltre, è prevista la possibilità di scambiare informazioni in modalità con connessione o senza connessione. Nel primo caso è anche possibile usare un controllo di flusso a finestra e dei meccanismi di ritrasmissione che possono essere sia di tipo *go back n* che di tipo *selective reject*. Tre sono le modalità di funzionamento base:

- ✓ Normal Response Mode (NRM)

  Una stazione primaria è collegata a una o più stazioni secondarie tipicamente in modalità half-duplex. Solo la stazione primaria può inviare i comandi e le stazioni secondarie trasmettono solo a seguito di un permesso (polling) esplicito inviato dalla stazione primaria

- ✓ Asynchronous Response Mode (ARM)

  Anche in questo caso come nel NRM il colloquio è di tipo sbilanciato, ma la stazione secondaria ha la possibilità di iniziare una trasmissione senza il permesso esplicito della stazione primaria iniziando così un colloquio full-duplex. A causa della possibilità di trasmissioni contemporanee da parte di più stazioni secondarie, questa modalità, per altro poco usata, è tipica di configurazioni punto-punto.

- ✓ Asynchronous Balanced Mode (ABM)

  Fornisce una modalità di funzionamento bilanciato su configurazioni punto-punto tra stazioni combinate che possono, in modalità full-duplex, inviare informazioni in modo indipendente ed asincrono.

Il formato base delle trame di HDLC è mostrato in Figura 20.

Flag **F**	Address **A**	Control **C**	Information **Info**	Frame Check Sequence **FCS**	Flag **F**

Figura 20: struttura base della trama di HDLC

Il campo flag (**F**) all'inizio e alla fine della trama consente la delimitazione delle diverse trame ed è costituito dalla sequenza **01111110.** Per evitare inserzioni accidentali della sequenza di flag negli altri campi viene adottata la tecnica di bit stuffing già descritta in precedenza. Nelle configurazioni nelle quali una stazione deve trasmettere continuamente del segnale anche nei periodi di inattività della sorgente di informazione viene continuamente trasmessa la sequenza di flag.

Il campo indirizzo (**A**) è normalmente di 8 bit ma può anche essere costituito da più byte. Per indicare in modo univoco di quanti byte è costituito l'indirizzo, l'ultimo bit di ogni byte non è usato per l'indirizzo ma come indicatore che è posto a 0 se segue un altro byte del campo **A** o a 1 se il byte è l'ultimo del campo. L'indirizzo contenuto nel campo **A** può essere o quello della stazione destinataria della trama o in quella sorgente a secondo della modalità di funzionamento. In particolare, nelle modalità sbilanciate (NRM e ARM) contiene sempre l'indirizzo della stazione secondaria sia nelle trame trasmesse dalla stazione primaria che nelle trame trasmesse dalle stazioni secondarie, mentre nella modalità bilanciata (ABM) contiene l'indirizzo della stazione destinataria.

Il campo di controllo (**C**) indica il tipo di trama. Per i diversi tipi di trama i sottocampi del campo **C** hanno significati differenti. Sono possibili modalità d'uso che prevedono campi di controllo di 8 (modalità normale) o di 16 bit (modalità estesa). Tre tipi di trama sono definiti:

- ✓ Information (I)

  Sono trame numerate per la trasmissione di informazione d'utente contenuta nel campo **I**.

- ✓ Supervisory (S)

  Sono trame numerate per il controllo dell'invio del flusso di informazione (ad esempio riscontri non associati ad informazione in senso opposto).

- ✓ Unnumbered (U)

  Sono trame non numerate usate per l'invio di informazione di controllo (ad esempio per l'instaurazione delle connessioni) o per l'invio di informazione in modalità senza connessione.

Il campo **I** può essere o può non essere presente. È presente, ovviamente, solo quando la trama trasporta informazione d'utente (tutte le trame I e le trame U usate per l'invio di informazione in modalità senza connessione) mentre non è presente nelle trame di solo controllo. La lunghezza del campo **I** è variabile.

Il campo **FCS**, lungo 2 byte, è un campo di controllo dove è inserita la ridondanza di un codice per la rivelazione d'errore.

In Figura 21 sono mostrati i formati dei tre tipi di trama I, S ed U.

I diversi tipi di trame sono distinti sulla base dei primi bit: il primo bit delle trame **I** è 0, i primi due bit delle trame **S** sono 10 e i primi due delle trame **U** sono 11.

Il campo di controllo delle trame **I** contiene i numeri SN e RN (in alcuni testi indicati con N(S) e N(R)), che servono per il controllo d'errore di tipo go-back-n, e il bit P/F (Polling bit) il cui uso è spiegato più avanti.

Le trame **S** servono per gli ACK e NACK nel caso in cui non è possibile usare la tecnica del piggyback, e anche per un controllo di flusso basato sul comando di RNR. Il campo **type** definisce il tipo di trama S.

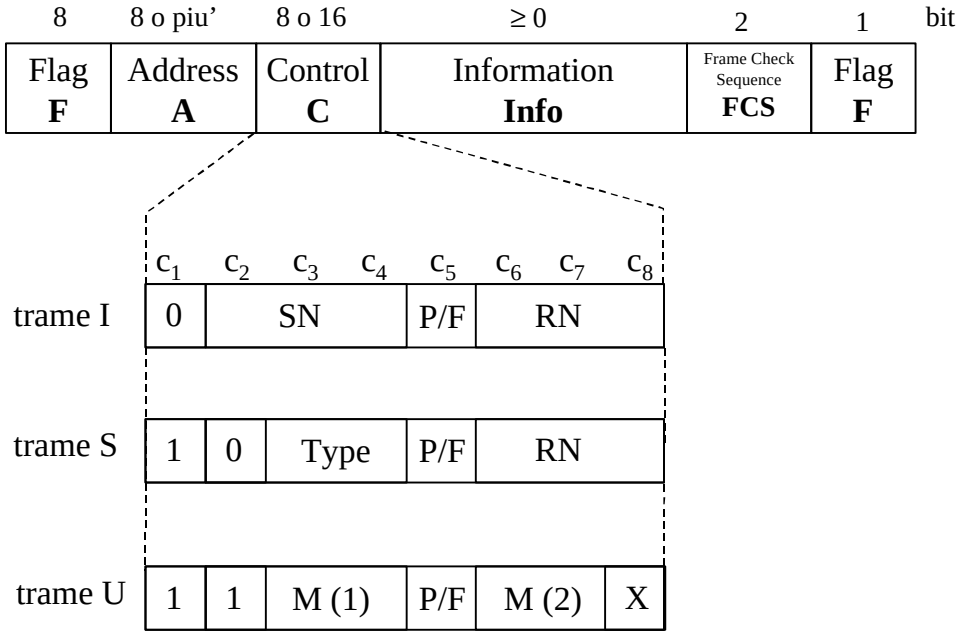

Figura 21: formato dei campi per i tre tipi di trame

Sono definite le trame S:

- ✓ RR (Receiver Ready), campo type 00, che è normalmente usato come ACK e il campo RN contiene la prossima trama attesa (riscontro delle trame fino a RN-1)
- ✓ RNR (Receiver Not Ready), campo type 10, serve a bloccare l'invio di trame da parte dell'altra stazione e, contemporaneamente a riscontare le trame fino a RN-1
- ✓ REJ (Reject), campo type 01, serve a richiedere la ritrasmissione delle trame da RN in avanti e, contemporaneamente, a riscontrare le trame fino a RN-1
- ✓ SREJ (Selective Reject), campo type 11, è usato per richiedere la ritrasmissione della sola trama con numero RN

L'uso delle trame REJ e SREJ è opzionale, ma occorre sottolineare che le trame ricevute errate (FCS errato) sono ignorate, mentre l'invio dei comandi di NACK può essere solo scatenato dall'arrivo di una trama fuori sequenza. Del resto in una trama errata l'uso del campo SN per la richiesta di una ritrasmissione non è sicuro in quanto non è possibile sapere se il campo SN stesso è o meno affetto da errore.

Le trame **U** hanno funzioni di controllo aggiuntivo e vengono usate per esempio nell'instaurazione della connessione. Esse vengono usate anche come trasmissione di informazione in modalità senza connessione e senza riscontro. Il campo M (di 4 bit divisi in due sotto-campi M(1) e M(2)) è usato per definire fino a 32 comandi (non tutti sono definiti ed usati), mentre il bit X non è definito. Un elenco dei comandi principali è mostrato in Tabella 1.

La trasmissione dell'informazione può avvenire in modalità senza connessione con trame non numerate (UI), ma molto più spesso avviene in modalità con connessione. La fase di instaurazione della connessione avviene mediante lo scambio i messaggi che consentono di definire il modo di trasferimento (SNRM, SARM, SABM). Alla fine della fase di trasferimento dati la connessione viene chiusa mediante il comando di DISC. In Figura 22 sono mostrati i casi di apertura di connessione tra una primaria e più secondarie in modalità NRM (a) e quello di apertura di connessione tra due stazioni combinate in modalità ABM (b); nella figura si è usata la simbologia (address, command, P/F bit) . Si osservi che nei comandi SNRM il flag P/F è settato con significato di *polling* per sollecitare una risposta da parte della secondaria. Nella risposta della secondaria il bit è settato con significato di *final* ad indicare che il controllo ripassa alla stazione primaria. Il bit P/F presuppone i ruoli di primaria e secondaria, ma può essere usato anche in modalità ABM quando è

necessario che le due stazioni combinate assumano temporaneamente proprio i ruoli di primaria e secondaria.

Comandi di definizione del modo	
SNRM	Set normal response mode (SNRME versione estesa)
SARM	Set asynchronous response mode (SARME versione estesa)
SABM	Set asynchronous balanced mode (SABME versione estesa)
SIM	Set inizialization modo
DISC	Disconnect
Risposte	
UA	Unnumbered acknowledgment
DM	Disconnect mode
RIM	Request inizialization mode
Trasferimento di informazione	
UI	Unnumbered information
UP	Unnumbered poll
Vari	
RSET	Reset
XID	Exchange identification

Tabella 1: alcuni comandi unnumbered

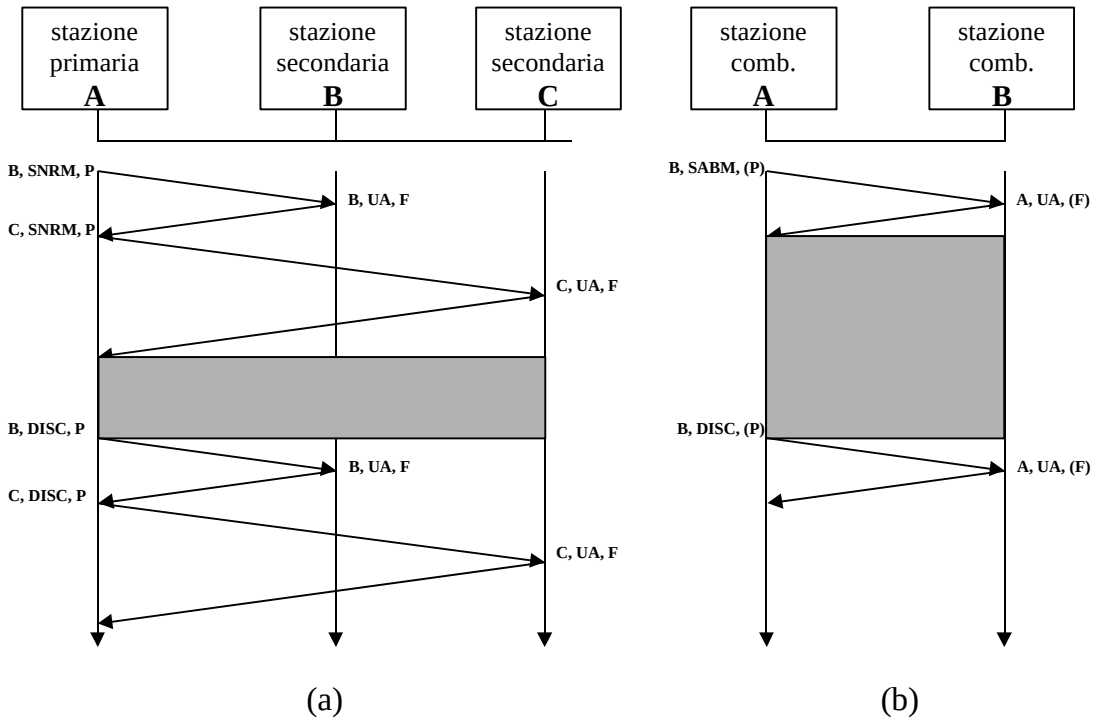

Figura 22: procedure di setup e di release della connessione in modalità NRM (a) e in modalità ABM (b)

Moltissimi sono i modi con i quali è possibile il trasferimento dell'informazione con il protocollo HDLC il cui uso dipende dall'implementazione specifica del protocollo. In Figura 23 sono mostrati alcuni esempi di funzionamento in fase dati nella modalità NRM, mentre in Figura 24 sono mostrati alcuni esempi di funzionamento in modalità ABM.

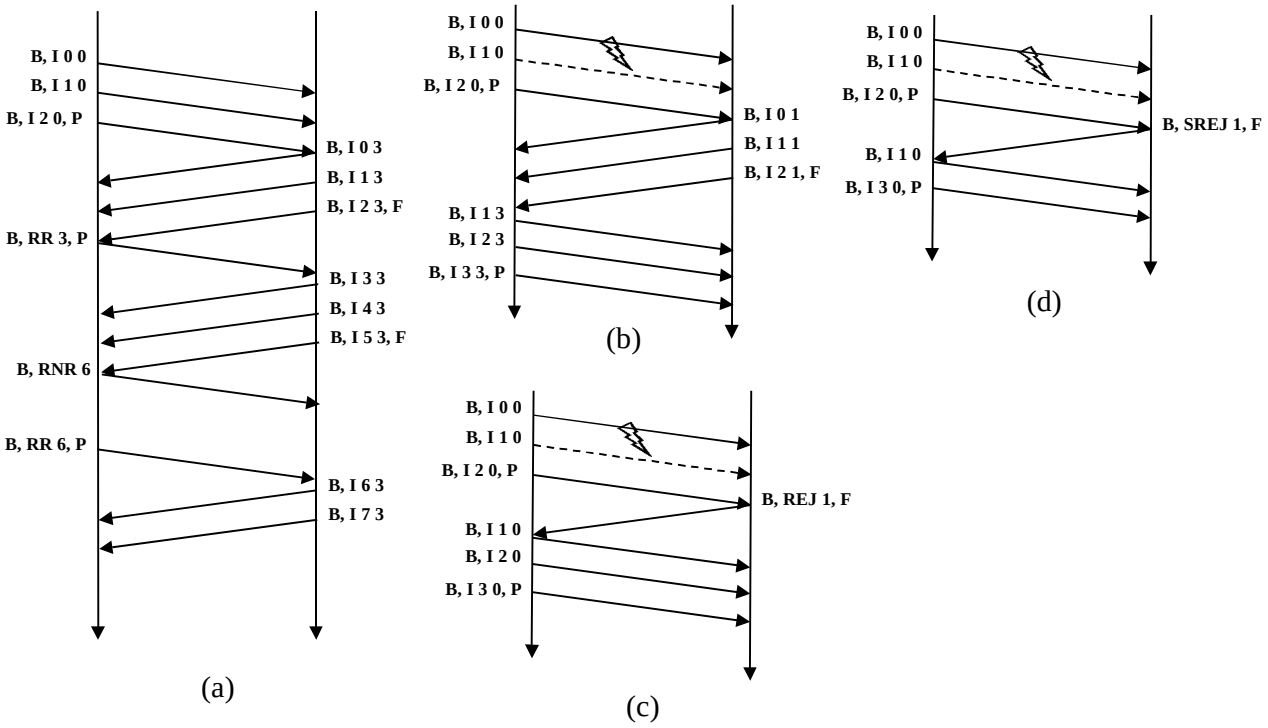

Figura 23: esempi di trasferimento dati in modalità NRM

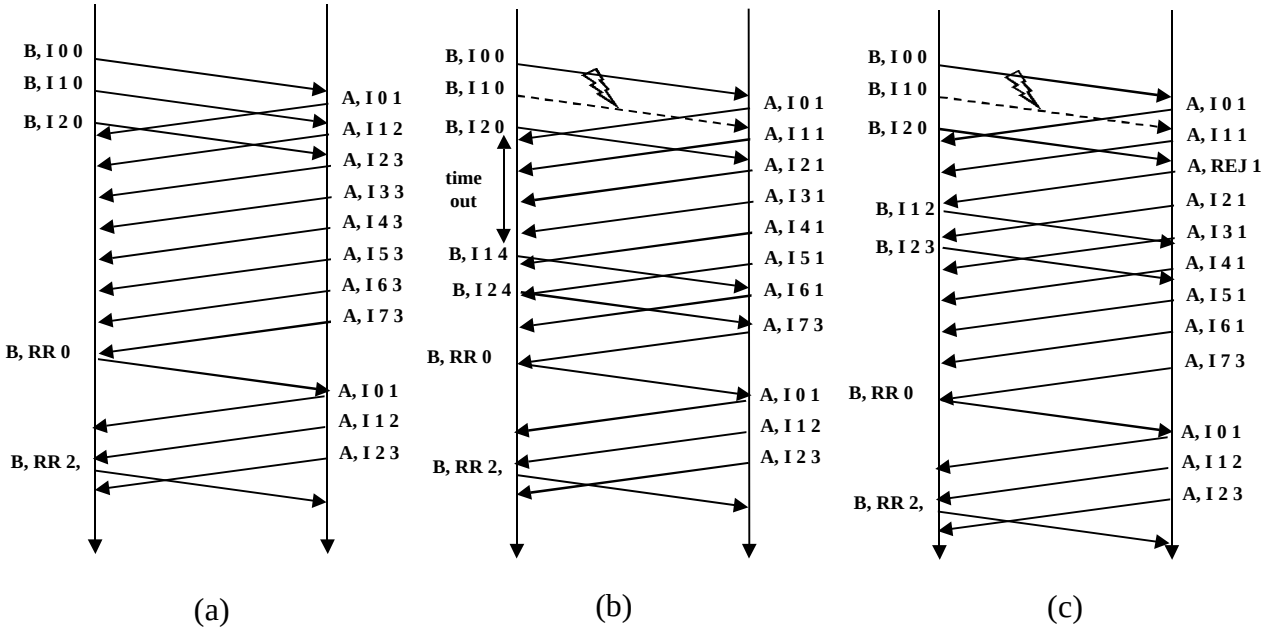

Figura 24: esempi di trasferimento dati in modalità ABM

Il protocollo HDLC ha dato origine a diversi altri protocolli di secondo livello utilizzati in vari standards. Alcuni di questi, sono molto simili al HDLC, ma introducono varianti per ovviare a mancanze nell'originale o per rispondere a requisiti specifici dell'ambiente:

- ✓ **LAPB (Link Access Procedure Balanced)**

  Questo protocollo è il protocollo di livello 2 utilizzato nella raccomandazione X.25 che definisce i primi tre livelli dell'interfaccia con la rete pubblica a commutazione di pacchetto. Il LAPB utilizza un sottoinsieme delle funzioni dell'HDLC. E' un protocollo a connessione sempre in Asynchronous Balanced Mode, ossia in modo full-duplex, con correzione d'errore. Il formato è quello già visto per HDLC, ma qui i campi di indirizzo sono fissi, valore 00000001 per il nodo di rete e valore 00000011 per il terminale d'utente. La procedura di controllo errore è sempre di tipo go-back-n mentre la numerazione dei pacchetti è effettuata modulo 8 o modulo 128, il che permette una finestra di valore 8 e 128 rispettivamente. All'arrivo di una trama fuori sequenza il protocollo invia una trama di REJ. In trasmissione, se scatta il time out del riscontro positivo, si può opzionalmente interrogare l'altra stazione col bit di P settato oppure ripartire a ritrasmettere la sequenza. Non viene usato il SREJ.

- ✓ **LAPD (Link Access Procedure for D channel)**

  E' questo il protocollo usato per la trasmissione a pacchetto su canale D di ISDN (si veda prossimo capitolo). Può funzionare solo in modalità ABM, come il LAPB. Viene usata solo la numerazione modulo 128 (SN e RN di 7 bit) e il campo address è sempre di 16 bit.

- ✓ **PPP (Point-to-Point Protocol)**

  Il protocollo PPP è un protocollo di livello 2 sviluppato nella comunità di Internet (RFC 1661 e seguenti) per venire incontro ad esigenze non supportate dall'HDLC e derivati, prima fra tutti la possibilità di multiplare diversi protocolli di livello 3. Il PPP supporta altre funzioni come la negoziazione di indirizzi IP e l'autenticazione. In realtà il PPP è formato da una serie di protocolli a vari sottolivelli del livello 2. Il protocollo di sottolivello più basso è il protocollo Line Control Protocol (LCP) il cui scopo primario è quello di costruire la trama finale che verrà consegnata al livello fisico. Il formato è mostrato in Figura 25 ed è derivato da quello HDLC con l'aggiunta del campo *Protocol*, che ha lo scopo di identificare i tipi di protocollo di livello 3. Rispetto allo HDLC, PPP introduce alcune limitazioni. Il campo Address deve contenere sempre tutti uni, che è la codifica dell'indirizzo di broadcast. Il campo Control deve contenere la sequenza 11000000, che indica che la trama è di tipo UI Unnumbered Information, quindi senza SN e acknowledgment e con trasmissione di tipo non connesso. Il campo Information ha lunghezza compresa tra 0 e 1500 byte. All'interno del campo Information può essere contenuto o un pacchetto del livello di rete o un pacchetto di controllo dei sottolivelli sopra LPC. In particolare questi ultimi sono usati per negoziare alcuni parametri come la lunghezza del campo I e per scambiare alcune informazioni di servizio relative al livello di rete da trasportare.

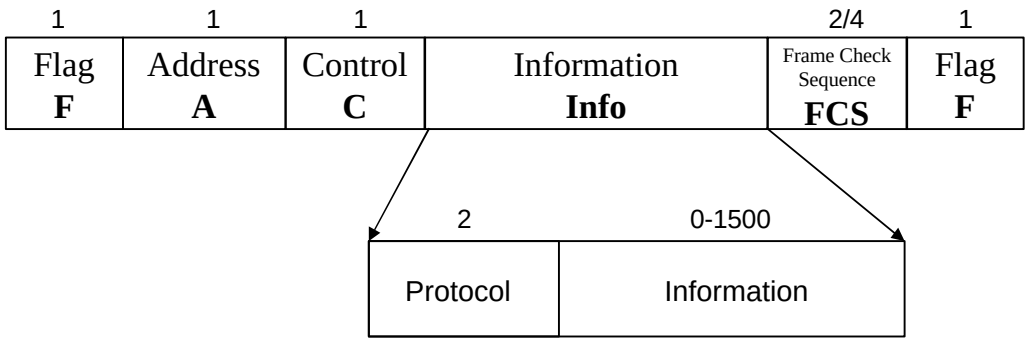

Figura 25: formato delle trame del PPP

5. Il controllo di congestione

Le reti a commutazione di pacchetto basano la loro efficienza sulla possibilità di condividere le risorse di rete da parte degli utenti. Le risorse che vengono condivise sono primariamente le risorse trasmissive, ovvero i canali di collegamento tra i nodi, ma anche i buffer di ricezione e trasmissione e la capacità di processamento dei pacchetti delle CPU dei nodi. Proprio questa condivisione di risorse è anche alla base di una caratteristica non desiderabile delle reti a pacchetto che è quella che si verifichi una congestione del traffico in rete.

In generale è possibile definire le situazioni di congestione come quelle nelle quali le risorse necessarie per gestire il traffico di pacchetti in rete sono più di quelle realmente disponibili. L'ovvio paragone è con la congestione del traffico automobilistico che percorre una rete di strade in un'ora di punta quando il flusso di auto è maggiore di quello smaltibile dalle strade. E' esperienza comune il fatto che l'aumento del traffico può generare delle situazioni che possono ridurre il numero di auto che riesce a trovare la via di uscita rispetto al numero consentito dalla capacità delle strade. Anche nelle situazioni di congestione delle reti a pacchetto è possibile che il traffico smaltito dalla rete venga ridotto molto più di quanto imposto dai limiti di capacità. In Figura 26 è mostrato un esempio di possibile andamento del traffico smaltito rispetto al traffico offerto in una rete nella quale non sia adottato alcun provvedimento contro la congestione (rete non controllata). Nella stessa figura è mostrato un possibile andamento nel caso in cui dei meccanismi efficaci di controllo di congestione vengano adottati. Dunque, scopo del controllo di congestione è quello di prevenire o risolvere le situazioni di congestione affinché il traffico smaltito dalla rete sia in ogni caso limitato solo dalla sua capacità.

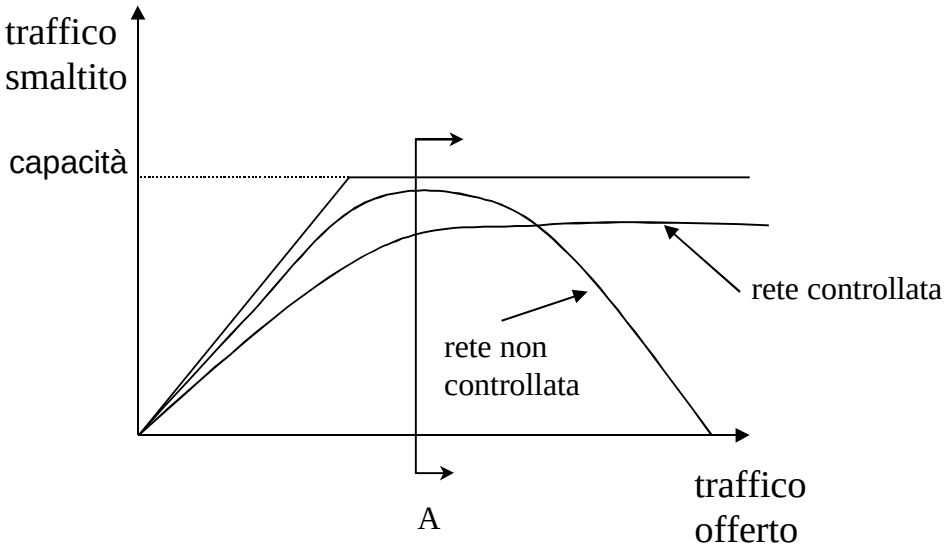

Figura 26: Controllo di congestione: tipico comportamento di reti controllate e non controllate

Un ovvio rimedio contro la congestione è quello di dimensionare la rete in modo che lavori sempre con traffico offerto minore della capacità (prima del punto A in figura). Sempre nell'analogia del traffico automobilistico, questo equivale dire che per rimediare alla congestione basta allargare le strade. Questa soluzione si scontra però con due problemi. Il primo è che non sempre è possibile in fase di progetto di una rete prevedere con precisione il traffico che dovrà sopportare. Il secondo problema deriva dal fatto che il traffico varia nel tempo e non risulterebbe conveniente dimensionare la rete sulla base del traffico dell'ora di punta.

Se la capacità di una rete non può essere modificata è sempre possibile "regolare il traffico", ovvero controllare dinamicamente il traffico in ingresso in rete in modo che non raggiunga livelli in grado di provocare la congestione.

Per comprendere i meccanismi con i quali può verificarsi della congestione si consideri la rete rappresentata in Figura 27 dove sui canali si indicata la capacità espressa in kbit/s.

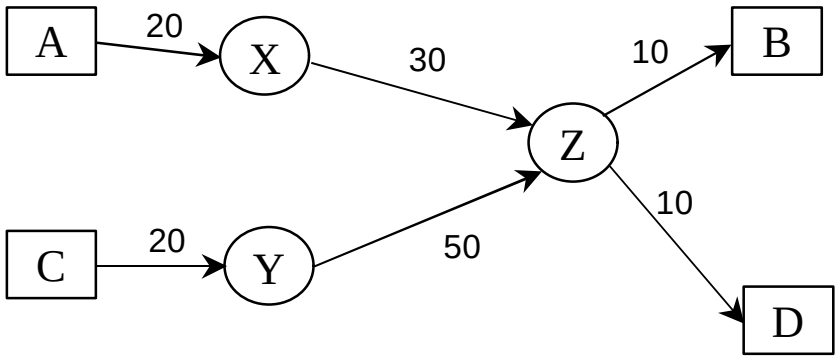

Figura 27: rete di esempio per controllo di congestione

Si assuma che la rete venga usata per il flussi di traffico da A a B e per il flusso da C a D, che il numero medio di pacchetti al secondo dei due flussi sia λ_{AB} e λ_{CD} rispettivamente e che i pacchetti siano lunghi 1000 bit. Si considerano diversi scenari.

Nel primo scenario si assume che $\lambda_{AB} = 9$ e $\lambda_{CD} = 0$. In questo caso non si verifica congestione perché il traffico offerto alla relazione di traffico AB è minore della capacità dei canali attraversati. Si supponga ora (secondo scenario) che il traffico λ_{AB} aumenti fino a 11 pacchetti al secondo. In questo caso si crea una situazione di congestione in quanto il traffico offerto è maggiore di quello del canale ZB. Nel nodo Z il buffer di trasmissione viene rapidamente riempito fino a che non si inizia a perdere dei pacchetti per trabocco. Se sul canale XZ è attivo un controllo d'errore con ritrasmissione o se tale controllo è attivo tra sorgente e destinazione, la perdita dei pacchetti in Z provocherà le ritrasmissioni dei pacchetti persi con conseguente ulteriore aumento del traffico. Se la rete permane in questa situazione anche i canali a monte di ZB risulteranno in breve tempo congestionati.

Si consideri adesso un terzo scenario nel quale $\lambda_{AB} = 9$ e $\lambda_{CD} = 9$. In questa situazione non si verifica congestione in quanto sia il traffico offerto dalla relazione di traffico AB che quello offerto dalla relazione di traffico CD risultano minori della capacità dei canali attraversati. Se adesso il traffico λ_{AB} aumenta fino a 11 pacchetti al secondo (scenario quattro) la rete entra in una situazione di congestione. La sola relazione di traffico che risulta in congestione è la relazione AB, ma si possono verificare funzionamenti nei quali anche la relazione di traffico CD può risentire della congestione. Si supponga, infatti che la capacità del buffer del nodo Z venga condivisa dai due canali in uscita. L'eccesso di traffico AB provoca il riempimento del buffer e la successiva perdita di pacchetti. Dato che il buffer è in comune, ed essere persi possono essere tanto pacchetti della relazione AB che sta provocando la congestione, che pacchetti della relazione CD.

Per fare in modo che il degrado di prestazioni dovuto alla congestione rimanga confinato sui pacchetti che attraversano i canali congestionati, come nello scenario quattro, basta dividere la memoria dei nodi in modo da ottenere dei buffer separati per canale d'uscita. Al contrario, l'unico modo per ovviare alla congestione nello scenario due e nello scenario quattro è quello di impedire alla sorgente A di emettere più di 10 pacchetti al secondo. Nell'esempio semplice considerato, però, scoprire a quanto limitare la velocità di generazione della sorgente è relativamente semplice, mentre in reti complesse nelle quali molti flussi di pacchetti condividono le risorse il limite non è di facile individuazione e soprattutto dipende dal traffico emesso dalla altre sorgenti che condividono gli stessi canali. Immaginare un controllore globale che controlla dinamicamente tutti i flussi di traffico e calcola i limiti per ognuno dei flussi è, dunque, di nessuna utilità pratica.

I meccanismi per il controllo della congestione comunemente adottati sono di tipo distribuito e coinvolgono più di un elemento della rete. In queste dispense i meccanismi di controllo di congestione vengono distinti in due grandi insiemi:

- ✓ meccanismi per reti che offrono un servizio senza connessione e senza garanzie di qualità (*meccanismi di controllo senza connessione*)
- ✓ meccanismi per reti che offrono un servizio orientato alla connessione e con garanzie di qualità (*meccanismi di controllo con connessione*)

Nei primi la rete deve far si che il flusso emesso da ciascuna sorgente venga regolato in modo tale che il flusso complessivo offerto a ciascun canale non superi la sua capacità. Il controllo deve essere esercitato in modo continuo su tutte le sorgenti e tutti i flussi possono essere ridotti in modo tale che la capacità della rete venga condivisa da tutti in misura se possibile uguale. Il servizio offerto dalla rete è senza garanzie di qualità in quanto la velocità di trasferimento dei flussi e le risorse da essi realmente utilizzate non sono predicibili, ma dipendono dalla situazione di traffico in rete. Tipico esempio di rete che usa meccanismi di questo tipo è Internet, almeno nella versione attualmente in uso.

Nelle reti che offrono servizi orientati alla connessione, come ad esempio le reti ATM, è sia possibile usare gli stessi meccanismi usati nelle reti con servizi senza connessione, sia usare dei meccanismi specifici particolarmente utili per garantire agli utenti una buona qualità. Durante la fase di setup della connessione, infatti, è possibile che la sorgente comunichi alla rete le caratteristiche del flusso dati (ad esempio la velocità media) e che la rete, sulla base della conoscenza delle caratteristiche dei flussi delle connessioni già in corso, decida se le risorse sono o meno sufficienti anche per la nuova connessione. Nel caso in cui le risorse non siano sufficienti la connessione viene rifiutata. Questo meccanismo di regolazione delle connessioni viene detto Connection Admission Control (CAC). Il compito del meccansimo di controllo di congestione, però, non si esaurisce al CAC in quanto dopo che una connessione è stata accettata è necessario controllare all'ingresso in rete che il flusso non ecceda le velocità dichiarate durante il setup. Gli utenti che accedono al servizio, se la connessione viene accettata, ricevono un servizio in un certo senso a qualità garantita in quanto i pacchetti non incontrano congestione e l'intero flusso con le caratteristiche negoziate viene trasferito a destinazione.

5.1 Meccanismi di controllo senza connessione

I meccanismi di controllo senza connessione di uso più comune sono sicuramente riconducibili alla famiglia dei meccanismi di controllo a finestra. Si è già parlato della tecnica a finestra in relazione al controllo di flusso e si è visto come questa tecnica è in grado di ridurre la velocità di invio dei pacchetti da parte della sorgente in relazione alla velocità di assorbimento del ricevitore. Lo stesso approccio può essere usato per ridurre la velocità della sorgente allo scopo questa volta di evitare situazioni di congestione in rete.

Se si assume trascurabile il tempo di riscontro, è possibile affermare che il controllo a finestra limita il numero di pacchetti in viaggio tra sorgente e destinazione. Tale numero può, in linea di principio, essere dimensionato per prevenire situazioni di congestione. Due differenti approcci possono essere seguiti: implementare uno stretto controllo a finestra, tipicamente a livello di linea, su ogni canale delle rete (controllo hop-by-hop), oppure implementare un controllo a finestra solo tra sorgente e destinazione finali ai bordi della rete (controllo end-to-end).

Nel caso di controllo hop-by-hop sembra relativamente semplice fare in modo che nessuno dei canali tra i nodi della rete vada in congestione regolando la dimensione della finestra in relazione allo spazio nel buffer. In realtà, la cosa non è priva di complessità se si pensa alla situazione di un nodo con più canali d'ingresso non congestionati e più canali d'uscita di cui uno in congestione. In questo caso, infatti, il nodo può limitare l'arrivo dei pacchetti sui canali d'ingresso indipendentemente dal fatto che i pacchetti in arrivo siano o meno da instradare verso il canale congestionato d'uscita. Ad ogni modo, la limitazione esercitata dal nodo congestionato ha come effetto quello di far propagare all'indietro, nodo per nodo, la limitazione fino alle sorgenti di traffico che saranno alla fine forzate a ridurre la propria velocità dalla finestra del primo canale (Figura 28).

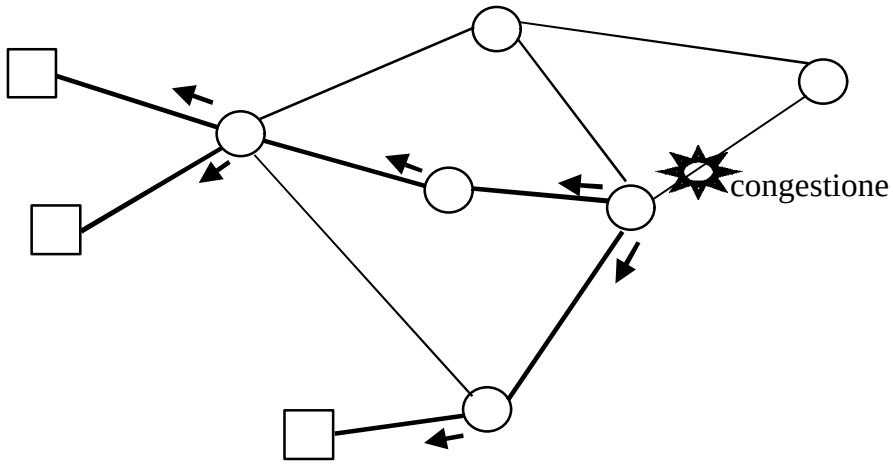

Figura 28: limitazione all'indietro esercitato dal controllo a finestra hop-by-hop

Il controllo a finestra hop-by-hop deve agire su tutto il traffico che attraversa un canale. Se questo non rappresentava un problema nelle reti con basse velocità di trasferimento, così non è per le attuali reti con linee ad altissima velocità come quelle consentite dalle fibre ottiche. Infatti, per impedire che il controllo abbassi l'efficienza di utilizzo del canale occorre che la finestra (in bit) sia almeno pari al prodotto della capacità per il ritardo di propagazione; tale valore nei collegamenti ad alta velocità può essere molto grande e difficilmente gestibile dai dispositivi addetti al controllo di linea. Per questo motivo il controllo hop-by-hop non è usato nelle reti ad alta velocità.

Il controllo a finestra di tipo end-to-end, invece, limita il numero di pacchetti in circolazione in rete per ogni flusso in ingresso. Il problema è che non si può sapere a priori dove siano i pacchetti di ogni flusso, che, per esempio, possono trovarsi tutti nello stesso punto della rete. Una soluzione sembrerebbe quella di limitare il numero di pacchetti per tutti i flussi in modo tale che anche nella situazione peggiore non siano in grado di congestionare alcun canale. Naturalmente, questa soluzione, anche se sicuramente previene la congestione, è molto limitante e tende a far sottoutilizzare la capacità della rete.

Di solito, quindi, l'approccio adottato è quello di variare dinamicamente le dimensioni delle finestre di trasmissione dei flussi quando in rete si è verificata o sta per verificarsi una situazione di congestione. Due differenti casi si possono presentare in relazione al fatto che la rete sia completamente passiva e non in grado di accorgersi della congestione o invece sia attiva, in grado cioè di accorgersi della congestione e di mandare delle notifiche esplicite alle sorgenti (explicit congestion notification).

Una rete attiva nel controllo di congestione si può ottenere con dei nodi in grado di monitorare lo stato di congestione mediante, ad esempio, una soglia sul livello di occupazione dei buffer o sul grado di utilizzo della CPU. La notifica della congestione alla sorgente può avvenire in molti modi. Ad esempio è possibile inviare dei pacchetti di segnalazione, o intercettare i pacchetti in verso opposto e settare un opportuno flag, oppure ancora settare un flag nei pacchetti in transito e lasciare alla destinazione il compito di ridurre la finestra mediante i messaggi del protocollo end-to-end. La sorgente può adottare varie politiche di reazione alla ricezione di una notifica di congestione; può ad esempio ridurre la finestra in modo proporzionale al numero di notifiche ricevute in un intervallo temporale oppure ridurre la finestra fino al valore minimo alla ricezione di una notifica e poi riaprire lentamente la finestra.

Se la rete è passiva, invece, le sorgenti devono essere in grado di capire se si è creata una situazione della congestione senza l'aiuto di notifiche esplicite. Un approccio semplice, adottato in pratica, consiste nel ipotizzare una congestione ogni volta che si rende necessaria la ritrasmissione di un pacchetto. La necessità della ritrasmissione può essere evidenziata sulla base di un time-out di attesa dell'acknowledgement. A sua volta il mancato arrivo dell'ACK può essere dovuto alla perdita del pacchetto per overflow di uno dei buffer dei nodi di rete o ad un elevato ritardo di attraversamento della rete, entrambe indicatori di una congestione in rete. Purtroppo la perdita del pacchetto o del relativo ACK può essere dovuta anche ad altre cause come ad esempio un errore nel pacchetto e quindi questo approccio risulta critico quando la probabilità che i pacchetti vengano persi per motivi indipendenti dalla congestione è elevata. Una volta scaduto il time-out la

sorgente può reagire in molti modo. Ad esempio può porre al valore minimo la finestra di trasmissione, e, successivamente, riaprire la finestra in misura pari al numero di ACK validi ricevuti (Figura 29). E' questo, ad esempio, l'approccio adottato nel protocollo di trasporto TCP utilizzato nelle reti IP.

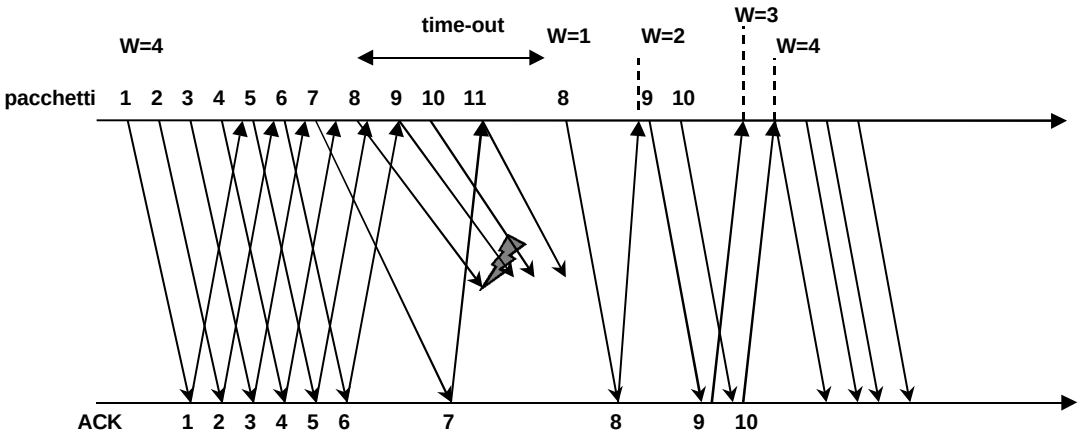

Figura 29: regolazione della finestra in funzione dei time-out e degli ACK

5.2 Meccanismi di controllo con connessione

I meccanismi di controllo di congestione per servizi con connessione e garanzia di qualità sono basati sul controllo di ammissione delle nuove connessioni e sul controllo del traffico in ingresso per le connessioni accettate.

Il controllo d'ammissione delle nuove connessioni è basato sui parametri descrittori del flusso dichiarati dalla sorgente durante la fase di setup. Quali e quanti parametri sono usati per descrivere il flusso dipendono dal tipo di rete considerata e dal tipo di servizio richiesto, come sarà meglio chiaro quando verrà descritta le tecnica ATM. Se ad esempio il flusso considerato è caratterizzato da un ritmo costante di generazione dei pacchetti, l'unico parametro da considerare è appunto il ritmo di generazione dei pacchetti. Durante il setup la rete deve decidere se sul cammino nella rete tra sorgente e destinazione c'è abbastanza capacità per un nuovo flusso con il ritmo dichiarato. In caso negativo la connessione è rifiutata (Figura 30), mentre in caso positivo la connessione è accettata (Figura 31) e il traffico in ingresso è monitorato per controllare che il ritmo di generazione non ecceda quello dichiarato.

Molti modi sono stati considerati per controllare che il traffico non ecceda i parametri dichiarati in fase di setup. Un schema spesso utilizzato è noto con il nome di *leaky bucket* ed è esemplificato in Figura 32.

In un buffer sono immagazzinati dei token (permessi) che vengono generati ad un ritmo costante R. Alla generazione di un nuovo pacchetto questo viene ammesso in rete solo se vi è un token disponibile, altrimenti viene messo in attesa in un buffer o viene scartato. Questo meccanismo assicura che il ritmo medio con il quale i pacchetti arrivano in rete è inferiore o pari ad R.

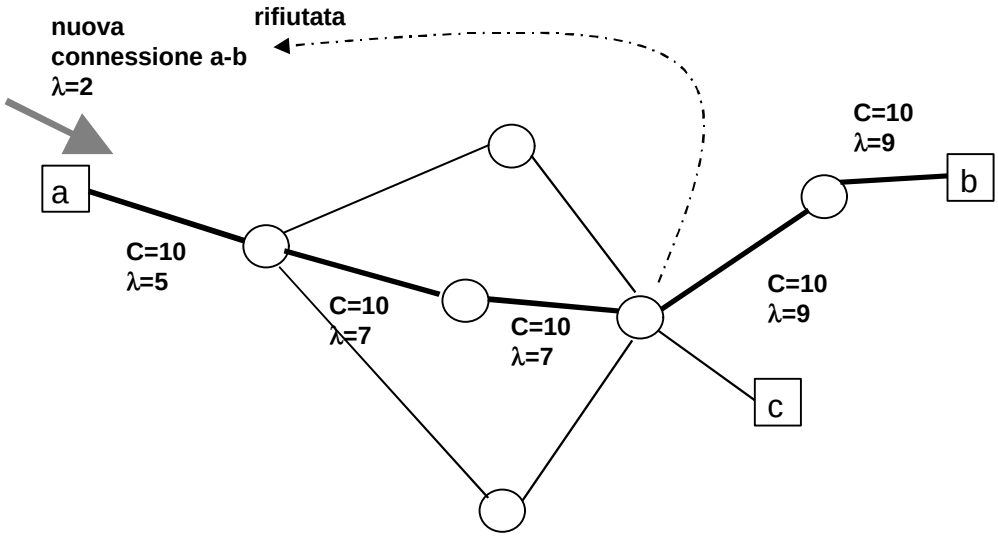

Figura 30: controllo di ammissione, rifiuto di una connessione

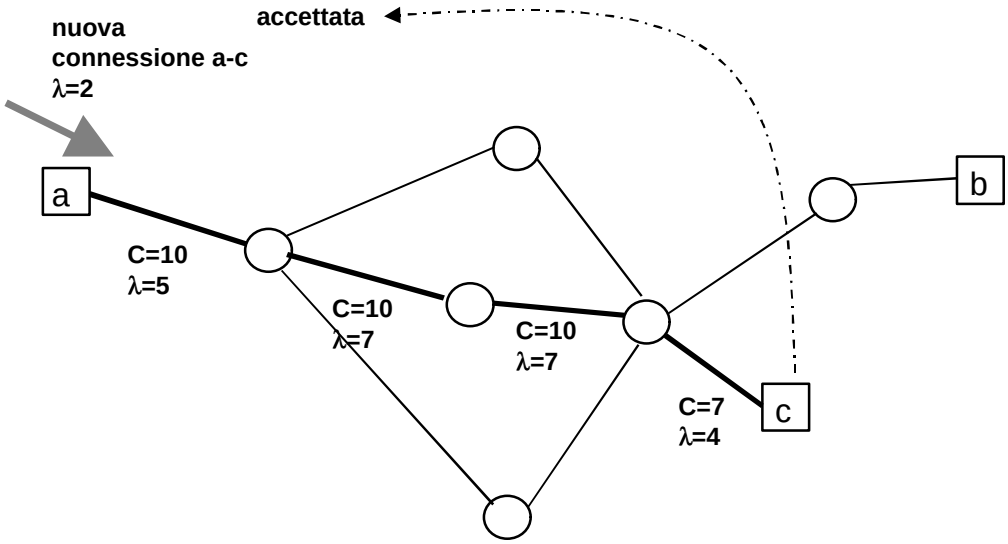

Figura 31: controllo di ammissione, accettazione di una chiamata

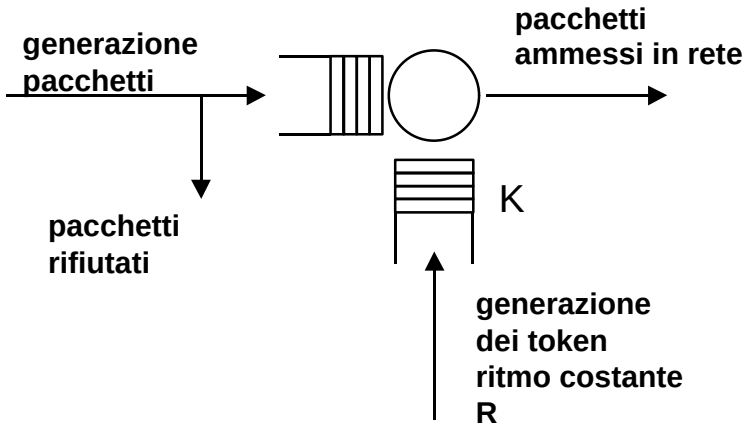

Figura 32: token bucket

Il buffer dei token ha dimensione pari a K. Se un token arriva e trova il buffer pieno viene perso. Se una sequenza numerosa di pacchetti arriva e trova il buffer dei token pieno, in rete vengono ammessi in sequenza al ritmo massimo P (ritmo di picco) i primi K pacchetti[1], mentre i successivi sono ammessi a ritmo pari ad R (Figura 33). Naturalmente se K=0 un pacchetto è ammesso solo se già presente al momento della generazione del token e quindi il ritmo istantaneo di immissione in rete è sempre minore o uguale ad R.

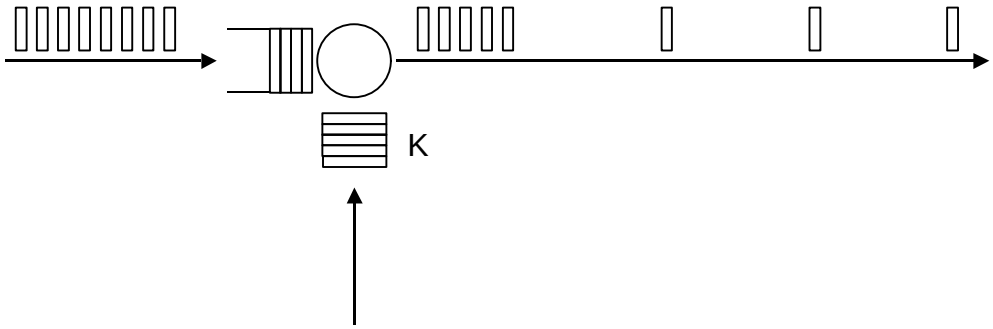

Figura 33: esempio di funzionamento del token bucket

[1] si ipotizza che il ritmo di picco P sia molto maggiore di R.