DAL LIVELLO DI LINEA AL LIVELLO DI TRASPORTO – IL LIVELLO DI LINEA 2

I vari tipi di servizi offerti 2

Il livello di linea o livello di data link 5

Struttura dei pacchetti 6

Character stuffing 8

Bit stuffing 9

Trattamento degli errori 10

Protocolli di comunicazione 14

Protocolli di tipo simplex non limitato 15

Simplex stop and wait 17

Protocolli a controllo e ritrasmissione 18

Protocollo BSC (Binary Synchronous Communication) 24

Protocolli piggybacking 29

Protocolli a finestra scorrevole 31

Protocollo a finestra scorrevole di un bit 32

Protocolli go-back-n e selective repeat 33

Dal livello di linea al livello di trasporto – Il livello di linea

I vari tipi di servizi offerti

I servizi offerti dai vari livelli che andiamo a descrivere sono classificabili in tre tipologie:

- servizi non orientati alla connessione e senza riscontro: un livello che offre servizi di questo tipo invia i dati verso il livello paritetico del computer destinatario senza preoccuparsi di assicurarsi che esso sia in ascolto. Inoltre questo tipo di servizi non prevede che si debba ricevere dall'interlocutore un messaggio di risposta che confermi l'avvenuta ricezione dei dati. Per fare un esempio potete pensare all'invio di informazioni ad un altro essere umano mediante lettere. Noi inviamo i nostri dati racchiusi nel pacchetto lettera al presunto indirizzo del destinatario senza essere sicuri che a tale indirizzo vi sia realmente la persona in grado di ricevere le nostre informazioni. Inoltre con la posta semplice non vi è alcuna possibilità di sapere se la lettera sarà ricevuta o meno.

- servizi non orientati alla connessione con riscontro. Anche in questo caso si inviano dati al destinatario senza aprire prima una connessione, senza accertarci cioè se il destinatario sia realmente presente all'indirizzo presunto. In questi tipi di servizi ci si aspetta però di ricevere un messaggio di riscontro o acknowledgment se l'invio dei dati va a buon fine. Possiamo fare l'esempio della lettera raccomandata AR che come la posta semplice non ci assicura che il destinatario sia presente all'indirizzo presunto ma, in caso di corretta ricezione della lettera ce ne da conferma mediante l'invio da parte del destinatario di cartolina di ritorno

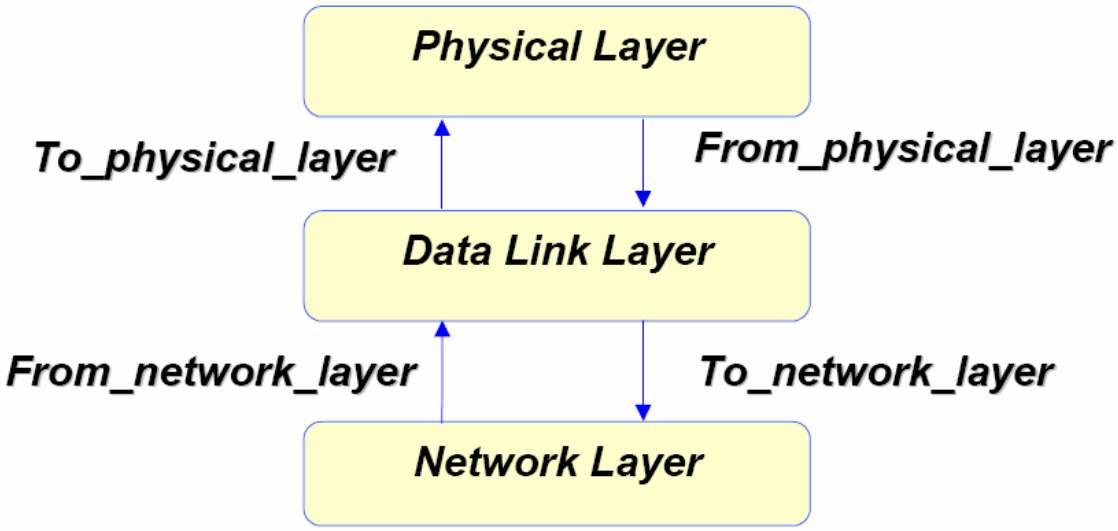
- servizi orientati alla connessione: questi tipi di servizio prevedono che prima di inviare i dati il computer mittente stabilisca prima una connessione logica con il computer destinatario. Un esempio è la comunicazione telefonica in cui prima di inviare informazioni al destinatario ci si assicura anzitutto che l'interlocutore sia disponibile all'altro capo della linea e disposto a comunicare

Una cosa da tener presente è il fatto che meno è affidabile il sevizio offerto da uno strato inferiore più è complesso il lavoro per lo strato superiore.

Supponiamo ad esempio che stiate giocando una partita a scacchi a distanza con una persona che si trovi su un'isola in cui non vi è copertura telefonica. Avete deciso di comune accordo di comunicarvi le mosse per lettera e, per risparmiare, utilizzate la posta ordinaria. È un sistema composto da due livelli: il livello fisico di inoltro dei messaggi costituito dal sistema postale e il vostro livello in cui vi occupate di giocare la partita. Poiché il livello fisico che utilizzate è inaffidabile fornendo un servizio non orientato alla connessione e senza riscontro, sarà il vostro livello a dover operare per aumentare la qualità della comunicazione, prevedendo, ad esempio, che quando uno dei due giocatori riceve una lettera dall'altro abbia l'obbligo di spedire una lettera di risposta in cui si ricopi la mossa inviata dall'avversario.

Il livello di linea o livello di data link

Costituisce il secondo strato del modello OSI e definisce le regole attraverso le quali avviene il dialogo fra due nodi di una rete. Esso si trova fra il livello fisico, il cui scopo è quello di garantire il corretto trasferimento di bit lungo un canale, ed il livello di rete che si occupa di gestire l'instradamento dei dati all'interno di una rete di computer.



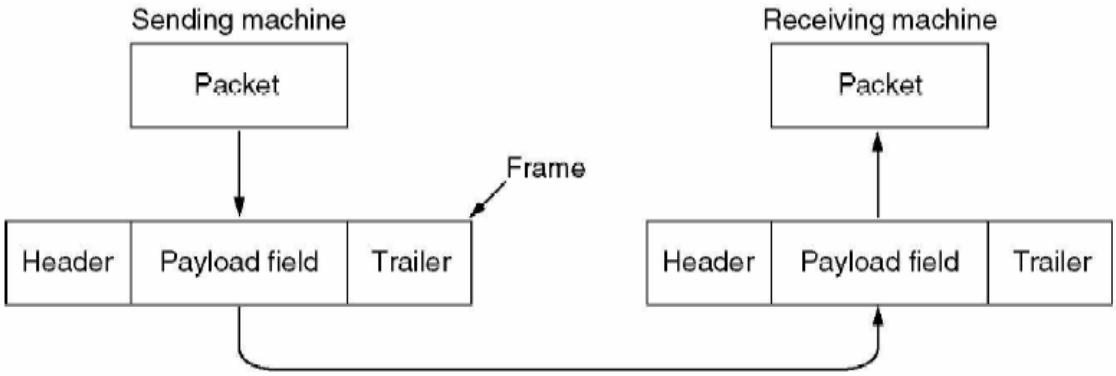
Nel livello di linea i dati vengono organizzati in frame (tipicamente poche centinaia di byte) che vengono trasmessi sequenzialmente. Tale livello si occupa inoltre

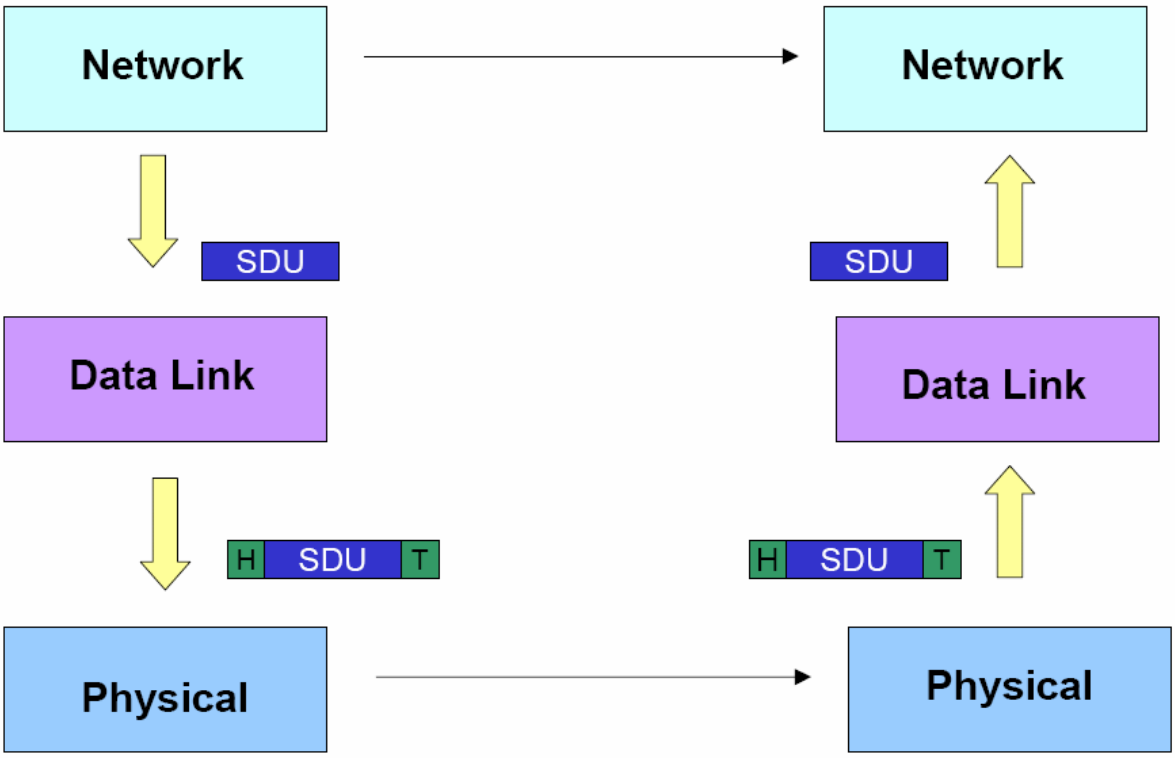
- delle regole per l'instaurarsi del dialogo fra trasmettitore e ricevitore,
- della loro sincronizzazione,
- delle regole per l'interruzione della comunicazione,

- della verifica della presenza di errori nella ricezione del frame utilizzando caratteri di controllo,
- della ritrasmissione degli stessi frame quando si rileva un errore.

Struttura dei pacchetti

Per prima cosa lo strato di linea riceve il pacchetto dati dal livello di rete e lo racchiude in un frame. Il frame è costituito dal pacchetto ricevuto dallo strato superiore con l'aggiunta di caratteri di controllo.





Il formato del frame può essere di diversi tipi a seconda del protocollo datalink utilizzato. In generale deve avere però un'organizzazione del seguente tipo.

Formato delle FRAMES

Synchronization Field	Address Field	Control Field	Data Field	Error check Field	Synchronization Field

Synchronization Field Consente di distinguere l'inizio e la fine di una frame, mediante l'uso di opportuni pattern di bit.

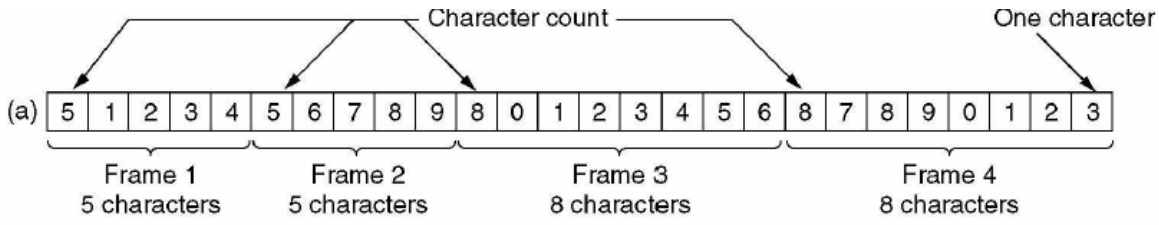
Address Field Contiene l'indirizzo del destinatario/sorgente.

Control Field Contiene una serie di informazioni di controllo utilizzate per individuare il tipo di frame.

Data Field Contiene un pacchetto del Network Layer.

Error Check Field Contiene una sequenza di byte utilizzati per la rilevazione di errori presenti nella frame.

Si pone innanzitutto il problema di distinguere all'interno del flusso continuo di bit che viaggiano fra trasmettitore e ricevitore un frame dall'altro. Una prima tecnica possibile tipo è il conteggio dei caratteri o byte: si utilizza un campo nell'header del frame per indicare quanti caratteri sono presenti nello stesso.



Questo tipo di formato, benché semplice, presenta, però, un problema. Se, infatti, si ha un errore di trasmissione proprio sul primo byte che indica la lunghezza del frame, il ricevitore non sarà in grado di recuperare l'informazione contenuta in esso. Supponiamo ad esempio, di avere un frame di 100 byte. Il primo byte dovrebbe essere

pari dunque, a 100. Se, per errore, tale byte si trasforma in 98, per il ricevitore gli ultimi due byte del frame non faranno parte di esso ma del frame successivo. L'errore si propaga in modo disastroso. Infatti il 99esimo byte del frame verrebbe interpretato come il byte che contiene la lunghezza del frame successivo per cui il ricevitore non riuscirà a delimitare il secondo frame e così via.

Character stuffing

Un secondo tipo di formattazione del frame prevede che esso inizi con la sequenza di due caratteri ascii:

DLE (DATA LINK ESCAPE)

STX (START OF TEXT)

e termini con un'altra coppia di caratteri

DLE (DATA LINK ESCAPE)

ETX (END OF TEXT)

Esempio

I dati da trasferire sono i seguenti

55	A6	02	F7	25	0B	15

Tenendo presente che il codice di DLE è 10, quello di STX è 02 e quello di ETX è 03 il frame diventa

10	*02*	55	A6	02	F7	25	0B	15	*10*	*03*

Supponiamo ora che i dati da trasferire diventino i seguenti

55	A6	02	F7	10	03	15

Il frame diventa

10	*02*	55	A6	02	F7	10	03	15	*10*	*03*

Si nota come ci sono due byte di dati che coincidono con i codici di controllo finali. Il ricevitore potrebbe interpretarli, allora, come fine del frame. Per evitare quest'equivoco si usa la tecnica di character stuffing o riempimento di caratteri. In

sostanza, ogni volta che nei dati appare un byte pari a 10 cioè DLE, viene aggiunto un altro byte pari a 10. nel nostro caso avremmo

10	02	**55**	**A6**	**02**	**F7**	10	10	03	**15**	10	03

Vedendo due byte pari a 10 che si seguono il ricevitore capisce che non si trova di fronte a caratteri di controllo ma a dati ed elimina il byte in più. Cosa succede se nei dati da trasmetter vi sono già due byte pari a 10?

10	**02**	55	A6	02	F7	**10**	**10**	**03**	15	**10**	**03**

Nessun problema! Dopo ogni byte del dato pari a 10 il trasmettitore inserisce un nuovo byte pari a 10

10	**02**	55	A6	02	F7	**10**	**10**	**10**	**10**	**03**	15	**10**	**03**

In ogni caso, poiché il ricevitore sa che il trasmettitore inserirà sempre un byte pari a 10 ogni volta che incontra nei dati un byte pari a 10, interpreta le coppie 10, 10 come dati e non caratteri di controllo ed elimina il secondo byte della coppia, ricostruendo il frame originale.

Naturalmente, se vi è un errore di trasmissione che modifica un byte di controllo, il ricevitore non riesce a delimitare un frame. Però quando arriva una nuova coppia DLE STX esso capisce che inizia un nuovo frame e può richiedere il rinvio del frame precedente. Un altro difetto consiste nel fatto che questo tipo di formattazione dei frame si base sul codice ascii che prevede che i caratteri siano sempre espressi su 8 bit. Questo è un limite poiché impedisce di utilizzare tecniche di compattazione dei dati basate sul fatto che i vari caratteri possano essere espressi anche con un numero di bit variabili.

Bit stuffing

Si usa allora una formattazione orientata al bit in cui il campo dati non è rappresentato da una sequenza di byte ma una stringa di bit di lunghezza variabile. Il frame è delimitato da una stringa di bit pari a 01111110 all'inizio e alla fine

Se vogliamo inviare, ad esempio, la seguente stringa 0001110011, essa diventa **01111110** 0001110011 **01111110**. Cosa succede se, all'interno della stringa dei dati da trasmettere compare proprio una stringa uguale alla stringa delimitatrice? Ad esempio 00*01111110*011? Si avrebbe il seguente frame

0111111000*01111110*011**01111110**

Nell'esempio il ricevitore interpreterebbe la sequenza in corsivo come delimitatore del frame. Riterrebbe di aver ricevuto solo il frame costituito dalla sequenza 00. Per evitare che il ricevitore possa interpretare la stringa di bit dato come il delimitatore della fine del frame, si usa la tecnica del bit stuffing. In sostanza ogni volta che vi sono 5 bit 1 in sequenza, il trasmettitore inserisce uno zero

0111111000*011111010*011**01111110**.

Nota bene: il trasmettitore inserisce sempre un bit pari a 0 dopo cinque bit pari ad 1. Anche se non ce n'è bisogno.

Esempio

0111111000*01111100*011**01111110**

diventa

0111111000*011111000*011**01111110**

Nell'esempio non c'era possibilità di interpretazione scorretta ma il bit 0 aggiuntivo viene inserito ugualmente per semplificare il metodo. In ogni caso il ricevitore sa che dopo cinque bit ad 1, il sesto bit va eliminato perché non fa parte dei dati. In ogni caso, con il bit stuffing non potrà mai succedere che il campo dati possa contenere una stringa di bit uguale alla stringa delimitatrice.

Trattamento degli errori

In un frame vi deve anche essere un campo che consenta in ricezione di rilevare se durante il processo di trasmissione il frame sia stato corrotto da errori.

Per progettare una codifica a semplice rilevazione di errore si utilizza la codifica polinomiale, detta anche controllo ciclico di ridondanza (CRC), in cui ogni sequenza di

bit è vista come un polinomio, ad esempio 110001 ha 6 bit, ed è vista come polinomio di grado 5 con coefficienti pari al valore dei bit. Ad esempio:

$$110001 = x^5 + x^4 + x^0$$

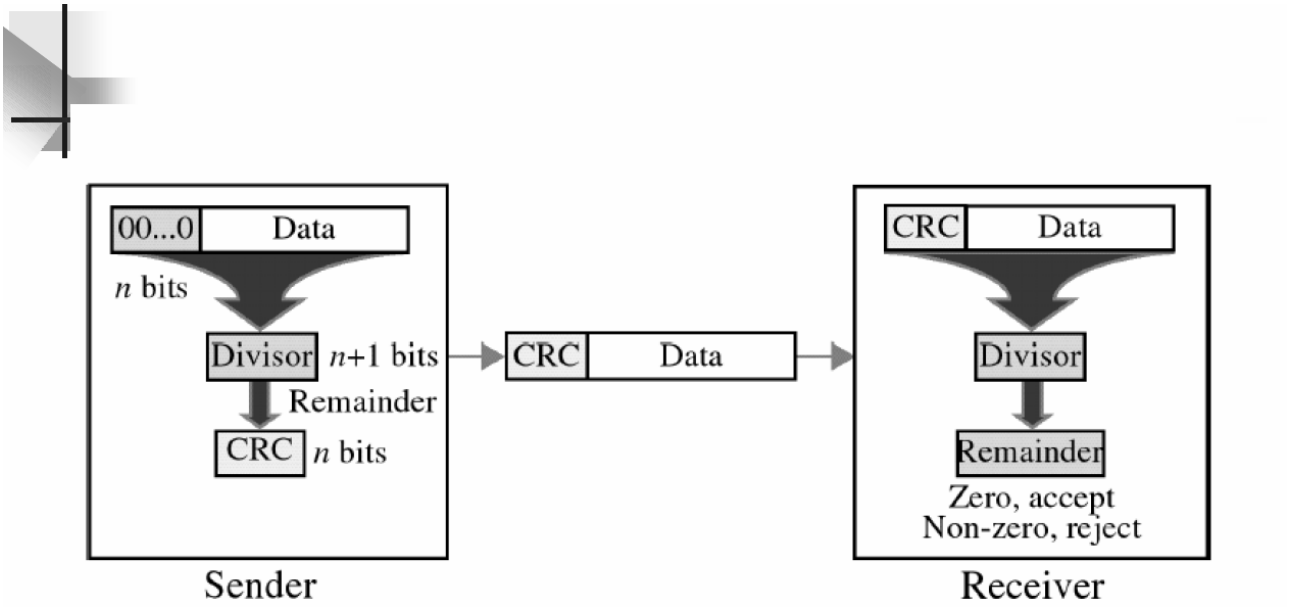
Su questo scenario di base, si aggiunge ad ogni frame una sequenza di bit detta checksum, in modo da rendere il polinomio 'frame+checksum' divisibile per un polinomio G(x) detto polinomio generatore e concordato fra trasmettitore e ricevitore. Il ricevitore divide il polinomio ricevuto per G(x), e se ottiene resto non nullo, sa che tale polinomio ha subito un errore (rilevazione)

L'algoritmo per la generazione del checksum è il seguente:

- Si moltiplica il frame originario M(x) per x^r, dove r è il grado del polinomio generatore G(x), shiftando di fatto M(x) di r-1 bit a sinistra, aggiungendo zeri alla destra
- Si divide il frame così ottenuto per G(x) e si calcola il resto
- Si sottrae tale resto al frame shiftato, ottenendo un 'frame+checksum' di fatto reso divisibile per G(x)

Per ben comprendere questo metodo bisogna tener presente che si utilizza un'aritmetica particolare detta aritmetica modulo 2. Per dirla in maniera semplice un'aritmetica modulo n è un'aritmetica in cui non sono possibili infiniti numeri interi come nell'aritmetica che utilizziamo normalmente ma soltanto i numeri 0, 1, 2… n-1. ne deriva che un'operazione aritmetica non può dare come risultato un numero superiore a n-1. Se eseguendo la stessa operazione nell'aritmetica normale ottenessimo un

numero superiore ad n, il risultato in aritmetica modulo n si ottiene come resto della divisione per n.



- Consideriamo ad esempio una aritmetica modulo 5: sono possibili i numeri da 0 a 4. Se eseguiamo l'operazione 1+2=3<5 e non c'è problema. Invece 3+4=7; il resto di 7/5 è 2 per cui 3+4=2 mod 5. In aritmetica modulo 2 sono possibili i numeri 0 ed 1. per cui 0+1=1 (no problem) ma 1+1=2-> 1+1=0 modulo 2. Se eseguiamo una sottrazione 0-1=1 modulo 2 infatti 1+1=0 modulo 2. Se sottraiamo due bit abbiamo allora la seguente tabella

Primo bit	Secondo bit	Risultato
0	0	0
0	1	1
1	0	1
1	1	0

che corrisponde alla tabella della x-or. Sottrarre due stringhe di bit corrisponde dunque ad effettuare la x-or bit a bit delle due stringhe. Vediamo l'esempio di figura

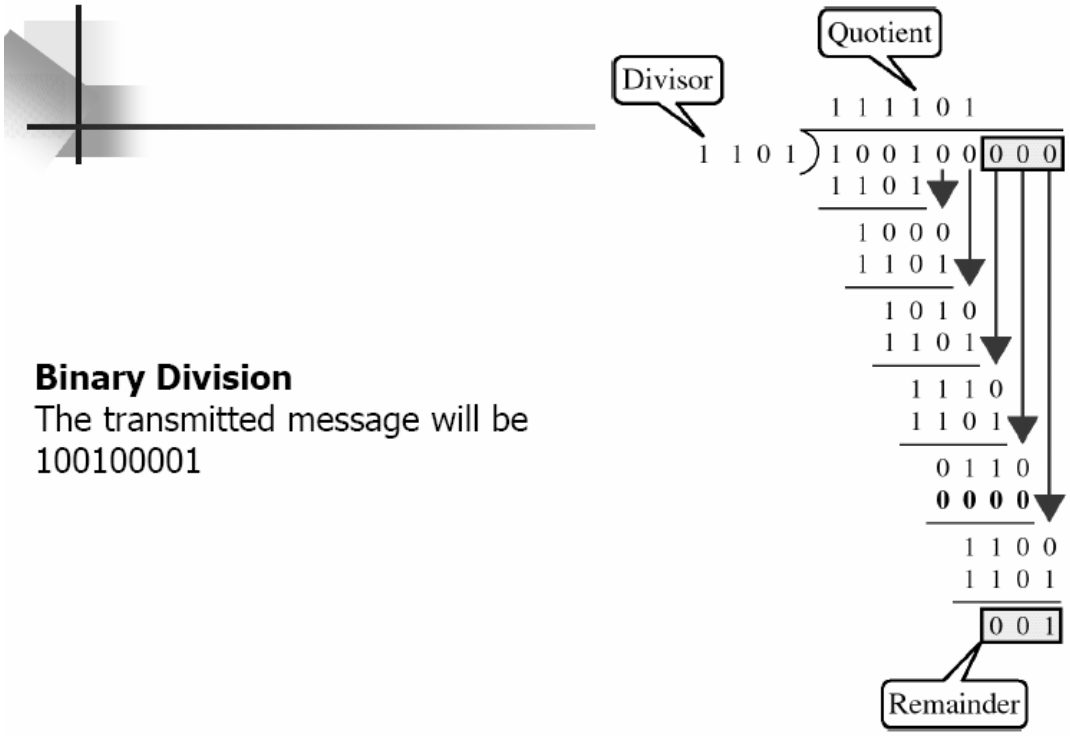
- **x3** + **1** can be divided by **x3** + **x2** + **1**
- and the remainder is
- 0 ×**x3** +1 ×**x2** +0 ×**x1**+ 0 ×**x0** = **x2**
- In terms of messages
- 1001 can be divided by 1101 and leaves a remainder of 0100

$$\begin{array}{r} 1001 \\ 1101 \\ \hline 0100 \end{array} \oplus$$

Proviamo a fare la verifica inversa: se moltiplichiamo il divisore per il quoziente (che nel nostro caso è 1) e sommiamo il resto dovremmo avere il polinomio di partenza

$(X^3+X^2+1)*1+X^2=X^3+(1+1)X^2+1=X^3+0*X^2+1=X^3+1$ che è il dividendo.

Consideriamo un esempio completo: il messaggio è 100100 è il polinomio generatore è X^3+X^2+1 corrispondente a 1101. poiché il grado del polinomio generatore è 3 dobbiamo moltiplicare il messaggio per X^3 cioè shiftare il messaggio a sinistra di tre bit.



Protocolli di comunicazione

Nella comunicazione fra trasmettitore e ricevitore si devono risolvere i problemi di sincronizzazione che consistono nel garantire la corrette trasmissione dei frame anche quando trasmettitore e ricevitore lavorano a velocità diverse ed inoltre garantire un controllo che informi il trasmettitore che il ricevitore ha ricevuto un frame in modo corretto ed, in caso contrario, gli rinvii il frame.

Protocolli di tipo simplex non limitato

I dati possono andare soltanto in una direzione. All'attivazione del collegamento viene definito quale nodo funzione da trasmettitore e quale da ricevitore. Il ricevitore, da questo momento in poi, può soltanto ricevere dati e spedire al massimo caratteri di conferma.

Protocollo 1: Heaven

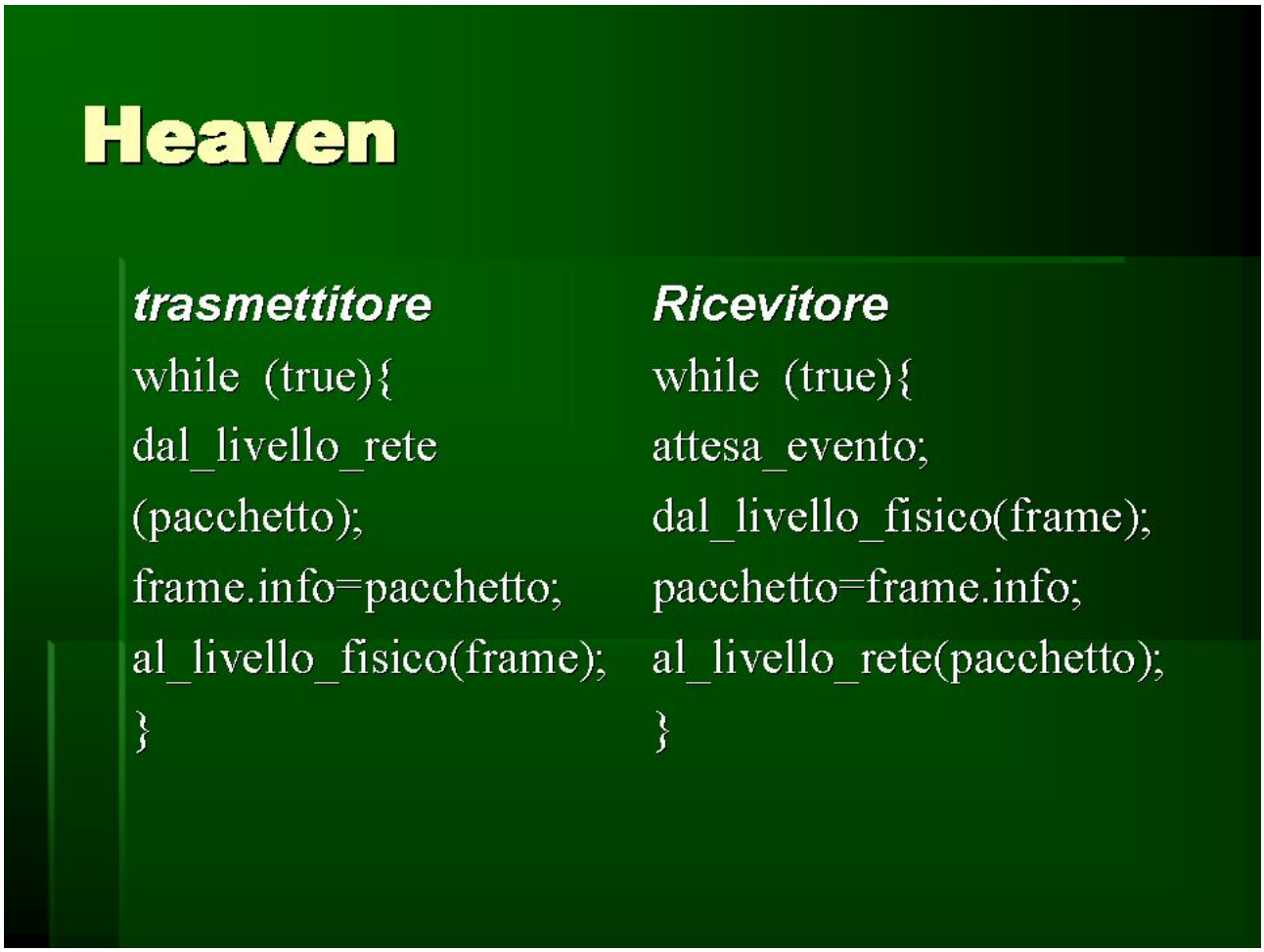
Il protocollo consiste di due procedure, relative rispettivamente al mittente e al destinatario.

Mittente (loop infinito):

```
1) attende un pacchetto dal livello di rete;
2) costruisce un frame dati;
3) passa il frame al livello fisico;
4) torna al passo 1).
```

Destinatario (loop infinito):

```
1) attende evento:
    * arriva frame da livello fisico:
2) estrae pacchetto;
3) lo passa al livello network;
4) torna ad 1).
```



Questo protocollo, per canale simplex, è molto semplice ed è basato sulle ipotesi (non realistiche) che:

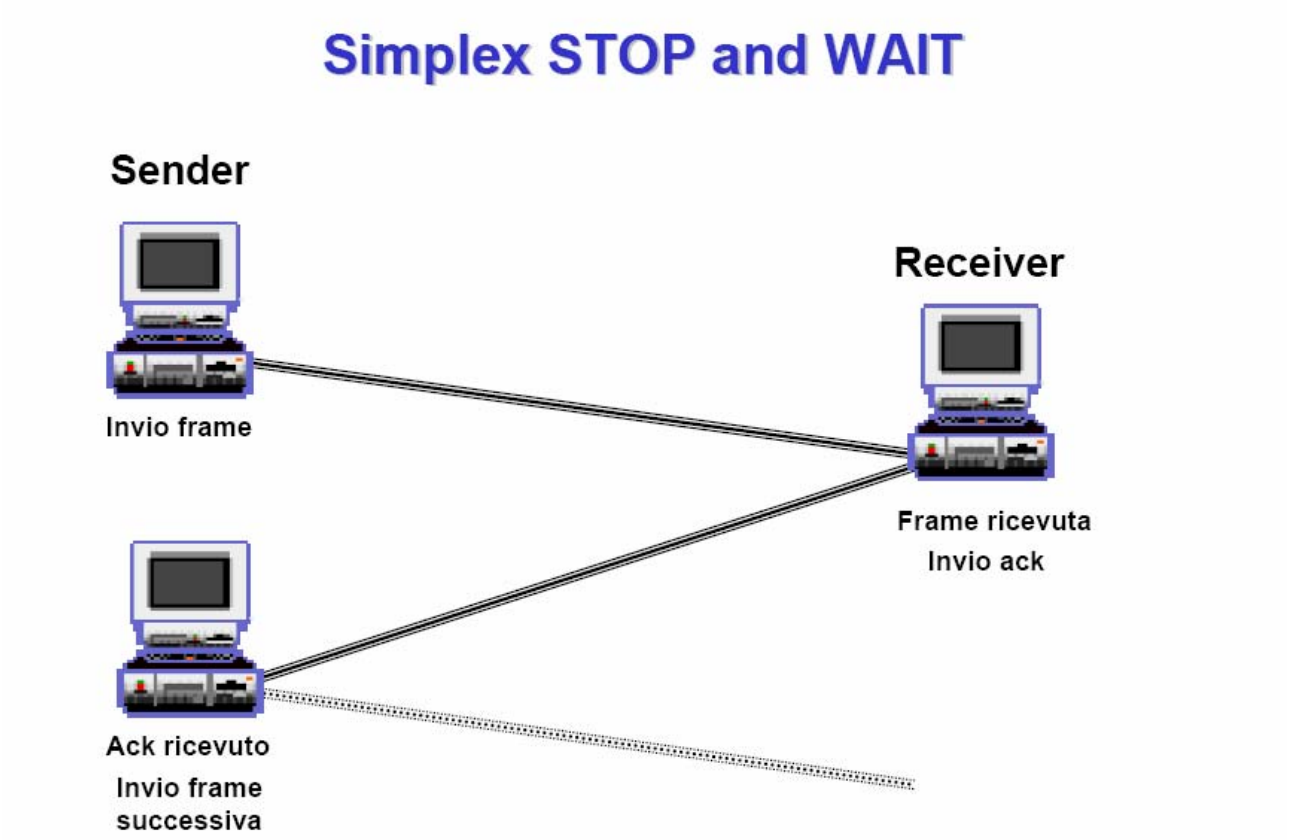
- i frame dati vengono trasmessi in una sola direzione;
- le peer entity di livello network sono sempre pronte (non devono mai attendere per inviare o ricevere al/dal livello data link) quindi il livello di linea del trasmettitore non si deve preoccupare di controllare se il livello di linea del ricevitore sia pronto a ricevere i dati;
- si ignora il tempo di elaborazione del livello data link quindi si presuppone che non ci siano ritardi introdotti dalla necessità del data link del ricevitore di elaborare ancora i pacchetti ricevuti precedentemente;
- c'è spazio infinito per il buffering nel ricevitore, cioè questo ha un'area di memoria in ingresso in cui si possono memorizzare i pacchetti procedenti dal trasmettitore in numero indefinito, per cui il trasmettitore può inviare

pacchetti senza remore poiché queti troveranno comunque posto nel buffer di ingresso del ricevitore;

- il canale fisico non genera mai errori per cui i pacchetti giungeranno sempre correttamente al computer destinatario.

Simplex stop and wait

Nel caso che il trasmettitore ed il ricevitore lavorino a velocità diverse il protocollo precedente non funziona. Il trasmettitore non può inviare un nuovo frame se il ricevitore non è riuscito ancora ad inviare al suo strato di rete il pacchetto contenuto nel frame precedente.



La soluzione più semplice è che il trasmettitore, dopo aver inviato un frame esegue un certo numero calcolato a priori di cicli di attesa per dare il tempo al ricevitore di completare il trattamento del frame che gli ha inviato precedentemente. Il problema è che risulta difficile calcolare a priori il ritardo nell'elaborazione del pacchetto da parte del ricevitore e poi la rete risulterebbe rallentata complessivamente dalla

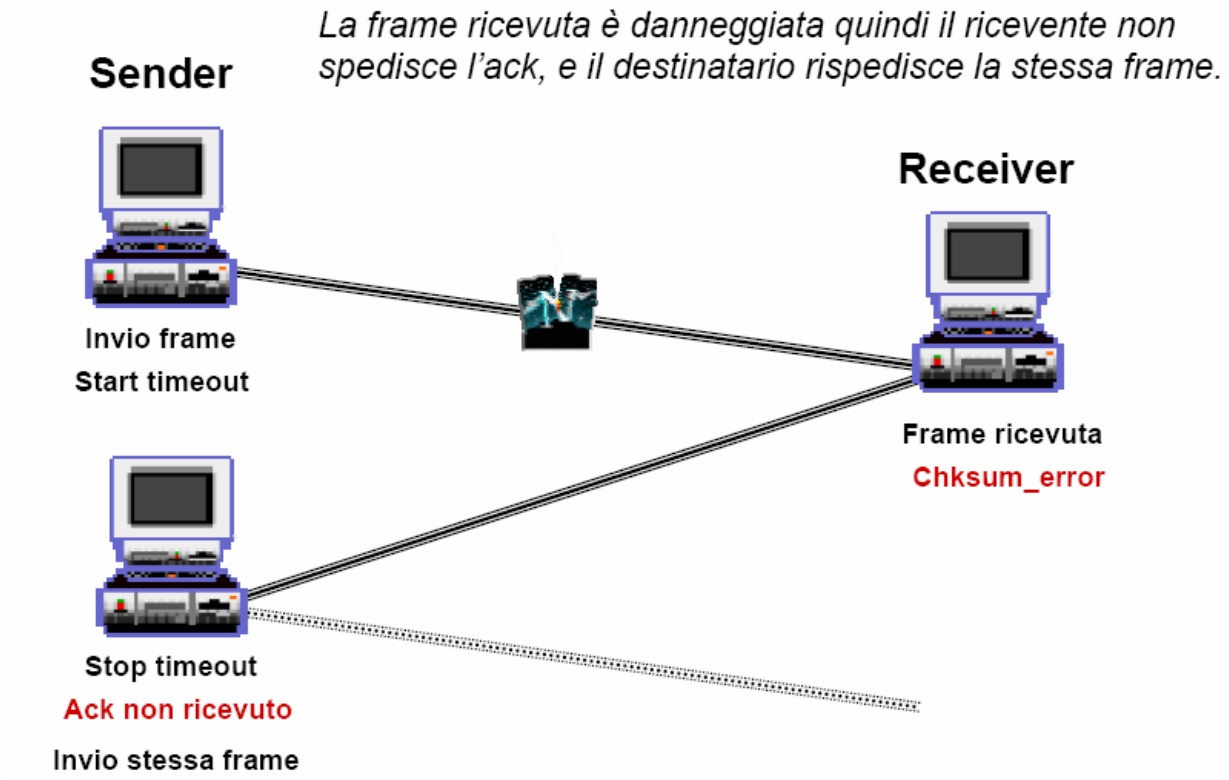
velocità del nodo più lento, poiché con questo metodo a priori tutti gli altri computer dovrebbero cadenzare la propria velocità di trasmissione sul computer più lento.

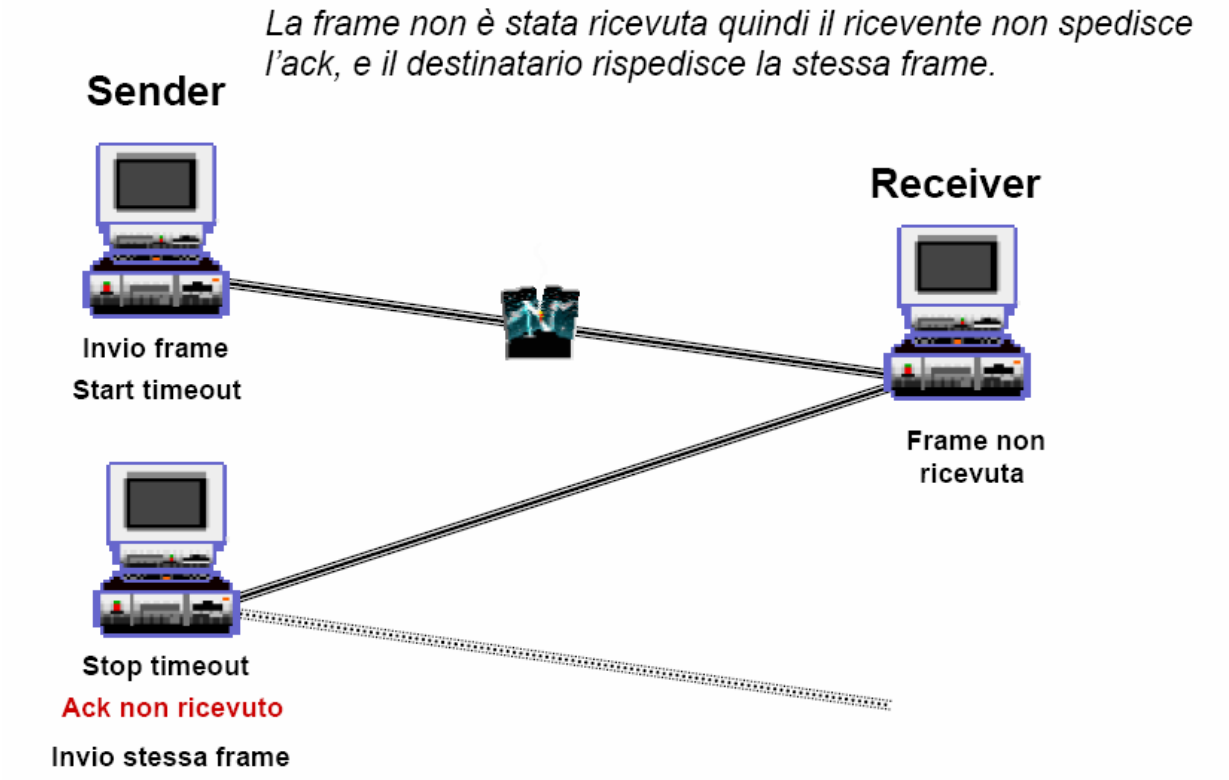
Una soluzione migliore è il protocollo stop and wait in cui il trasmettitore, dopo aver inviato un frame, attende un frame di riscontro dal ricevitore che gli da, in questo modo, via libera per l'invio di un nuovo frame.

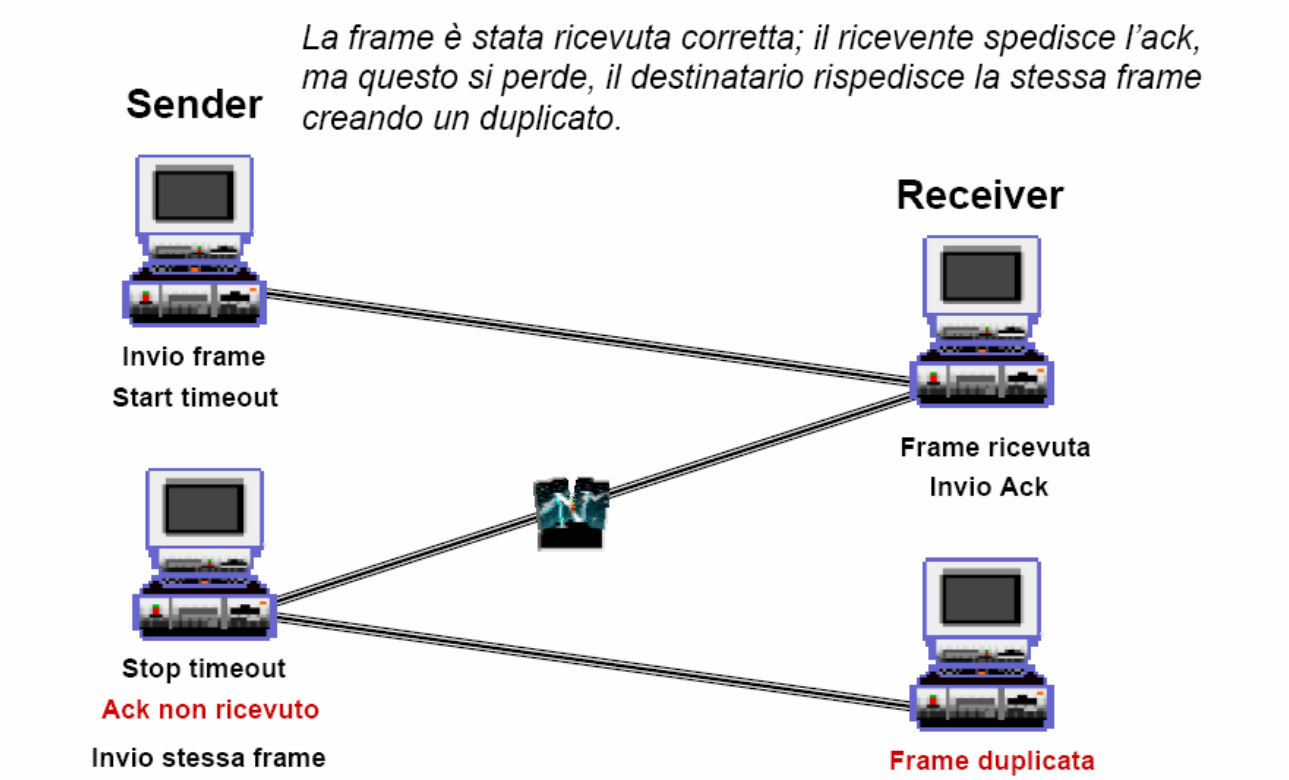
Protocolli a controllo e ritrasmissione

I protocolli visti non prevedono la gestione di eventuali errori nella trasmissione. Una prima soluzione è quella di modificare il protocollo stop and wait ed introdurre nel frame dei byte di checksum. Il ricevitore, se riscontra, tramite il confronto dei checksum, l'assenza di errori, invia al trasmettitore un messaggio di accettazione. Se vi è stato un errore il messaggio di acknowledged non viene inviato. In tal caso il trasmettitore invia di nuovo il frame precedente. Il problema può sorgere dal fatto

che gli errori di trasmissione riguardino proprio il messaggio ACK del ricevitore che potrebbe andare perso.

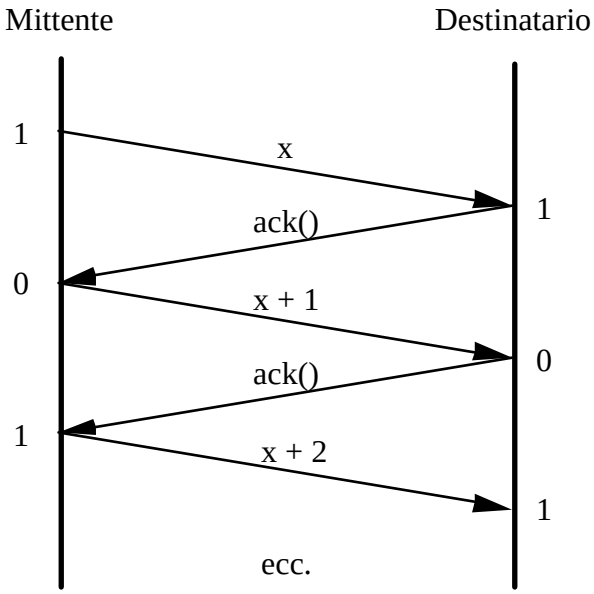




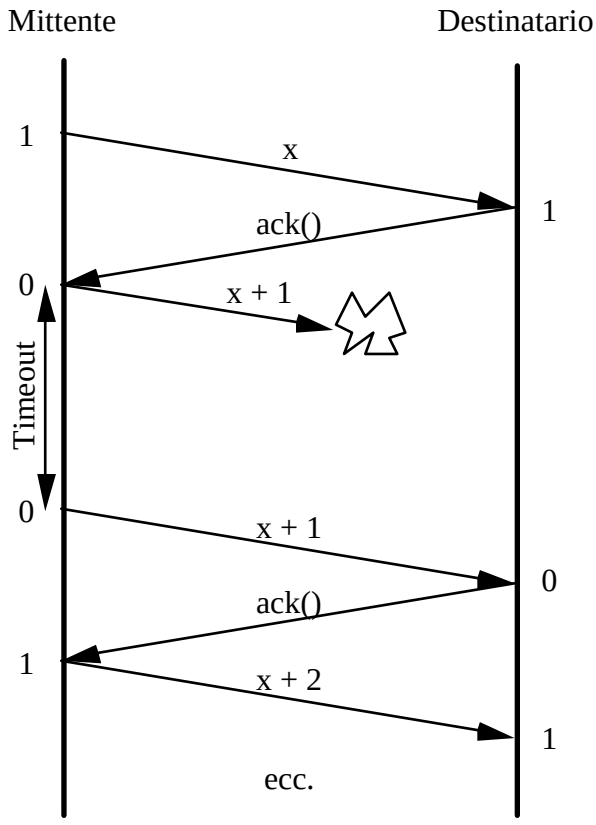


In tal caso il trasmettitore invia una seconda volta il frame pensando che il precedente sia giunto danneggiato. Il ricevitore lo interpreterebbe come un nuovo frame mentre, in realtà, si tratta del precedente frame che è stato ritrasmesso. Nei protocolli PAR (positive Acknowledgment with retransmission) il problema viene risolto poiché alla struttura del frame viene aggiunto un campo che segnala se il frame inviato è quello originale o una copia inviata dal trasmettitore perché ha ritenuto che l'originale non sia stato ricevuto correttamente. Notiamo che basta un bit per il numero di sequenza, poiché l'unica ambiguità in ricezione è tra un frame ed il suo immediato successore: infatti, fino a che un frame non viene confermato, è sempre lui ad essere ritrasmesso, altrimenti è il suo successore. Il mittente trasmette i frame dati alternando zero ed uno nel campo seq; passa a trasmettere il prossimo frame solo quando riceve l'ack di quello precedente.

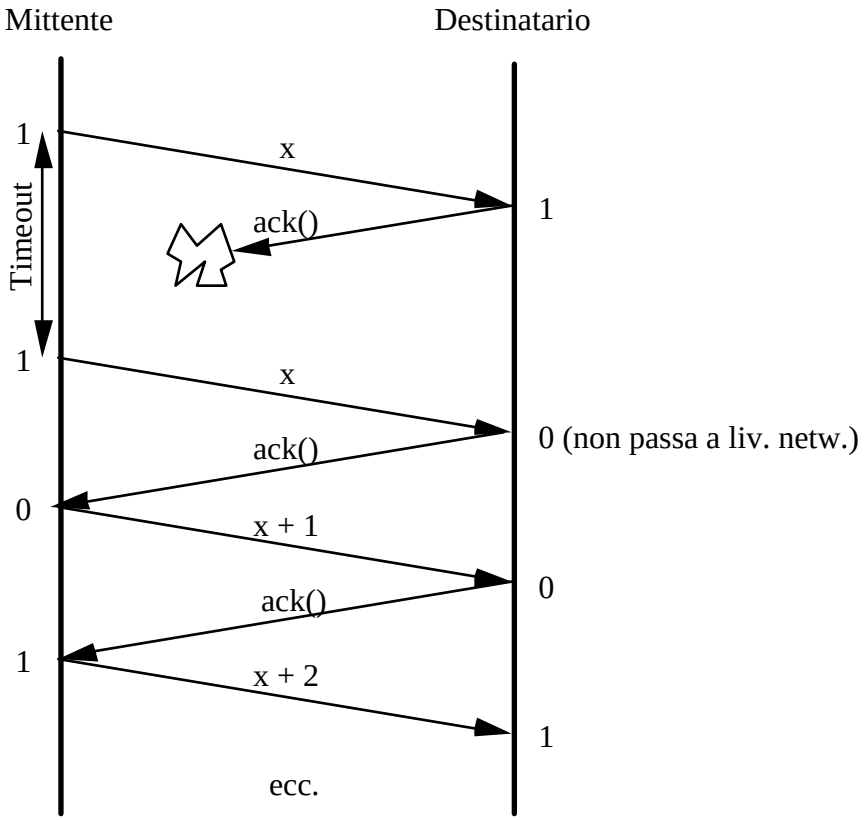
Vediamo alcuni esempi. In assenza di errori il funzionamento è il seguente:



Nel caso in cui un frame dati si perda o si danneggi, la situazione è la seguente:

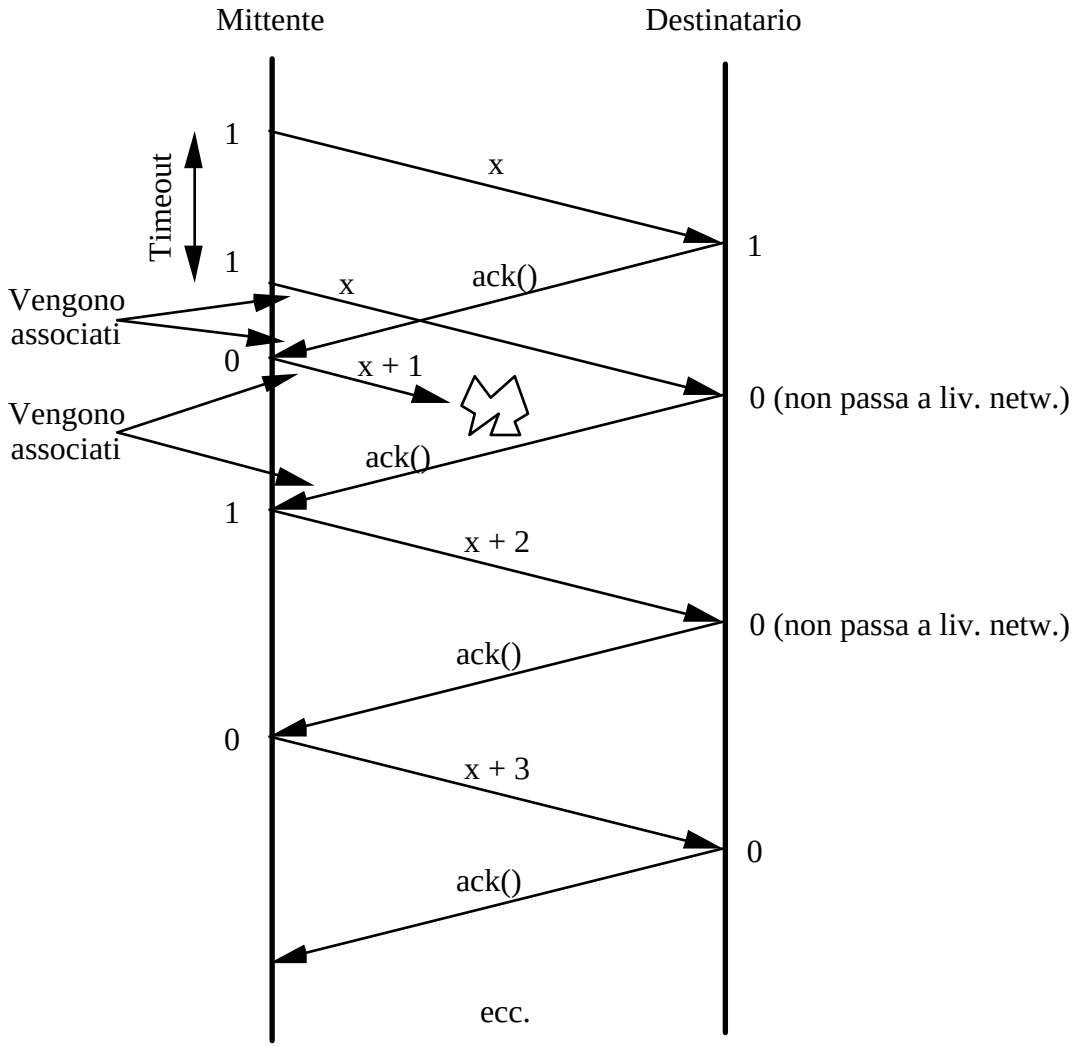


Nel caso in cui invece si perda o si danneggi un frame di ack, la situazione è la seguente:



Il trasmettitore ha inviato un pacchetto copia poiché non ha ricevuto l'ACK. Il ricevitore, verificando che è una copia del pacchetto che ha già ricevuto lo scarta e non lo passa al livello superiore

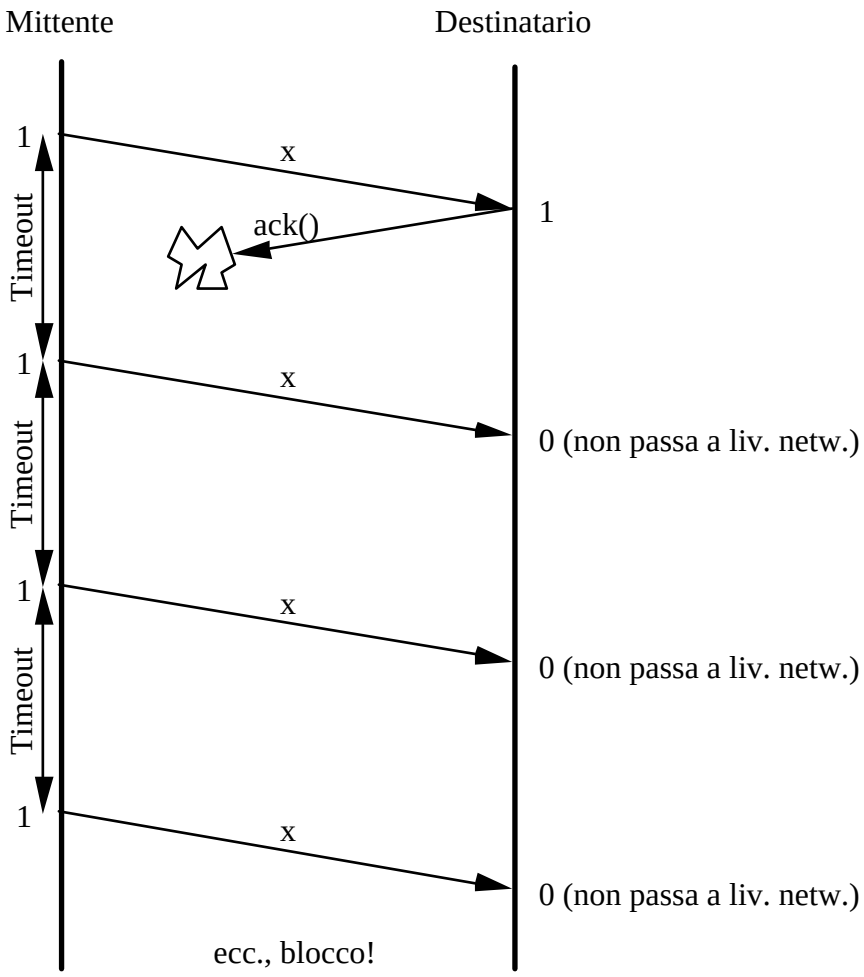
Un problema importante è legato, come abbiamo già accennato, alla lunghezza del timeout. Se esso è troppo breve, può succedere questo:



Come si nota dal diagramma il pacchetto x viene rinviato poiché un timeout troppo breve ha fatto ritenere erroneamente al trasmettitore che la sua prima copia non fosse giunta a destinazione. Il ricevitore, intanto, scarta la seconda copia di x e invia un secondo ACK di riscontro. Finalmente giunge il primo ACK al trasmettitore il quale invia il frame x+1. questo si perde in linea e non raggiungerà mai al ricevitore, ma intanto giunge il secondo ack relativo alla copia del pacchetto x precedente e il trasmettitore ritiene erroneamente che sia il riscontro del pacchetto x+1. il trasmettitore prosegue tranquillamente inviando i frame successivi e non saprà mai che il frame x+1 non è mai giunto a destinazione.

Nell'esempio, i frame dati (x+1) e (x+2) si perdono (nel senso che non vengono consegnati al livello network) e nessuno se ne accorge. D'altronde, se cambiamo il protocollo mandando un frame di ack solo per i frame dati che non sono duplicati, ci si

può bloccare come nell'esempio seguente: infatti l'ack si perde e il trasmettitore invia una copia

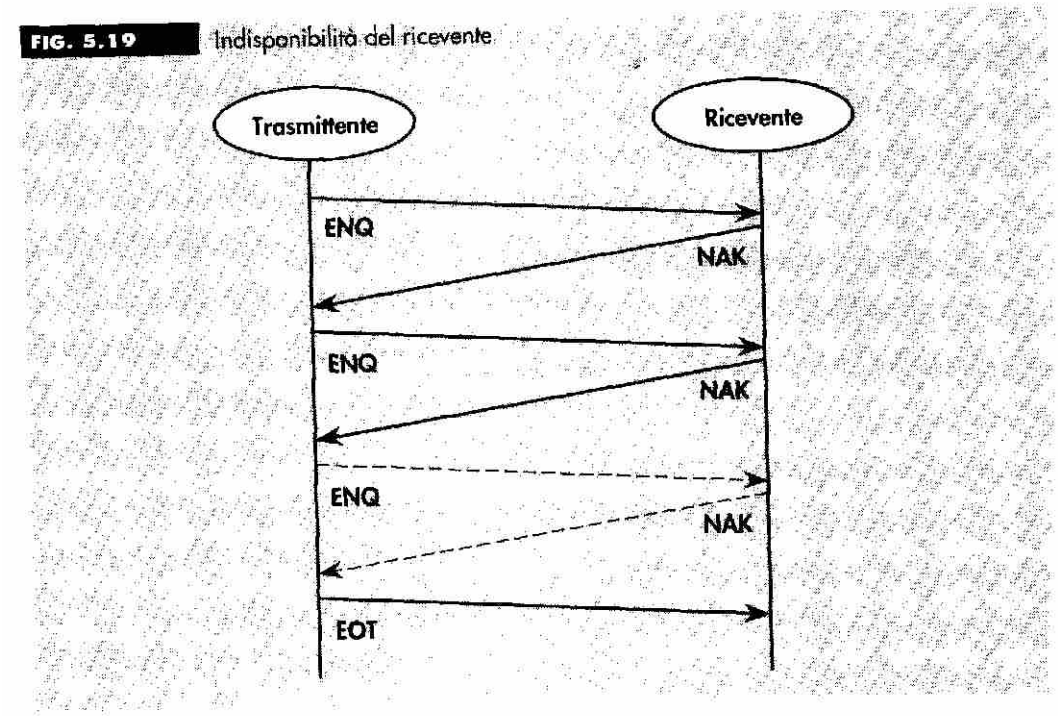


Il ricevitore non invia l'ack perché capisce che è una copia. Il trasmettitore una nuova copia perché non ha ricevuto l'ack e così via all'infinito.

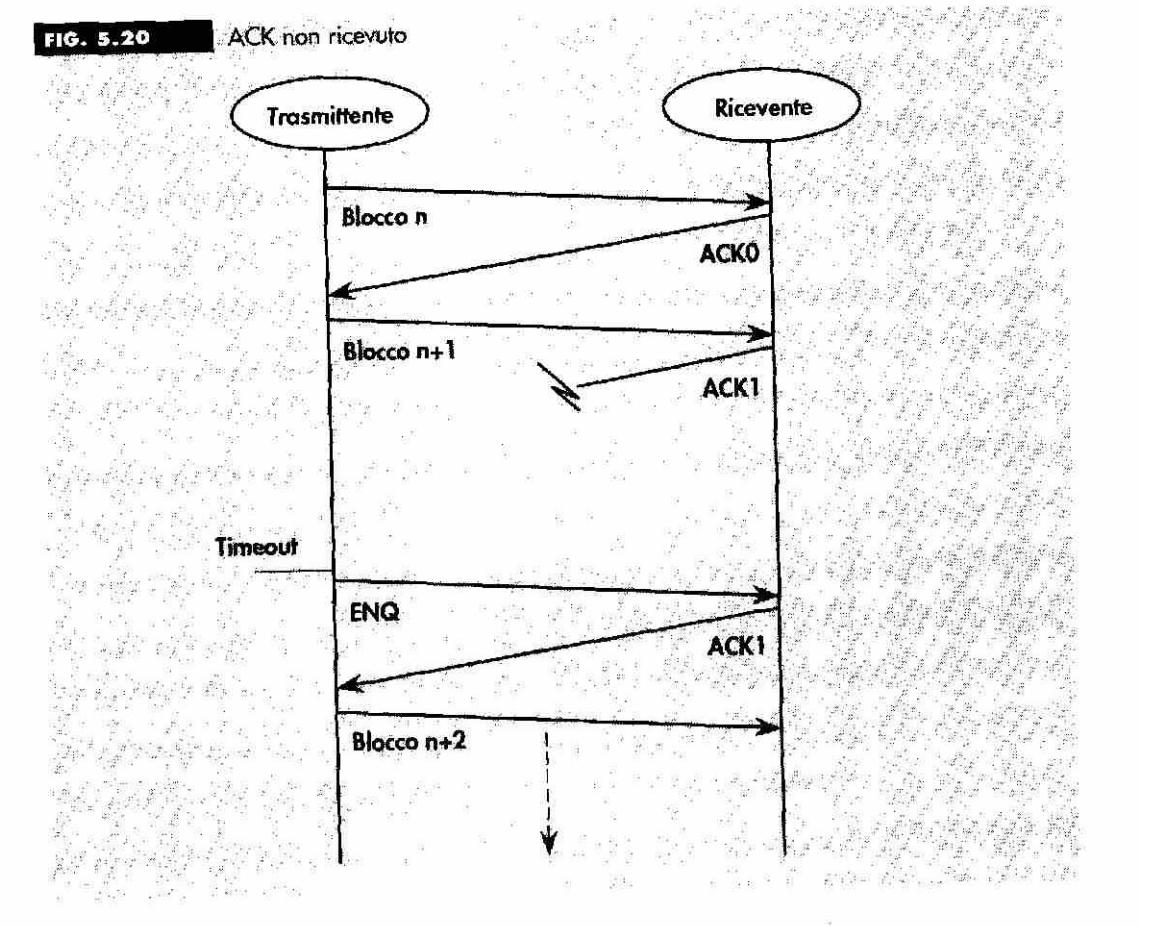
Protocollo BSC (Binary Synchronous Communication)

Vediamo un altro esempio di regole di sincronizzazione. Il trasmettitore cerca di stabilire una comunicazione con il ricevitore inviandogli un messaggio di controllo ENQ (Enquiry).

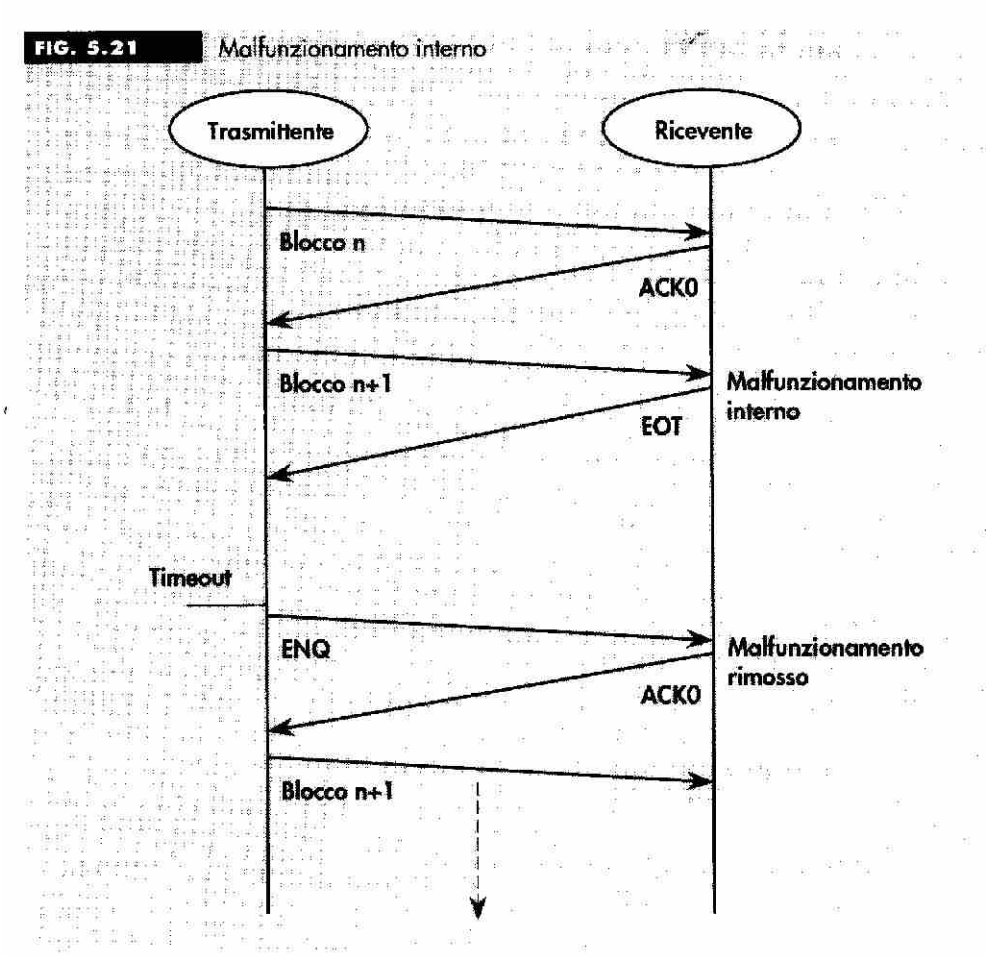
Se il ricevitore non è disponibile, risponde con un messaggio Not Acknowledge NAK. Il trasmettitore riprova più volte ad inviare un ENQ. Dopo un certo lasso di tempo di timeout, se non è riuscito a collegarsi con il ricevitore, il trasmettitore interrompe il collegamento.



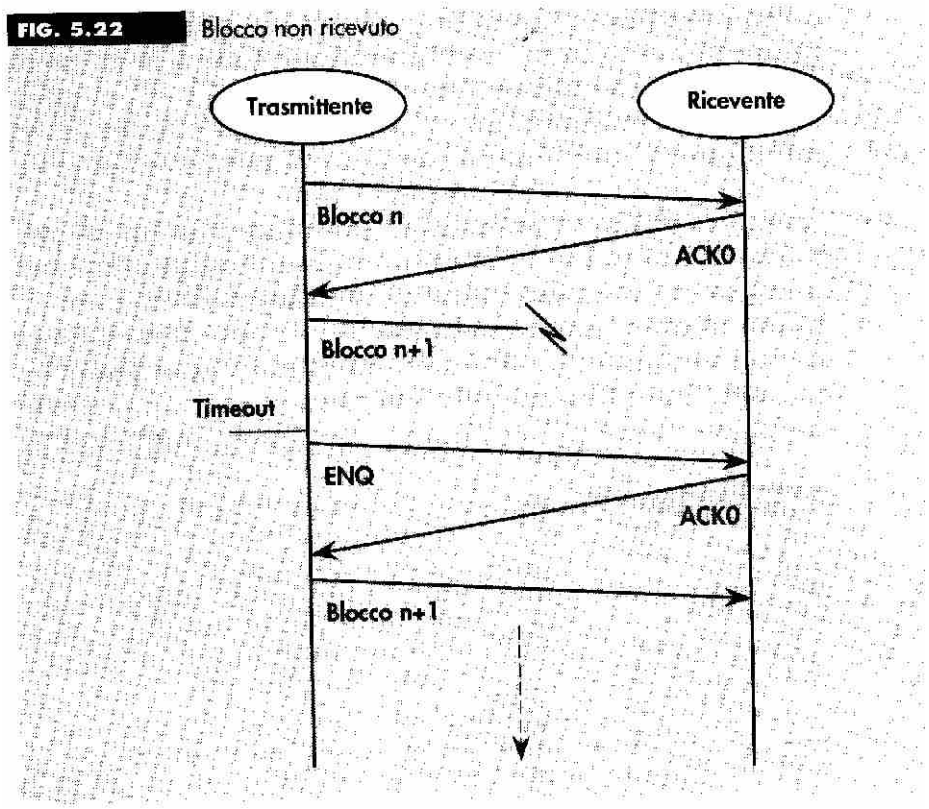
Se il ricevitore, invece, è disponibile al dialogo, risponde all'Enquiry mediante un messaggio di Acknowledge ACK. Nel BSC sono previsti due messaggi di riscontro ACK0 e ACK1. Questi due messaggi vengono inviati alternativamente e questo meccanismo consente di riscontrare la presenza di problemi di trasmissione. Supponiamo ad esempio che il trasmettitore invii il blocco numero N e che il ricevitore abbia risposto con il messaggio ACK0. Ora viene trasmesso il blocco N+1 a cui il ricevitore dovrebbe rispondere con il messaggio ACK1. Per un errore di trasmissione tale messaggio non viene ricevuto dal trasmettitore. Il trasmettitore attende il messaggio di riscontro per un tempo di timeout e poi invia un messaggio di ENQ. Il ricevitore reagisce inviando di nuovo l'ultimo messaggio di riscontro che aveva inviato, nel nostro caso ACK1. Vedendo che è stato spedito il messaggio di riscontro ACK1 il trasmettitore capisce che il blocco N+1 era stato ricevuto correttamente da parte del ricevitore, e che l'errore di trasmissione si riferisce soltanto al pacchetto di riscontro, per cui provvede a trasmettere normalmente il blocco N+2



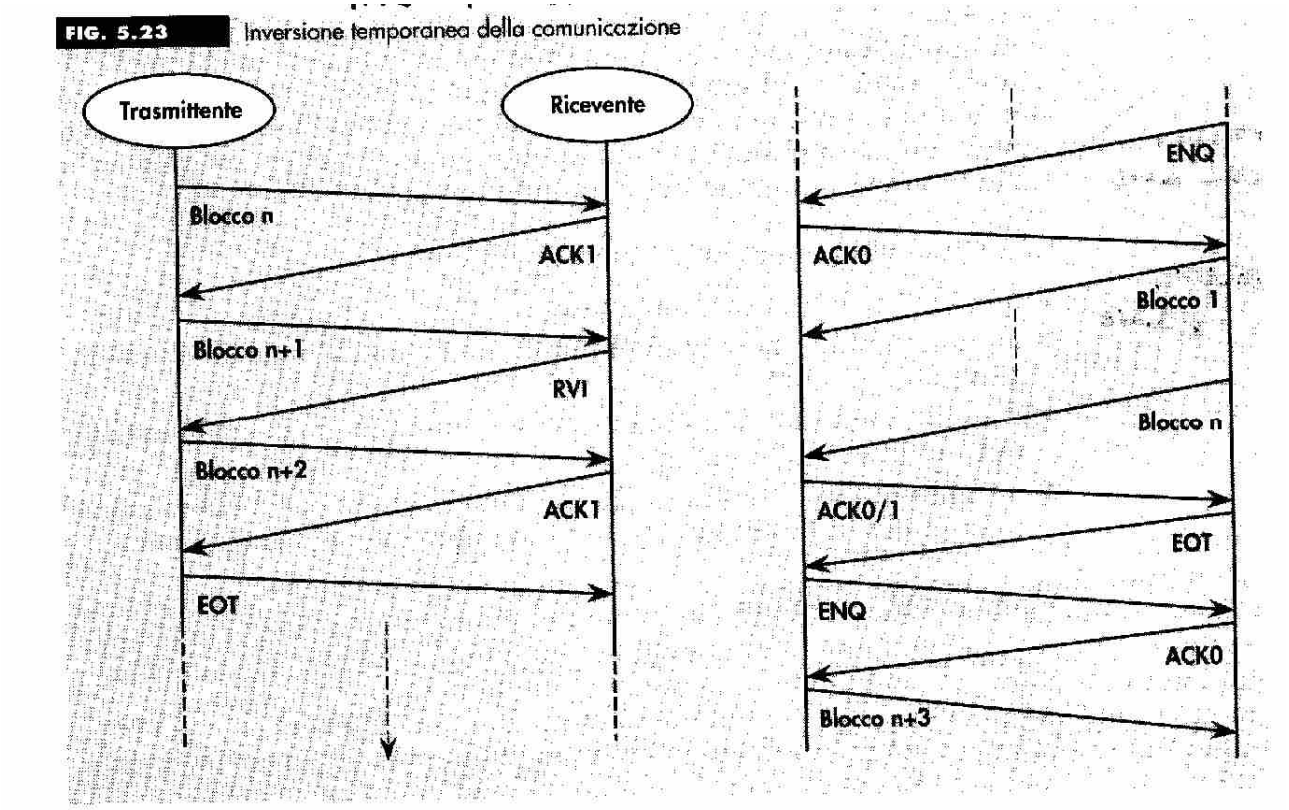
Supponiamo ora che, a causa di un malfunzionamento, il ricevitore non riceva il blocco N+1 e tenti di scollegarsi per un malfunzionamento interno, inviando il messaggio EOT (End of transmission). In tal caso, passato un tempo di timeout, il trasmettitore tenta di nuovo di ricollegarsi al ricevitore con un messaggio di ENQ. Il ricevitore, se di nuovo disponibile, invia l'ultimo messaggio di riscontro che aveva inviato, nell'esempio ACK0. Da ciò il trasmettitore capisce che il blocco N+1 non è stato ricevuto e lo ritrasmette



Un funzionamento analogo si ha nel caso che il blocco N+1 venga perso per un disturbo sulla linea



il diagramma seguente mostra, invece, come viene gestita l'inversione del senso di comunicazione. Il ricevitore, per diventare il nuovo trasmettitore invia il messaggio RVI (reverse Interrupt). Il trasmettitore termina il suo invio di dati mediante il messaggio di EOT. Il ricevitore diventa il nuovo trasmettitore ed invia alla prima stazione un segnale di Enquiry. Da questo momento in poi la procedura è identica.



Protocolli piggybacking

Nei casi precedenti i frame dati viaggiano in una sola direzione e i frame di ack nella direzione contraria, quindi esistono dei frame che viaggiano in entrambe le direzioni. Dunque, volendo stabilire una comunicazione dati bidirezionale è necessario disporre di due circuiti, il che ovviamente è uno spreco di risorse. Un'idea migliore è usare un solo circuito, nel quale far convivere tutte le esigenze.

Supponendo che la comunicazione sia fra A e B, si avrà che:

- nella direzione da A a B viaggiano i frame dati inviati da A a B e i frame di ack inviati da A a B (in risposta ai frame dati inviati da B ad A);
- nella direzione da B a A viaggiano i frame dati inviati da B a A e i frame di ack inviati da B a A (in risposta ai frame dati inviati da A ad B);
- il campo ***kind*** serve a distinguere fra i due tipi di frame, dati e di ack, che viaggiano nella stessa direzione.

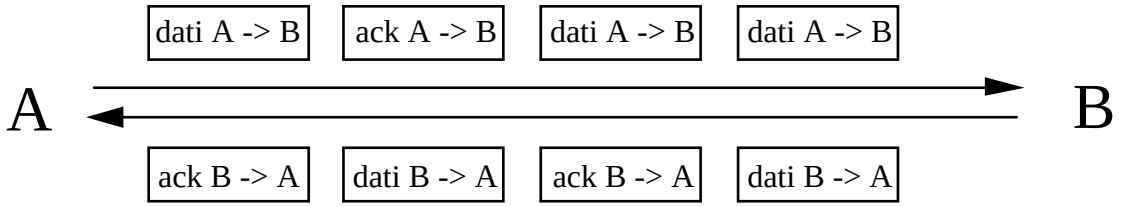


Figura 3-7: Comunicazione dati bidirezionale su unico canale

Però c'è un'idea ancora migliore: se, quando si deve inviare un ack da B ad A, si aspetta un pò fin che è pronto un frame dati che B deve inviare ad A, si può "fare autostop" e mettere dentro tale frame dati anche le informazioni relative all'ack in questione. Questa tecnica si chiama ***piggybacking*** (letteralmente, portare a spalle).

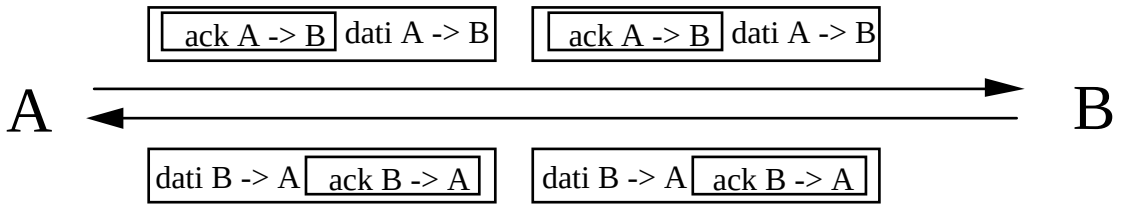


Figura 3-8: Piggybacking

Il campo ***ack*** serve proprio a questo scopo, infatti è il campo in cui viene trasportato, se c'è, un ack.

Questa tecnica consente un notevole risparmio di:

- banda utilizzata;
- uso di cpu.

Infatti, con questa tecnica le informazioni di ack non richiedono la costruzione di un apposito frame (e quindi il tempo necessario alla creazione ed al riempimento della struttura, al calcolo del checksum, ecc.) né la sua trasmissione (e quindi l'uso di banda).

Però c'è un aspetto da non trascurare: per quanto si può aspettare un frame su cui trasportare un ack che è pronto e deve essere inviato? Non troppo, perché se l'ack non arriva in tempo il mittente ritrasmetterà il frame anche se ciò non è necessario. Dunque si stabilisce un limite al tempo di attesa di un frame sul quale trasportare l'ack; trascorso tale tempo si crea un frame apposito nel quale si mette l'ack.

Protocolli a finestra scorrevole

I protocolli che vedremo ora appartengono alla classe dei ***protocolli sliding window*** (finestra scorrevole):

- ogni frame inviato ha un numero di sequenza, da 0 a 2n-1; ad ogni istante il mittente mantiene una finestra scorrevole sugli indici dei frame, e solo quelli entro la finestra possono essere trasmessi. I numeri di sequenza entro la finestra rappresentano frame da spedire o spediti, ma non ancora confermati:
    - quando arriva dal livello network un pacchetto, un nuovo indice entra nella finestra;
    - quando arriva un ack, il corrispondente indice esce dalla finestra;
    - i frame dentro la finestra devono essere mantenuti in memoria per la possibile ritrasmissione; se il buffer è pieno, il livello data link deve costringere il livello network a sospendere la consegna di pacchetti;
- analogamente, il destinatario mantiene una finestra corrispondente agli indici dei frame che possono essere accettati:
    - se arriva un frame il cui indice è fuori dalla finestra:
        - il frame viene scartato (e non si invia il relativo ack);
    - se arriva un frame il cui indice è entro la finestra:
        - il frame viene accettato;

        - viene spedito il relativo ack;
        - la finestra viene spostata in avanti;
    - si noti che la finestra del destinatario rimane sempre della stessa dimensione, e se essa è pari a 1 il livello accetta i frame solo nell'ordine giusto (ma per dimensioni maggiori di 1 questo non è più detto).
- le finestre di mittente e destinatario non devono necessariamente avere uguali dimensioni, né uguali limiti inferiori o superiori.

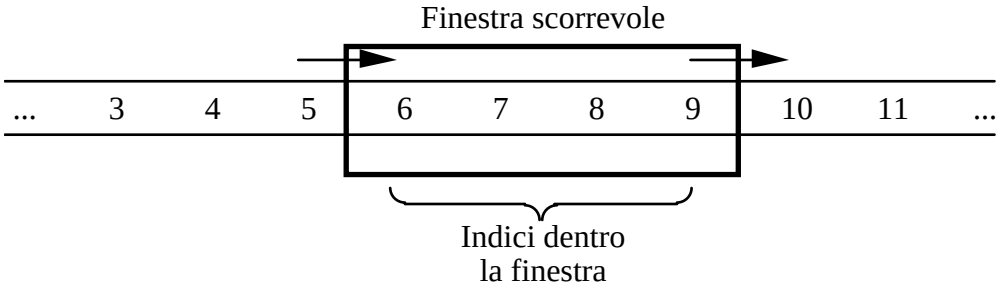


Figura 3-9: Finestra scorrevole sugli indici dei frame

Protocollo a finestra scorrevole di un bit

In questo protocollo sia mittente che destinatario usano una finestra scorrevole di dimensione uno. Di fatto questo è un protocollo stop-and-wait.

Il funzionamento, molto semplice, è il seguente:

- il mittente, quando invia un frame, fa partire un timer:
    - se prima che scada il timer arriva un ack con lo stesso numero di sequenza del frame che si sta cercando di trasmettere, si avanza la finestra e si passa a trasmettere il frame successivo;
    - se arriva un ack diverso o scade il timer, si ritrasmette il frame;
- il destinatario invece:
    - quando arriva un frame corretto, senza errori, invia un ack col corrispondente numero di sequenza;
    - se il frame non è un duplicato lo passa al livello network e avanza la finestra.

Protocolli go-back-n e selective repeat

Se il tempo di andata e ritorno del segnale (***round-trip time***) è alto, come ad esempio nel caso dei canali satellitari nei quali è tipicamente pari a 500 + 500 msec, c'è un enorme inefficienza coi protocolli stop-and-wait, perché si sta quasi sempre fermi aspettando l'ack.

Per migliorare le cose, si può consentire l'invio di un certo numero di frame anche senza aver ricevuto l'ack del primo. Questa tecnica va sotto il nome di ***pipelining***.

Ciò però pone un serio problema, perché se un frame nel mezzo della sequenza si rovina molti altri frame vengono spediti prima che il mittente sappia che qualcosa è andato storto.

Il primo approccio al problema è quello del protocollo ***go-back-n***:

- se arriva un frame danneggiato o con un numero di sequenza non progressivo, il destinatario ignora tale frame e tutti i successivi, non inviando i relativi ack.
- il mittente ad un certo punto va in time-out sul frame sbagliato, e poi su tutti quelli successivi (scartati dal destinatario), e quindi provvede a ritrasmettere la sequenza di frame che inizia con quello per il quale si è verificato il time-out.

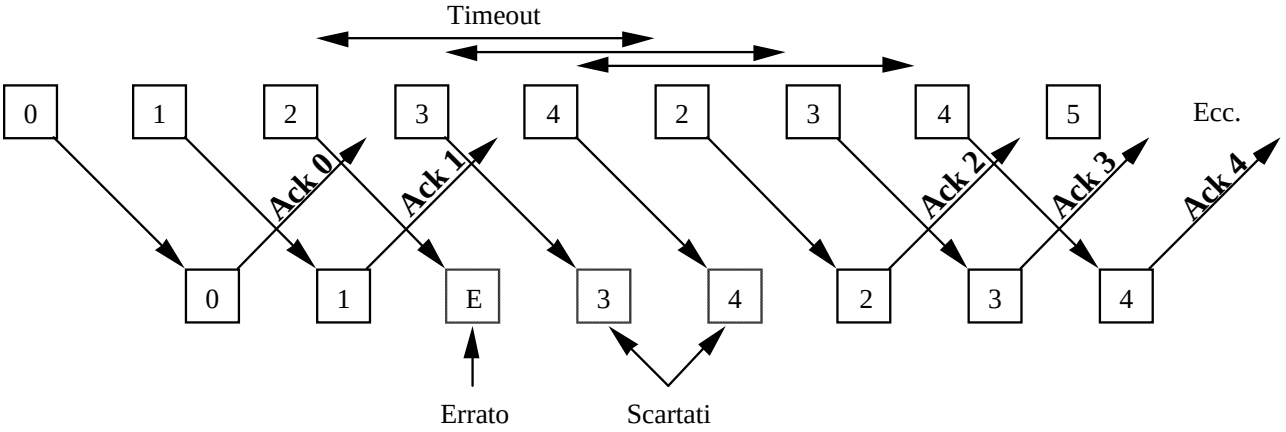


Figura 3-10: Funzionamento del protocollo go-back-n

Si noti che il mittente deve mantenere in un apposito buffer tutti i frame non confermati per poterli eventualmente ritrasmettere. Se il buffer si riempie, il mittente deve bloccare il livello network fino a che non si ricrea dello spazio. Inoltre, vi è spreco di banda se il tasso d'errore è alto e/o il time-out è lungo.

Il secondo approccio è più efficiente, ed è chiamato ***selective repeat***:

- il destinatario mantiene nel suo buffer tutti i frame ricevuti successivamente ad un eventuale frame rovinato; non appena questo arriva nuovamente (senza errori), esso e tutti i successivi frame contigui che il destinatario ha mantenuto nel buffer vengono consegnati al livello network;
- per ogni frame arrivato bene, il destinatario invia un ack col numero più alto della sequenza completa arrivata fino a quel momento;
- quando si verifica un timeout, il mittente rispedisce il frame corrispondente.

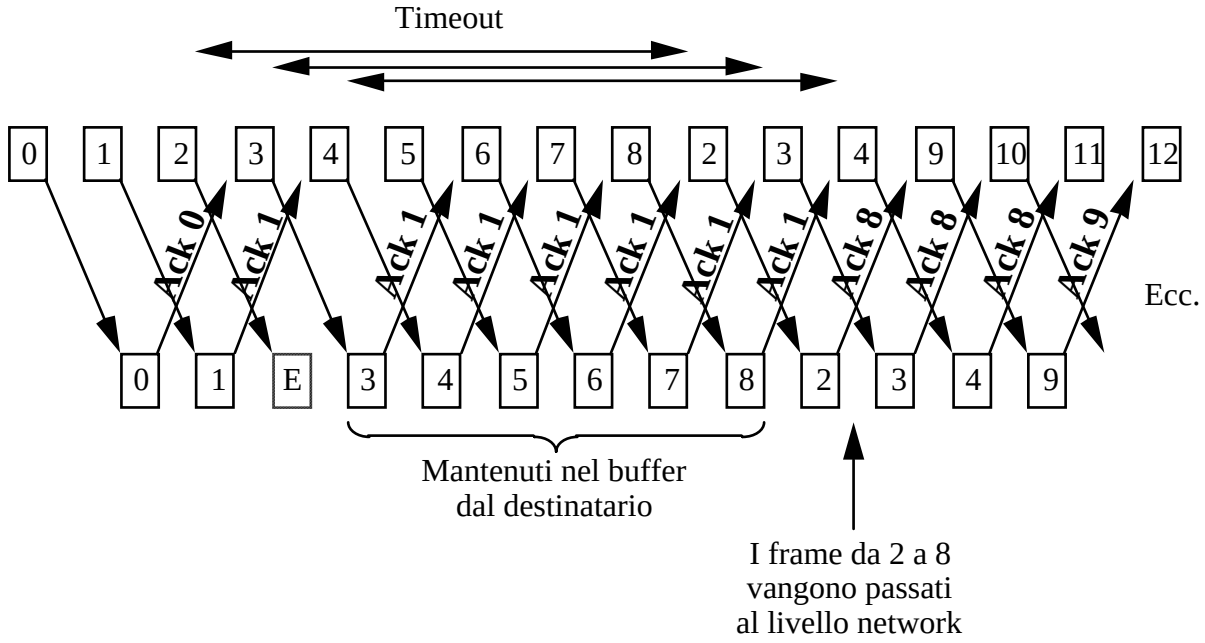


Figura 3-11: Funzionamento del protocollo selective repeat

Alcune considerazioni a proposito del protocollo selective repeat:

- mittente e destinatario devono entrambi gestire un buffer per mantenervi i frame:
    - non confermati (mittente);
    - successivi ad un errore (destinatario);
- vi è un basso spreco di banda, che si si può ulteriormente diminuire mandando un ***NACK*** (***Negative ACKnowledgement***) quando:
    - arriva un frame danneggiato;
    - arriva un frame diverso da quello atteso (ciò può indicare l'avvenuta perdita del frame precedente).

Infine, si noti che per entrambi i precedenti protocolli:

- è necessaria la gestione di timer multipli (uno per ogni frame inviato e non confermato);
- il ricevente, per inviare gli ack, usa il piggybacking se possibile, altrimenti invia un apposito frame.