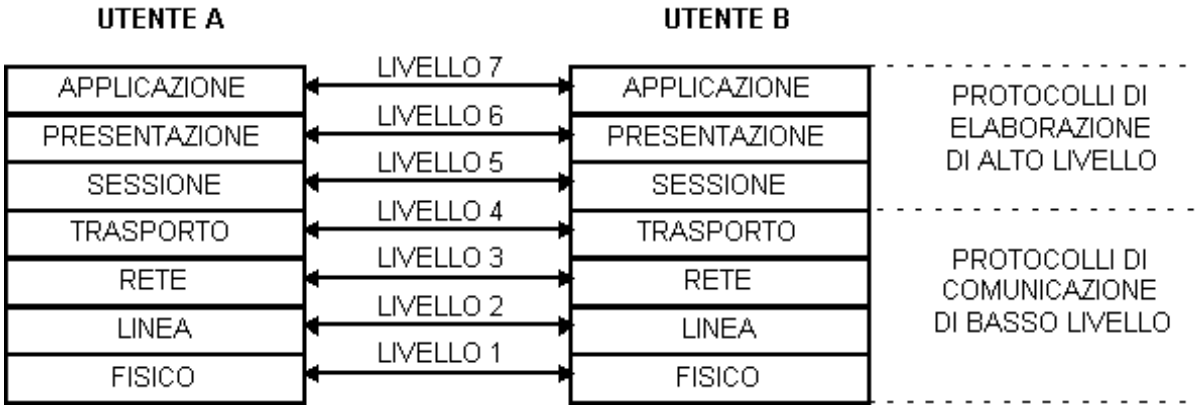
Appunti di Reti di Telecomunicazioni
Capitolo 5 - Protocolli di linea (parte I)

INTRODUZIONE

Consideriamo lo schema generale dei 7 livelli del **modello ISO-OSI**:



Schema logico di due utenti (A e B) connessi tramite il modello OSI

L'insieme dei 7 livelli (layer) garantisce tutte le funzioni necessarie alla rete comunicativa tra sistemi, nonché una gamma molto ampia di funzioni opzionali (come ad esempio la compressione e la cifratura dei dati).

Il livello più basso è quello *fisico*, che quindi ha a che fare con la trasmissione di bit "grezzi" su un canale di comunicazione: esso riceve i pacchetti (**frame**) di bit dal livello superiore e trasmette i

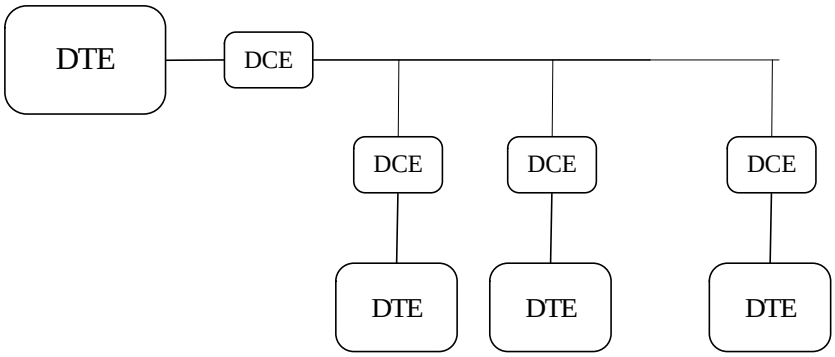
bit, uno per uno[1], sul mezzo fisico usato per il collegamento alla rete. E' ovvio che tale livello abbia uno strettissimo legame con le caratteristiche meccaniche, elettriche e procedurali delle ***interfacce di rete*** (componenti che connettono l'elaboratore al mezzo fisico) e con le caratteristiche del mezzo fisico.

Noi non ci occupiamo del livello fisico, ma concentriamo la nostra analisi sul livello superiore, detto **livello di protocollo di linea**. Esistono molti protocolli di linea; tutti quanti, con tecniche più o meno efficienti, hanno dei compiti precisi:

- devono provvedere alla *sincronizzazione* tra la stazione trasmittente e la stazione ricevente, sia prima della trasmissione sia, eventualmente, nel corso della stessa;
- devono dare una *struttura formale* ai dati scambiati, in modo da distinguere i dati dell'utente dalle informazioni di controllo dei dati stessi[2];
- devono provvedere a gestire il *flusso* e gli scambi di dati;
- devono controllare gli *errori di trasmissione* e provvedere alla loro correzione.

E' possibile definire alcune **categorie** di protocolli di linea. Quella più numerosa è basata sulla **relazione primario/secondario** (spesso chiamata anche **master/slave**). Questa relazione significa che, tra le due o più stazioni connesse ad un'unica linea, una stazione ha il ruolo di stazione primaria, che gestisce sostanzialmente l'accesso alla linea, mentre le altre sono secondarie, nel senso che si adeguano alle regole imposte dalla stazione master.

Questa situazione è tipica di una **rete multi-punto** del tipo seguente:



*Tipica configurazione **multi-punto**, in cui un'unica linea connette diversi DTE (cioè diversi calcolatori, ognuno dei quali è collegato alla linea tramite un dispositivo di interfaccia, ad esempio un modem, che si indica brevemente con DCE); uno di questi DTE, situato generalmente ad un estremo della linea, si comporta da MASTER, mentre gli altri sono SLAVE*

[1] Per semplicità, ci riferiremo sempre alla trasmissione dei bit uno alla volta sul canale di trasmissione. In realtà, ci sono tecniche di trasmissione (**modulazione numerica multilivello**) che consentono di aumentare la velocità di trasmissione, a parità di banda disponibile, trasmettendo i bit a gruppi. In pratica, al posto di considerare, per la trasmissione, solo due forme d'onda elementari, da associare una al bit 0 ed una al bit 1, si possono considerare più forme d'onda, da associare a gruppi di bit: se si scelgono 4 forme d'onda, ognuna corrisponderà a 2 bit (00,01,10,11); se si scelgono 8 forme d'onda, ciascuna corrisponderà a 3 bit (000,001,010,011,100,101,110,111) e così via.

[2] Le informazioni di controllo di cui si parla possono essere di vario tipo. Infatti, un generico pacchetto di bit presenta una serie di campi, ciascuno di opportuna lunghezza, contenenti una serie di informazioni. Possiamo sommariamente indicare i seguenti campi (che comunque differiscono da protocollo a protocollo): un campo iniziale, detto **flag**, che individua l'inizio del pacchetto; un campo contenente l'**indirizzo** del destinatario del pacchetto; un campo contenente un numero (**numero di sequenza**) identificativo del pacchetto; un campo contenente i **dati** veri e propri; un campo contenente i codici per il **controllo degli errori**. Se i pacchetti sono di lunghezza variabile, ci sarà anche in coda un flag necessario ad individuare la fine del pacchetto.

In una **rete multipunto** di questo tipo, è necessaria la presenza di "qualcuno" che stabilisca, sulla base di precise regole, quale stazione possa trasmettere in un determinato momento. Questo "qualcuno" è generalmente la stazione connessa ad un estremo della linea e prende il nome di **master**[1]. Le altre stazioni sono invece dette **slave** e possono comunicare (trasmettere o ricevere) solo dietro autorizzazione del master.

L'altra grande categoria dei protocolli di linea è quella dei **protocolli alla pari**: in questo caso, le stazioni hanno uguali diritti alla trasmissione e quindi non esiste il ruolo di stazione primaria.

Esiste una ulteriore categoria che è quella dei **protocolli ibridi**, che prevedono sostanzialmente entrambe le opzioni citate prima: una stessa linea può essere usata, alternativamente, con un protocollo alla pari oppure con un protocollo master/slave.

Nella figura seguente sono elencati alcuni tra i protocolli più diffusi, alcuni dei quali usati su **WAN** (*reti geografiche*) altri su **LAN** (*reti locali*):

Protocolli con relazione primario/secondario	
Polling	S/S Stop and Wait BSC SDLC HDLC (SNRM) Continuous ARQ
Non Polling	RTS/CTS X-on/X-off TDMA
Protocolli ibridi	
HDLC (SABM)	LAP LAPB LAPD LAPX LLC IEEE 802.2
Protocolli alla pari	
Non priorità	TDM Register Insertion ALOHA CSMA/CD (IEEE 802.3)
Priorità	Token passing (802.5) Token bus (802.4) Priority slot Slotted ALOHA CSMA collision free

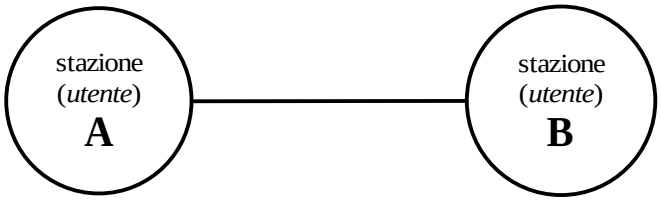
Nei paragrafi che seguiranno prenderemo in esame alcuni di questi protocolli. In particolare, la prima parte dei nostri discorsi sarà incentrata sui **protocolli di linea per connessioni punto-a-punto**, ossia relativi solo alla comunicazione tra due generiche stazioni A e B connesse tramite la

[1] Il master deve dunque svolgere un lavoro ulteriore rispetto ai normali compiti applicativi e puramente trasmissivi: esso deve dedicare risorse per gestire in modo opportuno l'assegnazione del diritto a trasmettere sulla linea.

rete in esame, senza addentrarci nello studio delle tecniche con cui una stazione acquisisce il diritto a trasmettere. Queste tecniche saranno invece descritte nei discorsi successivi.

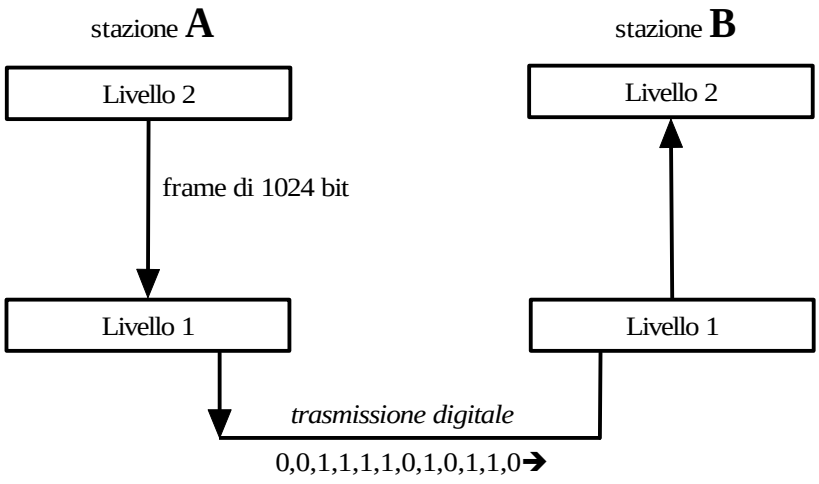
CONCETTI GENERALI SULLE COMUNICAZIONI PUNTO-A-PUNTO

Consideriamo due generiche stazioni A e B connesse tramite una linea di comunicazione (doppino telefonico, cavo coassiale, fibra ottica e così via):



Perché queste due stazioni possano comunicare, è necessario che la linea di comunicazione consenta il traffico delle informazioni[1] in entrambi i sensi. Quindi, come minimo si tratterà di una linea di tipo **half-duplex**, nella quale cioè la trasmissione possa avvenire in entrambi i sensi, ma non contemporaneamente: o si trasmette in un senso (A→B) oppure nell'altro (B→A). Potrebbe però anche trattarsi di una linea **full-duplex**, nella quale cioè la trasmissione nei due sensi può anche essere contemporanea. Capiremo più avanti quando è sufficiente una linea half-duplex e quando invece ne serve una full-duplex.

Come è noto, se facciamo riferimento al *livello di linea* (livello 2) del modello OSI, i dati sono suddivisi in **pacchetti** (che chiamiamo **frame**, cioè *trame*), ciascuno composto da un certo numero di bit (per esempio 1024 bit o più). Supponiamo allora che la stazione A generi un frame e voglia trasmetterlo alla stazione B. Quindi A è la **sorgente**, mentre B è la **destinazione**. Il pacchetto generato dal livello 2 della stazione A viene inviato al livello 1 della stessa stazione, il quale si occupa della trasmissione vera e propria, ossia prende i bit e li invia uno ad uno sulla linea[2]:



[1] Ricordiamo che nelle reti di telecomunicazioni che noi consideriamo, le informazioni sono sempre digitali, cioè bit. Questa precisazione nasce dal fatto che anche la rete di diffusione televisiva via etere (cioè una rete di tipo **broadcast**) è una rete di telecomunicazioni, ma di tipo analogico (se si escludono ovviamente le nuove trasmissioni televisive di tipo digitale trasmesse su cavo o via satellite).

[2] Le tecniche con cui, fisicamente, vengono trasmessi i bit (modulazione, velocità di trasmissione e così via) non ci interessano in questo momento.

I bit arrivano al livello 1 della stazione destinazione; il livello 1 provvede a fornire tali bit, nuovamente riuniti in pacchetti, al livello superiore. *Di fatto, quindi, è come se la comunicazione sia avvenuta tra i livelli 2 delle due stazioni, anche se praticamente non è stato così.*

La prima cosa da valutare è quanto tempo il trasmettitore della stazione A impiega per immettere i vari bit sulla linea. Si ragiona, evidentemente, in termini di *numero di bit immessi in linea per unità di tempo*, per cui possiamo parlare di **velocità di trasmissione**, misurata appunto in **bit/sec**.

Da osservare che questa velocità è ben diversa da quella con cui il segnale elettrico si propaga sulla linea, che è una **velocità di propagazione**: se stessimo trasmettendo via radio, cioè usando l'atmosfera terrestre come mezzo trasmissivo, la velocità di propagazione corrisponderebbe ai soliti **$3*10^8$ m/sec**; su un mezzo ad onde guidate, invece, come il *doppino telefonico* o il *cavo coassiale*, la velocità di propagazione sarà inferiore; per fare dei conti di massima, che quindi prescindano dal tipo di mezzo usato, è lecito considerare un valore approssimativo di **$2*10^8$ m/sec**.

E' chiaro che il numero di bit che compongono il generico pacchetto (frame) e la velocità di immissione dei bit sulla linea determinano quanto tempo è necessario al trasmettitore per inviare tutti i bit del pacchetto sulla linea: infatti, se T è questo tempo, esso sarà

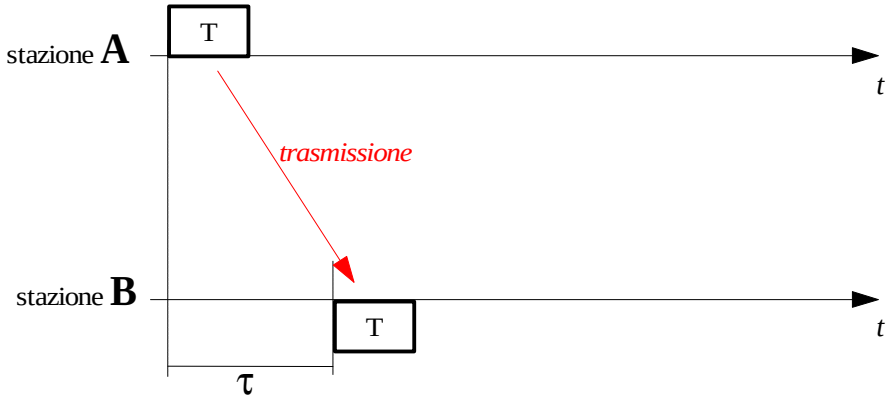
$$T(\text{sec}) = \frac{L(\text{bit})}{v_T(\text{bit}/\text{sec})}$$

dove abbiamo indicato con L la **lunghezza** del pacchetto e con v_T la *velocità di trasmissione* dei bit sulla linea.

Quindi, il trasmettitore A impiega un certo tempo T per immettere sulla linea tutti i bit del pacchetto. Ciascuno di questi bit impiega un certo tempo di propagazione τ per giungere alla stazione destinazione B. Questo tempo è legato, evidentemente, alla distanza d tra le due stazioni ed alla velocità di propagazione v_p, secondo la relazione

$$\tau(\text{sec}) = \frac{d(\text{m})}{v_p(\text{m}/\text{sec})}$$

Possiamo usare un diagramma temporale, del tipo illustrato nella figura seguente, per capire quello che accade:



Abbiamo qui riportato l'asse temporale con riferimento alle due stazioni. A partire da un certo istante t_0, la stazione A comincia a trasmettere il proprio pacchetto ed impiega un tempo T per immettere l'intero pacchetto sulla linea. Ciascun bit del pacchetto impiega un tempo τ per arrivare

alla destinazione, per cui la stazione B comincia a ricevere il pacchetto nell'istante $t_0+\tau$ e termina di riceverlo nell'istante t_0+T.

Abbiamo in tal modo inquadrato la situazione in modo del tutto generale. Adesso dobbiamo cominciare a studiare quello che deve accadere perché tutto il meccanismo della comunicazione tra A e B possa funzionare.

Il problema essenziale da affrontare riguarda gli errori di trasmissione, che sappiamo essere inevitabili nei vari mezzi trasmissivi che abbiamo a disposizione.

Quando la stazione B riceve il pacchetto inviatole da A, per prima cosa deve controllare se questo pacchetto presenta degli errori, dovuti alla trasmissione, oppure no. La possibilità di eseguire questo controllo è garantita dal fatto che ogni pacchetto contiene, oltre ai dati veri e propri, degli opportuni **codici di controllo**. La stazione B, quindi, sfrutta questi codici al fine di verificare la presenza o l'assenza di errori nel pacchetto ricevuto. Per compiere questa operazione, essa necessita di un certo **tempo di elaborazione**, che indichiamo con T_E.

Al termine di questa elaborazione, le possibilità sono due: se la prima è quella per cui il pacchetto non ha subito errori dalla trasmissione, mentre l'altra è che c'è stato almeno un errore.

Supponiamo che vengano utilizzati solo **codici a rivelazione di errore** che consentano l'individuazione di un qualsivoglia numero di errori sul singolo pacchetto. Questo significa che la stazione B è in grado sicuramente di individuare gli eventuali errori, ma non è in grado di correggerli: questo implica che, in presenza di errori, la stazione A debba necessariamente ritrasmettere lo stesso pacchetto.

E' quindi necessario un dialogo tra la stazione sorgente e la stazione destinazione, in modo che la destinazione possa segnalare alla sorgente eventuali errori di trasmissione e chiedere la ritrasmissione. Subentrano allora delle regole, racchiuse in un **protocollo di linea**, che le due stazioni devono rispettare per poter comunicare tenendo conto della considerazioni appena fatte.

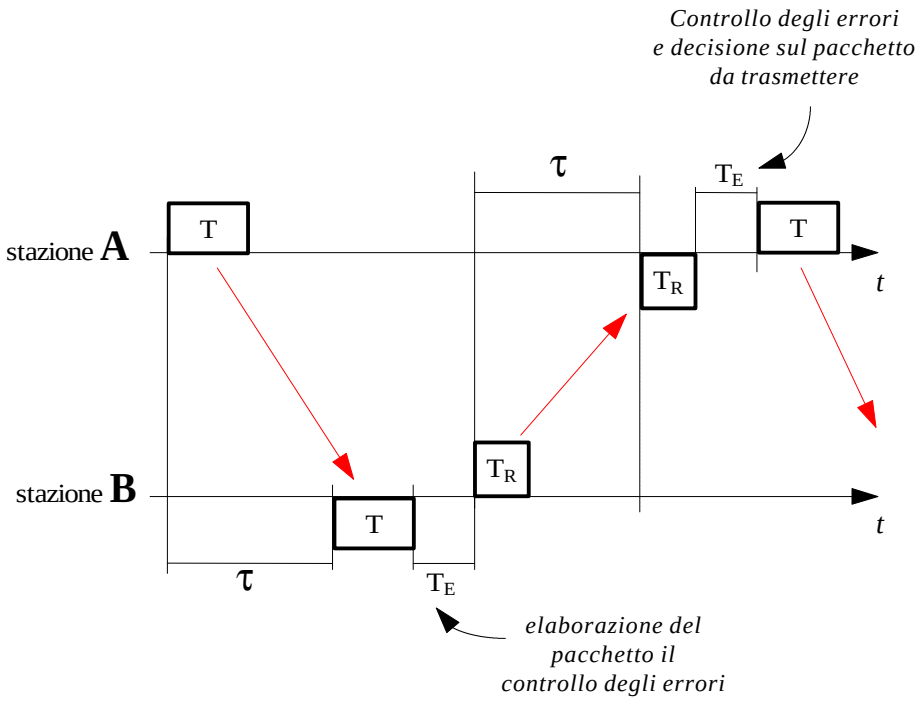
In linea generale, ciascun protocollo prevede che la stazione destinazione, dopo aver ricevuto ed elaborato un pacchetto, debba inviare alla sorgente un cosiddetto **pacchetto di riscontro**; questo deve semplicemente notificare alla sorgente l'esito della trasmissione: se l'esito è positivo, la sorgente non ha di che preoccuparsi; se l'esito è negativo, invece, la sorgente deve ritrasmettere il pacchetto.

Premesso questo principio di base, valido per tutti i protocolli di linea, vediamo come ciascun protocollo gestisca il problema degli errori.

PROTOCOLLO STOP AND WAIT

Il primo protocollo che analizziamo è il cosiddetto **protocollo stop and wait**. *Il concetto di base di questo protocollo è che la stazione sorgente, una volta trasmesso il generico pacchetto, debba necessariamente aspettare il pacchetto di riscontro prima di poter riprendere la trasmissione.* Ovviamente, se il pacchetto di riscontro è positivo (**ACK**, che sta per *ACKnowledgement*), allora la sorgente potrà trasmettere il messaggio successivo, mentre invece, se il pacchetto di riscontro è negativo (**NAK**, *Negative ACK*), dovrà ritrasmettere lo stesso pacchetto di prima. La ritrasmissione dello stesso pacchetto avverrà fin quando non arriverà un riscontro positivo.

Il diagramma temporale seguente mostra bene quello che accade:



*Funzionamento del protocollo **stop and wait**: la stazione A trasmette il suo pacchetto verso B e si ferma; una volta ricevuto il pacchetto per intero, B lo analizza (per un tempo T_E) e genera un pacchetto di riscontro, che sarà positivo se non ci sono stati errori nella trasmissione o negativo in caso contrario; il pacchetto di riscontro viene inviato in linea (impiegando un tempo $T_R<T$ data la minore lunghezza rispetto ad un pacchetto dati) ed impiega anch'esso un tempo t per arrivare ad A; quando A riceve (completamente) il pacchetto di riscontro, lo elabora (ancora per un tempo T_E), al fine di capire quale sia stato l'esito della trasmissione del pacchetto: se l'esito è stato positivo, allora trasmette il pacchetto successivo, mentre invece, se l'esito è stato negativo, ritrasmette il pacchetto precedente*

E' ovvio che `anche il pacchetto di riscontro, essendo composto da bit che viaggiano sulla linea, può subire degli errori durante la trasmissione.` Di conseguenza, anche questo pacchetto conterrà dei codici per il controllo degli errori e il tempo di elaborazione T_E indicato in figura è dovuto essenzialmente proprio all'elaborazione per il controllo degli errori. Per semplificare la nostra analisi, possiamo supporre che tali codici consentano comunque la correzione di eventuali errori, per cui possiamo essere certi che la stazione A, una volta ricevuto il pacchetto, possa sicuramente decidere se ritrasmettere il pacchetto precedente o trasmettere il nuovo pacchetto.

Concetto del "time-out"

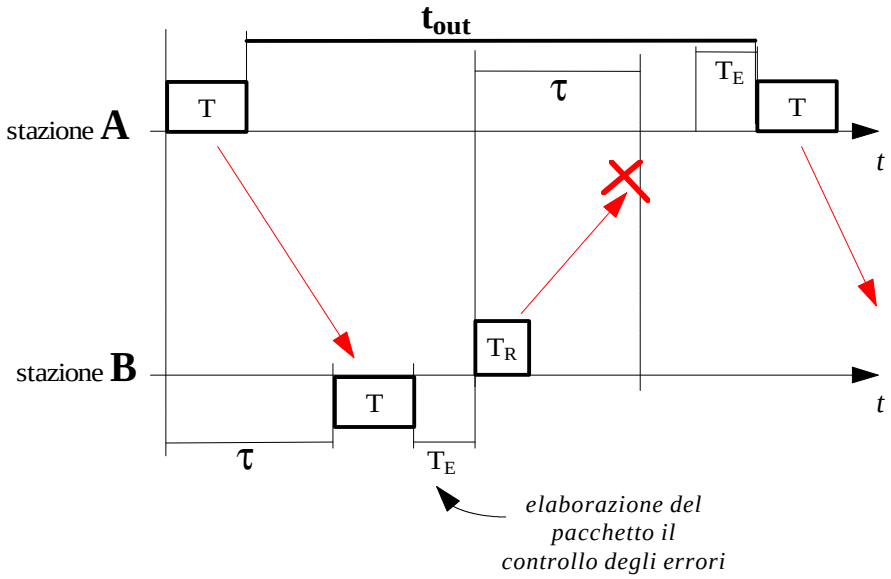
Le considerazioni fatte non bastano però a garantire il corretto funzionamento del sistema. Per esempio, supponiamo che la stazione B abbia ricevuto un pacchetto, lo abbia elaborato ed abbia inviato ad A il corrispondente pacchetto di riscontro (non ci interessa se positivo o negativo). Supponiamo però che, per un qualche motivo, questo pacchetto non giunga alla stazione A: ad esempio, possiamo supporre che il segnale elettrico subisca una attenuazione tale, durante la propagazione, da scendere sotto la *soglia del rumore*, per cui diventa indecifrabile in ricezione[1]. La stazione A, dal canto suo, non può trasmettere niente finche non arriva il pacchetto di riscontro, per cui, in questo caso, aspetterebbe in eterno e la trasmissione non andrebbe a buon fine.

[1] Un'altra possibilità, seppure meno frequente, è quella di una perdita di sincronizzazione tra trasmettitore e ricevitore

Per evitare che si crei questa situazione, l'unica possibilità è che la stazione A debba aspettare il pacchetto di riscontro non oltre un tempo massimo, detto **time out**: se questo *time out* scade e la stazione A non ha ricevuto alcun pacchetto di riscontro, la stazione A deve ritrasmettere lo stesso pacchetto.

Il motivo per cui si procede così è semplice: il pacchetto di riscontro potrebbe non giungere ad A sia perché è diventato indecifrabile, come supposto prima, sia perché potrebbe non essere stato trasmesso. In altre parole, potrebbe capitare che sia stata la stazione B a non ricevere il pacchetto inviatole da A, per cui, non ricevendo niente, non ha nemmeno trasmesso alcun riscontro. Dato che A non ha modo di sapere quale delle due situazioni si sia verificata (perdita del pacchetto dati o perdita del pacchetto di riscontro), la cosa più logica da fare è ritrasmettere il pacchetto originale.

La figura seguente mostra, con i soliti assi temporali, il modo di procedere:



*Uso del **time-out** nel protocollo stop-and-wait: la stazione A, nel momento in cui termina di inviare in linea i bit del pacchetto, fa scattare un orologio (clock); quando questo orologio segna un tempo pari al time-out (t_{out}), che è stato ovviamente fissato a priori, e non è stato ancora ricevuto alcun pacchetto di riscontro, la stazione A ritrasmette lo stesso pacchetto*

La figura mette in evidenza una cosa molto importante e cioè quanto deve essere lungo il time-out: esso dovrà essere non inferiore al tempo necessario perché, in A, giunga il pacchetto di riscontro. In altre parole, la stazione A dovrà comunque aspettare il tempo minimo necessario perché l'eventuale pacchetto di riscontro possa arrivare. In caso contrario, il time out scadrebbe sempre e quindi la stazione A trasmetterebbe sempre la stessa cosa.

E' allora facile calcolare il minimo valore del time-out, in quanto basta calcolare il tempo minimo che intercorre tra la trasmissione del pacchetto dati, da parte di A, e la ricezione, sempre da parte di A, del corrispondente pacchetto di riscontro. Possiamo far riferimento all'ultima figura: c'è inizialmente un tempo τ affinché il pacchetto trasmesso dalla stazione A venga interamente ricevuto da B, la quale impiega un tempo T_E per l'elaborazione ed un successivo tempo T_R per inviare in linea il pacchetto di riscontro; segue un ulteriore tempo τ affinché A riceva il completo pacchetto di riscontro ed un tempo T_E per l'elaborazione di tale riscontro. Deduciamo che

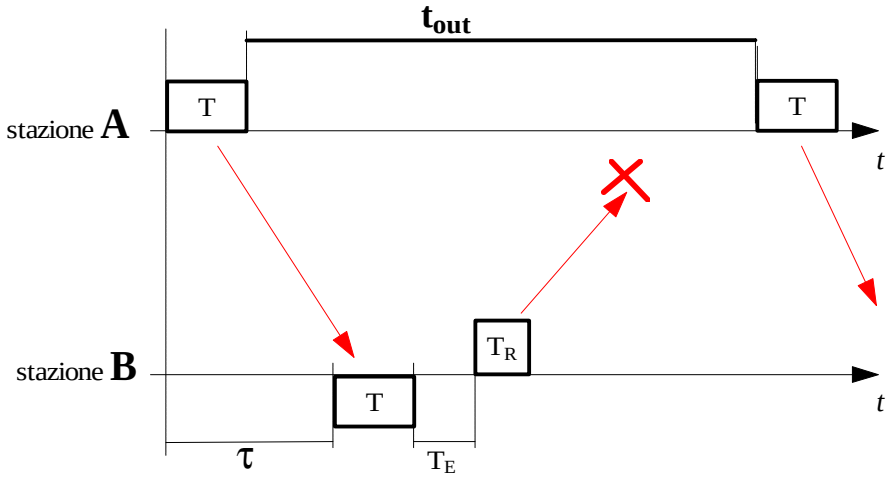
$$t_{out} \geq 2\tau + 2T_E + T_R$$

Il minimo valore da attribuire a t_{out} è dunque $2\tau+2T_E+T_R$. La scelta concreta del valore da usare dipende da considerazioni che in questa sede non facciamo. Ad ogni modo, si capisce che la quantità $2\tau+2T_E+T_R$ sia comunque molto approssimativa, in quanto deriva da diverse ipotesi semplificative: per esempio, non è detto che il tempo di elaborazione sia lo stesso per B (che riceve i pacchetti di dati) e per A (che riceve i pacchetti di riscontro) e non è nemmeno detto che tale tempo sia costante. Bisognerà perciò garantire una sufficiente elasticità da parte del time-out.

Concetto del "numero di sequenza"

L'uso del time-out, se da un lato evita che la stazione sorgente aspetti un tempo infinito per trasmettere, dall'altro crea un altro problema. Abbiamo infatti osservato prima che una situazione in cui il time-out scade è quella in cui il pacchetto di riscontro inviato da B non viene ricevuto da A. Se il pacchetto di riscontro era positivo, il problema è tutto di A, che non riceve appunto il riscontro, mentre invece B ha ricevuto tranquillamente il suo pacchetto senza errori. Nel momento in cui A vede scadere il time-out e rinvia lo stesso pacchetto, la stazione B non ha modo di sapere che si tratta dello stesso pacchetto di prima (perché non può sapere che il suo pacchetto di riscontro non è arrivato a destinazione), per cui lo prende come un nuovo pacchetto, commettendo un errore.

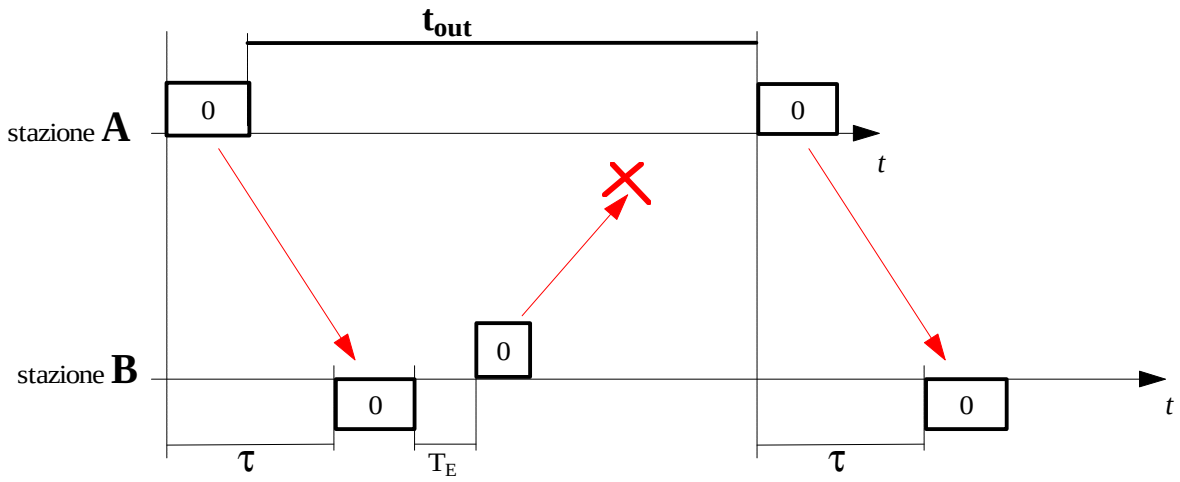
Si tratta in pratica di quello che accade nella figura seguente:



La stazione A, quando termina di inviare in linea i bit del pacchetto, avvia il time-out; la stazione B, nel frattempo, riceve regolarmente il pacchetto trasmesso da A e invia il corrispondente pacchetto di riscontro, contenente un ACK. Dato che questo pacchetto si perde durante la trasmissione, la stazione A vede scadere il time-out e ritrasmette lo stesso pacchetto di prima; la stazione B, non sapendo che il suo pacchetto di riscontro non è arrivato ad A, prende il nuovo pacchetto e lo considera come successivo al primo, commettendo un errore

Per evitare quest'altro tipo di inconveniente, l'unica soluzione è quella di identificare i pacchetti ed i corrispondenti riscontri, ad esempio numerandoli. Si usa perciò il cosiddetto **numero di sequenza del pacchetto**: *ciascun pacchetto ha un proprio numero di sequenza e lo stesso numero è contenuto nel pacchetto di riscontro (positivo o negativo) relativo.* Questo consente sempre di evitare errori.

Ad esempio, consideriamo la stessa situazione vista poco fa, includendo però questa volta i numeri di sequenza:



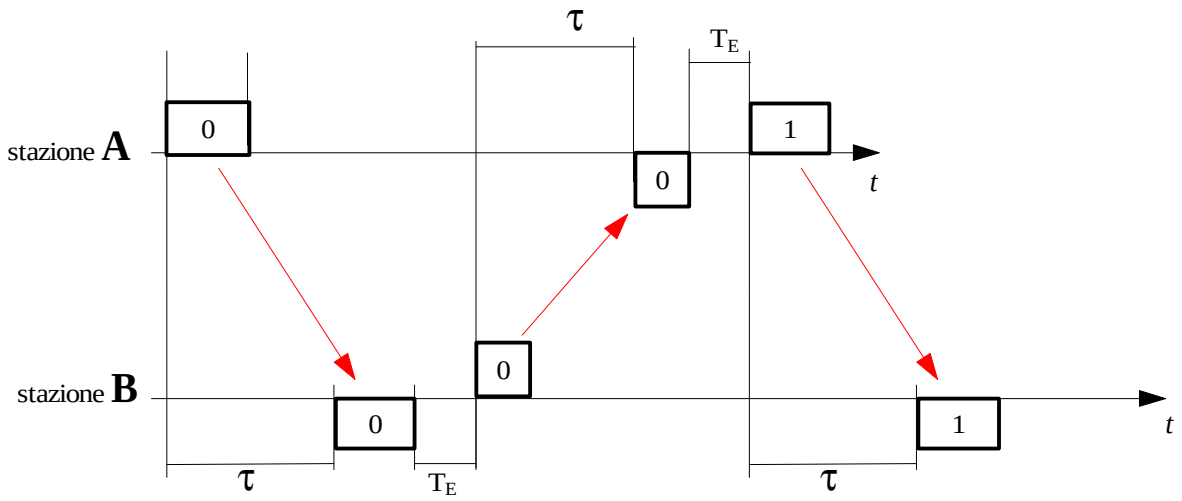
La stazione A genera un pacchetto e lo numera con 0; lo invia sulla linea di trasmissione e fa scattare il time-out; la stazione B riceve regolarmente il pacchetto trasmesso da A e invia un pacchetto di riscontro, contenente un ACK ed il numero di sequenza 0 corrispondente. Supponiamo che questo pacchetto si perda durante la trasmissione: la stazione A vede scadere il time-out e ritrasmette lo stesso pacchetto di prima, il quale quindi avrà ancora numero di sequenza 0; la stazione B non sa che il suo pacchetto di riscontro non è arrivato ad A, per cui si aspetta un nuovo pacchetto (cioè con numero di sequenza uguale ad 1); tuttavia, essa riceve un pacchetto il cui numero di sequenza è ancora 0. Da qui deduce che c’è stato qualche problema nella stazione A: avendo già ricevuto il pacchetto 0, scarta la replica appena arrivata ed aspetta la successiva trasmissione.

Per quanto riguarda la numerazione, si potrebbe effettuare una numerazione progressiva (cioè 1,2,...), ma in effetti è sufficiente, in questo tipo di protocollo, alternare i valori 0 ed 1. Questa scelta è vantaggiosa in quanto il campo contenente il numero di sequenza potrà essere di 1 solo bit.

Un grosso vantaggio della numerazione in sequenza, che peraltro sarà più evidente nei protocolli che esamineremo più avanti, si ha quando i pacchetti emessi da una stazione possono seguire in rete percorsi diversi (come accade in alcune *reti a commutazione di pacchetto*): in questi casi, infatti, può accadere che un pacchetto trasmesso dopo arrivi prima di un altro e il numero di sequenza è l’unica cosa che permette di ricostruire l’ordine giusto.

Analisi di efficienza del protocollo

Un importante indice di prestazione di un protocollo di linea riguarda la cosiddetta **efficienza** di utilizzo del canale di trasmissione: *questo parametro è definito come il rapporto tra il tempo di utilizzo del canale per la trasmissione dati ed il tempo totale di utilizzo del canale.* Vediamo di spiegare bene questa definizione, considerando la situazione descritta nella figura seguente:



La stazione A invia il pacchetto 0; questo viene ricevuto da B e supponiamo che non ci siano errori, per cui B invia un riscontro positivo ad A; quando A riceve tale riscontro, lo elabora, capisce che la trasmissione del pacchetto 0 è andata a buon fine e quindi passa a trasmettere il pacchetto successivo, numerato cioè con 1.

Si capisce dunque che questo meccanismo di comunicazione impegna il canale, per la trasmissione del pacchetto 0, per un tempo totale pari a

$$T_{TOT} = \tau + T + T_E + \tau + T_R + T_E = 2\tau + T + 2T_E + T_R$$

Questo tempo, però, non è interamente impiegato per la trasmissione del pacchetto 0. Al contrario, il tempo di trasmissione del pacchetto 0 è pari solo a T, mentre tutti gli altri tempi sono necessari per le varie procedure di controllo descritte nei precedenti paragrafi.

Questo discorso vale ovviamente per qualunque pacchetto, per cui diciamo che l'efficienza di utilizzo del canale è in questo caso

$$E = \frac{T}{T_{TOT}} = \frac{T}{2\tau + T + 2T_E + T_R} = \frac{1}{2\frac{\tau}{T} + 1 + 2\frac{T_E}{T} + \frac{T_R}{T}}$$

In realtà, questo non è il valore generico dell'efficienza di questo protocollo, ma è il valore migliore cui possiamo aspirare. Il motivo è semplicemente nel fatto che il tempo T_{TOT} appena valutato è quello che impieghiamo nel caso in cui la trasmissione vada a buon fine, ossia nel caso non sia necessaria alcuna ritrasmissione:

$$E_{max} = \frac{T}{T_{TOT}} = \frac{1}{2\frac{\tau}{T} + 1 + 2\frac{T_E}{T} + \frac{T_R}{T}}$$

Al contrario, se si dovesse verificare qualche errore, sarebbe necessario ritrasmettere, per cui, mentre rimane invariato il numeratore (=T) di quella frazione, il denominatore raddoppia, perché facciamo due trasmissioni: quindi avremo a denominatore $2T_{TOT}$. Se anche sulla seconda trasmissione si verificano degli errori, dovremo nuovamente ritrasmettere e quindi avremmo a denominatore $3T_{TOT}$ e così via.

Sulla base di ciò, se vogliamo dare una definizione generale di efficienza, che cioè tenga conto delle possibili ritrasmissioni, ci basterà inserire a denominatore un fattore moltiplicativo pari al numero medio di trasmissioni effettuate:

$$E = \frac{T}{E[n_T] \cdot T_{TOT}} = \frac{1}{E[n_T] \cdot \left(2\frac{\tau}{T} + 1 + 2\frac{T_E}{T} + \frac{T_R}{T} \right)}$$

E' ovvio il motivo per cui consideriamo un numero *medio* di trasmissioni: facendo un discorso aprioristico, non possiamo sapere quante trasmissioni dovremo effettuare per il singolo pacchetto, per cui il numero di trasmissioni n_T è una variabile aleatoria della quale non possiamo far altro che calcolare la media.

Il calcolo di questa media è stato più volte effettuato nel corso di *Teoria dei Segnali*, per cui non lo ripetiamo e ci limitiamo a riportarne il risultato: se indichiamo con P la probabilità che il generico pacchetto arrivi a destinazione con errori[1], risulta che

$$E[n_T] = \frac{1}{1-P}$$

Si tratta peraltro di un risultato abbastanza intuitivo: all'aumentare di P, il numero medio di trasmissioni aumenta, mentre invece, per $P \to 0$, il numero medio di trasmissioni tende all'unità.

Sostituendo questa espressione in quella dell'efficienza, abbiamo dunque che

$$E = \frac{1}{\frac{1}{1-P} \cdot \left(2\frac{\tau}{T} + 1 + 2\frac{T_E}{T} + \frac{T_R}{T} \right)} = \frac{1-P}{2\frac{\tau}{T} + 1 + 2\frac{T_E}{T} + \frac{T_R}{T}}$$

Nella maggior parte delle applicazioni, i termini $2T_E/T$ e T_R/T sono trascurabili rispetto agli altri termini a denominatore, per cui possiamo concludere, con buona approssimazione, che l'**efficienza** di utilizzo del canale, con il protocollo stop-and-wait, vale

$$\boxed{E \cong \frac{1-P}{1+2\frac{\tau}{T}}}$$

E' evidente che, per P=0, cioè se il pacchetto trasmesso non subisce errori e viene perciò trasmesso una sola volta, ricadiamo nel valore E_{max} calcolato prima (trascurando sempre i termini $2T_E/T$ e T_R/T).

Il termine **τ/T** (cioè il tempo di propagazione del segnale normalizzato al tempo di trasmissione del generico pacchetto) ricorrerà spesso nei nostri discorsi, per cui lo indicheremo nel seguito con il simbolo *a*:

$$E \cong \frac{1-P}{1+2a}$$

[1] E' bene non confondere la probabilità P con la probabilità di errore sul singolo bit: infatti, P è la probabilità che, sugli L bit che compongono il pacchetto, ce ne sia almeno uno sbagliato.

Esempio numerico

Facciamo un esempio numerico per renderci conto almeno degli ordini di grandezza delle quantità che abbiamo considerato negli ultimi discorsi.

Supponiamo che il trasmettitore della stazione sorgente lavori a velocità v_T=1Gbit/sec, ossia immette in linea 10^9 bit al secondo. Supponiamo inoltre che la sorgente e la destinazione distino d=10 km e che il generico pacchetto sia lungo L=1000 bit. Valutiamo l'efficienza di utilizzo del canale.

In primo luogo, calcoliamo il ritardo di propagazione τ, che sappiamo essere funzione della distanza *d* nonché della velocità v_P di propagazione del segnale elettrico: supponendo per quest'ultima il valore di $2*10^8$ m/sec, abbiamo che

$$\tau(\text{sec}) = \frac{\text{d(m)}}{\text{v}_\text{p}(\text{m/sec})} = \frac{10 \cdot 10^3 (\text{m})}{2 \cdot 10^8 (\text{m/sec})} = 0.5 \cdot 10^{-4} (\text{sec})$$

Calcoliamo inoltre il tempo T di trasmissione del generico pacchetto sulla linea:

$$\text{T(sec)} = \frac{\text{L(bit)}}{\text{v}_\text{T}(\text{bit/sec})} = \frac{10^3 (\text{bit})}{10^9 (\text{bit/sec})} = 10^{-6} (\text{sec})$$

Questi due parametri ci permettono di calcolare *a* e quindi anche la massima efficienza in cui possiamo sperare:

$$\text{a} = \frac{\tau}{\text{T}} = 50 \longrightarrow \text{E}_{\max} \cong \frac{1}{1+2\text{a}} = 0.01$$

Abbiamo dunque una efficienza di appena lo 0.01%, cioè estremamente bassa. In termini pratici, questo può essere interpretato dicendo che, mentre il trasmettitore invia i bit in linea alla velocità di 10^9(bit/sec), la destinazione, nella migliore delle ipotesi (cioè senza alcuna ritrasmissione), riceve tali bit ad una velocità che è l'1% di 10^9(bit/sec), ossia 10^7(bit/sec). La conseguenza di questo fatto è che è inutile affannarsi a progettare un trasmettitore così veloce, perché comunque il protocollo di linea ne limita inevitabilmente le prestazioni.

Ad esempio, supponiamo di considerare un nuovo trasmettitore, che trasmetta a $v_T=10^8$(bit/sec), ossia un ordine di grandezza in meno rispetto a prima. Ripetendo i calcoli, abbiamo che

$$\text{T(sec)} = \frac{10^3 (\text{bit})}{10^8 (\text{bit/sec})} = 10^{-5} (\text{sec}) \longrightarrow \text{a} = \frac{\tau}{\text{T}} = \frac{0.5 \cdot 10^{-4}}{10^{-5}} = 5 \longrightarrow \text{E}_{\max} \cong \frac{1}{1+2\text{a}} = \frac{1}{1+2 \cdot 5} = 0.091$$

Abbiamo una efficienza maggiore rispetto a prima, pari a circa il 9%. Ciò significa che la destinazione riceve i bit ad una velocità che è il 9% di 10^8(bit/sec), ossia $1.1*10^7$(bit/sec). Abbiamo ottenuto un valore molto prossimo al precedente (10^7 bit/sec), con la differenza che abbiamo usato un trasmettitore più lento.

Se volessimo migliorare ancora l'efficienza, anche sacrificando la velocità, potremmo prendere un trasmettitore che trasmetta a $v_T=10^6$(bit/sec). Facendo i conti, si trova una efficienza massima di circa il 99%. Questo significa che la destinazione riceve i bit praticamente alla stessa velocità con cui essi sono inviati in linea.

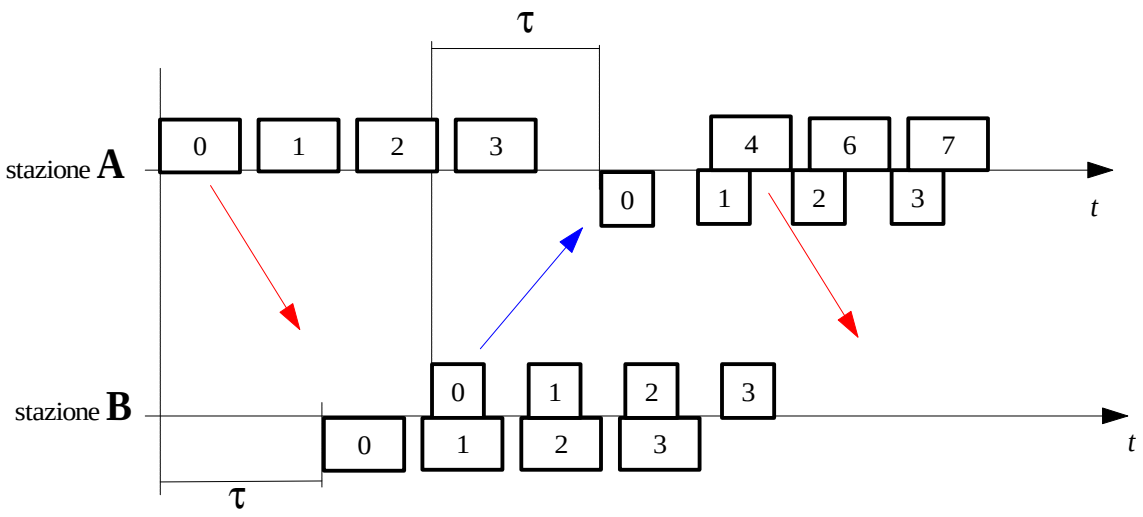
E' ovvio che abbiamo ottenuto un risultato ancora diverso rispetto al precedente: infatti, è vero che abbiamo aumentato l'efficienza, ma è anche vero che, comunque, la velocità è inferiore a prima. A seconda del contesto, si tratta di privilegiare l'una o l'altra esigenza: alta efficienza e minore velocità oppure bassa efficienza ma maggiore velocità.

Ad ogni modo, si deduce, in generale, che *il valore dell’efficienza dipende strettamente dal tempo di propagazione t del segnale e quindi sia dalla velocità di propagazione del segnale sia dalla distanza tra sorgente e destinazione.*

PROTOCOLLO CONTINUOUS ARQ

Il **protocollo Continuous ARQ** è di grande importanza pratica, in quanto consente un notevole miglioramento dell’efficienza di utilizzo della linea rispetto al protocollo precedente. Il concetto di base è il seguente: *la stazione sorgente può emettere un certo numero di pacchetti consecutivi senza dover attendere conferma della ricezione per ognuno di essi.* Il numero massimo N di messaggi consecutivi prende il nome di **finestra** (*window*) e differisce da caso a caso: ovviamente, si tratta di un valore prefissato. Vediamo i dettagli.

In primo luogo, questo protocollo, al contrario del precedente, presuppone l’uso di una linea di tipo full-duplex, nella quale cioè siano ammesse trasmissioni contemporanee in entrambi i sensi. Questo consente di mettere in piedi il meccanismo illustrato nella figura seguente:



La stazione A comincia a trasmettere sulla linea un certo numero *n* di pacchetti, che nella figura è assunto essere pari a 4. Dopo aver trasmesso questi 4 pacchetti, deve necessariamente fermarsi ed aspettare che arrivi il riscontro al primo di tali pacchetti: nel caso rappresentato in figura, il primo pacchetto è numerato con 0, per cui essa deve aspettare il pacchetto di riscontro contrassegnato con numero di sequenza 0. Appena arriva tale riscontro, essa lo elabora e ci sono quindi due possibilità, a seconda che il riscontro sia positivo o negativo. Per il momento, supponiamo che il riscontro sia sempre positivo: in questo caso, la sorgente può trasmettere un’altra finestra, cioè altri 4 pacchetti, dopo di che dovrà nuovamente fermarsi per aspettare i corrispondenti pacchetti di riscontro.

Quindi, ogni volta che viene trasmessa una finestra, la sorgente deve aspettare la conferma, positiva o negativa, della controparte. Come vedremo e come peraltro si può già intuire, *quanto maggiore è la dimensione della finestra, tanto migliore è lo sfruttamento della linea.*

Con un meccanismo di questo tipo, diventa ovviamente indispensabile attribuire una numerazione a ciascun messaggio, in accordo a quanto detto nel paragrafo precedente per il *protocollo stop and wait*. Si procede allora nel modo seguente:

- la stazione che trasmette possiede una *variabile interna*, V(S), il cui valore viene incrementato di 1 ad ogni pacchetto trasmesso; il contenuto di V(S) viene così a rappresentare il numero di sequenza del pacchetto e quindi viene inviato con esso;
- la stazione remota, che riceve i pacchetti, possiede a sua volta una *variabile interna*, V(R), il cui valore viene incrementato di 1 per ogni pacchetto ricevuto. Quando essa riceve il generico pacchetto, si comporta nel modo seguente:
  - se non ha riscontrato errori, invia un riscontro positivo (ACK) alla stazione sorgente, inserendo il valore di V(R), in modo che la sorgente possa dedurre, da tale valore, quale sia il pacchetto cui il riscontro fa riferimento;
  - se invece ha riscontrato errori, invia alla sorgente un riscontro negativo (NAK) recante il numero di sequenza del pacchetto su cui sono stati riscontrati errori.

Quando la sorgente riceve un NAK, deve provvedere alla correzione e ci sono ancora una volta due strategie perseguibili:

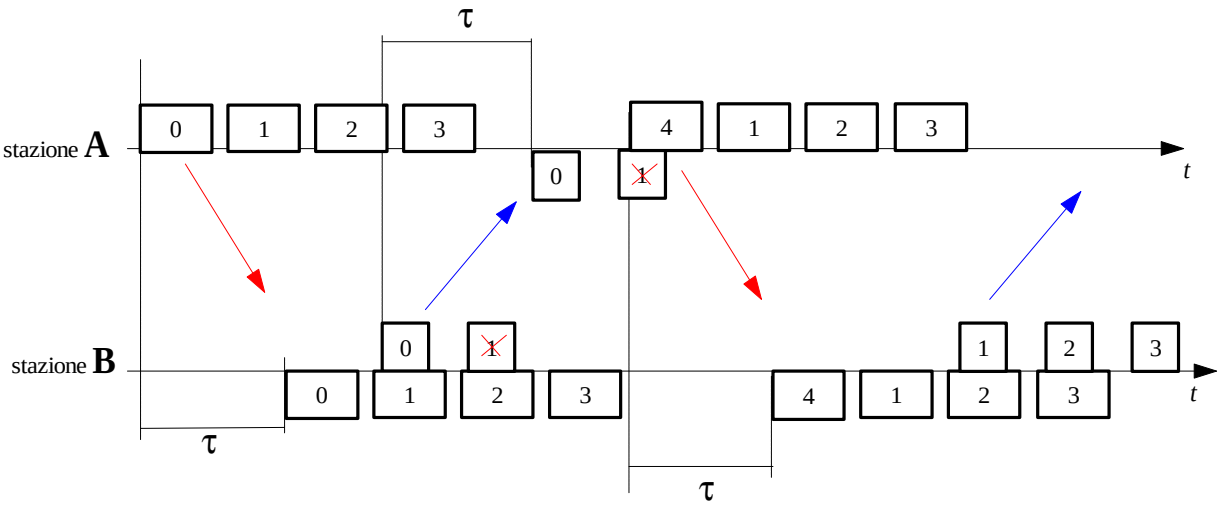
- la prima è il **selective repeat** (*ripetizione selettiva*), in cui si ritrasmette solo il pacchetto errato; questa scelta ha però un difetto: se ci sono stati errori non solo sul quel pacchetto, ma anche su uno o più pacchetti successivi, la sorgente riceverà altri NAK e quindi dovrà effettuare altre ritrasmissioni;
- la cosa si può invece risolvere drasticamente con la tecnica del **go back n** (*torna al numero n*), con la quale la sorgente provvede direttamente a ritrasmettere tutti i pacchetti a partire da quello segnalato come errato.

Nei successivi paragrafi saranno analizzate nel dettaglio queste due tecniche. Per il momento, ci limitiamo a dire che la scelta di quale tecnica seguire dipende dal tipo di linea: se la linea è di qualità elevata, è ragionevole aspettarsi pochi errori, per cui la ritrasmissione del singolo pacchetto errato non dovrebbe essere seguita da altre ritrasmissioni; al contrario, se la linea è di bassa qualità, è presumibile che il pacchetto sbagliato non sia uno solo ed è quindi opportuno ritrasmettere anche i pacchetti successivi al posto di aspettare probabili altri NAK.

Tecnica del "go back n"

Come detto poco fa, nel caso si utilizzi la tecnica del **go back n**, la sorgente ritrasmette tutti i messaggi a partire da quello che la stazione destinazione ha segnalato come errato[1]. Ad esempio, supponiamo di usare una finestra di n=4 pacchetti e che la destinazione rilevi errori sul pacchetto di numero 1:

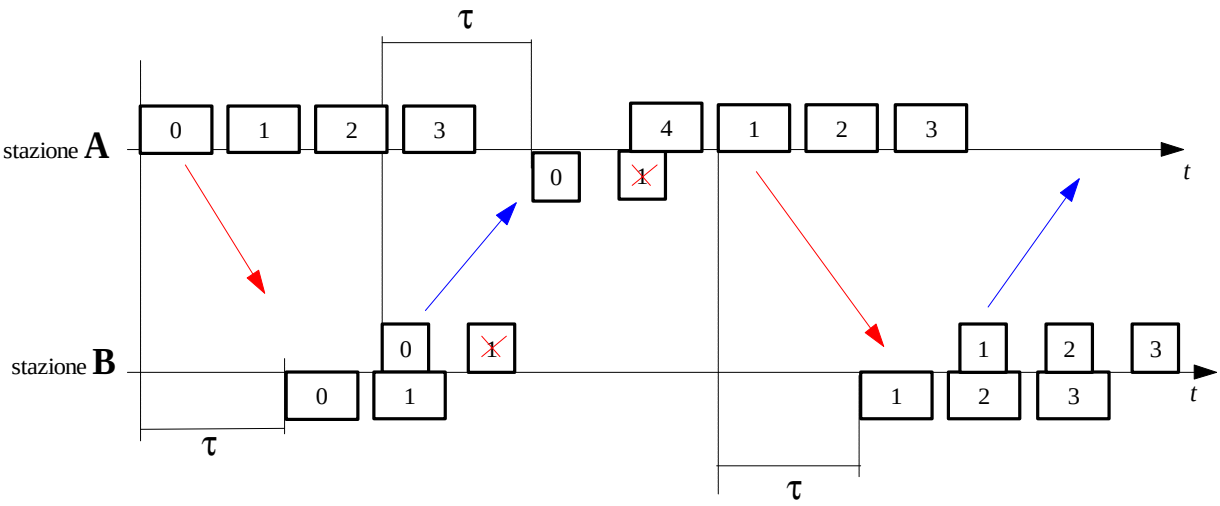
[1] Se n=1, si torna ovviamente al protocollo stop-and-wait, in quanto la sorgente deve aspettare il riscontro a ciascun pacchetto prima di riprendere a trasmettere.



La stazione B riceve dunque il pacchetto 0 e, non rilevando errori su di esso, trasmette un riscontro positivo; la stazione A che nel frattempo si era fermata, riceve il riscontro positivo del pacchetto 0 e trasmette il pacchetto 4. Mentre trasmette questo pacchetto, arriva il riscontro negativo del pacchetto 1; accorgendosi di esso, la stazione A non trasmette più il pacchetto 5, come avrebbe voluto, ma riprende a trasmettere dal pacchetto 1 fino al pacchetto 3, in modo da completare la finestra. Dopo di che, si pone in attesa dei corrispondenti riscontri: se i riscontri sono positivi, essa riprenderà la trasmissione rispettando la numerazione, ossia ripartendo di nuovo dal pacchetto 4.

Facciamo osservare una cosa: nella figura, si vede che la stazione B, dopo aver ricevuto errato il pacchetto 1 ed aver inviato un corrispondente riscontro negativo, non invia più riscontri per i pacchetti 2 e 3 né per il successivo pacchetto 4. Il motivo è evidente: la stazione B sa che la stazione A, una volta ricevuto il riscontro negativo per il pacchetto 1, ritrasmetterà sia il pacchetto 1 sia i pacchetti successivi; di conseguenza, la stazione B, dopo aver ricevuto il pacchetto 1 sbagliato, si disinteressa di tutti i pacchetti ricevuti successivamente ed aspetta di ricevere nuovamente il pacchetto 1.

Lo schema logico corretto da considerare è dunque il seguente:



Qui è evidenziato il fatto che i pacchetti successivi al pacchetto 1 (giunto errato) vengono scartati dalla stazione B, finche non giunge nuovamente il pacchetto 1. Quando questo arriva, supponendo che non vengano rilevati errori, riprende il normale funzionamento, per cui la stazione B riprende ad accettare i pacchetti in arrivo e a inviare i corrispondenti pacchetti di riscontro.

Prima di scendere in ulteriori dettagli, relativi all'analisi dell'efficienza di questo tipo di protocollo, è opportuno fare una serie di considerazioni:

- in primo luogo, da quanto detto risulta evidente che la trasmissione e la ricezione dei pacchetti avvengono spesso in contemporanea, il che significa che devono essere necessariamente svolte da hardware diversi: quindi, ciascuna stazione avrà dell'hardware per la trasmissione dei pacchetti e dell'hardware per la ricezione (e successiva elaborazione) dei pacchetti;
- in secondo luogo, la procedura per cui, in presenza di errori, si riprende a trasmettere dal primo messaggio rilevato come errato non sempre può essere opportuna; pensiamo, ad esempio, al caso di una trasmissione telefonica digitale (come quella dei telefoni cellulari che usano la **rete GSM**), nella quale la voce viene digitalizzata e trasmessa a pacchetti: se un singolo pacchetto viene rilevato come errato, ci vorrà un certo tempo perché esso venga ritrasmesso insieme a quelli successivi e questo potrebbe non essere gradevole durante la conversazione, in quanto questa non avverrebbe più in tempo reale; d'altra parte, *in una conversazione telefonica, la comunicazione non risente certo di un singolo pacchetto errato,* per cui è un tipico caso in cui non viene effettuata la ritrasmissione ma si prosegue come se niente fosse accaduto. Questo è comunque un caso limite; *un metodo, invece, più ragionevole consiste nell'attribuire all'utente, cioè il destinatario finale dei vari pacchetti, la decisione se un dato pacchetto, individuato come errato, deve essere ritrasmesso:* in pratica, quindi, la destinazione accetta tutti i pacchetti e l'utente decide poi in proprio se i pacchetti errati sono indispensabili[1], per cui è necessario ritrasmetterli, oppure se ne può fare a meno, per cui la trasmissione può continuare come se niente fosse successo. Con una scelta di questo tipo, si semplifica notevolmente l'hardware dedicato alla ricezione dei pacchetti, in quanto tutti i controlli vengono demandati all'utente (e quindi saranno verosimilmente compiuti via software);
- l'ultima osservazione da fare riguarda ancora una volta il **time-out**: anche nel protocollo go-back-n, infatti, non si può prescindere dal time-out, in quanto la stazione sorgente, dopo aver trasmesso una finestra, si pone sempre in attesa del riscontro al primo pacchetto della finestra; se questo riscontro non dovesse arrivare, la sorgente aspetterebbe in eterno e questo non è possibile. Quindi, *la sorgente, ad ogni trasmissione di un pacchetto, attiva un time-out; se il riscontro di quel pacchetto non arriva entro il time-out, la sorgente interrompe la normale trasmissione e riprende la trasmissione da quel pacchetto* (così come farebbe se fosse arrivato un riscontro negativo). *Il fatto che i pacchetti siano numerati garantisce che la stazione destinazione capisca ciò che è successo ed agisca di conseguenza.* E' ovvio che se la stazione B ha comunque ricevuto correttamente i pacchetti, nonostante A non lo abbia saputo e quindi li abbia ritrasmessi, non è necessario, per B, riprendere nuovamente i pacchetti; quindi, la stazione B capisce quale sia stato il problema di A e, pur inviando i pacchetti di riscontro[2], scarta i pacchetti che via via arrivano, fin quando non ne arriva uno nuovo, che cioè non sia stato precedentemente ricevuto.

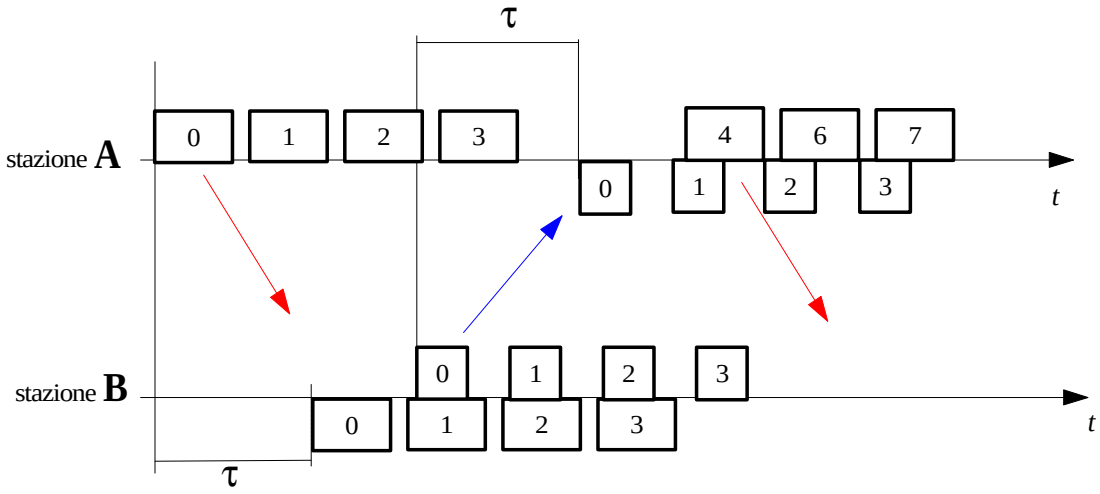
[1] Ad esempio, nella trasmissione dati, se si stanno trasmettendo dei file eseguibili, è necessario avere tutti i bit corretti, mentre invece, se si trasmettono immagini o suoni, si può anche tollerare che qualche bit sia errato.

[2] La stazione B non può esimersi dal mandare i pacchetti di riscontro, anche se sta scartando i pacchetti che le arrivano, in quanto, se facesse così, imporrebbe ad A di trasmettere sempre le stesse cose. E' chiaro che tutti i pacchetti di riscontro (fittizi, potremmo dire) inviati da B saranno positivi (ACK).

Osservazione

E' importante sottolineare un preciso legame esistente tra la dimensione *n* della finestra prevista dal protocollo ed il numero di bit che ciascun pacchetto riserva per la propria numerazione.

Abbiamo detto che, per il protocollo go-back-n, la sorgente può trasmettere un numero massimo *n* di pacchetti prima di ricevere il primo riscontro dalla destinazione:

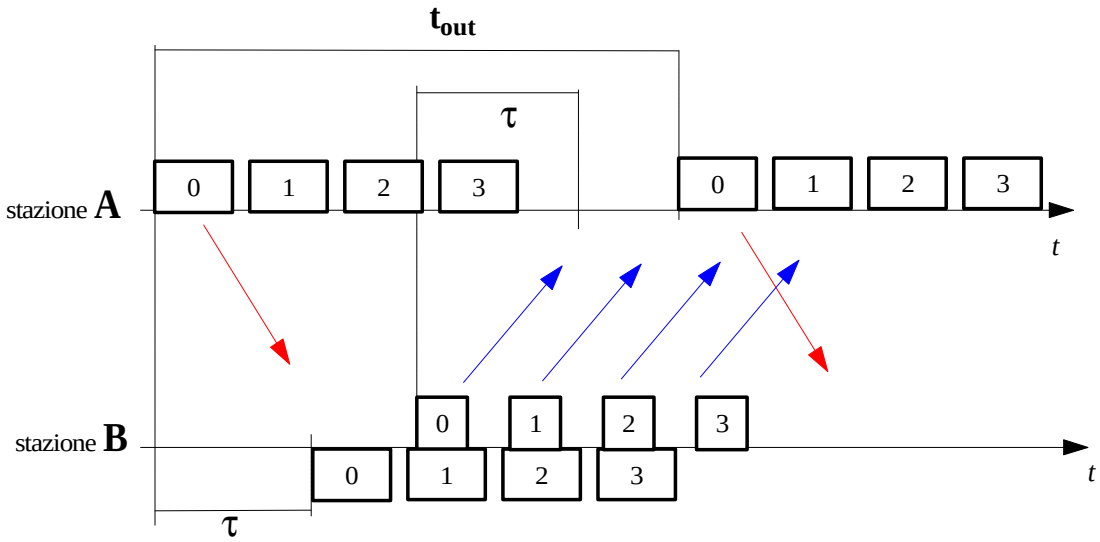


Solo dopo l'arrivo del primo riscontro, e la successiva elaborazione su di esso, la sorgente può riprendere la trasmissione, che partirà da un nuovo pacchetto se il riscontro è positivo (come in figura) oppure dal primo pacchetto se il riscontro è negativo.

Per quanto riguarda la numerazione dei pacchetti, dobbiamo prevedere, in ciascun pacchetto, un campo di un certo numero M di bit. In tal modo, potremo numerare i pacchetti da 0 a 2^M-1; il pacchetto successivo a quello di numero 2^M-1 sarà nuovamente numerato con 0 e la numerazione ripartirà progressivamente.

Sotto questa ipotesi, affinché la comunicazione possa funzionare è necessario numerare i pacchetti in modo da rispettare la condizione $\mathbf{n<2^M}$, dove n è la dimensione della finestra. Ad esempio, se la finestra è lunga n=4, come nei casi considerati prima, dovrà essere M>2.

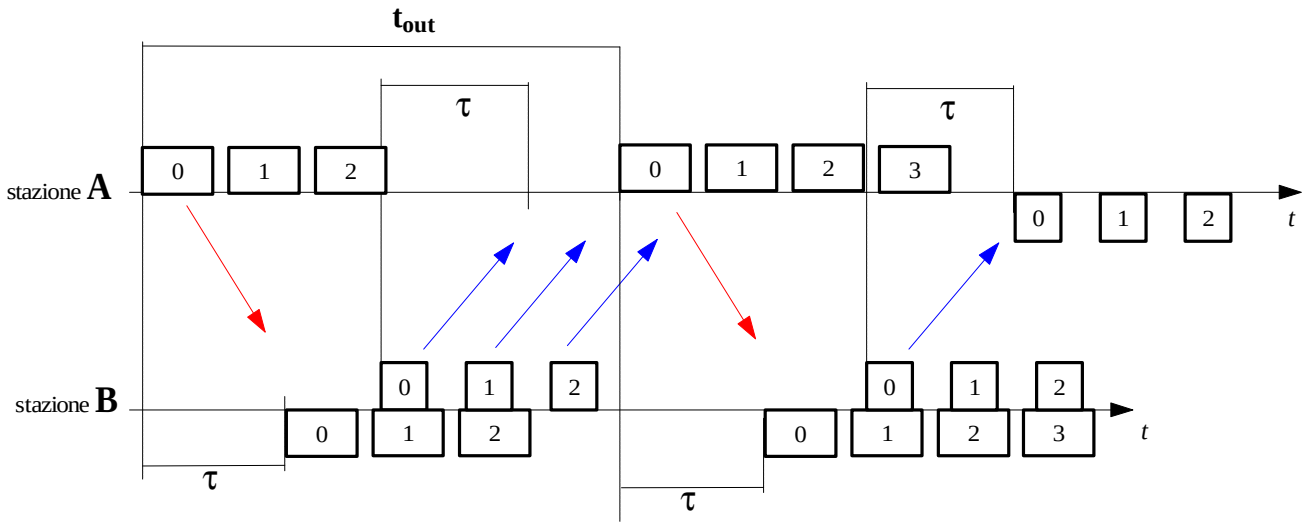
Per capire il motivo di questo vincolo, facciamo un esempio concreto, prendendo n=4 e M=2: abbiamo cioè una finestra di 4 pacchetti e tali pacchetti possono essere numerati da 0 a 3. Supponiamo dunque che la sorgente invii i primi quattro pacchetti (numerati perciò da 0 a 3) e si ponga in attesa del primo riscontro. La destinazione riceve i pacchetti e supponiamo che non rilevi alcun errore, per cui invia 4 successivi riscontri positivi:



Mettiamoci adesso in una condizione molto particolare, già rappresentata in figura: supponiamo cioè che nessuno dei 4 riscontri giunga alla stazione A[1]. Che succede? Succede che la stazione A, avendo attivato un time-out per ognuno dei pacchetti inviati, vede scadere il primo time-out (relativo cioè al pacchetto 0) senza aver ricevuto alcun riscontro. Deduce che c'è stato qualche problema e quindi riinvia per intero la finestra.

I pacchetti ritrasmessi arrivano dunque alla stazione B, la quale incorre in un errore: infatti, in base alla numerazione, essa si aspettava di ricevere pacchetti numerati con 0,1,2,3 e in effetti li riceve; essa, quindi, prende i pacchetti ricevuti come nuovi, perché non ha modo di sapere che invece si tratta dei primi pacchetti ritrasmessi.

L'unico modo per evitare questo problema è quello o di aumentare M o di diminuire n. Ad esempio, diminuiamo la dimensione della finestra, prendendo n=3:



In questo caso, il problema è risolto, in quanto la stazione B, in base alla numerazione, si aspetta di ricevere il pacchetto 3 (infatti, abbiamo supposto che la numerazione sia ancora da 0 a 3), mentre invece vede arrivare un pacchetto numerato con 0. Essa è dunque in grado di capire quello che è successo, per cui si comporta nel modo seguente: scarta tutti i pacchetti, pur inviando i corrispondenti riscontri positivi (dato che si tratta di pacchetti già ricevuti correttamente), fin quando non vede arrivare il pacchetto 3. Quando arriva il pacchetto 3, riprende il funzionamento normale, a meno ovviamente di ulteriori errori.

Analisi di efficienza

Ripetiamo adesso per la tecnica go-back-n gli stessi discorsi visti, per il protocollo stop-and-wait, a proposito dell'efficienza con cui viene utilizzato il canale di trasmissione. E' subito intuitivo aspettarsi migliori prestazioni, se non altro per il fatto che la *linea full-duplex* (necessaria per il protocollo go-back-n) garantisce trasmissioni contemporanee nei due sensi e quindi un maggiore impegno della linea.

Con il protocollo go-back-n, abbiamo visto che la stazione sorgente trasmette con continuità i pacchetti che compongono la finestra, dopo di che aspetta il riscontro del primo dei pacchetti della finestra: se il riscontro è positivo, prende a trasmettere i pacchetti della finestra successiva (mentre

[1] Anche se si tratta di una condizione molto particolare, che difficilmente si può verificare, è comunque necessario prevederla

continuano ad arrivare riscontri), mentre invece, in presenza di un riscontro negativo, riprende a trasmettere dal primo messaggio indicato come errato.

Per fare i conti, facciamo tre ipotesi semplificative:

- la prima è quella di trascurare l'intervallo di tempo che esiste tra l'immissione in linea di un pacchetto e quella del pacchetto successivo: questo significa che, se T è il tempo di trasmissione del singolo pacchetto e *n* la dimensione di una finestra, il tempo di trasmissione di una finestra è nT;
- la seconda ipotesi, più "forte", è quella per cui la scelta della lunghezza n della finestra sia tale per cui, appena la sorgente ha terminato di inviare l'ultimo pacchetto della finestra, arrivi il riscontro al primo pacchetto della stessa finestra.
- l'ultima ipotesi, analoga alla precedente, è che anche il time-out sia scelto esattamente uguale al tempo minimo che la stazione sorgente deve aspettare per ottenere il riscontro al primo pacchetto della finestra: questa ipotesi, unita alla precedente, dice sostanzialmente che la stazione sorgente, appena esaurita la finestra, non debba mai attendere, ma debba solo scegliere se trasmettere un nuovo pacchetto (nel caso sia arrivato un riscontro positivo) oppure ritrasmettere la finestra (nel caso sia arrivato un riscontro negativo o non sia arrivato alcun riscontro).

Così come abbiamo fatto nel caso del protocollo stop-and-wait, consideriamo dapprima il caso migliore, nel quale cioè non si verificano errori durante la trasmissione: in questo caso, è intuitivo che l'efficienza sia uguale ad 1, in quanto abbiamo la linea perennemente impegnata dalla sorgente, senza pause (stiamo infatti trascurando le pause tra un pacchetto e l'altro ed i tempi di elaborazione dei riscontri). Questa rappresenta già una grossa differenza con il protocollo stop-and-wait, nel quale abbiamo invece visto che, anche in assenza di errori (P=0), l'efficienza arrivava ad un valore massimo $E_{max} \cong \frac{1}{1+2a}$ che è comunque inferiore 1.

Adesso consideriamo il caso generale in cui siamo costretti a ritrasmettere una o più volte[1]. Consideriamo il generico pacchetto: per le ipotesi fatte, ci si rende conto che il tempo totale necessario alla sua trasmissione è

$$\begin{aligned}
&\text{no ritrasmissione} \longrightarrow && T \\
&1 \text{ ritrasmissione} \longrightarrow && T + 1 \cdot (T + t_{out}) \\
&2 \text{ ritrasmissioni} \longrightarrow && T + 2 \cdot (T + t_{out}) \\
&3 \text{ ritrasmissioni} \longrightarrow && T + 3 \cdot (T + t_{out}) \\
&\ldots\ldots
\end{aligned}$$

Quindi, se effettuiamo n_R ritrasmissioni (cioè n_R+1 trasmissioni totali) del generico pacchetto, il tempo totale impiegato è

$$T_{TOT} = T + n_R (T + t_{out})$$

[1] Osserviamo che, se n_T è il numero di trasmissioni necessarie per uno stesso pacchetto, sarà sicuramente $n_T=1+n_R$, dove n_R è il numero di ritrasmissioni: infatti, sulle n_T trasmissioni, una sarà quella avvenuta con successo e le altre saranno solo (ri)trasmissioni errate.

Per calcolare l'efficienza, dobbiamo fare il rapporto tra il tempo T di trasmissione del pacchetto sulla linea e il valor medio di T_{TOT}:

$$E = \frac{T}{E[T_{TOT}]} = \frac{T}{E[T + n_R (T + t_{out})]} = \frac{T}{T + E[n_R](T + t_{out})}$$

dove l'ultimo passaggio dipende dal fatto che l'unica quantità aleatoria, non deterministica, in T_{TOT} è proprio n_R.

Dobbiamo dunque calcolare il numero medio di ritrasmissioni: riprendendo ancora una volta un risultato visto nel corso di *Teoria dei Segnali*, affermiamo che $E[n_R] = \frac{P}{1-P}$, dove P è sempre la probabilità che il generico pacchetto sia giunto errato.

Sostituendo nell'espressione di E, otteniamo dunque che

$$E = \frac{T}{T + \frac{P}{1-P} \cdot (T + t_{out})} = \frac{1}{1 + \frac{P}{1-P} \cdot \left(1 + \frac{t_{out}}{T}\right)} = \frac{1-P}{(1-P) + P\left(1 + \frac{t_{out}}{T}\right)} = \frac{1-P}{1 + P\frac{t_{out}}{T}}$$

Possiamo anche fare qualche altro passaggio, in quanto stiamo supponendo che il time-out sia stato posto uguale al suo valore minimo, che sappiamo essere $(t_{out})_{min} = 2\tau + 2T_E + T_R$. Sostituendo allora anche questa espressione, otteniamo

$$E = \frac{1-P}{1 + P\frac{2\tau + 2T_E + T_R}{T}} = \frac{1-P}{1 + P\left(2\frac{\tau}{T} + 2\frac{T_E}{T} + \frac{T_R}{T}\right)}$$

Trascurando anche in questo caso i termini $2T_E/T$ e T_R/T e ricordando che $\tau/T=a$, possiamo concludere che l'**efficienza** di utilizzo del canale del protocollo go-back-n con buona approssimazione vale

$$\boxed{E \cong \frac{1-P}{1+2aP}}$$

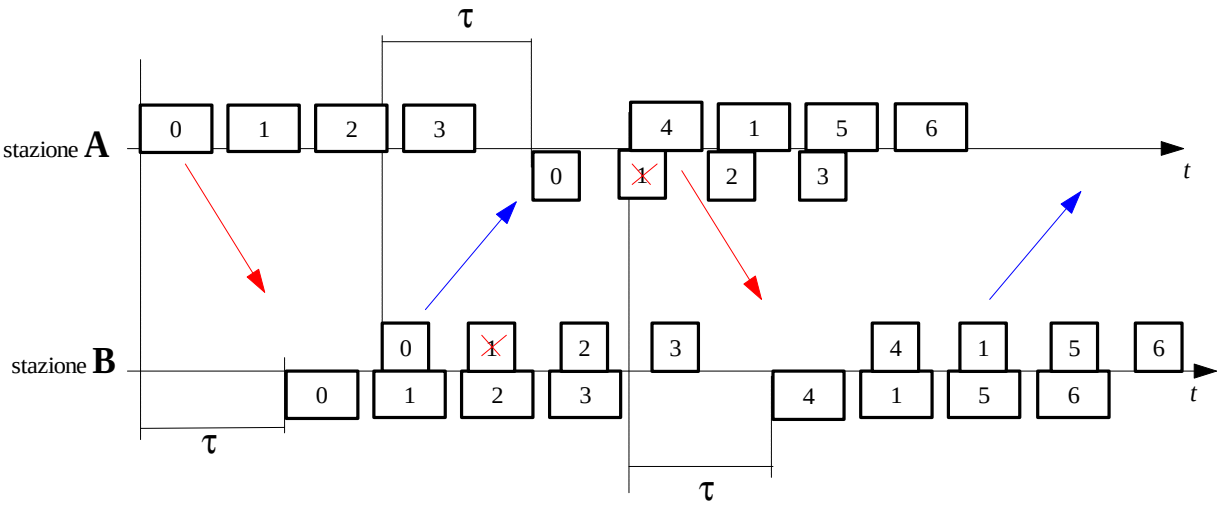
Essendo P molto piccola (dell'ordine di 10^{-9}), è evidente che l'efficienza è in ogni caso molto prossima all'unità. Nel caso ideale di P=0, essa vale proprio 1. Rispetto al protocollo stop-and-wait, il miglioramento è netto.

Nonostante tutta questa analisi sia basata su diverse ipotesi semplificative, è bene sottolineare che i risultati ottenuti sono abbastanza congruenti con la realtà.

Tecnica del selective repeat

Come anticipato in precedenza, *questa tecnica differisce dal go-back-n per il semplice fatto che, in presenza di un riscontro negativo, la sorgente ritrasmette solo il pacchetto che è stato indicato come errato.* Questo è possibile, ancora una volta, grazie alla numerazione dei pacchetti e dei corrispondenti riscontri.

Una possibile situazione è quella della figura seguente:



La sorgente trasmette in sequenza i primi pacchetti, sempre rispettando la dimensione della finestra. La destinazione riceve correttamente il pacchetto 0, per cui invia un riscontro positivo, mentre riceve con errori il pacchetto 1, per cui invia un riscontro negativo; poi riceve i successivi due pacchetti (2 e 3) correttamente e invia i corrispondenti riscontri positivi.

La stazione A riceve il riscontro al primo pacchetto (0) e, osservando che esso è positivo, prende a trasmettere il primo pacchetto (4) della finestra successiva; sennonché, mentre trasmette il pacchetto 4, riceve un riscontro negativo per il pacchetto 1; allora ritrasmette il pacchetto 1. Mentre ritrasmette il pacchetto 1, la sorgente riceve riscontri positivi per i pacchetti 2 e 3, per cui capisce che solo il pacchetto 1 era sbagliato e quindi riprende la trasmissione nell’ordine normale, inviando cioè i pacchetti 5 e 6.

Le considerazioni da fare, per questo protocollo, sono assolutamente simili a quelle del protocollo go-back-n. Si verifica inoltre, nel calcolo dell’efficienza (che qui non facciamo), un ulteriore miglioramento, cioè una efficienza praticamente unitaria sempre.

Un protocollo di questo tipo, come già detto, è opportuno per linee con basso *tasso di errore* (**bit error rate**). D’altra parte, l’implementazione di un simile protocollo è più complicata e quindi più costosa, in quanto sia in trasmissione sia in ricezione è necessario un accurato controllo della numerazione dei pacchetti e dei riscontri; in particolare, in ricezione, dati gli inevitabili errori e le corrispondenti ritrasmissioni, i pacchetti non saranno mai ordinati correttamente, per cui bisognerà poterli riordinare a sufficiente velocità.

Autore: **SANDRO PETRIZZELLI**
e-mail: sandry@iol.it
sito personale: http://users.iol.it/sandry
succursale: http://digilander.iol.it/sandry1