Il data link deve supportare i servizi richiesti dal livello Network:

Connectionless non confermato (comune nelle LAN)

Connectionless confermato (utile su canali non affidabili tipo wireless)

Connection oriented confermato

Comunicazioni *connectionless non confermate*

- non si stabilisce una connessione;
- si inviano frame;
- i frame non vengono confermati;
- i frame persi non si recuperano;

appropriate per:

- canali con tasso d'errore molto basso;
- traffico real-time (es. voce);
- LAN (è infatti molto comune).

Comunicazioni *connectionless confermate*

* i frame vengono confermati;
* se la conferma non arriva, il mittente rispedisce il frame;

La perdita di ack causa la trasmissione di più copie dello stesso frame.

La conferma a questo livello è un'ottimizzazione, perché può sempre essere fatta a livello superiore, ma con linee disturbate risulterebbe molto più gravosa.

Comunicazioni ***connection oriented confermate***

E’ il servizio più sofisticato e prevede tre fasi:

- apertura connessione;
- invio dati;
- chiusura connessione;

Ogni frame è ricevuto una sola volta e nell'ordine giusto (fornisce al livello network un flusso di bit affidabile).

Il data link, quindi, oltre al framing, rilevazione e correzione di errori deve anche garantire, se richiesto dal servizio, la ricezione ordinata dei frame spediti.

Servizi connectionless non confermati: non c'è bisogno di alcuna conferma;

Servizi connectionless confermati: sono arrivati tutti e senza errori?

Servizi connection oriented confermati: sono arrivati tutti, senza errori e nell'ordine giusto?

Acknowledgement (messaggio inviato dal destinatario al mittente)

positive ack: il frame è arrivato correttamente;

negative ack (***Nack***) : il frame è errato.

Frame dati: frame che trasporta informazioni generate nel colloquio fra le peer entity;

Frame di ack: frame il cui solo scopo è trasportare un acknowledgement.

Con l'ack sorgono alcuni problemi:

Un frame può anche sparire del tutto, per cui il mittente rimane bloccato in attesa di un ack che non arriverà mai (*Il mittente stabilisce un time-out per la ricezione dell'ack. Se questo non arriva in tempo, ritrasmette di nuovo il frame*).

Se sparisce l'ack, il destinatario può trovarsi due (o più) copie dello stesso frame (*Il mittente inserisce un numero di sequenza all'interno di ogni frame dati*).

Un altro aspetto importante è il controllo del flusso

Feedback dal destinatario al mittente per impedire che il mittente spedisca dati più velocemente di quanto il destinatario sia in grado di ricevere.

Meccanismi basati sull'esplicita autorizzazione: il destinatario informa il mittente riguardo il numero di frame che può ricevere.

In trasmissione:

*quando il livello data link riceve un pacchetto dal livello network, lo incapsula tra un ***header*** ed un ***trailer***;

*calcola (con un apposito HW del livello data link) il ***checksum*** e i ***delimitatori***;

*infine passa il frame al livello sottostante, che trasmette il tutto;

In ricezione:

l' HW di livello data link identifica i delimitatori, estrae il frame, ricalcola il checksum:

se è sbagliato, il SW di livello data link viene informato dell'errore;

altrimenti il SW di livello data link riceve il frame (senza checksum)

In ricezione:

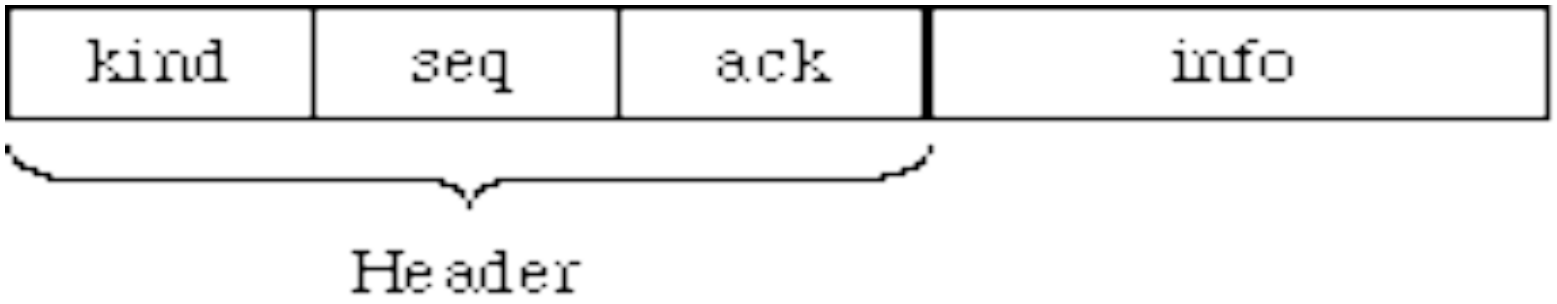
l' HW di livello data link identificati i delimitatori, ed estratto il frame, individua l'occorrenza di eventuali errori ricalcolando il checksum.

il SW esaminate le informazioni di controllo (l'header ed il trailer):

se tutto è OK consegna il pacchetto, e solo quello, al livello network;

altrimenti non consegna il pacchetto ma attiva le operazioni previste per il caso.

Struttura di un frame:



kind: tipo di frame (dati, controllo, ecc.)

seq: numero progressivo del frame

ack : informazioni legate all'acknowledgement

info: pacchetto di livello network completo.

Protocollo Heaven

Mittente (loop infinito):

1) attende un pacchetto dal livello network;
2) costruisce un frame dati;
3) passa il frame al livello fisico;
4) torna ad 1).

Destinatario (loop infinito):

1) attende l'arrivo di un frame da livello fisico:
2) estrae il pacchetto;
3) lo passa al livello network;
4) torna ad 1).

Il protocollo Heaven ipotizza:

* canali simplex (cioè i frame vengono trasmessi in **una sola direzione**);

*i livelli network non devono mai attendere per inviare e/o ricevere al/dal livello data link;

*il **tempo di elaborazione** del livello data link è **nullo**;

*il ricevitore ha un **buffer infinito**;

*il canale fisico è **esente da errori**.

Se il ricevitore non ha un buffer infinito, il mittente deve essere opportunamente rallentato.

O si ***ritardano le trasmissioni***, effettuandole ogni Δt (Δt va calibrato sul caso peggiore), oppure si utilizza un protocollo ***stop-and-wait***.

Protocollo ***stop and wait*** (canale almeno half-duplex):

il mittente invia il prossimo frame solo se esplicitamente autorizzato dal destinatario

Protocollo *stop and wait*

Mittente (loop infinito):
1) attende l'arrivo di un pacchetto dal livello network;
2) costruisce un frame dati;
3) passa il frame al livello fisico;
4) attende l'ack;
5) torna ad 1).

Destinatario (loop infinito):
1) attende l'arrivo di un frame dati da livello fisico:
2) estrae il pacchetto;
3) consegna il pacchetto al livello network;
4) invia un frame di ack al mittente;
5) torna ad 1).

In un canale rumoroso un frame può essere

danneggiato: l'HW di controllo (checksum) lo rileva ed informa il SW di livello data link;

perso: non è possibile rilevare la perdita di un frame.

Possibile soluzione

mittente: quando invia un frame dati, fa anche partire un *timer*. Alla scadenza del timer se l'ack per il frame trasmesso non è ancora arrivato, reinvia il frame;

destinatario: per ogni frame dati arrivato senza errori invia un ack

Se il frame dati x arriva bene, ed il relativo ack si perde, il frame dati x viene inviato di nuovo.

Per riconoscere eventuali doppioni, il mittente mette nel campo seq dell'header il numero di sequenza del frame dati inviato.

Poiché l'ambiguità in ricezione può aversi solo tra un frame ed il successivo, come numero di sequenza può essere impiegato anche un unico bit.

Successione di numeri di sequenza: ...01010101...

Il mittente, che trasmette i frame dati alternando 0 ed 1 nel campo seq, passa a trasmettere il prossimo frame solo quando riceve l'ack di quello precedente.

Il destinatario invia un frame di ack per tutti quelli ricevuti senza errori, ma passa al livello network solo quelli con il numero di sequenza atteso.

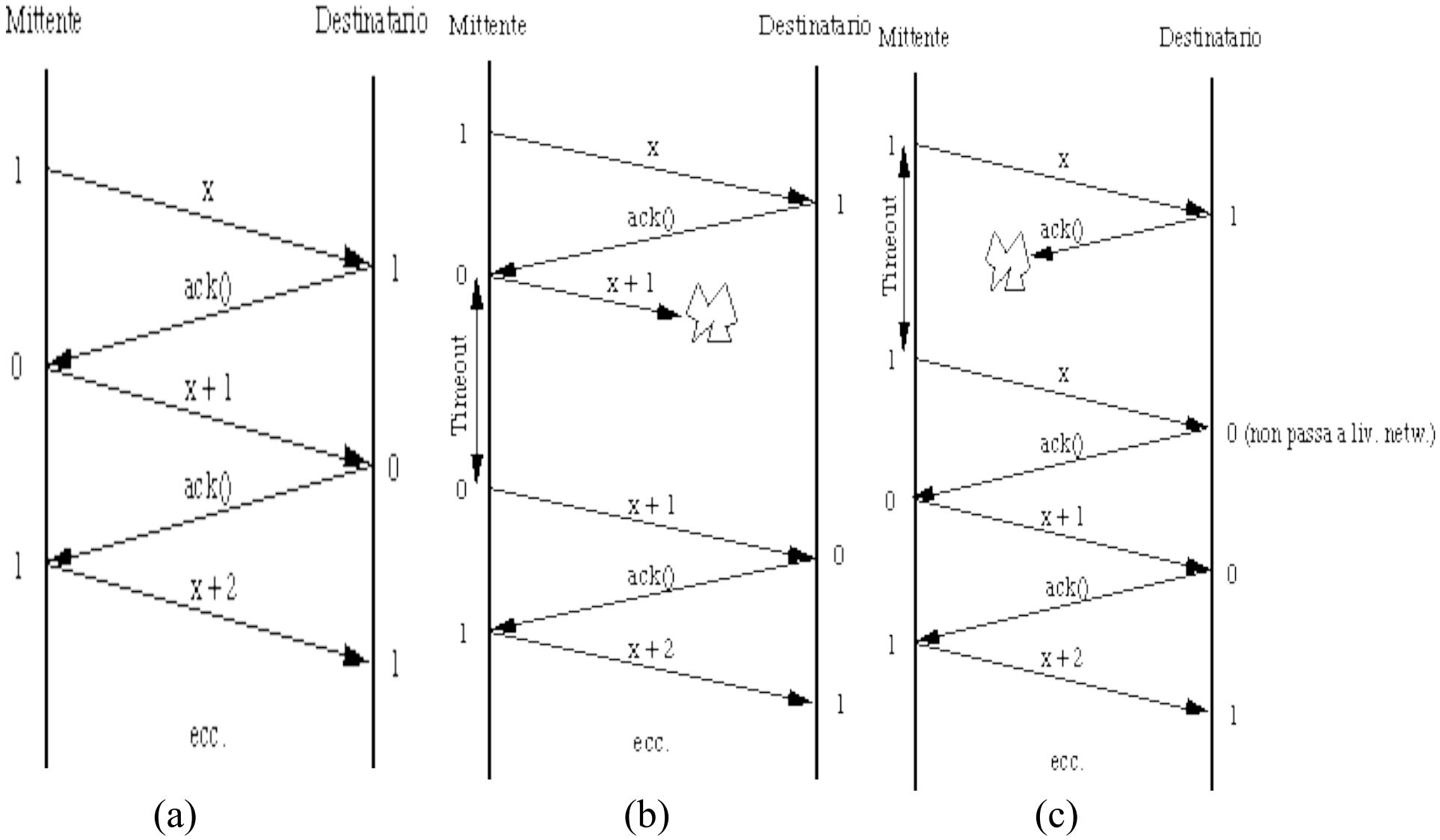
Mittente (loop infinito):

0) n_seq = 0;
1) n_seq = 1 - n_seq;
2) attende un pacchetto dal livello network;
3) costruisce frame dati e copia n_seq in [seq];
4) passa il frame dati al livello fisico;
5) resetta il timer;
6) attende un evento:
    * timer scaduto: torna a 4)
    * arriva frame di ack non valido: torna a 4)
    * arriva frame di ack valido: torna ad 1)

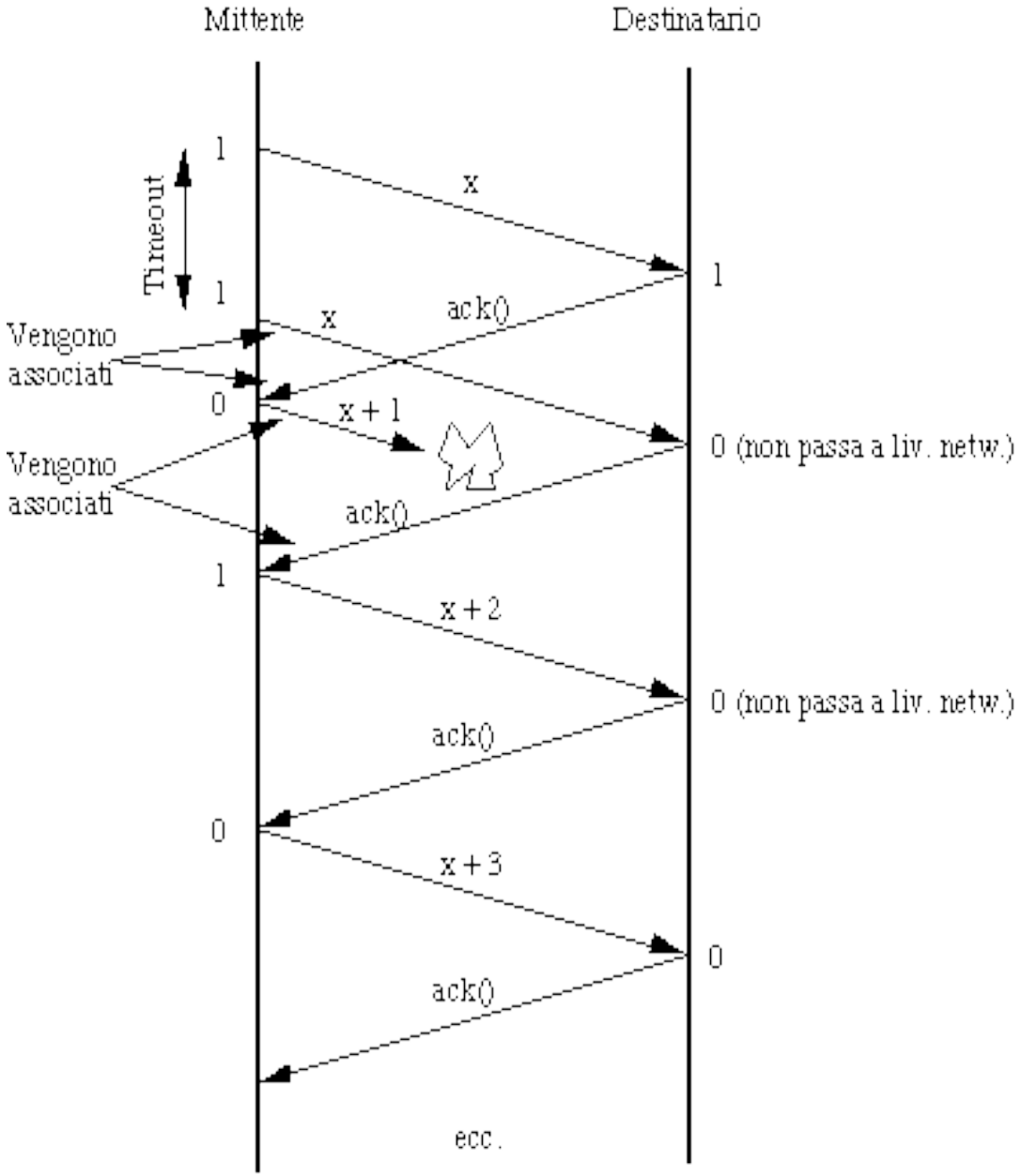
Destinatario (loop infinito):

0) n_exp = 1;
1) attende evento;
    * arriva un frame dati valido dal livello fisico:
        2) se ([seq] == n_exp)
            2.1) estrae pacchetto
            2.2) lo consegna al livello network
            2.3) n_exp = 1 - n_exp
        3) invia frame di ack
        4) torna ad 1)

    * arriva frame non valido: torna ad 1)

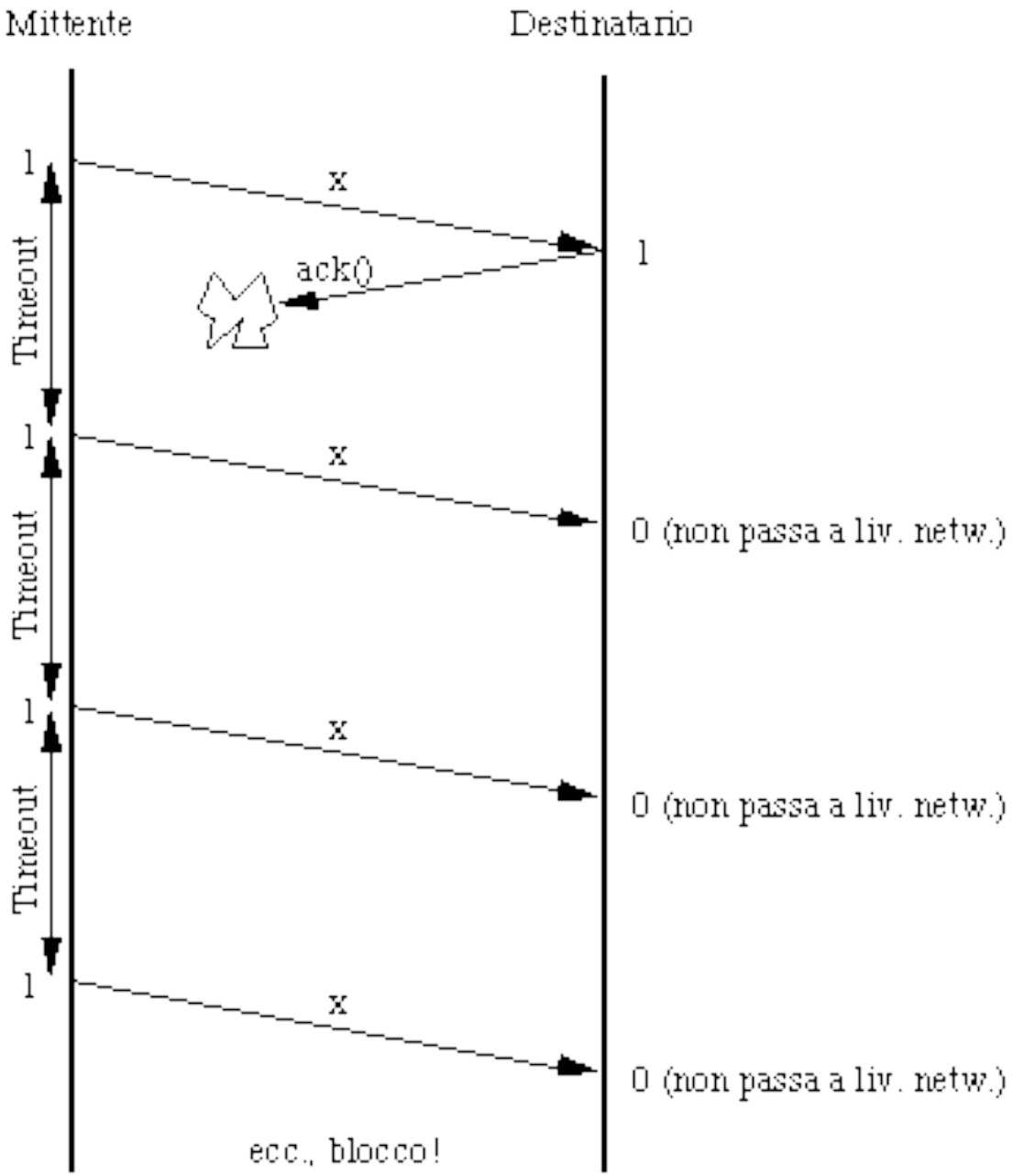


(a)Normale funzionamento; Perdita (o danneggiamento) (b) di un frame (c)di un ack



Timeout troppo ridotto:
i frame dati (x+1) e (x+2) si perdono

Se (x+1) si perde, l'ack inviato per la seconda copia di x potrebbe essere scambiato per l'ack di (x +1) e quindi non ci si accorgerebbe che quest'ultimo è perso. Ovviamente (x+2), sebbene ricevuto regolarmente sarà non passato al livello rete perché il numero di sequenza non è corretto, e quindi (x+2) viene scambiato per una copia di x.



Ovviamente il problema non lo si risolve se si invia l'Ack per i soli frame non duplicati. Infatti la perdita di ack causerebbe un blocco dal lato mittente che non ricevendo l'ack continuerebbe ad inviare ininterrottamente il frame per il quale non avrebbe mai riscontro.