Corso di Laurea in Ingegneria Informatica

Corso di Reti di Calcolatori I

Roberto Canonico (roberto.canonico@unina.it)
Giorgio Ventre (giorgio.ventre@unina.it)

Il livello trasporto:

tecniche di trasmissione affidabile dei dati

I lucidi presentati al corso sono uno strumento didattico che NON sostituisce i testi indicati nel programma del corso

Nota di copyright per le slide COMICS

Nota di Copyright

Autori:
Simon Pietro Romano, Antonio Pescapè, Stefano Avallone,
Marcello Esposito, Roberto Canonico, Giorgio Ventre

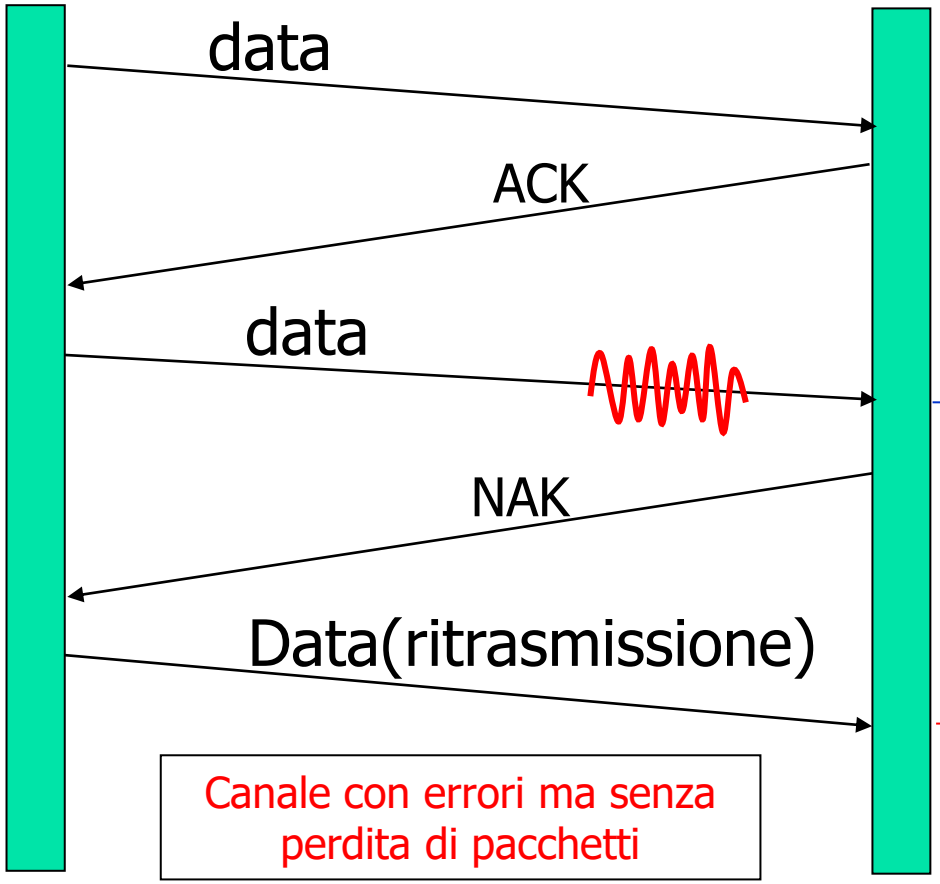
Realizzare una trasmissione affidabile

- Se il livello rete è inaffidabile, nella comunicazione end-to-end si potrà verificare:
  - Presenza di errori
  - Perdita di pacchetti
  - Ordine dei pacchetti non garantito
  - Duplicazione di pacchetti
- Il livello trasporto si può fare carico di rimediare a queste circostanze a favore delle applicazioni
- Inoltre al livello trasporto possono essere realizzati dei meccanismi che tengano in considerazione:
    - i buffer del computer ricevente sono di capacità limitata:
      - Controllo di flusso
    - I buffer dei router sono di capacità limitata
      - Controllo di congestione

Realizzare una trasmissione affidabile - 2

- Soluzioni:
- Rete che presenta errori di trasmissione:
  - In questo caso il ricevente deve effettuare:
    - Rilevamento degli errori e:
      - 1) Correzione degli errori
        oppure
      - 2) Notifica al mittente
        Richiesta ritrasmissione
  - La prima soluzione introduce complicazioni, la seconda introduce possibili duplicazioni sulla rete che il ricevente non è in grado di interpretare

Realizzare una trasmissione affidabile - 3



ACK = notifica

NAK = notifica negativa

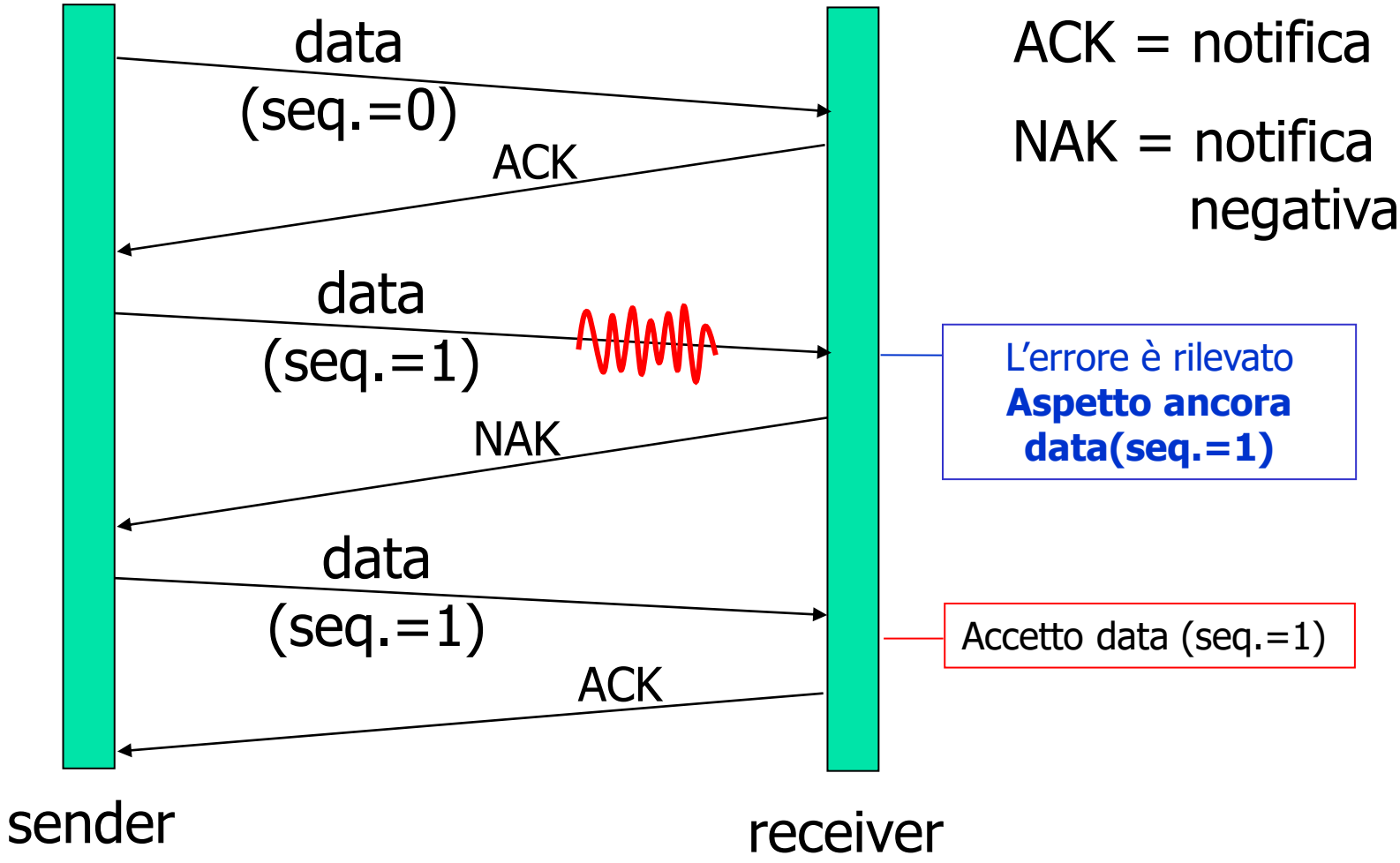
L'errore è rilevato

Il ricevente non può sapere se la richiesta di ritrasmissione è arrivata al sender, quindi non sa interpretare se quello appena ricevuto è il segmento ritrasmesso dal sender, o un segmento contenente nuovi dati

Realizzare una trasmissione affidabile - 4

- Per risolvere il problema dei duplicati che il ricevente non è in grado di interpretare, occorre inserire nell'header del segmento da inviare un' ulteriore informazione:
  - **numero di sequenza**
- Nel caso di protocolli che inviano un messaggio e quindi aspettano un riscontro prima di ritrasmettere un nuovo messaggio (stop & wait), è sufficiente un numero di sequenza su un bit (0,1). Vediamo un esempio:

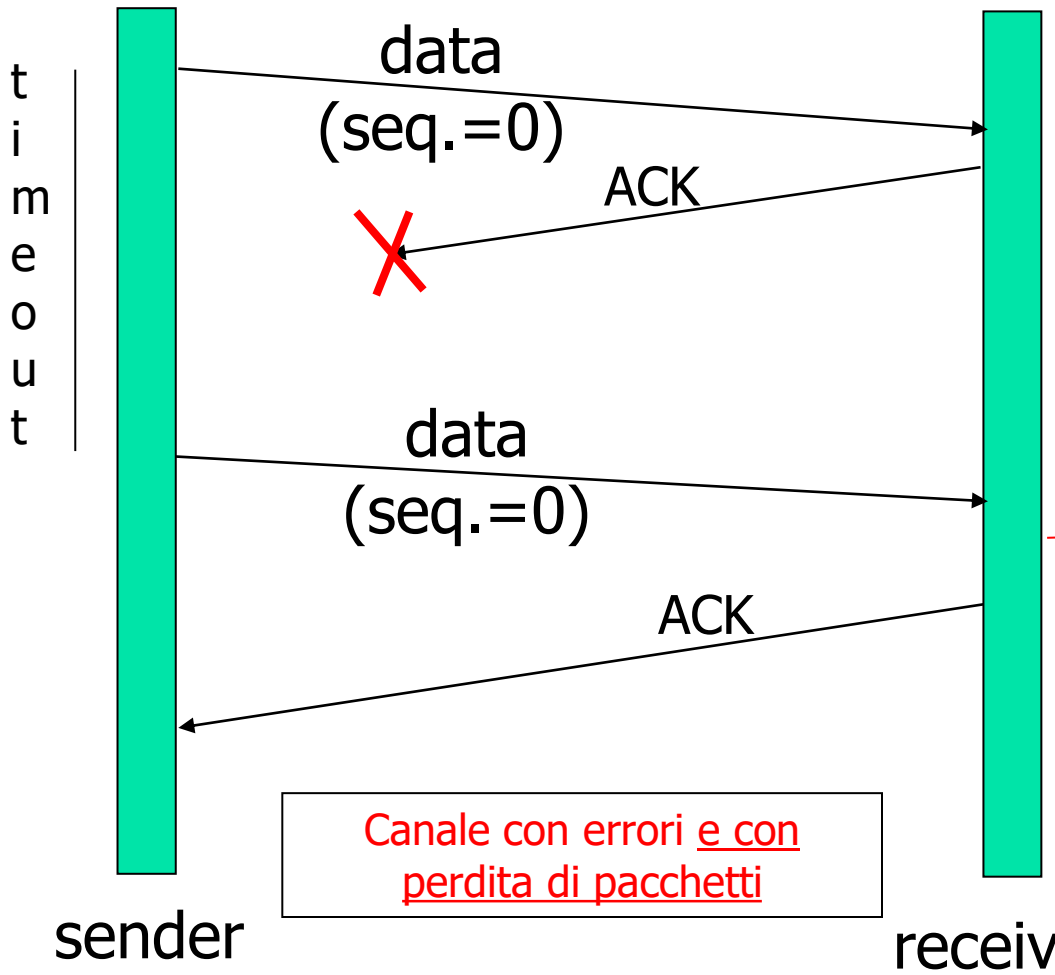
Realizzare una trasmissione affidabile - 5



Un protocollo senza NAK

- Stessa funzionalità del precedente, utilizzando soltanto gli ACK
- Al posto del NAK, il destinatario invia un ACK per l'ultimo pacchetto ricevuto correttamente
  - il destinatario deve includere *esplicitamente* il numero di sequenza del pacchetto con l'ACK
- Un ACK duplicato presso il mittente determina la stessa azione del NAK: *ritrasmettere il pacchetto corrente*

Realizzare una trasmissione affidabile - 6



Canale con errori e con perdita di pacchetti

Nel caso di canale che introduce perdita di pacchetti, è necessario introdurre un altro parametro: il tempo

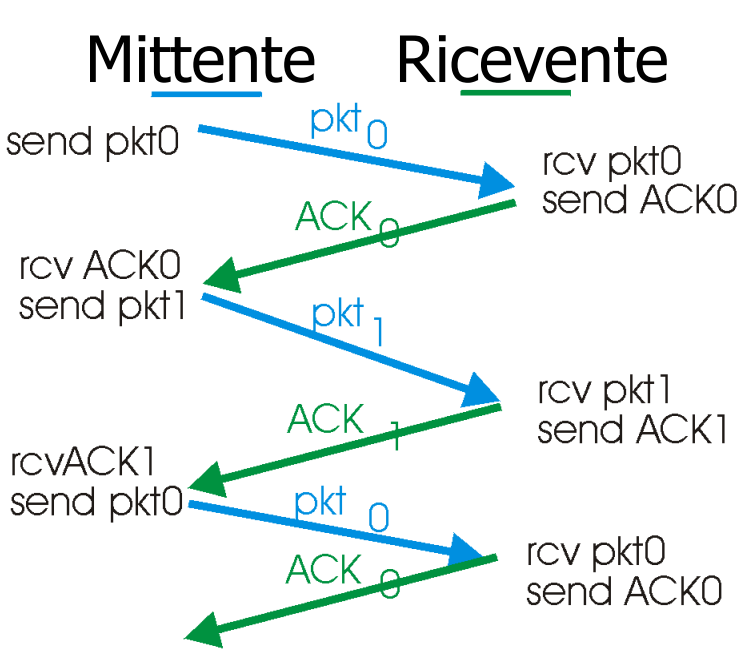
In particolare un conto alla rovescia: timeout

Si può fare a meno dei NAK

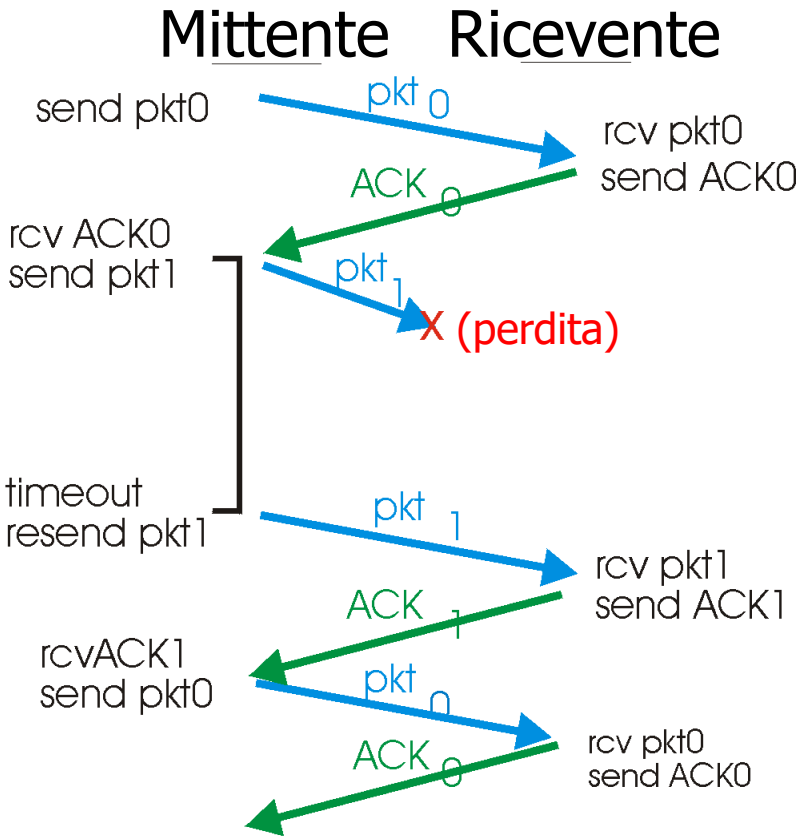
Il receiver si accorge di aver già ricevuto data (seq.=0), scarta tale segmento e invia nuovamente l'ACK al mittente

Problema: quanto dovrà essere grande il timeout?

Stop&Wait: Ricapitolando (1/2)

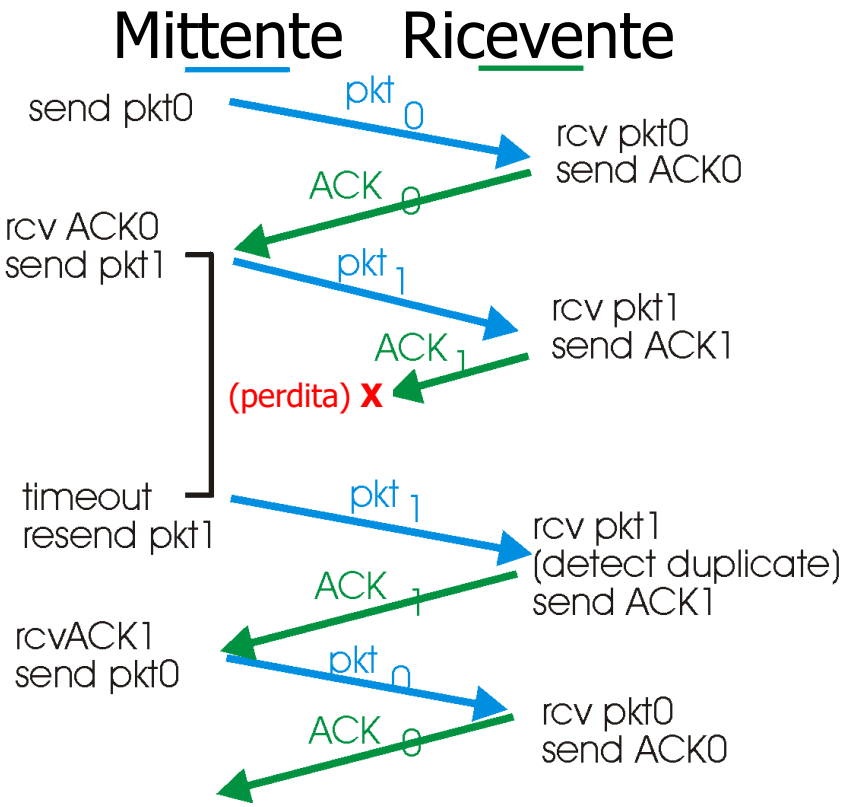


a) Operazioni senza perdite

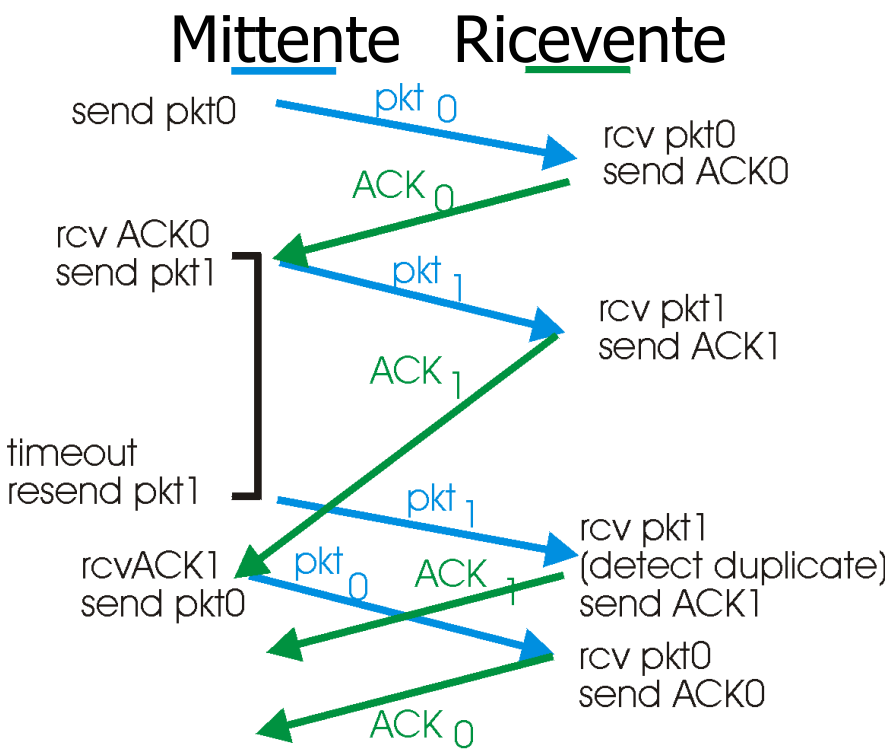


b) Perdita di pacchetto

Stop&Wait: Ricapitolando (2/2)



c) Perdita di ACK



d) Timeout prematuro

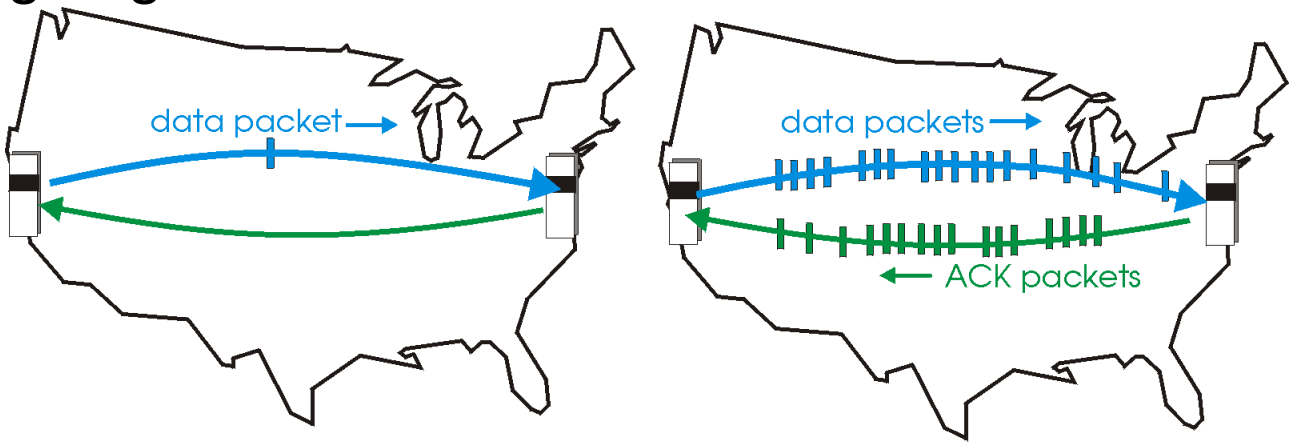
... anche detto "protocollo ad alternanza di bit"

Aumentare l'efficienza

In alternativa al semplice Stop & Wait …

Pipelining: il mittente invia pacchetti prima di ricevere il riscontro dei precedenti

- Occorre aumentare l'intervallo dei num. sequenza
- Aggiungere buffer nel sender e/o receiver



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

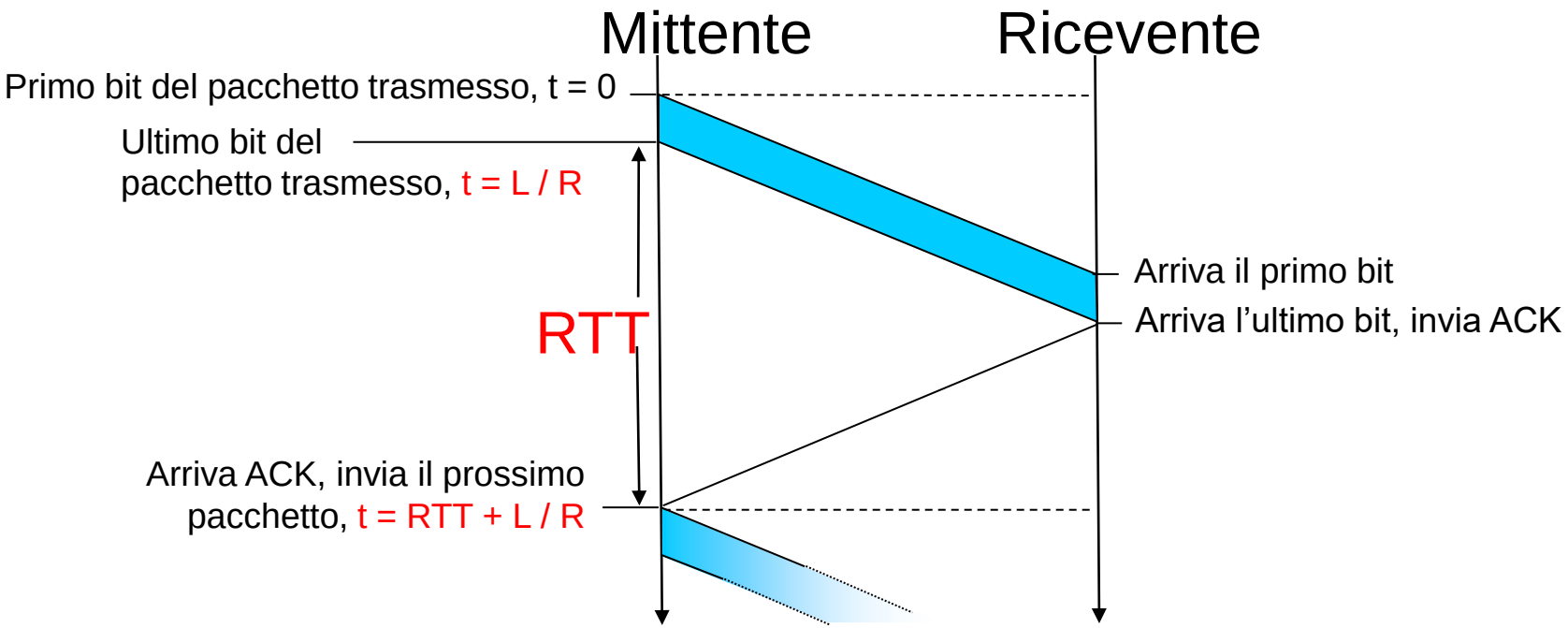
- Due alternative per il pipelining: *go-Back-N, selective repeat*

Performance

- Funziona, ma le performance…
- Esempio: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet:

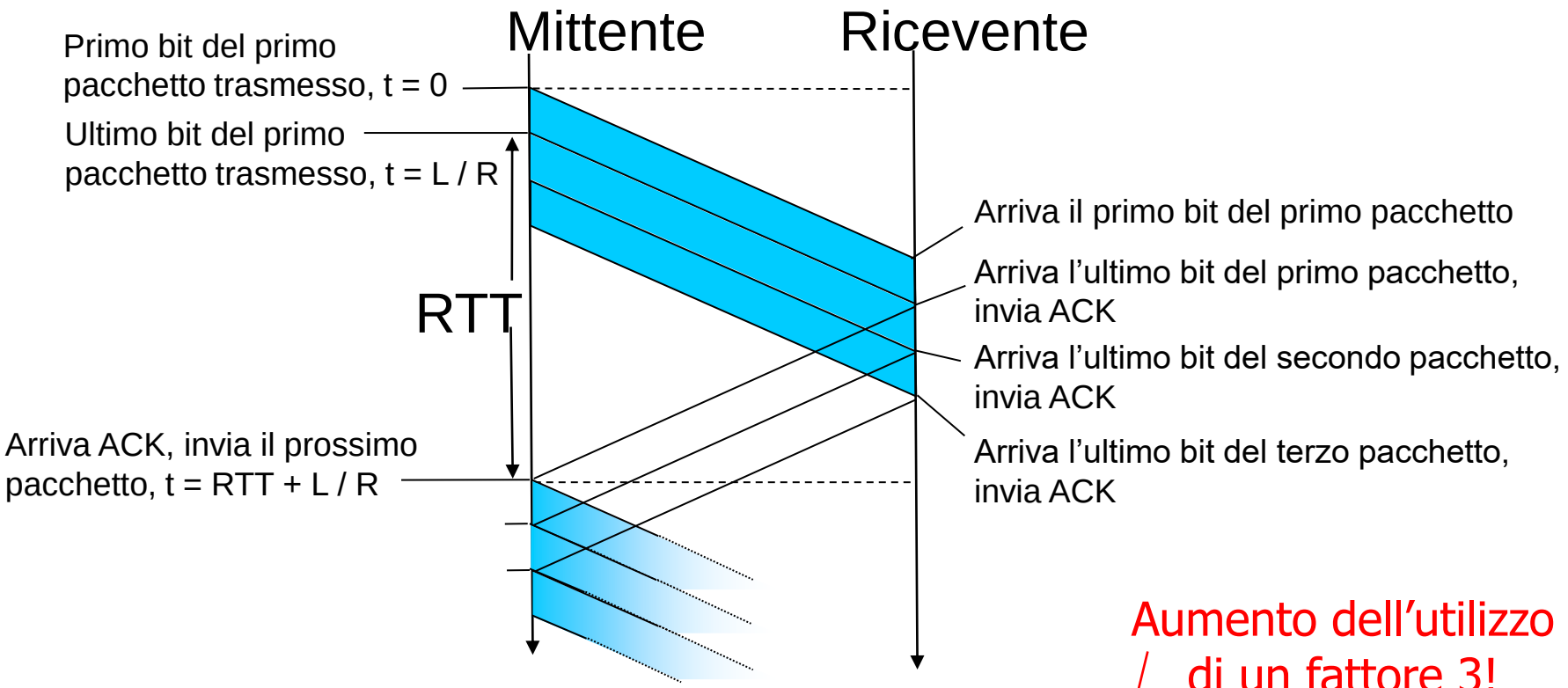
$$T_{transmit} = \frac{\text{L (packet length in bits)}}{\text{R (transmission rate, bps)}} = \frac{\text{8kb/pkt}}{\text{10**9 b/sec}} = 8 \text{ microsec}$$

Funzionamento con stop-and-wait



$$U_{mitt} = \frac{L / R}{RTT + L / R} = \frac{0{,}008}{30{,}008} = 0{,}00027 = 0{,}027\ \%$$

Pipelining: aumento dell'utilizzo



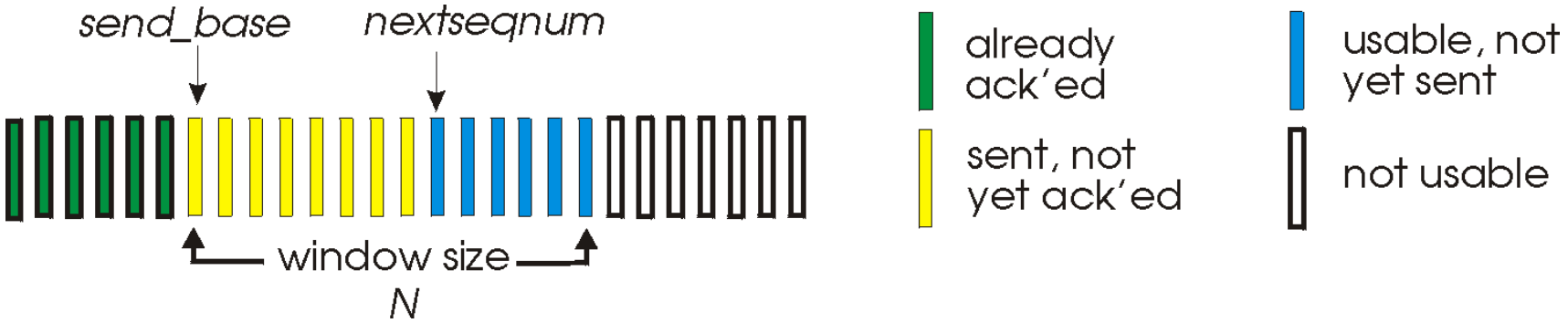
Aumento dell'utilizzo di un fattore 3!

$$U_{mitt} = \frac{3 * L / R}{RTT + L / R} = \frac{0{,}024}{30{,}008} = 0{,}0008$$

Go Back-N

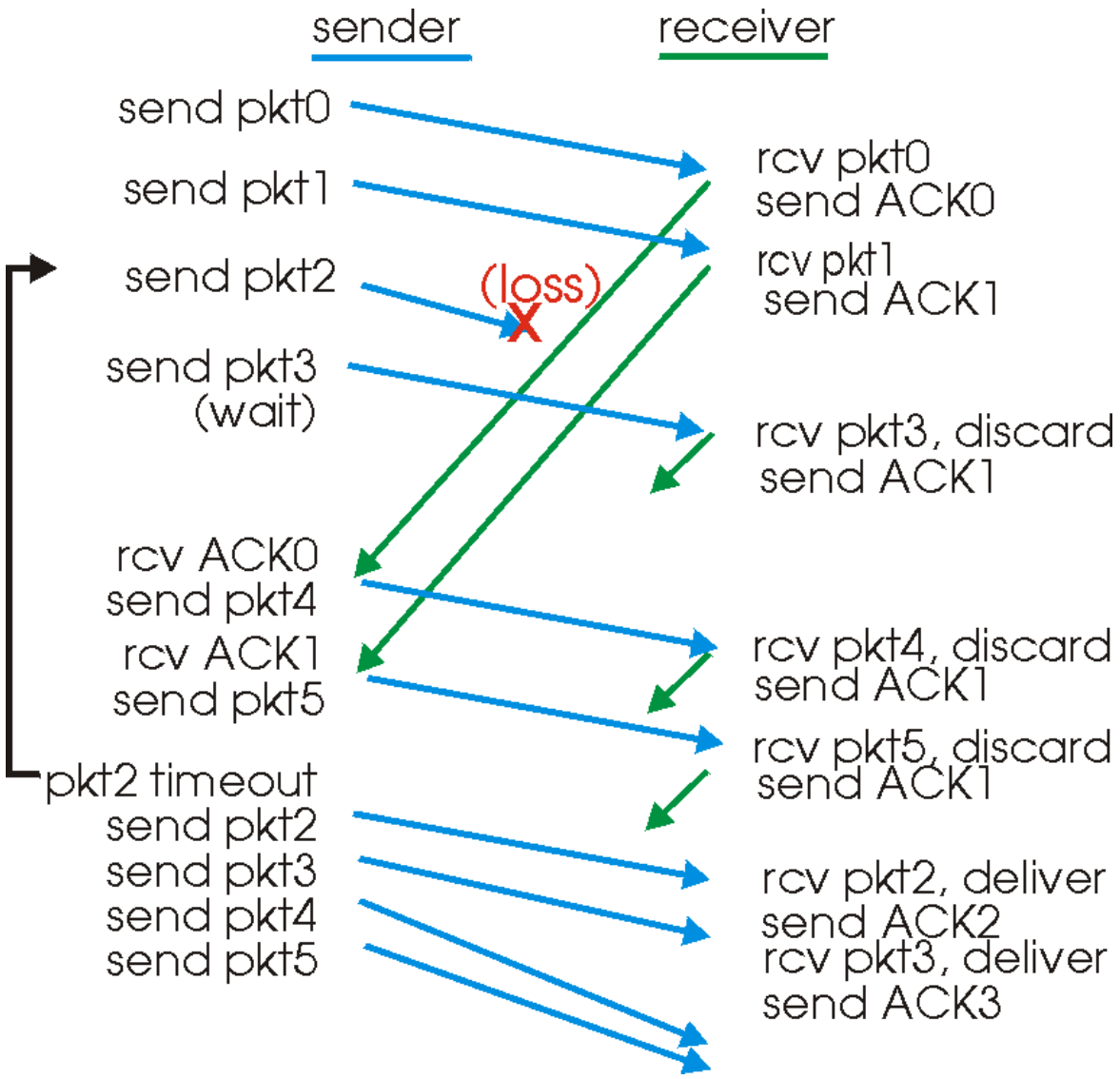
Sender:

- Nell'header del segmento k-bit per il num. sequenza
- Una finestra di max N pacchetti senza riscontro
- ACK numerati



- ACK cumulativo: ricevere ACK(n) significa che tutti i pkts precedenti l'n-esimo sono stati ricevuti correttamente
- Un solo timer per il primo pacchetto trasmesso e non ancora riscontrato
- *timeout(n):* ritrasmetti pkt n e tutti i pacchetti che seguono n
- Il ricevente non deve accumulare i pacchetti arrivati: se il pacchetto arrivato non è quello atteso, il pacchetto è scartato

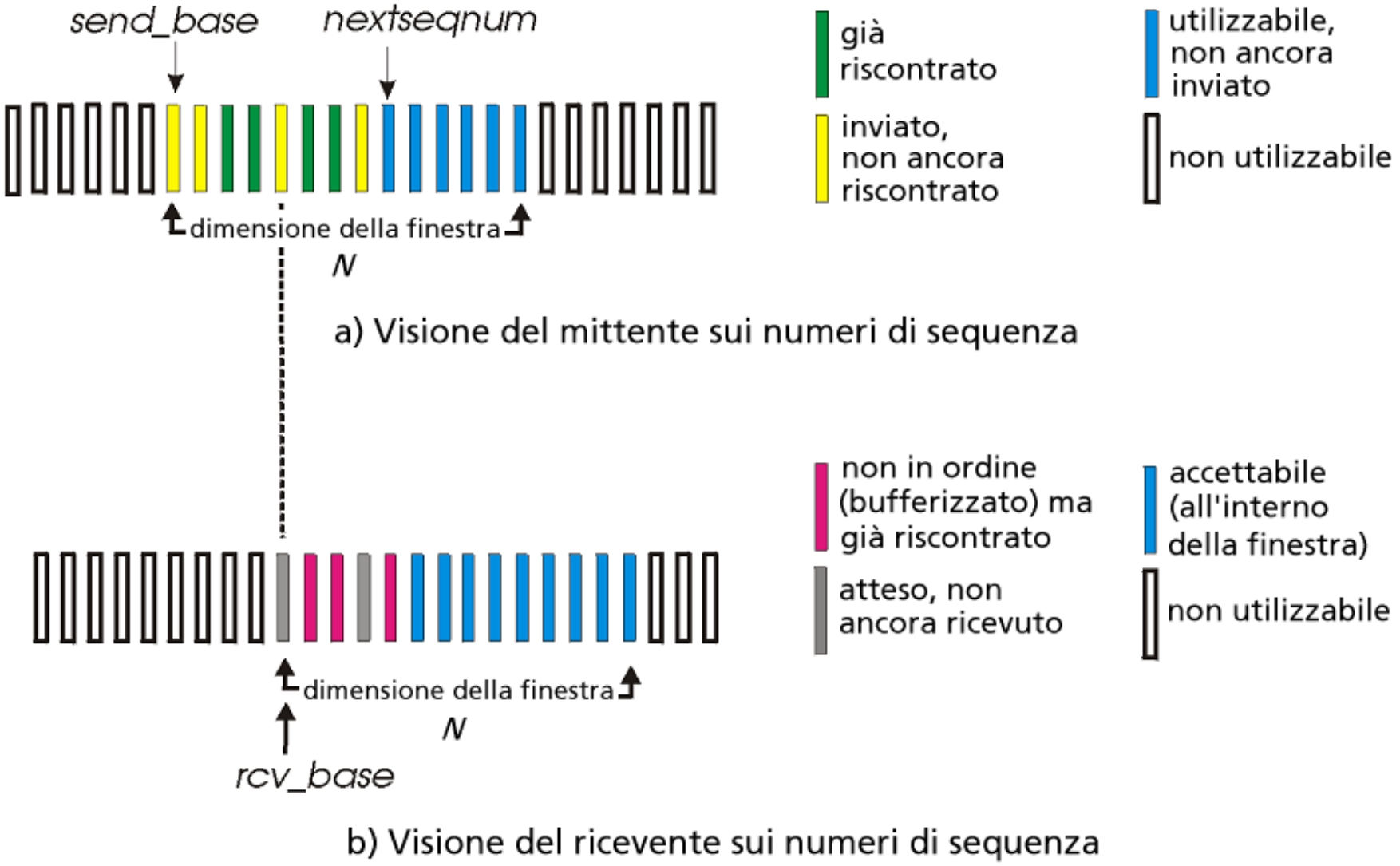
Go Back-N in azione



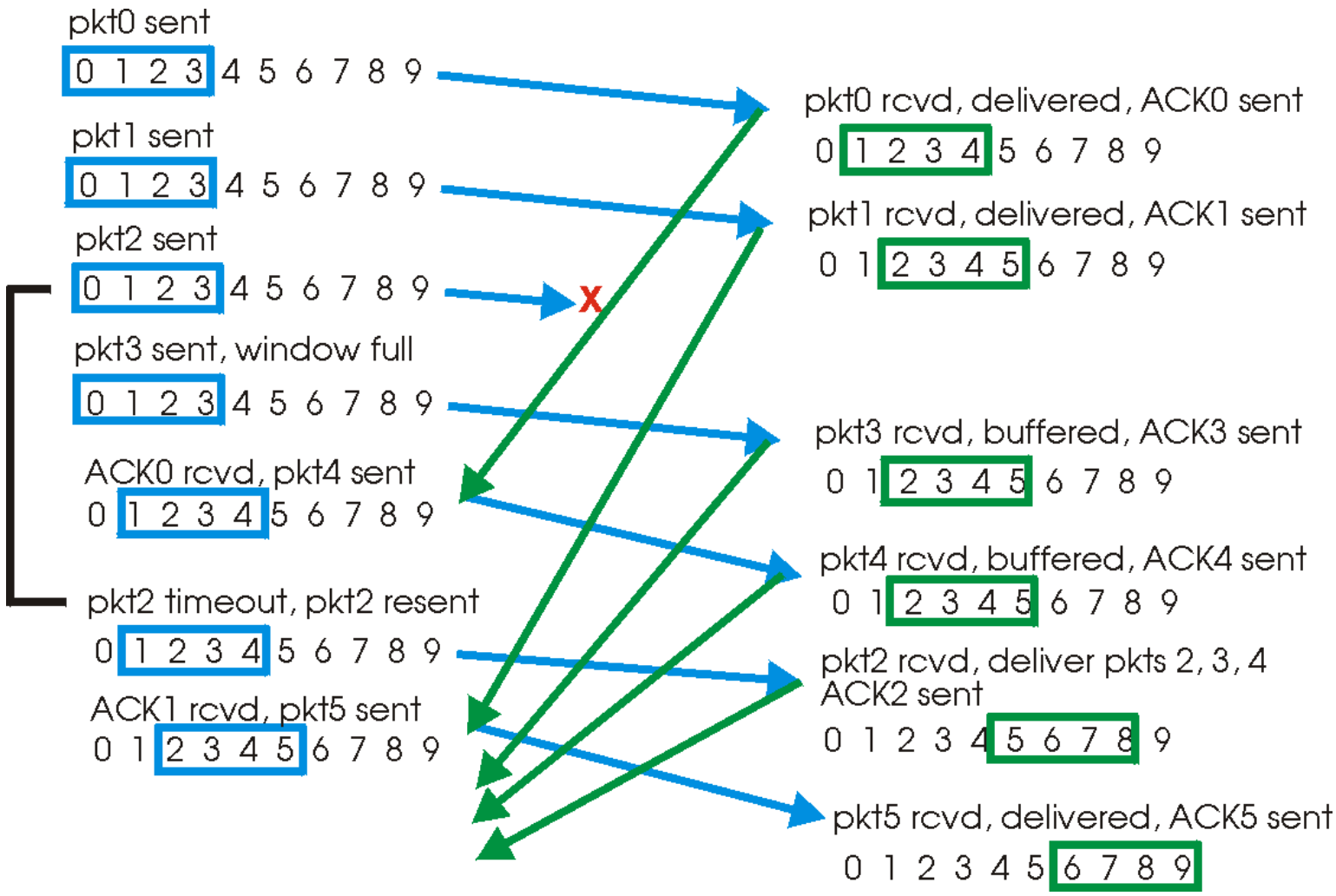
Ripetizione selettiva

- Il ricevente invia riscontri *specifici* per tutti i pacchetti ricevuti correttamente
  - buffer dei pacchetti, se necessario, per eventuali consegne in sequenza al livello superiore
- Il mittente ritrasmette soltanto i pacchetti per i quali non ha ricevuto un ACK
  - timer del mittente per ogni pacchetto non riscontrato
- Finestra del mittente
  - N numeri di sequenza consecutivi
  - limita ancora i numeri di sequenza dei pacchetti inviati non riscontrati

Selective Repeat



Selective Repeat in azione

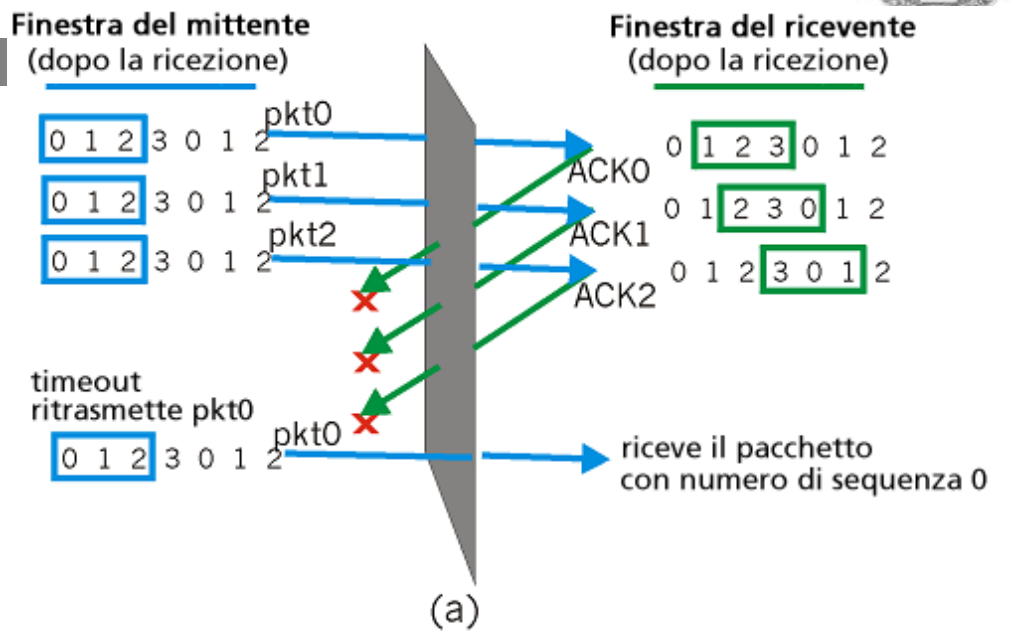


Ripetizione selettiva: dilemma

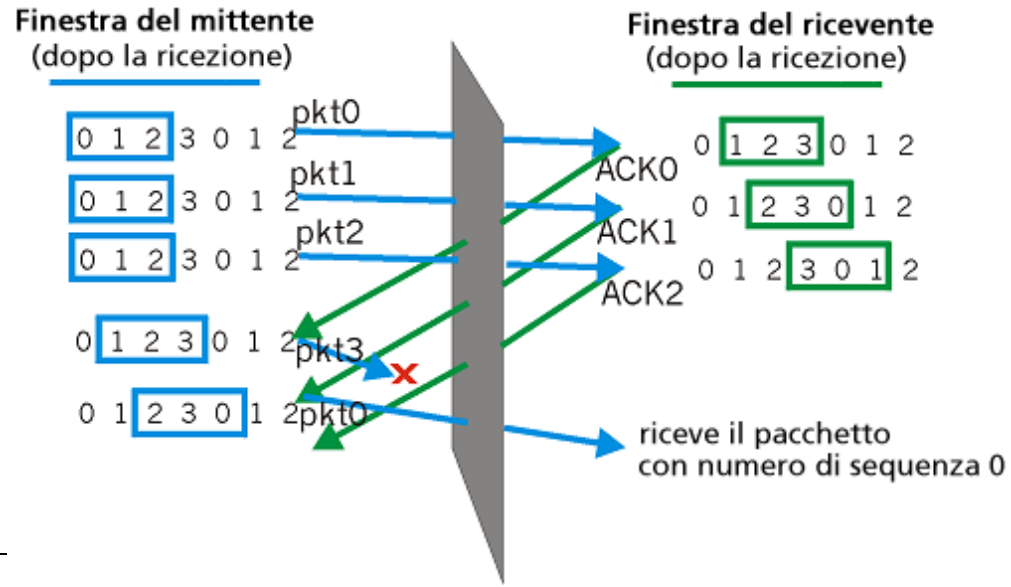
Esempio:

- Numeri di sequenza: 0, 1, 2, 3
- Dimensione della finestra = 3

- Il ricevente non vede alcuna differenza fra i due scenari!
- Passa erroneamente i dati duplicati come nuovi in (a)



(a)



(b)