



POLITECNICO  
MILANO 1863

# Fondamenti di TELECOMUNICAZIONI

Prof. Marco Mezzavilla

# Lezione 11 – Livello di Trasporto 2

# INDICE

## 11. LIVELLO DI TRASPORTO 2

### 1. Servizio di trasporto

### 2. Protocollo UDP

### 3. Trasporto affidabile I

1. Stop & Wait

Parte I (ieri)

### 4. Trasporto affidabile II

2. Go-Back-N
3. Selective Repeat
4. Controllo di flusso a finestra mobile

Parte II (oggi)

### 5. Protocollo TCP

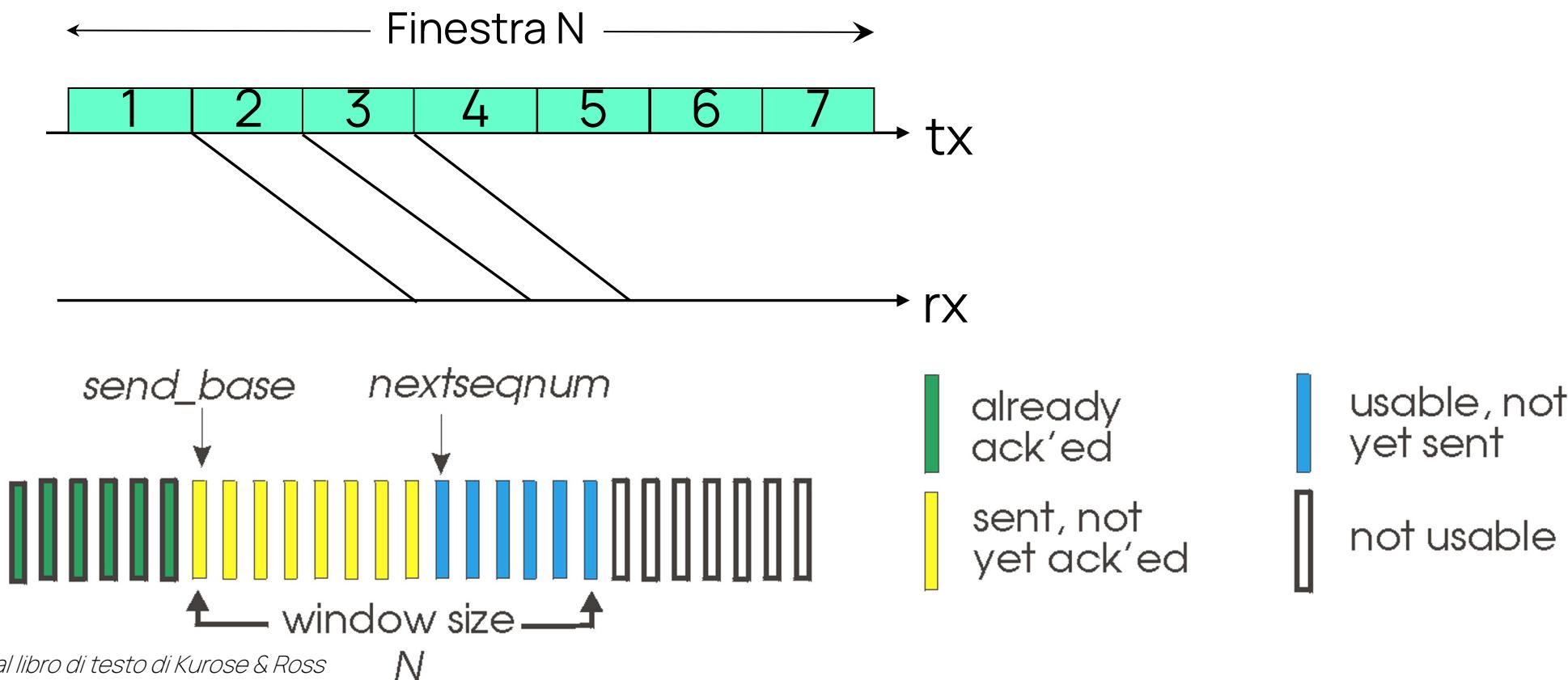
Parte III (domani)

# GO-BACK-N

01

# Protocollo Go-Back-N

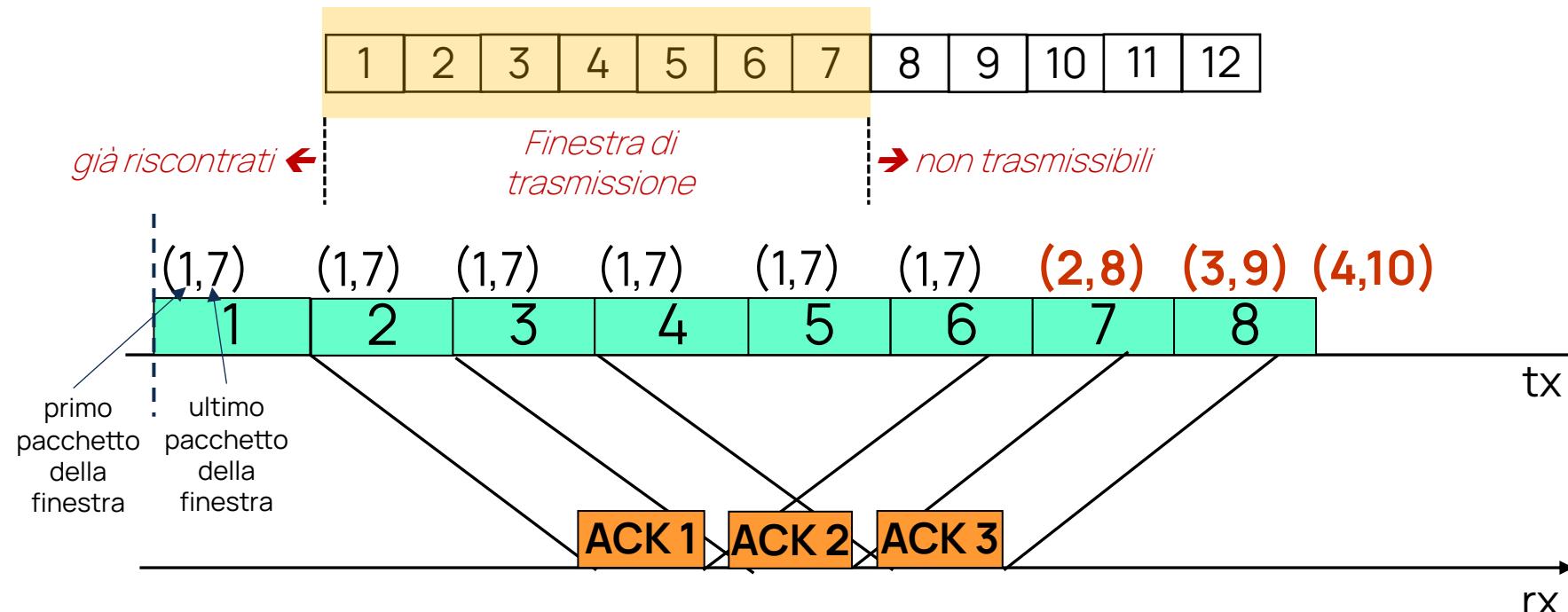
- ❑ Variante rispetto allo *Stop and Wait*
- ❑ Si possono trasmettere fino a  $N$  pacchetti (finestra) senza aver avuto il riscontro



\*tratto dal libro di testo di Kurose & Ross

# Finestra Go-Back-N

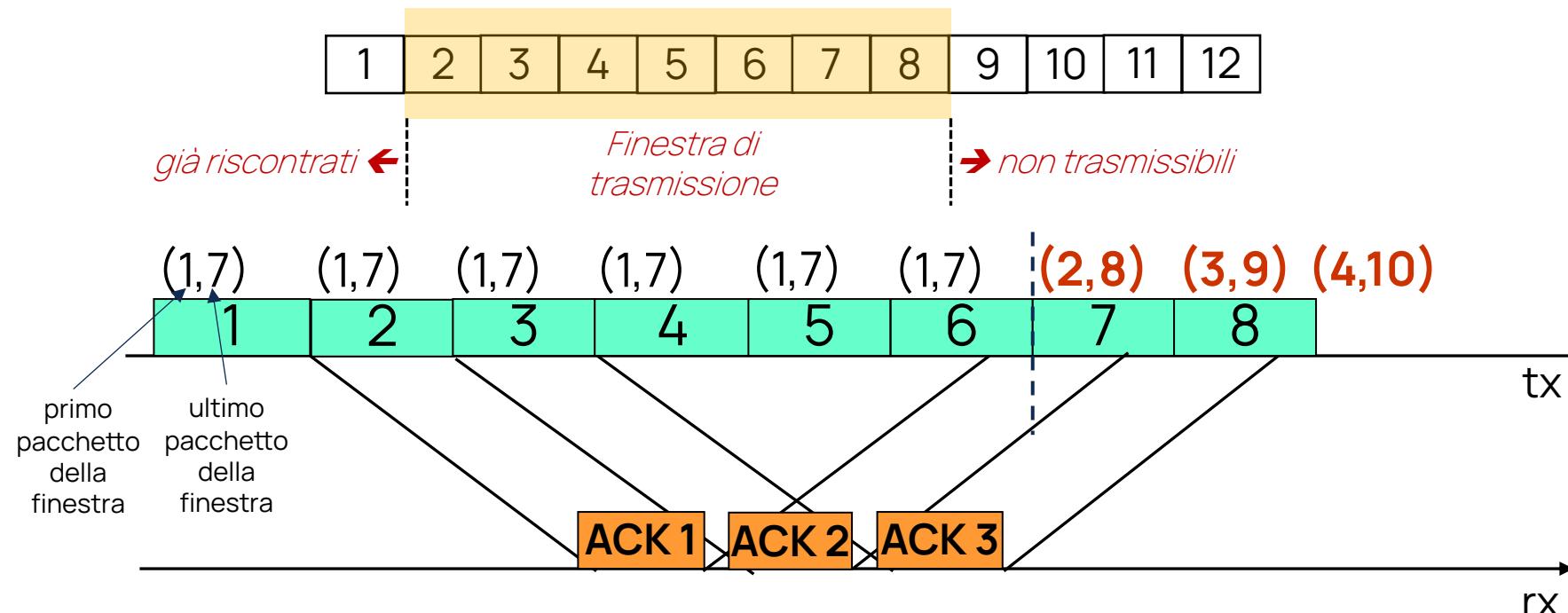
- Se il riscontro del primo pacchetto arriva prima della fine della finestra, la finestra viene fatta scorrere di una posizione (**sliding window**)



Se non ci sono errori la trasmissione non si ferma mai (efficienza 100%)

# Finestra Go-Back-N

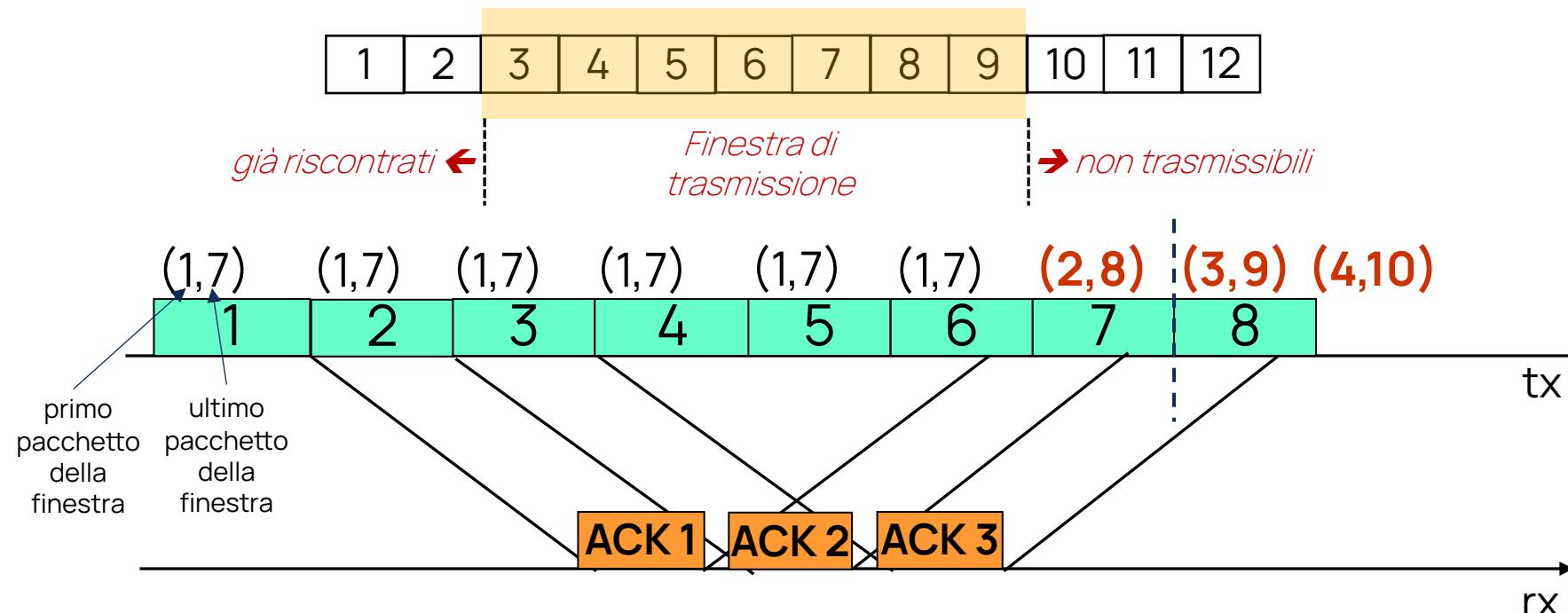
- Se il riscontro del primo pacchetto arriva prima della fine della finestra, la finestra viene fatta scorrere di una posizione (**sliding window**)



Se non ci sono errori la trasmissione non si ferma mai (efficienza 100%)

# Finestra Go-Back-N

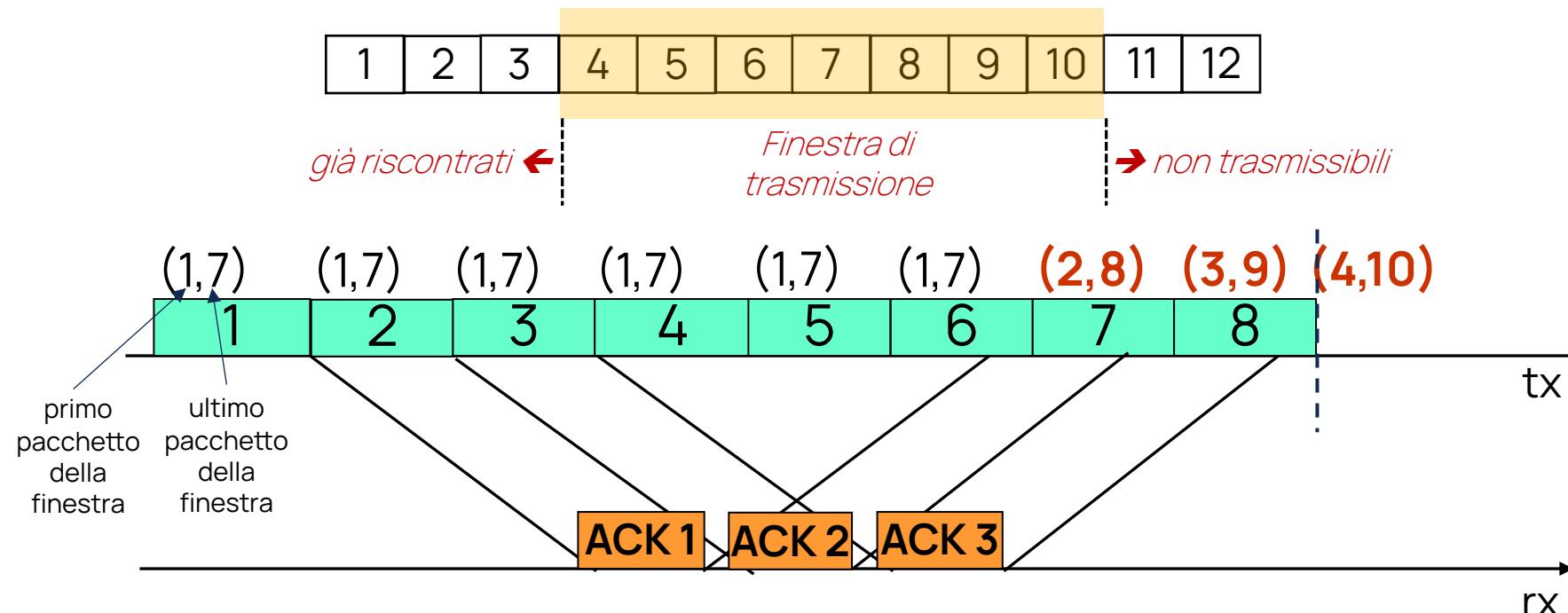
- Se il riscontro del primo pacchetto arriva prima della fine della finestra, la finestra viene fatta scorrere di una posizione (**sliding window**)



Se non ci sono errori la trasmissione non si ferma mai (efficienza 100%)

# Finestra Go-Back-N

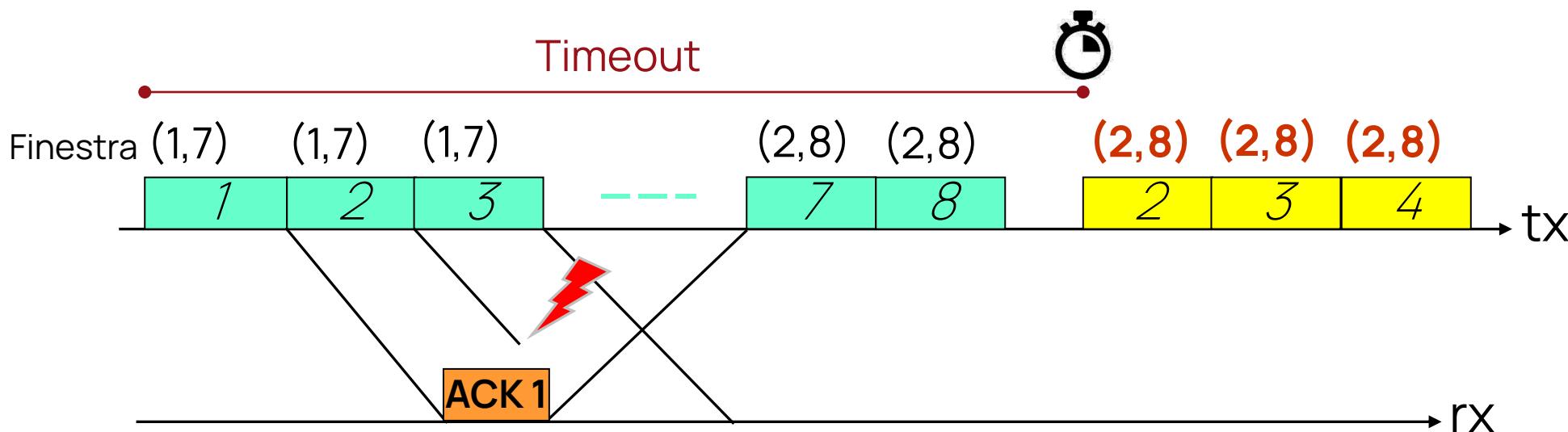
- Se il riscontro del primo pacchetto arriva prima della fine della finestra, la finestra viene fatta scorrere di una posizione (**sliding window**)



Se non ci sono errori la trasmissione non si ferma mai (efficienza 100%)

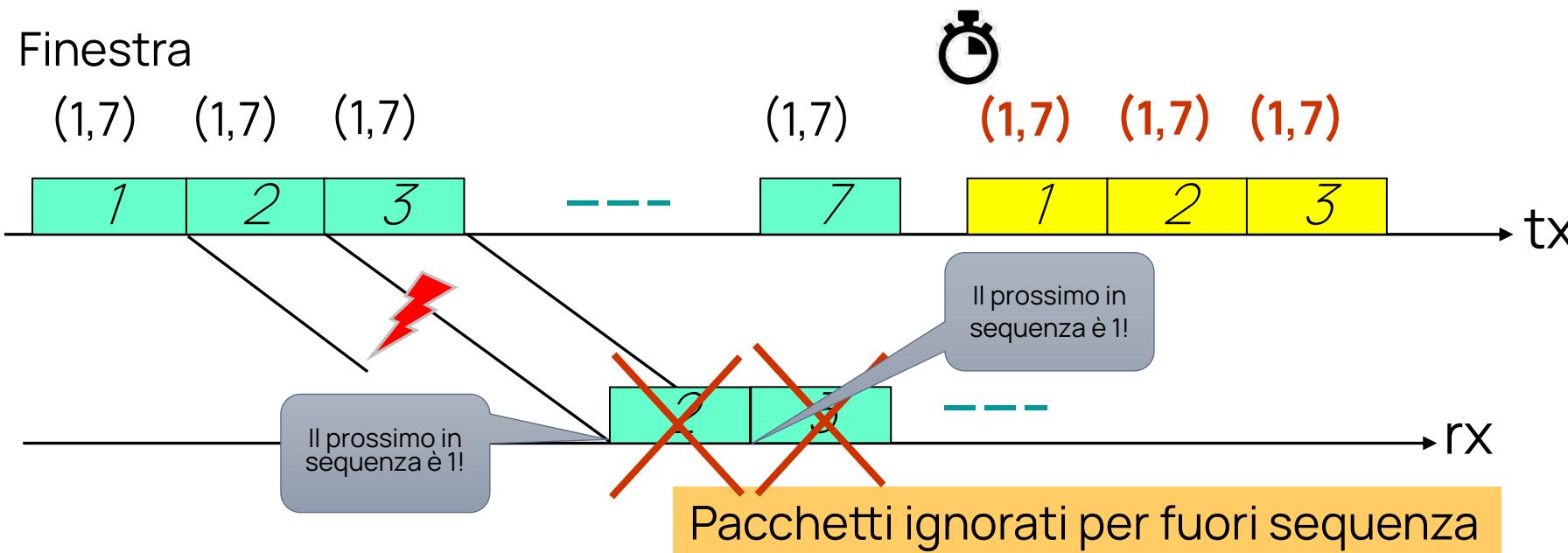
# Finestra Go-Back-N

- ❑ ... altrimenti -quando si verifica un errore- si ricomincia a trasmettere la finestra dal primo pacchetto non riscontrato
  - ❑ “Torna indietro di N pacchetti”
- ❑ il time out ha lo stesso significato dello Stop&Wait
  - ❑ Raggiunto l’ultimo pacchetto della finestra, la trasmissione si blocca in attesa di un nuovo ACK o della scadenza del timeout
  - ❑ La ritrasmissione del primo pacchetto non riscontrato inizia allo scadere del timeout



# Finestra Go-Back-N

- Ciò può causare la ritrasmissione di pacchetti corretti, ma semplifica il funzionamento perché
- .. permette al ricevitore di ignorare le ricezioni fuori sequenza (l'ordine è mantenuto automaticamente)



# Go-Back-N in action

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

sender

send pkt0  
send pkt1  
send pkt2  
send pkt3  
(wait)

receiver

receive pkt0, send ack0  
receive pkt1, send ack1

receive pkt3, discard,  
(re)send ack1

receive pkt4, discard,  
(re)send ack1  
receive pkt5, discard,  
(re)send ack1

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

rcv ack0, send pkt4  
rcv ack1, send pkt5



*pkt 2 timeout*

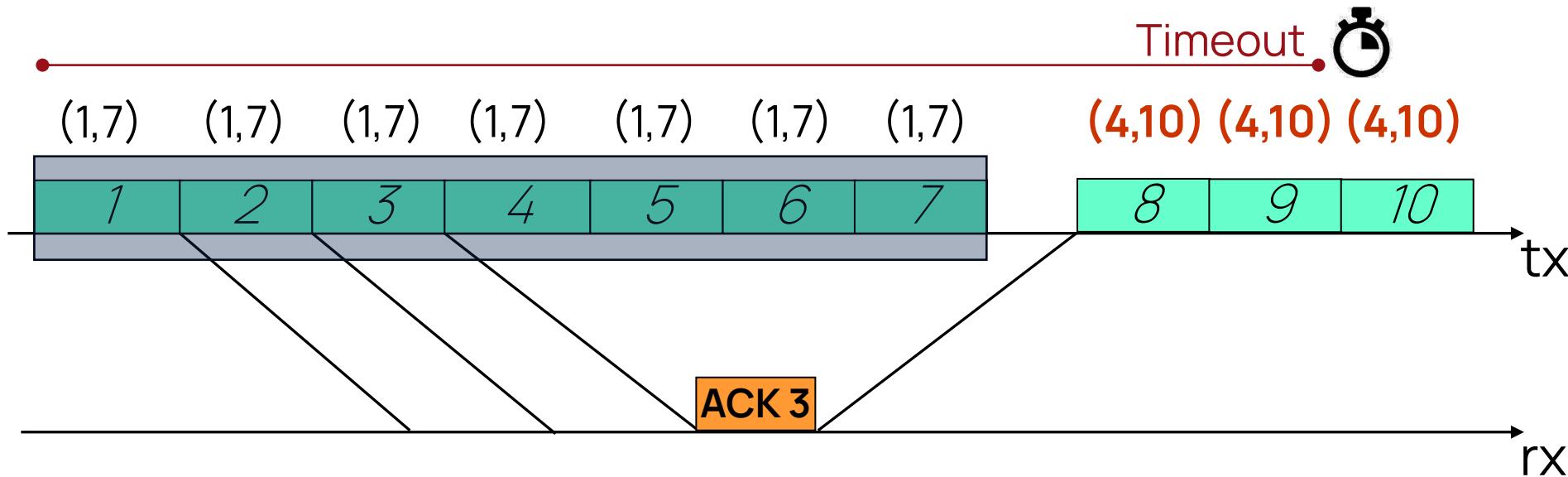
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

send pkt2  
send pkt3  
send pkt4  
send pkt5

rcv pkt2, deliver, send ack2  
rcv pkt3, deliver, send ack3  
rcv pkt4, deliver, send ack4  
rcv pkt5, deliver, send ack5

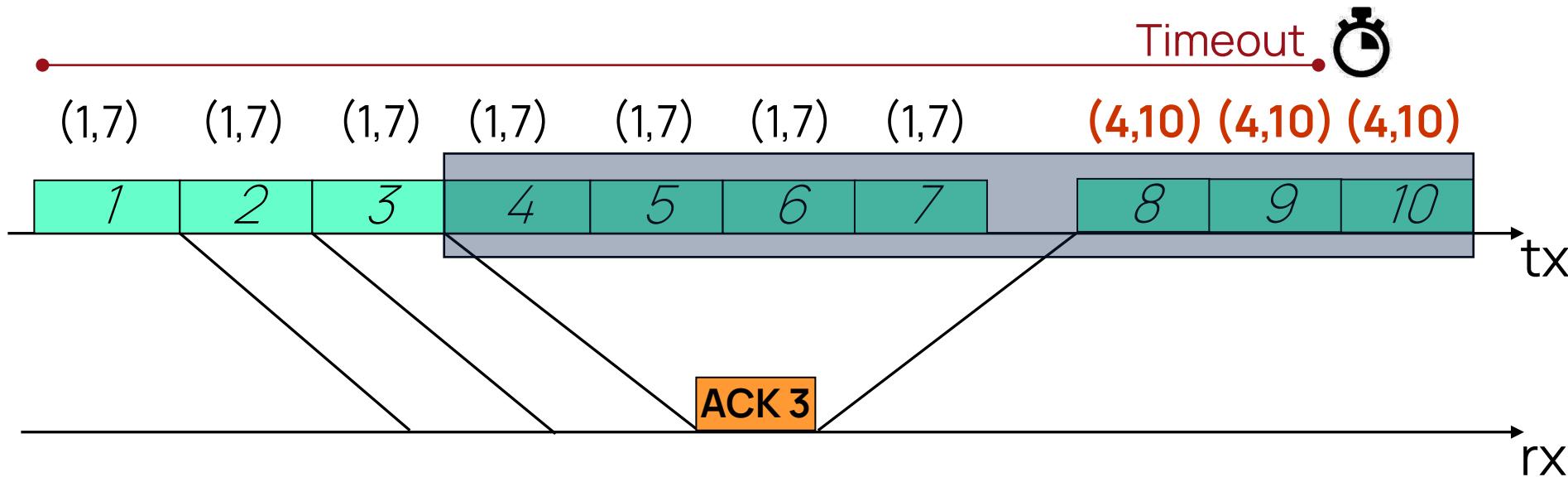
# Go-Back-N: ACK collettivo

- Escludendo il fuori sequenza, il riscontro (ACK) può essere collettivo (cumulative)
- Questo, entro certi limiti, rimedia alla perdita di ACK



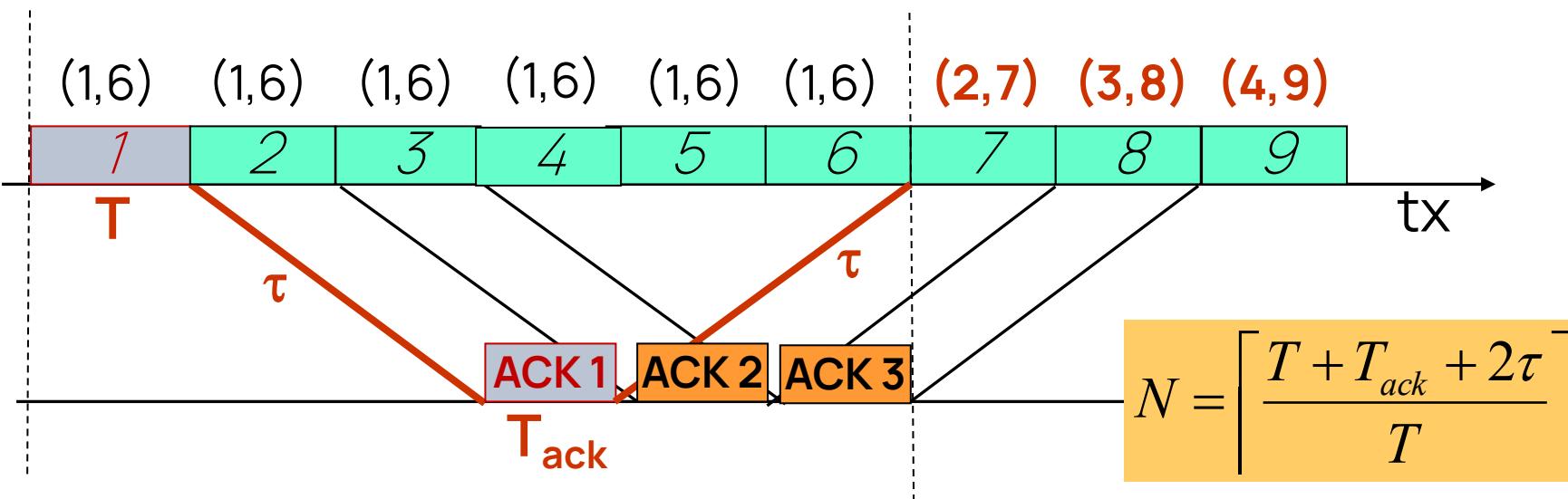
# Go-Back-N: ACK collettivo

- Escludendo il fuori sequenza, il riscontro (ACK) può essere collettivo (cumulative)
- Questo, entro certi limiti, rimedia alla perdita di ACK



# Dimensionamento della finestra

- La finestra ottima coincide con il **Round Trip Time**
  - Trasmissione pacchetto + Propagazione tx-rx + Trasmissione ACK + Propagazione rx-tx
  - La finestra aumenta di un pacchetto alla volta e la trasmissione non si interrompe mai



# Dimensionamento della finestra

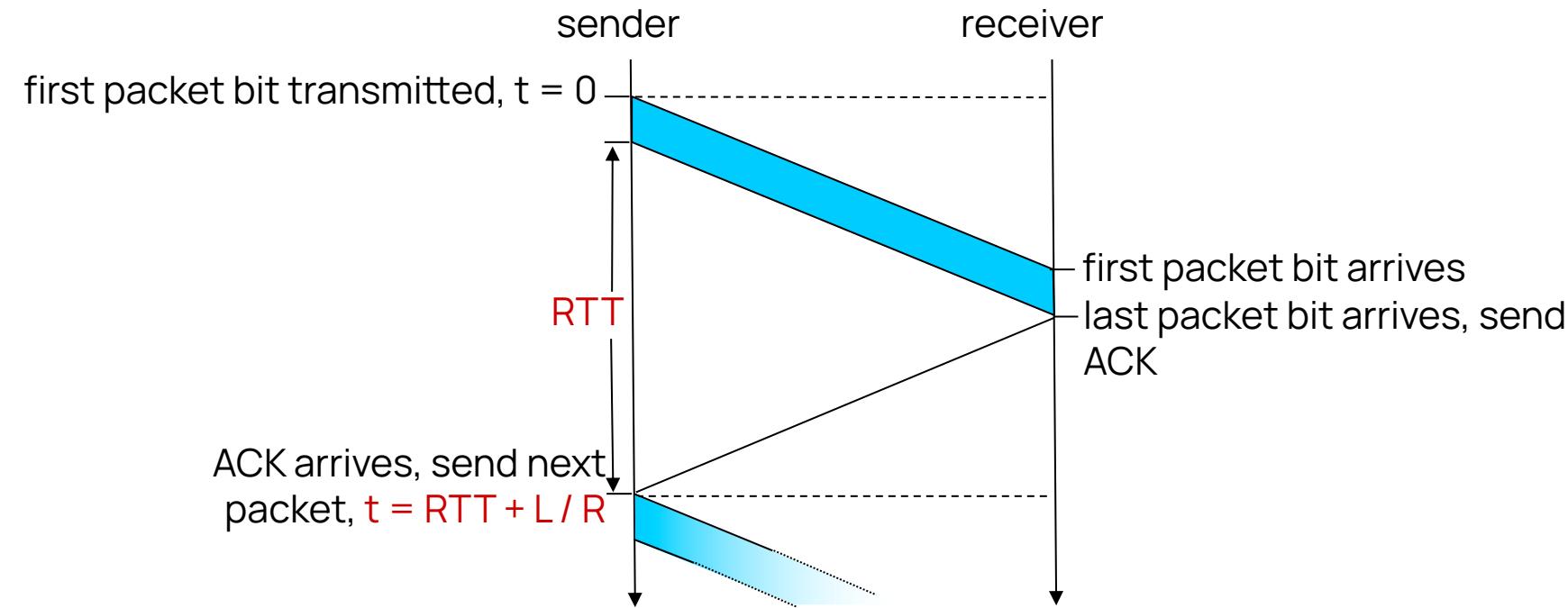
- La finestra può anche essere dimensionata in tempo, in byte, ...
- Il dimensionamento si complica se i tempi di propagazione non sono noti
  - es. con Go-back-N a livello di trasporto i tempi di attraversamento della rete ( $\tau$ ) possono variare col tempo..
  - e/o i pacchetti sono di lunghezza variabile
- Rimedi
  - Fare la **finestra grande**
    - Non pregiudica il funzionamento in assenza d'errore
    - Ma in caso d'errore, aumenta il ritardo con cui si scopre ed il numero di ritrasmissioni inutili
  - Uso del **NACK**
  - Stimare il tempo di **RTT** e adattare la finestra o il timeout

# Efficienza a confronto

$$\eta = \frac{T}{RTT + T}$$

$$= \frac{.008}{30.008}$$

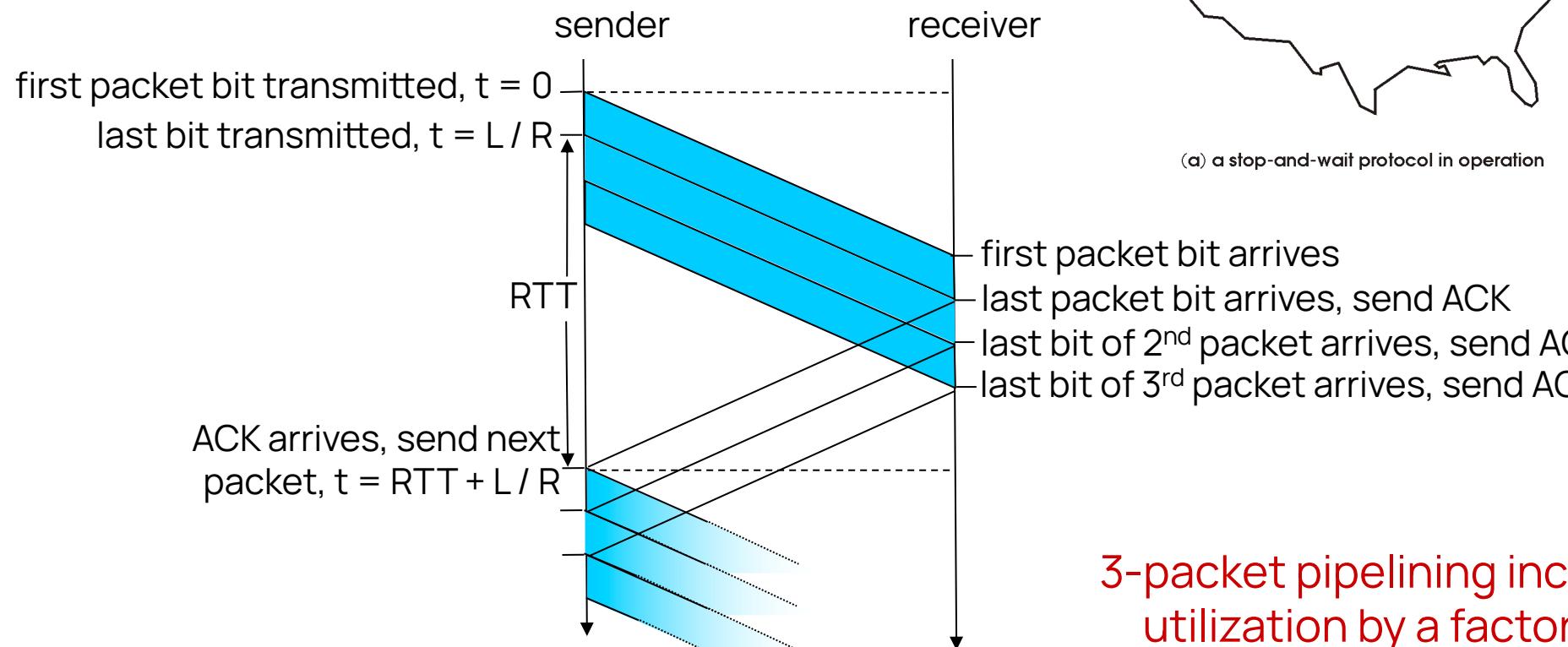
$$= 0.00027$$



- example: 1 Gbps link, 15 ms prop. delay, 8000 bit packet

- Time to transmit packet into channel:  $T = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$

# Efficienza a confronto



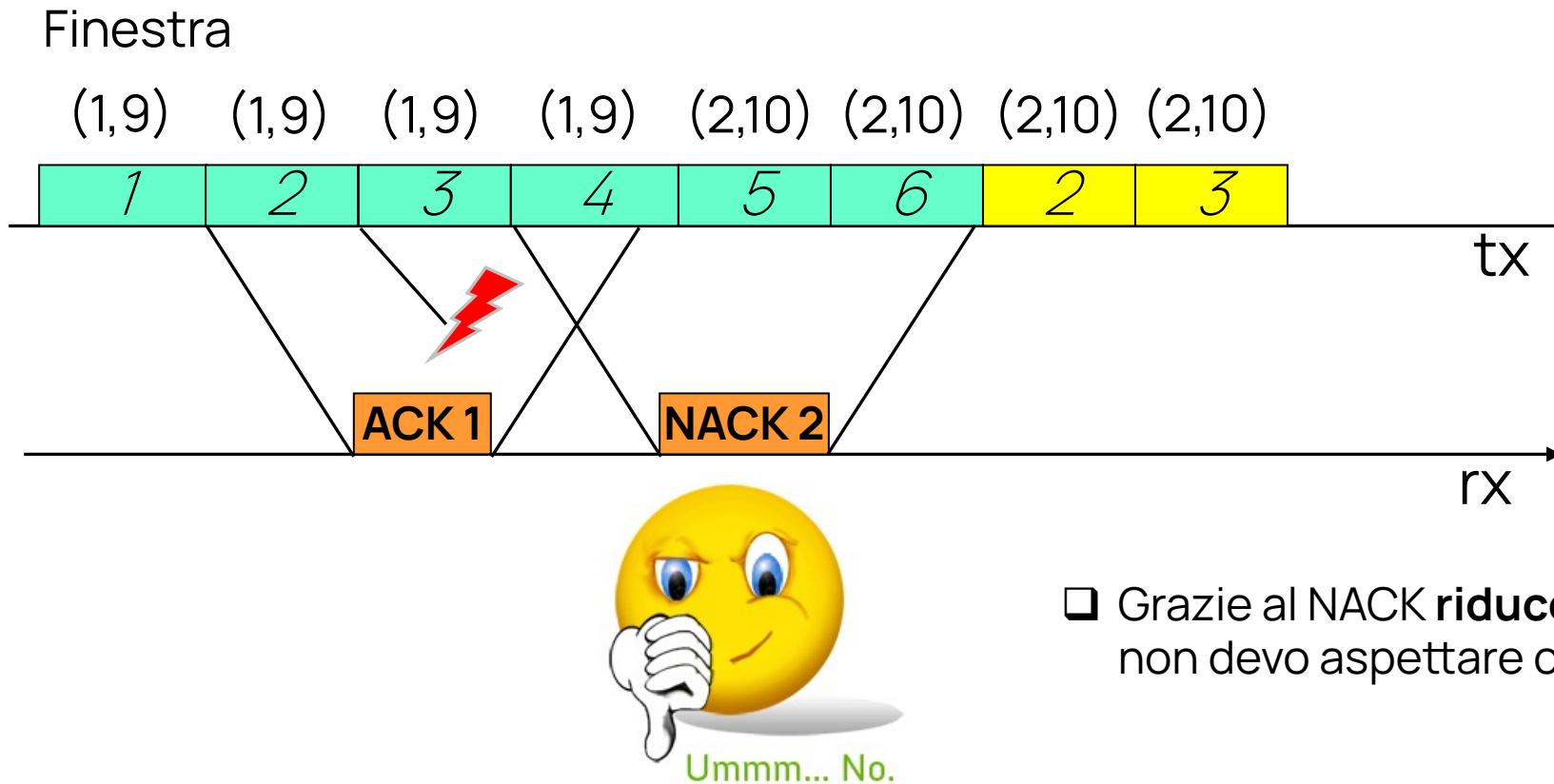
3-packet pipelining increases utilization by a factor of 3!

$$\eta = \frac{3L/R}{RTT + L/R} = \frac{.0024}{30.008} = 0.00081$$

\*tratto dal libro di testo di Kurose & Ross



# Uso del NACK

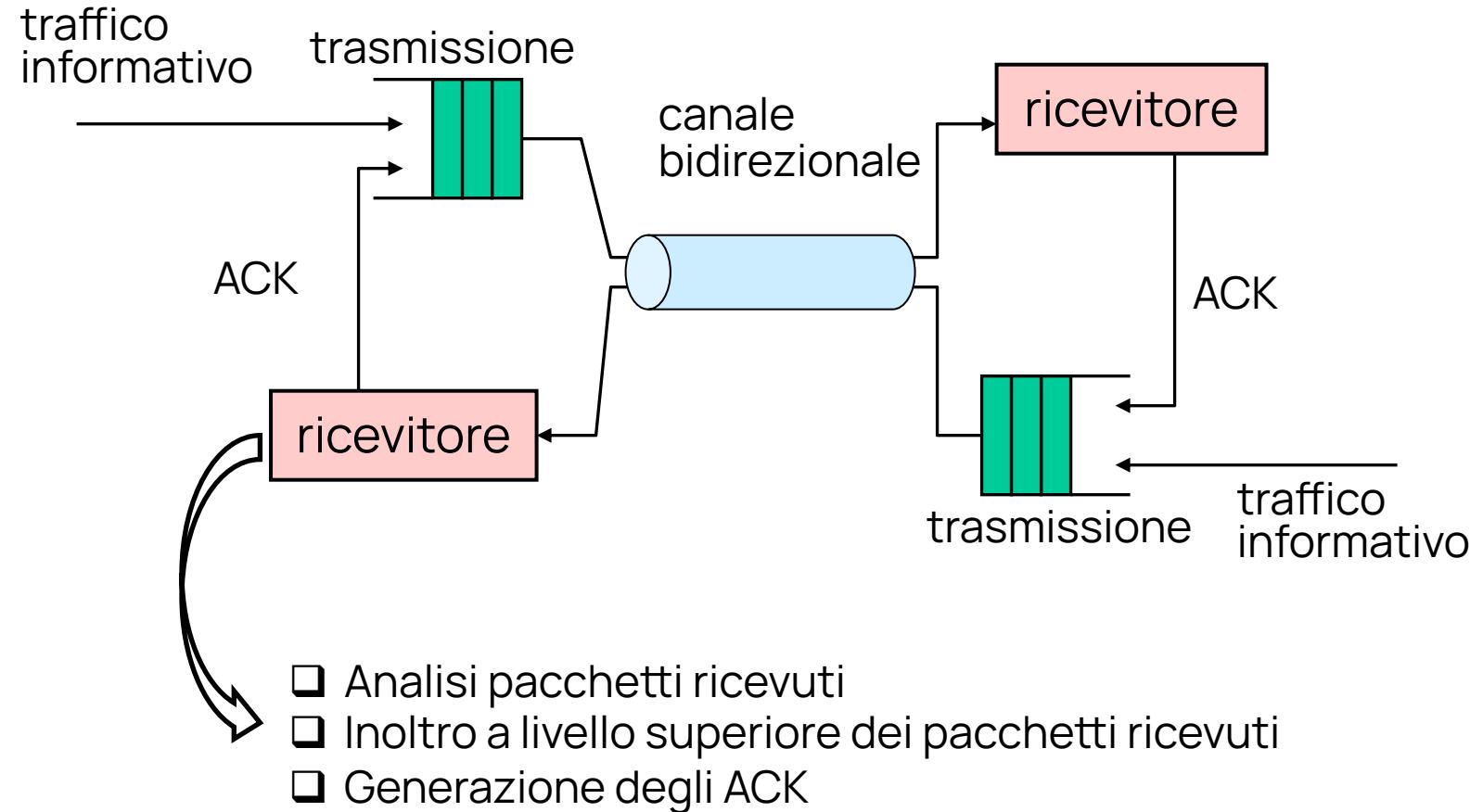


- Grazie al NACK **riduco i tempi di attesa**, non devo aspettare che la finestra termini

# Uso del NACK

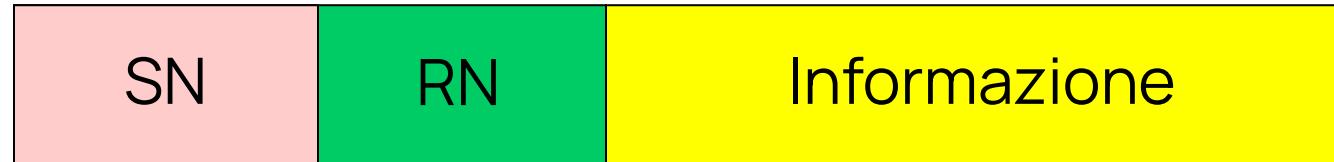
- L'uso del NACK può abbreviare i tempi di ritrasmissione in caso d'errore, evitando di aspettare la fine della finestra
- Non è possibile inviare immediatamente il NACK perché il ricevitore dovrebbe conoscere il SN del pacchetto errato, ma se è errata ...
- Se arriva un pacchetto fuori sequenza, posso ipotizzare che sia andata perso il pacchetto precedente
- Non è possibile applicarlo nei meccanismi in cui non è garantita la consegna in ordine (livelli superiori al livello di link)

# Go-Back-N full-duplex



# Go-Back-N e piggy-backing

- Gli ACK possono anche essere inseriti negli *header* dei pacchetti dati che viaggiano in direzione opposta (*Piggy-backing*, i.e., *portare qualcosa sopra qualcos'altro*)



- **SN**: numero di sequenza del pacchetto trasmesso (canale diretto)
- **RN**: numero di sequenza del pacchetto atteso in direzione opposta (numero ACK), vale come riscontro cumulativo dei pacchetti fino a **RN-1**

# Regole Go-Back-N

- Trasmettitore:

- $N$ : dimensione finestra
- $N_{last}$ : ultimo riscontro ricevuto
- $N_C$ : numero pacchetto in trasmissione
- Intervallo  $N_{last}$  e  $(N_{last} + N - 1)$ : finestra di trasmissione

- **Regole:**

- Ogni nuovo pacchetto viene accettato per la trasmissione solo se  $N_C < N_{last} + N$ , altrimenti viene messo in attesa;
- Se  $N_C < N_{last} + N$  il pacchetto viene trasmesso con SN pari a  $N_C$ ;
- Viene inizializzato il timer di timeout e il valore di  $N_C$  viene incrementato di uno ( $N_C := N_C + 1$ );
- Ad ogni riscontro RN ricevuto, si pone  $N_{last} = RN$ ;
- In caso di scadenza di timeout la ritrasmissione deve ripartire dal pacchetto  $N_{last}$

# Regole Go-Back-N

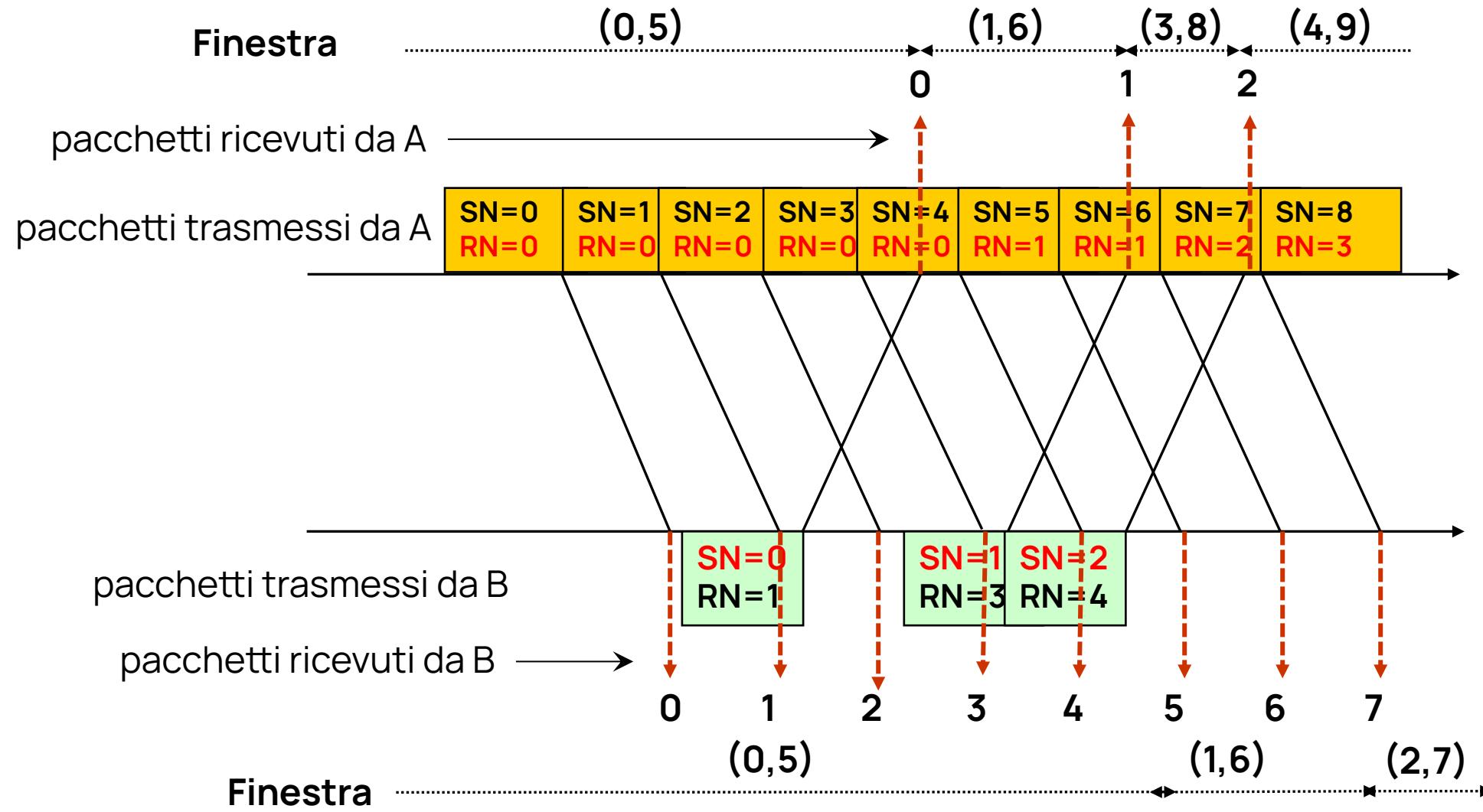
- ❑ Ricevitore:

- ❑ RN: stato dei riscontri corrente (SN che il ricevitore si aspetta di ricevere)

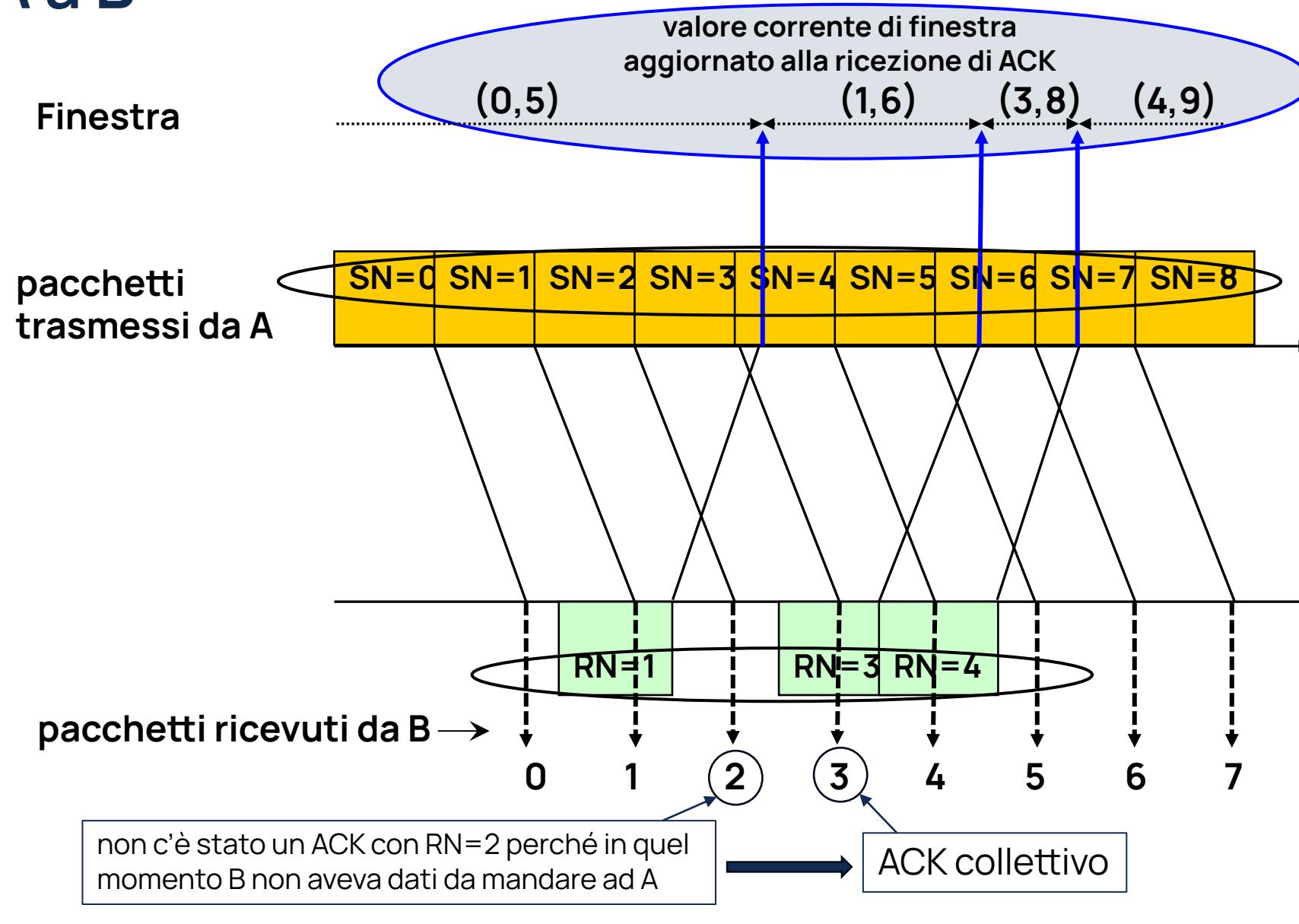
- ❑ **Regole:**

- ❑ Se viene ricevuto correttamente un pacchetto con  $SN = RN$ , questo viene inoltrato ai livelli superiori e si pone  $RN := RN + 1$ ;
  - ❑ Ad istanti arbitrari ma con ritardo finito, RN viene trasmesso al mittente utilizzando i pacchetti in direzione opposta.

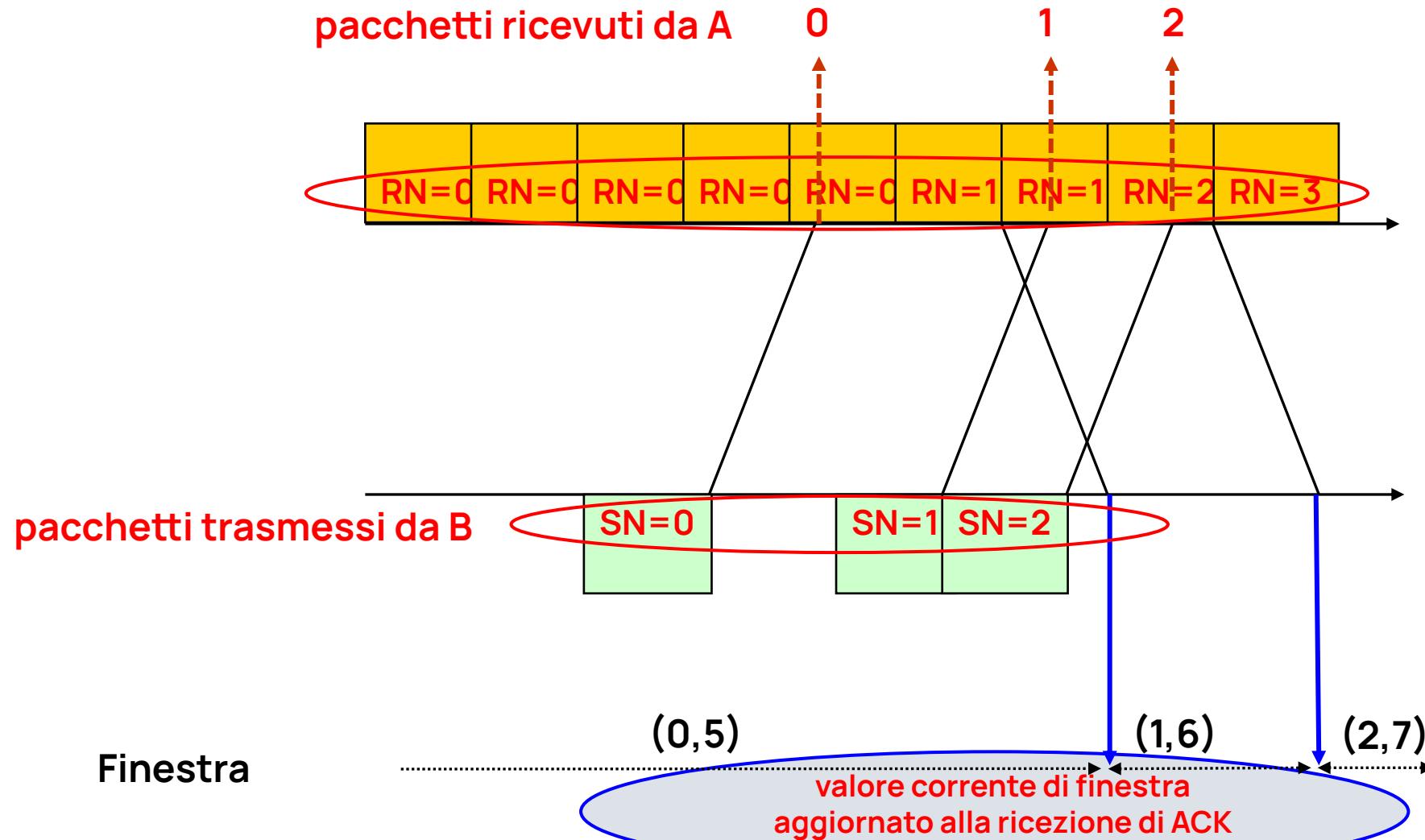
# Go-back-N full-duplex senza errori



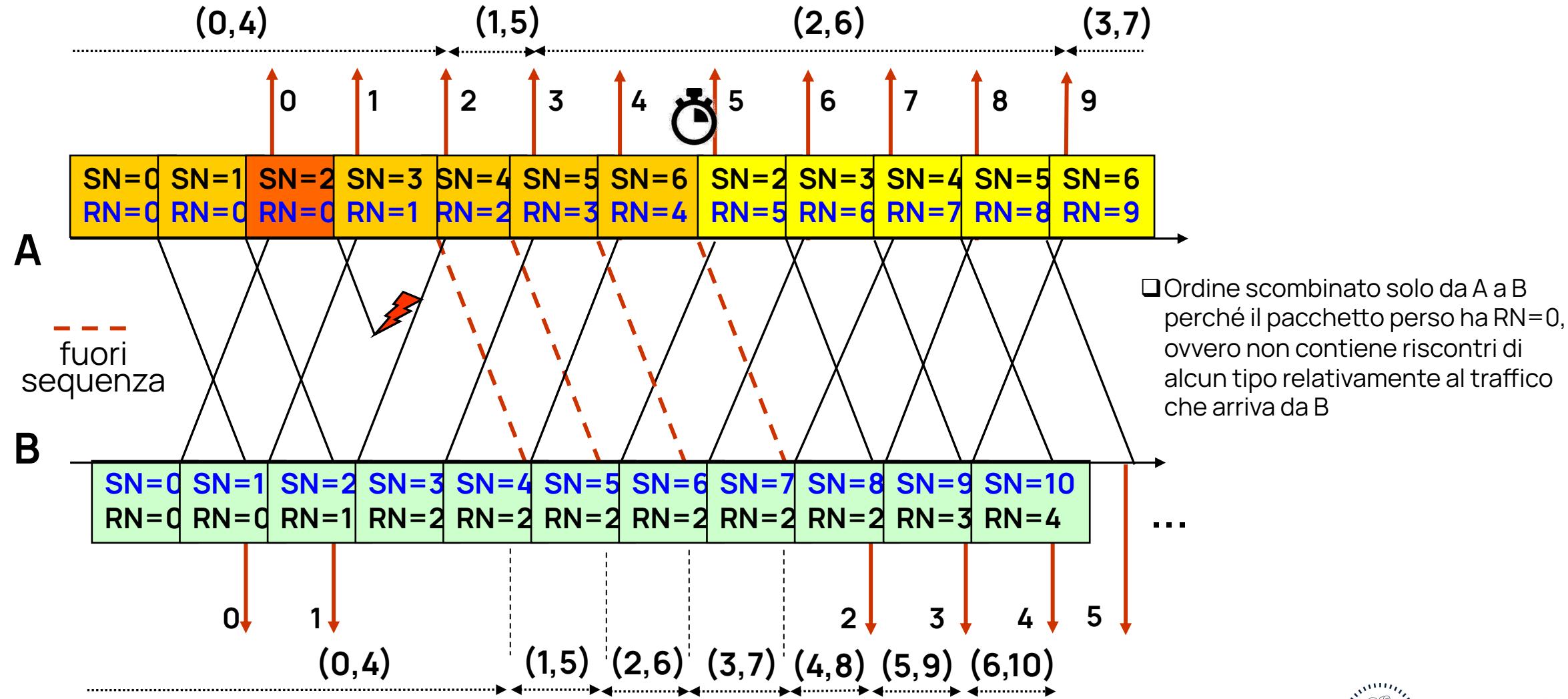
# Da A a B



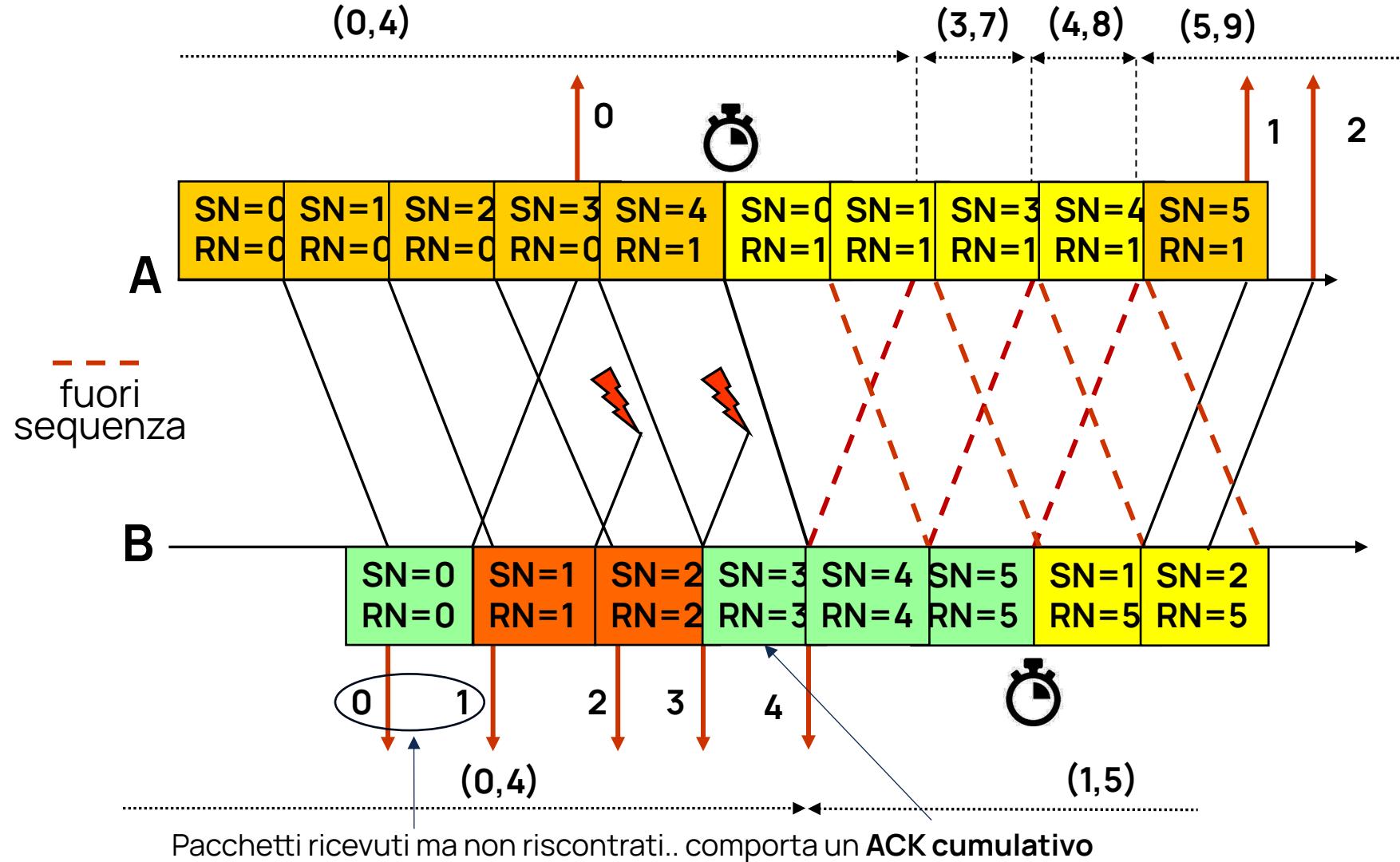
# Da B ad A



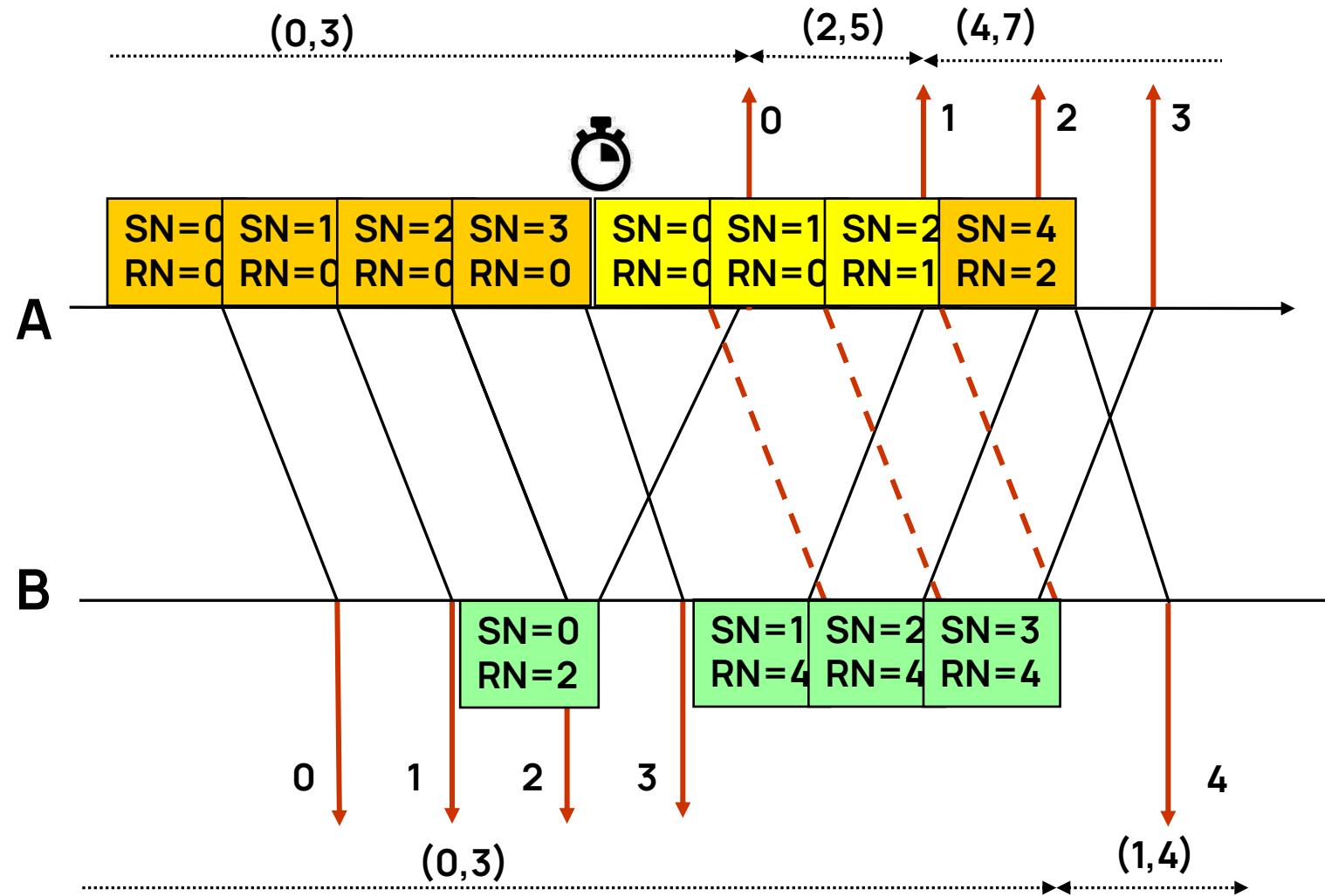
# Go-Back-N full-duplex con un errore in una direzione



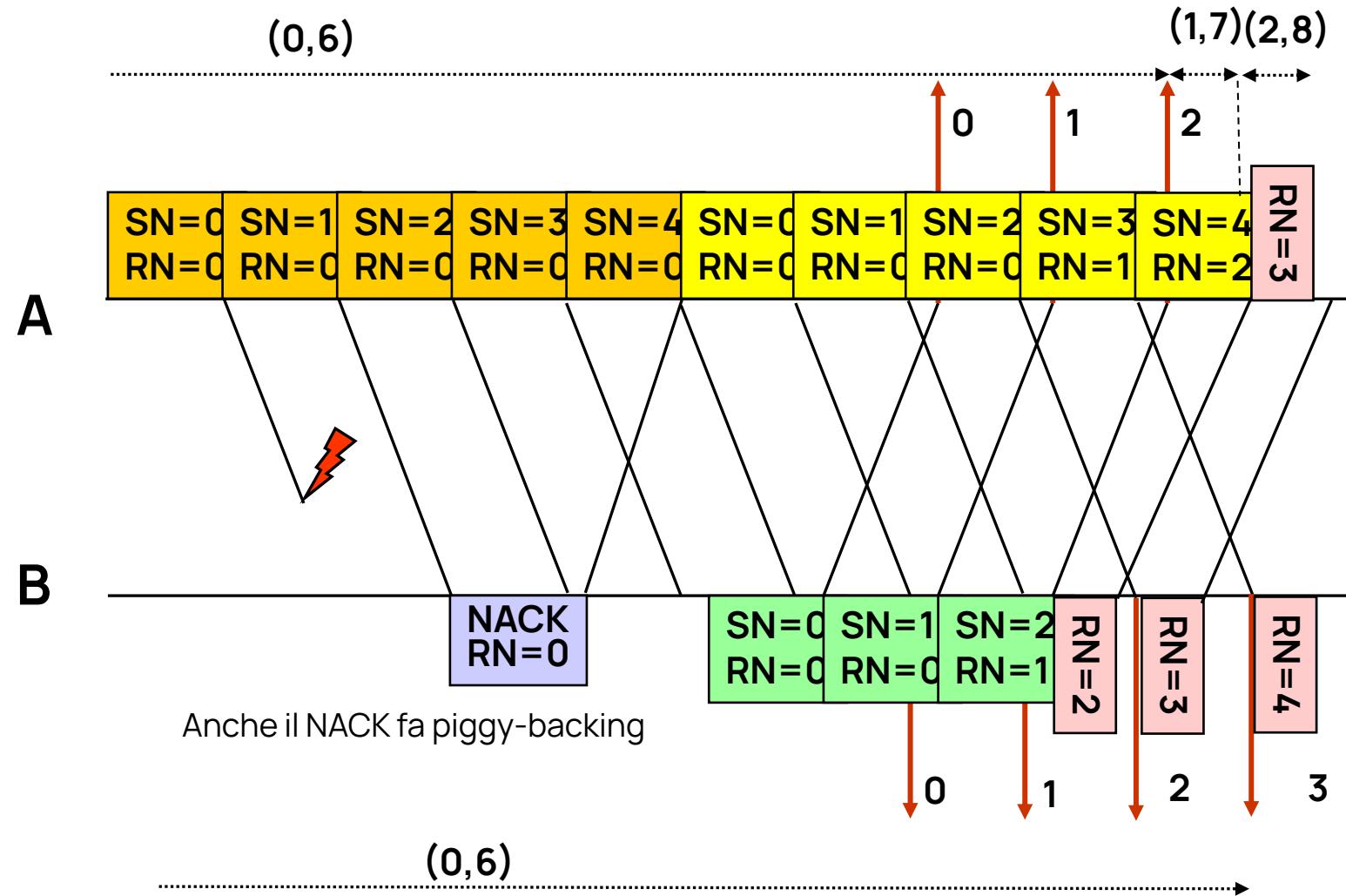
# Go-Back-N full-duplex con errori in due direzioni



# Go-Back-N full-duplex con ACK ritardati



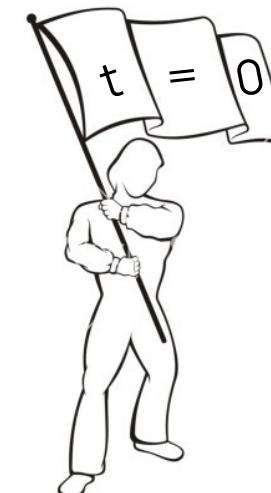
# Go-Back-N full-duplex con NACK



# Osservazioni sul controllo d'errore

- Necessità di **inizializzare** il protocollo → *sincronizzazione* dei timers
  - I numeri SN e RN devono essere inizializzati
  - Deve esistere un momento di inizio non equivocabile in cui scambiare l'informazione per l'inizializzazione
  - Questo è il motivo per cui la fase di inizializzazione della connessione TCP è così importante

Occorre un meccanismo a connessione che stabilisca l'istante  $t=0$

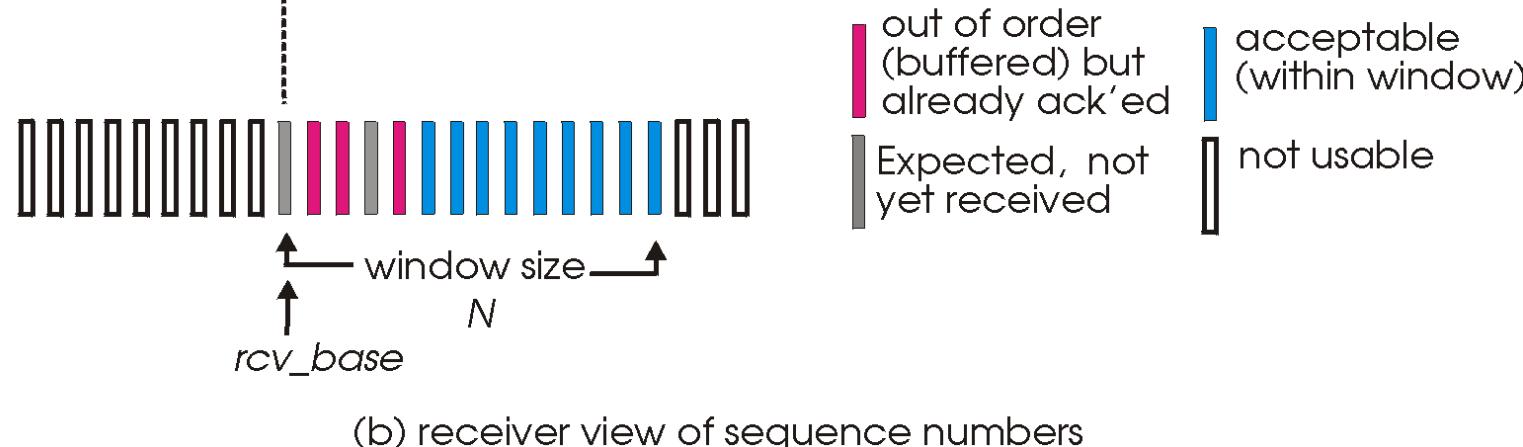
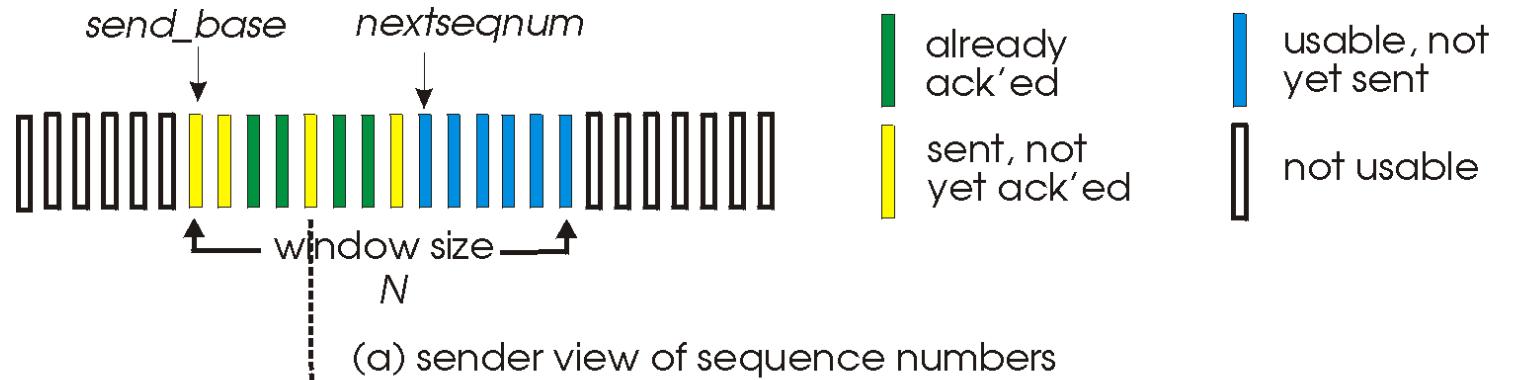


# SELECTIVE REPEAT

02

# Selective Repeat

- Il ricevitore fa un riscontro (ACK) di tutti i pacchetti ricevuti, inclusi quelli che non sono stati ricevuti in ordine, e li mette in un **buffer**



# Selective Repeat

sender window ( $N=4$ )

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

sender

send pkt0  
send pkt1  
send pkt2  
send pkt3  
(wait)

rcv ack0, send pkt4  
rcv ack1, send pkt5

record ack3 arrived  
 **pkt 2 timeout**  
send pkt2  
(but not 3,4,5)

receiver

receive pkt0, send ack0  
receive pkt1, send ack1

receive pkt3, buffer,  
send ack3

receive pkt4, buffer,  
send ack4

receive pkt5, buffer,  
send ack5

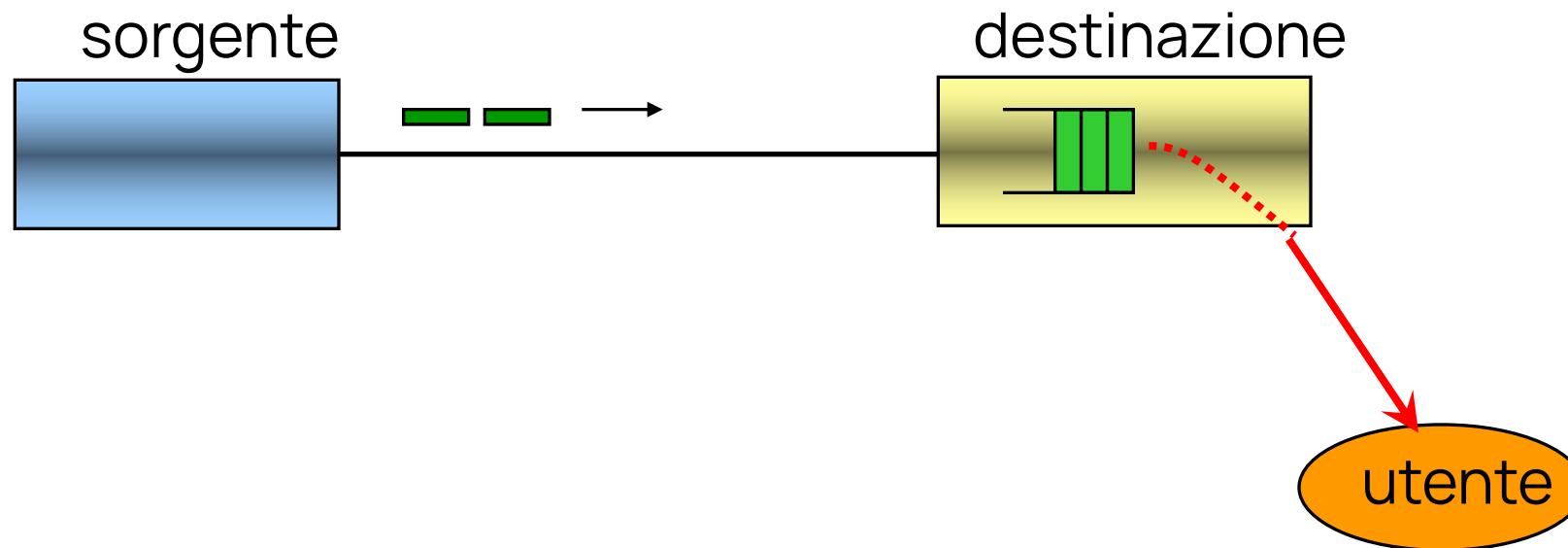
rcv pkt2; deliver pkt2,  
pkt3, pkt4, pkt5; send ack2

# CONTROLLO DI FLUSSO A FINESTRA MOBILE

02

# Controllo di flusso

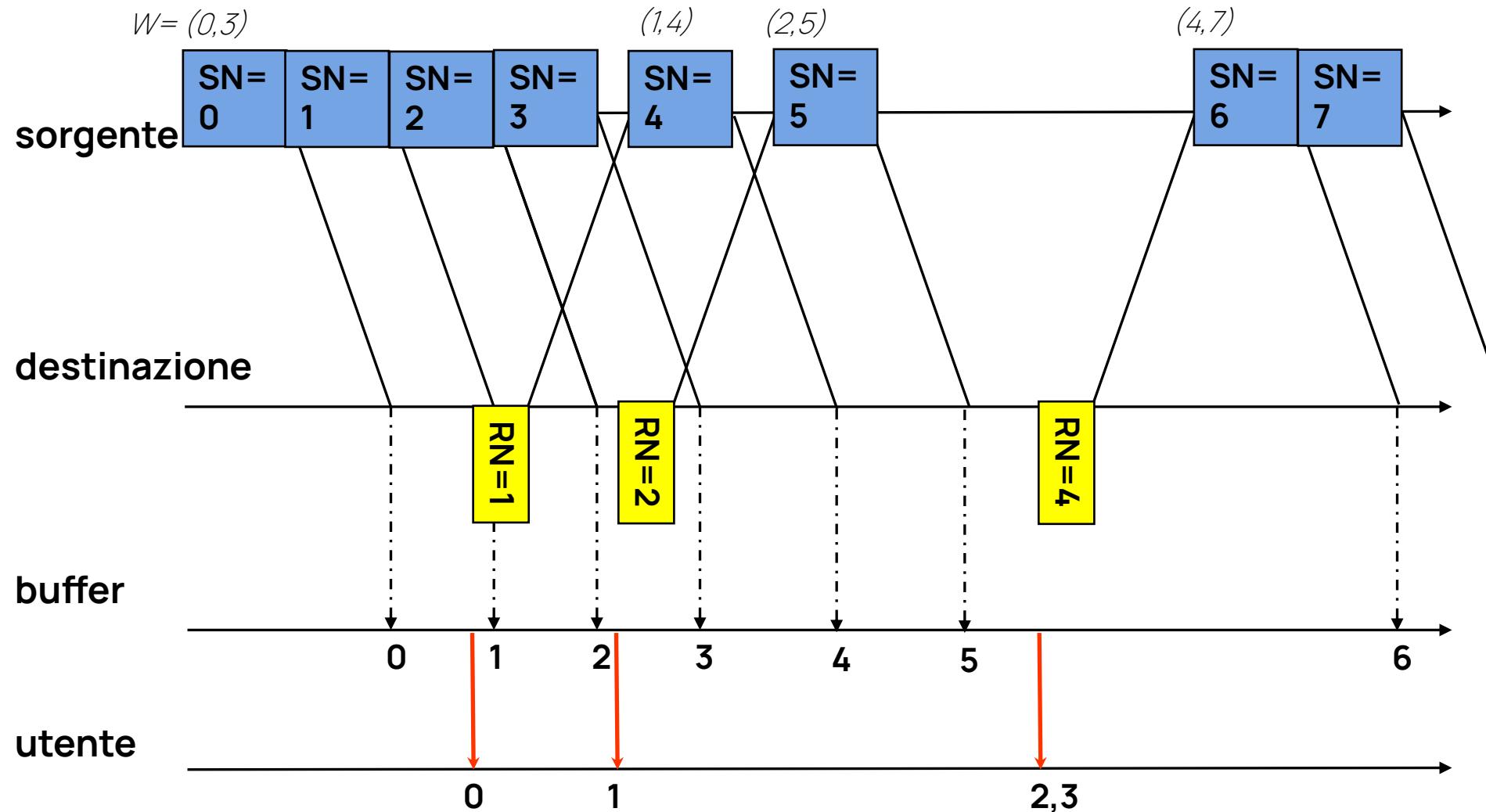
- Buffer di ricezione limitato a **W** posizioni
- Ritmo di assorbimento dell'utente
- **Obiettivo:** regolando il ritmo di invio, vogliamo evitare che pacchetti vadano persi perché all'arrivo trovano il buffer pieno



# Sliding window flow control

- Controllo di flusso a finestra mobile
- E' possibile usare un meccanismo come quello del Go-back-N
- La sorgente non può inviare più di **W** trame (stessa funzione del parametro N)  
senza aver ricevuto il riscontro
- I riscontri vengono inviati dal ricevitore solo quando i pacchetti vengono letti  
(tolti dal buffer) dal livello superiore

# Sliding window flow control



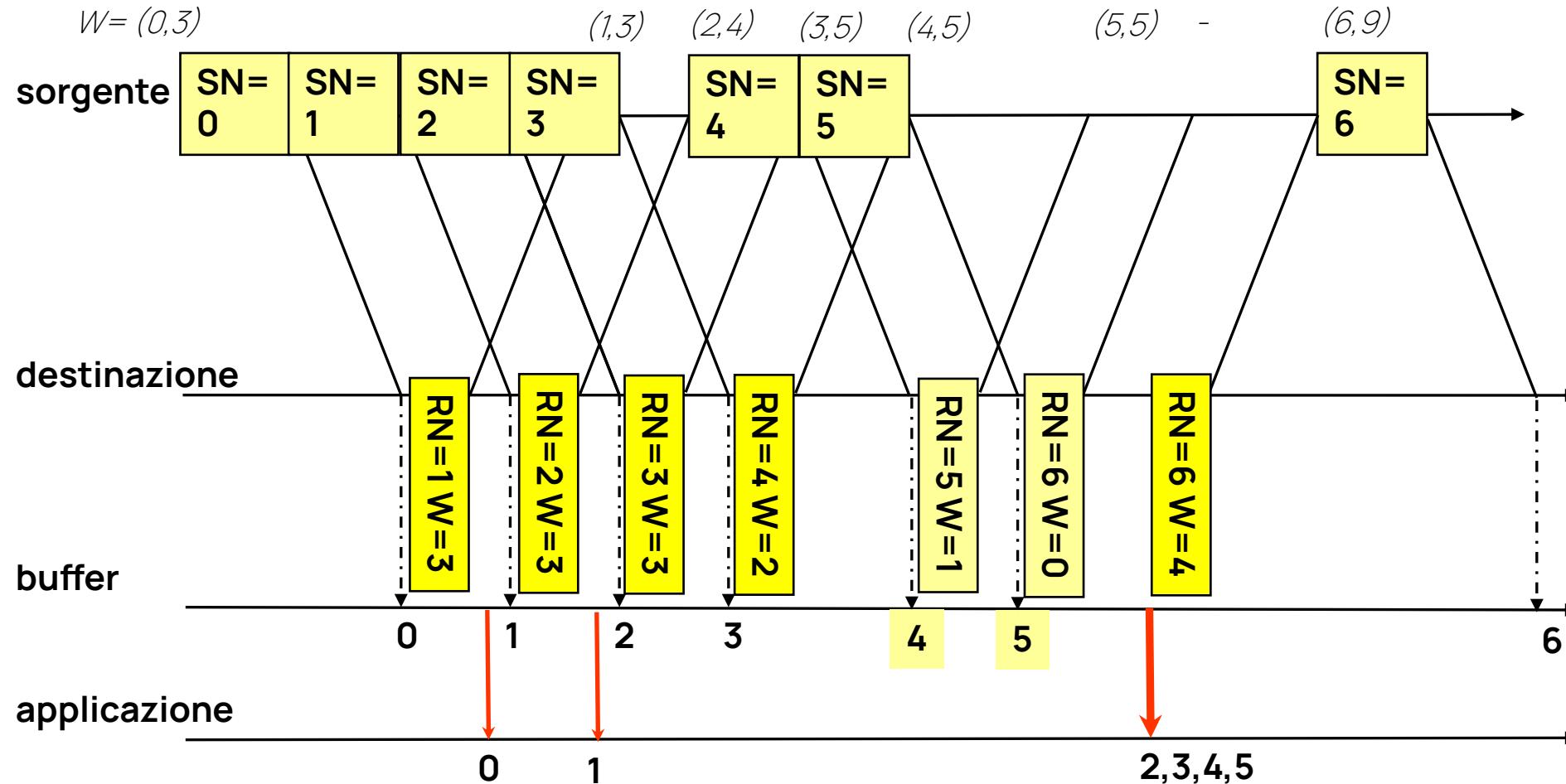
# Problema delle ritrasmissioni

- ❑ Il meccanismo di controllo di flusso descritto nelle slide precedenti è strettamente legato al meccanismo di controllo d'errore e questo può essere fonte di problemi
- ❑ Se il ricevitore ritarda molto l'invio dei riscontri a causa del livello superiore lento, il trasmettitore inizia la ritrasmissione perché scade il time-out
- ❑ Aumentare troppo il time-out non è ovviamente una soluzione in quanto l'aumento del time-out aumenta i ritardi in caso di errore

# Uso del campo W

- ❑ Il problema può essere risolto in modo radicale separando i meccanismi di controllo d'errore e di controllo di flusso a finestra
- ❑ Si inserisce nei riscontri (o nell'header delle trame in direzione opposta) un campo finestra W (insieme a quella del GBN)
  - ❑ Il ricevitore invia i riscontri sulla base dell'arrivo dei pacchetti
  - ❑ E usa il campo W per indicare lo spazio rimanente nel buffer

# Uso del campo W



# Uso del campo W

- Gestione della finestra del controllo di flusso
  - Non è necessario che il ricevitore dica la “verità” sullo spazio restante R
  - Può tenersi un margine di sicurezza ( $W=R-m$ )
  - Può aspettare che il buffer si sia svuotato per una frazione (ad es.  $W=0$  se  $R < K/2$  ed  $W=R$  altrimenti)
  - Può usare dei meccanismi adattativi

# Fondamenti di TELECOMUNICAZIONI

Prof. Marco Mezzavilla  
[marco.mezzavilla@polimi.it](mailto:marco.mezzavilla@polimi.it)