



Università
degli Studi
della Campania
Luigi Vanvitelli

Reti di Calcolatori e Cybersecurity

Principi di Trasmissione Affidabile

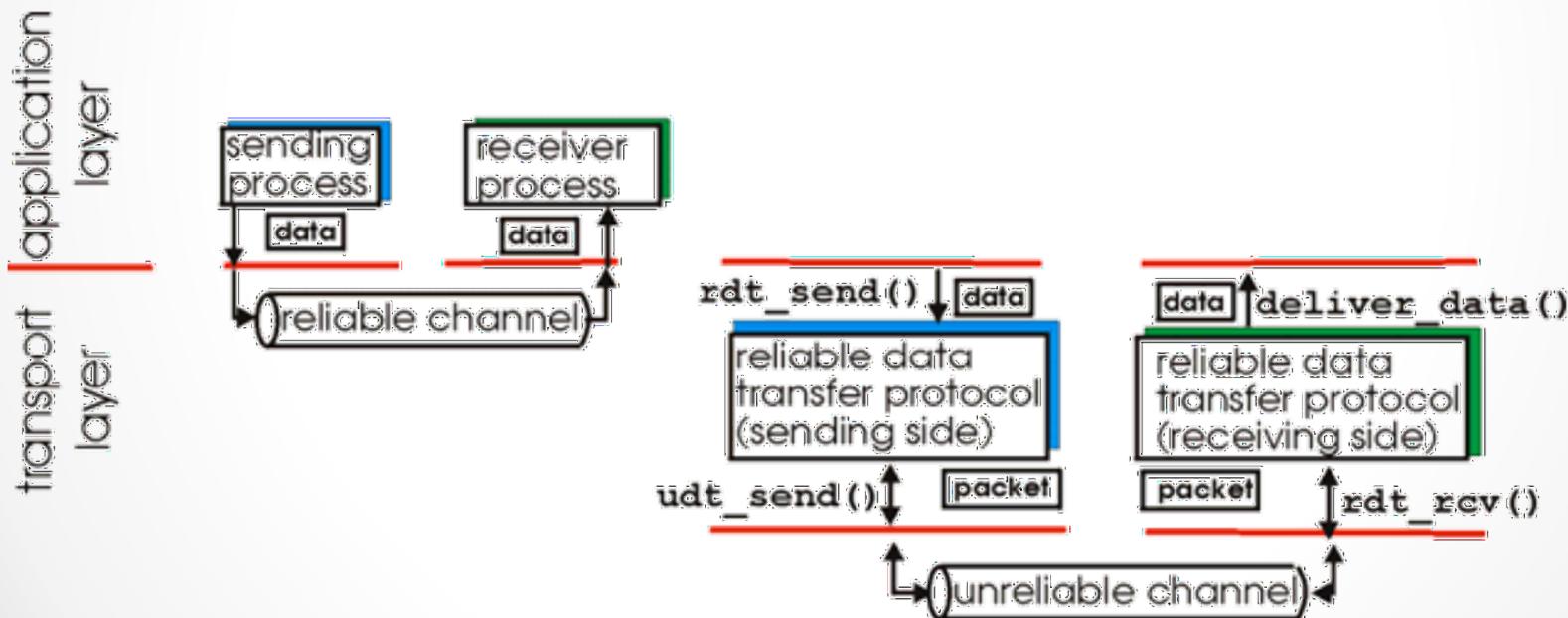
Ing. Vincenzo Abate

Trasmissione Affidabile

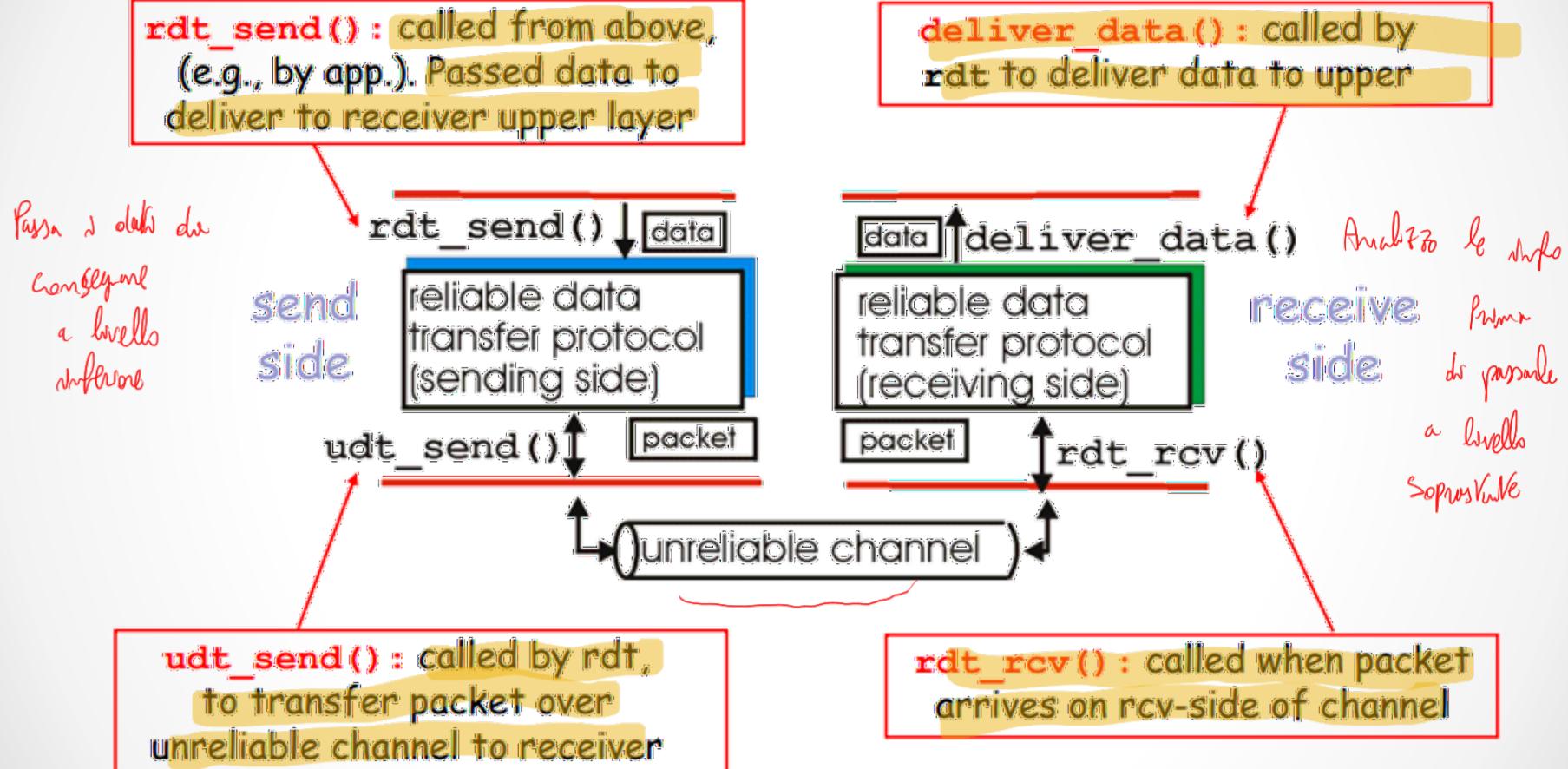
Importante in app., livelli trasporto e collegamento

Uno dei più importanti aspetti delle reti

Caratteristico di un canale inaffidabile è la complessità di un protocollo di trasferimento dati affidabile



Trasmissione Affidabile



Livello trasporto

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 a livello trasporto possono essere realizzati dei meccanismi che tengano in considerazione:

- Che i buffer del computer ricevente sono di capacità limitata:
 - Controllo di flusso
- Che i buffer dei router sono di capacità limitata
 - Controllo di congestione

Tenendo in considerazione il limite fisico nonché sulla rete

Canale con Errori

Il canale sottostante può invertire i bit nel pacchetto

- checksum per rilevare errori di bit

la domanda: come recuperare dagli errori?

Correzione o Notifica al mittente e ritrasmissione

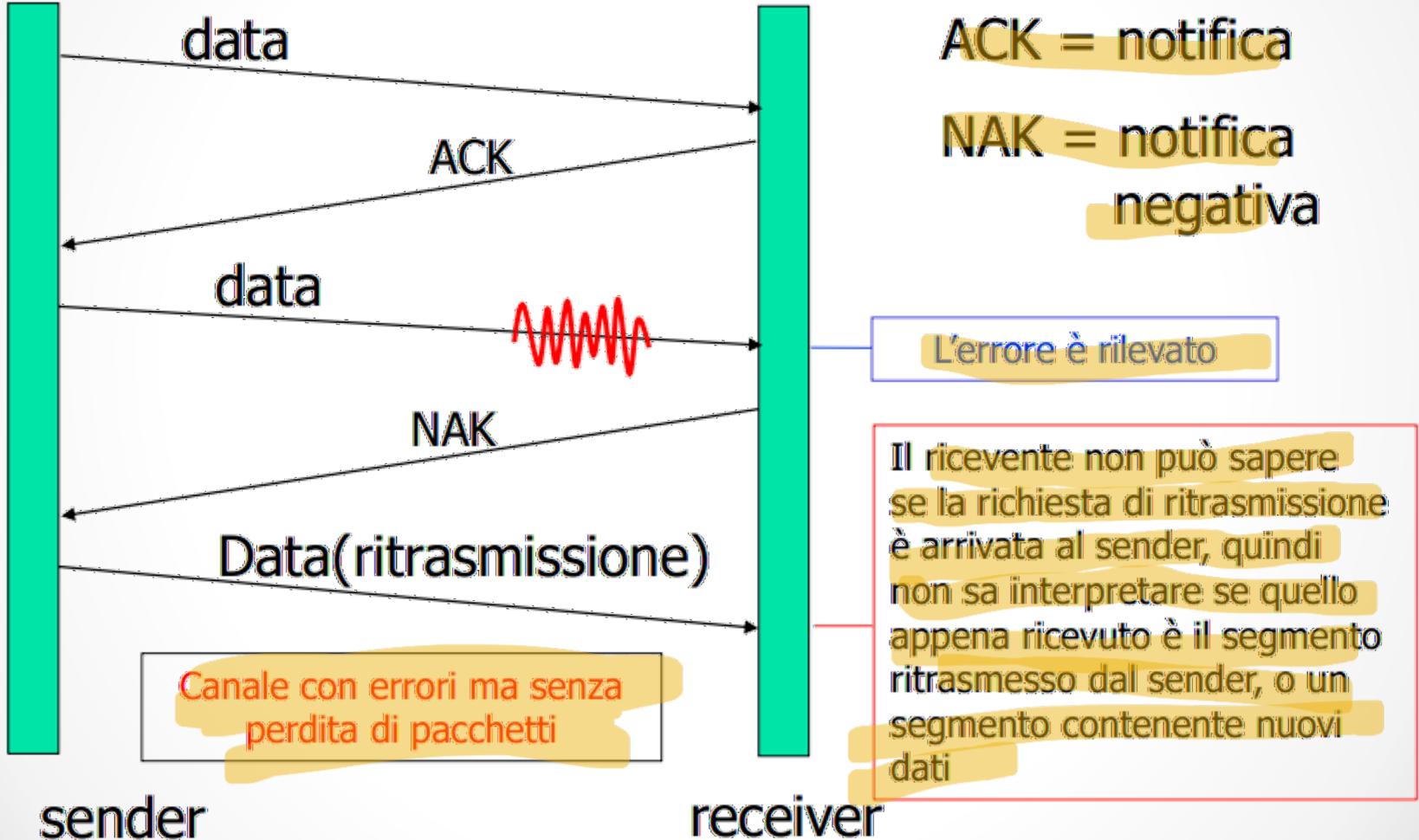
La prima soluzione introduce complicazioni, la seconda introduce possibili duplicazioni sulla rete che il ricevente non è in grado di interpretare

Riconoscimenti (**ACK**): il destinatario dice esplicitamente al mittente che pkt ricevuto OK

Riconoscimenti negativi (**NAK**): il destinatario dice esplicitamente al mittente quel pkt aveva errori e il mittente ritrasmette il pacchetto alla ricezione di NAK



Canale con Errori



Canale con Errori

Cosa succede se ACK/NAK corrotti?

il mittente non lo sa cosa è successo a ricevitore! non può semplicemente ritrasmettere: possibile duplicato

Gestione dei duplicati:

il mittente aggiunge nell'header il numero di sequenza per ogni pkt

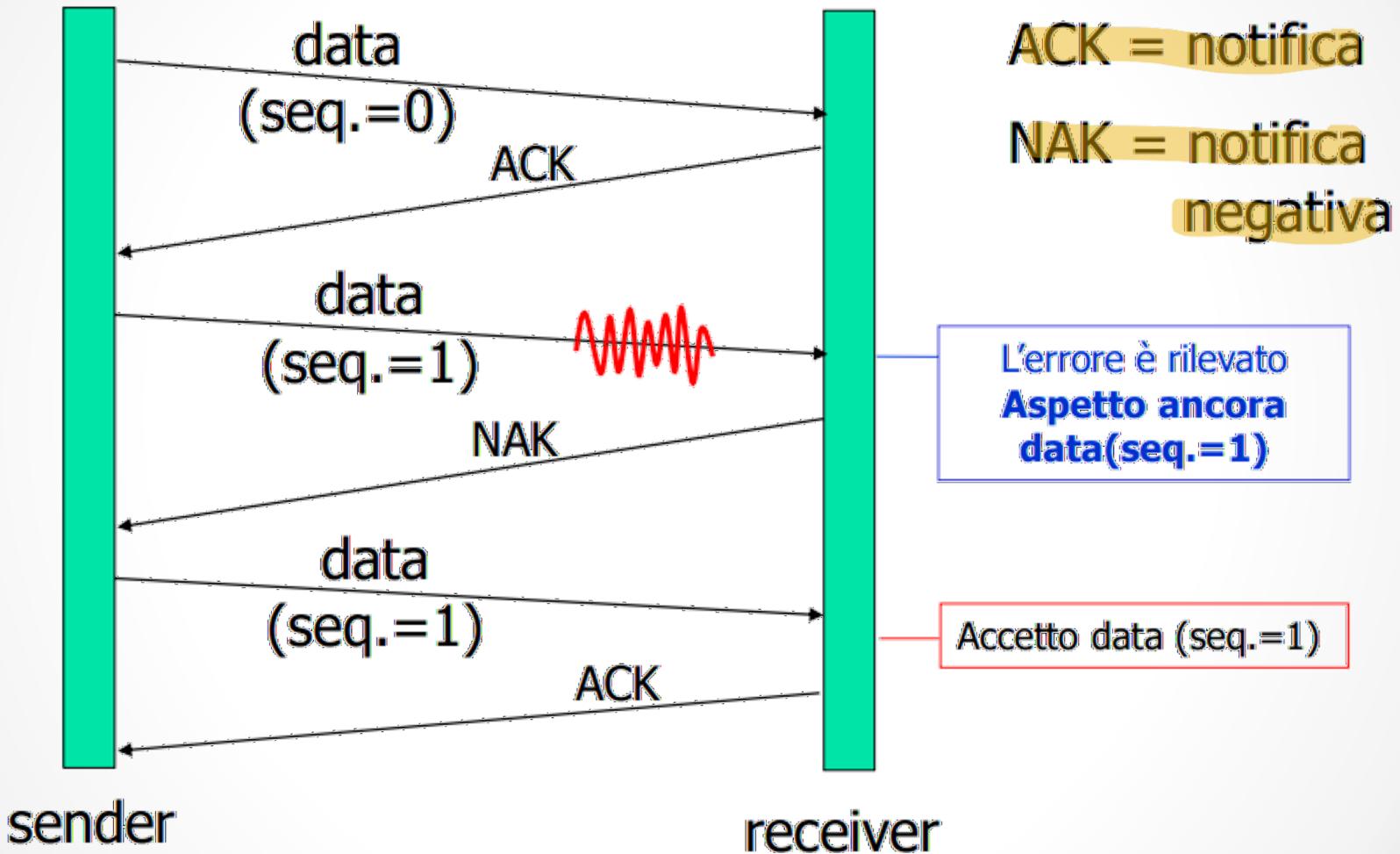
il mittente ritrasmette pacchetto corrente se ACK/NAK confuso

il ricevitore scarta pkt duplicati

Protocollo **stop & wait**: il mittente invia un pkt e aspetta la risposta del destinatario. In questo caso è sufficiente un numero di sequenza su un bit (0,1)

↓
dovendo sempre

Canale con Errori



nota: il ricevitore non può sapere se l'ultima ACK/NAK ricevuto dal mittente

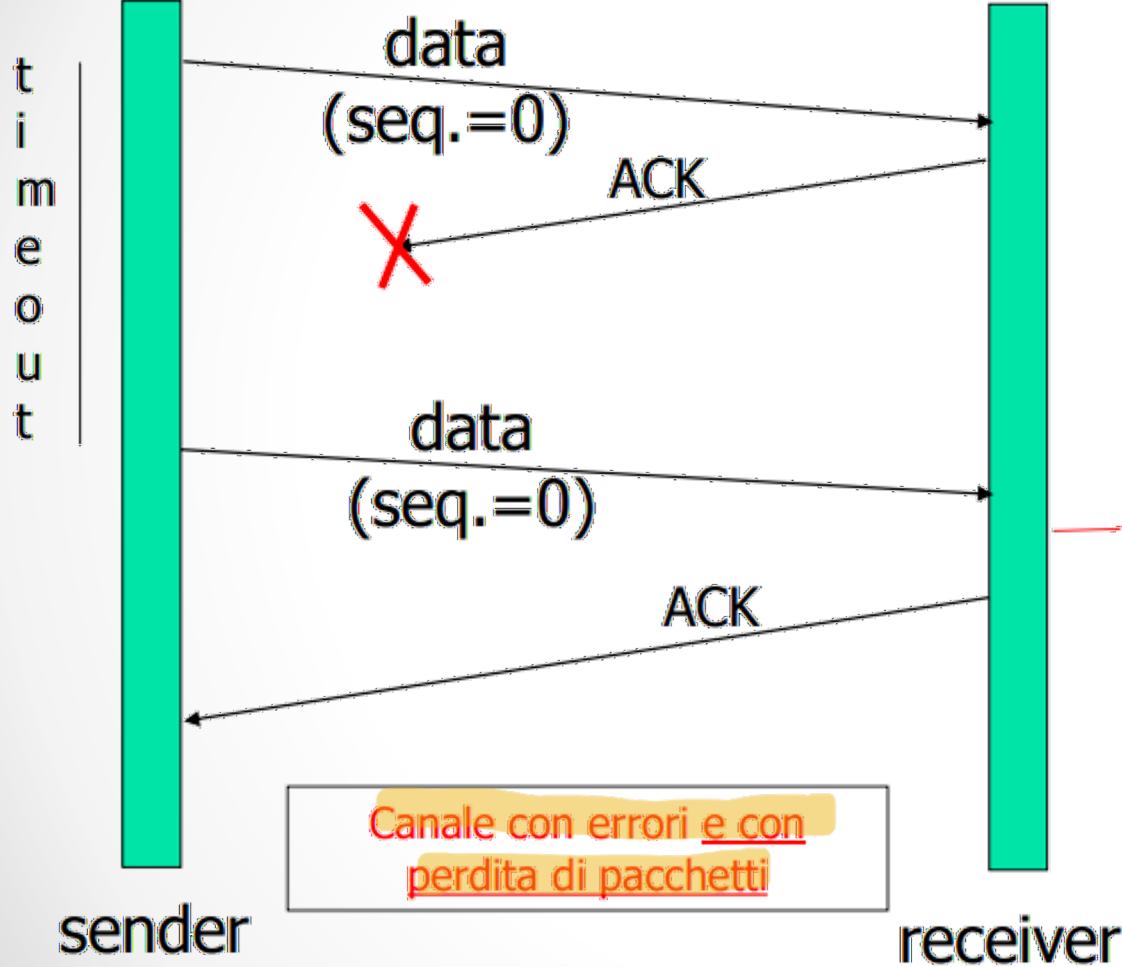
Protocollo senza NAK

Stessa funzionalità del precedente, utilizzando soltanto gli ACK

Al posto del NAK, il destinatario invia un ACK per l'ultimo pacchetto ricevuto correttamente *Manda ACK associato al num di sequenza*
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

Protocollo senza NAK



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

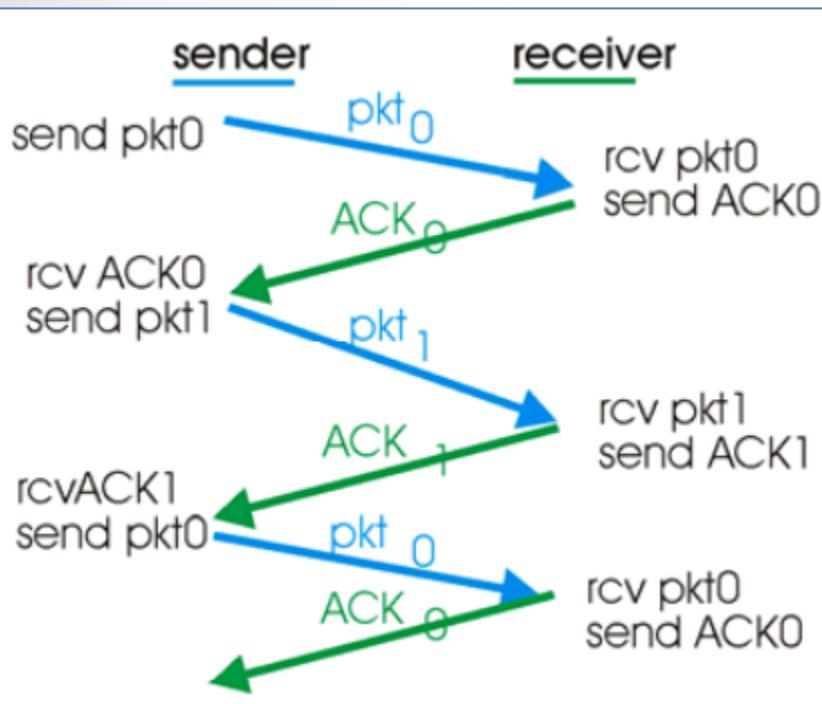
In particolare un conto alla rovescia: timeout

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

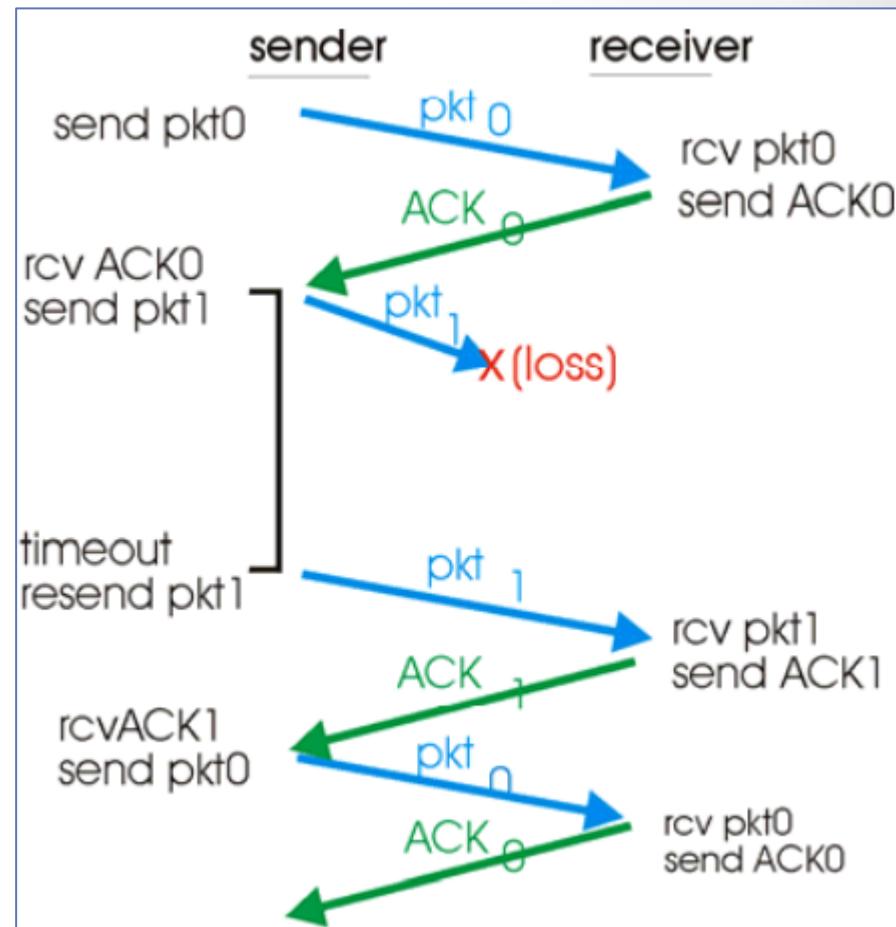
Problema: quanto dovrà essere grande il timeout?

Riassumendo...



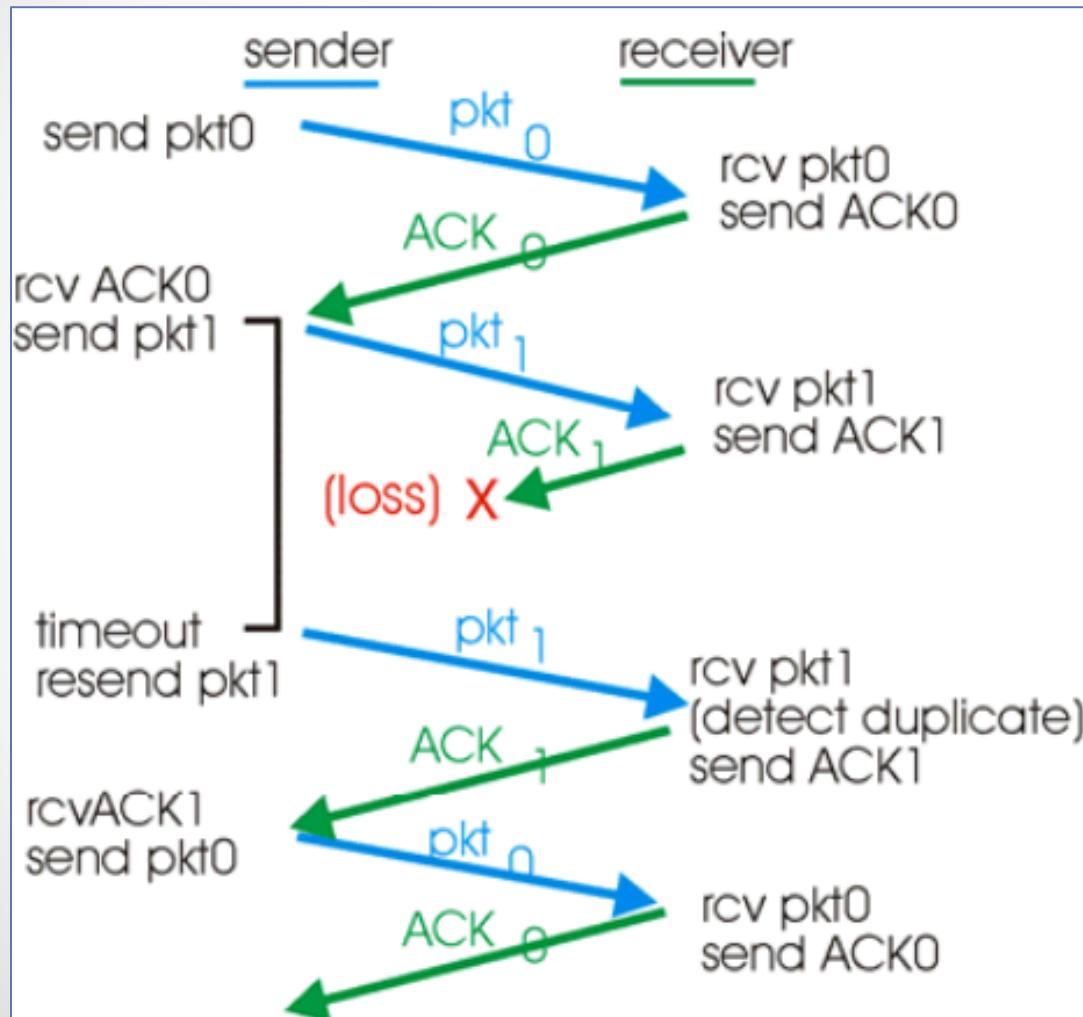
Operazione senza perdite

Turbo H_2 w/o



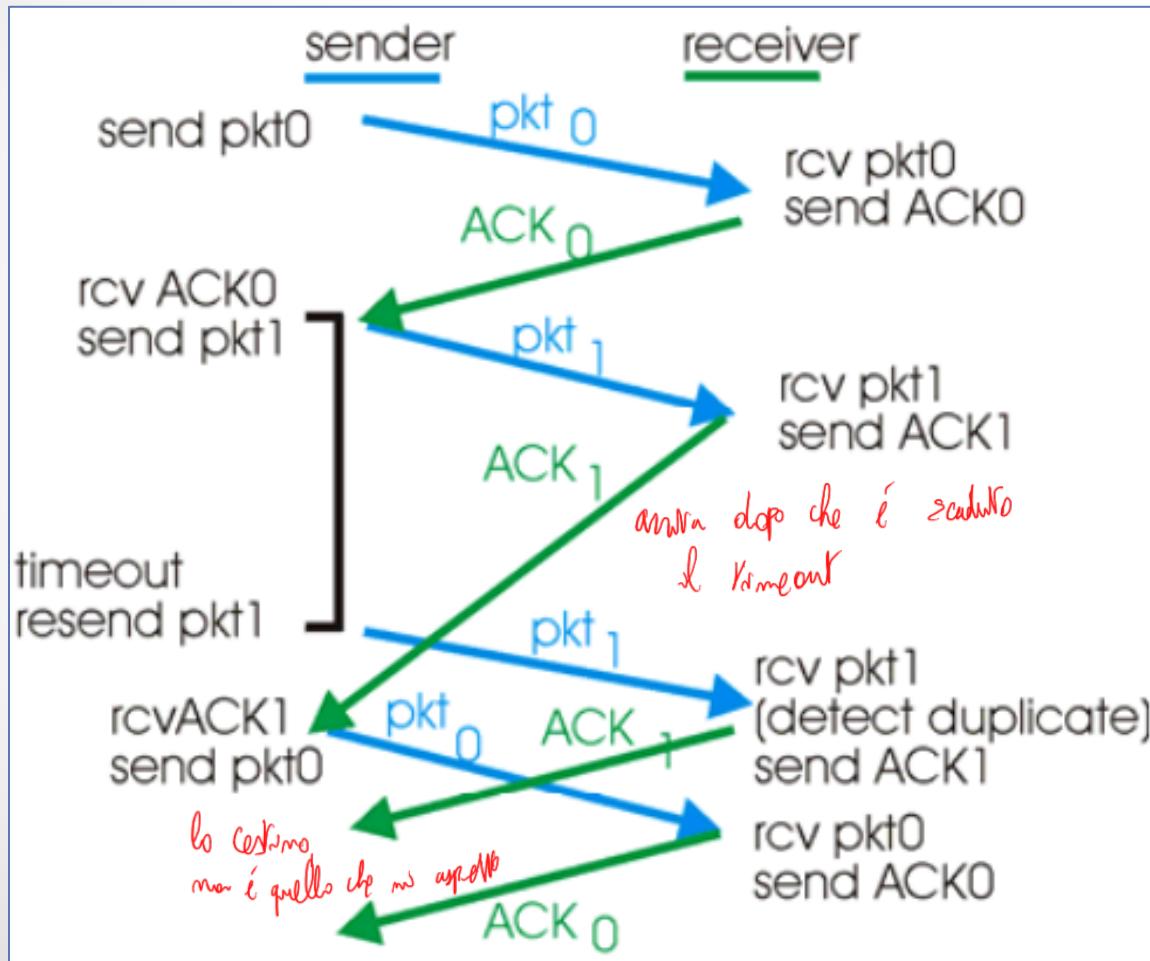
Operazione con perdita di pacchetto

Riassumendo...



Operazione con
perdita ACK

Riassumendo...



Operazione con
timeout prematuro
(anche detto
«protocollo ad
alternanza di bit»)

Performance

Funziona, ma le **performance**?

Esempio: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet

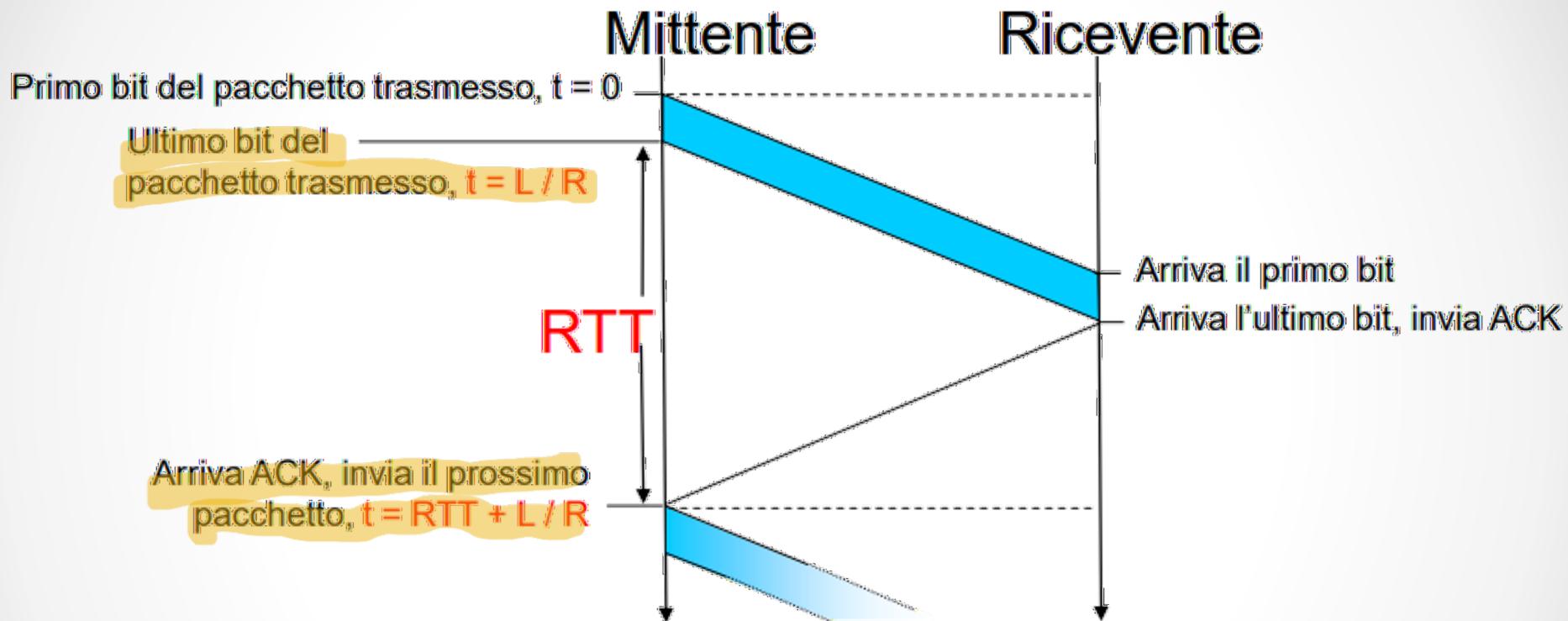
$$T_{\text{transmit}} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kb/pkt}}{10^{10} \text{ b/sec}} = 8 \text{ microsec}$$

$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Tempo che manda
sta al fine obiettivo è
minima pake: devo
aspettare RTT.

- U_{sender} : utilizzo – **frazione di tempo del mittente impegnato nell'invio**
- 1KB pkt ogni 30 msec -> 33kB/sec di throughput su 1 Gbps di collegamento
- **Il protocollo di rete limita l'uso delle risorse fisiche!**

Stop&wait



$$U_{\text{sender}} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

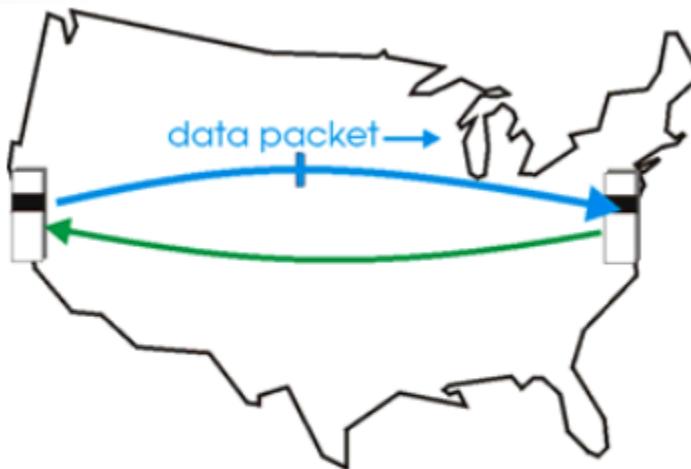
Stop & Wait (oppo) jestslowo \Rightarrow niski pipelining

Protocolli con pipeline

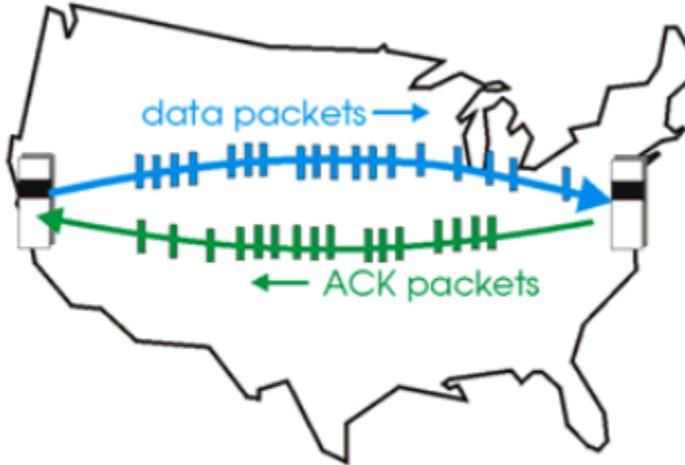
In alternativa al semplice stop&wait

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

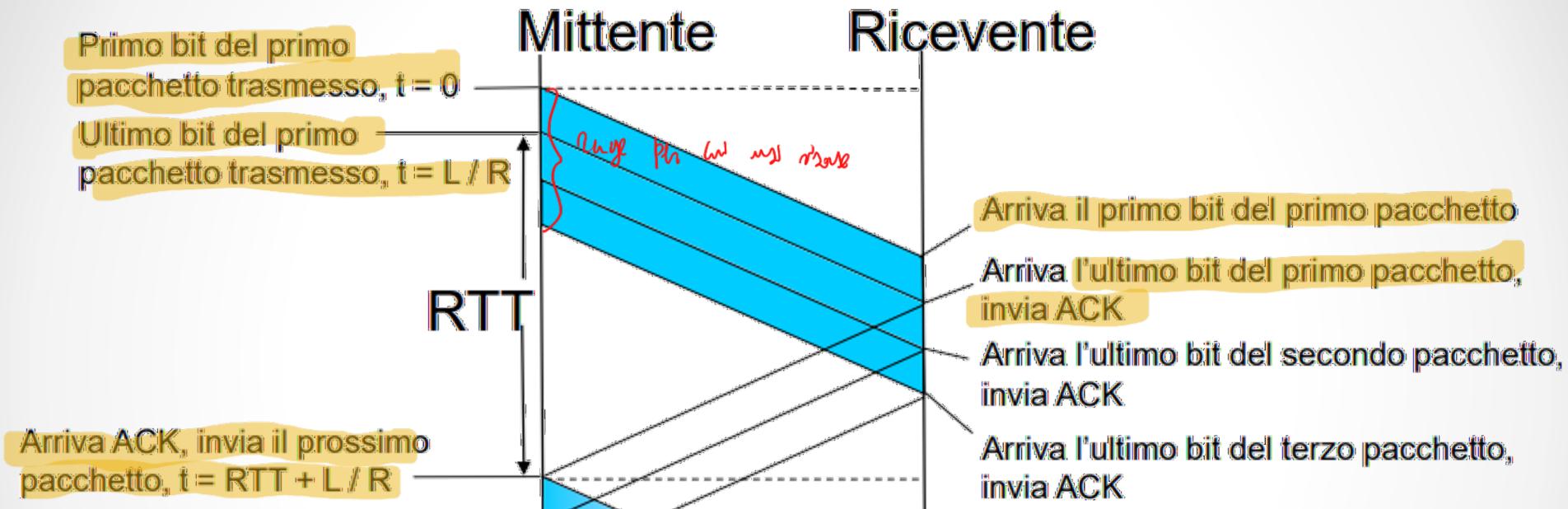


(b) a pipelined protocol in operation

- Due generiche forme di protocolli con pipelining:
 - go-Back-N, selective repeat

Non so se il primo arriva: prendo presto
di buffer che libra da rimuova quelli non ho ancora fatto ACK

Pipelining e aumento utilizzo



Aumento dell'utilizzo
di un fattore 3!

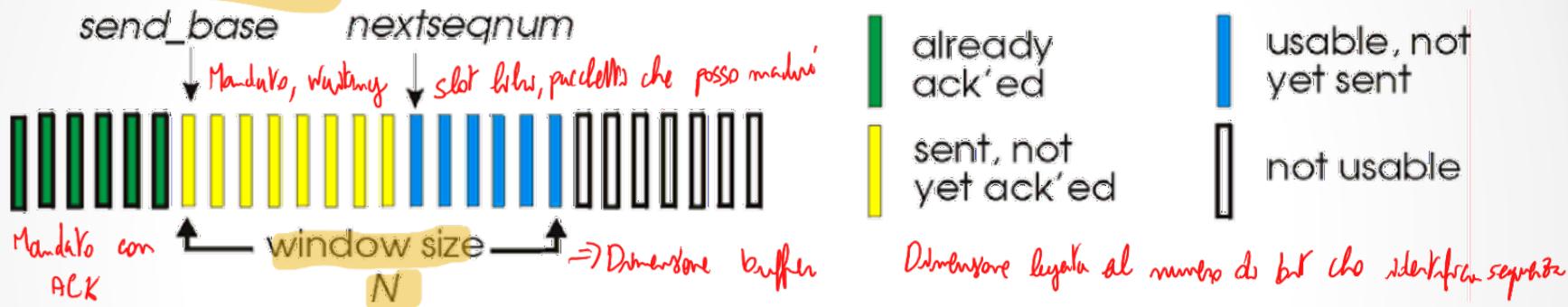
$$U_{\text{mitt}} = \frac{3 * L / R}{RTT + L / R} = \frac{0,024}{30,008} = 0,0008$$

Go back-N

Sender:

↳ Ho buffer nr. nullo

- 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 pkt 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

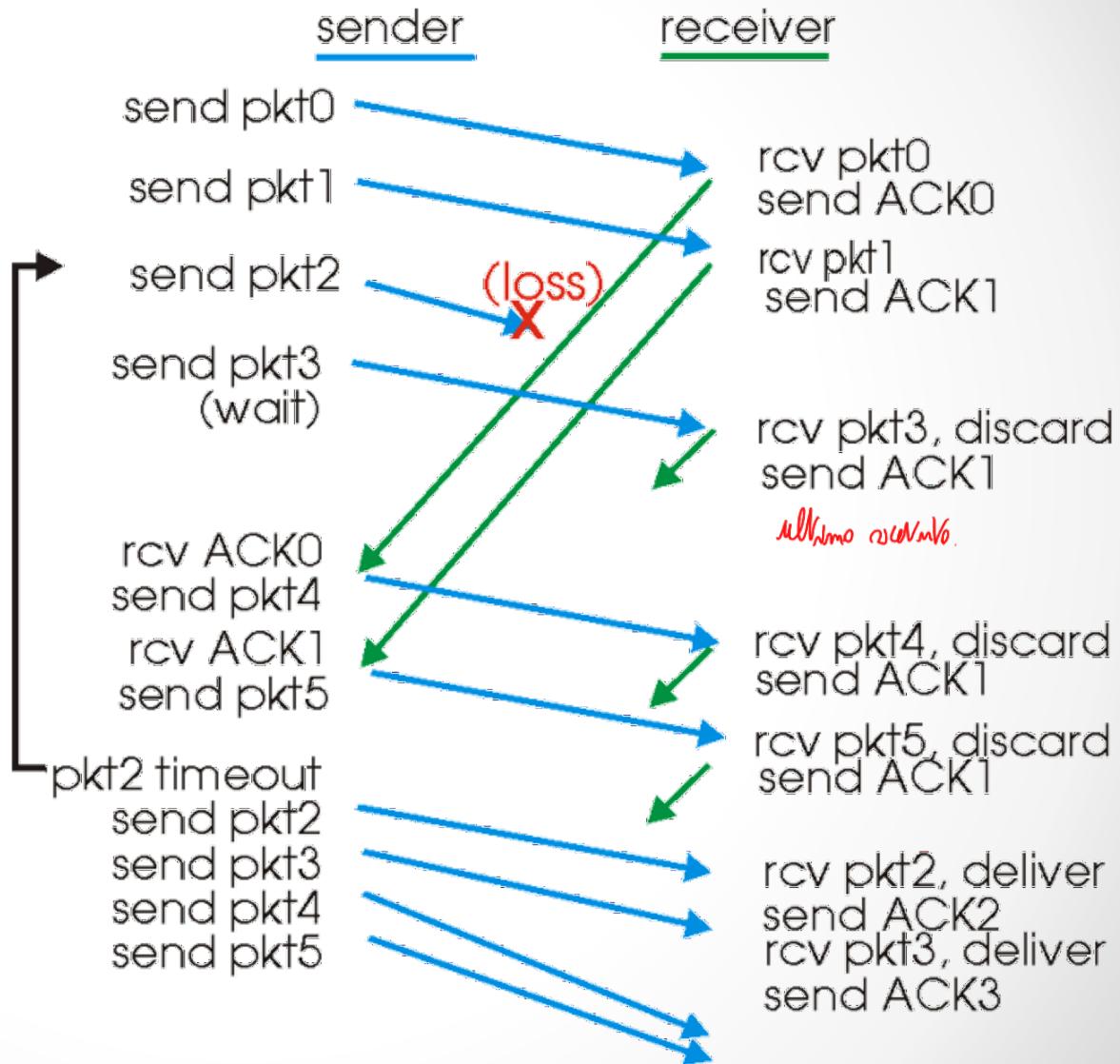
Go back-N

Finestra 6 pkt

Dimensione finestra

$$(W) \leq (2^n) - 1$$

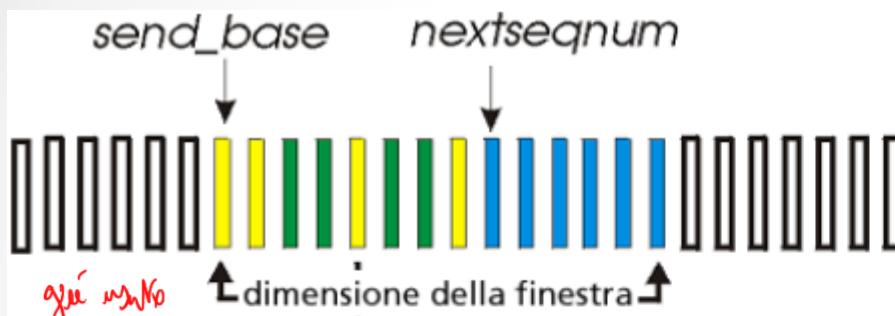
dove n è il numero di bit
utilizzati per i numeri di
sequenza



Selective repeat

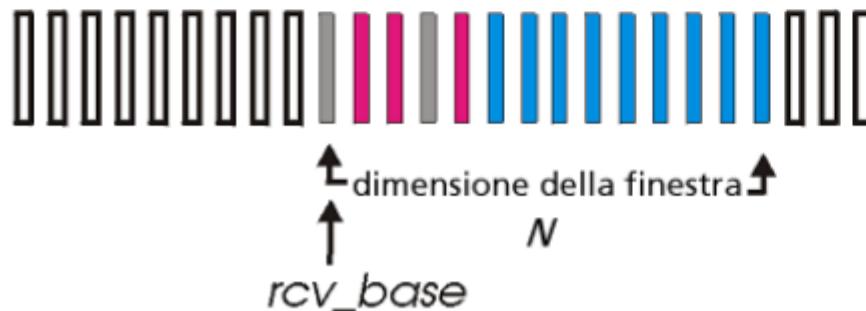
- Il ricevente invia riscontri specifici per tutti i pacchetti ricevuti correttamente
 - buffer dei pacchetti, se necessario, per eventuali consegne in sequenza al livello superiore
- Il mittente ritrasmette soltanto i pacchetti per i quali non ha ricevuto un ACK
 - timer del mittente per ogni pacchetto non riscontrato *Non 1 kilmb ma uno per ogni pacchetto*
- Finestra del mittente
 - N numeri di sequenza consecutivi
 - limita ancora i numeri di sequenza dei pacchetti inviati non riscontrati

Selective repeat



N Non posso far arrivare frame se non ricevercelo più prima

a) Visione del mittente sui numeri di sequenza



b) Visione del ricevente sui numeri di sequenza

Selective repeat

Sender

Data da inviare:

Se il prossimo num di seq nella finestra, invia pkt

Timeout(n): rimanda pkt n e riavvia timer

Ack(n) in
[[sendbase, sendbase+N]-1]
↑
Finestra

Marca il pkt n come ricevuto

Se è il più piccolo pkt non acked avanza la window base al prossimo pacchetto non acked

Receiver

Pkt n in [rcvbase, rcvbase+N-1]

Invia Ack(n)

Non in ordine: buffer

In ordine: consegna, avanza finestra al prossimo pkt non ancora ricevuto

Pkt n in [rcvbase-N, rcvbase-1]

Ack (n)

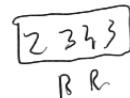
Altrimenti ignora

↑
oluphicio

Selective repeat

pkt0 sent

0 1 2 3 4 5 6 7 8 9



pkt1 sent

0 1 2 3 4 5 6 7 8 9

pkt2 sent

0 1 2 3 4 5 6 7 8 9

pkt3 sent, window full

0 1 2 3 4 5 6 7 8 9

ACK0 rcvd, pkt4 sent

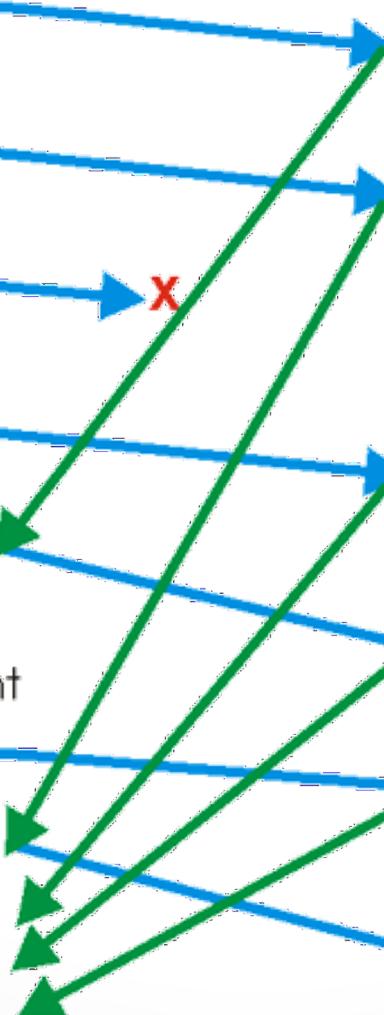
0 1 2 3 4 5 6 7 8 9

pkt2 timeout, pkt2 resent

0 1 2 3 4 5 6 7 8 9

ACK1 rcvd, pkt5 sent

0 1 2 3 4 5 6 7 8 9



Rcvd 0, shift

pkt0 rcvd, delivered, ACK0 sent

0 1 2 3 4 5 6 7 8 9

pkt1 rcvd, delivered, ACK1 sent

0 1 2 3 4 5 6 7 8 9

2 more arrvng

3, 4 arrvng
pkt3 rcvd, buffered, ACK3 sent

0 1 2 3 4 5 6 7 8 9

pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9

pkt2 rcvd, deliver pkts 2, 3, 4
ACK2 sent

0 1 2 3 4 5 6 7 8 9

pkt5 rcvd, delivered, ACK5 sent
0 1 2 3 4 5 6 7 8 9

Selective repeat - problema

HP: finestre 3 elementi

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)

