

# **Tecniche ARQ (protocolli a finestra)**

***Prof. S. Marano  
Università della Calabria  
A.A. 2012-2013***

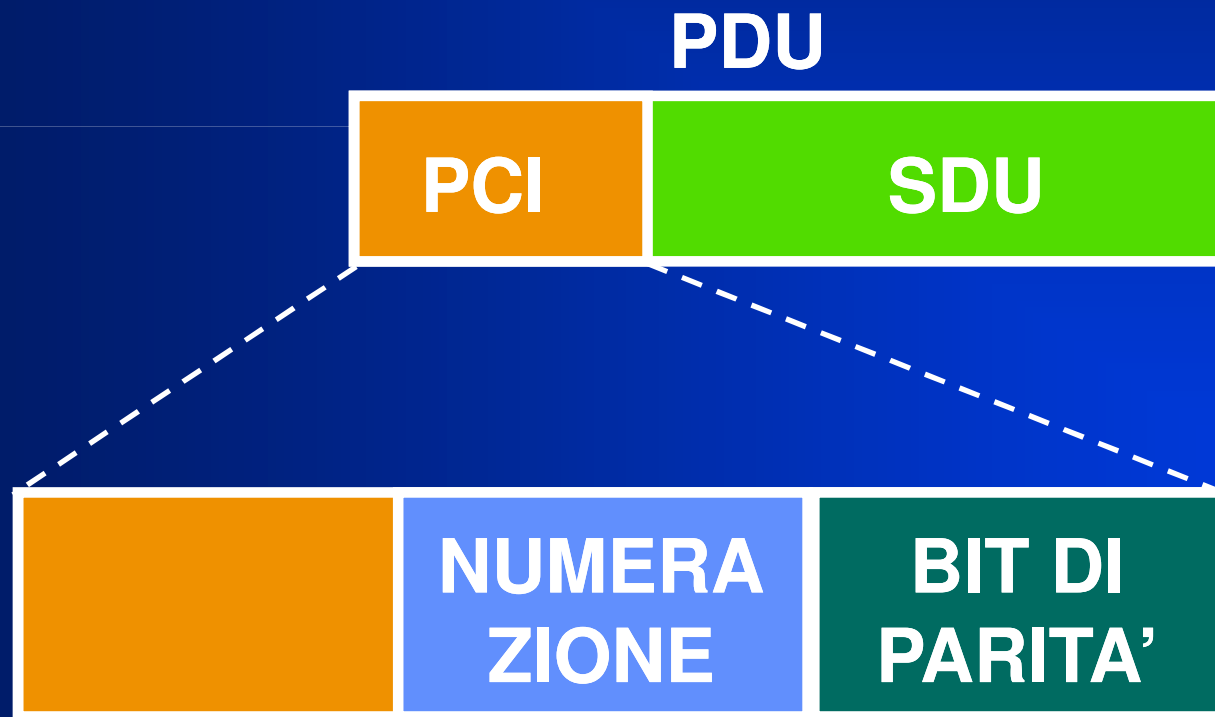
**ARQ (automatic retransmission request)**

**controllo congiunto di**

- **errore**
- **flusso**
- **sequenza**

**su una connessione**

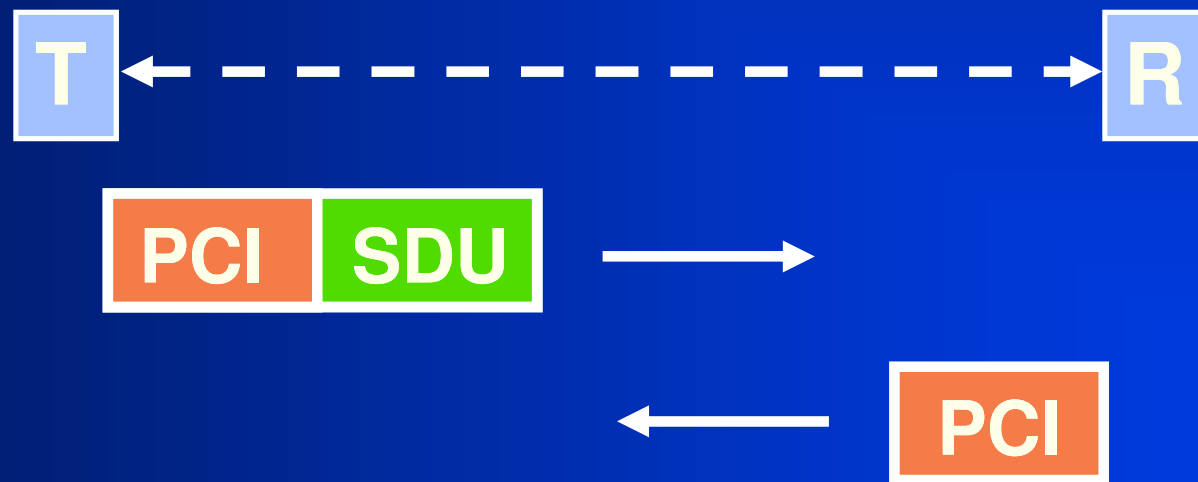
**Si introducono bit di numerazione  
tra le informazioni di controllo  
all'interno delle PDU**

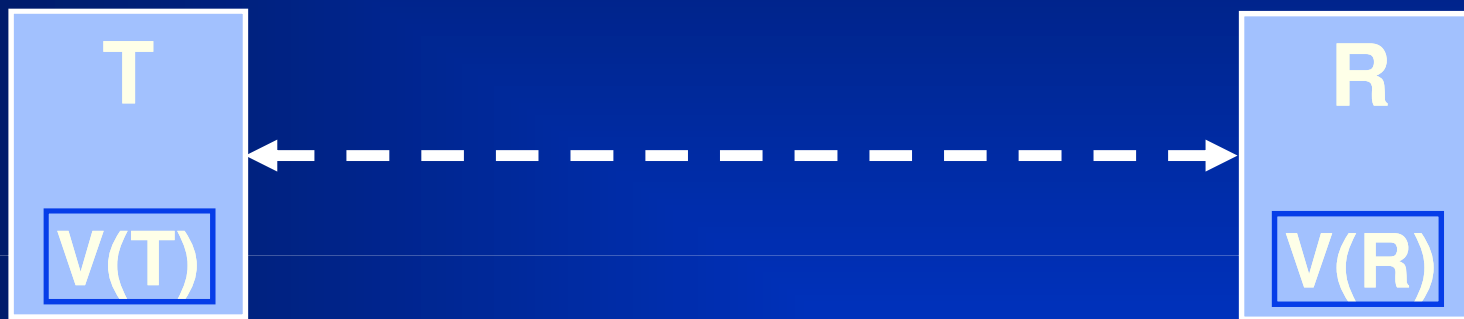


## **Tre tecniche ARQ**

- **Stop and wait (Alternating bit)**
- **Go back N**
- **Selective repeat**

**descriviamo le tre tecniche in un ambiente di comunicazione unidirezionale**





- bit di parità
- $N(T)$   
numero d'ordine
- indirizzi



• **N(R)**

**numero d'ordine atteso**

• **indirizzi**

## **Stop and wait**

**il trasmettitore**

- **invia una PDU**
- **attiva un orologio (tempo di timeout)**
- **si pone in attesa della conferma di ricezione (acknowledgment - ACK)**
- **se scade il timeout prima dell'arrivo della conferma, ripete la trasmissione**



## **Stop and wait**

**il trasmettitore, quando riceve un ACK**

- controlla la correttezza dell'ACK**
- controlla il numero di sequenza**
- se l'ACK è relativo all'ultima PDU trasmessa, si abilita la trasmissione della prossima PDU**

## **Stop and wait**

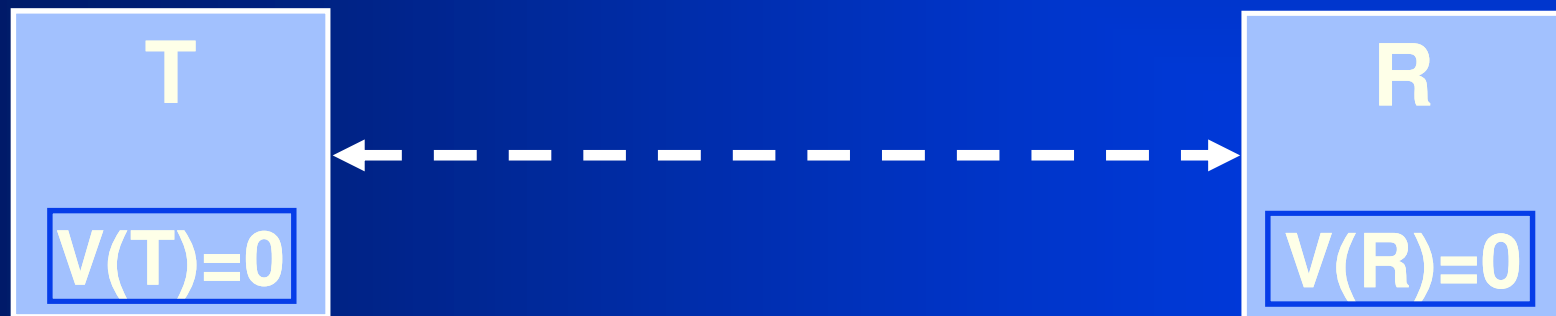
**il ricevitore**

- **riceve una PDU**
- **controlla la correttezza della PDU**
- **controlla il numero di sequenza**
- **se la PDU è corretta invia la conferma di ricezione**

## Inizializzazione

$V(T) = 0$  al trasmettitore

$V(R) = 0$  al ricevitore



## Trasmissione di una PDU con $N(T) = V(T)$ Avvio dell' orologio



**Ricezione di una PDU**

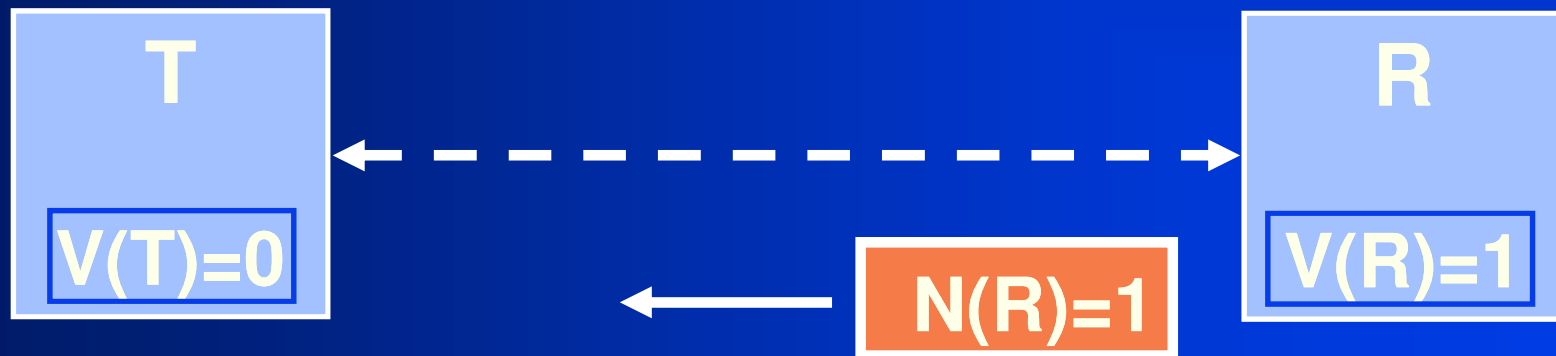
**Controllo di correttezza**

**Controllo di sequenza:  $N(T) = V(R)$  ?**



**Incremento di  $V(R)$**

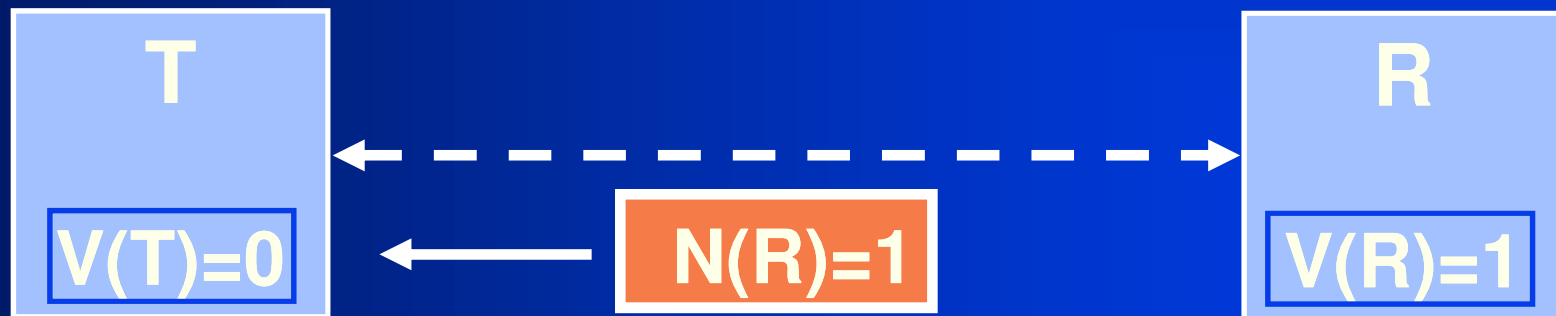
**Trasmissione di un ACK con  $N(R) = V(R)$**



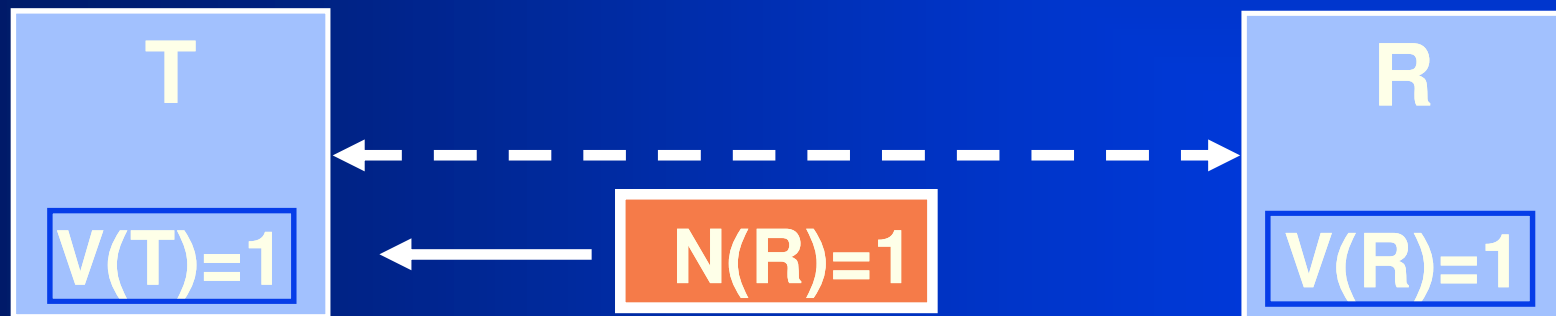
**Ricezione di un ACK**

**Controllo di sequenza:  $N(R) = V(T) + 1$  ?**

**Arresto dell' orologio**

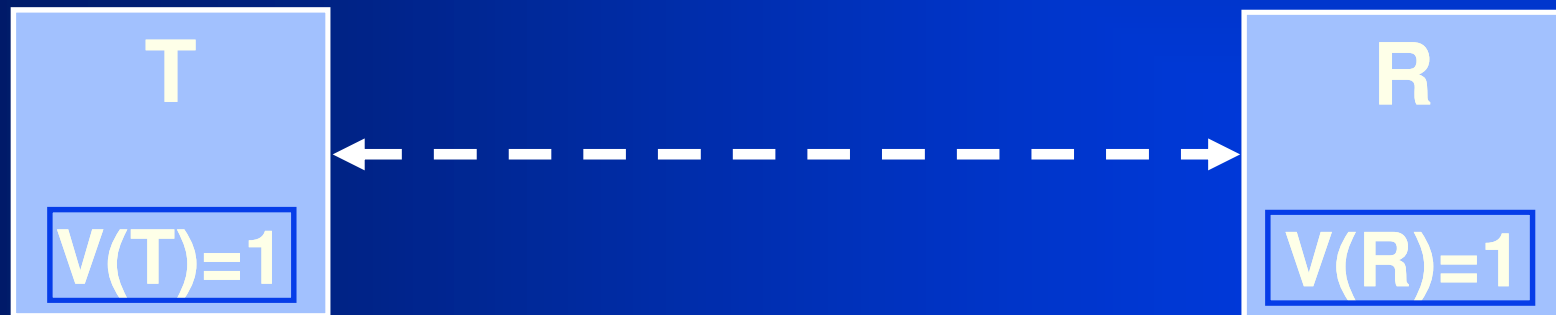


## Incremento di $V(T)$

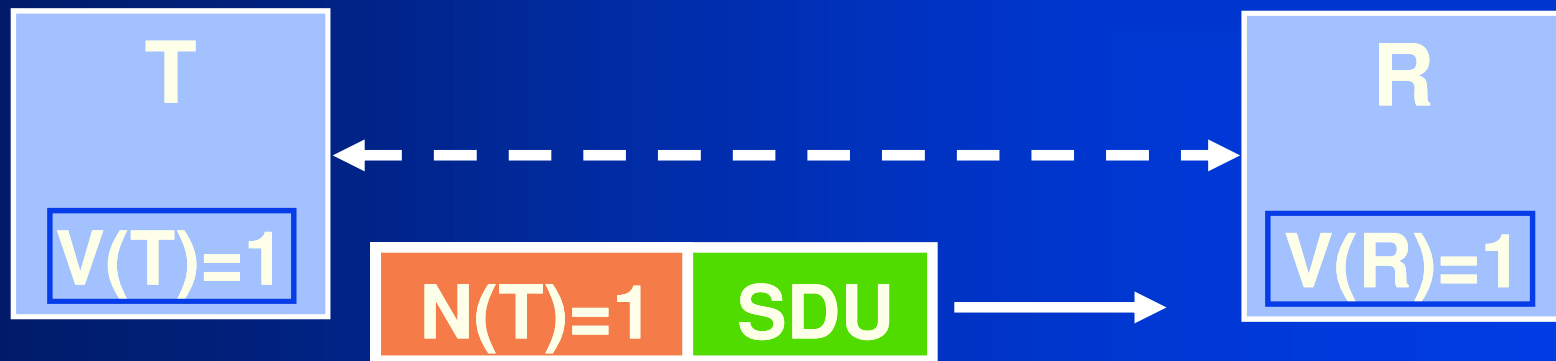




$V(T) = 1$  al trasmettitore  
 $V(R) = 1$  al ricevitore



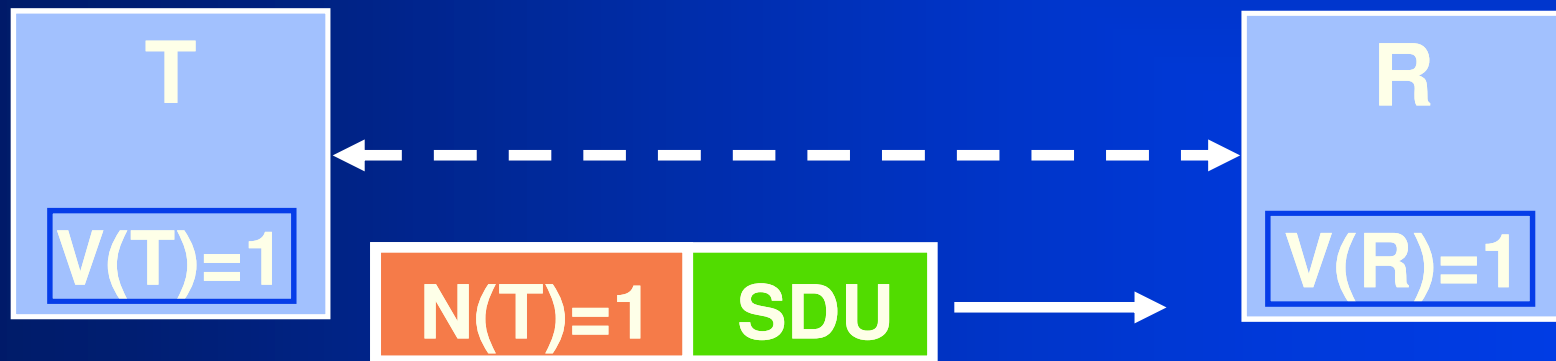
## Trasmissione di una PDU con $N(T) = V(T)$ Avvio dell' orologio



**Ricezione di una PDU**

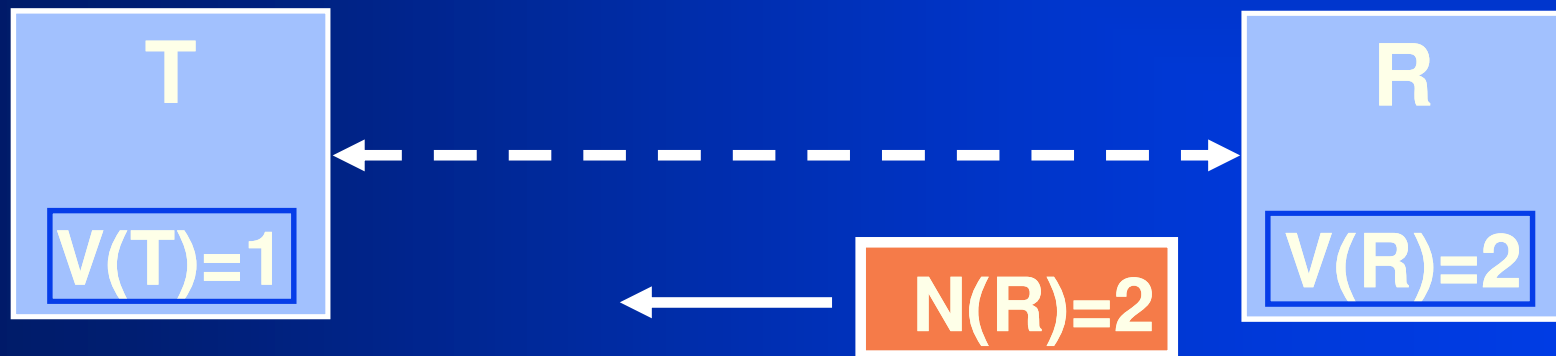
**Controllo di correttezza**

**Controllo di sequenza:  $N(T) = V(R)$  ?**



**Incremento di  $V(R)$**

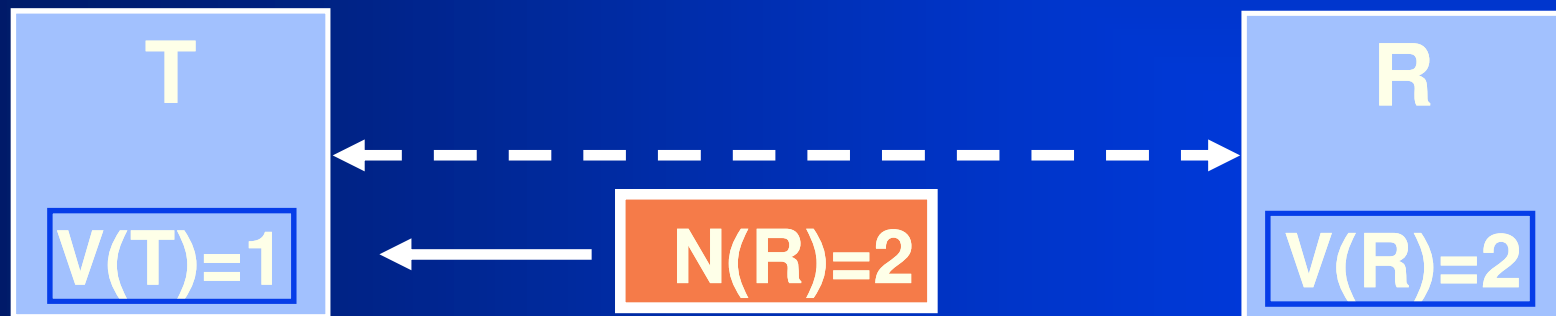
**Trasmissione di un ACK con  $N(R) = V(R)$**



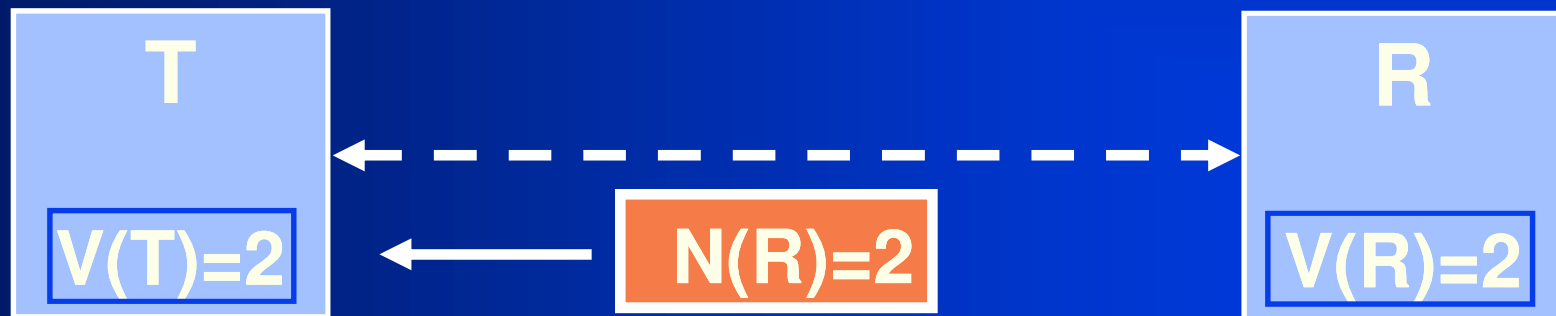
**Ricezione di un ACK**

**Controllo di sequenza:  $N(R) = V(T) + 1$  ?**

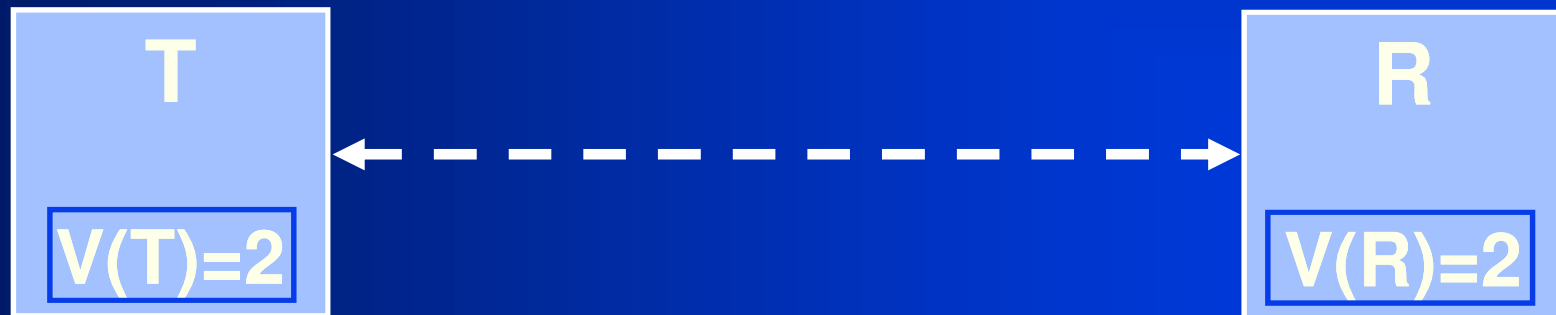
**Arresto dell' orologio**

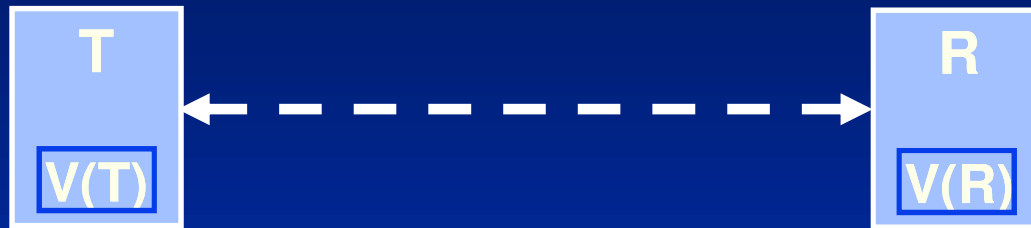


## Incremento di $V(T)$



$V(T) = 2$  al trasmettitore  
 $V(R) = 2$  al ricevitore





$$V(T) = 0$$

$$V(R) = 0$$

$$N(T) = 0$$

$$V(T) = 0$$

$$V(R) = 1$$

$$N(R) = 1$$

$$V(T) = 1$$

$$V(R) = 1$$

$$N(T) = 1$$

$$V(T) = 1$$

$$V(R) = 2$$

$$N(R) = 2$$

$$V(T) = 2$$

$$V(R) = 2$$

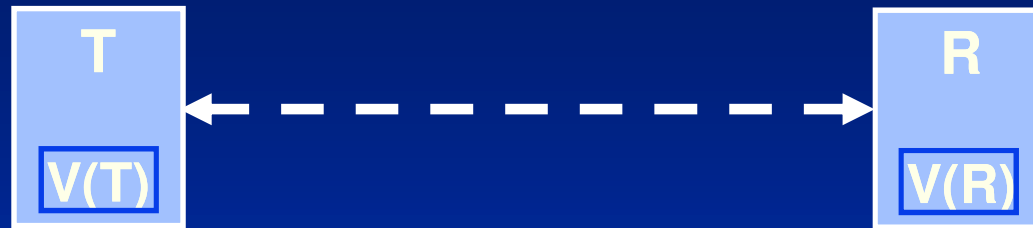


## **La numerazione delle PDU e`**

- indispensabile**
- ciclica**

**Basta un solo bit per la numerazione**

**Alternating bit protocol**

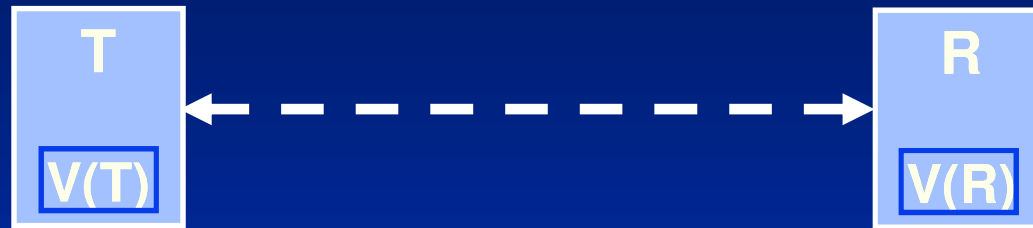

$$V(T) = 0$$
$$V(R) = 0$$

# $N(T) = 0$

$$V(T) = 0$$
$$V(R) = 1$$

# N (R) = 1

$$V(T) = 1$$
$$V(R) = 1$$
$$N(T) = 1$$
$$V(T) = 1$$
$$V(R) = 0$$
$$N(R) = 0$$
$$V(T) = 0$$
$$V(R) = 0$$



$$V(T) = 0$$

$$V(R) = 0$$

$$N(T) = 0$$

$$V(T) = 0$$

$$V(R) = 1$$

$$N(R) = 1$$

$$V(T) = 1$$

$$V(R) = 1$$

$$N(T) = 1$$

$$V(T) = 1$$

$$V(R) = 0$$

$$N(R) = 0$$

$$V(T) = 0$$

$$V(R) = 0$$



$V(T) = 0$        $N(T) = 0$        $V(R) = 0$

$V(T) = 0$        $N(R) = 1$        $V(R) = 1$

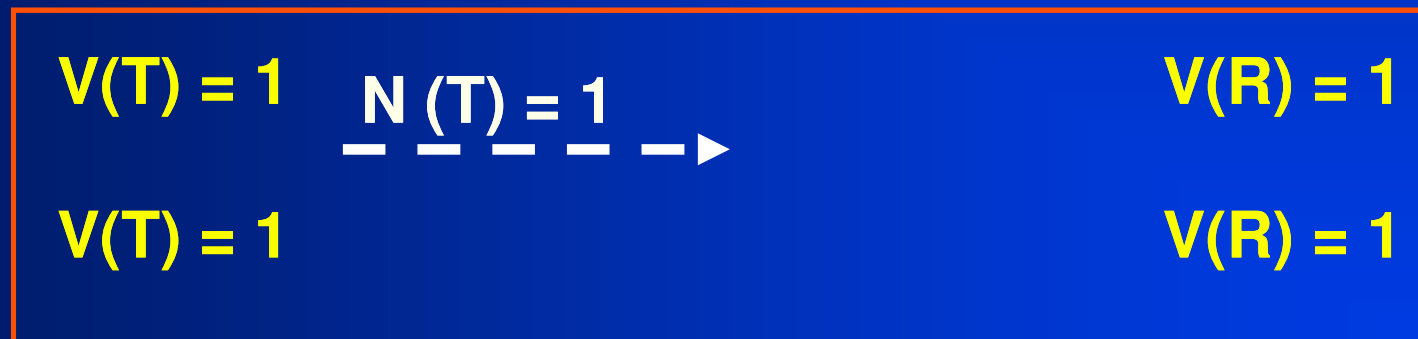
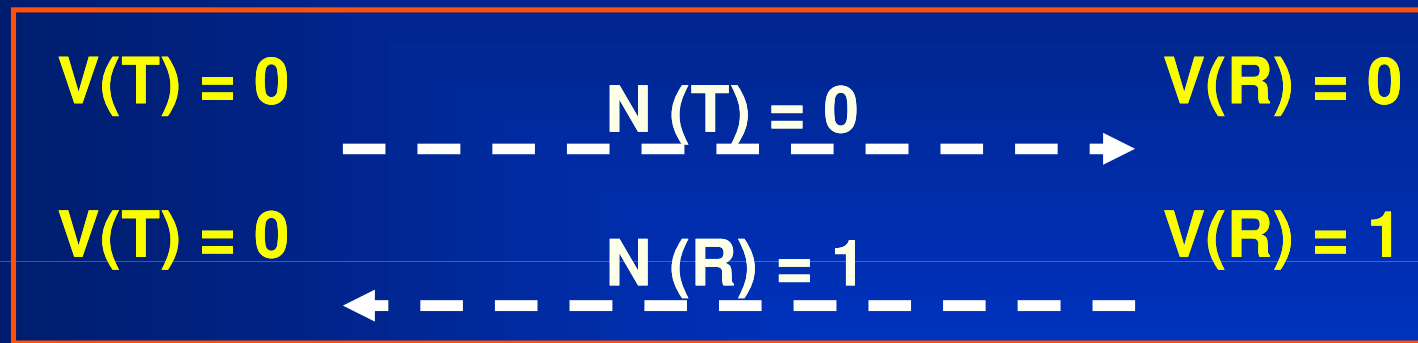
$V(T) = 1$        $N(T) = 1$        $V(R) = 1$

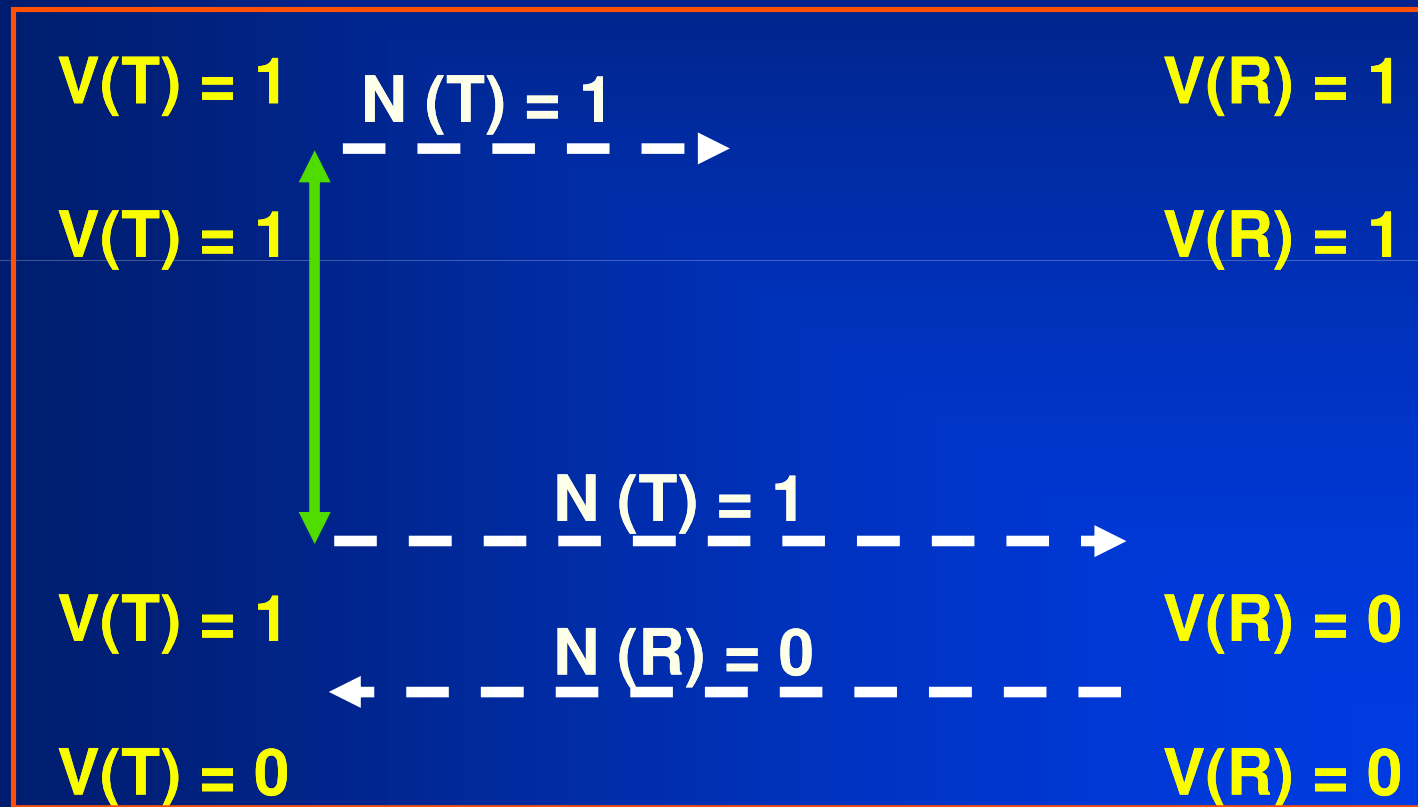
$V(T) = 1$        $N(R) = 0$        $V(R) = 0$

$V(T) = 0$

$V(R) = 0$

**Ricezione di una PDU errata**

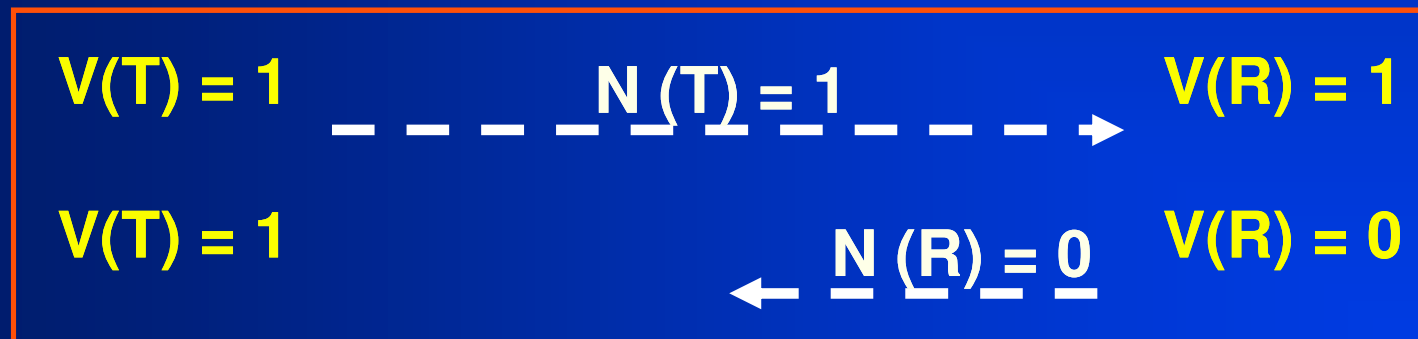
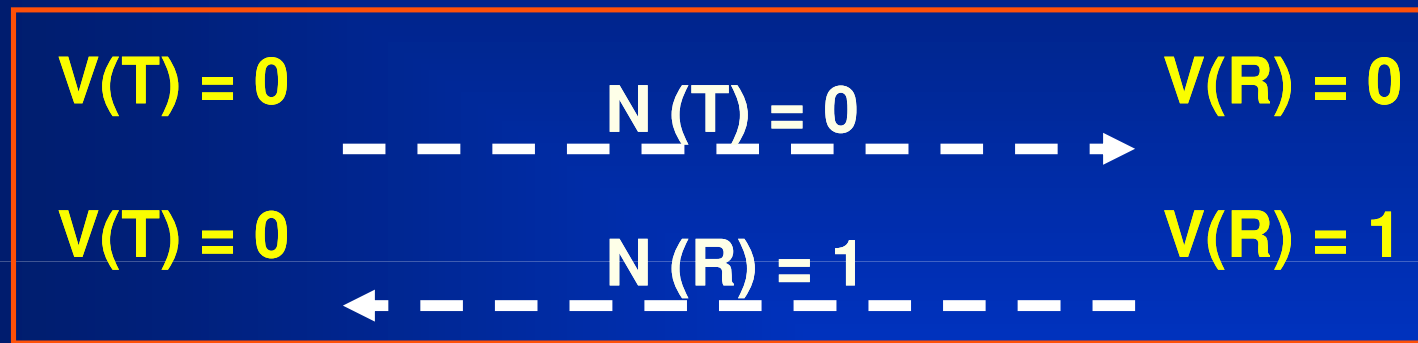


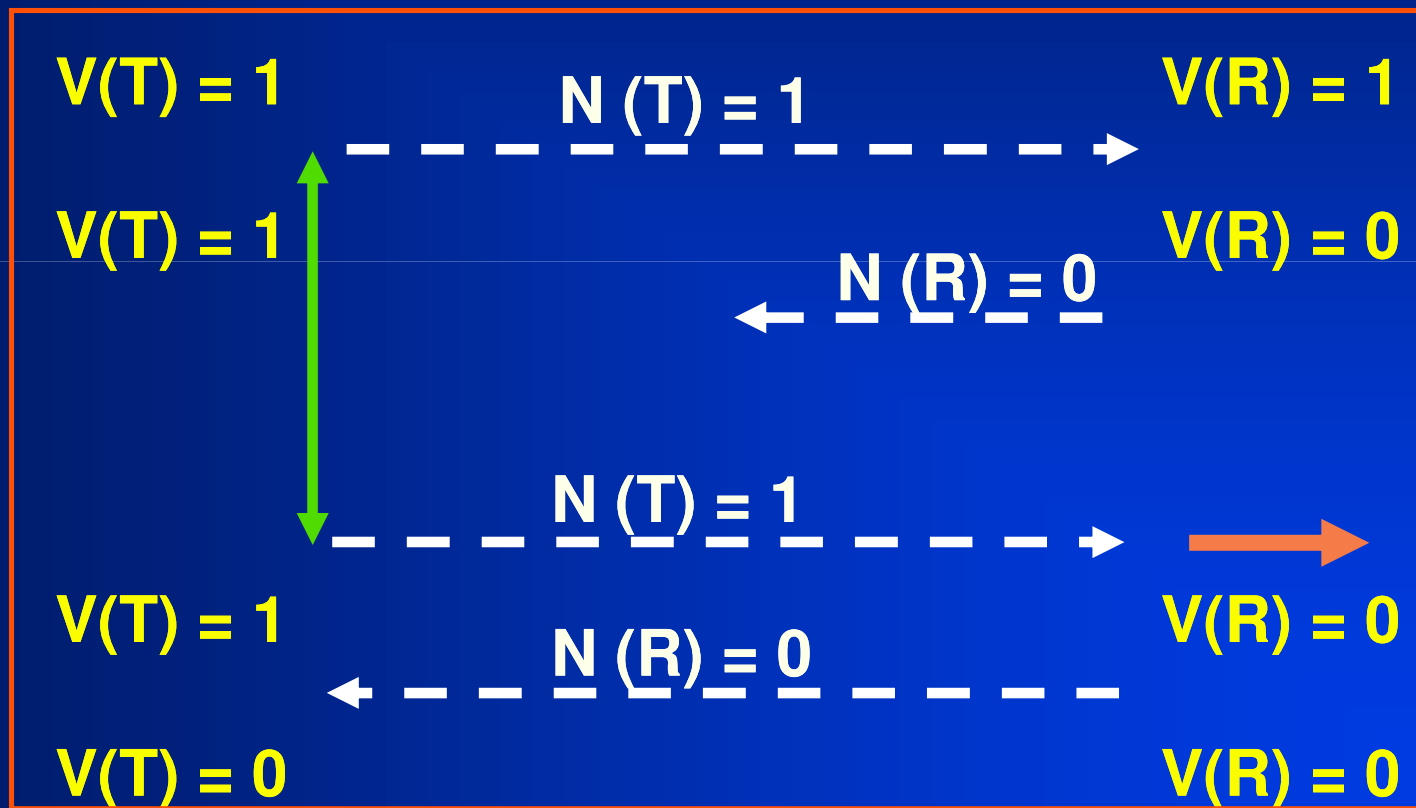




**La regolazione del timeout e` delicata**

**Ricezione di una conferma errata**





**Il trasmettitore non distingue i due casi:**

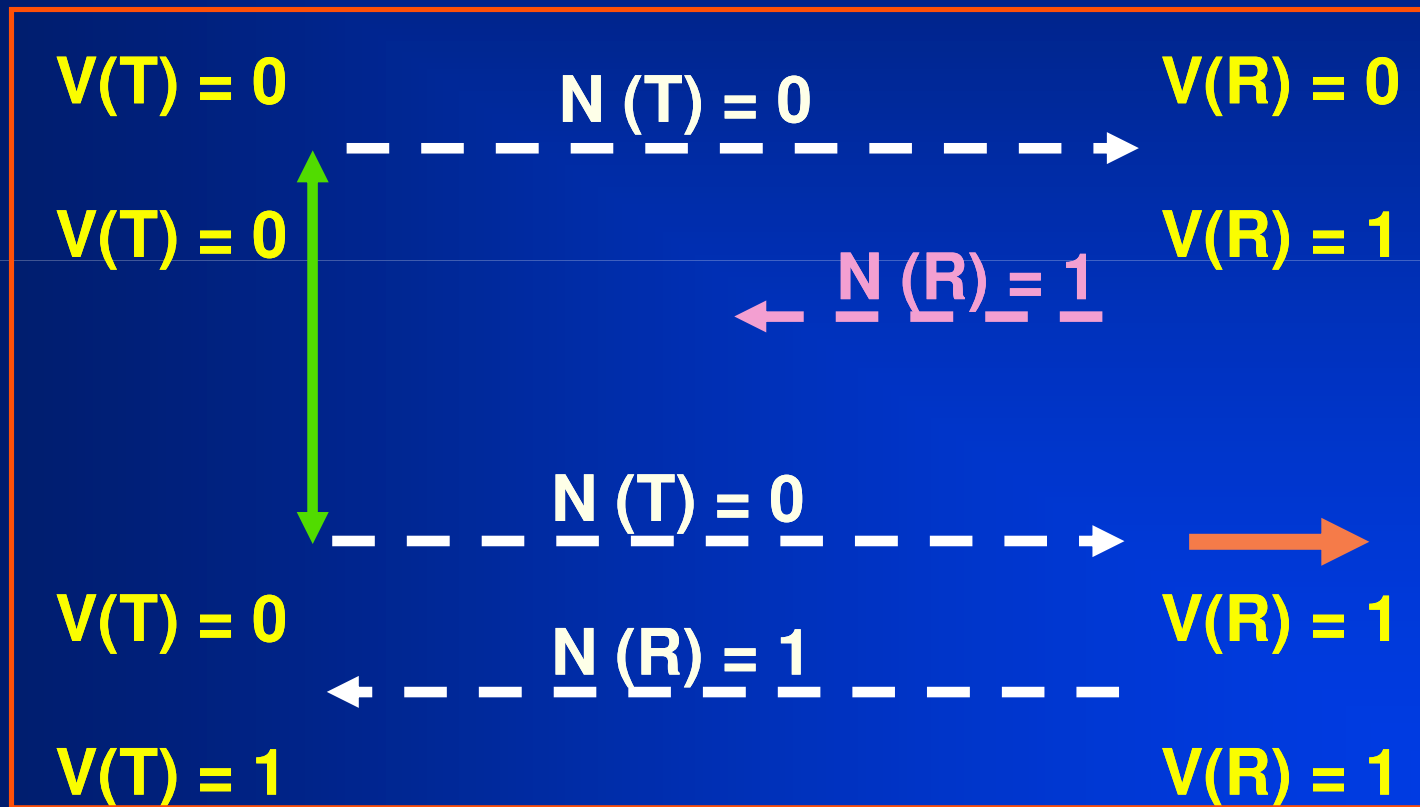
- Ricezione di una PDU errata**
- Ricezione di una conferma errata**

**In entrambi i casi aspetta lo scadere del timeout per ritrasmettere la PDU dati.**

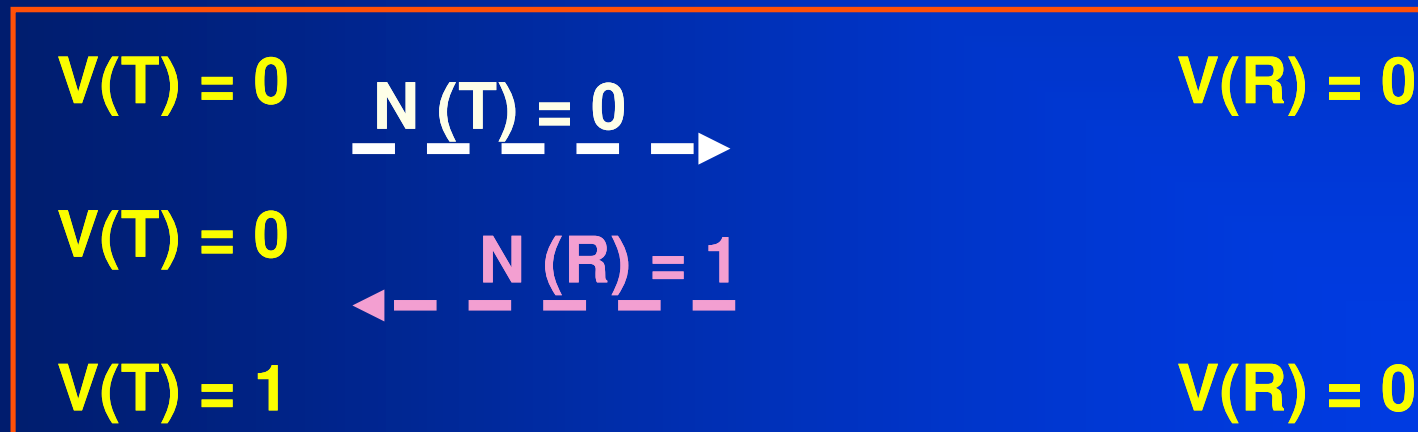
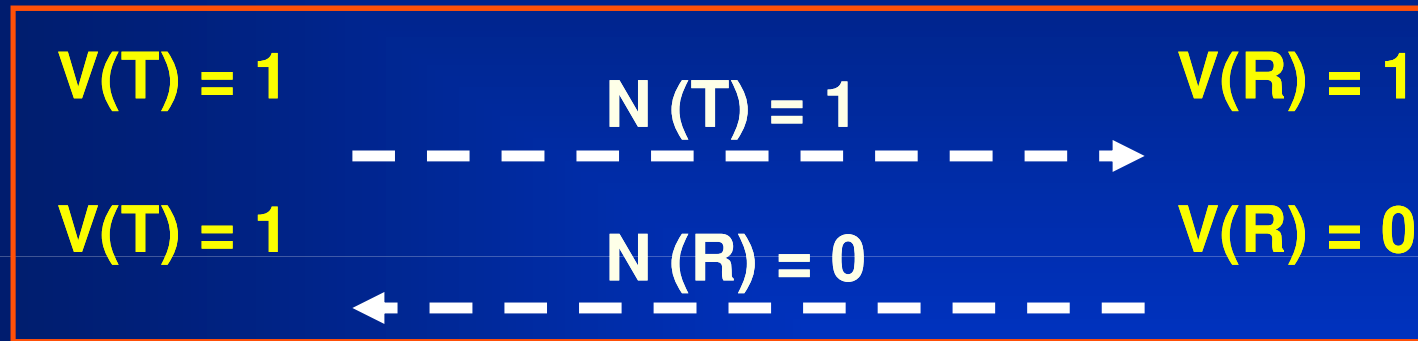
**Canale non sequenziale:  
il canale non mantiene la sequenza delle PD  
trasmesse**

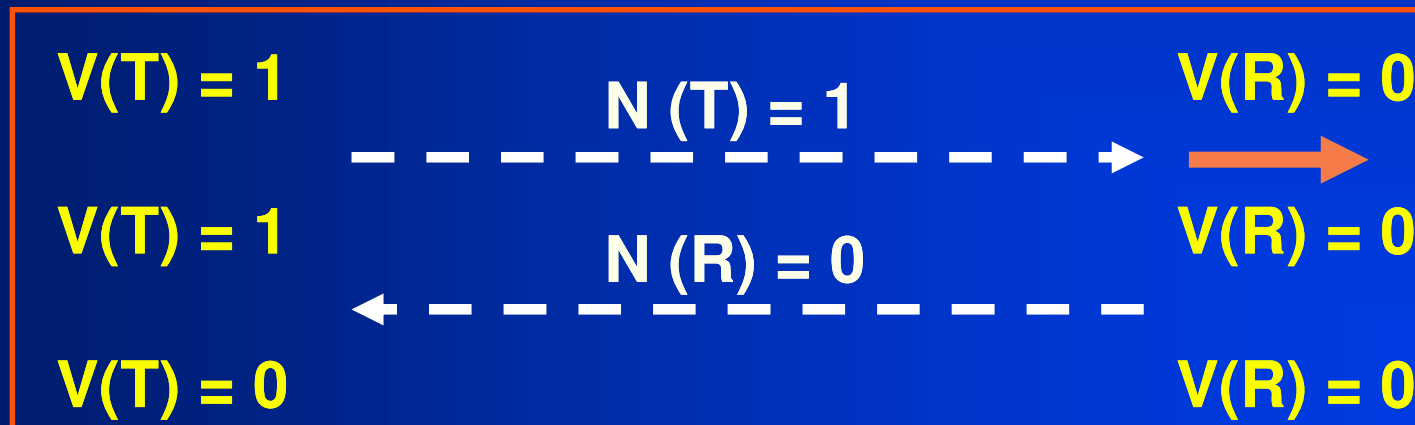
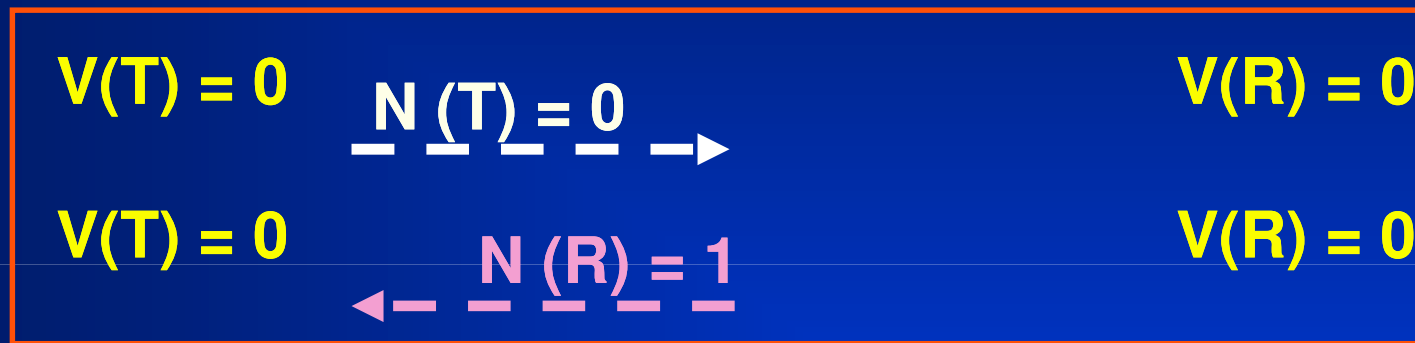
## **Si verificano malfunzionamenti**

- **perdita di PDU**
- **duplicazione di PDU**











$V(T) = 0$ 
 $N(T) = 0$ 
 $V(R) = 0$

$V(T) = 0$ 
 $N(R) = 1$ 
 $V(R) = 1$

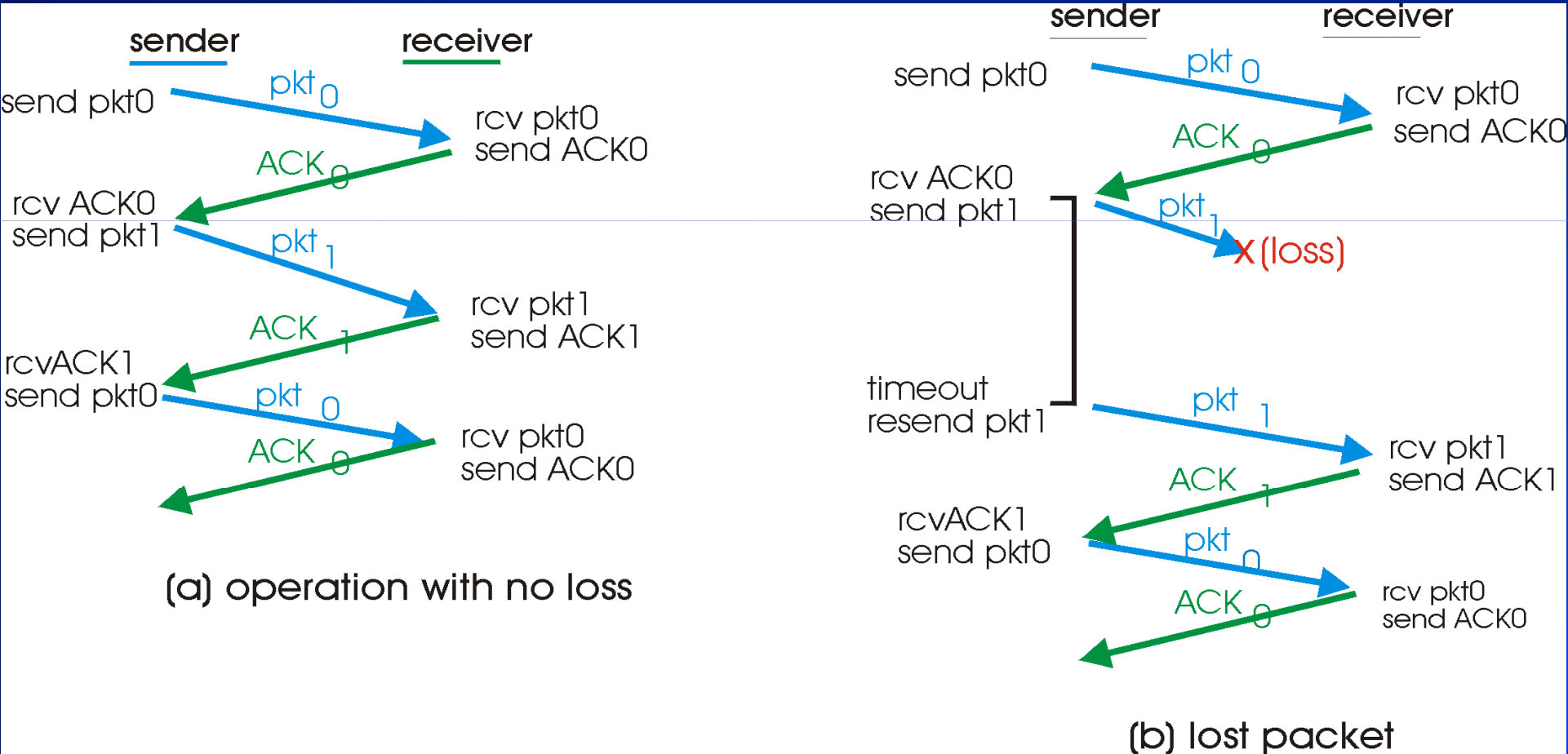
$V(T) = 1$ 
 $N(T) = 1$ 
 $V(R) = 1$

$V(T) = 1$ 
 $N(R) = 0$ 
 $V(R) = 0$

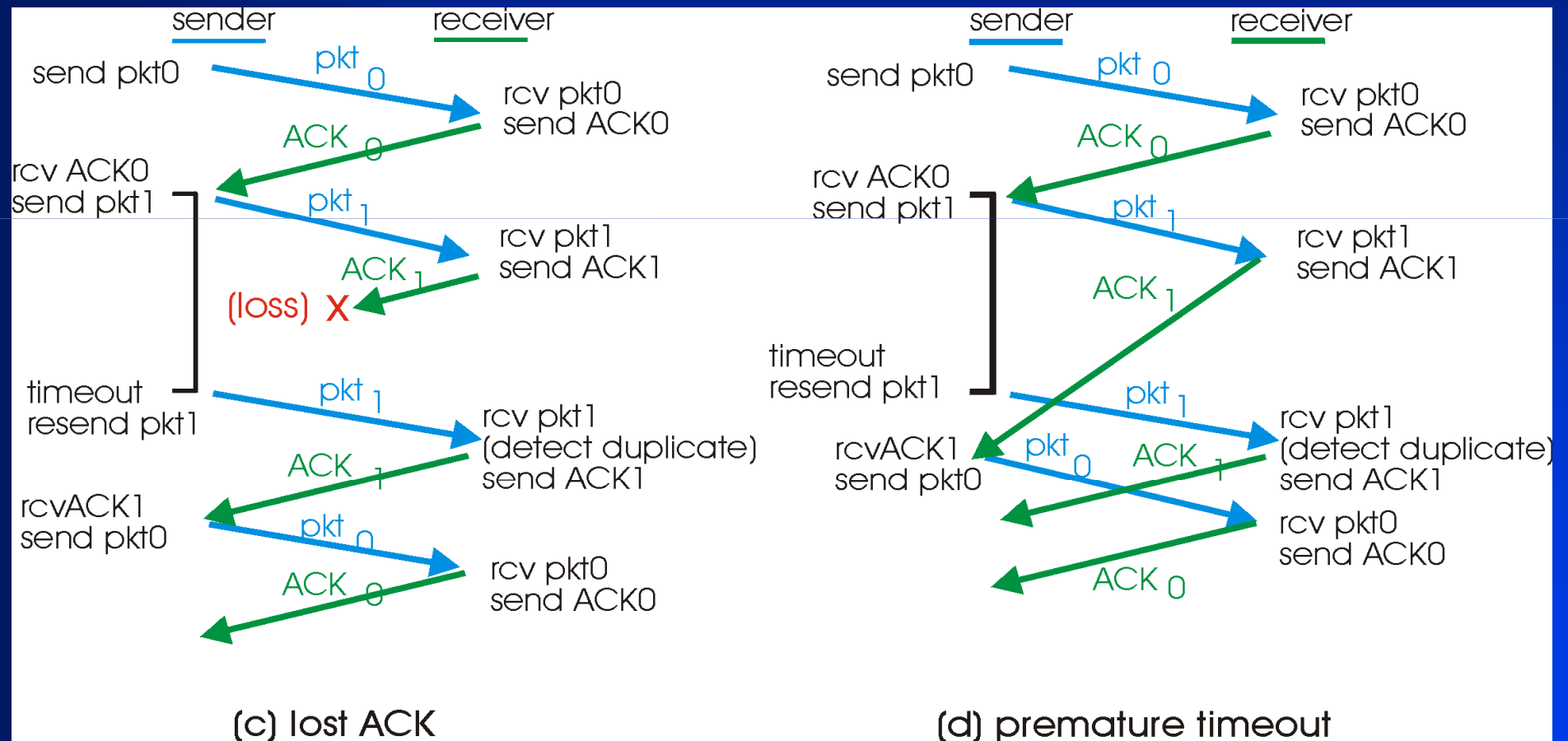
$V(T) = 0$

$V(R) = 0$

# Stop&Wait



# Stop&Wait



**Quando il flusso dati è bidirezionale  
è possibile includere nell'intestazione  
della PDU dati un campo con l'informazione  
di riscontro (ACK) per il flusso dati che sta  
fluendo in direzione opposta**

**La tecnica è detta “piggybacking”**

**Il protocollo Stop and wait  
puo`essere poco efficiente  
a causa di elevati ritardi  
di attesa delle conferme**

## *Prestazioni di Stop&Wait*

**Esempio: link da 1 Gbps, ritardo di propagazione end-to-end 15 ms, pacchetti da 1KB**

$$T_{\text{transmiss.}} = \frac{8 \text{ kbit/pkt}}{10^9 \text{ bit/s}} = 8 \mu\text{s}$$

$$\text{Utilizzazione} = U = \frac{\text{Frazione di tempo in cui il trasmettitore è occupato}}{30.016 \text{ ms}} = \frac{8 \mu\text{s}}{30.016 \text{ ms}} = 0.00015$$

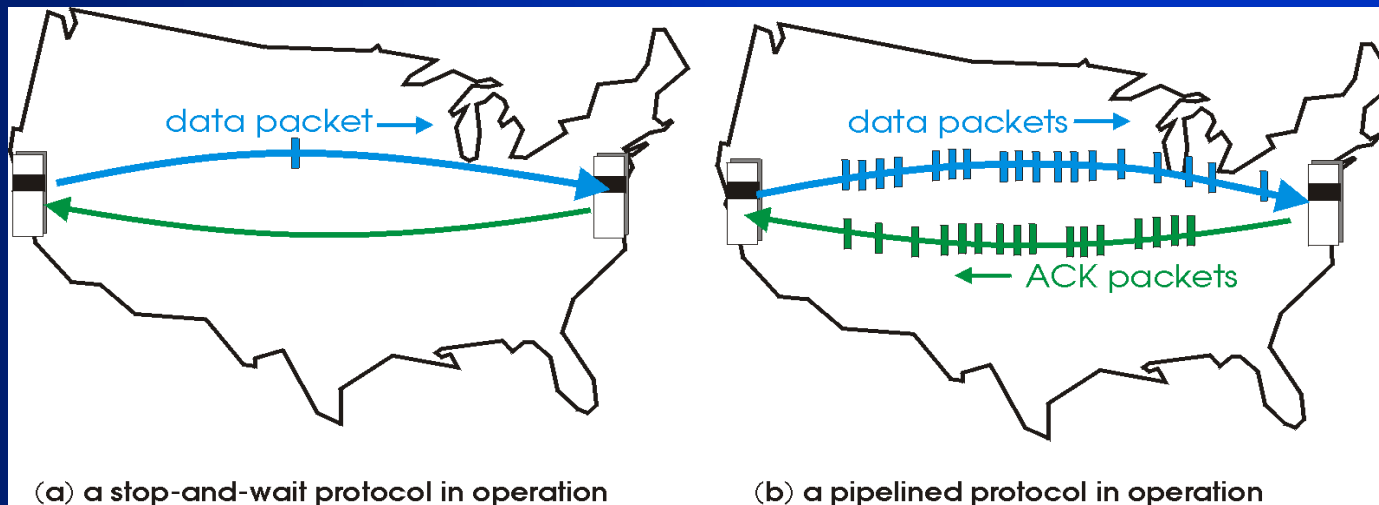
- un pacchetto da 1KB ogni 30 ms -> 33kB/sec di throughput su un link da 1 Gbps
- il protocollo limita l'uso delle risorse fisiche!



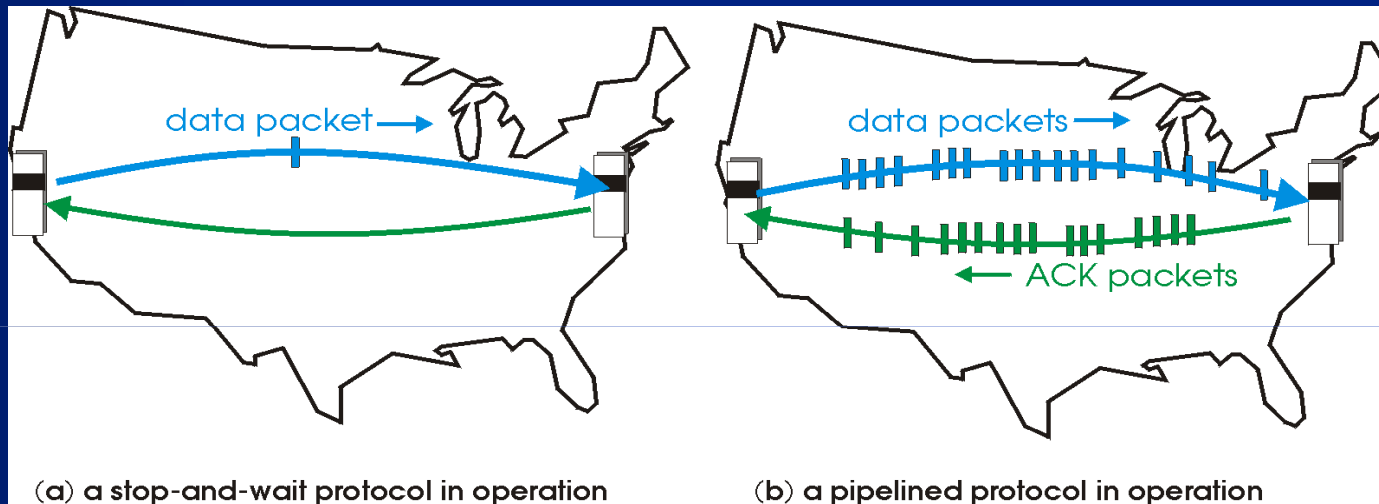
# Protocolli “pipeline”

**Pipelining:** al sender è consentito l'invio di pacchetti multipli senza che debba aspettare i riscontri

- i pacchetti in transito è come se riempissero un canale (pipeline)
- il range di numeri di sequenza aumenta
- necessità di buffering al sender e/o al receiver



# Protocolli “pipeline”



Due tipi di protocolli pipeline: *go-Back-N*, *selective repeat*