

# Il problema in generale...

- ◆ C'è quando il prodotto **bandwidth \* round-trip-delay** è grande: i protocolli visti funzionano male, perché stiamo ***sottoutilizzando*** il canale



# L'utilizzo della linea



- ◆ Se il canale ha capacità **C** (bit/s), la taglia del frame è **S** (bits), e il tempo di round-trip è **R**, ...
- ◆ ... possiamo calcolare **l'utilizzo della linea** nel caso di protocolli con ack:  
$$S / (S + C * R)$$
- ◆ Se  **$S < CR$** , abbiamo un'efficienza ***minore del 50%***



# Soluzione?



# Soluzione? Pensiamo...

- ◆ In analogia: stiamo gestendo trasporto merci su camion dall'Italia alla Spagna
- ◆ Mandiamo un camion di merci, e non ne mandiamo altri finchè non è tornato un camion con la ricevuta di ritorno (!)
- ◆ Non faremmo così, manderemmo più camion, preoccupandoci solo dopo un po' se il primo camion non è tornato
- ◆ Quello che si chiama anche ***pipelining***



# Sliding Windows



- ◆ La tecnica delle ***sliding windows*** sfrutta proprio quest'idea
- ◆ Invece di essere così apprensivi, ci rilassiamo un po', e ci preoccupiamo non quando c'è un solo frame dal destino incerto, ma un numero maggiore (***n***)
- ◆ Tipicamente, il numero maggiore lo prendiamo una potenza di due (2, 4, 8, ...) così non sprechiamo bits

# La sliding window



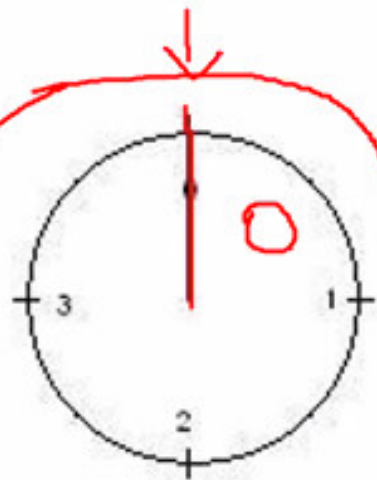
- ◆ Tiene conto di quanto siamo stressati: più la apriamo, più siamo rilassati e più pacchetti lasciamo andare senza bisogno di conferma
- ◆ La **taglia** della sliding window **può variare** sia per il sender che per il receiver, dando luogo a vari protocolli
- ◆ In altre parole, come nella vita, tra due che parlano uno può essere più o meno stressato dell'altro



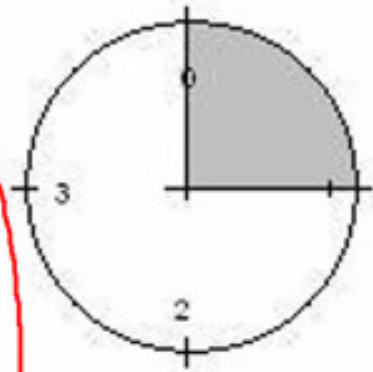
Vediamo...



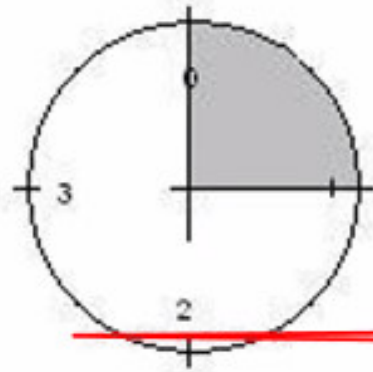
# Esempio ( $n=4$ , size 1)



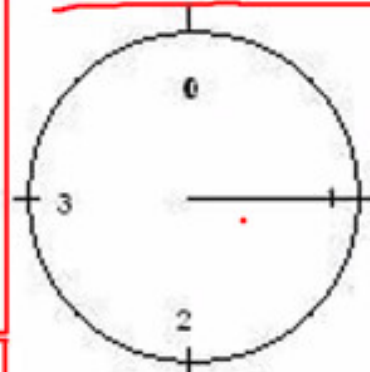
Initially



After 1st  
frame is  
sent

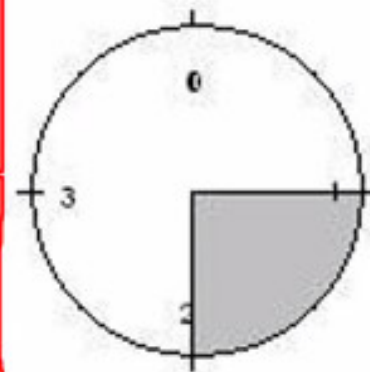
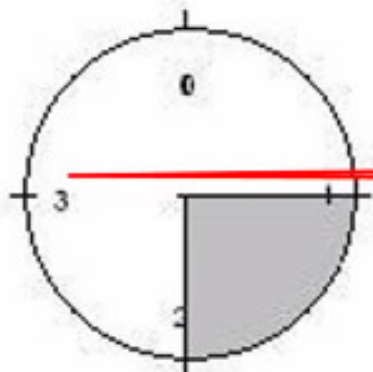
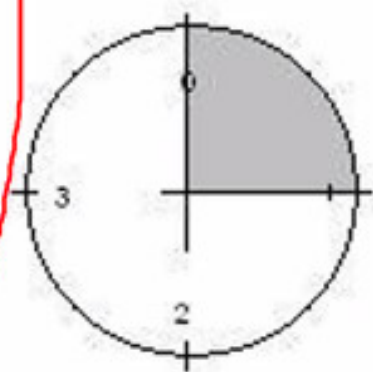
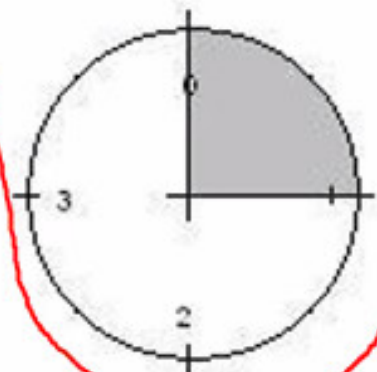


After first  
frame is  
received



After first  
acknowledgement  
is received

Sending Window



Receiving Window



# I protocolli "Go Back N"

- ◆ Si hanno quando *la taglia della sliding window di chi riceve è 1*
- ◆ Cioè, noi siamo rilassati (N), mentre il nostro interlocutore è super-apprensivo
- ◆ Funziona bene quando non ci sono molti errori ma il prodotto bandwidth \* round-trip-delay è alto



# Notare

- ◆ Il nostro "rischio" è aumentato, perché nel caso peggiore (Murphy docet...) abbiamo  $n$  camion fuori, e non sappiamo se torneranno...





# Il rischio si paga, quindi..

- ◆ ... dobbiamo avere altri n camion con lo stesso carico pronti, per fronteggiare il caso peggiore



# Quindi

- ◆ → dobbiamo avere un **buffer** di taglia  $n$  frames
- ◆ Assieme a  **$n$  timer** per l'eventuale ritrasmissione
- ◆ Ovviamente, se esauriamo il buffer, dobbiamo aspettare e non inviare più camion (frames)





I selective repeat



# I protocolli "selective repeat"



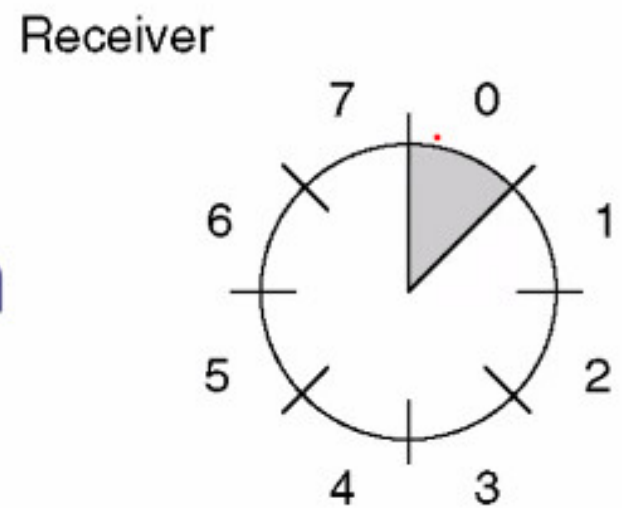
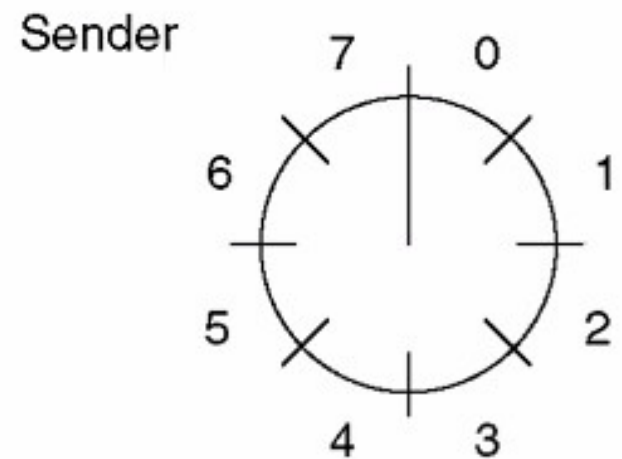
- ◆ Sono quelli in cui anche il nostro interlocutore, finalmente, si rilassa un po', e allarga la benedetta finestra...
- ◆ Funziona bene, ha il problema che ora anche il receiver deve stavolta allocare un buffer di taglia la sua apertura di finestra





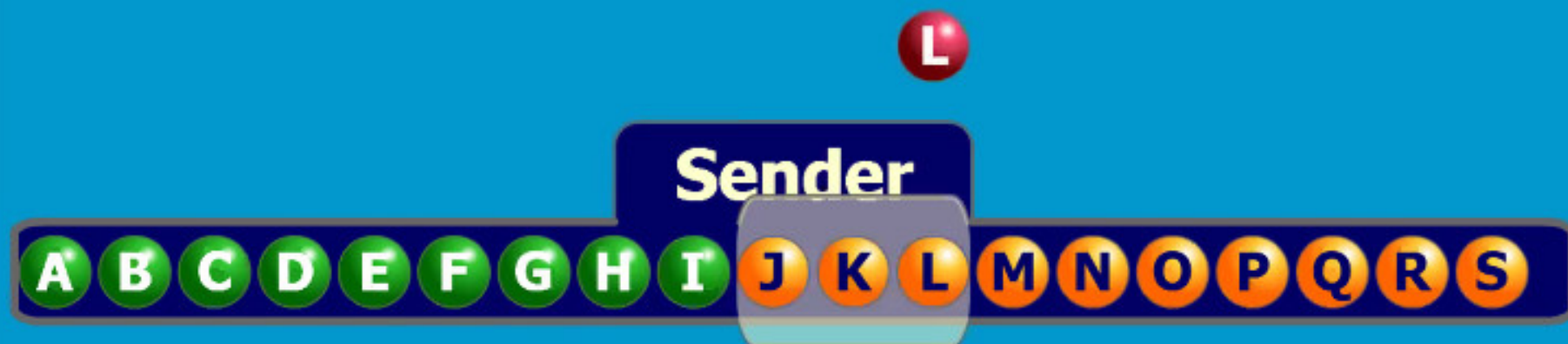
# Buffers

- ◆ Notare (l'abbiamo già detto ma meglio sia chiaro):
- ◆ La taglia richiesta del buffer è l'ampiezza ***dell'apertura massima*** della finestra, non la grandezza complessiva della finestra!



(a)

# TCP SLIDING WINDOW



window  
size  
3

retransmit  
timer  
34

RTT  
30

loss%  
30

speed  
8

step  
222

stop  
restart



# Problemi delle sliding windows

- ◆ Intuitivamente, a parte lo svantaggio in termini di allocazione di risorse, sembra che per la trasmissione convenga permettere una finestra quanto più aperta possibile
- ◆ Invece, se la taglia è troppo grande  
ovviamente si torna a fare confusione, (specie se non è garantita la ***sequenzialità*** del canale)

# Esempio



- ◆ Abbiamo una finestra di grandezza (ciclo) 4, e apertura massima 3
- ◆ All'inizio, il sender ha la finestra 0 1 2 3 ed il receiver è pronto a ricevere con **0 1 2 3** (i numeri in bold sono gli slot aperti della finestra corrente)
- ◆ Murphy è ispirato...

D:\MM\UNIPD\RETI-partial\lezioni-reti\lezione10\material\mrp3.flv



# Esempio (cont.)



- ◆ Il sender deve mandare un flusso di pacchetti **P0, P1, P2, P3, P4** etc...
- ◆ Invia i pacchetti **P0/0, P1/1, P2/2**
- ◆ Ora sia sender che receiver hanno come finestra **0 1 2 3**
- ◆ Murphy sta prendendo un caffè, quindi i pacchetti arrivano



# Esempio (cont.)

- ◆ Il receiver manda tutti gli ACK corrispondenti, e quindi avanza la sua finestra, che passa da



0 1 2 3

- ◆ Il flusso di dati finora è **P0, P1, P2...**
- ◆ Murhpy ha finito il caffè (era un espresso), e rovina l'ack di P0/0
- ◆ Al sender arrivano gli ack di P1/1, P2/2





# Esempio (cont.)



- ◆ Il tempo passa, e il sender si insospettisce visto che non riceve l'ack di P0/0, pensando che Murphy c'entri qualcosa...
- ◆ → dopo un po' (timeout), rimanda il pacchetto P0, sempre come P0/0
- ◆ Murphy nel frattempo si è seduto a godersi lo spettacolo e non fa nulla...

# Esempio (cont.)



- ◆ Il receiver aveva come finestra **0 1 2 3**, quindi accetta il pacchetto P0/0 e manda l'ack corrispondente
- ◆ Il sender riceve anche l'ack di P0/0, e quindi tutto contento sposta anche lui la sua finestra a **0 1 2 3**, e manda i pacchetti P3/3, P4/0, P5/1



## Esempio (cont.)



- ◆ → il flusso di dati finora è stato **P0, P1, P2, P3, P0 (!!!)**
- ◆ E non è finita...

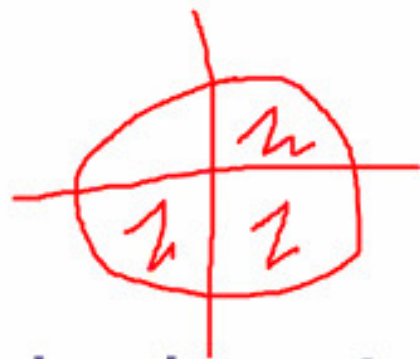
# Esempio (cont.)



- ◆ Il receiver aveva come finestra **0 1 2 3**
- ◆ Murphy distoglie il pacchetto P4/0 e lo rallenta
- ◆ → il receiver riceve prima i pacchetti P3/3 e P5/1
- ◆ Che accetta, e quindi avanza la finestra a **0 1 2 3 ...**
- ◆ → il flusso di dati è stato **P0, P1, P2, P3, P0, P5 (!!!)**



# Morale



◆ Il punto è che le due situazioni

**0 1 2 3**

e

**0 1 2 3**



hanno uno 0 che si sovrappone, quindi da un passaggio consecutivo all'altro, la posizione 0 resta ambigua, perché il receiver non sa più distinguere un pacchetto da un altro in base al numero dello slot nella finestra!

# Morale 2



- ◆ Conviene avere una apertura che sia al massimo la metà della grandezza della sliding window

