Reti di Calcolatori I

Prof. Roberto Canonico
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Corso di Laurea in Ingegneria delle Telecomunicazioni
Corso di Laurea in Ingegneria dell'Automazione

A.A. 2017-2018

Il livello trasporto:

controllo di flusso in TCP

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

Università degli Studi di Napoli Federico II

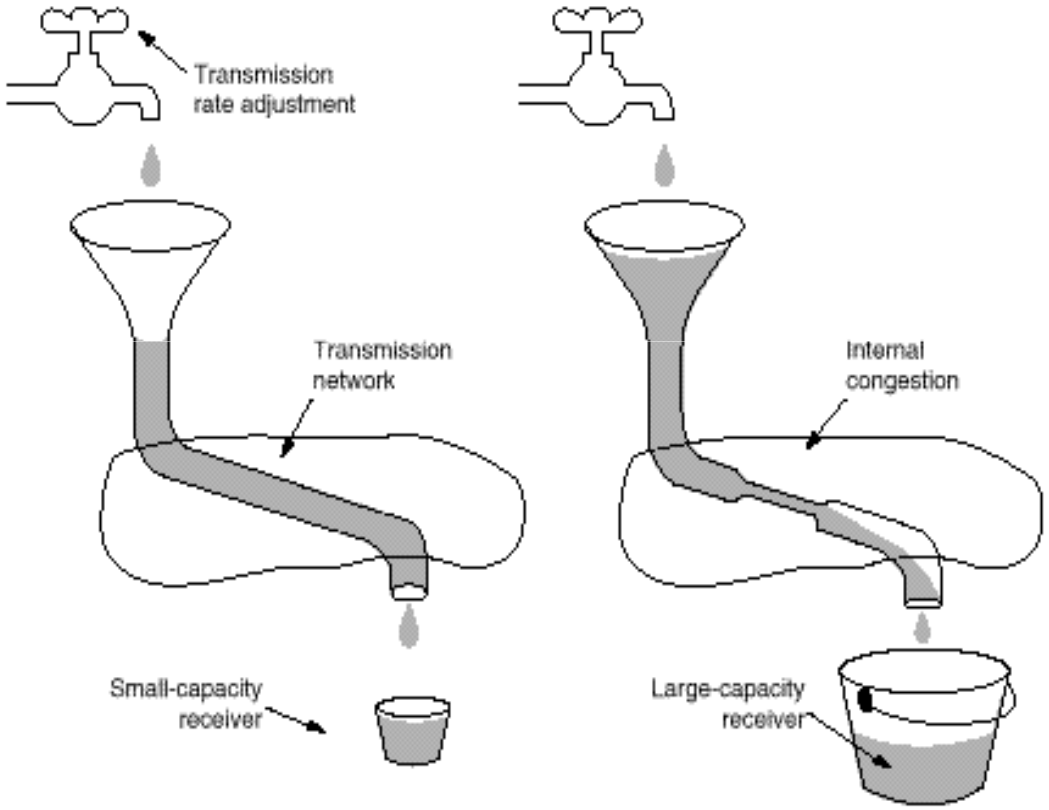
Nota di copyright per le slide COMICS

Nota di Copyright

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

Università degli Studi di Napoli
Federico II

TCP: Controllo di Flusso e di Congestione



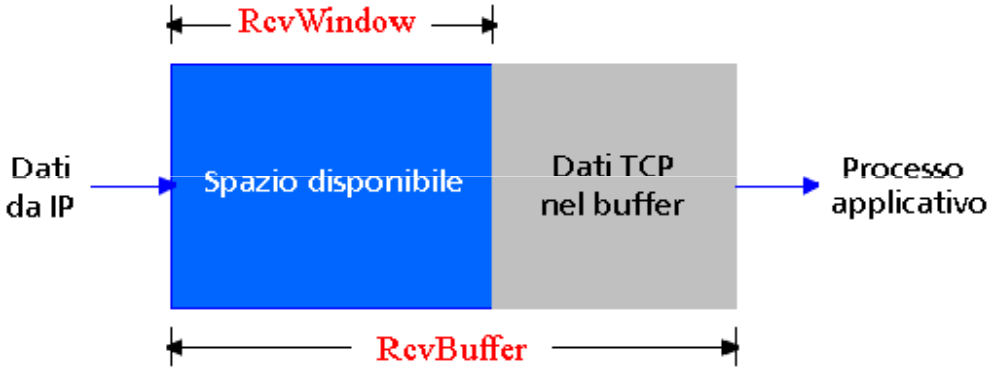
Come gestire entrambi i tipi di controllo?

- *Receiver window*: dipende dalla dimensione del buffer di ricezione
- *Congestion window*: basata su una stima della capacità della rete

I byte trasmessi corrispondono alla dimensione della finestra più piccola

TCP: controllo di flusso

- Il lato ricevente della connessione TCP ha un buffer di ricezione:



- Il processo applicativo potrebbe essere rallentato dalla lettura nel buffer

Controllo di flusso

- Servizio di corrispondenza delle velocità: la frequenza d'invio deve corrispondere alla frequenza di lettura dell'applicazione ricevente

(supponiamo che il destinatario TCP scarti i segmenti fuori sequenza)

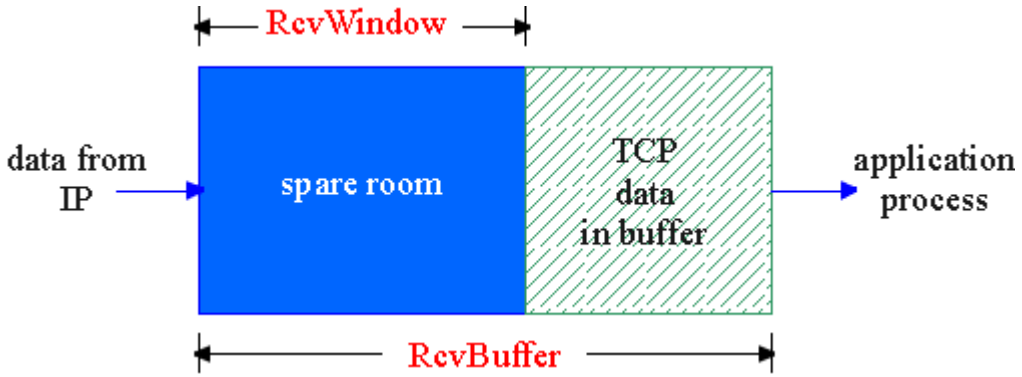
TCP Flow Control

flow control

Il mittente non dovrà sovraccaricare il ricevente inviando dati ad una velocità troppo elevata

RcvBuffer = size of TCP Receive Buffer

RcvWindow = amount of spare room in Buffer



receiver buffering

ricevente: comunica dinamicamente al mittente la dimensione corrente del buffer

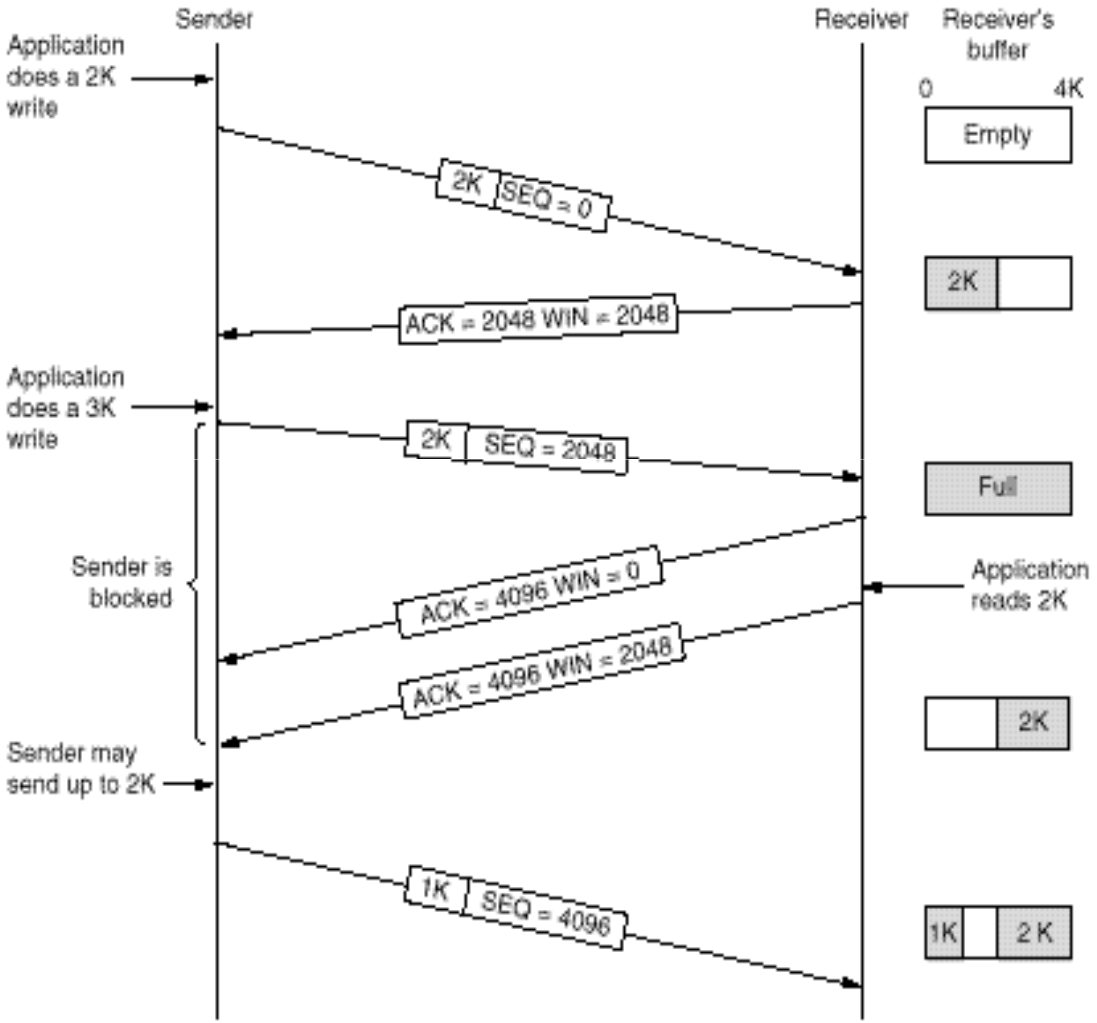
- campo **RcvWindow** nel segmento TCP:

RcvWindow = RcvBuffer – [LastByteRcvd - LastByteRead]

mittente: conserva i dati già trasmessi ma non riscontrati e limita tale quantità all'ultima **RcvWindow** ricevuta:

LastByteSent – LastByteAcked ≤ RcvWindow

TCP: Transmission Policy



Silly Window Syndrome

- Silly Window Syndrome (ricevitore): il ricevitore svuota lentamente il buffer di ricezione e invia segmenti di ack con dimensione della finestra molto piccola, quindi il trasmettitore invia segmenti corti con molto overhead. Soluzione con l'algoritmo di Clark: il ricevitore indica una finestra nulla finchè il buffer di ricezione non si è svuotato per metà o per una porzione uguale a MSS.

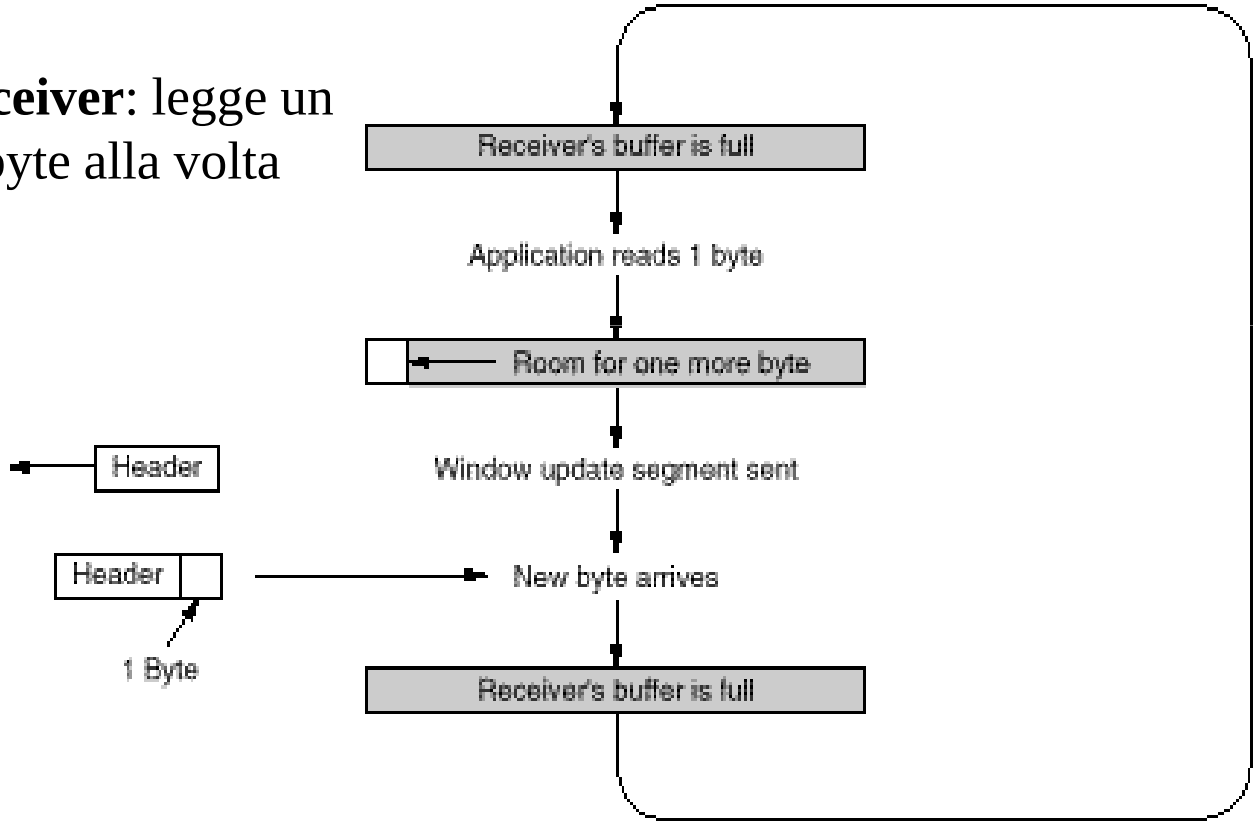
- Silly Window Syndrome (trasmettitore): l'applicazione genera dati lentamente, invia segmenti molto piccoli così come vengono prodotti. Soluzione algoritmo di Nagle: il TCP sorgente invia la prima porzione di dati anche se corta e gli altri vengono inviati solo se o il buffer di uscita contiene almeno MSS byte, oppure se si riceve un ack per il segmento precedente.

- In un caso vengono inviati segmenti corti con molto overhead perché la finestra al ricevitore è grande un byte, nell'altro caso perché è l'applicazione che genera un byte alla volta molto lentamente.

La sindrome della Silly Window al ricevitore

Sender: invia blocchi grandi

Receiver: legge un byte alla volta



Soluzione di Clark: Impedisce al receiver di aggiornare la finestra un byte alla volta

Il ricevitore indica una finestra nulla finchè il buffer di ricezione non si è svuotato per metà o per una porzione uguale a MSS.

L'algoritmo di Nagle

- Per applicazioni che inviano dati un byte alla volta (es:Telnet):
  - invia il primo byte e bufferizza il resto finché non giunge ACK
  - in seguito:
    - invia, in un unico segmento, tutti i caratteri bufferizzati
    - ricomincia a bufferizzare finché non giunge l'ACK per ognuno di essi
- Elimina la sindrome della Silly Window al trasmettitore

```
if there is new data to send
        if window size >= MSS and available data is >= MSS
                send complete MSS segment now
        else    if there is unconfirmed data still in the pipe
                        enqueue data in the buffer until an ack is received
                else
                        send data immediately
                end if
        end if
end if
```