Reti di Calcolatori:
Internet, Intranet e Mobile Computing
a.a. 2007/2008

http://www.di.uniba.it/~lisi/courses/reti/reti0708.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

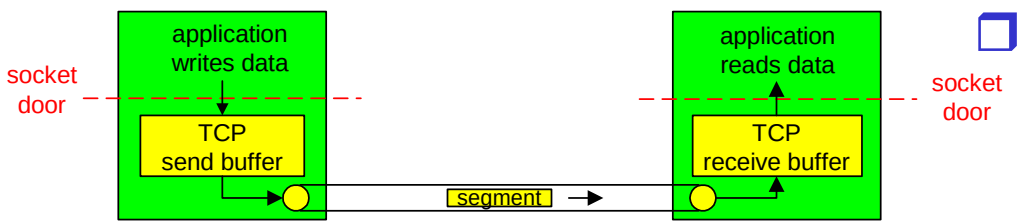
dib

Sommario della lezione di oggi: Lo strato di trasporto (2/3)

- Servizi e protocolli dello strato di trasporto
- Multiplazione e demultiplazione delle applicazioni
- Trasporto senza connessione: UDP
- Trasporto con connessione: TCP
- Il controllo della congestione nel TCP

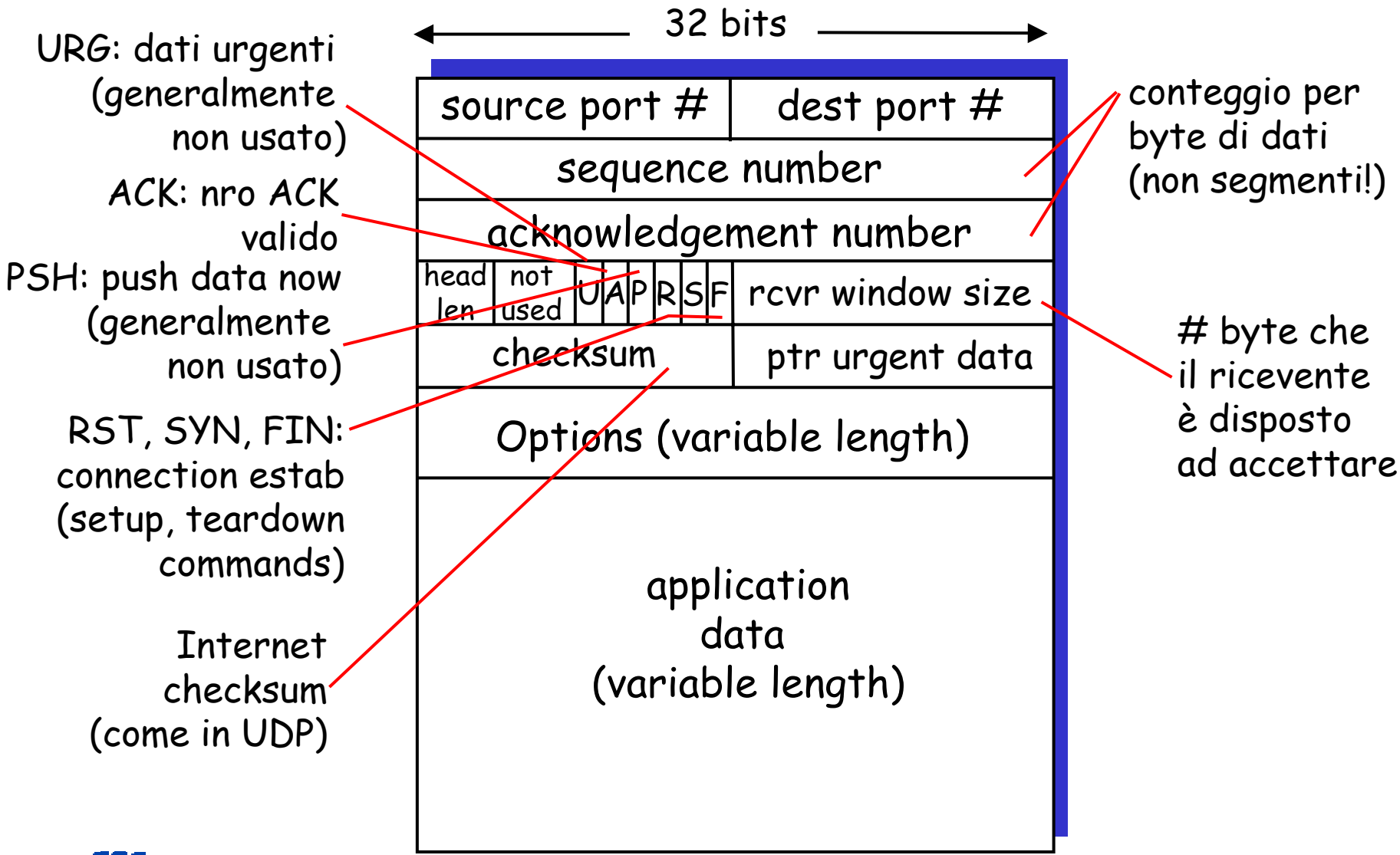
Trasporto con connessione: TCP [RFC 793, 1122, 1323, 2018, 2581]

- punto-a-punto:
  - un mittente, un ricevente
- *byte stream* affidabile ed in ordine:
  - no "message boundaries"
- buffer di invio e ricezione



- full duplex:
  - flusso dati bi-direzionale nella stessa connessione
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (scambio di msg di controllo) inizializza stato di mittente/ricevente prima di scambio dati
- flow-controlled:
  - mittente non inonderà ricevente

TCP: struttura del segmento



TCP: numeri di sequenza e riscontri

Numeri di sequenza:

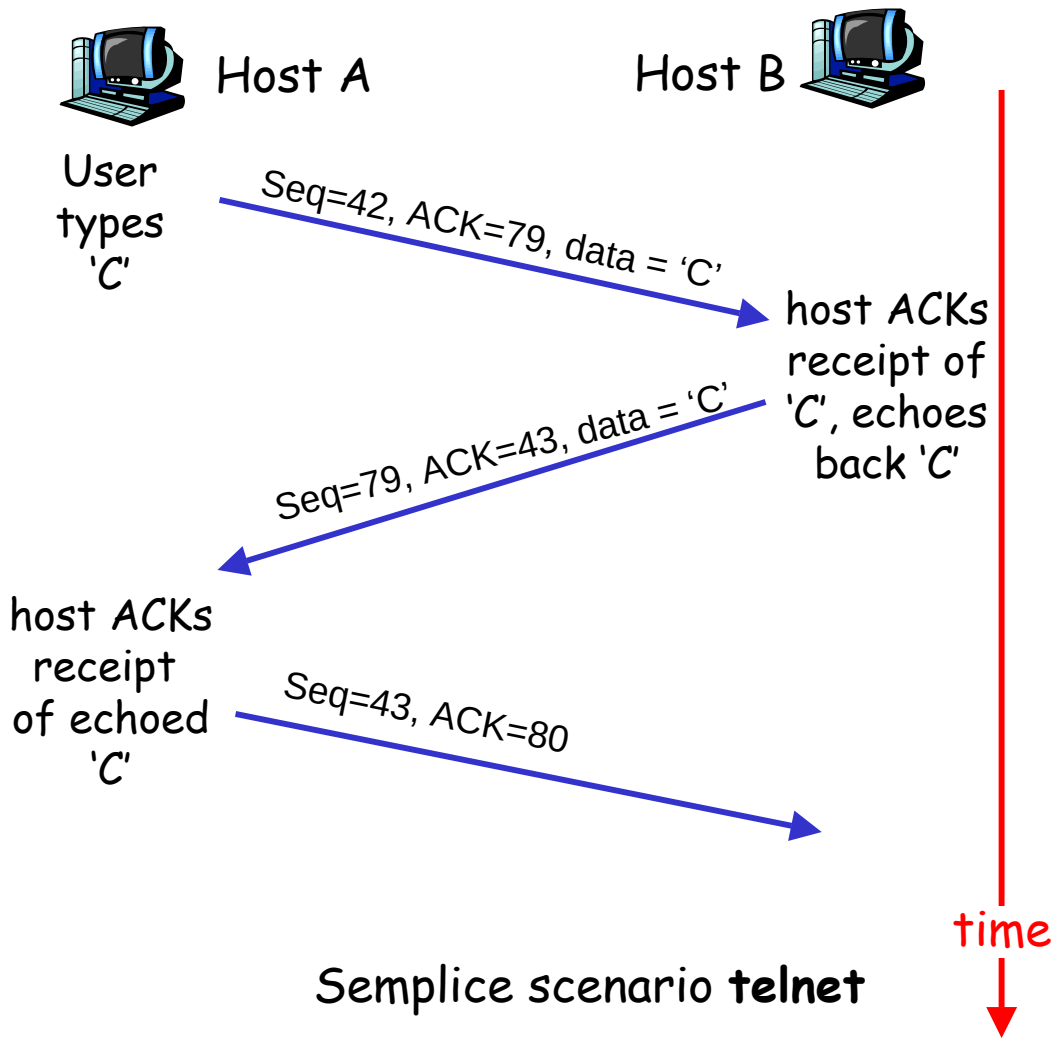
- numero all'interno del flusso del primo byte nel segmento

Numeri di riscontro:

- num. del prossimo byte atteso dall'altro lato
- ACK cumulativi

Q: come gestire segmenti in disordine?

- Soluzione spetta all'implementatore del TCP



Semplice scenario **telnet**

Trasferimento dati affidabile nel TCP: eventi del mittente

Dati ricevuti dall'applicazione:

- Crea un segmento con il numero di sequenza
  - Il numero di sequenza è il numero del primo byte del segmento nel flusso di byte
- Avvia il timer, se non è già in funzione (pensate al timer come se fosse associato al più vecchio segmento non riscontrato)
  - Intervallo di scadenza: `TimeOutInterval`

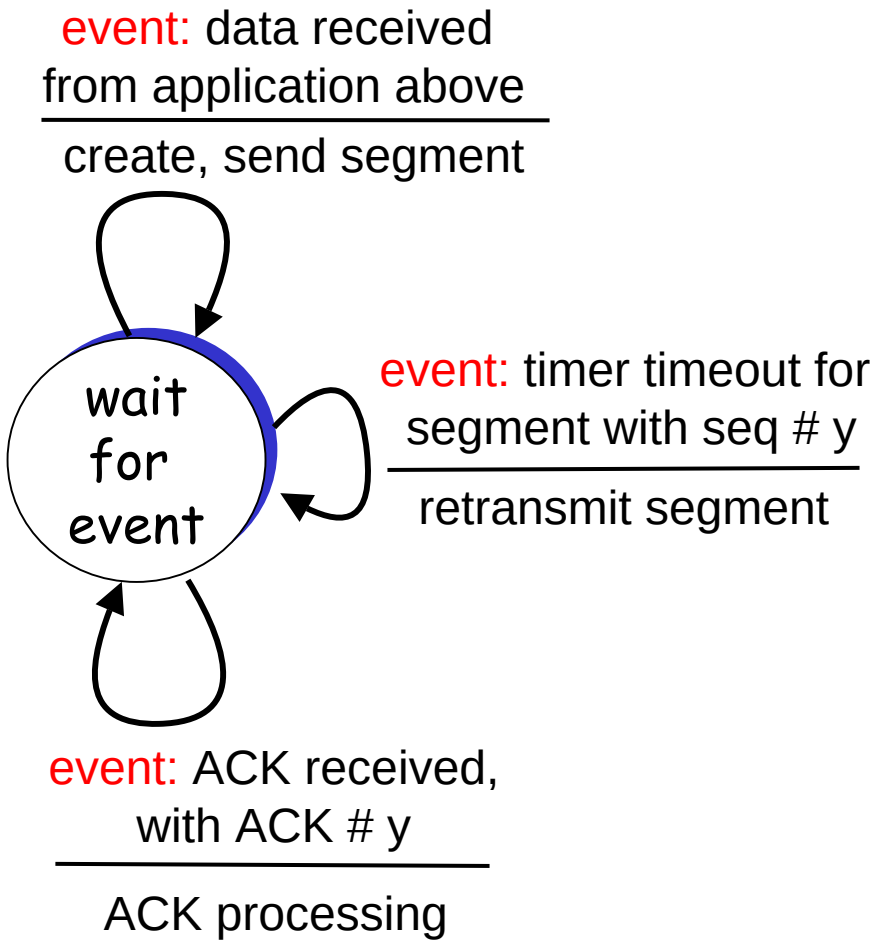
Timeout:

- Ritrasmette il segmento che ha causato il timeout
- Riavvia il timer

Ricezione di ACK:

- Se riscontra segmenti precedentemente non riscontrati
  - aggiorna ciò che è stato completamente riscontrato
  - avvia il timer se ci sono altri segmenti da completare

Trasferimento dati affidabile nel TCP: una modellazione FSA del mittente



Trasferimento dati affidabile nel TCP: implementazione semplificata del mittente

Assunzioni:
- Trasferimento dati ad una via
- nessun controllo di flusso e di congestione

Commento:
- SendBase-1: ultimo byte cumulativamente riscontrato

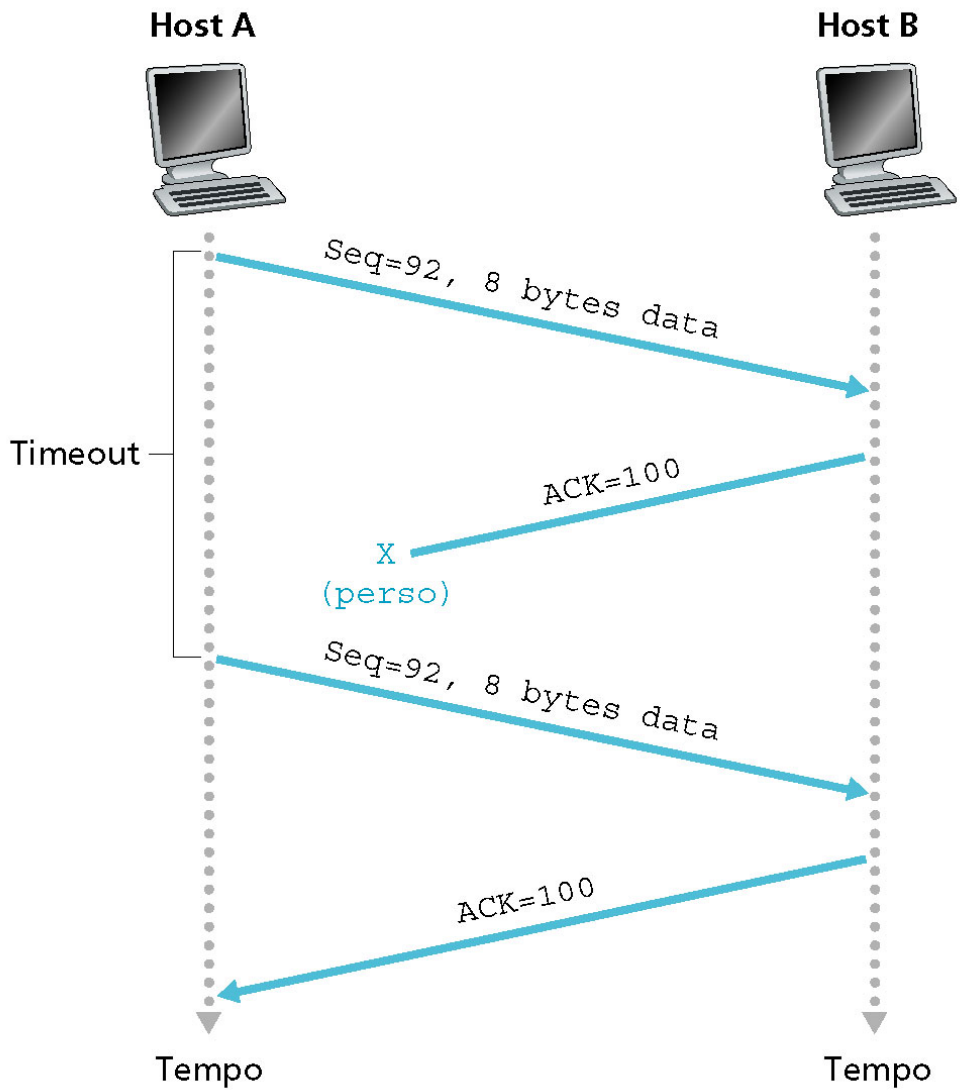
Esempio:
- SendBase-1 = 71;

y = 73, quindi il destinatario vuole 73+ ;
y > SendBase, allora vengono riscontrati tali nuovi dati

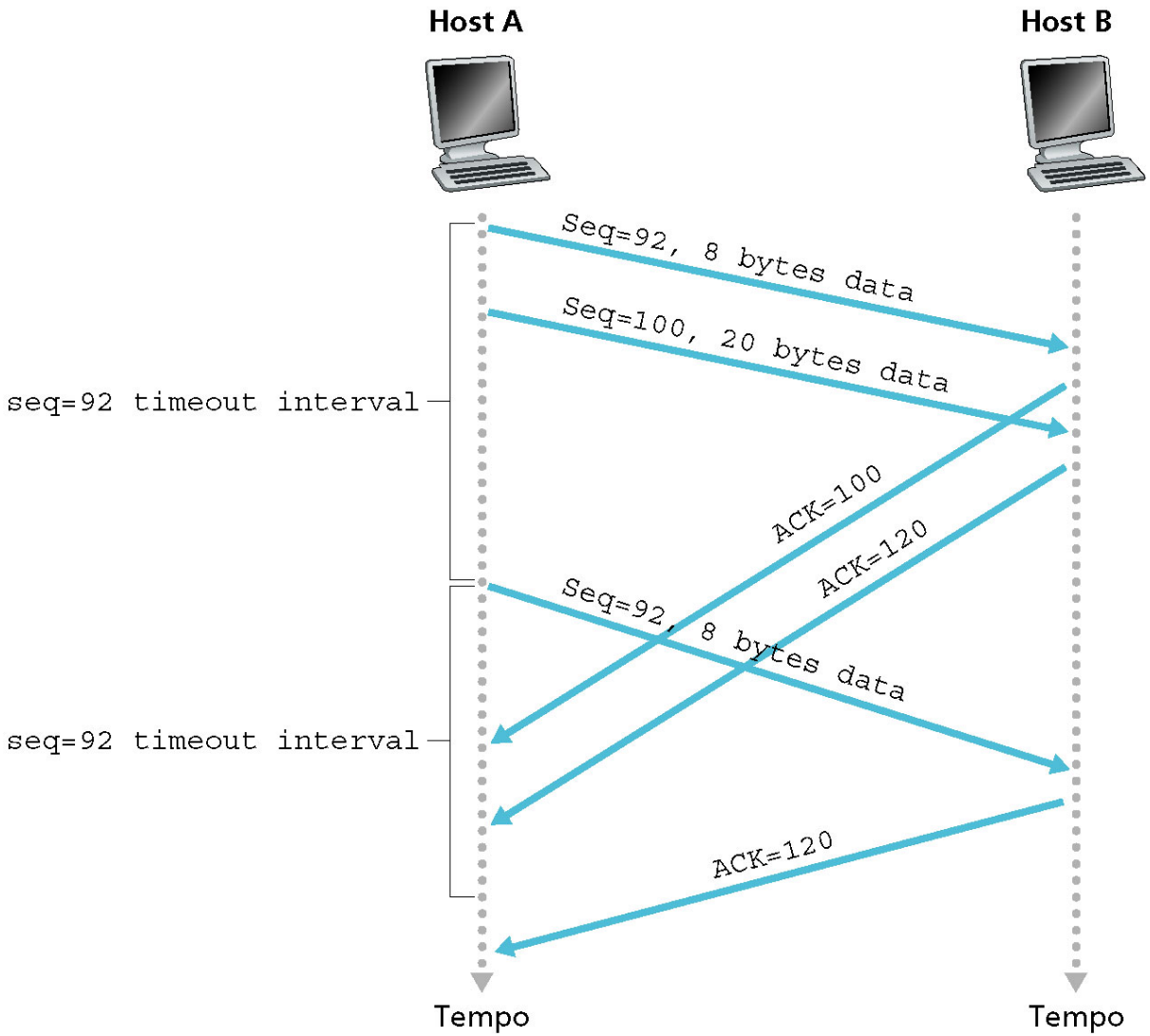
/* Si è assunto che il sender non sia limitato dal controllo di flusso o di congestione del TCP, che i dati da sopra siano di dimensioni inferiori all'MSS e che il trasferimento dei dati avvenga in una sola direzione.*/

```
NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
   switch(event)

      event: data received from application above
         create TCP segment with sequence number NextSeqNum
         if (timer currently not running)
            start timer
         pass segment to IP
         NextSeqNum=NextSeqNum+length(data)
         break;

      event: timer timeout
         retransmit not-yet-acknowledged segment with
            smallest sequence number
         start timer
         break;

      event: ACK received, with ACK field value of y
         if (y > SendBase) {
            SendBase=y
            if (there are currently any not-yet-acknowledged
               segments)
               start timer
            }
            break;

   } /* end of loop forever */
```

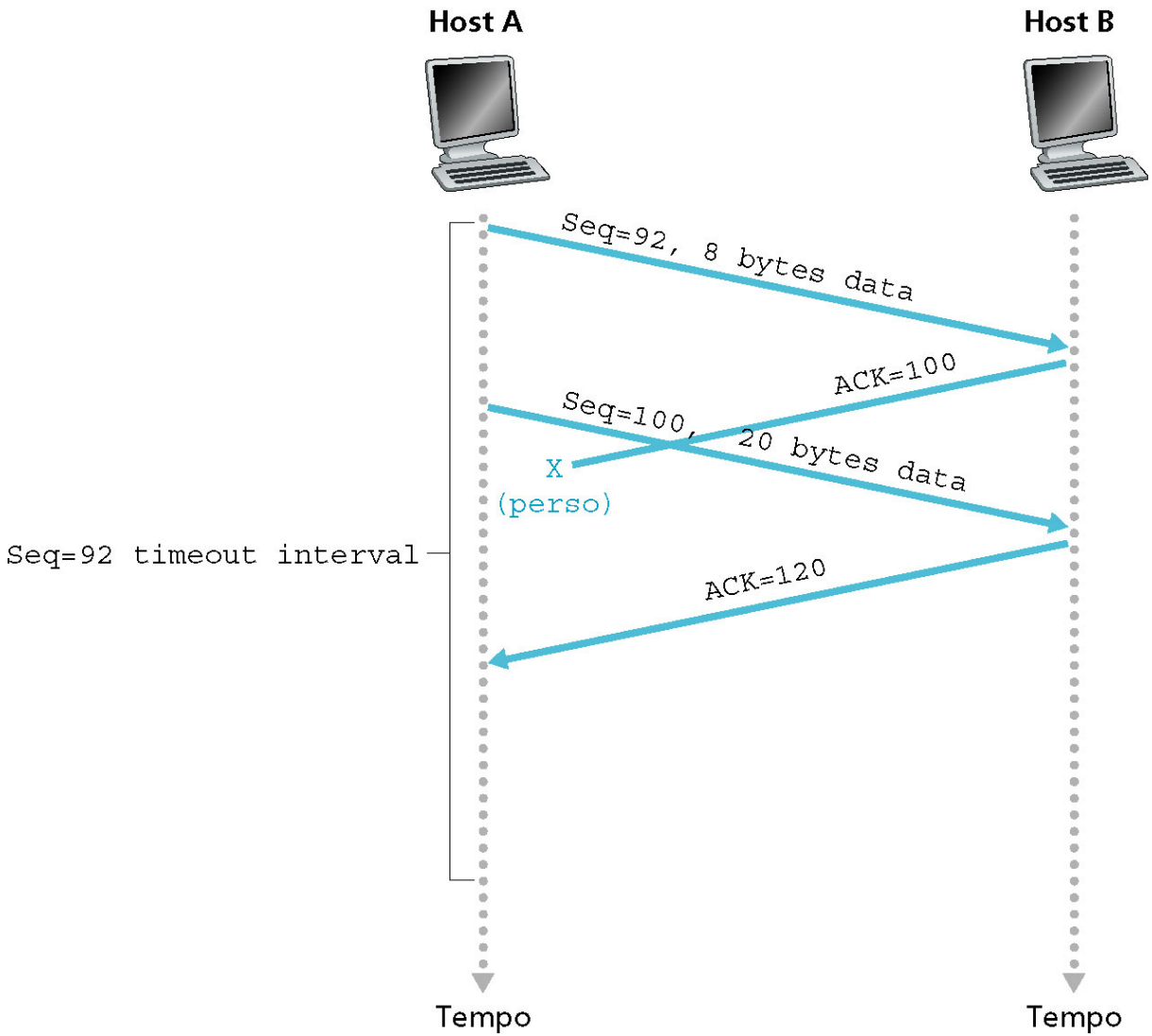
Trasferimento dati affidabile nel TCP: scenario n.1



Trasferimento dati affidabile nel TCP: scenario n.2



Trasferimento dati affidabile nel TCP: scenario n.3



Trasferimento dati affidabile nel TCP: stima del timeout

D: come impostare il valore di timeout?

- + lungo di RTT
  - N.B. RTT varierà
- se troppo breve, timeout prematuro
  - ritrasmissioni non necessarie
- se troppo lungo, reazione lenta alla perdita del segmento

D: come stimare RTT?

- **SampleRTT**: tempo misurato dalla trasmissione del segmento fino alla ricezione di ACK
  - ignora ritrasmissioni, segmenti cumulativamente riconosciuti
- **SampleRTT** varierà, vuole il RTT stimato "+ dolce"
  - usa diverse misure recenti, non solo attuale **SampleRTT**

Trasferimento dati affidabile nel TCP: stima del timeout (cont.)

EstimatedRTT = (1-x)*EstimatedRTT + x*SampleRTT

- Media esponenziale mobile pesata
- influenza di un certo campione decresce esponenzialmente
- valore tipico di x: 0.1

Impostare il timeout

- **EstimatedRTT** + "margine di sicurezza"
- variazione larga in **EstimatedRTT ->** + largo margine di sicurezza

Timeout = EstimatedRTT + 4*Deviation

Deviation = (1-x)*Deviation +
x*|SampleRTT-EstimatedRTT|

Trasferimento dati affidabile nel TCP: generazione di ACK [RFC 1122, 2581]

Evento nel destinatario	Azione del ricevente TCP
Arrivo ordinato di un segmento con numero di sequenza atteso. Tutti i dati fino al numero di sequenza atteso sono già stati riscontrati.	ACK ritardato. Attende fino a 500 ms l'arrivo del prossimo segmento. Se il segmento non arriva, invia un ACK.
Arrivo ordinato di un segmento con numero di sequenza atteso. Un altro segmento è in attesa di trasmissione dell'ACK.	Invia immediatamente un singolo ACK cumulativo, riscontrando entrambi i segmenti ordinati.
Arrivo non ordinato di un segmento con numero di sequenza superiore a quello atteso. Viene rilevato un buco.	Invia immediatamente un ACK duplicato, indicando il numero di sequenza del prossimo byte atteso.
Arrivo di un segmento che colma parzialmente o completamente il buco.	Invia immediatamente un ACK, ammesso che il segmento cominci all'estremità inferiore del buco.

Trasferimento dati affidabile nel TCP: ritrasmissione rapida

- Il periodo di timeout spesso è relativamente lungo:
  - lungo ritardo prima di ritrasmettere il pacchetto perduto.
- Rileva i segmenti perduti tramite gli ACK duplicati.
  - Il mittente spesso invia molti segmenti.
  - Se un segmento viene smarrito, è probabile che ci saranno molti ACK duplicati.
- Se il mittente riceve 3 ACK per lo stesso dato, suppone che il segmento che segue il dato riscontrato è andato perduto:
  - ritrasmissione rapida: rispedisce il segmento prima che scada il timer.

Trasferimento dati affidabile nel TCP: ritrasmissione rapida (cont.)

```
evento: ACK ricevuto, con valore del campo ACK pari a y
            if (y > SendBase) {
                SendBase = y
                if (esistono attualmente segmenti non ancora riscontrati)
                    avvia il timer
              }
            else {
                incrementa il numero di ACK duplicati ricevuti per y
                if (numero di ACK duplicati ricevuti per y = 3) {
                    rispedisci il segmento con numero di sequenza y
                }
```

un ACK duplicato per un segmento già riscontrato

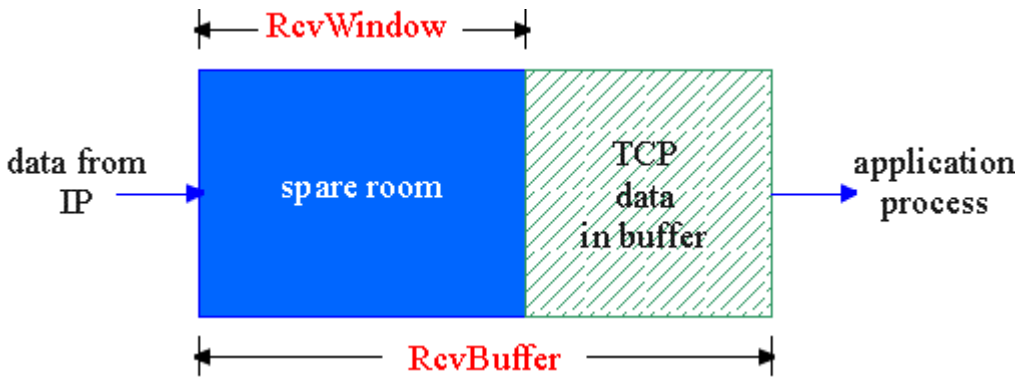
ritrasmissione rapida

Controllo di flusso nel TCP

Controllo di flusso

Il mittente non prevaricherà i buffer del ricevente trasmettendo troppo e troppo veloce

`RcvBuffer` = grandezza del buffer di ricezione
`RcvWindow` = grandezza della finestra di ricezione (spazio libero nel buffer di ricezione)



receiver buffering

ricevente: informa esplicitamente il mittente di (dinamicamente mutevole) ammontare di spazio libero di buffer

- **campo `RcvWindow`** nel segmento TCP

mittente: tiene l'ammontare dei dati trasmessi e non ancora riscontrati inferiore del **`RcvWindow`** più recentemente ricevuto

Gestione della connessione TCP: apertura della connessione

Ricorda: sender e receiver TCP stabiliscono una "connessione" prima di scambiarsi i segmenti

- inizializzano le variabili TCP:
  - numeri di sequenza
  - buffer, info su controllo di flusso (e.g. **RcvWindow**)
- *client:* iniziatore della connessione

```
Socket clientSocket = new
Socket("hostname","port
number");
```

- *server:* contattato dal client

```
Socket connectionSocket =
welcomeSocket.accept();
```

Handshake a tre vie:

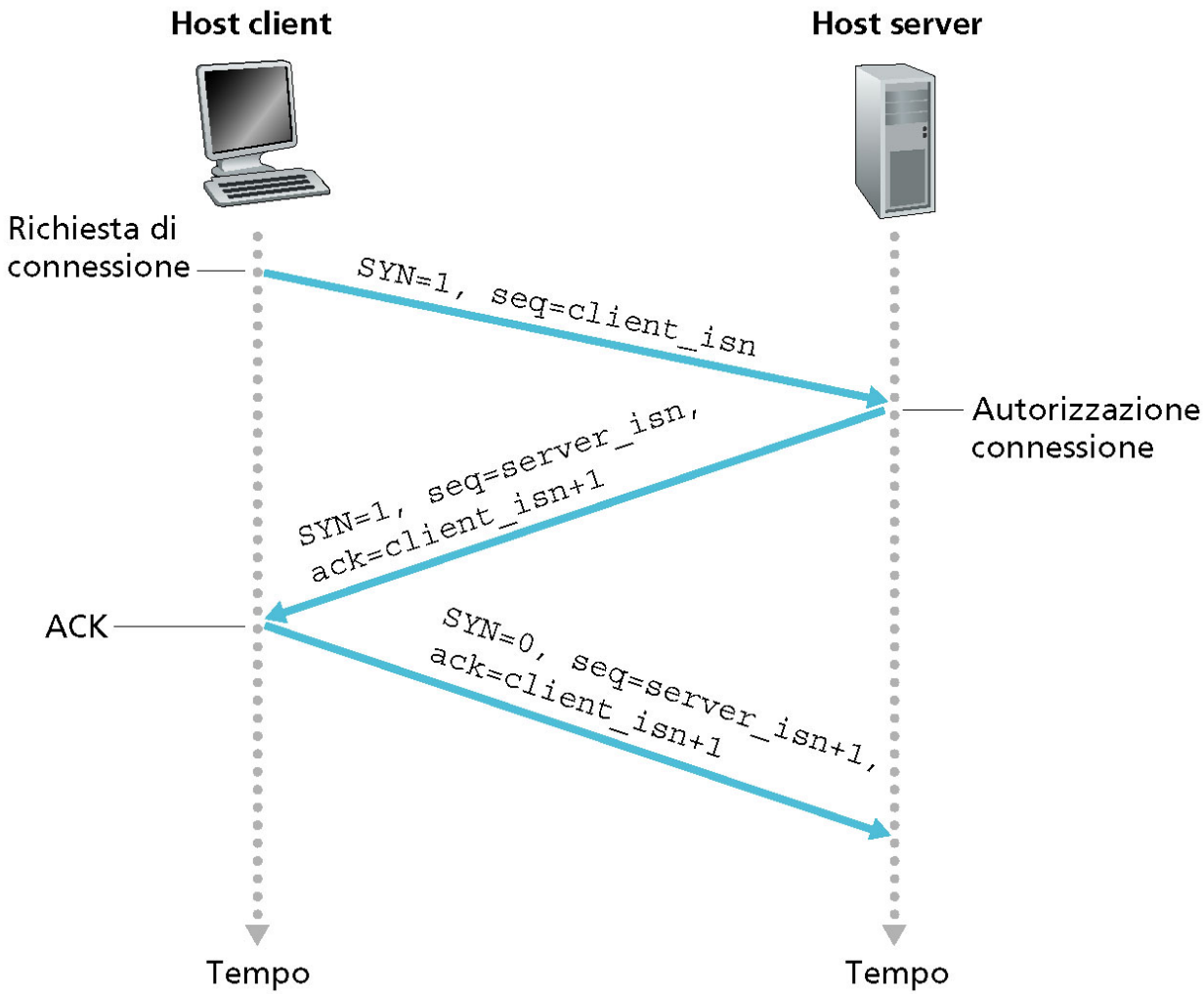
Step 1: il client invia SYN al server, specificando il numero iniziale di sequenza

Step 2: il server riceve SYN, replica con SYNACK

- riconosce SYN ricevuto
- alloca buffer
- specifica il proprio numero iniziale di sequenza

Step 3: riscontro finale

Gestione della connessione TCP: apertura della connessione (cont.)



Gestione della connessione TCP: chiusura della connessione

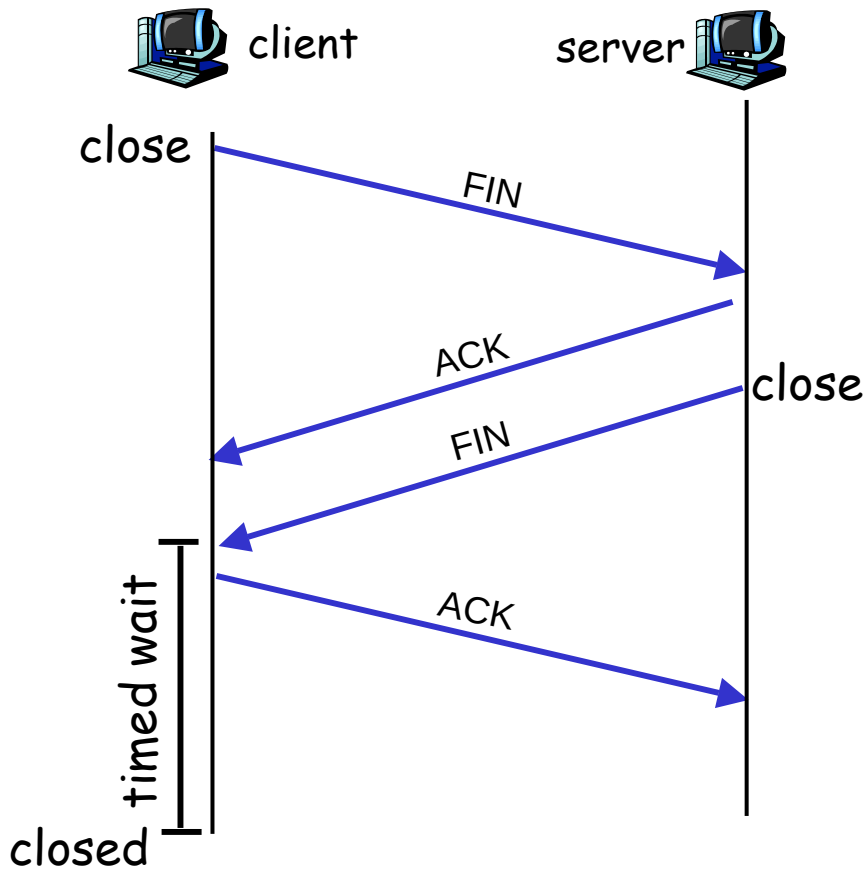
Step 1: il client chiude il proprio socket, invia FIN al server

Step 2: il server riceve FIN, replica con ACK, chiude il proprio socket, invia FIN.

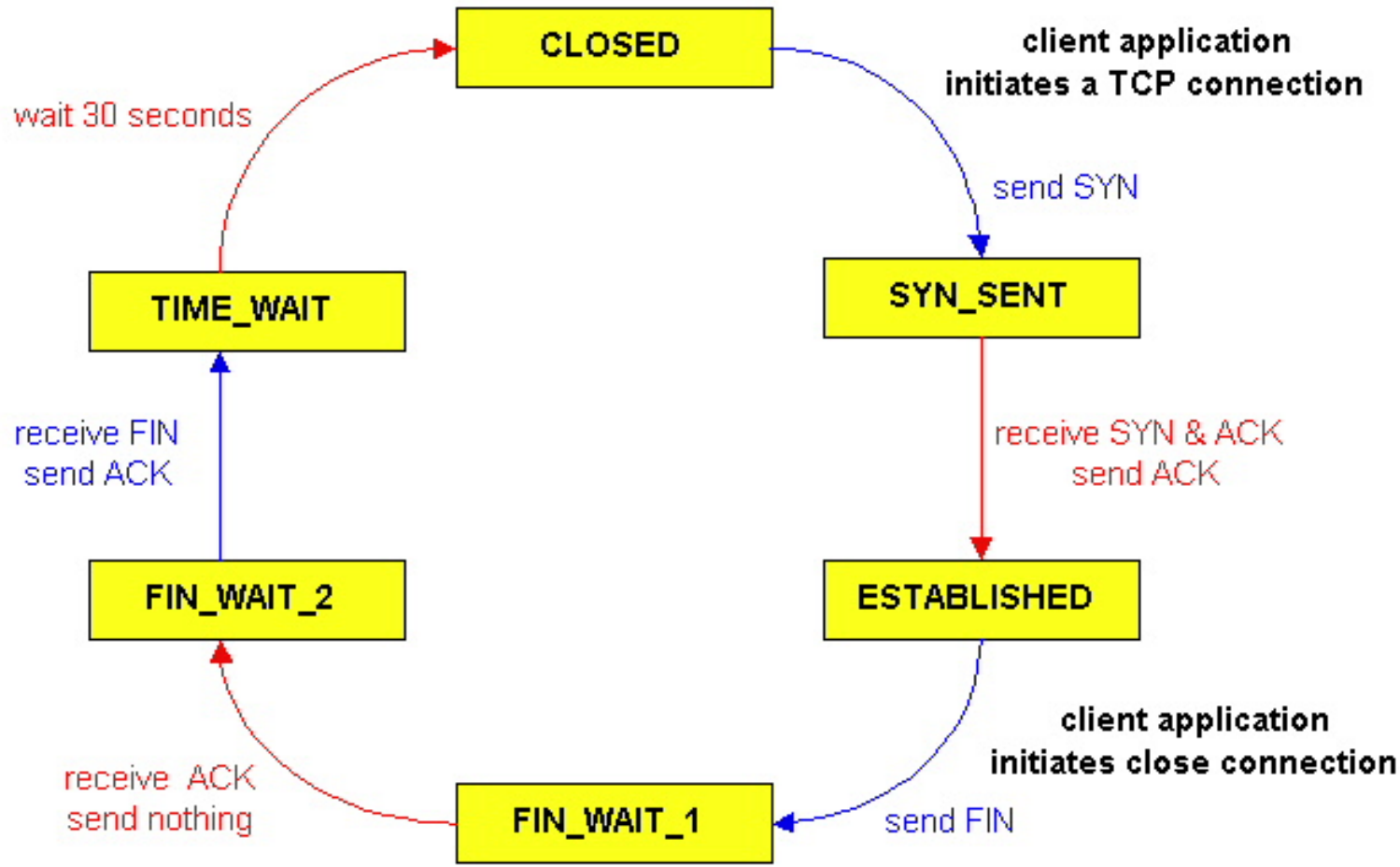
Step 3: il client riceve FIN, replica con ACK, e si pone nel "timed wait"

Step 4: il server riceve ACK. Connection closed.

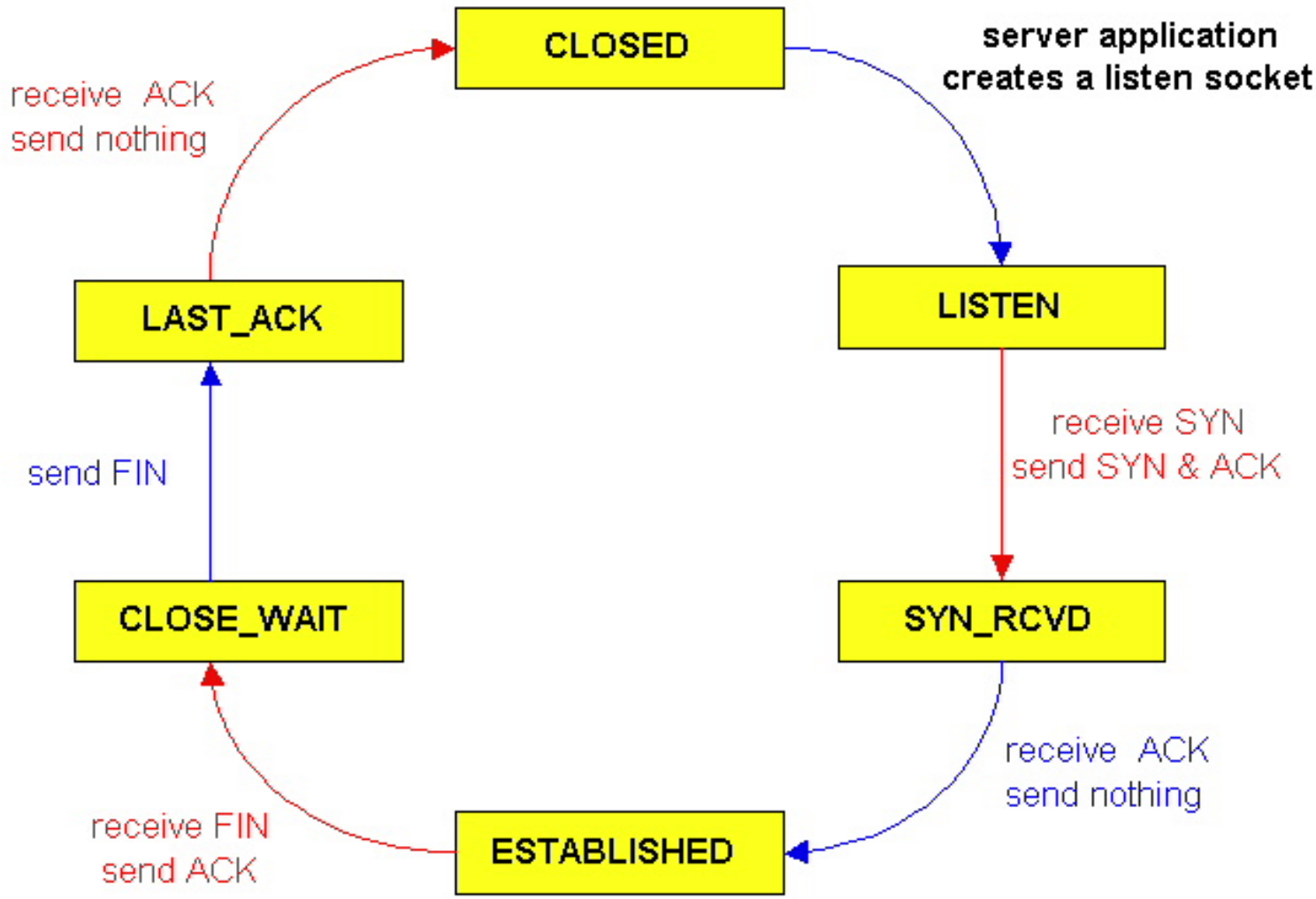
Nota: con piccole modifiche può gestire FIN simultanei



Gestione della connessione TCP: ciclo di vita del lato client



Gestione della connessione TCP: ciclo di vita del lato server



Sommario della prossima lezione: Lo strato di trasporto (3/3)

- Servizi e protocolli dello strato di trasporto
- Multiplazione e demultiplazione delle applicazioni
- Trasporto senza connessione: UDP
- Trasporto con connessione: TCP
- Il controllo della congestione nel TCP