Laboratorio di Programmazione in Rete

a.a. 2005/2006

http://www.di.uniba.it/~lisi/courses/prog-rete/prog-rete0506.htm

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

Orario di ricevimento: mercoledì ore 10-12

N.B. Il presente materiale didattico è stato
prodotto da: dott.ssa V. **Carofiglio**
rielaborato da: dott.ssa F.A. **Lisi**

dib

Sommario della lezione di oggi

- Famiglie e tipi di socket
- Ciclo di vita di una socket per TCP
- Applicazioni Winsock
  - Inizializzazione
  - Sviluppo con DEV-C ++

Socket: famiglie

- Ogni famiglia
  - riunisce i socket che utilizzano gli stessi protocolli (Protocol Family) sottostanti,
  - supporta un sottoinsieme di stili di comunicazione e
  - possiede un proprio formato di indirizzamento (Address Family)
- Esempi di famiglia sono:
  - **Unix Domain sockets**: consentono il trasferimento di dati tra processi sulla stessa macchina Unix
  - **Internet socket (AF_INET):** consentono il trasferimento di dati tra processi posti su macchine remote connesse tramite LAN o Internet

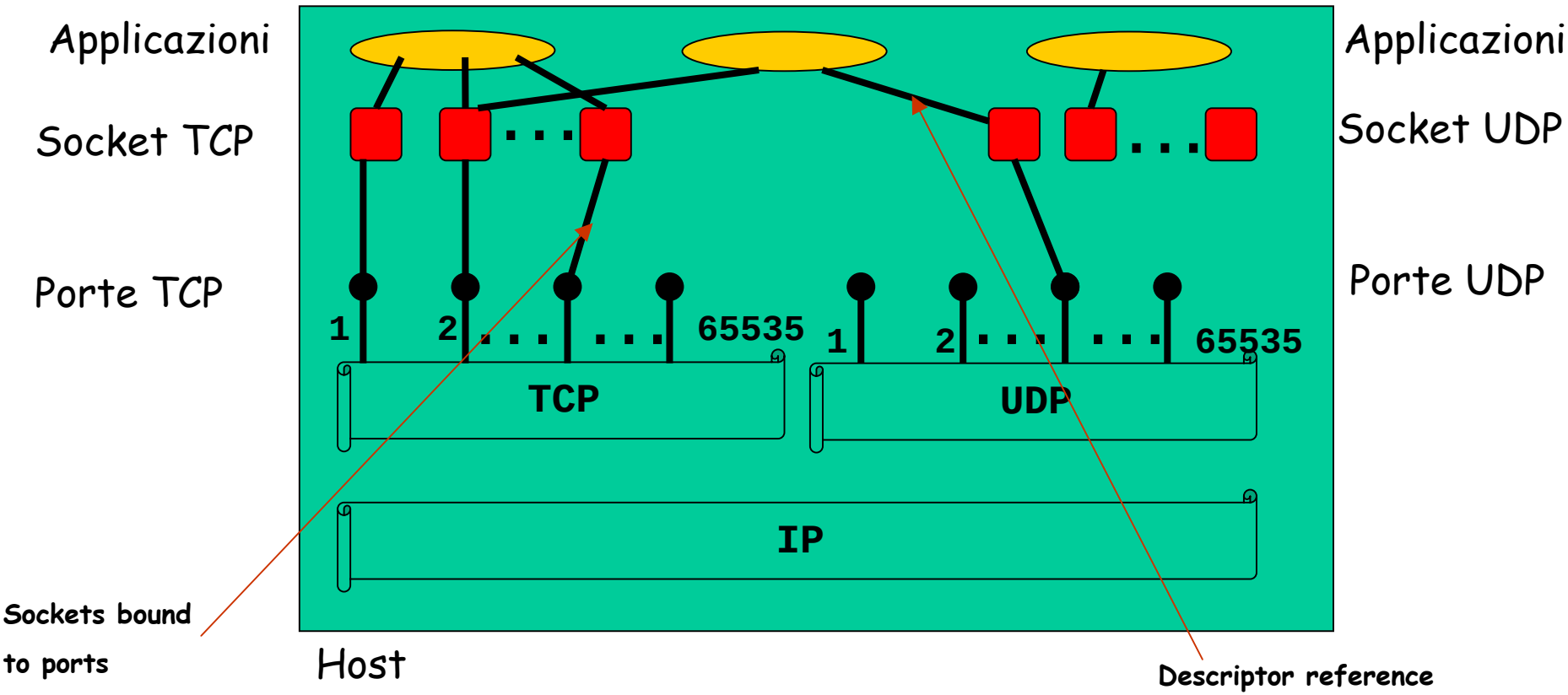
Socket: tipi

- Il tipo della socket definisce la modalità di trasporto adottata per inviare dati:
  - *Streaming Socket* (SOCK_STREAM): Tratta i dati come flusso di byte. Supporta un trasferimento orientato alla connessione, affidabile e full-duplex.
  - *Datagram Socket* (SOCK_DGRAM): Tratta i dati come datagrammi, cioè pacchetti di lunghezza massima prefissata. Supporta un trasferimento privo di connessione, inaffidabile.

- ***Nota bene:***
  - Una socket TCP è data da AF_INET + SOCK_STREAM
  - Una socket UDP è data da AF_INET + SOCK_DGRAM

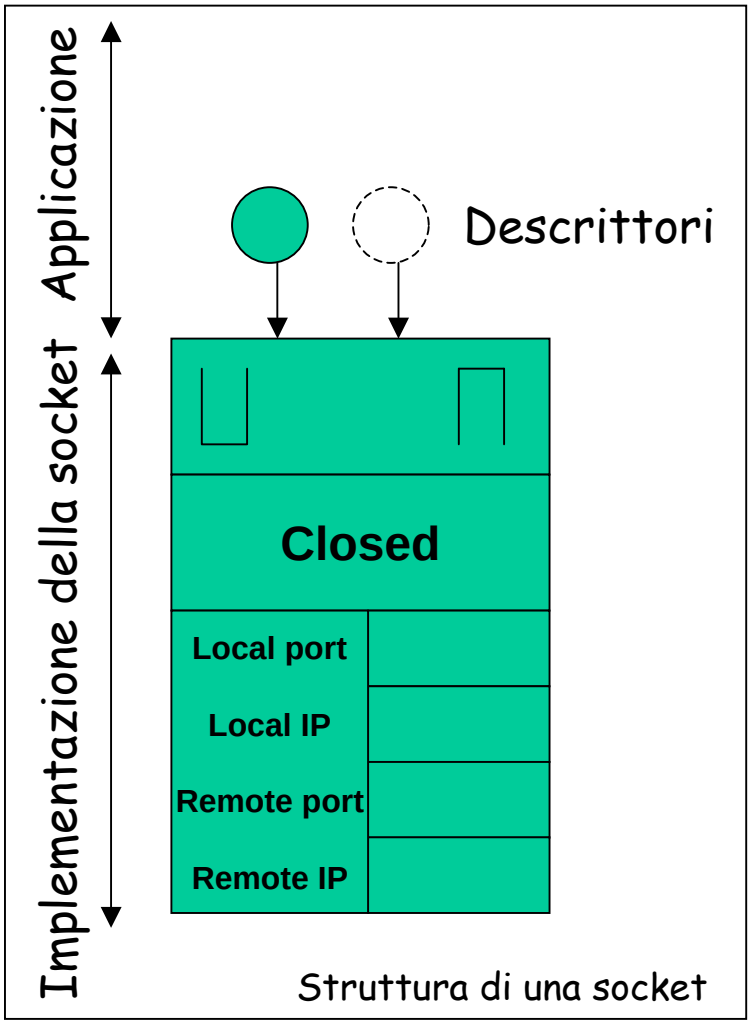
Socket: identificatori

In Internet una socket è univocamente determinata da un *indirizzo IP*, un *protocollo di trasporto* (TCP o UDP) e un *numero di porta*



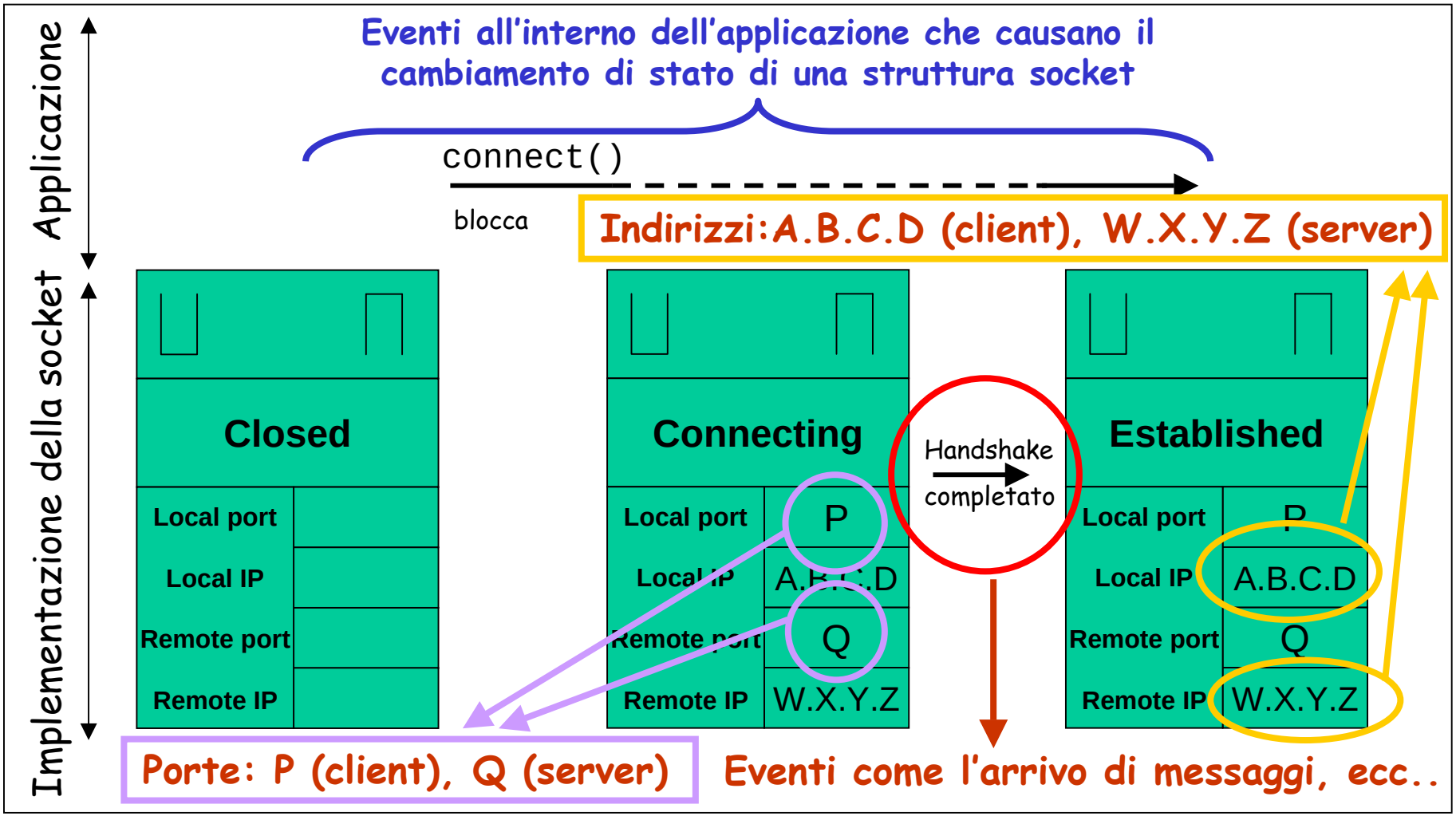
Socket: una astrazione

Strutture dati associate ad una socket TCP

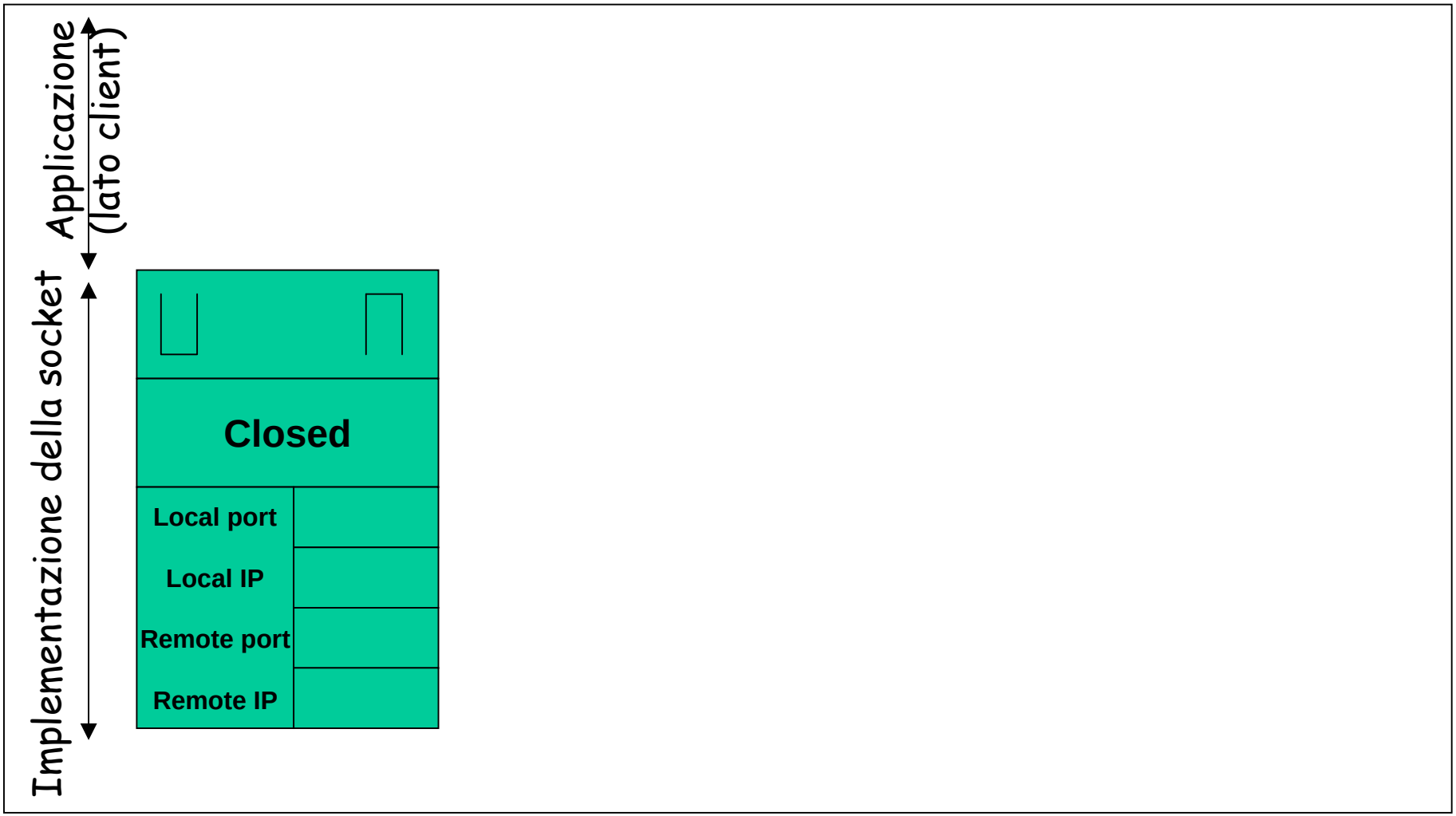


- L'applicazione fa riferimento alla struttura di una socket tramite descrittori
- differenti processi possono fare riferimento alla stessa struttura socket
- informazioni associate alla struttura socket:
  - code di ricezione e invio
  - indirizzi internet
  - informazioni sullo stato dell'handshake (per una socket TCP)

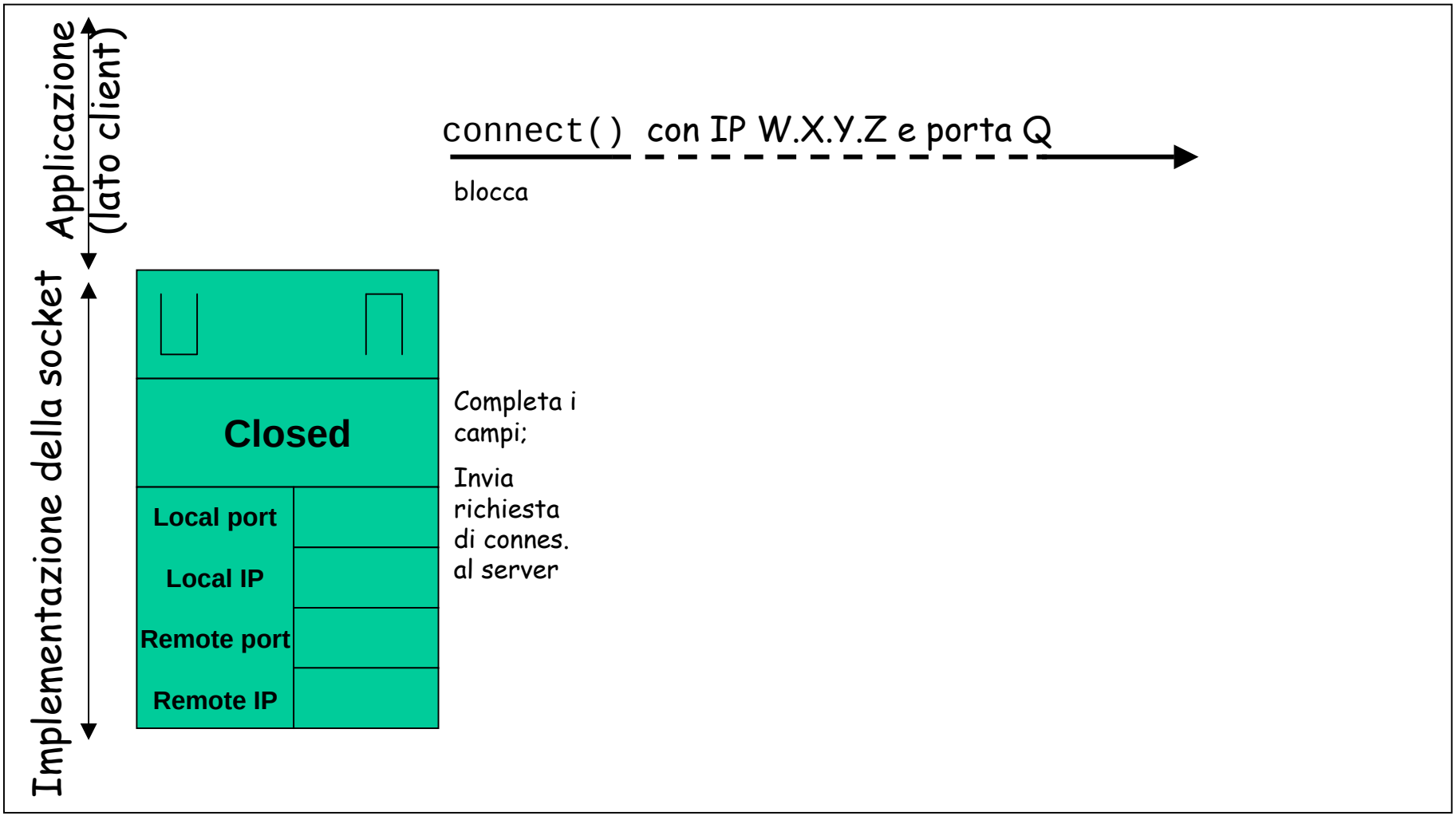
Il ciclo di vita di una socket TCP: Notazione



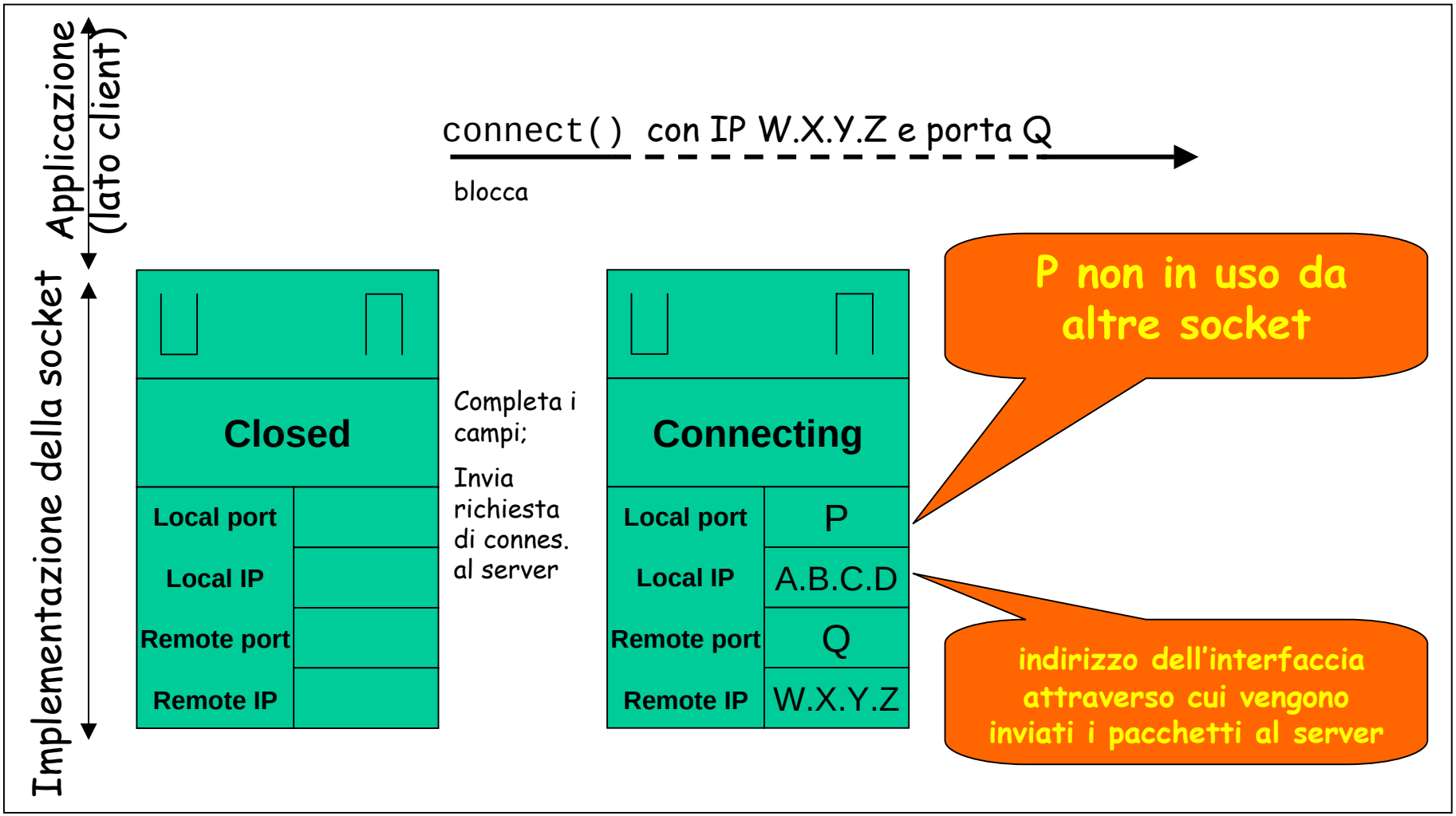
Il ciclo di vita di una socket TCP: Stabilire una connessione (lato *client*)



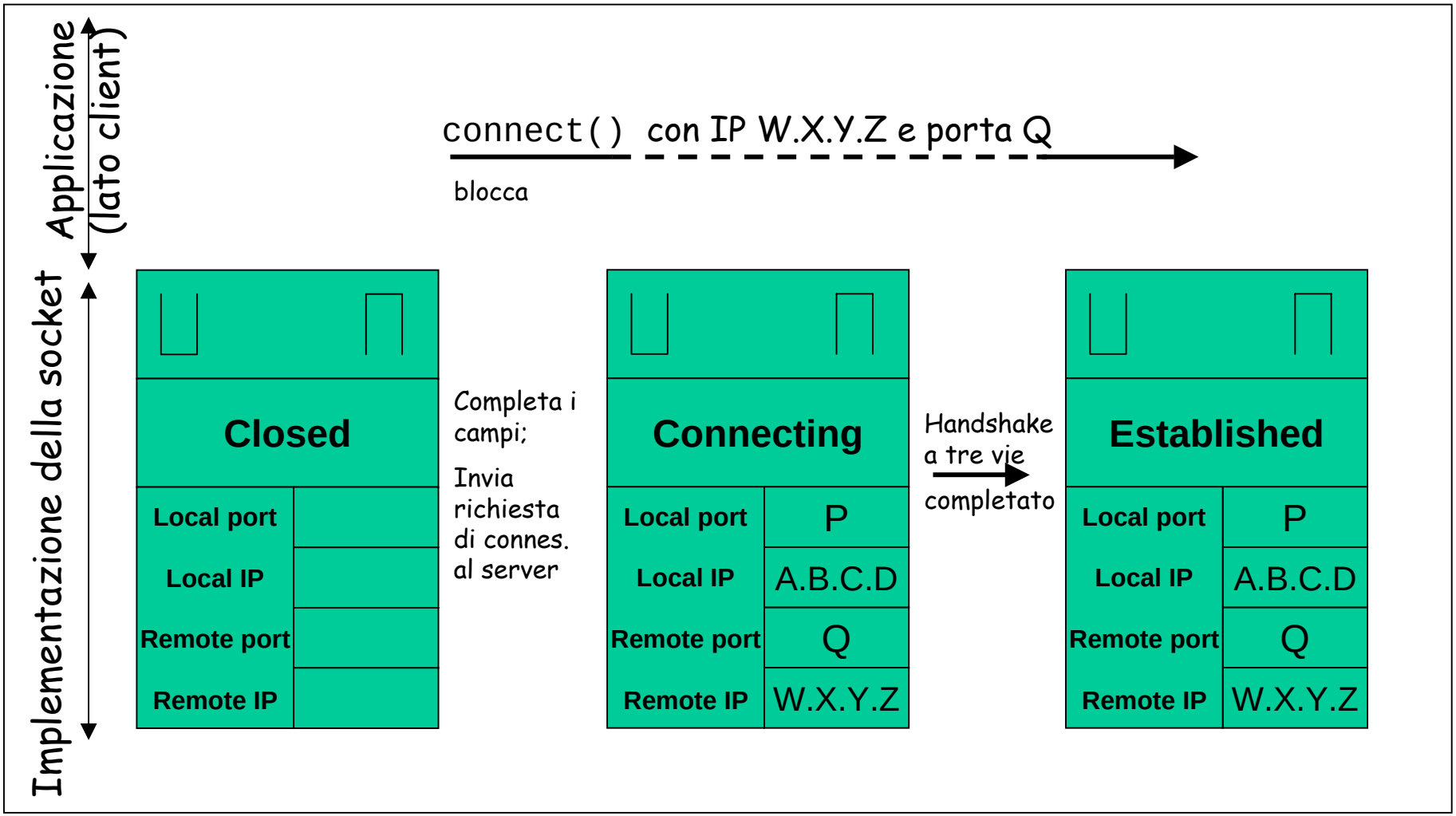
Il ciclo di vita di una socket TCP: Stabilire una connessione (lato *client*)



Il ciclo di vita di una socket TCP: Stabilire una connessione (lato *client*)

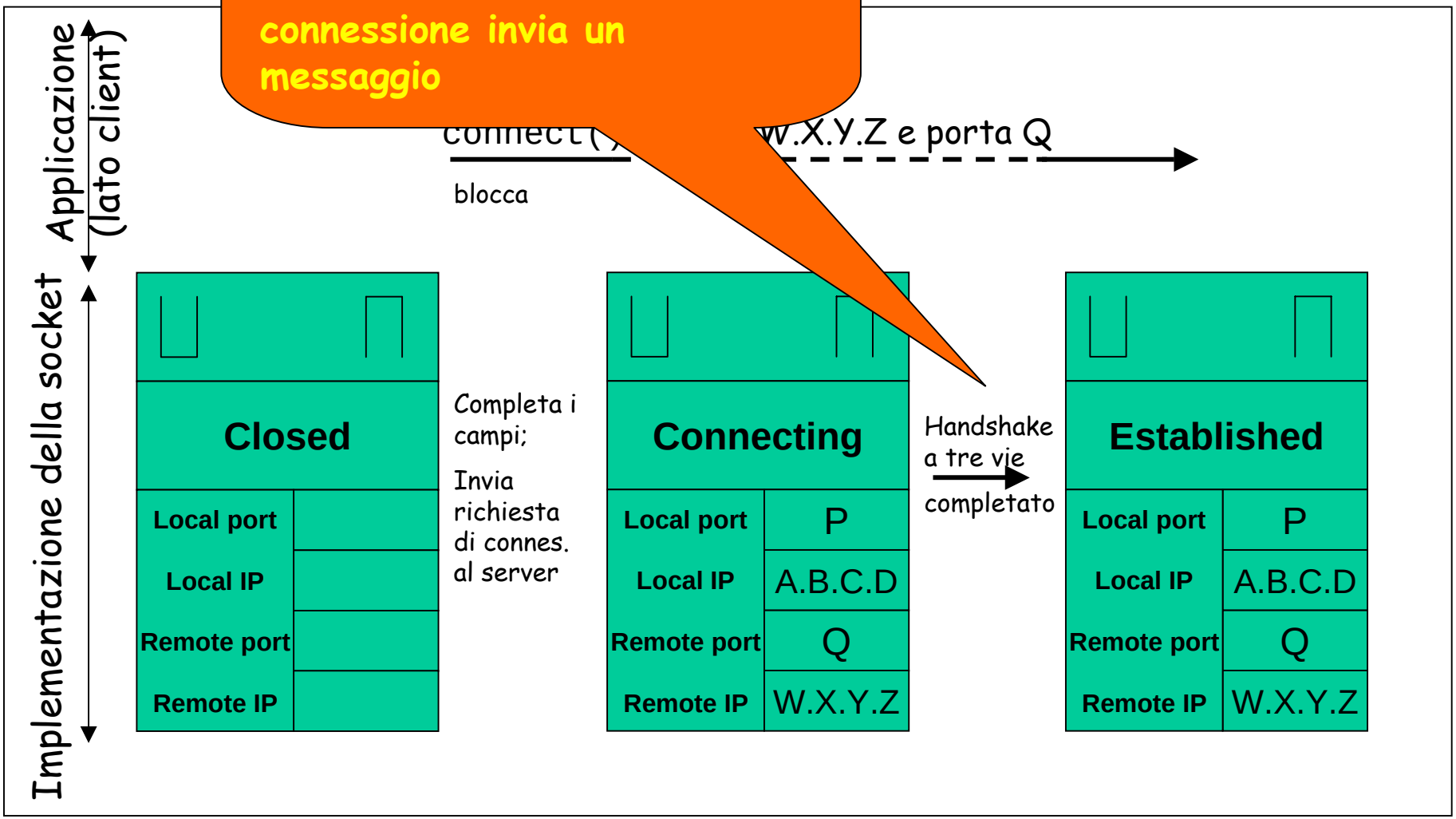


Il ciclo di vita di una socket TCP: Stabilire una connessione (lato *client*)

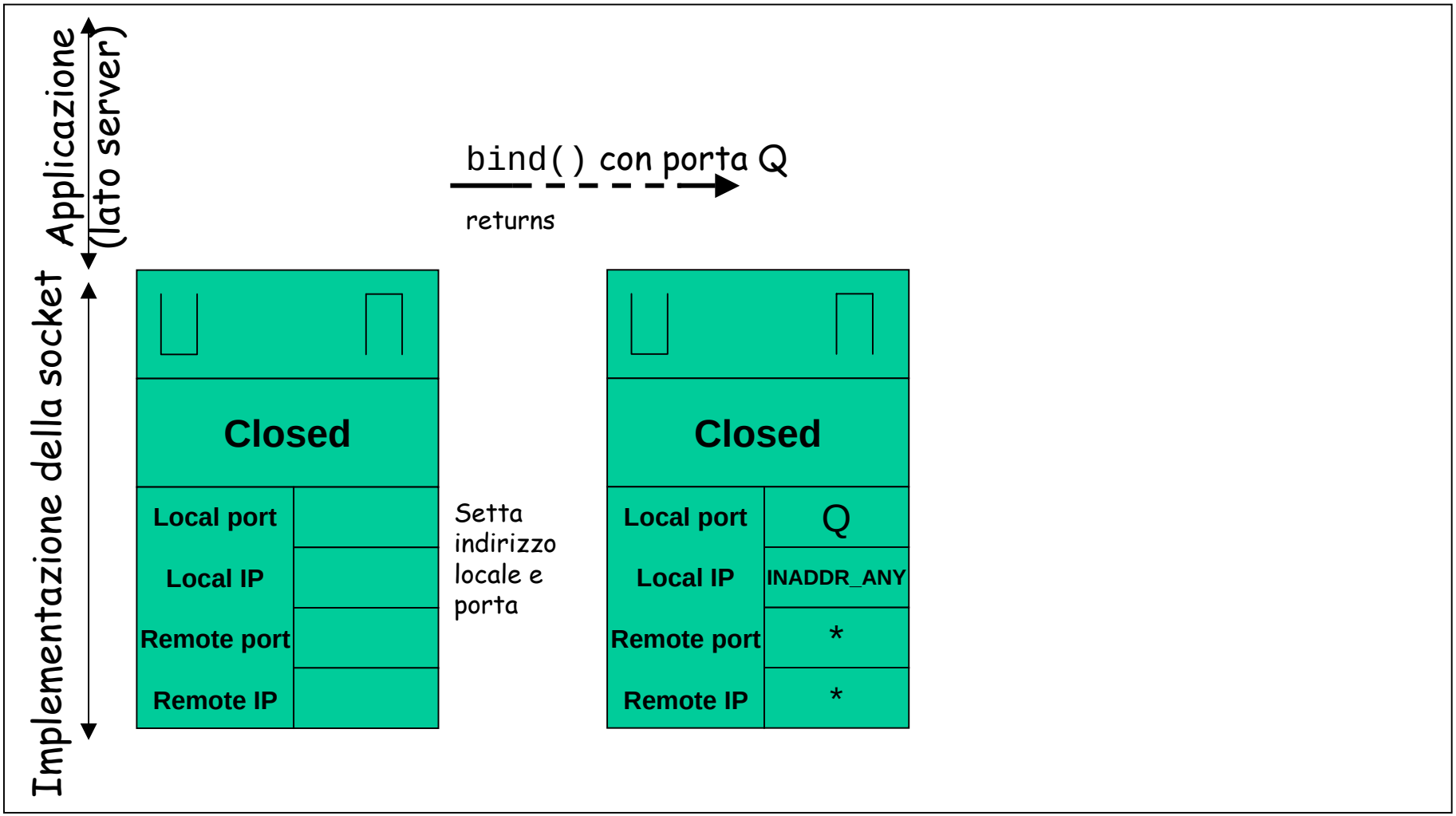


Il ci ocket TCP: Stab e (lato *client*)

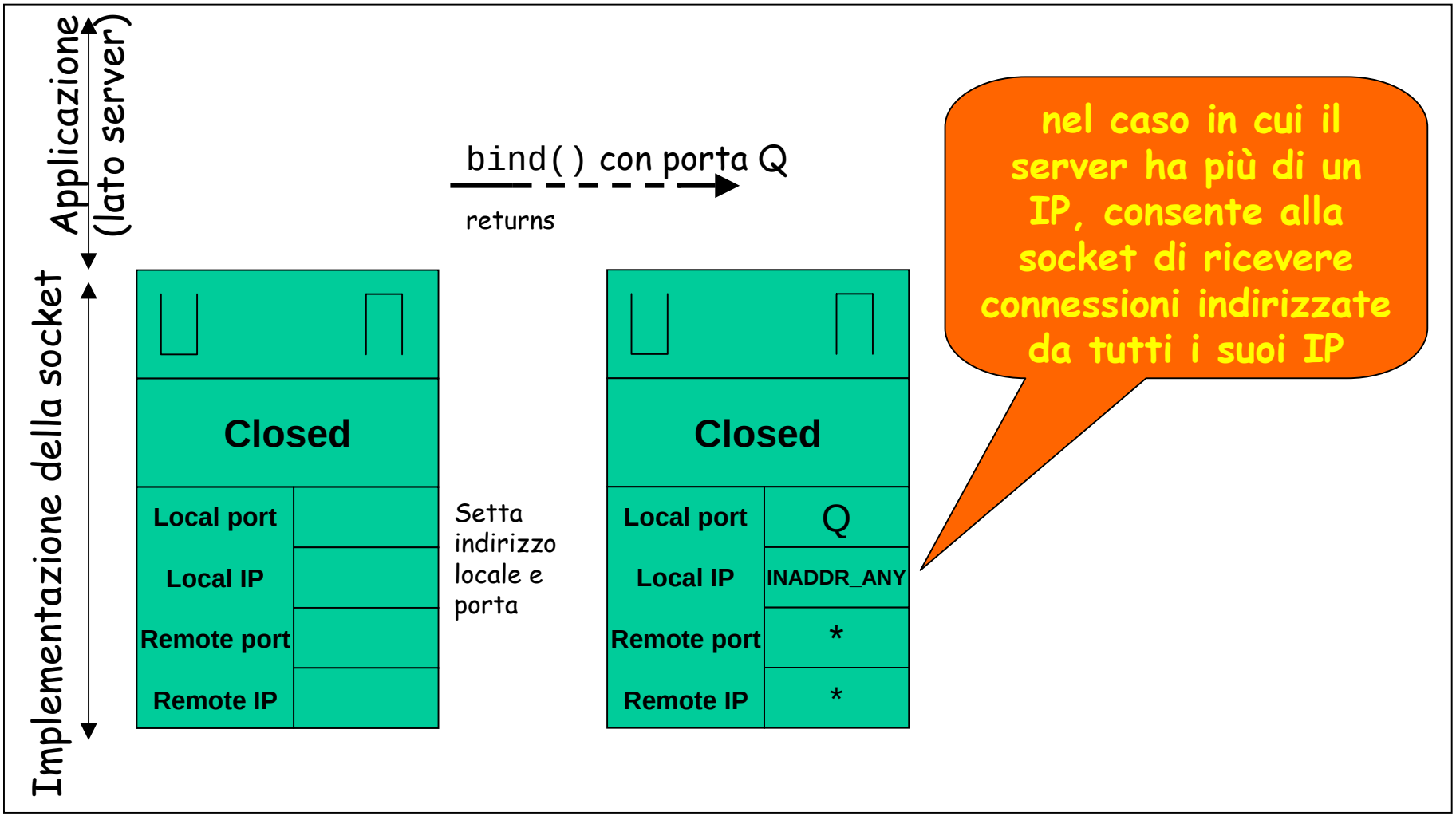
- il client considera la connessione stabilita non appena riceve l'ack dal server
- se il server non accetta una connessione invia un messaggio



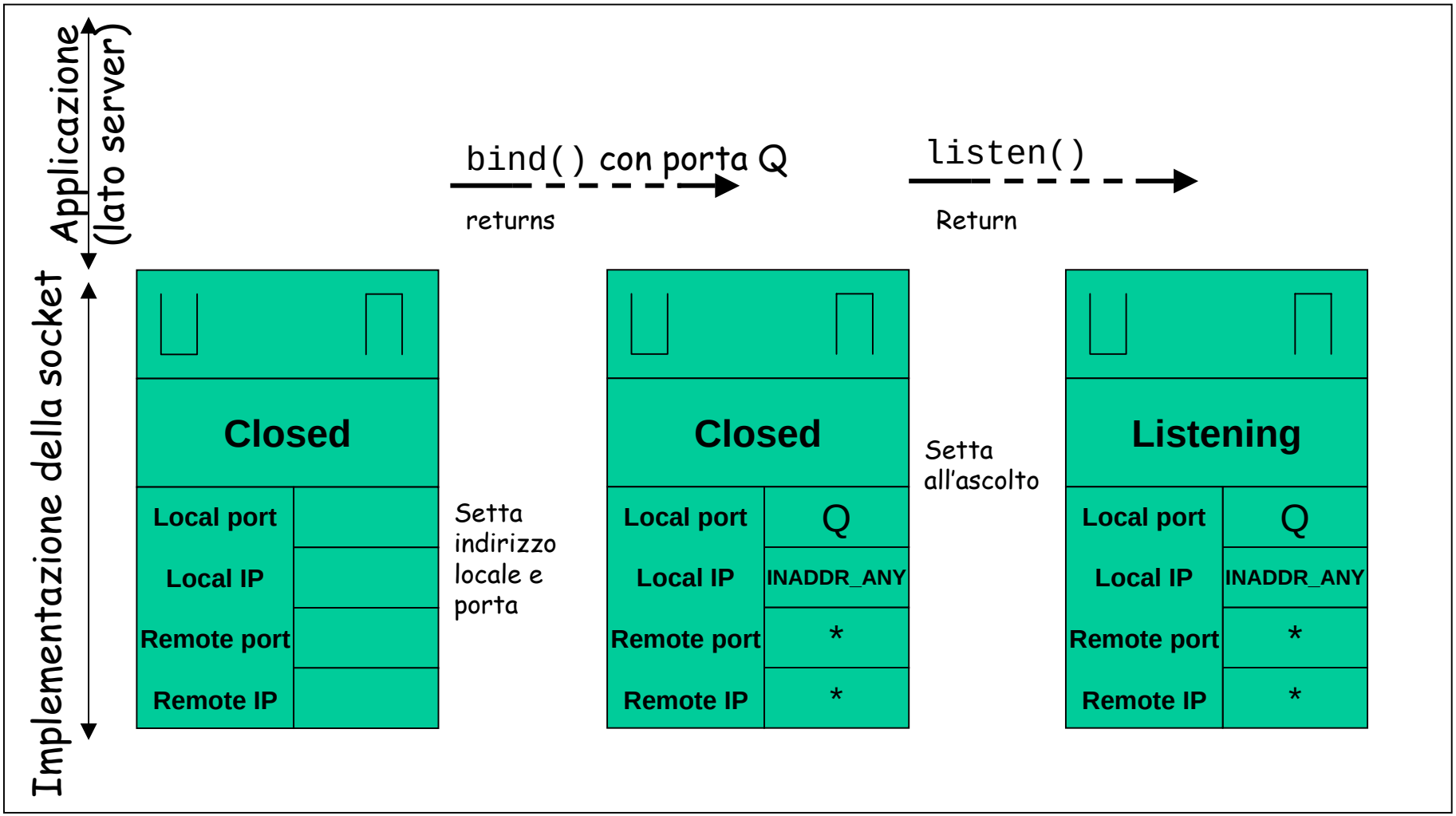
Il ciclo di vita di una socket TCP: Setup della socket (lato *server*)



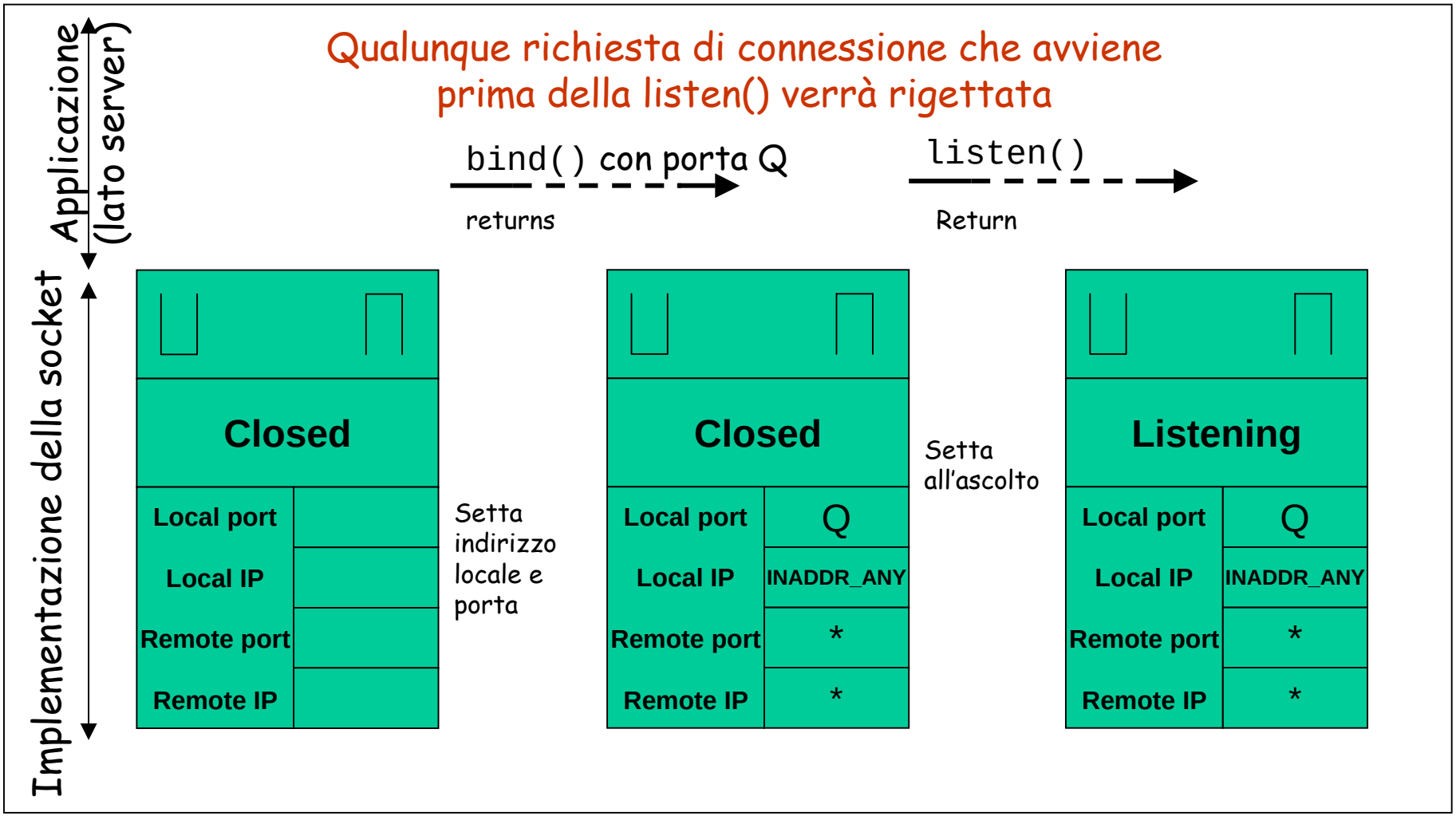
Il ciclo di vita di una socket TCP: Setup della socket (lato *server*)



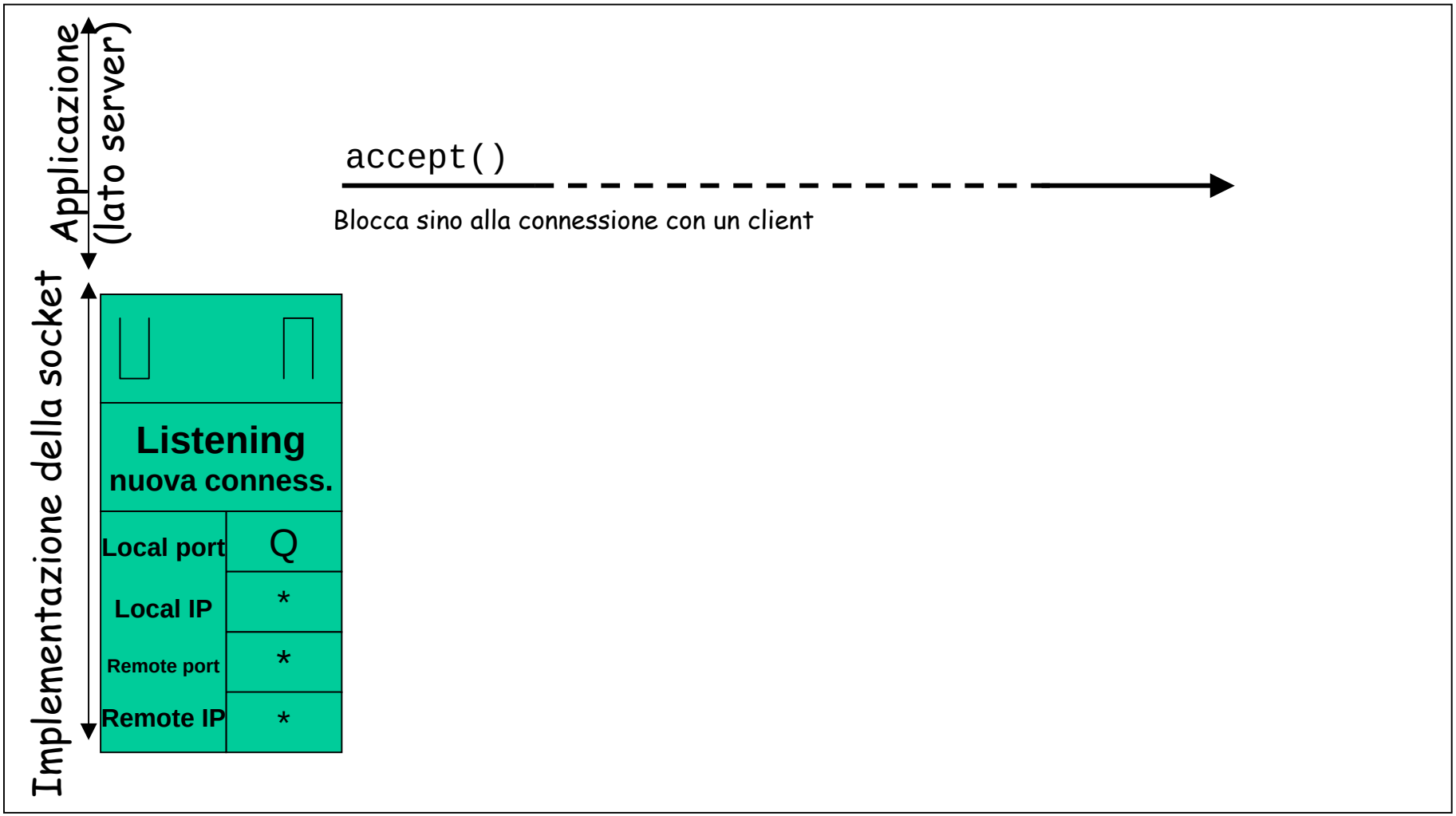
Il ciclo di vita di una socket TCP: Setup della socket (lato *server*)



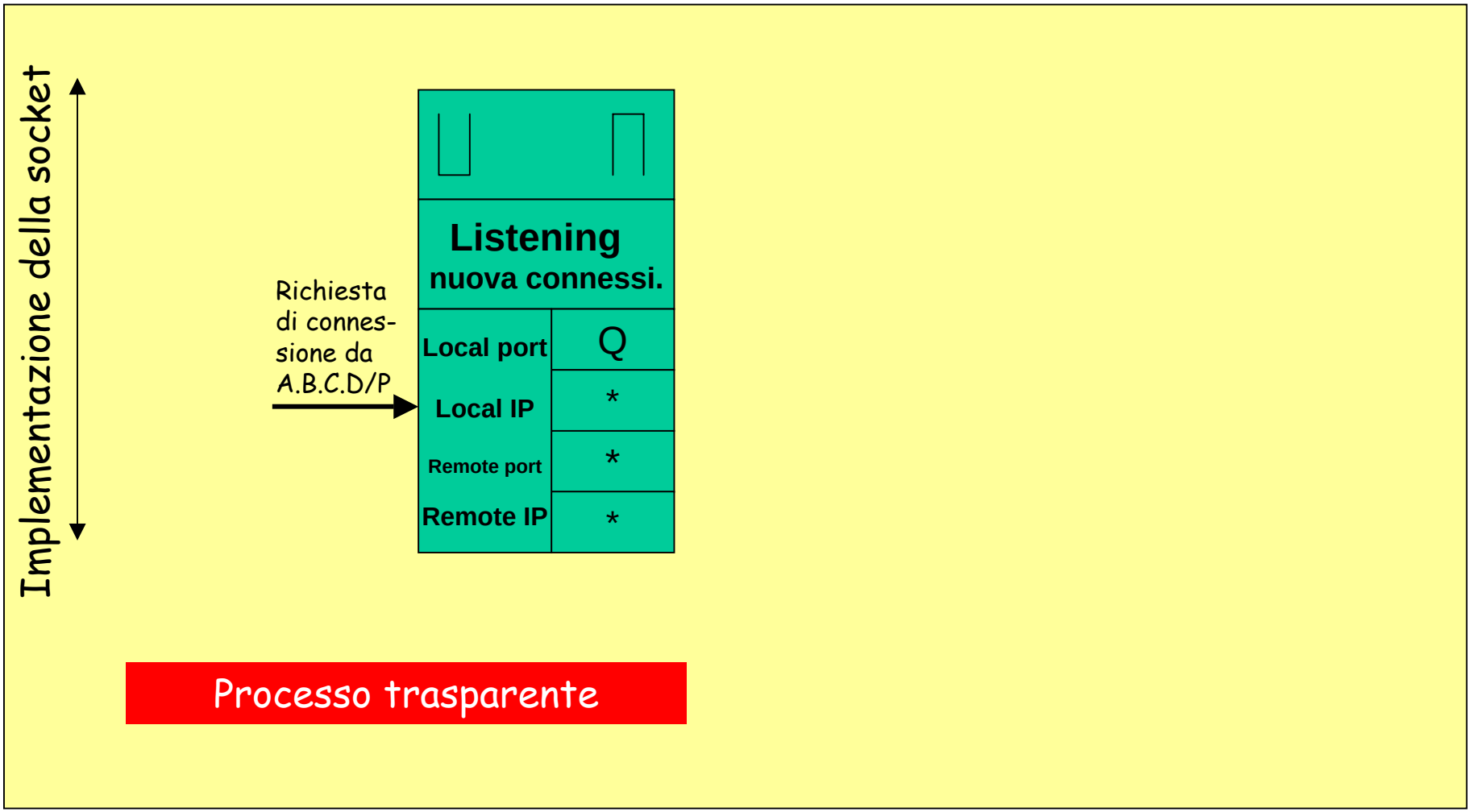
Il ciclo di vita di una socket TCP: Setup della socket (lato *server*)



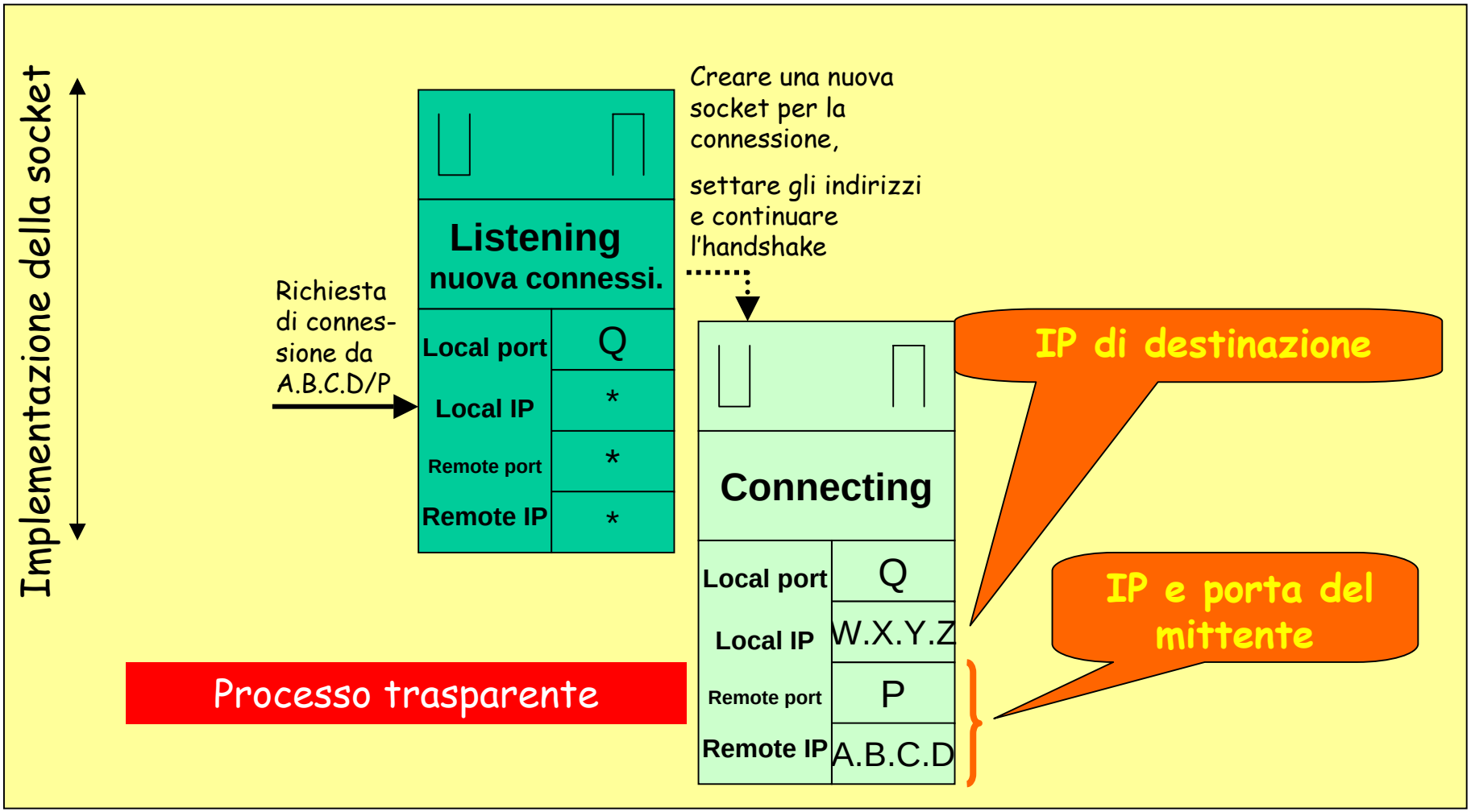
Il ciclo di vita di una socket TCP: Accettazione della connessione (lato *server*)



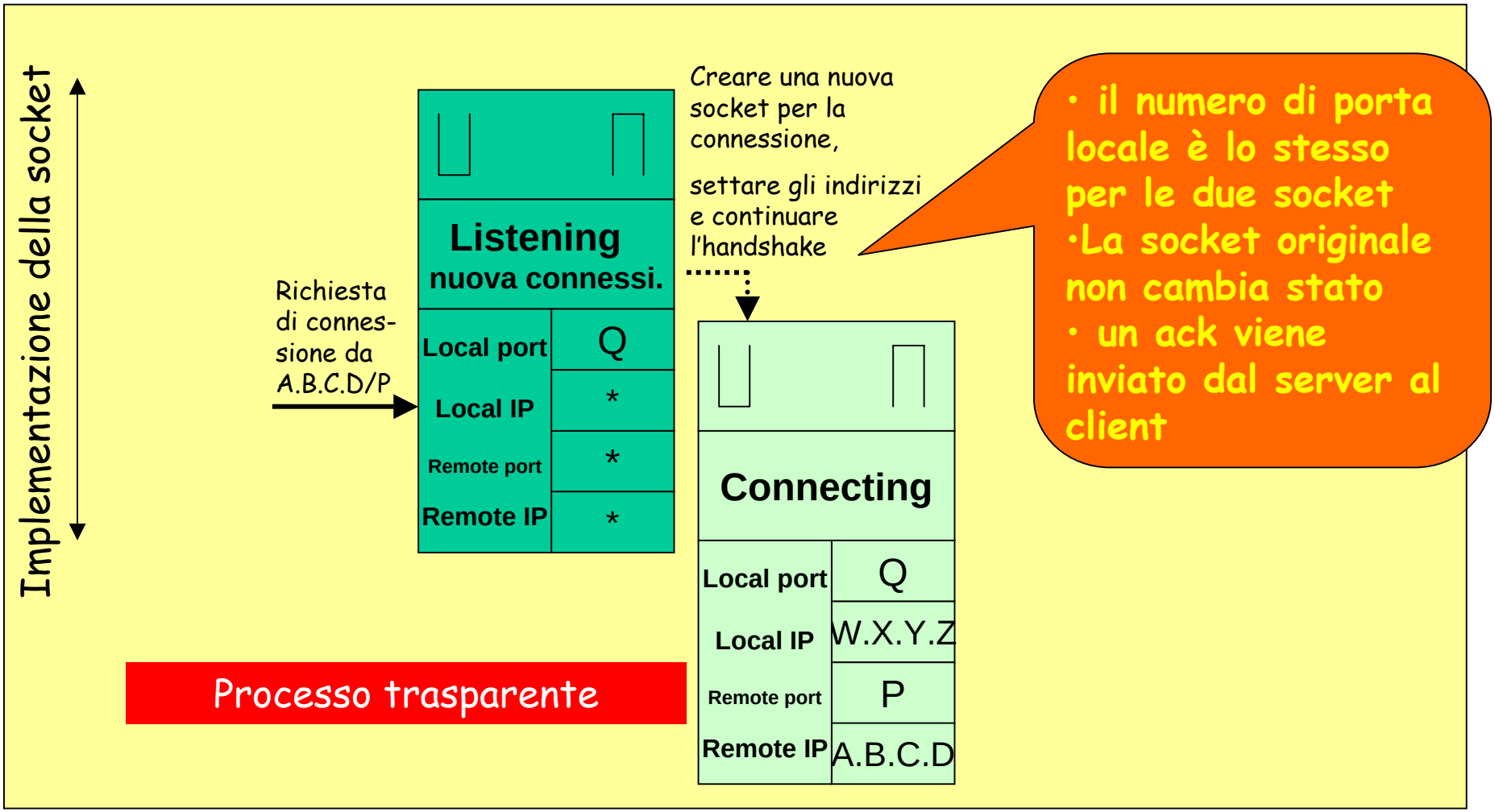
Il ciclo di vita di una socket TCP: Gestione della connessione (lato *server*)



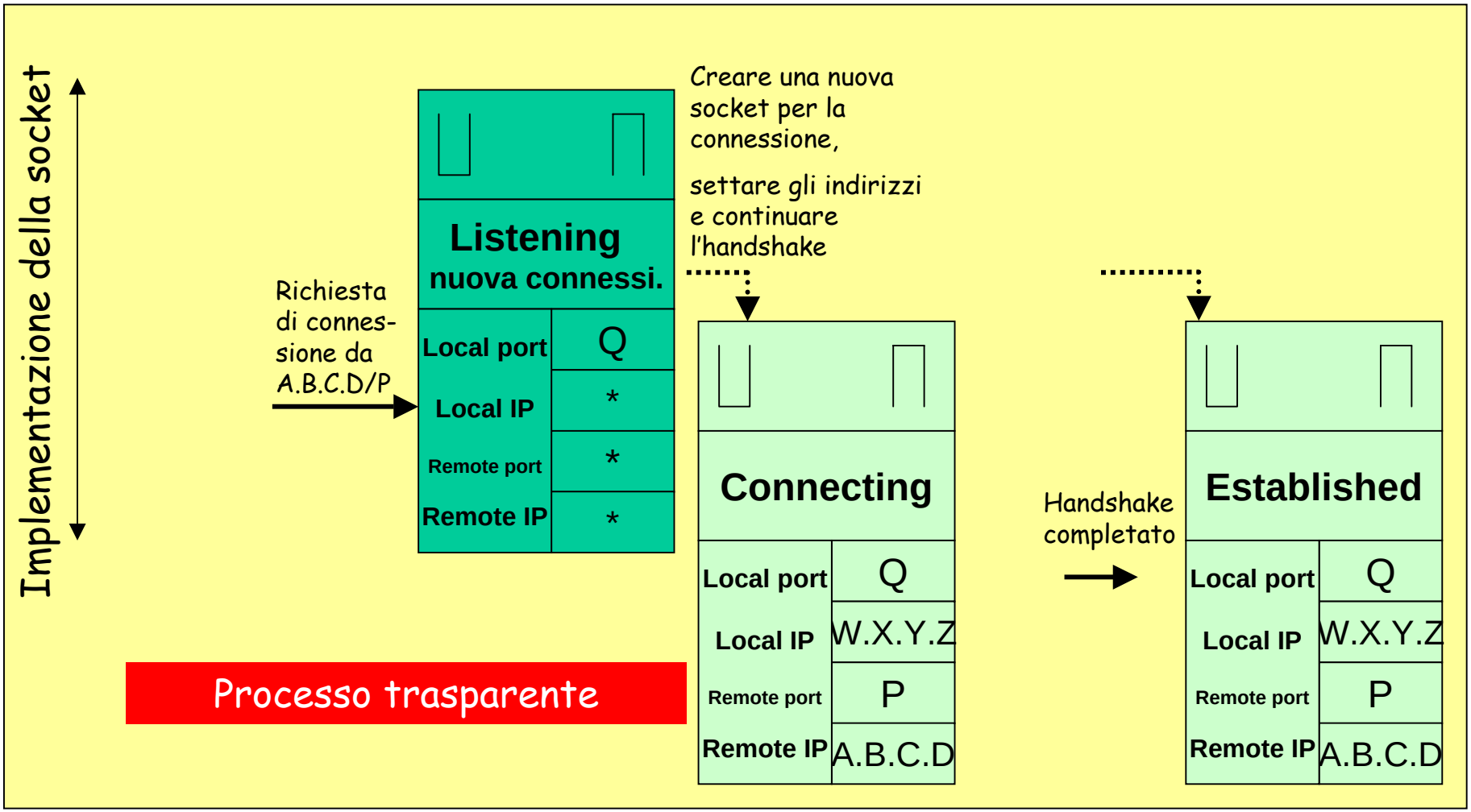
Il ciclo di vita di una socket TCP: Gestione della connessione (lato *server*)



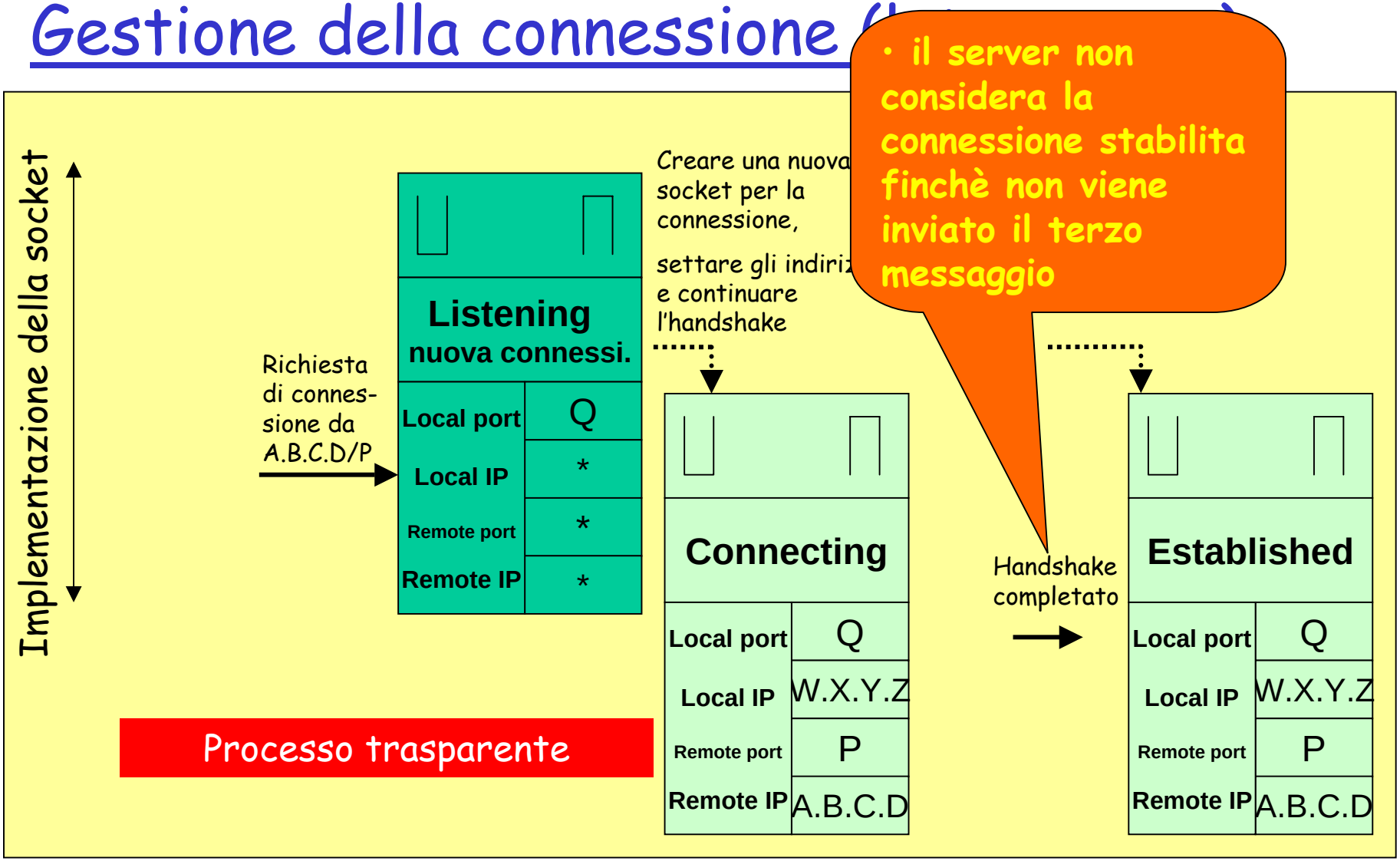
Il ciclo di vita di una socket TCP: Gestione della connessione (lato *server*)



Il ciclo di vita di una socket TCP: Gestione della connessione (lato *server*)

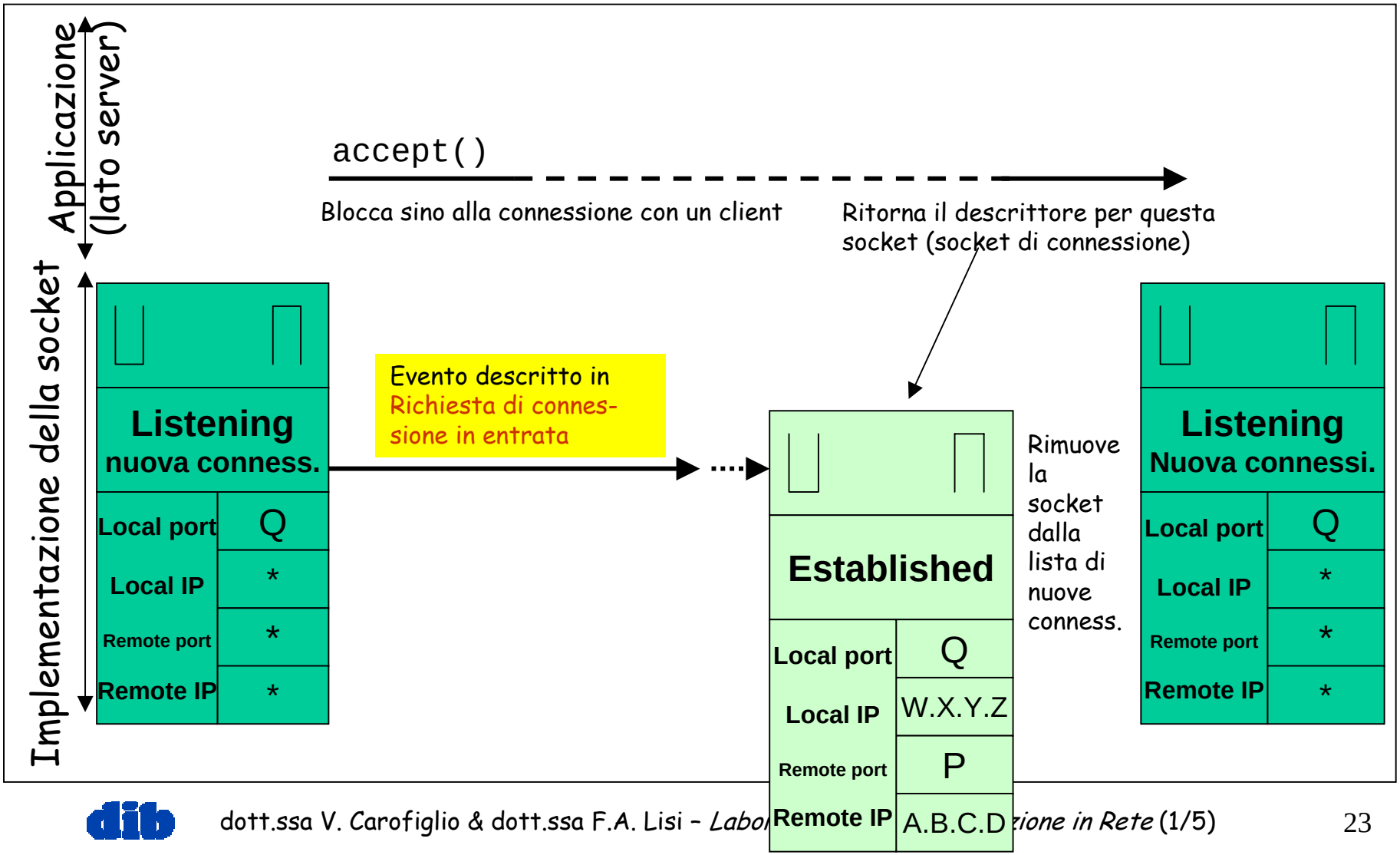


Il ciclo di vita di una socket TCP: Gestione della connessione

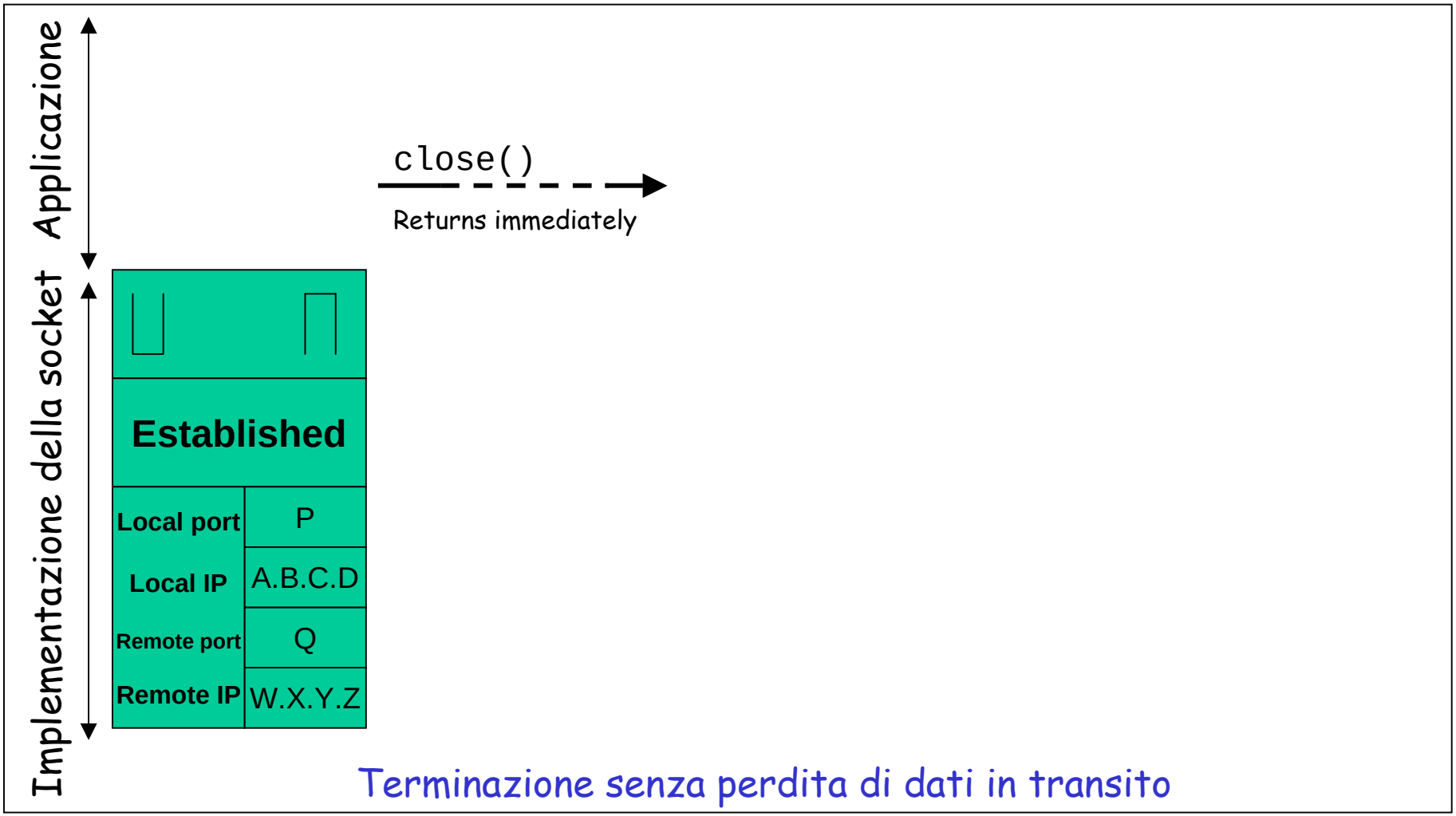


- il server non considera la connessione stabilita finchè non viene inviato il terzo messaggio

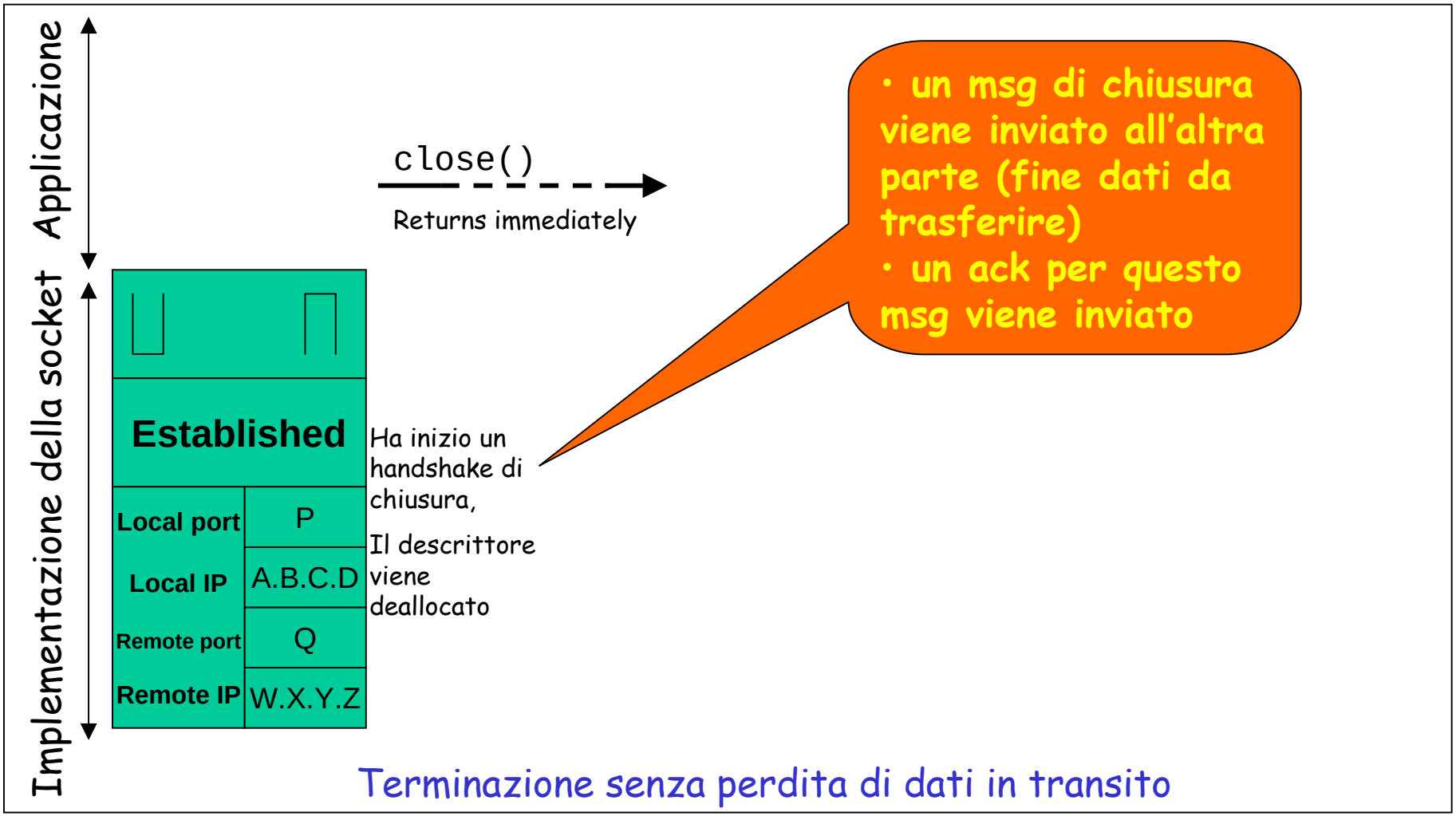
Il ciclo di vita di una socket TCP: Gestione della connessione (lato *server*)



Il ciclo di vita di una socket TCP: Chiusura della connessione

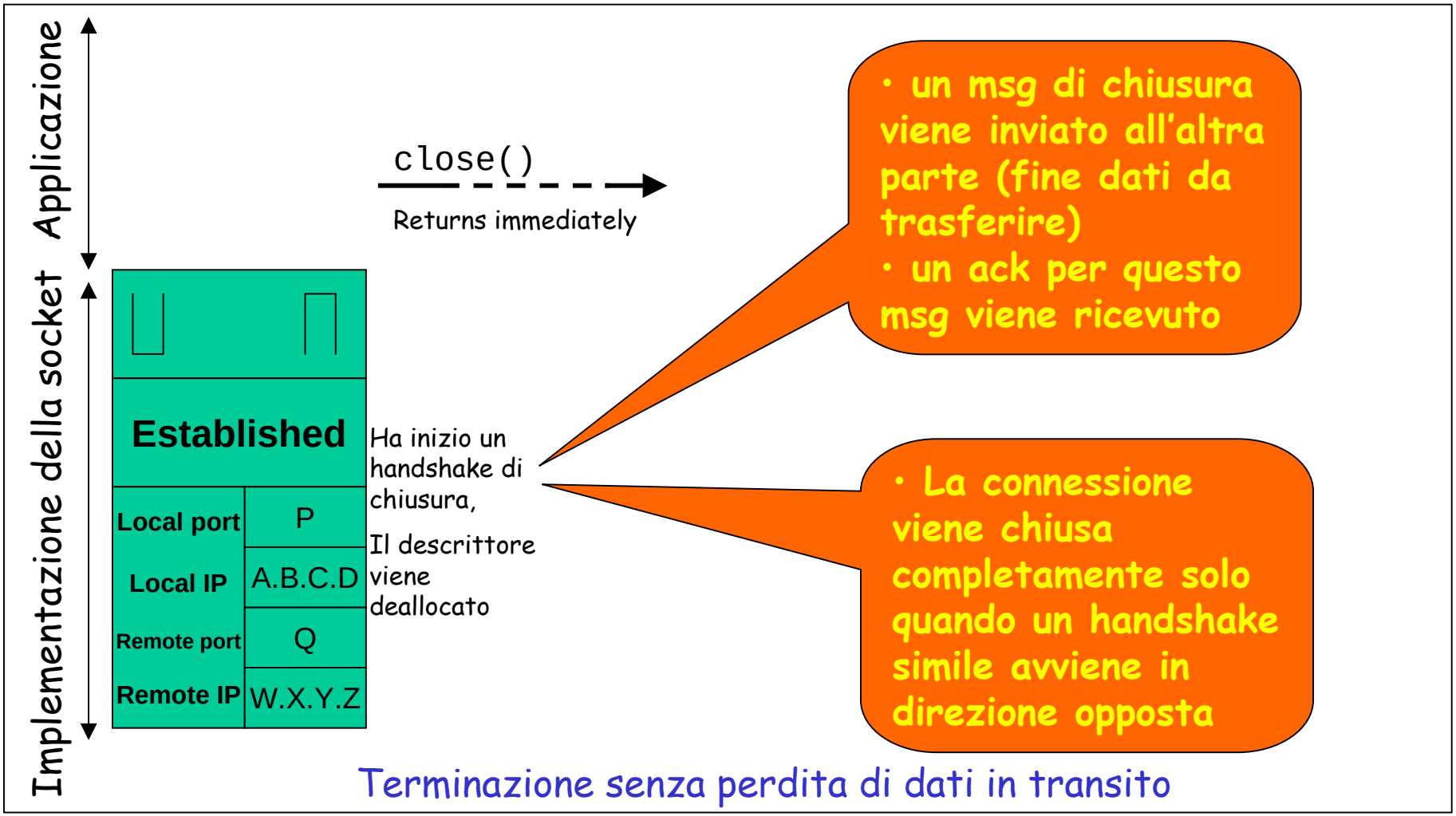


Il ciclo di vita di una socket TCP: Chiusura della connessione



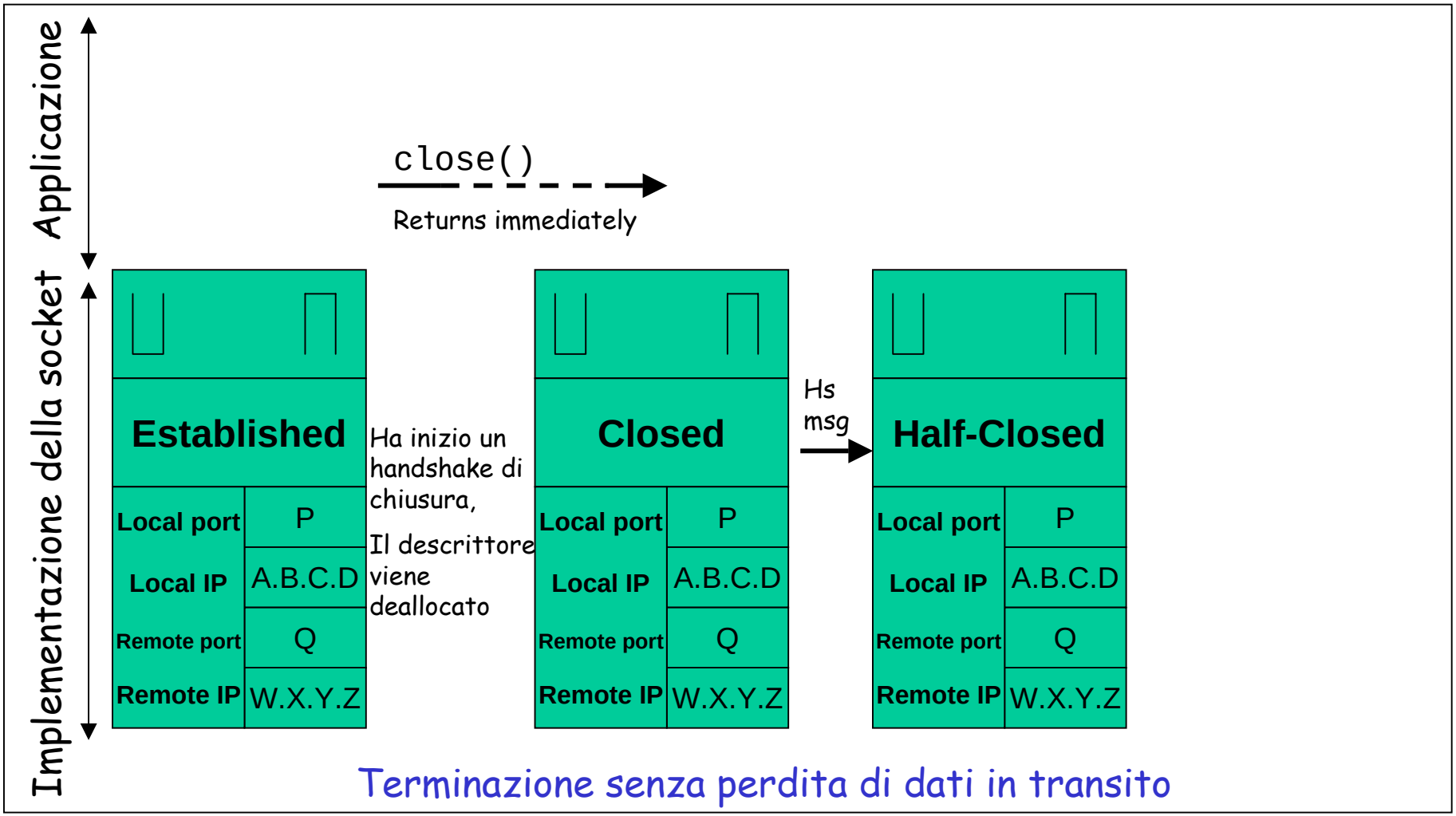
Terminazione senza perdita di dati in transito

Il ciclo di vita di una socket TCP: Chiusura della connessione



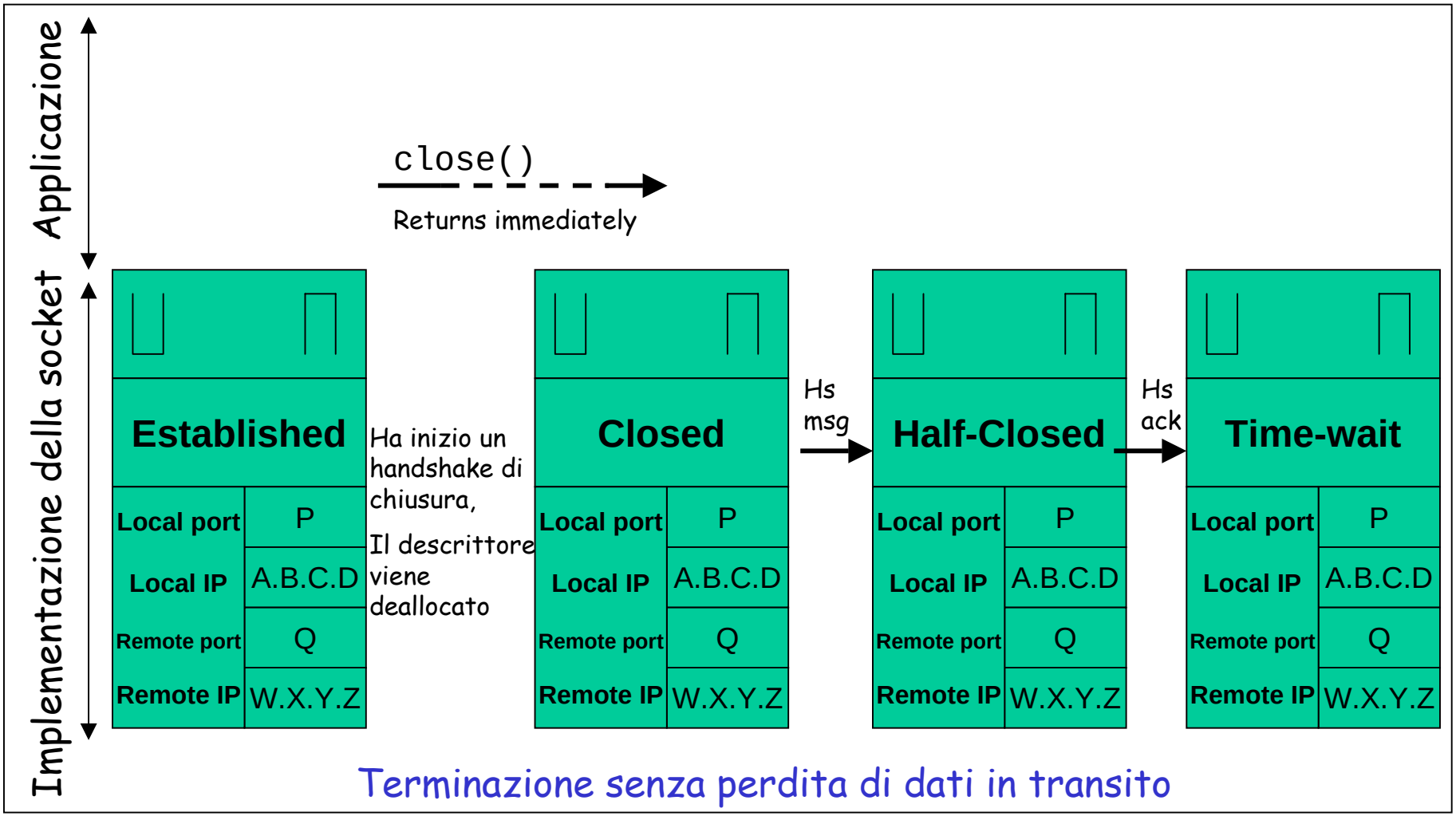
Terminazione senza perdita di dati in transito

Il ciclo di vita di una socket TCP: Chiusura della connessione



Terminazione senza perdita di dati in transito

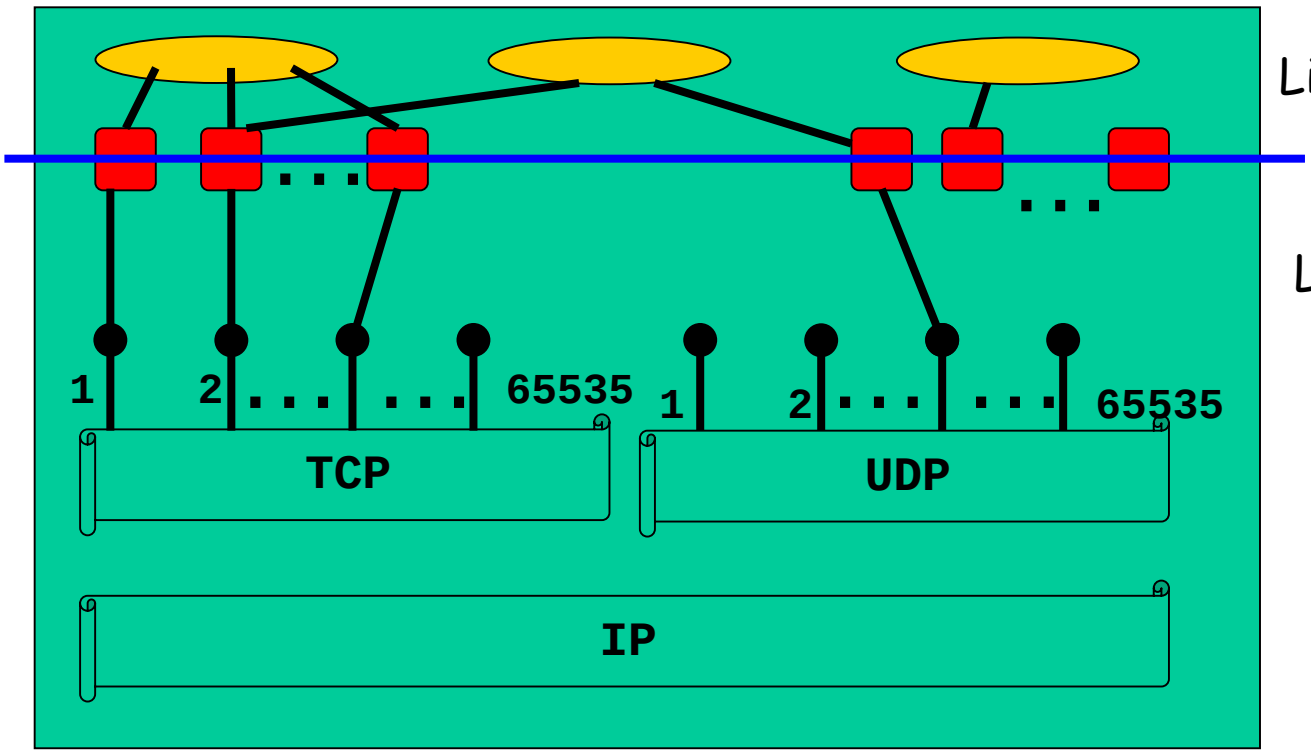
Il ciclo di vita di una socket TCP: Chiusura della connessione



Terminazione senza perdita di dati in transito

Programmazione client/server basata sulle socket

Lo sviluppatore ha il controllo degli aspetti della socket che stanno dal lato del livello applicativo, ma ha poco controllo di quelli che stanno dal lato del livello di trasporto (alcuni parametri)



... in C per Windows

- Le applicazioni client-server basate su socket in C per Windows:
  - sono brevemente indicate con il nome di **applicazioni WinSock** (*Windows Socket Application*)
  - utilizzano la API WinSock (file `winsock.h`)

```
#include<stdio.h>
#include<winsock.h>

void main()
{
 return;
}
```

Applicazioni WinSock: inizializzazione

Tutte le applicazioni WinSock devono essere inizializzate per essere sicuri che le socket Windows siano supportate dal sistema

Per inizializzare un'applicazione Winsock:

- Creare una variabile di tipo WSADATA:

```
WSADATA wsaData;
```

La struttura WSADATA contiene informazioni sull'implementazione delle socket Windows

Applicazioni WinSock: inizializzazione

Tutte le applicazioni WinSock devono essere inizializzate per essere sicuri che le socket Windows siano supportate dal sistema

Per inizializzare un'applicazione Winsock:

- Creare una variabile di tipo WSADATA:

```
WSADATA wsaData;
```

La struttura WSADATA contiene informazioni sull'implementazione delle socket Windows

```
typedef struct WSAData {
  WORD wVersion;
  WORD wHighVersion;
  char szDescription[WSADESCRIPTION_LEN+1];
  char szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short iMaxSockets;
  unsigned short iMaxUdpDg;
  char FAR* lpVendorInfo;
} WSADATA
```

wVersion: versione per la specifica di socket utilizzata

wHighVersion: versione massima per a specifica di socket windows supportabile

Applicazioni WinSock: inizializzazione

Tutte le applicazioni WinSock devono essere inizializzate per essere sicuri che le socket Windows siano supportate dal sistema

Per inizializzare un'applicazione Winsock:

- Creare una variabile di tipo WSADATA:

```
WSADATA wsaData;
```

- Specificare la versione di socket Windows richiesta e recuperare i dettagli della implementazione di tale versione dal sistema operativo

```
Int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
If (iResult != NO_ERROR)
        printf("error at WSASturtup\n");
```

Applicazioni WinSock: inizializzazione

Tutte le applicazioni WinSock devono essere inizializzate per essere sicuri che le socket Windows siano supportate dal sistema

Per inizializzare un'applicazione Winsock:

- Creare una variabile di tipo WSADATA:

```
WSADATA wsaData;
```

- Specificare la versione di socket Windows richiesta e recuperare i dettagli della implementazione di tale versione dal sistema operativo

```
Int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
If (iResult != NO_ERROR)
        printf("error at WSASturtup\n");
```

wVersionRequested: versione di socket windows che il chiamante può usare.

```
int WSAStartup(
    WORD wVersionRequested,
    LPWSADATA lpWSAData
);
```

wVersion: puntatore alla struttura WSDATA che contiene informazioni per l'implementazione della socket

Restituisce 0 in caso di successo. Il codice di errore altrimenti

Applicazioni WinSock: inizializzazione

Tutte le applicazioni WinSock devono essere inizializzate per essere sicuri che le socket Windows siano supportate dal sistema

Per inizializzare un'applicazione Winsock:

- Creare una variabile di tipo WSADATA:

```
WSADATA wsaData;
```

- Specificare la versione di socket Windows richiesta e recuperare i dettagli della implementazione di tale versione dal sistema operativo

```
Int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
If (iResult != NO_ERROR)
        printf("error at WSASturtup\n");
```

specifica il numero di versione di WinSock sul sistema e lo costruisce correttamente

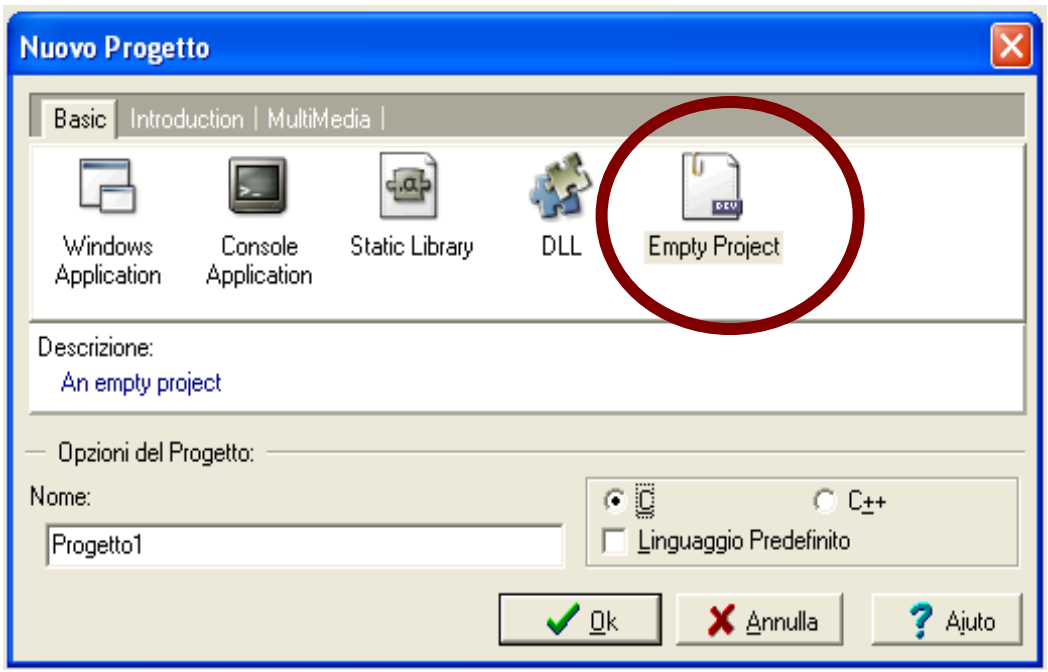
Applicazioni WinSock: inizializzazione

```
WORD wVersionRequested;
WSADATA wsaData;
int err;

wVersionRequested = MAKEWORD( 2, 2 );

err = WSAStartup( wVersionRequested, &wsaData );
if ( err != 0 ) {
    /* Tell the user that we could not find a usable */
    /* WinSock DLL.                                  */
    return;
}

/* Confirm that the WinSock DLL supports 2.2.*/
/* Note that if the DLL supports versions greater    */
/* than 2.2 in addition to 2.2, it will still return */
/* 2.2 in wVersion since that is the version we      */
/* requested.                                        */

if ( LOBYTE( wsaData.wVersion ) != 2 ||
        HIBYTE( wsaData.wVersion ) != 2 ) {
    /* Tell the user that we could not find a usable */
    /* WinSock DLL.                                  */
    WSACleanup( );
    return;
}

/* The WinSock DLL is acceptable. Proceed. */
```

Un esempio!

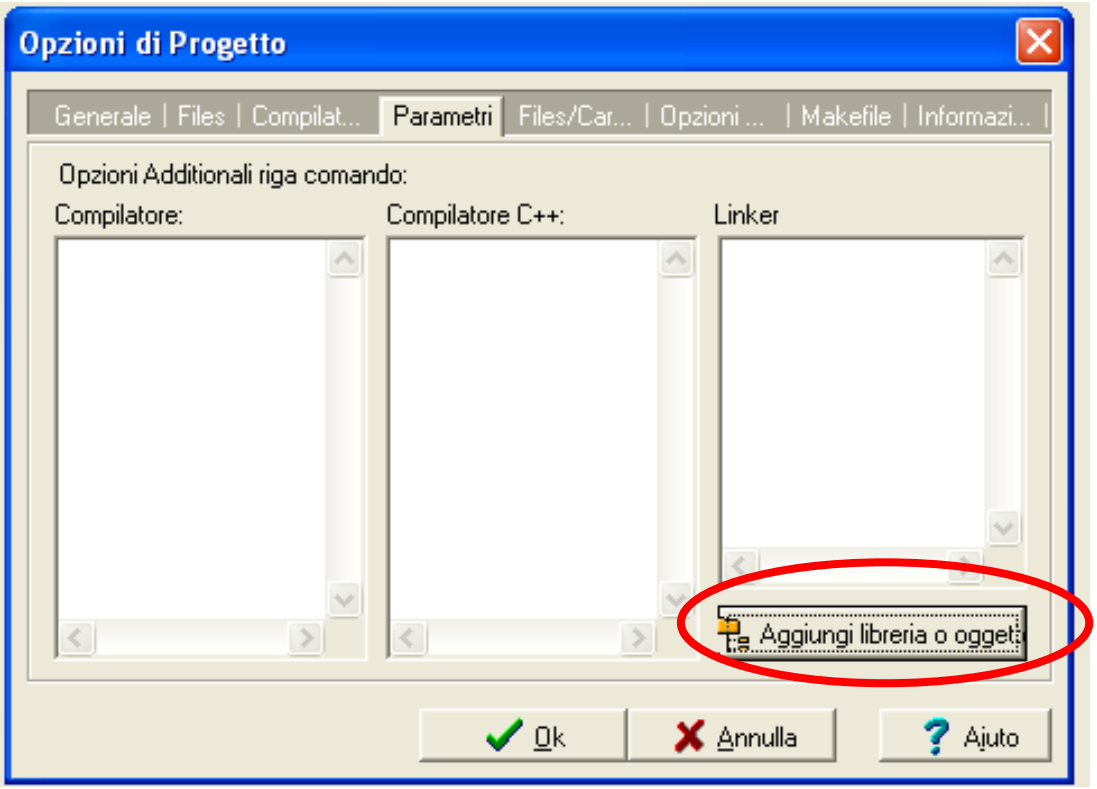
Applicazioni WinSock: sviluppo con DEV-C++

- Selezionare la voce "*File* → ***New Projects***"
- Selezionare "***Empty Project***"



- *Salvare il progetto in "…"*

Applicazioni WinSock: sviluppo con DEV-C++ (cont)



- Clicca su ***parametri*** → ***Aggiungi libreria o oggetto***

- Naviga nella directory "Dev-C++\Lib" e aggiungi il file ***libwsock32*** (oppure ***libws2_32*** per *Winsock2*)

NB: I nomi delle librerie possono cambiare a seconda del compilatore

Es: Visual C++ wsock32.lib (ws2_32.lib)

Commenti conclusivi

- Le esercitazioni per questo corso saranno condotte con:
  - Windows XP
  - Dev C++ (v. V)
- Tuttavia mostreremo le differenze nell'uso delle API socket per UNIX
- Il materiale didattico e alcuni degli esempi mostrati potranno essere scaricati dalla pagina del corso:
  - N.B. Gli esempi scaricabili sono testati e possono essere compilati ed eseguiti in DEV-C++ senza alcuna modifica. Tuttavia in base alla posizione dei file header (.h) e delle dipendenze varie (es. librerie) sul vostro sistema, talvolta potrebbero essere necessarie piccole modifiche.
- Nella prossima lezione affronteremo i dettagli implementativi della interazione client-server in C su TCP per Windows