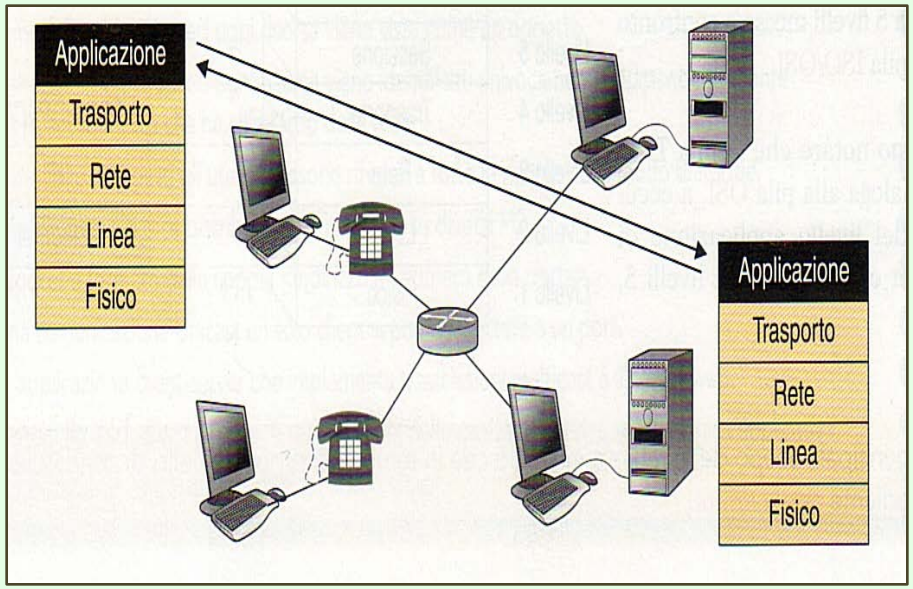
I socket

I Socket

Socket e porte

I sistemi operativi multitasking possono fare girare contemporaneamente **più processi** dove **ogni processo** può rendere disponibili anche **più servizi**. Questi devono essere indirizzabili separatamente.

Per questo motivo con il **protocollo TCP** come anche con il **protocollo UDP** vengono definite le **interfacce speciali** per la comunicazione dati: le **porte**.



Questa **estensione del trasferimento host-to-host** ad **un trasferimento process-to-process**, viene anche chiamato multiplexaggio e demultiplexaggio dell'applicazione.

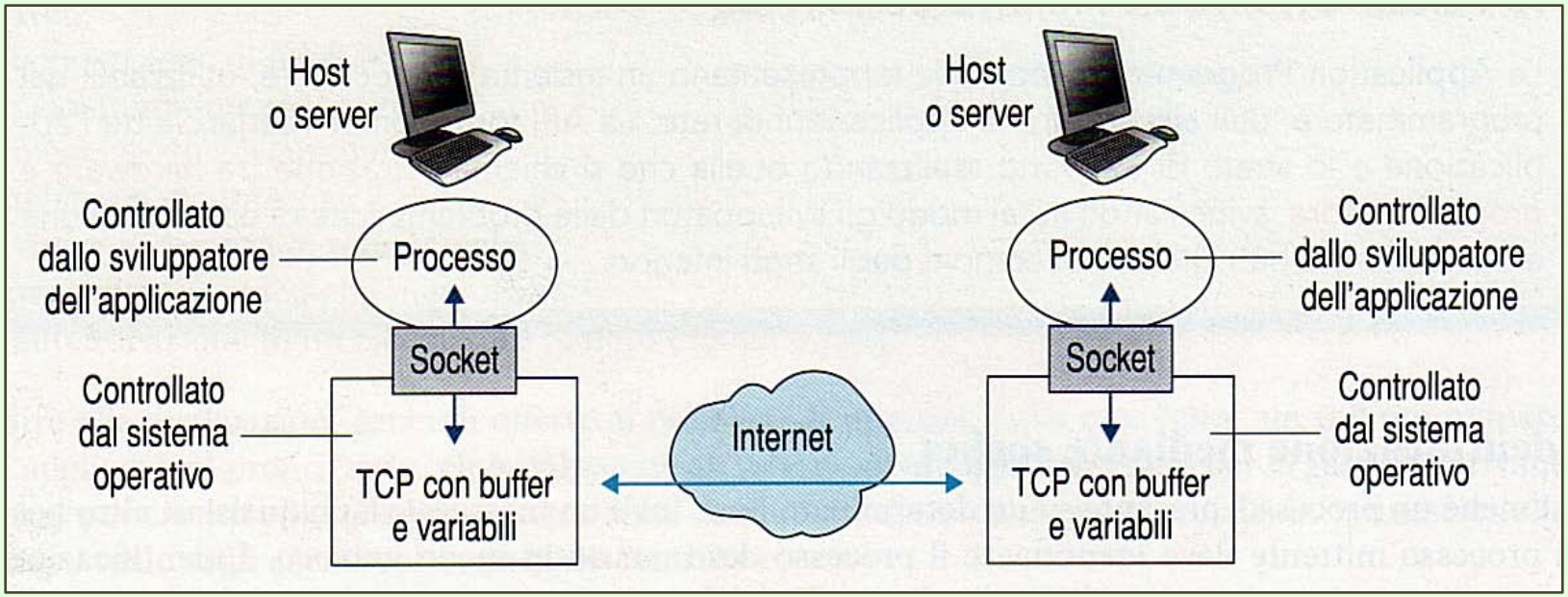
Si definisce **SOCKET** l'indirizzo univoco del programma applicativo nell'intera rete, costituito dalla coppia:

< **indirizzo IP** di una stazione ; **numero di porta** >

Con un socket si può quindi indirizzare un qualsiasi servizio di un processo su una stazione all'interno di una rete.

I Socket

Un processo che vuole inviare un messaggio lo fa uscire dalla **propria** "**interfaccia**" (**socket del mittente**) sapendo che un'infrastruttura esterna lo trasporterà attraverso la rete fino alla "**interfaccia**" **del processo di destinazione** (**socket del destinatario**).



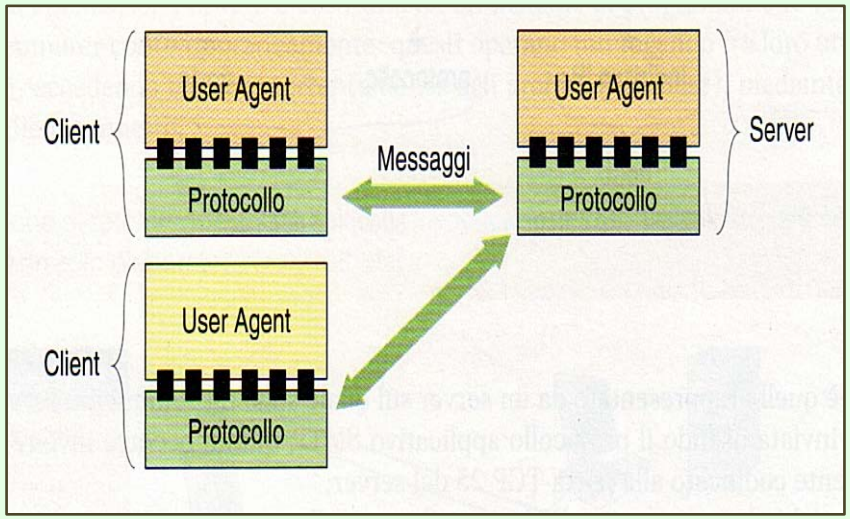
Un **socket** consente di comunicare attraverso la rete utilizzando la pila TCP/IP ed è quindi **parte integrante del protocollo**.

I Socket

Le **API** (Application Programming Interface) mettono a disposizione del programmatore gli strumenti necessari a codificare la connessione per utilizzare il protocollo TCP o UDP.

Esse forniscono

l'**interfaccia** tra l'**applicazione** e lo **strato trasporto**



realizzando un'astrazione tra hardware e programmatore che in questo modo non dovrà preoccuparsi dei problemi di comunicazione e trasferimento dati che sono compiti degli strati inferiori.

Un'**applicazione di rete** può essere vista come composta da due parti:

- una **user agent**, che serve da interfaccia tra l'utilizzatore e gli aspetti comunicativi
- l'**implementazione dei protocolli** che permettono all'applicazione di operare in rete.

Esempio
In un browser Web possiamo individuare le due componenti:

- l'**interfaccia utente** che visualizza le pagine web, ne permette la navigazione e ne richiede di nuovi specificando l'URL
- il **motore del browser** che invia le richieste ai server e riceve le risposte.

I Socket

Gli **standard dei protocolli di trasporto non specificano** come questi debbano **interagire con gli applicativi**.

Nella realizzazione di un'applicazione di rete

il **Socket: è l'interfaccia (software)**

attraverso cui

il livello di trasporto

scambia dati

con le applicazioni

- L'**interfaccia** fra applicazione e TCP/UDP (**Socket) dipende** dall'implementazione del **sistema operativo**
  - uso di **primitive di sistema**
- **Indirizzo del Socket** **indirizzo IP** concatenato ad un **un numero di porta**
  - es.: **137.204.57.85:80**
  - Nel trasporto UDP basta conoscere l'**indirizzo del socket di destinazione**
    - l'indirizzo del socket di origine serve per poter confezionare eventuali risposte
    - il mittente non ha certezza sull'arrivo a destinazione dei datagram
  - Nel trasporto TCP sono **necessari entrambi: socket mittente e socket destinatario**
    - la coppia di indirizzi di socket identifica univocamente le connessioni attive

I Socket

Esempio

Sull'host A ci sono due applicazioni attive rispettivamente sulle porte 3301 e 3300 e richiedono entrambe un servizio HTTP all'host C.

Host A: <137.204.59.10:3301>
Host A: <137.204.59.10:3300>

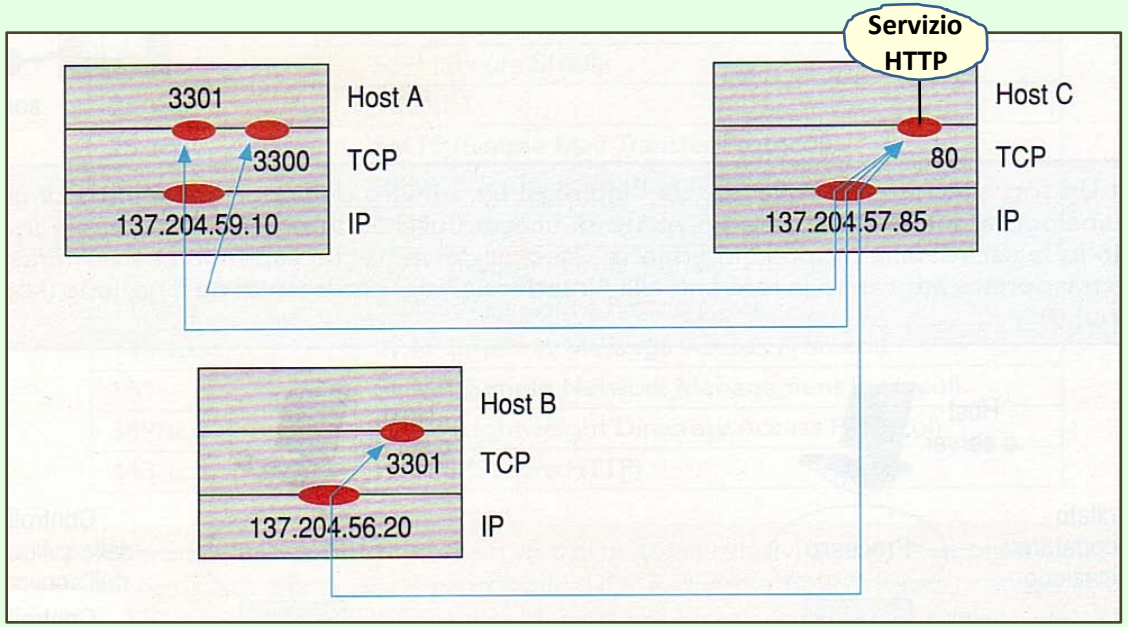
Sull'host B c'è un'applicazione attiva sulla porta 3301 che richiede anch'essa un servizio HTTP all'host C.

Host B: <137.204.56.20:3301>

Sull'host C c'è un'applicazione attiva sulla porta 80 che serve le richieste di tipo HTTP.

Host C: <137.204.57.85:80>

L'esempio mostra la necessità dell'**indirizzo del socket** per individuare univocamente l'applicazione a cui recapitare la risposta, infatti non sarebbe sufficiente il solo indirizzo IP.



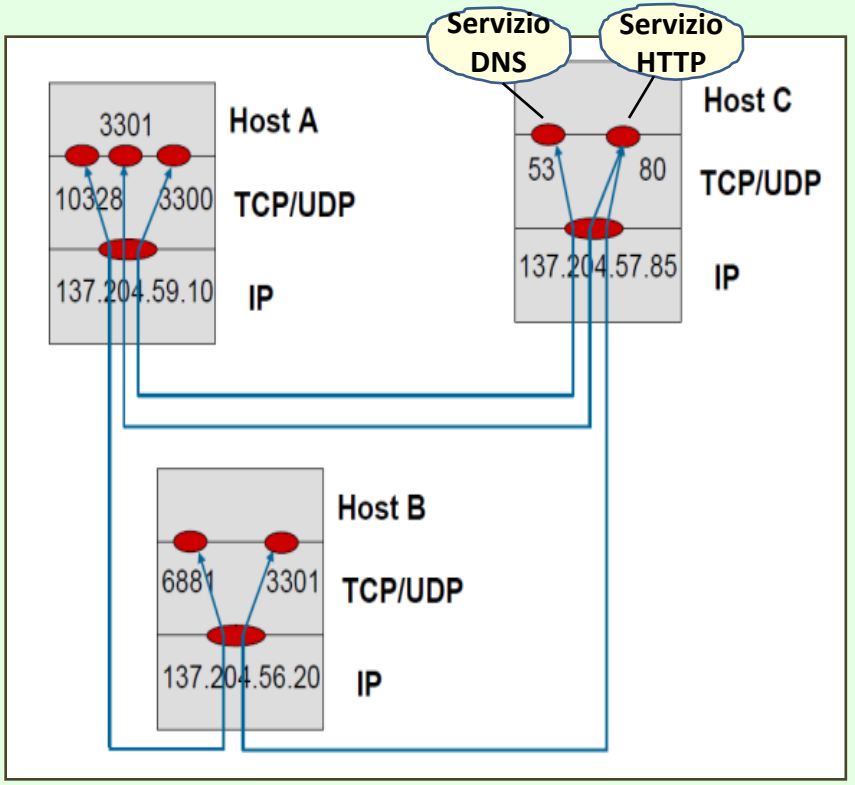
I Socket

Esempio

Host A: <137.204.59.10:10328>
Host A: <137.204.59.10:3301>
Host A: <137.204.59.10:3300>

Host B: <137.204.56.20:6881>
Host B: <137.204.56.20:3301>

Host C: <137.204.57.85:53>
Host C: <137.204.57.85:80>



Famiglie e tipi di socket

Esistono varie **famiglie di socket** (dette anche ***domini***) dove ogni famiglia riunisce i socket che utilizzano gli stessi protocolli (protocol family) sottostanti.

Tra le **famiglie di socket** importanti:

- **Internet socket** (**AF_INET**): permette il trasferimento di dati tra processi posti su macchine remote connesse tramite **LAN** o **Internet.**
  In questo caso l'**indirizzo** è specificato da due valori:
  - l'***indirizzo IP*** (a 32 bit) che individua univocamente un host Internet
  - il ***numero di porta*** (a 16 bit) che specifica una particolare porta dell'host
- **Unix Domain socket** (**AF_UNIX**): permette il trasferimento di dati tra processi posti sulla **stessa macchina Unix**.
  In questo caso l'**indirizzo** è specificato dal ***pathname*** valido nel file system della macchina.

In **ambiente Windows** per utilizzare un socket, in linguaggio C, per la **famiglia AF_INET** si deve includere la libreria **WinSock2.h**.

Tra i **tipi di socket** possiamo individuerne **tre** che ci permettono **tre modalità di connessione:**

- **datagram socket**
- **stream socket**

} usati a livello applicativo

- **raw socket** utilizzato nello sviluppo di protocolli.

La connessione tramite socket

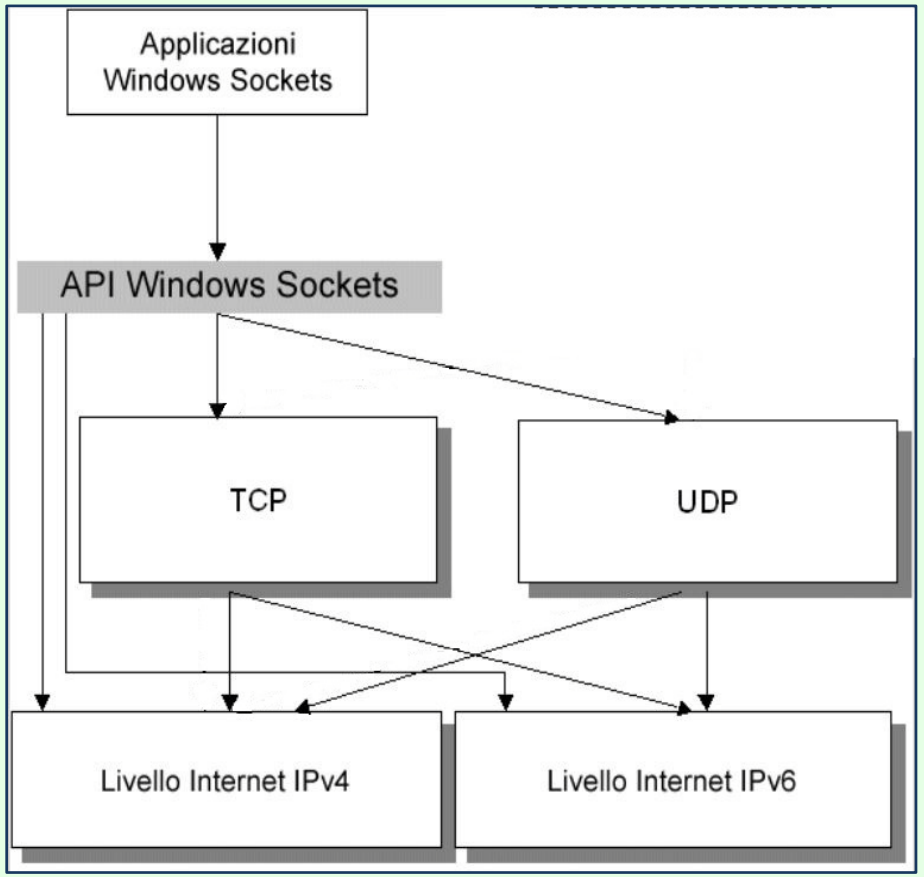
Ogni sistema operativo mette a disposizione nelle API degli strumenti per realizzare l'interfacciamento tra diversi protocolli: le **Socket API**.
Sono delle specifiche API che hanno origine con il sistema operativo UNIX BSD 4.2 (uscito nel 1983) della **BSD (Berkeley Software Distribution) UNIX**, sviluppate presso l'Università di *Berkeley* in California e disponibili oggi sia su piattaforme Linux che Windows (libreria WinSock).

I servizi forniti da **Windows Sockets** consentono alle applicazioni di:

- **utilizzare un indirizzo IP e una porta specifici**,
- **inizializzare** e **accettare una connessione** a un indirizzo IP e una porta di destinazione specifici,
- **inviare** e **ricevere** dati,
- **chiudere** una connessione.

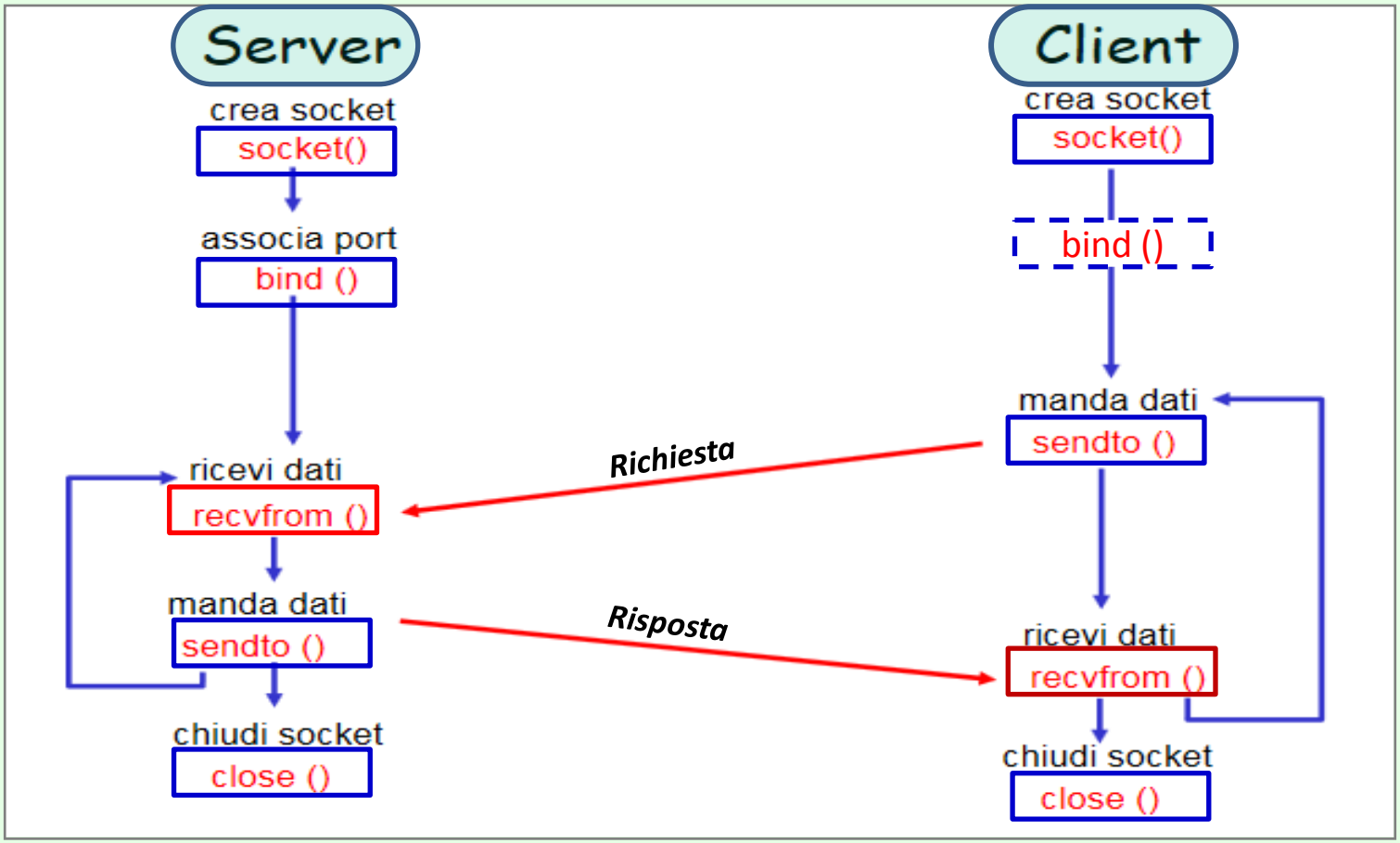
I tre tipi di socket sono:

- Un **socket di flusso** che fornisce tramite TCP un flusso di dati bidirezionale, affidabile, in sequenza e non duplicato.
- Un **socket di datagramma** che fornisce un flusso di dati bidirezionale tramite UDP.
- Un **socket di tipo raw** che consente ai protocolli di accedere direttamente al protocollo IP senza utilizzare TCP o UDP.



Comunicazioni UDP tramite Socket in ambiente Windows

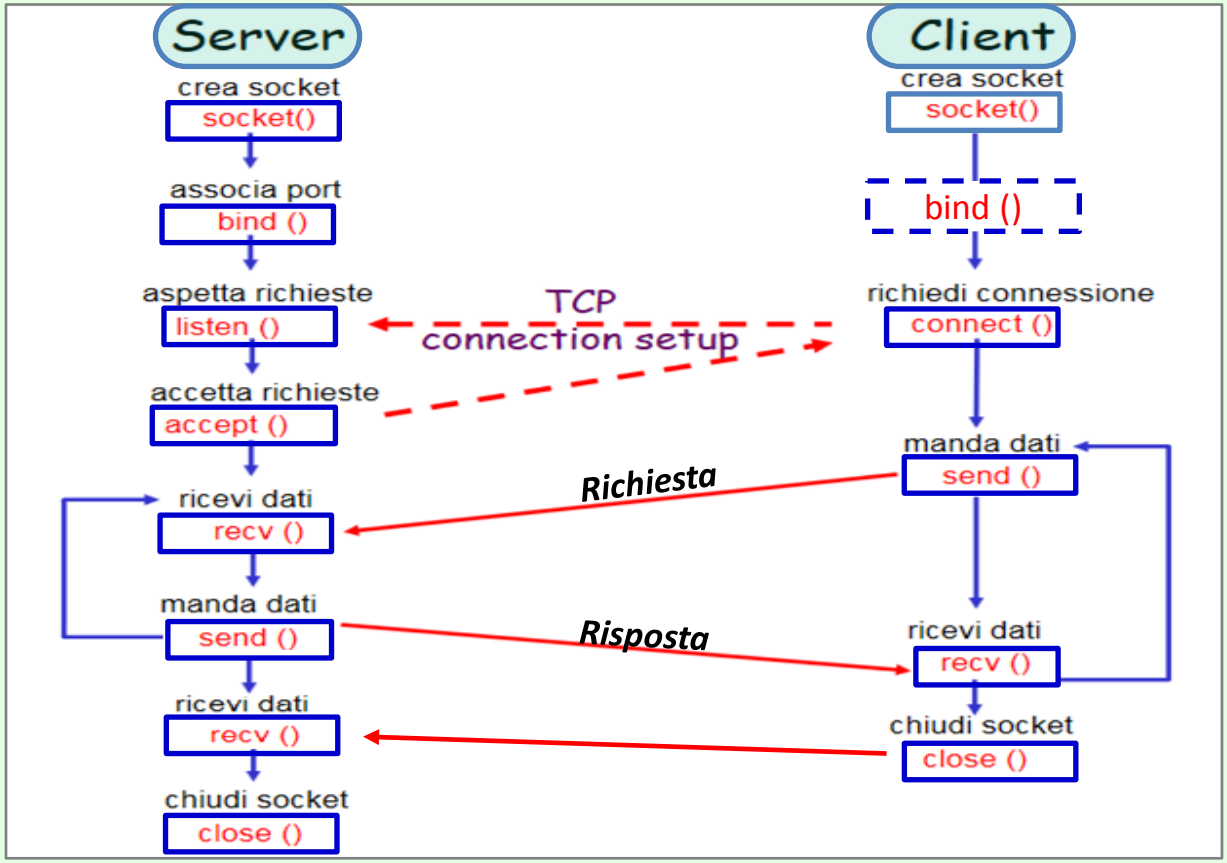
Schema logico della comunicazione UDP in **linguaggio C** mediante **Datagram socket**.



Con i **datagram socket** (**SOCK_DGRAM**) viene realizzata una comunicazine che permette di scambiare dati senza connessione (i messaggi contengono l'indirizzo di destinazione e di provenienza); non viene garantito l'ordine e neppure l'arrivo dei pacchetti.

Comunicazioni TCP tramite Socket in ambiente Windows

Schema logico della comunicazione TCP in **linguaggio C** mediante **stream socket**.



Con socket di flusso di byte (**SOCK_STREAM**) è necessario stabilire una connessione, il canale è bidirezionale e affidabile. Viene gestito a livello di trasporto con il protocollo TCP.

Le operazioni primitive di **connect**, lato client, e **listen**, lato server, consentono ai socket in questione di negoziare la connessione la quale, successivamente, dà luogo al vero e proprio trasferimento dati con le operazioni di **accept** e **recv**, lato server e **send**, lato client.

Al termine, entrambi i processi coinvolti chiudono la connessione, tramite una **close**.