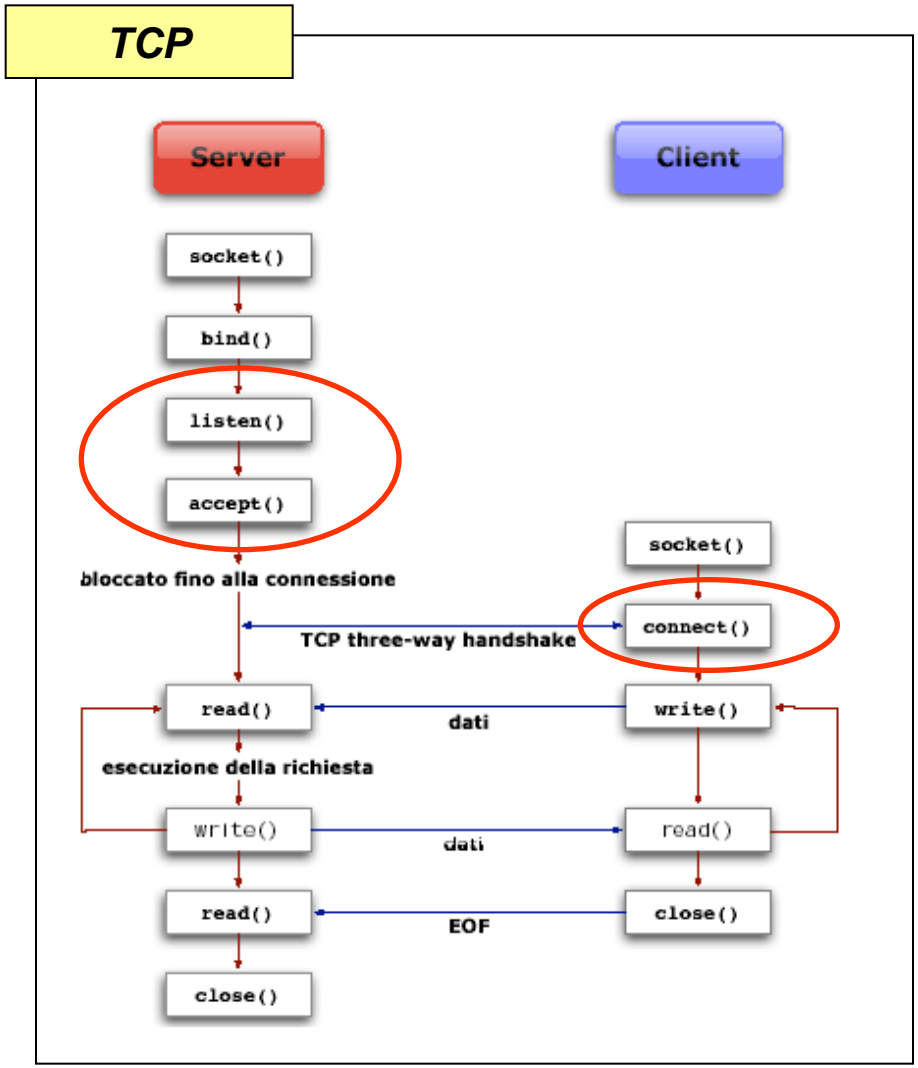
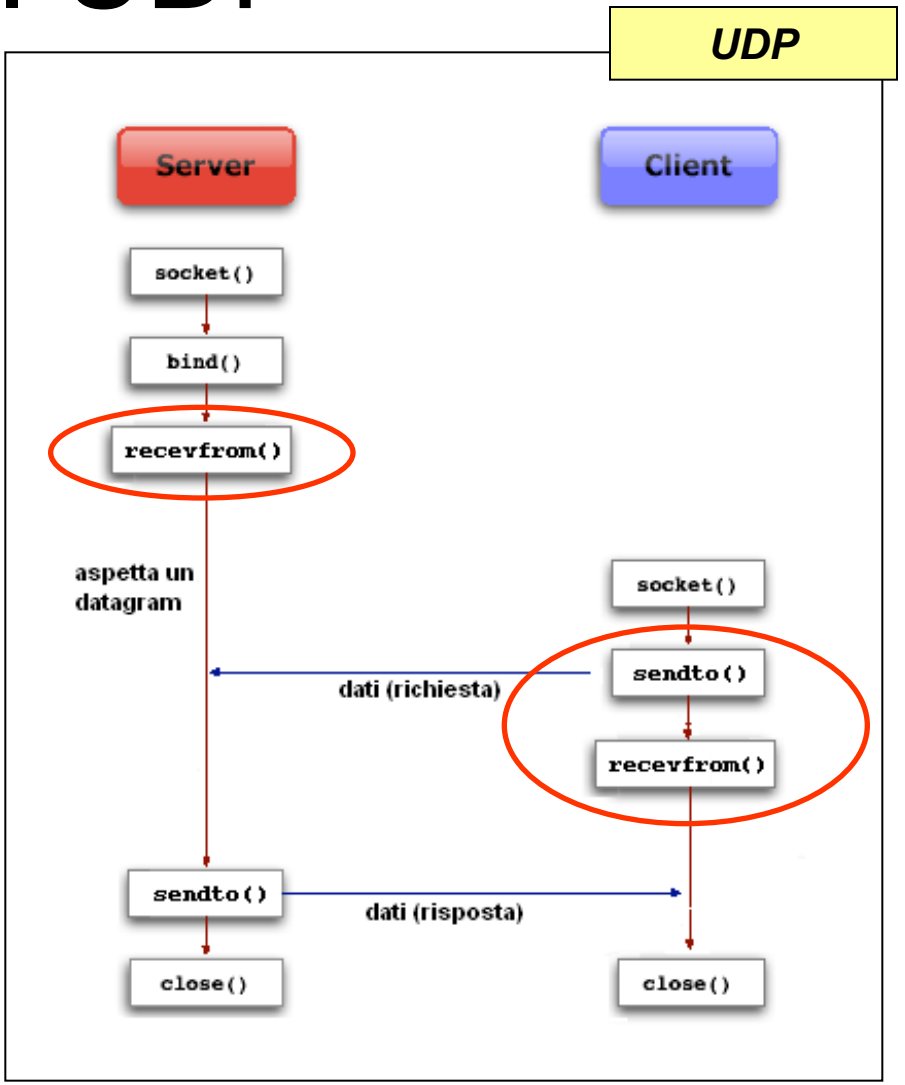
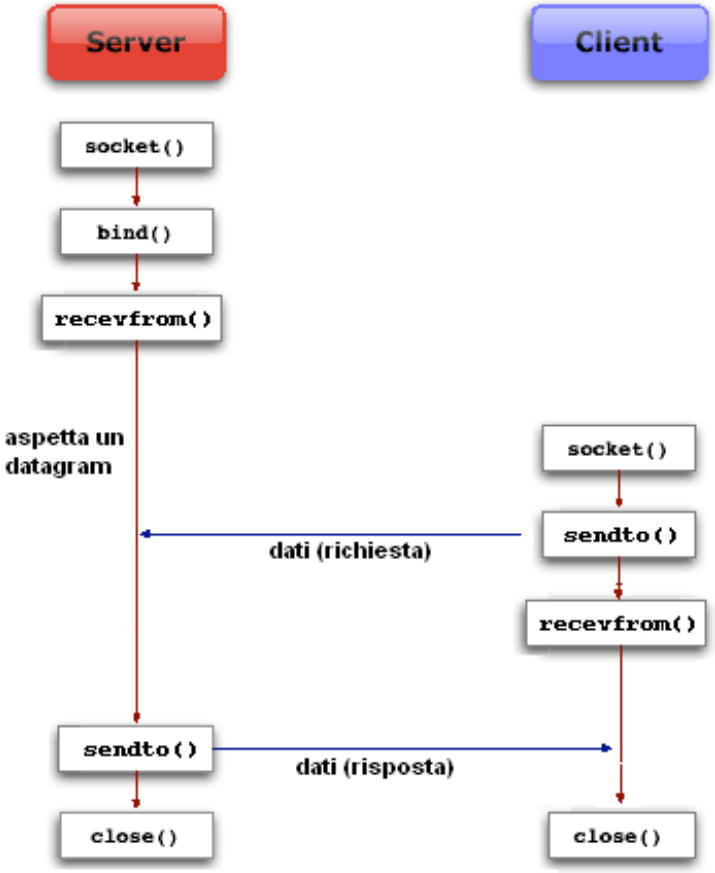
L’uso di Socket UDP

TCP vs. UDP


TCP
Server
Client
socket()
bind()
listen()
accept()
bloccato fino alla connessione
socket()
connect()
TCP three-way handshake
read()
dati
write()
esecuzione della richiesta
write()
dati
read()
read()
EOF
close()
close()


UDP
Server
Client
socket()
bind()
recvfrom()
aspetta un datagram
socket()
sendto()
dati (richiesta)
recvfrom()
sendto()
dati (risposta)
close()
close()

Interazione UDP Client/Server



Server

1. Creare un socket
2. Assegnare un local address al socket
3. Iterativamente:
   a. Inviare e ricevere dati
4. Chiudere il socket

Client

1. Creare un Socket
2. Inviare e ricevere dati
3. Chiudere il socket

Funzione socket()

- Crea una socket dedicata ad un fornitore di servizi specifico

```
int socket( int pf, int type, int protocol );
```

Famiglia di protocolli

(PF_INET: Internet Protocol Family)

Tipo di socket

Particolare protocollo da usare con la socket per la PF e il type indicati

(posto a 0 indica il protocollo di default per la coppia [pf, type])

Tipi di Socket

Tipo	Significato
SOCK_STREAM	**Fornisce una connessione sequenziale, affidabile e full-duplex.** ***Il protocollo TCP è basato su questo tipo di socket.***
SOCK_DGRAM	**Supporta i datagrammi (privo di connessione, messaggi inaffidabili di una lunghezza massima prefissata).** ***Il protocollo UDP è basato su questo tipo di socket.***

Funzione sendto()

Invia dati ad una specifica destinazione

```
int sendto( int s, const char* buf, int len, int flags, const struct sockaddr* to, unsigned int tolen);
```

Descrittore di una socket datagram

Buffer contenente i dati da trasmettere

Lunghezza in byte dei dati in buf

Flag, 0 = comportamento di default

Puntatore ad una struttura che contiene l'indirizzo della socket target

Lunghezza dei dati in to, in byte (valore!)

La funzione è usata per socket non orientate alla connessione per inviare datagram ad una specifica socket identificata dai parametri.

La funzione restituisce il numero di byte trasmessi in caso di successo.

Un codice di errore, altrimenti

Funzione recvfrom()

Riceve un datagram e memorizza l'indirizzo da cui i dati sono stati inviati

```
int recvfrom( int s, char* buf, int len, int flags, struct sockaddr* from, unsigned int* fromlen);
```

- **Descrittore di una socket datagram**
- **Buffer contenente i dati in ingresso**
- **Lunghezza in bytedei dati in buf**
- **Flag, 0 = comportamento di default**
- **Puntatore ad una struttura che contiene l'indirizzo della socket target**
- **Lunghezza dei dati in from, in byte (puntatore!)**

La funzione è usata per socket non orientate alla connessione.
L'indirizzo locale della socket deve essere noto
Per applicazioni Server, questo è fatto esplicitamente con la funzione bind()

Server UDP: un esempio di codice…

```
#if defined WIN32
#include <winsock.h>
#else
#define closesocket close
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#endif
#include <stdio.h>
#include <string.h> /* for memset() */

#define ECHOMAX 255
#define PORT 48000

void ErrorHandler(char *errorMessage) {
        printf(errorMessage);
}

void ClearWinSock() {
        #if defined WIN32
        WSACleanup();
        #endif
}
```

…server UDP: un esempio di codice…

```
int main() {
#if defined WIN32
WSADATA wsaData;
int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
if (iResult != 0) {
        printf ("error at WSASturtup\n");
        return EXIT_FAILURE;
}
#endif

int sock;
struct sockaddr_in echoServAddr;
struct sockaddr_in echoClntAddr;
unsigned int cliAddrLen;
char echoBuffer[ECHOMAX];
int recvMsgSize;

// CREAZIONE DELLA SOCKET
if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
        ErrorHandler("socket() failed");
```

…server UDP: un esempio di codice

```
// COSTRUZIONE DELL'INDIRIZZO DEL SERVER
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_port = htons(PORT);
echoServAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

// BIND DELLA SOCKET
if ((bind(sock, (struct sockaddr *)&echoServAddr, sizeof(echoServAddr))) < 0)
        ErrorHandler("bind() failed");

// RICEZIONE DELLA STRINGA ECHO DAL CLIENT
while(1) {
cliAddrLen = sizeof(echoClntAddr);
recvMsgSize = recvfrom(sock, echoBuffer, ECHOMAX, 0, (struct
                                    sockaddr*)&echoClntAddr, &cliAddrLen);

printf("Handling client %s\n", inet_ntoa(echoClntAddr.sin_addr));
printf("Received: %s\n", echoBuffer);

// RINVIA LA STRINGA ECHO AL CLIENT
if (sendto(sock, echoBuffer, recvMsgSize, 0, (struct sockaddr *)&echoClntAddr,
                                    sizeof(echoClntAddr)) != recvMsgSize)
        ErrorHandler("sendto() sent different number of bytes than expected");

}
```

Client UDP: un esempio di codice

```
#if defined WIN32
#include <winsock.h>
#else
#define closesocket close
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#endif
#include <stdio.h>
#include <string.h> /* for memset() */

#define ECHOMAX 255
#define PORT 48000

void ErrorHandler(char *errorMessage) {
printf(errorMessage);
}

void ClearWinSock() {
#if defined WIN32
WSACleanup();
#endif
}
```

…client UDP: un esempio di codice…

```
int main() {
#if defined WIN32
WSADATA wsaData;
int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
if (iResult != 0) {
printf ("error at WSASturtup\n");
return EXIT_FAILURE;
}
#endif

int sock;
struct sockaddr_in echoServAddr;
struct sockaddr_in fromAddr;
unsigned int fromSize;
char echoString[ECHOMAX];
char echoBuffer[ECHOMAX];
int echoStringLen;
int respStringLen;

printf("Inserisci la stringa echo da inviare al server\n");
scanf("%s", echoString);
if ((echoStringLen = strlen(echoString)) > ECHOMAX)
ErrorHandler("echo word too long");
```

…client UDP: un esempio di codice…

```
// CREAZIONE DELLA SOCKET
if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
        ErrorHandler("socket() failed");

// COSTRUZIONE DELL'INDIRIZZO DEL SERVER
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = PF_INET;
echoServAddr.sin_port = htons(PORT);
echoServAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

// INVIO DELLA STRINGA ECHO AL SERVER
if (sendto(sock, echoString, echoStringLen, 0, (struct
        sockaddr*)&echoServAddr, sizeof(echoServAddr)) !=
        echoStringLen)
                ErrorHandler("sendto() sent different number of bytes
than expected");

// RITORNO DELLA STRINGA ECHO
fromSize = sizeof(fromAddr);
respStringLen = recvfrom(sock, echoBuffer, ECHOMAX, 0, (struct
                                sockaddr*)&fromAddr, &fromSize);
```

…client UDP: un esempio di codice

```
if (echoServAddr.sin_addr.s_addr != fromAddr.sin_addr.s_addr)
{
fprintf(stderr, "Error: received a packet from unknown source.\n");
exit(EXIT_FAILURE);
}

echoBuffer[respStringLen] = '\0';  // inutile con memset
printf("Received: %s\n", echoBuffer);

closesocket(sock);
ClearWinSock();
system("pause");
return EXIT_SUCCESS;
}
```