

Esercitazione di Reti degli elaboratori

Prof.ssa Chiara Petrioli



SAPIENZA
UNIVERSITÀ DI ROMA

Laboratorio di C

Luca Iezzi

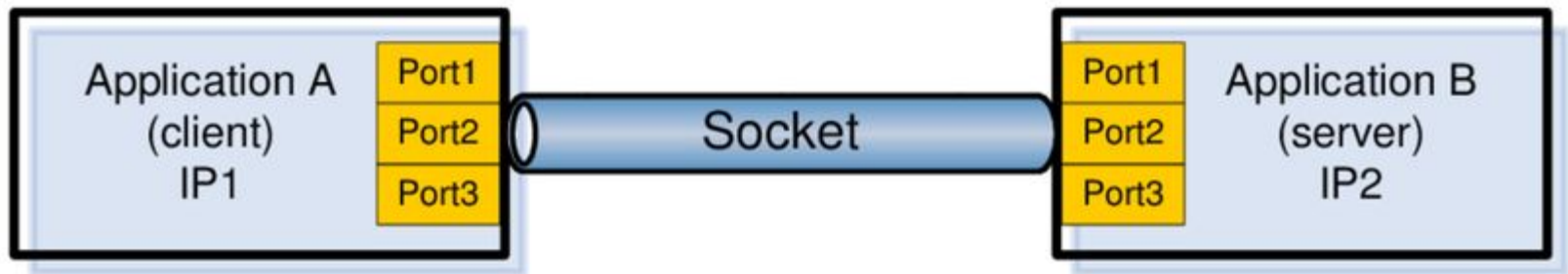
iezzi@diag.uniroma1.it

Socket



Interfacce fisiche usate per trasferire corrente elettrica

Socket nelle reti

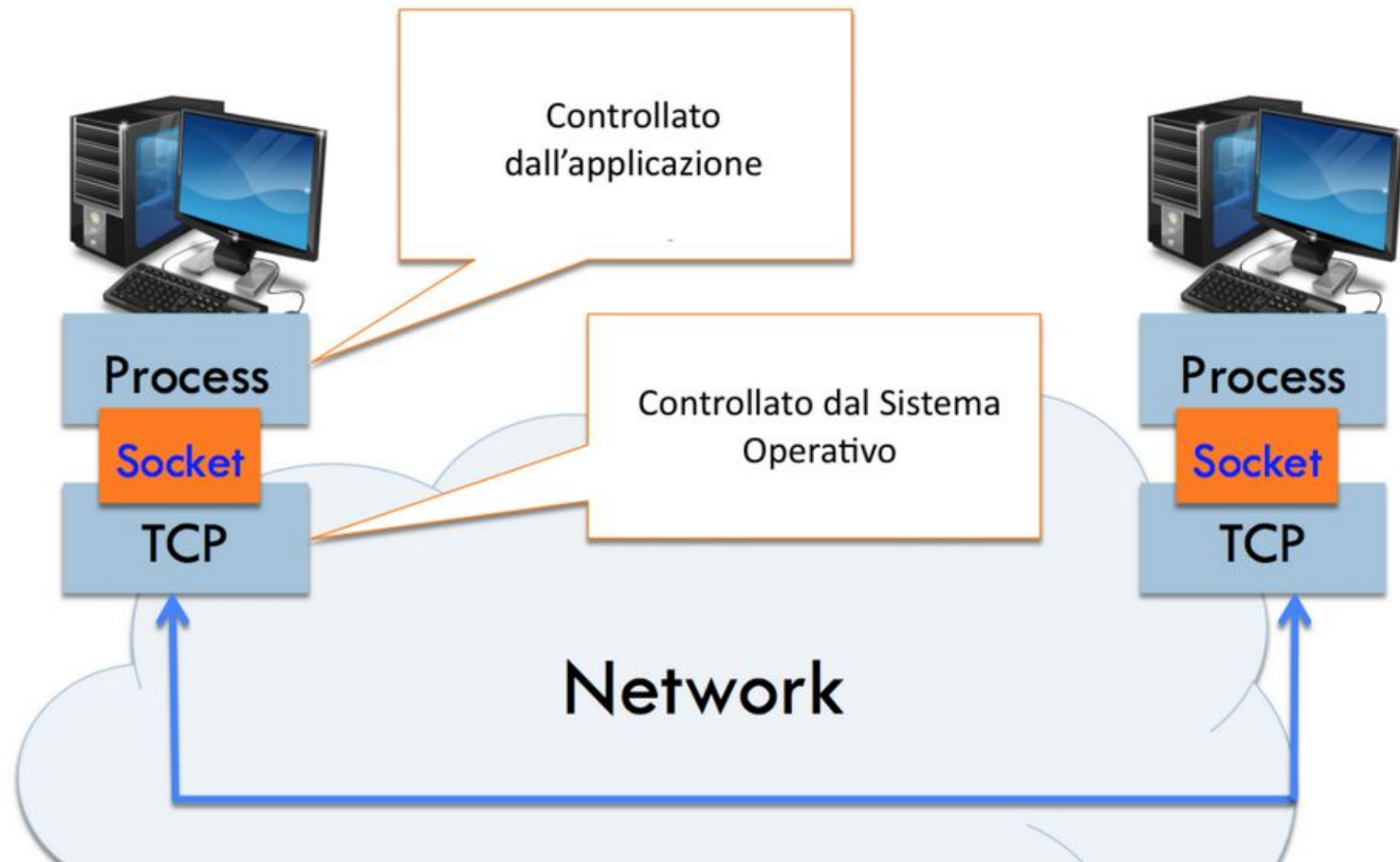


Socket API

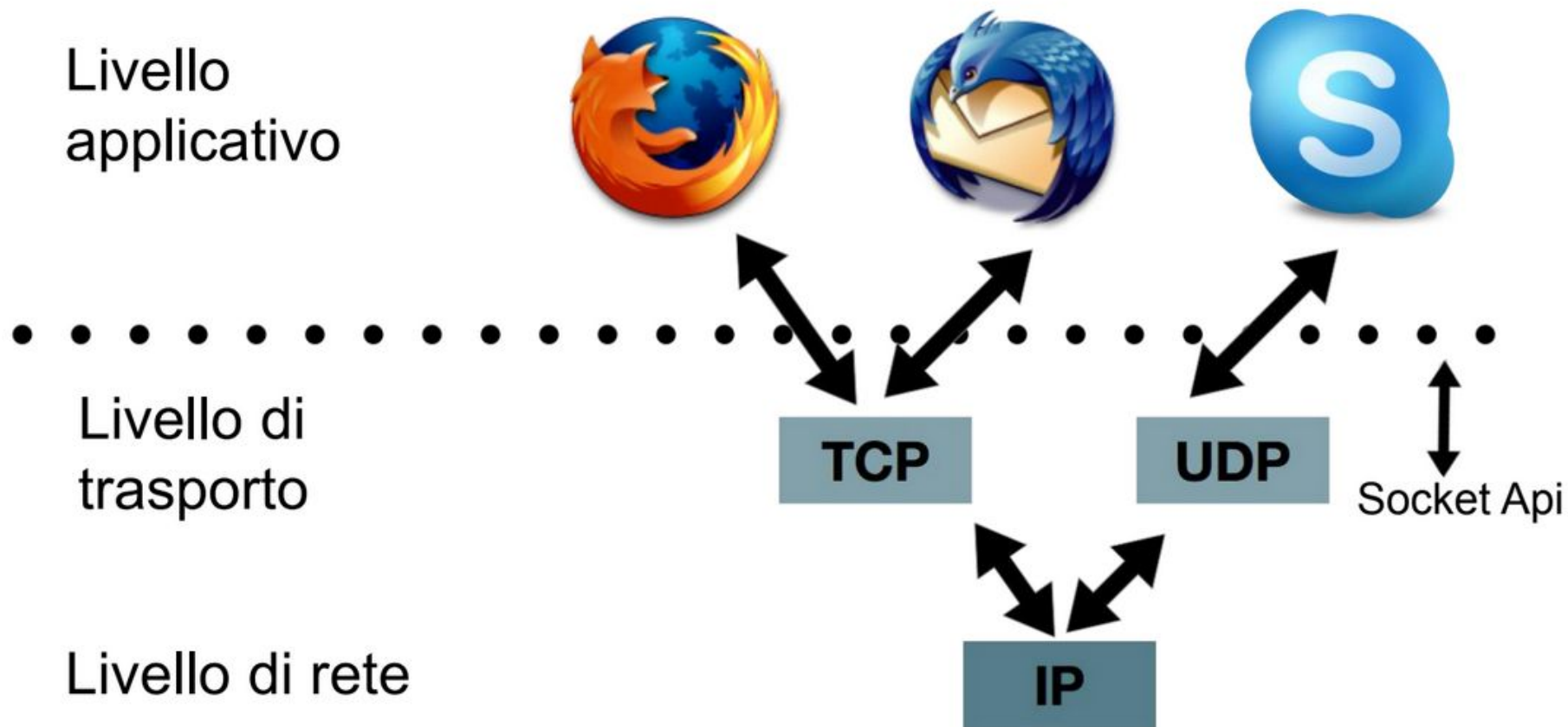
- Usate dai nodi della rete per comunicare e trasferire le informazioni da un nodo all'altro.
- Interfaccia tra il livello applicativo e la rete.
 - Usando le interfacce fornite nelle Application Programming Interfaces (API)
- Il livello applicativo può ricevere/mandare informazioni attraverso l'uso delle socket.
- Vengono create in coppia per stabilire una comunicazione

Socket  Porta

Socket API



Socket API



Socket API

- Noi useremo le socket API fornite dal linguaggio C.
- Le socket API contengono diverse **strutture dati**, **funzioni** e **system calls** che ci aiutano a scrivere un programma per comunicare nella rete in maniera efficiente.

Tipi di Socket

Datagram socket (aka UDP)

- Connection-less
- Consegna non affidabile (best-effort)
- Non bidirezionali (possono o mandare o ricevere)
- Usate da applicazioni VoIP (es. skype)

Stream socket (TCP)

- Connection-oriented
- Consegna affidabile
- Ordine garantito
- Bidirezionali
- Usate da: HTTP, FTP,...

Tipi di Socket - Esempio

- Quando si mandano due messaggi «ciao» e «come va?»
- TCP tratta i messaggi come un singolo byte stream

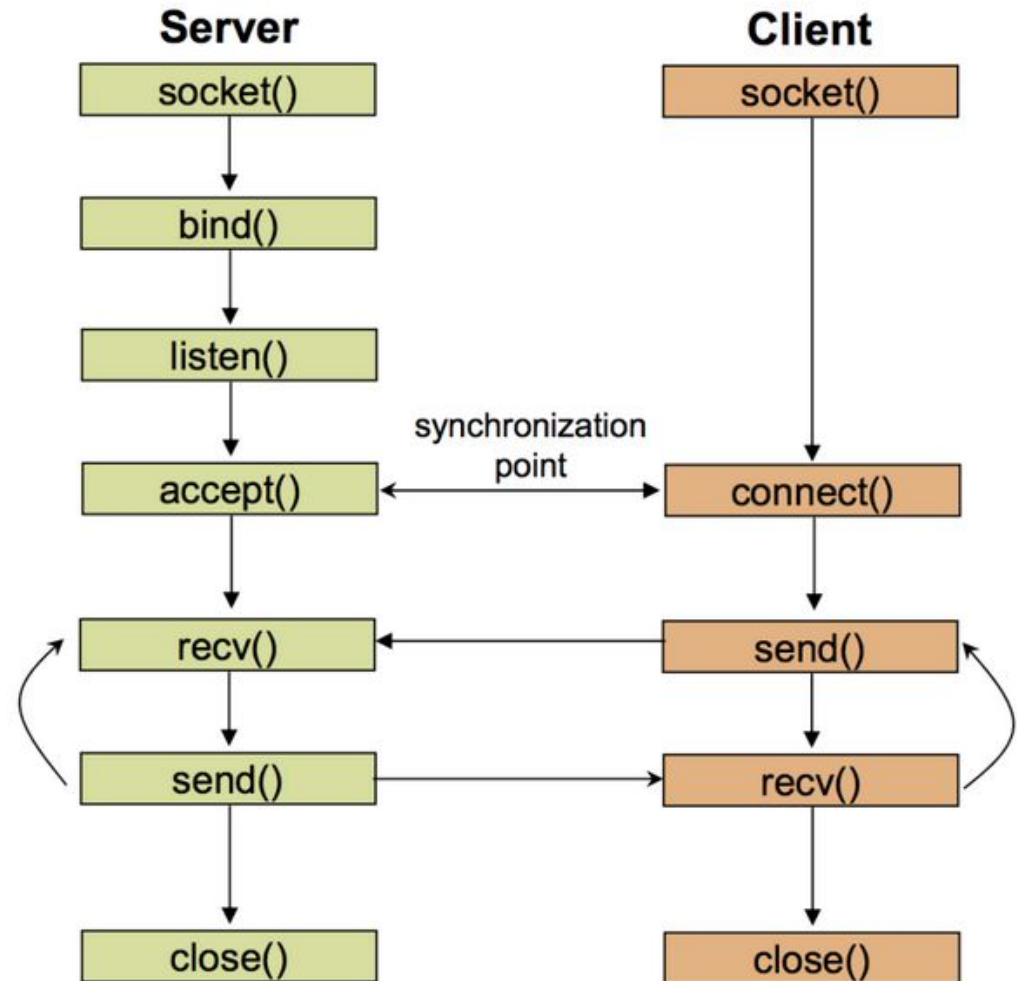


- UDP li tratta come singoli messaggi



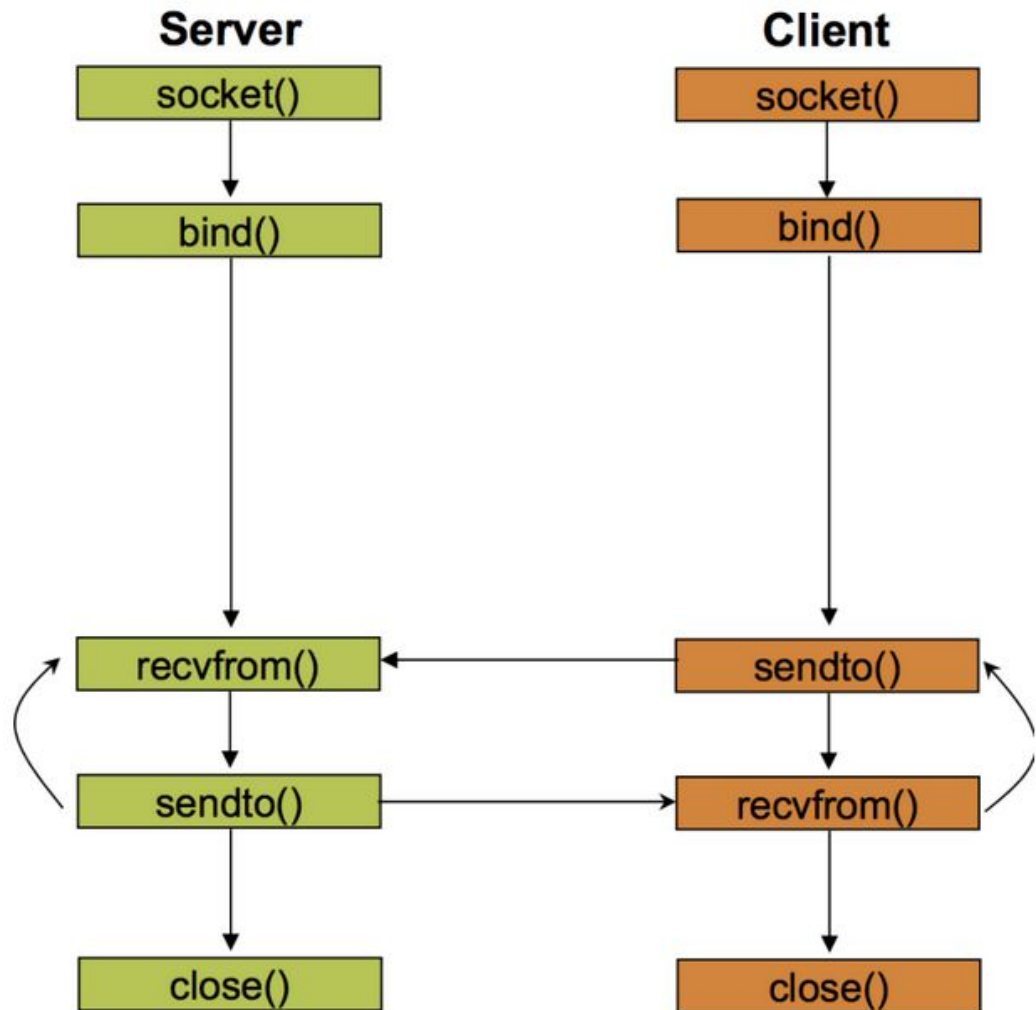
Stream socket TCP – ciclo di vita

- Crea una socket su entrambi i nodi coinvolti
- Si collega a una porta sul server
- Rimane in attesa di una comunicazione in ingresso (server)
- Il client si connette
- Il server accetta la connessione
- Trasmissione dati
- Chiusura socket



Datagram socket UDP – ciclo di vita

- Crea una socket su entrambi i nodi coinvolti
- Invia i dati a una porta sul server
- Trasmissione dati
- Chiusura socket



Socket in C - Server TCP

```
/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}
```

```

/* A simple server in the internet domain using TCP
   The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
              sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

Definizioni dei tipi di dati usati per la programmazione delle socket.

Definizione delle struct usate nelle socket.

Definizione delle costanti e strutture necessarie per gli indirizzi internet.


```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

```

```

void error(char *msg) {
    perror(msg);
    exit(1);
}

```

```

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

Questa funzione viene chiamata quando una chiamata di sistema fallisce. Viene stampato su stderr la stringa contenuta in msg.

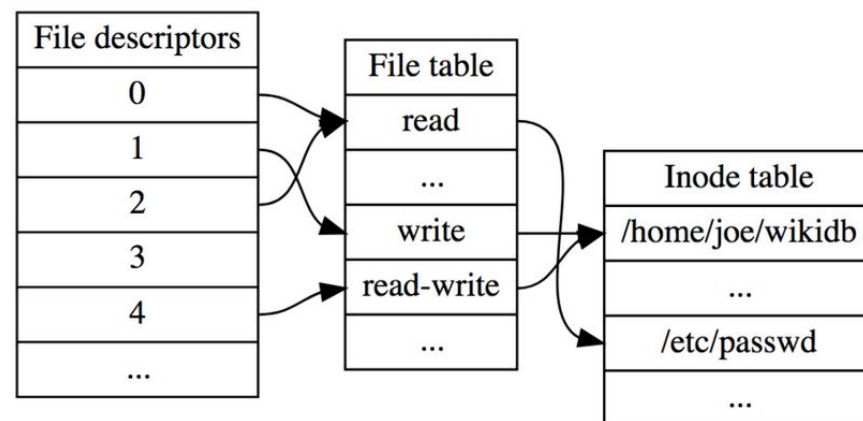
sockfd e **newsockfd** sono variabili che rappresentano il descrittore della socket.

- il descrittore della socket è un intero positivo che rappresenta la socket.
- Precisamente il nome tecnico è file descriptor
- **Questo perché le socket vengono trattati come file...**

Parentesi sui File Descriptor

- Un **file descriptor** è un intero che identifica univocamente un **file aperto**, le info su quel file e il modo in cui può essere acceduto (read/write/...)
- Quando un processo richiede accesso ad un file, il **Kernel** ritorna un **fd** che punta ad una **File Table** contenente informazioni di accesso sul file, **inode** del file, ...

Le applicazioni non hanno accesso diretto al file o alle tabelle inode, interagiscono con il kernel che gli restituisce un fd.



```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- **portno** memorizza il numero di porta sul quale il server accetterà le connessioni in entrata.
- **clilen** memorizza la dimensione dell'indirizzo del client
- **n** è il valore restituito dalle funzioni read() e write() e rappresenta *il numero di caratteri letti o scritti*.


```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

```

```

void error(char *msg) {
    perror(msg);
    exit(1);
}

```

```

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

I caratteri che vengono letti dal server vengono memorizzati nell'array buffer

sockaddr_in è una struttura che contiene gli indirizzi. La struttura è definita in **<netinet/in.h>** ed è definita come segue:

```

struct sockaddr_in {
    short sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

```

```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

```

```

void error(char *msg) {
    perror(msg);
    exit(1);
}

```

```

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- La system call **socket()** crea una nuova socket. La funzione prende in input tre parametri:
 - Il primo è il **dominio** degli indirizzi della socket. Esistono due possibili domini di indirizzi (dominio UNIX usato per comunicazione tra processi e dominio **INET** → quello che interessa a noi, che indica il **dominio degli indirizzi di rete**)
 - Il secondo indica il **tipo** della socket. SOCK_STREAM crea una **stream socket**. SOCK_DGRAM crea una **datagram socket**.
 - Il terzo indica il **protocollo** da usare (sempre 0 ad eccezione di alcune circostanze). Se è 0 il sistema operativo seleziona il protocollo di trasporto più appropriato per il tipo di socket. Quindi:
 - Se SOCK_STREAM usa TCP
 - Se SOCK_DGRAM usa UDP

```
/* A simple server in the internet domain using TCP
The port number is setted with #define */
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000
```

```
void error(char *msg) {
    perror(msg);
    exit(1);
}
```

```
int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}
```

- La funzione **bzero()** setta tutti i valori del buffer a zero. Prende due argomenti:
 - Il **puntatore al buffer** e la **dimensione**.
Quindi, questa funzione inizializza *serv_addr* a zero.
- Setta il **numero di porta** sul quale il server ascolta le trasmissioni


```

/* A simple server in the internet domain using TCP
The port number is setted with #define */

```

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

```

```

void error(char *msg) {
    perror(msg);
    exit(1);
}

```

```

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
              sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- La variabile **serv_addr** è una struttura di tipo *sockaddr_in*. Ha quattro campi.
 - sin_family**, rappresenta la famiglia di indirizzi. Noi lo setteremo sempre con AF_INET.
- serv_addr.sin_addr** è una struttura di tipo *in_addr* che contiene un singolo campo, **unsigned long s_addr** → contiene l'indirizzo IP dell'host. Lato server è l'indirizzo IP della macchina sulla quale sta girando il programma. Viene indicato con la costante simbolica INADDR_ANY che contiene tale indirizzo.
- Settiamo il campo **sin_port**, della struct che rappresenta il numero di porta in ascolto, con il valore che abbiamo fissato in precedenza

```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- La system call **bind()** lega la socket a un indirizzo. In questo caso all'indirizzo dell'host corrente e al numero di porta sul quale il server sta girando. La funzione prende in input tre argomenti:
 - la variabile che descrive la socket;
 - l'indirizzo
 - dimensione dell'indirizzo
- L'esecuzione di questa funzione può fallire per diverse ragioni. La più comune è che esiste già una socket in uso su quella porta nella stessa macchina.

```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- La system call **listen()** permette di ascoltare le connessioni *in ingresso* sulla socket. Ha due argomenti:
 - Il primo argomento è il **descrittore** della socket;
 - il secondo è la **dimensione** della coda del backlog
 - numero di connessioni che possono essere in attesa mentre il processo gestisce una determinata connessione.
 - impostato su 5, la dimensione massima consentita dalla maggior parte dei sistemi.
- Se il primo argomento è un descrittore di socket valido, questa chiamata non può fallire e quindi il codice non controlla gli errori.


```

/* A simple server in the internet domain using TCP
The port number is setted with #define */

```

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

```

```

void error(char *msg) {
    perror(msg);
    exit(1);
}

```

```

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- La system call **accept()** permette al server di accettare le connessioni in entrata da parte del client.
- Restituisce un **nuovo socket descriptor** e tutte le trasmissioni su questa connessione useranno il nuovo descrittore socket (*newsockfd*). Quindi:
 - **newsockfd**: è rappresenta il socket descriptor della socket sulla quale avviene la comunicazione attuale tra il server e un client
 - **sockfd**: rappresenta il socket descriptor della socket di ascolto per le connessioni
- La system call **accept()** ha tre argomenti:
 - Il primo è il **descrittore della socket** di ascolto.
 - Il secondo è un **puntatore all'indirizzo** del client.
 - Il terzo è la **dimensione della struttura** che contiene la descrizione del client
- Viene usata nei server che implementano le socket di tipo stream (connection-oriented)

```

/* A simple server in the internet domain using TCP
The port number is setted with #define */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define BUFF_SIZE 256
#define PORT_NUM 5000

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int sockfd, newsockfd, portno, clilen;
    char buffer[BUFF_SIZE];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = PORT_NUM;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,BUFF_SIZE);
    n = read(newsockfd,buffer,BUFF_SIZE-1);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}

```

- Se viene eseguito questo blocco di codice significa che un client si è connesso correttamente al server. Questa porzione di codice esegue:
 - Inizializzazione a 0 del buffer usato per ricevere i messaggi dal client
 - La funzione read() legge il contenuto nella socket e lo memorizza in buffer. Se la lettura fallisce restituisce un valore negativo
- Una volta che una connessione con una socket TCP è stata effettuata si può effettuare una comunicazione bidirezionale e il server può rispondere al client usando la funzione **write()**. La write prende come parametri:
 - Il descrittore della socket sulla quale avvengono le attuali trasmissioni;
 - Il messaggio di risposta;
 - La lunghezza del messaggio

Socket in C - Client TCP

```
int main()
{
    int sockfd, portno, n;

    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[BUFF_SIZE];

    portno = PORT_NUM;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(HOST_NAME);
    if (server == NULL) {
        fprintf(stderr, "ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR connecting");
    printf("Please enter the message: ");
    bzero(buffer, BUFF_SIZE);
    fgets(buffer, BUFF_SIZE-1, stdin);
    n = write(sockfd, buffer, strlen(buffer));
    if (n < 0)
        error("ERROR writing to socket");
    bzero(buffer, BUFF_SIZE);
    n = read(sockfd, buffer, BUFF_SIZE-1);
    if (n < 0)
        error("ERROR reading from socket");
    printf("%s\n", buffer);
    return 0;
}
```



```

int main()
{
    int sockfd, portno, n;

    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[BUFF_SIZE];

    portno = PORT_NUM;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(HOST_NAME);
    if (server == NULL) {
        fprintf(stderr, "ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
          (char *)&serv_addr.sin_addr.s_addr,
          server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR connecting");
    printf("Please enter the message: ");
    bzero(buffer, BUFF_SIZE);
    fgets(buffer, BUFF_SIZE-1, stdin);
    n = write(sockfd, buffer, strlen(buffer));
    if (n < 0)
        error("ERROR writing to socket");
    bzero(buffer, BUFF_SIZE);
    n = read(sockfd, buffer, BUFF_SIZE-1);
    if (n < 0)
        error("ERROR reading from socket");
    printf("%s\n", buffer);
    return 0;
}

```

- La variabile **serv_addr** contiene l'indirizzo del server al quale ci si vuole connettere.
- La variabile **server**, di tipo **hostnet** invece contiene le informazioni necessarie per la **definizione dell'host** (ad esempio: nome, indirizzo)
- La funzione **gethostbyname()** prende come argomento il *nome dell'host nella rete* (ad esempio mario@sapienza.it oppure, se la socket gira all'interno della stessa macchina *localhost*).
- La funzione restituisce un puntatore di tipo **hostnet** che contiene le **informazioni dell'host (server)**.

```

int main()
{
    int sockfd, portno, n;

    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[BUFF_SIZE];

    portno = PORT_NUM;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(HOST_NAME);
    if (server == NULL) {
        fprintf(stderr, "ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");
    printf("Please enter the message: ");
    bzero(buffer,BUFF_SIZE);
    fgets(buffer,BUFF_SIZE-1,stdin);
    n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)
        error("ERROR writing to socket");
    bzero(buffer,BUFF_SIZE);
    n = read(sockfd,buffer,BUFF_SIZE-1);
    if (n < 0)
        error("ERROR reading from socket");
    printf("%s\n",buffer);
    return 0;
}

```

- Questa porzione di codice è molto simile al server. Tuttavia, visto che il campo **server->h_addr** è un array di caratteri usiamo la **bcopy(char* s1, char *s2, int num_byte)**.
- *(Reminder)* La **bcopy** copia **num_byte** bytes da **s2** a **s1**.
- La funzione **connect()** viene chiamata dal client per stabilire una connessione al server. Sono necessari tre argomenti:
 - il **descrittore** della socket;
 - l'**indirizzo dell'host** a cui si desidera connettersi (incluso il numero di porta)
 - la **dimensione di questo indirizzo**.
- Questa funzione restituisce 0 in caso di successo e -1 se fallisce.


```

int main()
{
    int sockfd, portno, n;

    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[BUFF_SIZE];

    portno = PORT_NUM;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(HOST_NAME);
    if (server == NULL) {
        fprintf(stderr, "ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR connecting");
    printf("Please enter the message: ");
    bzero(buffer, BUFF_SIZE);
    fgets(buffer, BUFF_SIZE-1, stdin);
    n = write(sockfd, buffer, strlen(buffer));
    if (n < 0)
        error("ERROR writing to socket");
    bzero(buffer, BUFF_SIZE);
    n = read(sockfd, buffer, BUFF_SIZE-1);
    if (n < 0)
        error("ERROR reading from socket");
    printf("%s\n", buffer);
    return 0;
}

```

- La porzione di codice prima **chiede all'utente di inserire un messaggio**, usa **fgets** per leggere il messaggio, **scrive nella socket**, **legge la risposta del server** e la **stampa a video**.

Socket in C - Esercizi

Esercizio 1

Scrivere un'applicazione client/server in cui:

- Il client, in un ciclo infinito:
 - Legge una stringa da standard input
 - Invia al server la stringa.
- Il server visualizza:
 - risponde al client mandando un messaggio di conferma
- Client e server terminano quanto l'utente digita la stringa "exit"
- **N.B usare le socket stream**

Socket in C - Esercizi

Esercizio 2

Scrivere un'applicazione client/server in cui:

- Il client, in un ciclo infinito:
 - Legge una stringa da standard input
 - Invia al server la stringa.
- Il server visualizza:
 - risponde al client mandando la stringa ricevuta ma invertita
- Client e server terminano quanto l'utente digita la stringa "exit"
- **N.B usare le socket stream**

Socket in C - Esercizi

Esercizio 3

Scrivere un'applicazione client/server in cui:

- Il client, in un ciclo infinito:
 - Legge una stringa da standard input
 - Invia al server la stringa.
- Il server visualizza:
 - risponde al client mandando la stringa ricevuta ma duplicata
- Esempio:
 - se il client manda *ciao* il server risponde con *ciaociao*
- Client e server terminano quanto l'utente digita la stringa "exit"
- **N.B usare le socket stream**

Socket in C - Esercizi

Esercizio 4

Scrivere un'applicazione client/server in cui:

- Il client, in un ciclo infinito:
 - Legge una stringa da standard input
 - Invia al server la stringa.
- Il server visualizza:
 - risponde al client mandando la stringa ricevuta ma con i caratteri maiuscoli
- Esempio:
 - se il client manda *ciao* il server risponde con *CIAO*
- Client e server terminano quanto l'utente digita la stringa "exit"
- **N.B usare le socket stream**
- *indizio:* cercate la funzione toupper...