Commento del codice: TCP Client/Server in C

Laboratorio di Reti e Sistemi Distribuiti @ UniME

February 27, 2025

1 Introduzione

Questo documento ad integrazione delle slide, spiega nel dettaglio il funzionamento del client e del server tcp.

2 Codice del Server TCP

2.1 Inclusione delle Librerie

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
```

Queste sono le librerie necessarie per il funzionamento del server:

- `stdio.h`: Fornisce funzioni di input/output.
- `stdlib.h`: Fornisce funzioni di utilità generale come `exit`.
- `string.h`: Fornisce funzioni per la manipolazione delle stringhe.
- `unistd.h`: Fornisce funzioni di sistema come `read` e `write`.
- `arpa/inet.h`: Fornisce funzioni per la gestione degli indirizzi di rete.

2.2 Definizione della Porta

```c
#define PORT 8080
```

La porta su cui il server ascolta le connessioni in entrata è definita come `8080`.

2.3 Funzione Principale

```
int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";
```

La funzione principale del server. Qui vengono dichiarate le variabili:

- `server_fd`: File descriptor del socket del server.
- `new_socket`: File descriptor del socket per la connessione accettata.
- `address`: Struttura che contiene l'indirizzo del server.
- `addrlen`: Lunghezza dell'indirizzo.
- `buffer`: Buffer per memorizzare i dati ricevuti.
- `hello`: Messaggio da inviare al client.

2.4 Creazione del Socket

```
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
```

Crea un socket TCP (`SOCK_STREAM`) utilizzando il dominio IPv4 (`AF_INET`). Se la creazione fallisce, viene stampato un messaggio di errore e il programma termina.

2.5 Configurazione dell'Indirizzo

```
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);
```

Configura l'indirizzo del server:

- `sin_family`: Imposta il dominio di indirizzamento a IPv4.
- `sin_addr.s_addr`: Imposta l'indirizzo IP a `INADDR_ANY`, che significa che il server accetterà connessioni su qualsiasi interfaccia di rete.
- `sin_port`: Imposta la porta su cui il server ascolta, convertendola in formato di rete con `htons`.

2.6 Binding del Socket

```
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address
    )) < 0) {
        perror("bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
```

Associa il socket all'indirizzo e alla porta specificati. Se il binding fallisce, viene stampato un messaggio di errore, il socket viene chiuso e il programma termina.

2.7 Ascolto delle Connessioni

```
    if (listen(server_fd, 3) < 0) {
        perror("listen");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
```

Mette il socket in ascolto per connessioni in entrata, con una coda massima di 3 connessioni pendenti. Se fallisce, viene stampato un messaggio di errore, il socket viene chiuso e il programma termina.

2.8 Accettazione di una Connessione

```
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address
    , (socklen_t*)&addrlen)) < 0) {
        perror("accept");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
```

Accetta una connessione in entrata. Se l'accettazione fallisce, viene stampato un messaggio di errore, il socket viene chiuso e il programma termina.

2.9 Lettura del Messaggio dal Client

```
    read(new_socket, buffer, 1024);
    printf("%s\n", buffer);
```

Legge il messaggio inviato dal client e lo stampa sulla console.

2.10 Invio di una Risposta al Client

```
    send(new_socket, hello, strlen(hello), 0);
    printf("Hello message sent\n");
```

Invia un messaggio di risposta al client.

2.11 Chiusura dei Socket

```
    close(new_socket);
    close(server_fd);
    return 0;
}
```

Chiude i socket e termina il programma.

3 Codice del Client TCP

3.1 Inclusione delle Librerie

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
```

Queste sono le librerie necessarie per il funzionamento del client, simili a quelle del server.

3.2 Definizione della Porta

```
#define PORT 8080
```

La porta su cui il client si connette al server.

3.3 Funzione Principale

```
int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from client";
    char buffer[1024] = {0};
```

La funzione principale del client. Qui vengono dichiarate le variabili:

- `sock`: File descriptor del socket del client.
- `serv_addr`: Struttura che contiene l'indirizzo del server.
- `hello`: Messaggio da inviare al server.
- `buffer`: Buffer per memorizzare i dati ricevuti.

3.4 Creazione del Socket

```
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }
```

Crea un socket TCP (`SOCK_STREAM`) utilizzando il dominio IPv4 (`AF_INET`). Se la creazione fallisce, viene stampato un messaggio di errore e il programma termina.

3.5 Configurazione dell'Indirizzo

```
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
```

Configura l'indirizzo del server:

- `sin_family`: Imposta il dominio di indirizzamento a IPv4.
- `sin_port`: Imposta la porta su cui il client si connette, convertendola in formato di rete con `htons`.

3.6 Conversione dell'Indirizzo IP

```
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }
```

Converte l'indirizzo IP da formato stringa a formato binario. Se la conversione fallisce, viene stampato un messaggio di errore e il programma termina.

3.7 Connessione al Server

```
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(
    serv_addr)) < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }
```

Tenta di connettersi al server. Se la connessione fallisce, viene stampato un messaggio di errore e il programma termina.

3.8 Invio di un Messaggio al Server

```
    send(sock, hello, strlen(hello), 0);
    printf("Hello message sent\n");
```

Invia un messaggio al server.

3.9 Lettura della Risposta dal Server

```
    read(sock, buffer, 1024);
    printf("%s\n", buffer);
```

Legge la risposta dal server e la stampa sulla console.

3.10 Chiusura del Socket

```
    close(sock);
    return 0;
}
```

Chiude il socket e termina il programma.