

# Trabalho Prático Individual 02

Desenvolvimento de Sistemas Distribuídos Simples

Multithreading com Conexão UDP

Emanuelly Carvalho

2021039905

BELO HORIZONTE

2024

# Introdução

Este projeto consiste no desenvolvimento de um sistema em formato de aplicação console que simula o funcionamento básico de aplicativos de streaming, com um servidor capaz de lidar com múltiplas conexões de clientes utilizando multithreading. A comunicação entre cliente e servidor é realizada por meio de uma conexão UDP, onde o servidor envia frases aleatórias de filmes para os clientes conectados a cada 3 segundos. No terminal do servidor, o número de clientes conectados é atualizado a cada 4 segundos.

## Sumário

<b>Introdução</b>	<b>2</b>
<b>Sumário</b>	<b>2</b>
<b>Descrição</b>	<b>3</b>
Servidor	3
Bind do Socket	3
Recebimento de Dados dos Clientes	4
Envio de Dados para os Clientes	4
Gerenciamento de Múltiplos Clientes	4
Cliente	5
Criação do Socket UDP	5
Envio de Dados para o Servidor	5
Recebimento de Dados do Servidor	5
<b>Demonstração</b>	<b>6</b>
<b>Instruções de Uso</b>	<b>7</b>


## Descrição

Abaixo estão trechos importantes do código que lidam com a conexão nos dois arquivos, `client.c` e `server.c`, explicando o que fazem. Esses trechos mostram como a comunicação é estabelecida entre o servidor e os clientes usando sockets UDP.

## Servidor

### Criação do Socket UDP


O servidor cria um socket UDP usando `socket()`, especificando `AF_INET` para IPv4 ou `AF_INET6` para IPv6. O socket é configurado para comunicação UDP usando `SOCK_DGRAM`.



```
s = socket(AF_INET, SOCK_DGRAM, 0); // Cria o socket UDP para IPv4
s = socket(AF_INET6, SOCK_DGRAM, 0); // Cria o socket UDP para IPv6
```

### Bind do Socket

O servidor associa o socket a um endereço IP e porta específicos usando `bind()`. Isso permite que o servidor receba dados dos clientes.



```
bind(s, (struct sockaddr *)&addr4, addr_len); // IPv4
bind(s, (struct sockaddr *)&addr6, addr_len); // IPv6
```

### Recebimento de Dados dos Clientes

O servidor recebe dados dos clientes usando `recvfrom()`. Essa função bloqueia até que dados sejam recebidos.



```
recvfrom(s, buf, BUFSZ, 0, (struct sockaddr *)&client_addr, &client_addr_len);
```

### Envio de Dados para os Clientes

O servidor envia dados para os clientes usando `sendto()`. Neste caso, envia frases dos filmes selecionados pelos clientes.



```
sendto(client->socket, buf, strlen(buf) + 1, 0, (struct sockaddr *)&client_addr, client_addr_len);
```

### Gerenciamento de Múltiplos Clientes

O servidor utiliza threads para lidar com múltiplos clientes de forma concorrente. Cada nova conexão de cliente é tratada em uma nova thread.



```
pthread_create(&tid, NULL, client_handler, client);
```

Cria uma nova thread para gerenciar a conexão do cliente

## Cliente

### Criação do Socket UDP

O cliente cria um socket UDP da mesma maneira que o servidor, usando

1. `socket(AF_INET, SOCK_DGRAM, 0)` para IPv4
2. `socket(AF_INET6, SOCK_DGRAM, 0)` para IPv6



```
s = socket(AF_INET, SOCK_DGRAM, 0); // Cria o socket UDP para IPv4  
s = socket(AF_INET6, SOCK_DGRAM, 0); // Cria o socket UDP para IPv6
```

### Envio de Dados para o Servidor

O cliente envia dados para o servidor usando `sendto()`. Neste caso, envia a escolha do filme para o servidor.



```
sendto(s, buf, strlen(buf) + 1, 0, (struct sockaddr *)&server_addr, server_addr_len);
```

### Recebimento de Dados do Servidor

O cliente recebe dados do servidor usando `recvfrom()`. Essa função bloqueia até que dados sejam recebidos.



```
recvfrom(s, buf, BUFSZ, 0, NULL, NULL);
```

## Demonstração

Em uma aba do terminal, inicio a conexão na porta escolhida

```
evsrc@EMANUELLY-PC:~/UFMG/Redes/tp02/bin$ ./server ipv4 50501
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
```

A cada 4 segundos, o terminal avisa a quantidade de clientes conectados. No segundos iniciais, nenhum cliente se conectou.

Em outra aba, dividi a tela para que fosse possível ver simultaneamente dois clientes conectados na mesma porta recebendo a cada 3 segundos frases dos filmes escolhidos:

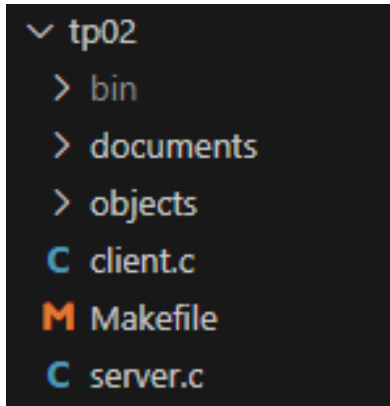
<pre>evsrc@EMANUELLY-PC:~/UFMG/Redes/tp02/bin\$ ./client ipv4 12 7.0.0.1 50501 Menu de filmes: 1) Senhor dos Anéis 2) O Poderoso Chefão 3) Clube da Luta Escolha um filme (1-3): 1 Frase: Um anel para todos governar Frase: Na terra de Mordor onde as sombras se deitam Frase: Não é o que temos, mas o que fazemos com o que temo s Frase: Não há mal que sempre dure Frase: O mundo está mudando, senhor Frodo</pre>	<pre>evsrc@EMANUELLY-PC:~/UFMG/Redes/tp02/bin\$ ./client ipv4 12 7.0.0.1 50501 Menu de filmes: 1) Senhor dos Anéis 2) O Poderoso Chefão 3) Clube da Luta Escolha um filme (1-3): 2 Frase: Vou fazer uma oferta que ele não pode recusar Frase: Mantenha seus amigos por perto e seus inimigos mais perto ainda  </pre>
--	--

Enquanto isso, na aba do servidor, o número de clientes conectados foi atualizado de acordo com as conexões feitas e finalizadas, vide *screenshot* ao lado.

```
evsrc@EMANUELLY-PC:~/UFMG/Redes/tp02/bin$
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 0
Clientes conectados: 1
Clientes conectados: 2
Clientes conectados: 2
Clientes conectados: 2
Clientes conectados: 1
Clientes conectados: 0
Clientes conectados: 0
^C
```

## Instruções de Uso

Após descompactar a pasta do envio, basta acessá-la no terminal e rodar o comando make, sem nenhum parâmetro adicional. Ao fazer isso, você verá uma estrutura de pastas semelhante a esta:



Para rodar o código, existem as seguintes opções

- Na pasta raiz
  - `./bin/server ipv4 50501`
  - `./bin/server ipv6 50501`
  - `./bin/client 127.0.0.1 50501`
  - `./bin/server ::1 50501`
- Na pasta bin
  - `./client ipv4 50501`
  - `./client ipv6 50501`
  - `./client 127.0.0.1 50501`
  - `./client ::1 50501`

Lembrando que, para haver a conexão bem sucedida, é fundamental que a porta seja a mesma para ambas as solicitações (cliente e servidor).