



Universidade Federal  
de São João del-Rei

# TRABALHO PRÁTICO 3

Lucas Gonçalves Nojiri  
Arthur Antunes Santos Silva

Neste trabalho iremos implementar uma transmissão unidirecional utilizando comunicação do tipo requisição resposta sobre o protocolo TCP, baseado no código em C para a disciplina Redes do curso de Ciência da Computação da Universidade Federal de São João del Rei.

São João del Rei  
Dezembro de 2022

# Sumário

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Metodologia . . . . .	2
<b>2</b>	<b>Testes</b>	<b>2</b>
<b>3</b>	<b>Descrição dos algoritmos e estruturas de dados</b>	<b>4</b>
3.1	Servidor.c . . . . .	4
3.1.1	void write_file(int sockfd) . . . . .	4
3.2	Cliente.c . . . . .	5
3.2.1	long getIORead() . . . . .	5
3.2.2	float time_diff(struct timeval *start, struct timeval *end) . . . . .	5
3.2.3	void send_file(FILE *fp, int sockfd) . . . . .	5
<b>4</b>	<b>Conclusão</b>	<b>6</b>
<b>5</b>	<b>Referências</b>	<b>7</b>

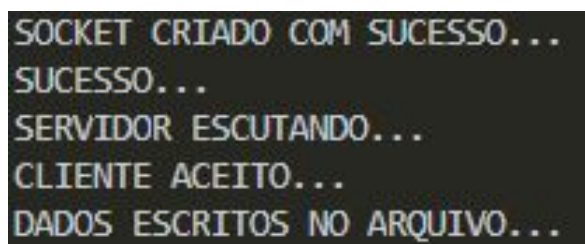
# 1 Introdução

Neste trabalho vamos implementar um cliente que deve se conectar ao servidor, enviar um string com o nome do arquivo desejado, receber o arquivo um buffer de cada vez e salvar os dados no disco à medida que eles chegam. Quando não houver mais bytes para ler, o cliente fecha a conexão e o arquivo, e gera uma linha com os dados da execução..

## 1.1 Metodologia

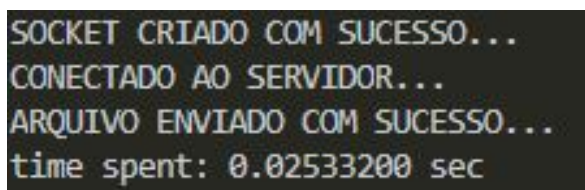
O cliente deve se conectar ao servidor, enviar um string com o nome do arquivo desejado, receber o arquivo um buffer de cada vez e salvar os dados no disco à medida que eles chegam. Quando não houver mais bytes para ler, o cliente fecha a conexão e o arquivo, e gera uma linha com os dados da execução. O servidor por sua vez deve operar de forma complementar. Durante a análise da execução foram testados diversas vezes em tamanhos diferentes de arquivo. Os tamanhos foram: >1MB, 1MB, 2MB, 3MB, 5MB e 10MB.

## 2 Testes



```
SOCKET CRIADO COM SUCESSO...  
SUCESSO...  
SERVIDOR ESCUTANDO...  
CLIENTE ACEITO...  
DADOS ESCRITOS NO ARQUIVO...
```

Figura 1: Servifor escutando



```
SOCKET CRIADO COM SUCESSO...  
CONECTADO AO SERVIDOR...  
ARQUIVO ENVIADO COM SUCESSO...  
time spent: 0.02533200 sec
```

Figura 2: arquivo com menos de 1 mb

---

```
SOCKET CRIADO COM SUCESSO...  
CONECTADO AO SERVIDOR...  
ARQUIVO ENVIADO COM SUCESSO...  
time spent: 0.00388700 sec
```

Figura 3: Arquivo com 1 mb

```
SOCKET CRIADO COM SUCESSO...  
CONECTADO AO SERVIDOR...  
ARQUIVO ENVIADO COM SUCESSO...  
Bytes read from disk: 0  
time spent: 0.00464600 sec
```

Figura 4: Arquivo com 2 mb

```
SOCKET CRIADO COM SUCESSO...  
CONECTADO AO SERVIDOR...  
ARQUIVO ENVIADO COM SUCESSO...  
Bytes read from disk: 0  
time spent: 0.00721200 sec
```

Figura 5: Arquivo com 3 mb

---

## 3 Descrição dos algoritmos e estruturas de dados

### 3.1 Servidor.c

Esta seção é responsável por armazenar as solicitações de conexão e configurações da estrutura do servidor. Nele são administradas as informações sobre os arquivos dos sockets criados, realiza a transmissão unidirecional com requisição resposta do protocolo TCP, vincula o socket recém criado para determinado IP e verificação.

Com isso o servidor está pronto para realizar a verificação e aceita o pacote de dados do cliente. O servidor recebe o arquivo que foi mandado pelo cliente e escreve nesse arquivo. Após isso o servidor fecha a conexão, o cliente desconecta e o programa se finaliza.

#### 3.1.1 void write\_\_file(int sockfd)

Essa função faz a escrita no arquivo recebido pelo cliente.

## **3.2   Cliente.c**

Este módulo é responsável por enviar o arquivo para o servidor. O socket é criado e realiza uma verificação e atribui um IP a porta. Conecta o socket do cliente ao socket do servidor e envia o arquivo para o servidor, assim que o servidor termina a conexão com o cliente, é printado o tempo de execução e a quantidade de bytes lidos no disco.

### **3.2.1   long getIORead()**

Função que faz a contagem de bytes no buffer do disco.

### **3.2.2   float time\_diff(struct timeval \*start, struct timeval \*end)**

Função referente ao gettimeofday para a medição de tempo de execução.

### **3.2.3   void send\_file(FILE \*fp, int sockfd)**

Função que faz o envio do arquivo para o servidor.

## 4 Conclusão

Neste trabalho foi aprendido a troca de arquivos usando o protocolo TCP, fazendo a comunicação entre servidor e cliente e realizando o envio de arquivos. Foi interessante notar as diferenças de tempos de execução dependendo do tamanho dos arquivos analisados e como o desempenho é afetado. De forma geral esse trabalho foi de grande aprendizado, apesar das dificuldades, sendo elas principalmente no envio e análise dos resultados.

---

## 5 Referências

<https://github.com/warSantos/REDES02>

<https://www.hostgator.com.br/blog/o-que-e-protocolo-tcp-ip/>

<https://embarcados.com.br/socket-tcp/>

<https://www.devmedia.com.br/forum/enviar-arquivo-via-tcp-ip/254753>