
**UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ**
**REDES DE COMPUTADORES 2023/2**
**RAFAEL SACHETTO OLIVEIRA**

**Trabalho prático 3 - Em Grupo (máximo 3 pessoas)**

# 1 O Trabalho

Implementar um par de programas que operem no modelo cliente-servidor e exercitem tanto a transmissão unidirecional quanto a comunicação do tipo requisição-resposta sobre o protocolo TCP. A implementação deve utilizar a biblioteca de sockets do Unix (Linux).

# 2 Operação

O processo de comunicação entre cliente e servidor deverá seguir um padrão simples, conforme ilustrado a seguir:

## Cliente

```
processa argumentos: host_do_servidor porta_servidor nome_arquivo tam_buffer
chama gettimeof day para tempo inicial
faz abertura de conexão para host_do_servidor: porta_servidor
envia string com nome do arquivo (terminada em \0)
abre arquivo que vai ser gravado - pode ser fopen(nome, "w+")
atualiza contagem de bytes recebidos
loop
    recv buffer até que perceba que o arquivo acabou
    escreve bytes do buffer no arquivo (fwrite)
fim_loop
fecha conexão e arquivo
chama gettimeofday para tempo final e calcula tempo gasto
imprime resultado: "Buffer = %5u byte(s), %10.2f kbps (u bytes em %3u.%06u s)"
fim_cliente.
```

### Servidor

```
processa argumentos: porta_servidor tam_buffer
faz abertura passiva e aguarda conexão
recebe a string com nome do arquivo
abre arquivo que vai ser lido - pode ser fopen(nome, "r")
se deu erro, fecha conexão e termina
loop
    lê o arquivo, tam_buffer por vez até fread retornar zero
    envia o buffer lido
    se quiser, contabiliza bytes enviados
fim_loop
fecha conexão e arquivo
chama gettimeofday para tempo final e calcula tempo gasto
se quiser, imprime nome arquivo e número de bytes enviados
fim_servidor.
```

De forma resumida, o cliente deve se conectar ao servidor, enviar um string com o nome do arquivo desejado, receber o arquivo um tam_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.

## 3 Medidas de desempenho

Uma vez que os programas estejam funcionando corretamente, deve-se desenvolver uma avaliação do desempenho do par de programas. Deve-se medir a vazão da comunicação, isto é, a taxa de transferência obtida entre cliente e servidor (basicamente, o número total de bytes enviados dividido pelo tempo medido no cliente). As medições devem verificar como o desempenho varia quando diferentes tamanhos de buffer são utilizados. Em particular, deve-se incluir nas medições os casos com mensagens de tamanho $2^i$ byte(s), $0 \leqslant i \leqslant 16$. Outros valores podem ser escolhidos conforme suas observações indicarem a necessidade. O tamanho do arquivo pode ser escolhido de forma a garantir um tempo de teste nem muito longo nem muito curto. Para testes em que o par de programas executa na mesma máquina, um arquivo de aproximadamente 3 MB pode ser um bom ponto de partida. As implementações devem ser feitas na linguagem C, usando a biblioteca padrão da linguagem.

# 4 Avaliação

**Deverá ser entregues uma documentação do trabalho contendo, no mínimo:**

- Introdução: descrição do objetivo do trabalho
- Metodologia: dados sobre os experimentos, como a configuração das máquinas utilizadas, a localização das mesmas na rede. Indique também como foram feitas as medições, quantas vezes o teste foi executado, se foram execuções diferentes do programa ou apenas um loop ao redor do programa todo para fazer tudo um certo número de vezes.
- Resultados: apresente a informação coletada, tanto na forma de tabelas quanto na forma de gráficos.