
**Aluno:** Vinícius de Oliveira Silva - 2013007820
**Curso:** Ciência da Computação
**Disciplina:** Redes de Computadores

# Documentação de Trabalho Prático

## 1. Introdução

É inegável que o advento das redes de computadores iniciado na segunda metade do último século revolucionou definitivamente a forma como a humanidade se comunica. O acesso a informação se tornou instantâneo e as distâncias no mundo moderno diminuíram consideravelmente. Tendo este cenário em mente e com objetivo de nos fazer praticar a utilização de ferramentas relacionadas à comunicação em redes, nos foi proposto um trabalho prático na disciplina de Redes de Computadores que consiste em implementar um pequeno sistema de transferência de arquivos. O sistema deveria ser baseado na arquitetura cliente-servidor e seu funcionamento fundamenta-se em um programa cliente solicitando um determinado arquivo a um programa servidor e este respondendo por meio do envio do arquivo solicitado através da rede.
A comunicação entre as entidades que compõem o sistema deveria ser feita através de sockets Unix e utilizando o protocolo TCP para garantir a entrega fim-a-fim das mensagens trocadas. Ao final da implementação uma avaliação do desempenho do sistema deveria ser feita e os resultados apresentados no formato de gráficos e tabelas.

# 2. Implementação e Execução

O sistema cliente-servidor proposto para este trabalho foi implementado utilizando a linguagem de programação C99 através da IDE Jetbrains CLion e compilado através da ferramenta GCC. Todo o ambiente de desenvolvimento e execução foi baseado em Linux.
Para compilar e executar os programas, deve se utilizar o comando *make* seguido dos seguintes comandos:

- `./clienteFTP <IP_servidor> <porta_de_acesso> <nome_arquivo> <tamanho_buffer>` (para o programa cliente);

- `./servidorFTP <porta_de_acesso> <tamanho_buffer>` (para o programa servidor).

É importante também salientar os seguintes pontos:

- O programa servidor deve ser iniciado antes do programa cliente;
- O argumento <IP_servidor> refere-se ao endereço IPv4 da máquina que executa o programa servidor. O formato esperado para esse argumento é: ###.###.###.###;
- O argumento <porta_de_acesso> é um número inteiro escolhido pelo usuário que deve ser (estritamente) maior que 1024 e (estritamente) menor que 65536. Este argumento deve ter necessariamente o mesmo valor tanto na entrada do programa cliente tanto do servidor;
- O argumento <tamanho_buffer> deve ser maior que 1 e também deverá ter o mesmo valor para os programas cliente e servidor;
- O argumento <nome_arquivo> é um string que deve ter menos de 100 caracteres;
- Os computadores que executam os programas cliente e servidor devem ser visíveis um ao outro. Isto é se o computador que executa o programa cliente tentar um *ping* <IP_servidor>, uma resposta válida deverá ser recebida.
- O arquivo requisitado deverá estar no mesmo diretório que o executável do programa servidor.

# 3. Metodologia

Além de contribuir para o aprendizado de técnicas de programação de sistemas comunicantes via redes, o trabalho prático proposto tem também o objetivo de fazer com que os alunos abordem o problema comunicação via rede de uma maneira analítica. Para se obter essa perspectiva, foi pedido que variados testes fossem executados com o sistema a fim de observar e analisar seu comportamento em uma rede real.
Para cumprir esse requisito, o sistema implementado foi submetido a diversos testes e a metodologia destes é apresentada nessa seção.

## 2.1 Dados sobre o experimento:

O experimento executado consiste de executar o programa servidor em uma máquina Linux do CRC - DCC/UFMG e fazer com que este ficasse aguardando por requests do programa cliente na porta 5000 do computador. Em uma outra máquina conectada à mesma rede, o programa cliente foi executado por diversas vezes em seqüência e em cada uma das execuções, solicitava o mesmo arquivo ao programa servidor.
As seguidas execuções do programa cliente contudo, não foram idênticas. A cada 30 execuções , o parâmetro que controla o tamanho do buffer (em bytes) a ser utilizado na comunicação era alterado. É importante notar que ao responder as já mencionadas 30 requisições dos programas cliente, o programa servidor deve ser reiniciado para se adequar ao tamanho do novo buffer a ser utilizado para as próximas 30 requisições.
O arquivo utilizado para ser enviado e recebido nesta série de testes é um PDF de 7.812.767 bytes (7.8 MB aprox.) que foi escolhido pelo fato de ser binário e ocupar um espaço em disco considerável, fazendo com que a transmissão leve um tempo razoável e a verificação de possíveis erros (de implementação) no processo de transmissão seja feita mais facilmente.

## 2.2 Configuração das máquinas:

As máquinas utilizadas para executar os programas implementados estão equipadas com um processador Intel Core i7, 16GB de RAM, executam o sistema operacional GNU/Linux Ubuntu 16.04 e respondem aos seguintes endereços IP: 150.164.6.24 (velhas.grad - máquina que executou o programa servidor) e 150.164.6.26 (doce.grad - máquina que executou o programa cliente).
É importante notar porém, que não foi possível garantir o isolamento dos ambientes de execução dos mais variados fatores externos, visto que as máquinas são compartilhadas por todos os alunos do DCC, o que abre a possibilidade de que múltiplos usuários estivessem fazendo uso dessas máquinas no momento em que os testes foram executados. Tal cenário pode ocasionar impactos reais nos resultados obtidos, o que pode ter comprometido a acurácia dos números apresentados na próxima seção.

## 2.3 Medições

De forma bem simples, a única medição requerida para o desenvolvimento deste trabalho foi a aferição do tempo decorrido entre o início e o fim do recebimento do arquivo por parte do programa cliente.
Dado que o tamanho do arquivo era previamente conhecido, apenas medindo o tempo levado para recebê-lo é possível estimar a velocidade de transmissão da rede. A técnica utilizada para se obter esse tempo consiste em utilizar a função *clock_gettime* da biblioteca *time.h* da linguagem C. Tal função permite que o programador tenha acesso à hora do

Sistema Operacional com precisão de nano-segundos. Para calcular o tempo que buscamos, basta chamar esse procedimento, armazenar o resultado retornado, transmitir o arquivo e então fazer uma nova chamada à função. Tendo os valores retornados por ambas as chamadas, basta fazer uma subtração e converter o valor resultante para a escala apropriada.

# 4. Resultados

Os valores medidos utilizando os métodos descritos na seção anterior estão apresentados na tabela a seguir:

| Tamanho da Mensagem (Bytes) | № de Msgs Enviadas | Banda Média Medida (kbps) | Tempo de Transmissão Médio (s) | Desvio Padrão - Tempo Médio de Transmissão (ms) |
|---|---|---|---|---|
| 2 | 3906384 | 7627,53 | 1,024 | 59,8 |
| 4 | 1953192 | 9495,64 | 0,822 | 57,87 |
| 8 | 976596 | 9884,15 | 0,791 | 59,8 |
| 16 | 488298 | 9848,12 | 0,793 | 59,16 |
| 32 | 244149 | 10055,79 | 0,776 | 64,94 |
| 64 | 122075 | 9809,23 | 0,796 | 68,8 |
| 128 | 61038 | 9856,2 | 0,792 | 56,58 |
| 256 | 30519 | 9908,01 | 0,788 | 63,01 |
| 512 | 15260 | 9623,8 | 0,811 | 70,09 |
| 1024 | 7630 | 9819,68 | 0,795 | 55,94 |
| 2048 | 3815 | 9541,91 | 0,818 | 69,44 |
| 4096 | 1908 | 9444,78 | 0,827 | 67,52 |
| 8192 | 954 | 9869,95 | 0,791 | 56,58 |
| 16384 | 477 | 9463,06 | 0,825 | 57,23 |
| 32768 | 239 | 9796,99 | 0,797 | 66,87 |
| 65536 | 120 | 9899,38 | 0,789 | 64,94 |

Gráficos também foram construídos com o objetivo de fornecer uma interpretação visual mais simples ao leitor. Os diagramas são apresentados a seguir:

Gráfico 1:

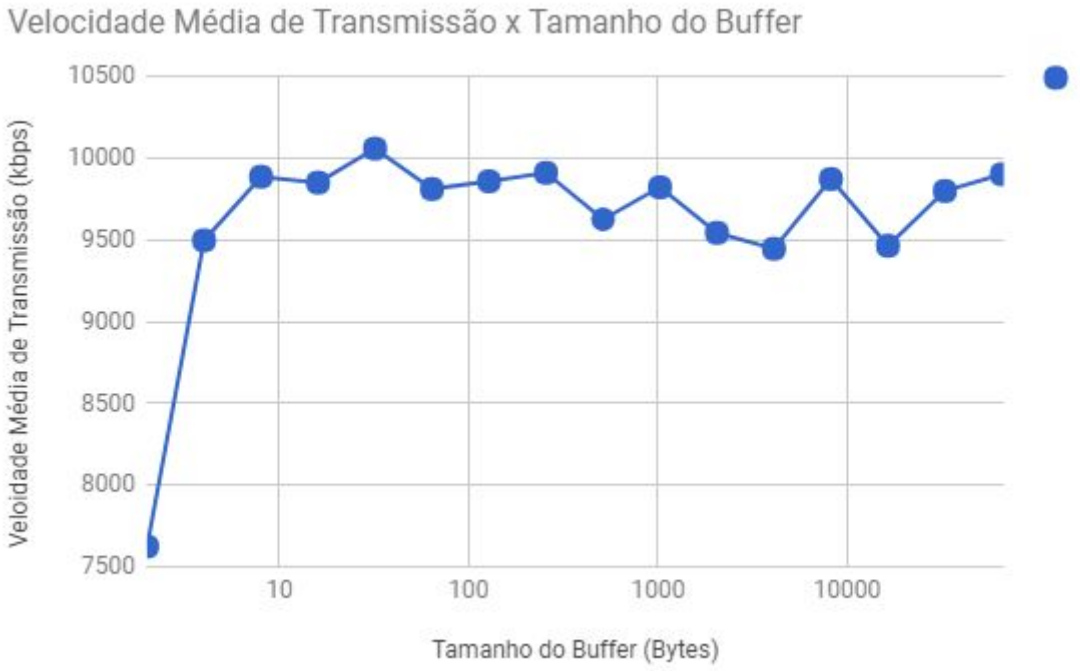
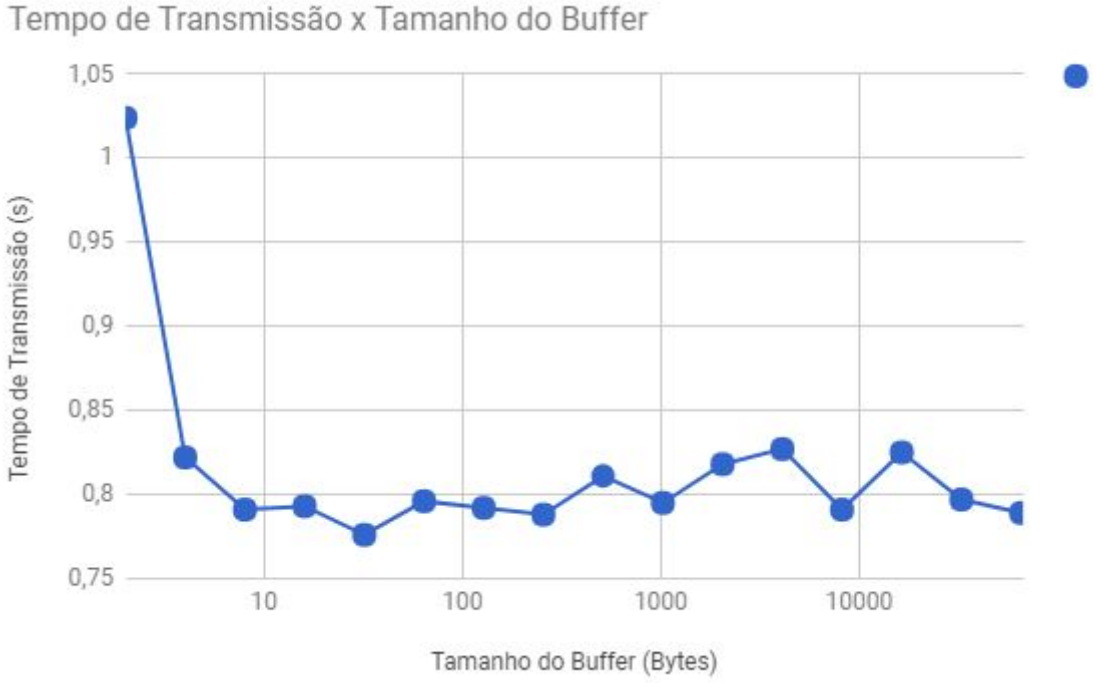



Gráfico 2:




Observação: Os gráficos apresentados foram construídos em escala logarítmica para que sua clareza não fosse comprometida.

# 5. Análise

Os resultados encontrados nos experimentos realizados não surpreendem e se mostram conforme o esperado. Analisando os números encontrados, podemos inferir que a rede local operante nos laboratórios Linux do CRC - DCC/UFMG trabalha em uma velocidade de aproximadamente 10 Mbps e esse valor é bastante consistente, apresentando pequenas variações. Um ponto que chama a atenção nos valores observados é o fato de que a velocidade média de transmissão de mensagens de apenas 2 bytes é bastante inferior às demais. É possível atribuir esse fenômeno ao grande overhead necessário para se transmitir uma mensagem de apenas 2 bytes, uma vez que muito processamento deve ser feito pelas implementações dos protocolos de rede para se enviar tão pouco da mensagem original.

# 6. Conclusão

O desenvolvimento deste trabalho ocorreu sem maiores problemas, uma vez que existem diversos materiais disponíveis gratuitamente para consulta na internet e os recursos computacionais utilizados oferecem alta disponibilidade e desempenho satisfatório.
Em suma, pode-se afirmar que o desenvolvimento do trabalho foi de grande serventia no aprendizado de Redes de Computadores, já que fez com que os alunos abordassem o assunto de uma maneira prática, agregando conhecimentos extremamente valiosos tanto para a Academia quanto para o Mercado de Trabalho.

# 7. Referências


[1] Socket Programming in C/C++
http://www.geeksforgeeks.org/socket-programming-cc/
Acessado em 30/09/2017

[2] Client/server implementation in C (sending data/files)
https://codereview.stackexchange.com/questions/43914/client-server-implementation-in-c-sending-data-files
Acessado em 30/09/2017