Trabalho Prático 1

Gilson Urbano Ferreira Dias - 2015430959
guf@ufmg.br

18 de setembro de 2018

1 Introdução

O objetivo deste trabalho é 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 será realizada utilizando a biblioteca de sockets do Unix (Linux).

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. O servidor por sua vez deve operar de forma complementar.

O protocolo TCP deve ser utilizado nesse tipo de aplicação pois ele cria um serviço de transferência confiável de dados, é orientado para conexão, antes que um processo de aplicação possa começar a enviar dados a outro, os dois processos precisam primeiro se "apresentar" — isto é, devem enviar alguns segmentos preliminares um ao outro para estabelecer os parâmetros da transferência de dados. Uma conexão TCP provê um serviço full-duplex: se houver uma conexão TCP entre o processo A em um hospedeiro e o processo B em outro hospedeiro, os dados da camada de aplicação poderão fluir de A para B ao mesmo tempo em que os dados da camada de aplicação fluem de B para A. A conexão TCP é sempre ponto a ponto, isto é, entre um único remetente e um único destinatário [6].

2 Metodologia

Diversos testes foram realizados de forma a verificar o correto funcionamento do par de programas cliente e servidor que foram implementados na linguagem C, e para coletar os dados de perfomance que serão analisados posteriormente. Os testes foram realizados em um computador com processador Intel Core i5-4210U rodando a 1.70 GHz, 6 Gb de memória RAM, e sistema operacional Ubuntu 16.04 LTS.

Para a execução dos testes foi utilizado um arquivo de texto com 10.233.868 bytes (10.23 MB), o objetivo principal e verificar como o desempenho do par de programas varia quando diferentes tamanhos de buffer são utilizados. Os programas foram colocados em pastas diferentes para garantir que a transferência do arquivo realmente ocorreu. Em alto nível o funcionamento dos programas segue os seguintes passos, o servidor deve ser iniciado e aguardar que o o cliente se conecte, enviando uma string com o nome do arquivo desejado, para que possa 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. O servidor por sua vez deve operar de forma complementar.

Os parâmetros do par de programas e a ordem que devem ser inseridos para o perfeito funcionamento devem seguir os exemplos a seguir. O programa servidorFTP recebe como parâmetros o número da Porta e o tamanho do buffer em bytes, por exemplo, Porta 5001 e buffer de 256 bytes.

```
urbanogilson@urbanogilson:~/servidorFTP$ ./servidorFTP 5001 256
```

No programa clienteFTP são necessários os parâmetros host, número da Porta do servidor, nome do arquivo que se deseja recuperar do servidor e o tamanho do buffer em bytes, por exemplo, host 127.0.0.1 IP loopback utilizado na realização dos testes na máquina atual, Porta 5001 do servidor, arquivo a ser recuperado file.txt e o buffer com tamanho de 256 bytes.

```
urbanogilson@urbanogilson:~/clienteFTP$ ./clienteFTP 127.0.0.1 5001 file.txt 256
```

Medições de tempo foram realizadas variando o tamanho da mensagens em 2^i bytes, com $0 \leq i \leq 16$. Para cada tamanho de mensagem (buffer = 2^i bytes), foram realizados 5 execuções diferentes do programa utilizando a função gettimeofday da biblioteca sys/-time.h para obter o tempo de execução, os resultados estão dispostos na Tabela 1. Na Tabela 2 foram realizadas as médias do tempos obtidos nos testes, medidas de dispersão dos tempos e vazão da comunicação.

3 Resultados

Os resultados dos tempos (em segundos) de execução obtidos utilizando a estratégia de realizar execuções diferente do programa para cada tamanho de buffer = 2^i bytes, foram tabulados e podem ser vistos abaixo Tabela 1.

i	2^i	**Tempo 1**	**Tempo 2**	**Tempo 3**	**Tempo 4**	**Tempo 5**
0	1	12.177584	11.989892	12.148617	12.185992	12.108762
1	2	5.566358	5.751543	5.872764	5.966270	5.865718
2	4	2.897371	3.010826	2.954516	2.916970	2.988919
3	8	1.478574	1.439905	1.559881	1.486682	1.599110
4	16	0.770207	0.750479	0.719846	0.743558	0.729637
5	32	0.391207	0.354101	0.375033	0.388005	0.389122
6	64	0.192399	0.188443	0.188700	0.190878	0.187017
7	128	0.111296	0.108765	0.121621	0.110878	0.110963
8	256	0.059986	0.061845	0.065082	0.068142	0.063862
9	512	0.043763	0.043446	0.037932	0.042238	0.036517
10	1024	0.033564	0.031252	0.034612	0.033291	0.033426
11	2048	0.027506	0.023891	0.027515	0.021806	0.027250
12	4096	0.020911	0.023728	0.021075	0.020803	0.024489
13	8192	0.018326	0.018596	0.021536	0.017296	0.017790
14	16384	0.013427	0.018251	0.019567	0.018452	0.017964
15	32768	0.013299	0.018617	0.018133	0.016113	0.014988
16	65536	0.016168	0.016137	0.014895	0.017213	0.018238

Tabela 1: Tempo de transmissão para um arquivo de 10.23 MB.

Utilizando os dados acima foram realizados algumas medidas, como a média dos tempos para cada buffer = 2^i bytes, medidas de dispersão como: desvio padrão e variância

dos tempos amostrados e uma medida da vazão da comunicação que é o número de bytes enviados dividido pelo tempo medido no cliente. É possível notar um decréscimo no tempo necessário para a transferência do aquivo de acordo o tamanho do buffer $= 2^i$ bytes aumenta, um gráfico Figura 1 foi criado para verificar essa conclusão visualmente.

i	2^i	**Tempo** $\overline{t}$	**Desvio Padrão** σ	**Variância** σ^2	**Vazão** $\overline{v}$
0	1	12.122169	0.079876	0.006380	844.23
1	2	5.804531	0.153373	0.023523	1763.08
2	4	2.953720	0.047504	0.002257	3464.74
3	8	1.512830	0.064872	0.004208	6764.72
4	16	0.742745	0.019435	0.000378	13778.43
5	32	0.379494	0.015549	0.000242	26967.17
6	64	0.189487	0.002135	0.000005	54008.17
7	128	0.112705	0.005084	0.000026	90802.58
8	256	0.063783	0.003117	0.000010	160447.20
9	512	0.040779	0.003332	0.000011	250958.04
10	1024	0.033229	0.001222	0.000001	307980.02
11	2048	0.025594	0.002614	0.000007	399860.43
12	4096	0.022201	0.001764	0.000003	460960.13
13	8192	0.018709	0.001658	0.000003	547008.25
14	16384	0.017532	0.002374	0.000006	583718.42
15	32768	0.016230	0.002206	0.000005	630552.56
16	65536	0.016530	0.001259	0.000002	619101.28

Tabela 2: Análise do tempos de transmissão para um arquivo de 10.23 MB.

Visualmente fica fácil notar um resultado que olhando apenas a Tabela 2 é difícil de perceber, com o tamanho do buffer $= 2^i$ bytes crescendo exponencialmente o tempo decresce exponencialmente.

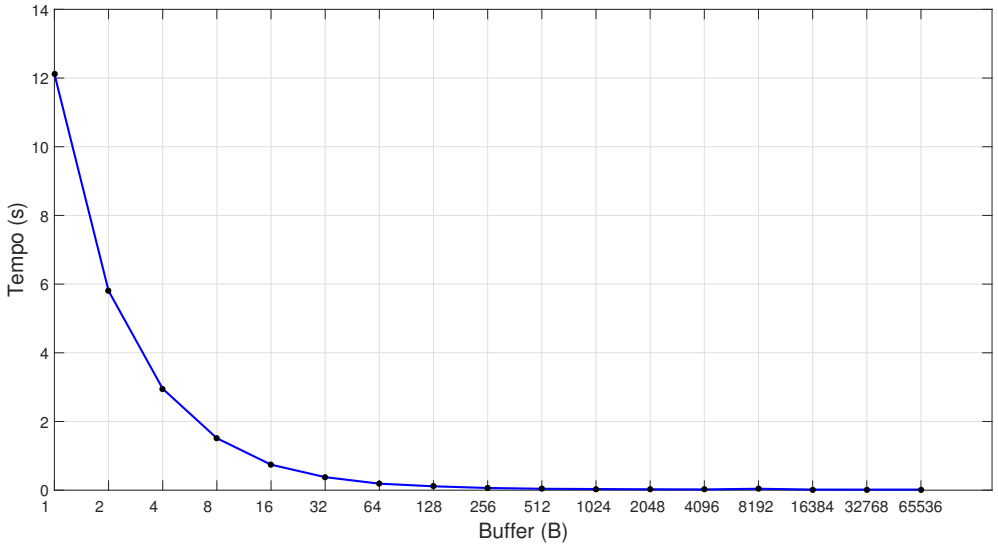

Figura 1: Relação Buffer x Tempo na transmissão de um arquivo com 10.23 MB.

Foram omitidas as barras verticais indicando a variância de cada buffer $= 2^i$ pois os valores ficaram baixos, quase insignificante em relação aos resultados obtidos, com isso não adicionaria nenhuma informação relevante para a análise.

O gráfico Figura 2 foi criado para mostrar que existe uma limitação de vazão com o aumento do buffer. Na próxima seção serão realizadas as análises dos resultados dessa seção inclusive a limitação mencionada anteriormente.

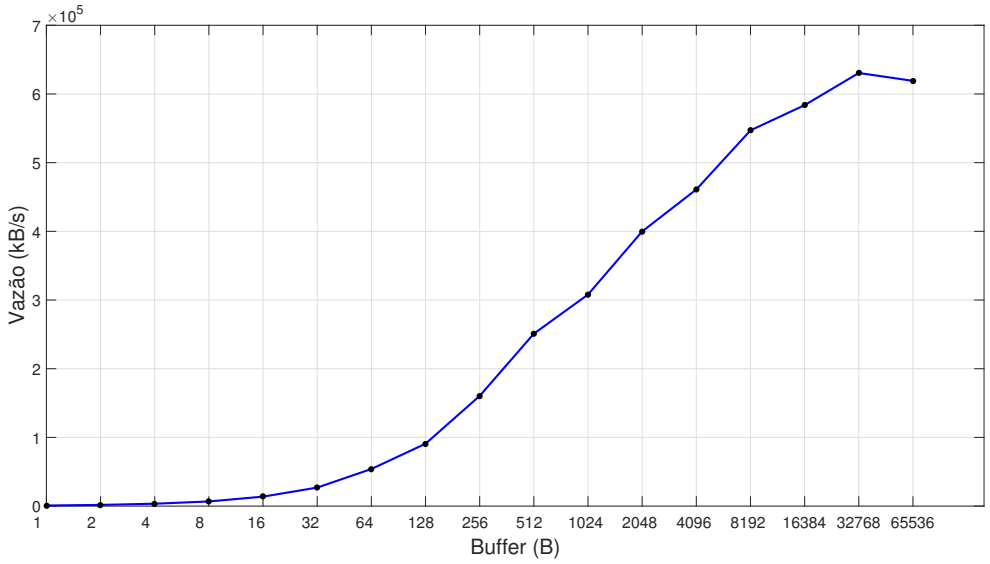

Figura 2: Relação Buffer x Vazão na transmissão de um arquivo com 10.23 MB.

4 Análise

A Tabela 2 mostra que o desvio padrão ficou baixo mesmo utilizando apenas 5 amostras de tempo, o maior desvio padrão relativo 13.59% aconteceu na amostra $i = 15$, nessa amostras os valores dos tempos medidos já estavam próximos de zero o que gera imprecisões no valores medidos, mas o valor médio do desvio padrão relativo de todas as amostras ficou em apenas 4.51%.

O resultado da Figura 1 já era esperado, pois um tamanho maior do buffer diminuiu o número de mensagens trocadas entre trocadas entre o cliente e o servidor, implicando em um decréscimo no tempo de transmissão.

Não só o formato do gráfico Figura 2 mas principalmente o último ponto buffer $=$ 65536, mostra que mesmo o buffer tendo um tamanho muito grande a vazão pode não acompanhar essa tendência de crescimento, para que a vazão continue crescendo o tempo de transmissão precisa continuar diminuindo mas algumas limitações não permitem isso, a primeira é o tempo para salvar os bytes no disco, mas o que mais pode afetar esse desempenho de tempo, é o tempo necessário para alocar memória dinamicamente para um buffer muito grande, isso pode levar mais tempo do que transmitir o mesmo arquivo utilizando um buffer de tamanho menor.

5 Conclusão

A implementação ocorreu de forma tranquila, o protocolo TCP utilizado já havia sido estudado em sala de aula, foram necessárias algumas pesquisas e revisões sobre o assunto, e um estudo das bibliotecas necessárias para a realização do trabalho. A maior dificuldade foi aprender o funcionamento da biblioteca de sockets do Unix, por ser um par de programas o nível de dificuldade também dobra para encontrar os erros, mesmo assim o trabalho foi concluído, com o resultado ficando dentro do esperado.

Alguns resultados eram esperados, por exemplo, a transmissão do arquivo utilizando o protocolo TCP ocorreu sem perdas de informação, o tempo de transmissão diminui quando o tamanho do buffer foi acrescido, mas como explicado na análise a vazão pode não seguir um padrão de crescimento proporcional ao crescimento do buffer, o que leva a possibilidade de aplicar algoritmos de otimização para que se possa otimizar o tamanho do buffer para uma dada aplicação, minimizando o tempo de transmissão e consumo de memória.

Referências

[1] arpa/inet.h. `http://pubs.opengroup.org/onlinepubs/009696899/basedefs/arpa/inet.h.html`. Accessado em: 15-09-2018.

[2] netinet/in.h. `http://pubs.opengroup.org/onlinepubs/000095399/basedefs/netinet/in.h.html`. Accessado em: 15-09-2018.

[3] sys/socket.h. `http://pubs.opengroup.org/onlinepubs/7908799/xns/syssocket.h.html`. Accessado em: 15-09-2018.

[4] sys/time.h. `http://pubs.opengroup.org/onlinepubs/7908799/xsh/systime.h.html`. Accessado em: 15-09-2018.

[5] sys/types.h. `http://pubs.opengroup.org/onlinepubs/009696699/basedefs/sys/types.h.html`. Accessado em: 15-09-2018.

[6] James F Kurose and Keith W Ross. *Redes de computadores e a Internet - uma abordagem top-down*. Pearson Education do Brasil, 6 edition, 2013.