Terceiro trabalho prático de Redes de Computadores: transferência de bytes na arquitetura cliente-servidor

Gabriel Gomes de Oliveira - 212050079

1 Introdução

Uma das tarefas mais importantes da rede de computadores é manipular e entregar os pacotes que trafegam pelos roteadores e enlaces de forma rápida e segura. Na atualidade, diversos protocolos foram desenvolvidos para tornar essa transferência cada vez melhor, seja na camada de aplicação, na camada de rede ou na camada de enlace. Sendo assim, o terceiro trabalho da disciplina de Rede de Computadores visa implementar, de forma simplificada, um par de programas que operem na arquitetura cliente-servidor cujo principal propósito desses programas é realizar a transferência unidirecional de dados entre os dois lados e, além disso, testar a comunicação do tipo requisição-resposta sobre o protocolo TCP. Ademais, o trabalho também tem como objetivo a parte analítica, no qual envolve apresentar e explicar os resultados numéricos advindos da transferência de dados (dados de desempenho).

2 Metodologia

A arquitetura cliente-servidor foi implementada totalmente na linguagem C. Foram implementados dois programas: um funcionando como o servidor, o lado que envia os dados requisitados, e outro funcionando como o cliente, o lado que deve receber esses dados. Cada programa está em um arquivo diferente. Além disso, a implementação contempla o tratamento da organização de arquivos, para que o local (pasta) de execução do usuário não fique repleto de arquivos e desorganizado.

Para os testes, foi utilizada somente uma máquina com as seguintes configurações:

- Sistema de 64 bits rodando Linux sob a distro Zorin OS 16.3;
- Processador Intel Core i3-1115G4 de 11ª geração com dois núcleos;
- Processamento de gráficos embutido dentro do processador (Intel UHD Graphics);
- Memória RAM de 4GB DDR4 2666MHz.

Durante a realização dos testes, é importante pontuar que todos os dados advindos do lado do cliente serão salvos em um arquivo de extensão *.csv* chamado "tabela_de_res.csv". Esse arquivo armazenará, para cada execução, o tamanho do arquivo em MB alvo da transferência, o tamanho de buffer em bytes da execução, a taxa de transferência em kBps (kiloBytes por segundo), a quantidade de bytes que o cliente recebeu, para confirmar que todos os bytes do arquivo original foram transferidos, e o tempo medido em segundos em que o arquivo foi transferido. Caso queria verificar um exemplo desse arquivo, entre na pasta *"exemplos"*. Esse arquivo foi gerado durante uma cadeia de execuções.

Ademais, esses dados também serão impressos na tela após cada execução, tanto os dados referentes ao cliente para o cliente, quanto os dados referentes do servidor para o servidor.

Além dos programas principais, foram desenvolvidos alguns programas básicos, escritos em outras linguagens, para auxiliar no processo de geração de arquivos e execução do par de programas:

- Arquivo *script_arqs.py*: Um programa, desenvolvido na linguagem Python, cujo objetivo é gerar arquivos de tamanhos diferentes, ou seja, quantidades de MB diferentes. Esse programa foi criado para auxiliar na execução de testes no par de programas. Para executa-lo, digite no terminal de linha de comando:

```
python3 script_arqs.py numero1 numero2 [...]
```

Essa linha de comando diz que serão gerados dois arquivos, porque foram passados somente dois argumentos, com *numero1* MB e outro com *numero2* MB. Porém, inúmeros argumentos podem ser passados, com diversos tamanhos conforme forem requisitados. Esses arquivos serão salvos na pasta *"arquivos"*, que é criada caso não exista.

Para as medidas de desempenho, foram aplicados tamanhos de buffer de 2^0 bytes até 2^{16} bytes, como recomenda a documentação do trabalho. Ademais, foram realizados testes para arquivos de tamanho 3MB, 10MB e 15MB.

3 Execução

Tratando do par de programas cliente-servidor implementado, a execução simples é uma execução de transferência de um arquivo que ocorre somente uma vez. Para executar o par de programas o usuário deve seguir o passo-a-passo:

1. Primeiro, é preciso que o terminal de linha de comando esteja aberto no diretório em que o projeto está, para que os comandos possam ser efetuados.

2. **Compilação**: Dentro da pasta do projeto há um arquivo *Makefile* que compila tanto o lado servidor quanto o lado cliente. Tudo o que o usuário deve fazer é digitar *make* na linha de comando.

3. **Arquivos**: Caso ainda não tenha os arquivos para transferência, é possível gerar com o programa *script_arqs.py*.

4. **Ativar servidor**: Para que comece o processo, o servidor deve ser ligado para começar a enviar os dados. O comando no terminal de comandos é:

```
./servidor porta tam_buffer
```

Onde *porta* é o número da porta dentro da rede local que os bytes passarão e onde o servidor estará esperando por requisições e *tam_buffer* é a quantidade de bytes que serão lidos do arquivo e enviados ao cliente. O valor de *tam_buffer* do servidor deve ser o mesmo valor de *tam_buffer* do cliente.

5. **Solicitação do cliente**: Com o servidor esperando uma conexão através do socket, o cliente se conecta ao servidor com o seguinte comando no terminal de comandos:

```
./cliente host_do_servidor porta_servidor nome_arquivo tam_buffer
```

Onde *host_do_servidor* é o endereço do servidor, no qual geralmente é o endereço da rede local (127.0.0.1), *porta_servidor* é número da porta dentro da rede local que os bytes passarão e onde o cliente receberá os dados, *nome_arquivo* é o nome do arquivo (por exemplo, arquivo3MB.txt) que o cliente requisita (esse arquivo deve estar na pasta *"arquivos"*) e *tam_buffer* é a quantidade de bytes que serão recebidos do servidor. Lembrando que o valor de *tam_buffer* do servidor deve ser o mesmo valor de *tam_buffer* do cliente.

6. **Encerramento**: Após a transferência ocorrer, os dados relevantes, como a taxa de transferência e o tempo, serão exibidos na tela do cliente e na tela do servidor. Além disso, esses dados serão gravados no arquivo *.csv* e as conexões do servidor e do cliente serão encerradas. O usuário pode verificar se todos os bytes foram transferidos de forma correta através do arquivo gerado na pasta *"resultados"*, que é o arquivo com os bytes que o cliente recebeu do servidor.

3.1 Execução múltipla

A execução múltipla compreende dois arquivos, escritos em Shell, que fazem a execução do par de programas várias vezes.

Esses arquivos são *script_exec_cliente.sh* e *script_exec_servidor.sh*. Para cada arquivo de teste gerado, disponível na pasta de "arquivos", são realizados 5 testes (transferindo o arquivo do lado servidor para o lado cliente) para cada tamanho de buffer. Por exemplo, para o arquivo de 3MB serão realizados, primeiramente, 5 testes com o tamanho de buffer de 1 byte, depois 5 testes com o tamanho de buffer de 2 bytes, e assim por diante.

Para testar nos moldes descritos acima, os dois lados devem estar compilados e deve existir arquivos na pasta *"arquivos"*.

Primeiro execute o script do servidor através do comando na linha de comando:

```
bash script_exec_servidor.sh
```

Agora, execute o script do cliente através da linha de comando em outra instância do terminal:

```
bash script_exec_cliente.sh
```

Espere a execução acabar e é possível verificar todos os resultados no arquivo *tabela_de_res.csv*.

4 Resultados

Os resultados foram obtidos através de análises no arquivo *.csv* e realizando a execução múltipla, descrita anteriormente. A visualização é simplificada por tabelas onde, para cada arquivo de tamanho diferente, estão dispostos as variações de tamanho do buffer relacionando com a taxa de transferência média, medida em kBps, e o tempo médio da transferência. Relembrando que o teste foi executado 5 vezes para cada tamanho de buffer em cada arquivo.

É possível verificar a veracidade dos dados abaixo no arquivo *.csv* disponível na pasta *"exemplos"*.

4.1 Arquivo com tamanho 10MB

	Tamanho do buffer (bytes)					
	2^0	2^1	2^2	2^3	2^4	2^5
Taxa de transferência média (kBps)	2511.90	5792.83	11688.93	28396.37	39660.24	69969.56
Tempo médio (s)	4.114	1.772	0.877	0.363	0.258	0.147

	Tamanho do buffer (bytes)					
	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Taxa de transferência média (kBps)	102775.22	170604.75	205483.56	300425.82	341748.86	393814.13
Tempo médio (s)	0.100	0.060	0.050	0.034	0.042	0.026

	Tamanho do buffer (bytes)				
	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Taxa de transferência média (kBps)	443314.51	328292.66	320558.32	335152.77	377711.01
Tempo médio (s)	0.024	0.031	0.049	0.043	0.028

4.2 Arquivo com tamanho 15MB

	Tamanho do buffer (bytes)					
	2^0	2^1	2^2	2^3	2^4	2^5
Taxa de transferência média (kBps)	2564.62	5811.39	12645.53	28967.53	45096.96	69845.24
Tempo médio (s)	5.991	2.645	1.215	0.532	0.344	0.220

	Tamanho do buffer (bytes)					
	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Taxa de transferência média (kBps)	123076.67	173607.21	235403.37	258982.64	286358.83	361221.81
Tempo médio (s)	0.125	0.088	0.065	0.0604	0.064	0.042

	Tamanho do buffer (bytes)				
	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Taxa de transferência média (kBps)	385923.10	374166.32	404813.15	410497.46	403992.49
Tempo médio (s)	0.0402	0.0412	0.039	0.037	0.0428

4.3 Arquivo com tamanho 3MB

	Tamanho do buffer (bytes)					
	2^0	2^1	2^2	2^3	2^4	2^5
Taxa de transferência média (kBps)	2560.48	5791.41	11989.56	24971.94	41783.15	73290.86
Tempo médio (s)	1.200	0.533	0.259	0.130	0.0743	0.041

	Tamanho do buffer (bytes)					
	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Taxa de transferência média (kBps)	122766.78	195398.37	284364.01	352362.25	425294.27	432611.25
Tempo médio (s)	0.0255	0.015	0.0108	0.008	0.007	0.007

	Tamanho do buffer (bytes)				
	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Taxa de transferência média (kBps)	481391.70	513187.46	551065.27	517205.25	562003.50
Tempo médio (s)	0.006	0.005	0.005	0.006	0.005

4.4 Análise final

Como é perceptível pelas tabelas exibidas, a transferência de arquivos com diferentes tamanhos de buffer aumenta a velocidade de transmissão dos dados durante um certo intervalo de valores. No caso máquina em que executou os testes aqui apresentados, a velocidade de transmissão teve variações notáveis conforme os tamanhos do buffer variavam entre 2^0 bytes (1 byte) e 2^{10} bytes (1024 bytes). Em contrapartida, a partir de 2^{11} bytes de buffer, notou-se um crescimento menor, o que pode-se dizer que a velocidade de transmissão se torna cada vez mais estável e o papel do aumento do buffer não é mais primordial. Essa estabilização pode ter ocorrido por limitações de hardware, mas pode estar ligado ao formato da execução.