Universidade Federal
de São João del-Rei

Departamento de Ciência da Computação

Vítor Augusto Niess Soares Fonseca, Matheus Tavares Elias

Teceiro Trabalho Prático - Redes de Computadores

São João del-Rei, Dezembro de 2023

Sumário

1 Introdução **3**

2 Metodologia **3**
2.1 Código do Cliente 3
2.2 Códgio do Servidor 3

3 Resultados e Discussão **4**

4 Considerações Finais **5**

5 Referências **6**

1 Introdução

Este documento apresenta a documentação técnica para um sistema cliente-servidor implementado para operar no contexto de transmissão unidirecional e comunicação do tipo requisição-resposta, utilizando o protocolo TCP. Desenvolvido com o intuito de proporcionar uma base sólida para aplicações distribuídas, este conjunto de programas visa demonstrar a eficácia na transmissão de dados em ambientes de rede.

O propósito fundamental deste projeto é a criação de um par de programas interconectados, onde um atua como cliente e o outro como servidor, estabelecendo uma comunicação confiável por meio do protocolo TCP. Este sistema é concebido para demonstrar a viabilidade e eficiência na transmissão unidirecional de dados, assim como na implementação de comunicação do tipo requisição-resposta entre sistemas distribuídos.

2 Metodologia

A implementação do sistema cliente-servidor para comunicação TCP envolve a criação de dois programas independentes, ambos desenvolvidos em liguagem "C", que se comunicam por meio do protocolo TCP/IP. A seguir, são apresentadas explicações detalhadas sobre os códigos fonte do cliente e do servidor, abordando suas principais rotinas e funcionalidades.

2.1 Código do Cliente

O código do cliente inicia estabelecendo uma conexão TCP/IP com o servidor por meio da função socket. A comunicação é direcionada ao endereço IP "127.0.0.1"na porta definida pela constante PORT. Após a conexão bem-sucedida, o cliente envia uma solicitação ao servidor por meio da função "send", com a mensagem "GET /file.txt". Essa solicitação indica ao servidor o tipo de conteúdo desejado.

Após o envio da solicitação, o cliente aguarda a resposta do servidor. A função "receive_file"é responsável por receber os dados do servidor por meio da função "recv"e processá-los conforme necessário. Com o arquivo recebido, os dados recebidos são simplesmente impressos na saída padrão. É importante destacar que a constante "MAX_PACKET_SIZE"foi ajustada durante as etapas de teste para otimizar o desempenho da transmissão.

Além disso, o tempo decorrido para a execução do cliente é medido utilizando a função "gettimeofday", fornecendo informações sobre o tempo de início e término da execução do programa.

2.2 Códgio do Servidor

O código do servidor inicia criando um socket utilizando a função socket e o associa à porta especificada pela constante PORT. Em seguida, aguarda conexões de clientes usando a função "listen". Durante a execução contínua do servidor, novas conexões são aceitas pela função "accept".

Ao receber uma conexão, o servidor lê a solicitação do cliente por meio da função "recv". Dependendo do conteúdo da solicitação, o servidor utiliza a função "serve_file"para enviar o conteúdo do arquivo correspondente ao cliente. Para isso, o servidor responde a solicitações para "GET /file.txt"e "GET /image.png"(embora os testes executados tenham feito uso apenas do "file.txt"), enviando os respectivos arquivos, enquanto outras solicitações recebem uma resposta simples.

O servidor envia os dados para o cliente por meio da função "send"e pode opcionalmente introduzir pequenos atrasos entre os pacotes usando "usleep(10000)"para simular condições de rede específicas.

Além disso, mbos os programas cliente e servidor utilizam a função "handle_error"para tratar erros de execução, exibindo mensagens de erro descritivas antes de encerrar a execução.

3 Resultados e Discussão

Essa seção apresenta as conclusões obtidas por meio de testes meticulosos realizados no sistema cliente-servidor, implementado em linguagem C, como mencionado. Durante os experimentos, o tamanho máximo do *buffer* foi variado ao se fazer uma requisição de um arquivo com, aproximadamente, três *megabytes* (3 MB), abrangendo potências de 2, desde 2^4 até 2^{16}. O objetivo desses testes foi analisar o desempenho do sistema em termos de tempo de execução, proporcionando informações sobre o impacto do tamanho do *buffer* na eficiência da transmissão de dados entre o cliente e o servidor. Os resultados estão detalhadamente tabulados na Tabela 1, demonstrada a seguir, destacando a relação entre o tamanho do *buffer* e o tempo decorrido para execução, em segundos, oferecendo uma visão abrangente sobre o comportamento do sistema em diferentes cenários.

Vale ressaltar que foi sugerido pelo professor que os testes fossem feitos com o tamanho do *buffer* variando de 2^1 até 2^{16}, porém os valores de 2^1 até 2^3 se demonstraram extremamente lentos, tornando os testes inviáveis.

Tamanho do Buffer	**Tempo (segundos)**
2^4	1987,593370
2^5	889,627348
2^6	439,640666
2^7	220,755718
2^8	110,391592
2^9	55,049279
2^{10}	27,169162
2^{11}	13,601877
2^{12}	6,798805
2^{13}	3,456990
2^{14}	1,699008
2^{15}	0,867376
2^{16}	0,440608

Tabela 1: Resultados dos testes variando o tamanho do *buffer*.

Além da tabela, também foi desenvolvido um gráfico com os dados obtidos, para mehor visualização.

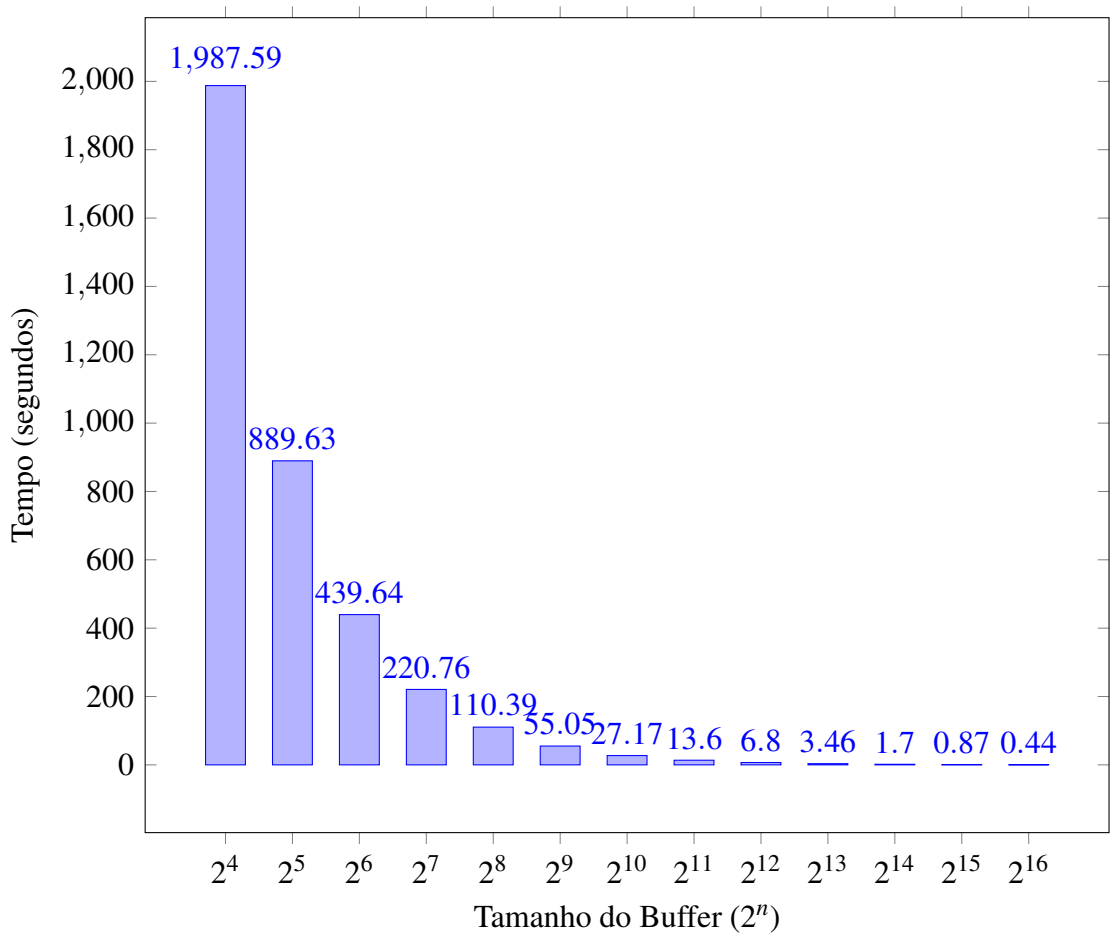

Figura 1: Resultados dos testes variando o tamanho do buffer.

4 Considerações Finais

Os resultados obtidos neste estudo demonstraram uma correspondência direta com as expectativas, evidenciando uma relação inversamente proporcional entre o tamanho do *buffer* e o tempo de execução do sistema cliente-servidor. Conforme antecipado, à medida que o tamanho do *buffer* aumentou exponencialmente, o tempo de execução foi consistentemente reduzido pela metade. Esta observação reforça a eficiência da implementação, indicando que o sistema responde de maneira ágil às variações no tamanho do *buffer*.

Este projeto também se revelou uma oportunidade valiosa para o aprendizado dos autores, oferecendo uma compreensão mais aprofundada dos desafios e considerações práticas ao implementar sistemas de comunicação cliente-servidor. A análise dos resultados não apenas contribuiu para a consolidação de conhecimentos teóricos, mas também proporcionou conhecimentos práticos sobre otimizações de desempenho em ambientes distribuídos.

Em síntese, a relação sistemática entre o tamanho do *buffer* e o desempenho do sistema, aliada à experiência adquirida durante o desenvolvimento, reforça a relevância e o impacto positivo deste trabalho no processo de aprendizado dos envolvidos.

5 Referências

[1] Kurose, J. F., Ross, K. W. (2013). Redes de Computadores e a Internet: Uma Abordagem Top-Down. Editora.

[2] Silberschatz, A., Galvin, P. B., Gagne, G. (2018). Operating System Concepts (10th ed.). Acessado em 19/11/2023.

https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf