CLOUD TP

Nome: Márcio Ribeiro Júnior
Matrícula: 223116349

Introdução

Este trabalho tem como objetivo analisar a comunicação entre sockets no modelo cliente-servidor utilizando a linguagem C. A comunicação via sockets é uma das formas mais comuns de troca de dados em redes de computadores, sendo amplamente empregada em sistemas distribuídos. Através deste estudo, buscamos compreender como implementar a comunicação bidirecional entre um cliente e um servidor, utilizando recursos da linguagem C para estabelecer a conexão, transmissão e recepção de mensagens de forma eficiente. A análise aborda tanto a implementação prática quanto às considerações teóricas necessárias para garantir o funcionamento adequado do modelo cliente-servidor em ambientes de rede.

Metodologia

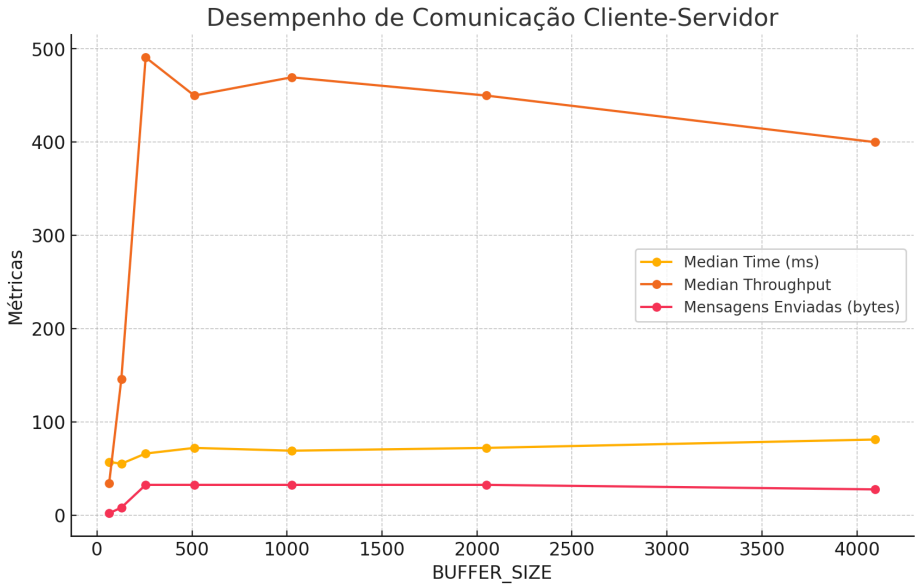
Os sockets, cliente-servidor, foram executados em ambiente WSL, na mesma rede, durante cinco vezes para diferentes tamanhos de *buffer* de mensagens 2^i | 6 <= i <= 12. Durante a execução, o cliente enviará uma listagem de todos os nomes dos arquivos em determinado diretório, especificado pelo usuário, para o servidor como conteúdo da mensagem a ser armazenada. Nesse sentido, calcula-se a média de tempo, utilizando a função *gettimeofday*, banda e mensagens enviadas em rede para análise de comportamento da comunicação.

A fim de evitar conflito entre dados e comandos enviados, o cliente utiliza a técnica *char stuffing,* adicionando um caractere especial para cada dado enviado i.e. identificando certa mensagem como nome de arquivo. Diante disso, o servidor é responsável por retirar o caractere adicionado e classificar a mensagem como um nome de arquivo. Por fim, caso uma mensagem seja recebida sem os caracteres especiais, será considerada um comando enviado pelo cliente.

Paralelo a isso, realiza-se um tratamento da mensagem enviada conforme o tamanho do *buffer de mensagem*. Caso a mensagem seja menor do que o *buffer* escolhido, a mensagem será complementada com caracteres especiais, para evitar vazamento de memória, a serem retirados pelo servidor durante o processo de recepção e armazenamento. Mensagens maiores que o *buffer* não serão enviadas.

Resultados

Nos quadros seguintes, apresentamos os resultados obtidos durante a análise da comunicação entre sockets. Através dos dados coletados, podemos observar como diferentes tamanhos de buffer impactam o desempenho da comunicação, influenciando tanto o tempo de envio quanto a taxa de transferência das mensagens. Esses resultados fornecem uma visão clara de como a variação do buffer pode afetar a eficiência da transmissão de dados em uma rede.



BUFFER_SIZE	Median Time (ms)	Median Throughput	Mensagens Enviadas (bytes)
64	57.0	34.30	1.955
128	55.0	145.51	8.003
256	66.0	490.71	32.387
512	72.0	449.82	32.385
1024	69.0	469.38	32.381
2048	72.0	449.82	32.387
4096	81.0	399.84	27.524

Esses dados demonstram como o aumento do tamanho do buffer impacta o desempenho da comunicação cliente-servidor, principalmente em termos de throughput e tempo de envio. Até 128 bytes, o aumento no buffer resulta em melhorias no throughput, mas após esse ponto, o benefício começa a se estabilizar. Além disso, embora o throughput melhore com buffers maiores, o tempo de envio não diminui consistentemente, o que sugere que um tamanho de buffer muito grande pode levar a overhead, diminuindo a eficiência.

Análise

Ao analisar os dados fornecidos sobre a comunicação cliente-servidor em sockets, podemos observar algumas tendências interessantes sobre o impacto do tamanho do buffer nas métricas de desempenho. O **tempo mediano** de envio das mensagens não segue uma relação linear simples com o aumento do tamanho do buffer. Inicialmente, entre os buffers de 64 e 128 bytes, o tempo de envio diminui ligeiramente de 57 ms para 55 ms. No entanto, a partir de 256 bytes, o tempo começa a aumentar gradualmente, alcançando 81 ms no buffer de 4096 bytes. Esse comportamento sugere que, após certo ponto, o aumento do buffer pode começar a causar overhead, possivelmente devido a fatores como processamento adicional ou congestionamento na rede ao lidar com pacotes maiores.

Por outro lado, o **throughput mediano**, que mede a eficiência da transmissão de dados, mostra um aumento considerável entre o buffer de 64 bytes (34,30) e o buffer de 128 bytes (145,51). Porém, após o buffer de 128 bytes, o throughput tende a se estabilizar, com valores variando entre 449,82 e 490,71 para buffers de 256 bytes a 2048 bytes. Esse comportamento sugere que, além de um certo limite, o aumento do buffer não resulta em melhorias substanciais na taxa de transmissão de dados. Assim, pode-se inferir que o sistema já atinge um ponto de máxima eficiência em torno de 256 bytes, após o qual o aumento do buffer não traz benefícios significativos.

Em relação às **mensagens enviadas** em bytes, o aumento do tamanho do buffer está diretamente relacionado ao aumento da quantidade de dados enviados de uma vez. Isso é esperado, pois buffers maiores permitem o envio de mensagens maiores sem a necessidade de fragmentação. No entanto, a partir de uma certa capacidade, o aumento do buffer não parece ser o principal fator limitante. O throughput estabiliza e o tempo de envio não diminui de forma consistente, indicando que fatores além do tamanho do buffer podem influenciar o desempenho da comunicação.

Conclusão

Em suma, os dados indicam que o **tamanho ideal do buffer** está entre 128 e 512 bytes, já que essa faixa oferece o melhor equilíbrio entre latência e throughput. Buffers maiores que 512 bytes não resultam em melhorias significativas e podem até introduzir overhead, o que afeta negativamente o tempo de envio. Portanto, para este cenário específico de comunicação cliente-servidor, **buffers de tamanho médio** parecem ser os mais eficientes, equilibrando a quantidade de dados transmitidos com a eficiência de processamento e a latência da rede.