Universidade Federal
de São João del-Rei

Trabalho Prático 3

Redes de Computadores

Gustavo Euller Honório Pedrosa

Pedro Garcia Nunes

São João del-Rei

Dezembro de 2023

1- Introdução

Neste trabalho prático da disciplina de Redes de Computadores, vamos 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.

2- Implementação do Servidor

Nesta seção, será analisado o funcionamento e as características do módulo Servidor, bem como as funções relacionadas a ele.

2.1- Funcionamento do Servidor

Em resumo,o servidor escuta por conexões de clientes, aceita uma conexão quando um cliente se conecta, recebe o nome de um arquivo do cliente, envia o conteúdo desse arquivo de volta ao cliente e, em seguida, fecha a conexão.

A função *send_file(),* recebe um arquivo (FILE *fp), um descritor de socket (int sockfd), e o tamanho do buffer a ser usado. Lê o conteúdo do arquivo em partes usando fgets e envia cada parte para o cliente usando send.

Já a *main()* cria um socket TCP, configura as informações do servidor (endereço IP e porta), liga o socket a uma porta específica, coloca o servidor no modo de escuta e aceita uma conexão de cliente. Depois, recebe o nome do arquivo a ser solicitado pelo cliente e abre o arquivo para leitura. Por fim, chama a função send_file para enviar os dados do arquivo ao cliente.

2.1.1- Lista de rotinas do Servidor

- *socket()* : cria um socket para comunicação.
- *listen()* : escuta as novas conexões.
- *accept()* : indica que a conexão foi aceita.
- *recv()* : recebe o nome do arquivo que o cliente deseja transferir.
- *fopen()* abre o arquivo solicitado para leitura.
- *send_file()* : envia o conteúdo do arquivo para o cliente.

- *close()* : fecha as conexões.

5- Implementação do Cliente

Nesta seção, será analisado o funcionamento e as características do módulo Cliente, bem como as funções relacionadas a ele.

5.1- Funcionamento do Cliente

O cliente basicamente se conecta a um servidor especificado, solicita um arquivo, recebe esse arquivo e calcula a taxa de transferência e o tempo total de transferência.

A função principal é a *receive_file()*; ela basicamente abre um arquivo para escrita, recebe dados do socket em um loop até que não haja mais dados para receber e escreve os dados recebidos no arquivo.

Já a *main()* basicamente estabelece uma conexão com o servidor, solicita ao usuário o nome do arquivo a ser transferido e o nome do arquivo de saída. Por fim, envia o nome do arquivo ao servidor. Também faz os cálculos de tempo e bytes transferidos.

5.1.1- Lista de rotinas do Cliente

- *receive_file()* : Recebe dados do soquete e escreve em um arquivo local.
- *gettimeofday()* : medi o tempo inicial e final da transferência.
- *fgets()* : obtém os nomes de arquivos do usuário.
- *strcspn()* : remove caracteres da nova linha.
- *socket()* : cria um socket.
- *connect()* : estabelece uma conexão com o servidor.
- *send()* : envia dados (nome do arquivo) ao servidor.
- *recv* : recebe dados do servidor.
- *close()* : fecha o socket após a transferência.

6- Testes e Comparações

Para os testes, medimos quantos bytes foram recebidos na função *receive_file()* e para o tempo colocamos gettimeofday antes de chamar a função *receive_file()* e um depois para o tempo final.

Começando o teste com um arquivo de 3mb e um buffer de 1024 bytes obtivemos os seguintes resultados :

[+]Total transfer time: 0.007424 seconds
[+]Total bytes transferred: 2864128 bytes
[+]Transfer rate: 385793103.45 Bps

Podemos ver que 3mb é muito pouco para ter uma noção pois como o tempo deu menos de 1 segundo a taxa de transferência estoura e fica muito alta. Vamos aumentar o tamanho do arquivo para 100 mb e manter o tamanho do buffer. Com essa configuração obtivemos os seguintes resultados :

[+]Total transfer time: 0.243326 seconds
[+]Total bytes transferred: 102465430 bytes
[+]Transfer rate: 421103499.01 Bps

Ainda o tempo é muito baixo, e a taxa de transferência continua bem alta, bem próxima à do primeiro teste. Podemos ver que com esse tamanho de buffer (1024 bytes) a taxa de transmissão irá se manter muito próxima a **400 mb/s** independentemente do tamanho do arquivo.

Para ter uma noção de como o tamanho do buffer influencia a taxa vamos diminuir pela metade agora, ou seja 512 bytes. Vamos testar primeiro com o arquivo de 100mb :

[+]Total transfer time: 0.430611 seconds
[+]Total bytes transferred: 102468608 bytes

[+]Transfer rate: 237960962.45 Bps

Como esperado, o tempo e a taxa também praticamente caíram para a metade com o buffer de 512 bytes em comparação ao de 1024 bytes. Com 1024 bytes a taxa foi próxima de **400 mb/s** e com 512 bytes próxima aos **200 mb/s .**

7- Instruções de uso

Instruções de compilação:

Para a compilação, basta compilar os 2 arquivos, servidor.c e cliente.c um de cada vez, ou usar o utilitário *make* presente na pasta para compilar todos.

Para a execução, primeiro é necessário executar o servidor, da seguinte forma : “./servidor <porta> <tam_buffer>“ . Logo depois em outro terminal, executar o cliente : “./cliente 127.0.0.1 <porta> <tam_buffer>“ . O cliente irá solicitar um nome de arquivo para ser recebido, esse arquivo deve estar na pasta root, logo depois solicitará um nome para um arquivo de saída, os dados arquivo que foi lido e enviado pelo servidor serão colocados nesse arquivo. É recomendado colocar a mesma extensão do arquivo que foi solicitado para sair no mesmo formato e nomes diferentes se não o arquivo original será reescrito.

8- Conclusão

Neste trabalho prático, desenvolvemos e analisamos um par de programas cliente-servidor que operam sobre o protocolo TCP, implementando transmissões unidirecionais e comunicação do tipo requisição-resposta. A implementação do servidor envolve a aceitação de conexões de clientes, recebimento do nome do arquivo solicitado, leitura e envio do conteúdo desse arquivo de volta ao cliente.

Ao realizar testes, observamos a influência do tamanho do buffer na taxa de transferência e no tempo total de transmissão. A análise comparativa entre diferentes tamanhos de buffer revelou uma relação direta entre o tempo, a taxa de transferência e o tamanho do buffer. Um tamanho maior de buffer resultou em uma

taxa de transferência mais alta, enquanto um tamanho menor resultou em uma taxa mais baixa.

A escolha do tamanho do buffer torna-se crítica em cenários de transmissão de arquivos, pois pode impactar significativamente o desempenho da aplicação. Para garantir eficiência, é importante adaptar o tamanho do buffer conforme as necessidades específicas do ambiente de comunicação.