Universidade Federal de Alagoas

Aluno: João Victor dos Santos Araujo

Aluno: Rafael Luiz dos Santos

Disciplina: Redes de Computadores 1 **Data:** 09/06/2022

Relatório de desenvolvimento de um chat em C++ que possui apenas uma sala de conversa

Introdução

O objetivo deste relatório é descrever o percurso, causas e problemas enfrentados durante a implementação da aplicação com nome de Chat++, que é executada via terminal do Windows 10 sob as condições impostas pelo professor da disciplina, Leandro Melo de Sales, e conselhos da monitora da disciplina, Paloma. Sendo este trabalho parte da nota da Avaliação Bimestral 2.

Como compilar e executar

Vide o arquivo README.md no repositório.

A biblioteca Winsock e a lógica do Chat++

A linguagem escolhida foi a C++, que nativamente não possui bibliotecas e recursos para desenvolvimento de Sockets. Sendo assim necessária uma biblioteca externa para utilização de socket. Como focamos em sistemas Windows, utilizamos a Winsock, mas a alternativa para sistemas Linux seria a biblioteca Socket.

Por meio da Winsock foram criados os sockets para o chat. Criamos um conjunto de sockets para aplicação: no máximo 6 para o servidor e um para o cliente. Para que isso ocorra, inicialmente criamos um objeto WSAData e em uma struct da Winsock explicitamos o tipo de IP utilizado (IPv4 no nosso caso), o protocolo de transporte e demais informações. Em seguida realizamos o bind entre o socket e a porta delimitada a partir do argumento de entrada do programa. Após isso, o socket pode ser utilizado livremente. É importante ressaltar que a cada etapa desde o início da criação do socket, efetuamos as verificações se ocorreu algum problema no sistema por meio de condicionais. Isso vale tanto para servidor quanto para cliente.

A partir deste ponto as implementações se diferem entre cliente e servidor.

Cliente

Na parte do cliente existem duas threads, sendo cada uma responsável pelo envio e recebimento de dados. O envio de dados ocorre por meio da função sender(), que envia os dados da aplicação (juntamente com o nome do usuário que está utilizando a aplicação) para

o servidor e, da mesma forma, o recebimento de dados ocorre por meio da função receiver() que recebe dados enviados a partir do servidor. Com a função main() (padrão do C++) pode-se dizer que toda a aplicação do lado do cliente tem em execução três threads.

Servidor

Já na parte do servidor, existe o socket do servidor, e um array de sockets servindo de uma abstração dos clientes (no caso, tais sockets permitem uma maior facilidade no envio e recebimento de dados a partir do servidor) no lado do servidor. Após a criação do socket do servidor instanciamos o array de sockets dos clientes, e efetuamos um loop para ouvir requisições de conexão ao servidor e aceitá-las (caso o limite de 5 clientes não seja atingido), para em seguida lançar a thread responsável por enviar e receber mensagens de tal cliente.

Nestas threads, os dados enviados pelo cliente são recebidos e encaminhados aos demais clientes conectados ao servidor. Ao total, podem haver até no máximo 6 threads no lado do servidor. E por fim, efetuamos o join das threads, esperando a desconexão dos clientes e limpamos a memória utilizada nos sockets, a fim de encerrar a aplicação no lado do servidor.

Problemas enfrentados

Inicialmente ocorreu um problema com um ponteiro responsável por encerrar a aplicação do cliente, pois este ponteiro era muitas vezes acessado e isso causava conflito na lógica de encerramento. Foram feitas pesquisas sobre o uso de “ponteiros inteligentes” para solucionar este problema, mas no fim a implementação final não precisou disso. Uma possibilidade com a programação assíncrona também surgiu durante o desenvolvimento, mas descobriu-se que esta solução era demasiada complexa e uma abordagem menos rebuscada resolvia a questão do uso de threads (por mais que a abordagem não usada pudesse economizar mais recursos de memória). Por fim, para resolver o problema do encerramento do lado do cliente bastou-nos apenas checar por meio de uma condicional se o cliente tinha digitado o comando “exit()” para sair do chat.

Conclusão

Programar sob essas condições: usando a linguagem C++, utilizando-se de threads e sockets; foi-nos relativamente árduo devido a inexperiência com sockets e threads, porém fora valiosíssimo como experiência, pois abriu as portas para perguntas nos problemas enfrentados que podem ser melhor exploradas observando-se a documentação da Winsock e demais textos sobre programação paralela e assíncrona.