Relatório Projeto Redes de Computadores A
Projeto:
Chat

Luiz Vinícius dos Santos Ruoso RA: 18233486,
Victor Felipe dos Santos RA: 18117820,
Victor Luiz Fraga Soldera RA: 18045674.

1. Introdução

Neste projeto foi proposta a implementação de um chat na linguagem de programação C. Com isso, para o desenvolvimento foram utilizados diversos conceitos de Sistemas Operacionais, Redes (TCP e sockets), e estrutura de dados para organização e construção de funcionalidades.

2. Desenvolvimento

2.1. Ambiente utilizado

Para o desenvolvimento utilizou-se o Subsistema Windows para Linux (WSL) que possibilitou a utilização de funções do compilador GCC no sistema Windows, como por exemplo a flag -pthread, que adiciona suporte ao multithreading utilizando a biblioteca de threads do sistema POSIX. Para o desenvolvimento do código houve a utilização da IDE (Ambiente de Desenvolvimento Integrado) Visual Studio Code.

2.2. Funcionamento

Como escolha de implementação, optamos pelo protocolo de comunicação TCP devido ao sistema de handshaking. Desta forma, há a garantia de que a mensagem chegará ao destinatário sem percas durante o envio. Além disso, escolheu-se a comunicação entre cliente e servidor, porque desta há o controle de envios e recebimento de mensagens e arquivos, de forma que, sem o servidor, o usuário não possue acesso ou controle a estes comandos.

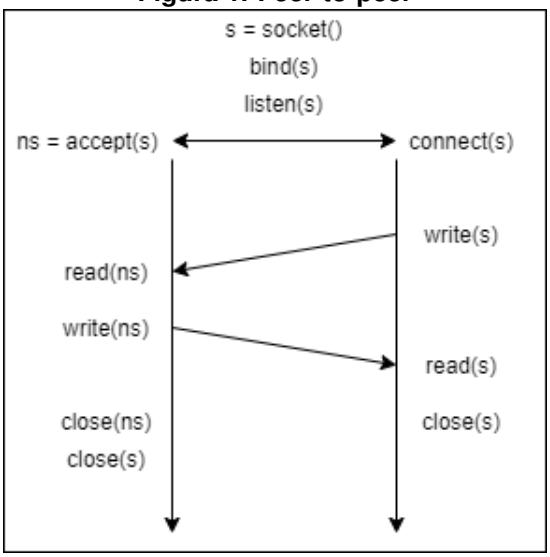
Para o envio de mensagens, optamos por definir uma estrutura de dados baseada em structs, por conterem diversos tipos e facilitarem o acesso ao conteúdo desejado. Todas as structs utilizadas são explicitadas abaixo:

```
struct{
char content[1024];
int nBytes;
}typedef charContentOps;

struct {
int operation;
charContentOps username;
```

```
charContentOps message;
char ip[15];
charContentOps userDestiny;
} typedef msg;

struct {
int statusCode;
int operation;
//char message[1024];
msg payload;
} typedef serverResponse;

struct {

int  *socket;
char username[1024];
int portNumber;
char ip[15];

}typedef user;


struct{
char nameFile[40];
long long blockSize;

}typedef fileTransfer;


struct{
int i;
char message[1024];
char from[1024];
}typedef messageReceived;
```

Para a comunicação peer-to-peer, foi realizado o desenvolvimento conforme a imagem abaixo:

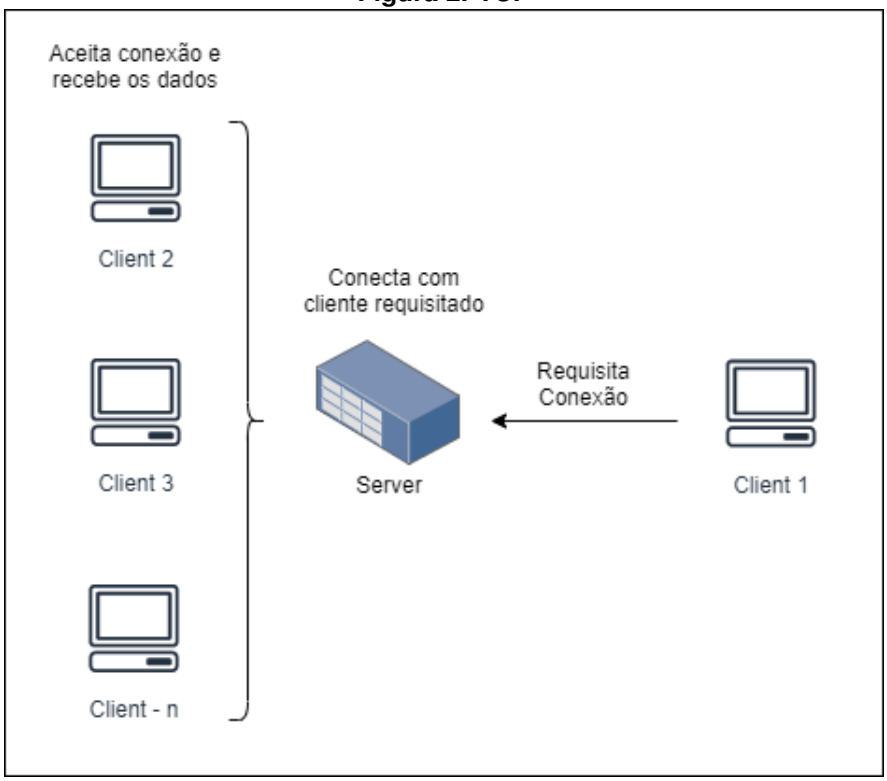
Figura 1. Peer-to-peer



Na figura acima, conecta-se, um usuário diretamente ao outro. De forma que os clientes funcionem também como servidor, e cada um dos clientes podem escrever e/ou ler conteúdo dos sockets que são criados para comunicação, quando demandado pelo servidor.

Para o sistema de Chat, baseado no protocolo TCP, realizou-se a implementação de um servidor como intermediário, conforme a imagem abaixo:

Figura 2. TCP



Desta forma, toda troca de informações entre clientes é tratada pelo servidor, desde de envio de mensagens simples, envio para usuários offline e solicitação de envio de arquivos, utilizando do conceito Peer-to-Peer.

2.3. Testes realizados

Durante a realização do projeto, o problema mais encontrado foi a necessidade de sempre limpar o valor de variáveis durante envios e recebimentos. Desta forma, durante as trocas de informações entre o cliente e o servidor foi necessário garantir que as mensagens trocadas não fossem alteradas, para strings, utilizamos a função *strncmp()* para garantir que o tamanho da string seja sempre fixo. E, para garantir que as váriaveis não tivessem conteúdo nenhum na atribuição de valores, usou-se a função *bzero()*.

Para o teste de envio de mensagens entre clientes passando pelo servidor, houve o envio de diversas mensagens entre os clientes, de forma, que toda vez que houvesse o envio de mensagens, imprimia-se a mensagem do lado do servidor e do lado do destinatário, confirmando, assim que as mensagens são entregues e tratadas de maneira correta.

Além disso, houve a utilização de mutex, durante o cadastro e envio de mensagens, para evitar problemas de starvation e deadlock.

Para os testes envolvendo arquivos, realizou-se a verificação de escrita e criação de forma manual dentro do diretório especificado. Para a leitura, foi impresso o conteúdo no terminal para verificação. Para garantir a segurança e confiabilidade dos dados escritos, fechavasse os arquivos após a leitura e escrita. Outros casos de teste, incluiam envio de textos e variações de tamanho de buffer enviado, as conferencias para consistencia incluiam verificar legibilidade, espaços e tabulações adicionais, além de qualquer pertubação no texto original.

3. Conclusão

Durante o trabalho utilizamos majoritariamente informações que foram obtidas através de experimentos anteriores, e também de livros e documentações. Vários componentes foram aproveitados dos experimentos da disciplinas de Sistemas Operacions e Estrutura de dados. Por fim, obtivemos sucesso na realização do projeto, de forma que, todas as funcionalidades implementadas realizam suas respectivas funções de acordo com a descrição do que foi proposto.

4. Referências

Documentação de funções

Redes de Computadores e a Internet: Uma Abordagem Top-Down