Especificação do Trabalho da Disciplina
Redes de Computadores
SSC-0142

Prof. Kalinka R. L. J. Castelo Branco / Estevam F. Arantes

Introdução

Ao longo deste projeto serão implementadas as diversas partes que compõe um cliente e servidor *IRC*, ou *Internet Relay Chat*, amplamente utilizado na década de 90 e até hoje por alguns grupos de computação.

A implementação a ser feita é uma adaptação das especificações dadas pelo RFC 1459, que define o IRC.

O protocolo IRC tem sido desenvolvido em sistemas utilizando o protocolo TCP/IP e é um sistema que **deve** suportar múltiplos clientes conectados em um único servidor, realizando a multiplexação dos dados recebidos por eles.

Aconselhamos que leia a especificação de todos os módulos antes de implementar os primeiros, dado que algumas escolhas de projetos podem facilitar os últimos. O código deste projeto deverá ser em linguagem C ou C++ padrão, compilado em gcc.

O primeiro módulo terá pontuação de no máximo 2 pontos, o módulo 2 de no máximo 4 pontos e o módulo 3 de no máximo 3 pontos. Serão dados até 2 pontos extras para bônus feitos no trabalho, de modo que o mesmo tenha valor máximo de 11 pontos.

O projeto será feito em grupos de até 3 pessoas.

É solicitado que seja mantido o mesmo grupo para os 3 módulos do projeto.

O que deve ser entregue

Um arquivo zip com todo o sistema com os códigos a cada módulo, um arquivo Makefile para compilar e executar o código e um README com os nomes dos membros do grupo, a versão do sistema operacional (Linux) e compilador utilizados, além de quaisquer instruções adicionais que julgarem necessárias.

OU

Um repositório do github com todos os arquivos acima.

Módulo 1 - Implementação de Sockets (entrega 26/04/2020)

Especificação da implementação

Imagine uma aplicação *online* como um *chat* ou um jogo. Como a comunicação dela é feita?

Neste módulo será desenvolvido uma aplicação para a comunicação entre Clientes na linguagem C ou C++, **sem o uso de bibliotecas externas**.

Para isso, deve ser implementado um *socket*, que define um mecanismo de troca de dados entre dois ou mais processos distintos, podendo estes estar em execução na mesma máquina ou em máquinas diferentes, porém ligadas através da rede. Uma vez estabelecida a ligação entre dois processos, eles devem poder enviar e receber mensagens um do outro.

Na aplicação a ser entregue devem ser implementados *sockets* TCP que permitam a comunicação entre duas aplicações, isso de modo que o usuário da aplicação 2 possa ler e enviar mensagens para o usuário da aplicação 1 e vice-versa.

O limite para o tamanho de cada mensagem deve ser de 4096 caracteres. Caso um usuário envie uma mensagem maior do que isso ela deverá ser dividida em múltiplas mensagens automaticamente.

Módulo 2 - Comunicação entre múltiplos clientes e servidor (entrega 25/05/2020)

Para este módulo, utilizando parte do que foi feito para o primeiro, deverá ser implementado um modelo de clientes-servidor que corresponda a um *chat*, de modo que uma mensagem de um cliente deverá ser enviada para todos os clientes passando por uma aplicação servidora.

Cada cliente deverá ter um apelido definido arbitrariamente, que para este módulo possa simplesmente ser um inteiro (*index*) ou uma *string* qualquer. As mensagens aparecerão para todos os usuários (inclusive para quem enviou) no formato **apelido: mensagem**. Cada mensagem será separada por um '$\backslash n'$. De forma semelhante ao primeiro módulo, cada mensagem deverá ser limitada por 4096 caracteres.

Para fechar a sua conexão, um cliente poderá enviar um comando de saída (/quit) ou um sinal de EOF (Ctrl + D).

Comandos a serem implementados do cliente para o servidor

- $/connect$ - Estabelece a conexão com o servidor;
- $/quit$ - O cliente fecha a conexão e fecha a aplicação;
- $/ping$ - O servidor retorna "pong"assim que receber a mensagem.

Pontos Importantes

O servidor deverá checar se os clientes receberam as mensagens. Caso eles não tenham recebido a mensagem ela deve ser enviada novamente. Após 5 tentativas falhas o servidor deve fechar a conexão com o cliente.

Não esqueça de tratar *deadlocks* e possíveis problemas que possam surgir com o uso de *threads*.

Deve ser necessário lidar com SIGINT (Ctrl + C) no *chat*, para isso a sugestão é adicionar um *handler* que ignore o sinal ou imprima alguma mensagem.

Módulo 3 - Implementação de múltiplos canais (entrega 22/06/2020)

Nesta etapa deverá ser feita a implementação de múltiplos canais e a função de administradores de canais.

Ao abrir a aplicação, o usuário deverá, por meio do comando *join*, especificar em qual canal ele quer se conectar. Caso este canal não exista ele deverá ser criado e o primeiro usuário a se conectar se torna o administrador do canal.

O nome de um canal deverá seguir as restrições apresentadas no RFC-1459.

Um administrador de um canal tem a permissão de usar os comandos *Kick*, *Mute* e *Whois* em usuários.

Ao abrir a aplicação pela primeira vez um usuário deverá definir um apelido por meio do comando *nickname*, limitando o nome do usuário a 50 caracteres ASCII.

Comandos a serem implementados

Além dos comandos apresentados no módulo anterior, devem ser implementados os seguintes comandos:

- $/join\ nomeCanal$ - Entra no canal;
- $/nickname\ apelidoDesejado$ - O cliente passa a ser reconhecido pelo apelido especificado;
- $/ping$ - O servidor retorna "pong"assim que receber a mensagem.

Comandos apenas para administradores de canais:

- $/kick\ nomeUsurio$ - Fecha a conexão de um usuário especificado
- $/mute\ nomeUsurio$ - Faz com que um usuário não possa enviar mensagens neste canal
- $/unmute\ nomeUsurio$ - Retira o *mute* de um usuário.
- $/whois\ nomeUsurio$ - Retorna o endereço IP do usuário **apenas para o administrador**

Item bônus

É possível que canais sejam apenas para usuários convidados. Leia no RFC como isso funciona e implemente esta funcionalidade, caso se interesse por bônus na pontuação final do trabalho :-).