Publish/Subscribe

Redes de Computadores

Gabriel Victor Carvalho Rocha

gabrielcarvalho@dcc.ufmg.br - 2018054907

1. Introdução

O projeto se resume à construção de um modelo cliente/servidor, onde o cliente pode se inscrever em determinadas tags ou enviar mensagens. Quando algum cliente envia uma mensagem contendo uma hashtag (#) seguida de uma tag, o servidor irá mandar essa mensagem para todos clientes que a possuem cadastrada previamente.

2. Implementação

O primeiro desafio para implementação do trabalho, foi entender como funcionava a biblioteca pthread, já que nunca a tinha utilizado. Logo em seguida, tive problemas em usar a estrutura de dados map, que por algum motivo não compartilhava os mesmos dados entre as threads do servidor.

Decidi então utilizar a estrutura de dados vector<subscription*>, onde cada subscription possui o socket do cliente e um set de tags em que está cadastrado, não tendo o problema descrito anteriormente no uso do map.

O cliente possui duas threads, uma para utilizar recv e outra para utilizar send, isso foi necessário já que mesmo que ele esteja digitando uma mensagem, ele deve estar disponível para receber possíveis mensagens vindas do servidor. Toda vez que um cliente se conecta ao servidor, ele é adicionado a esse vector com um set de tags vazio, o servidor então recebe as mensagens vindo do socket do cliente até que o cliente não se desconecte (count == 0) ou até que o último caracter recebido seja um '\n', isso por si só garante também que não sejam enviadas mensagens com mais de 500 bytes além de cobrir o caso de mensagens particionadas, já que o loop só se encerra após ter recebido o caractere de finalização (\n).

Após o recebimento da mensagem, armazenada no buf, são validadas condições que farão com que o programa se encerre, como por exemplo: Caracteres fora do ASCII; Uma segunda checagem no buf para verificar que o último caractere não é '\n' ; e se count foi 0, significando que o cliente se desconectou. Em seguida deleta-se o cliente, retornando 1 e finalizando sua thread através do break.

Agora, é criado um vetor auxiliar de char[BUFSZ] chamado tmpbuf, aqui trataremos o

caso de múltiplas mensagens em um único recv. O buf será percorrido por completo, até encontrar o primeiro '\n', após ser encontrado, é criada uma substring começando de start e contendo (i - start) + 1 caracteres, onde é verificado se o tmpbuf possui a mensagem '##kill", fazendo com que o servidor seja desligado, caso contrário, a mensagem é tratada pela função handle_buf, que será verificada se é uma mensagem de inscrição (contendo "+"), de desinscrição (contendo "-") ou se é de fato uma mensagem que deve ser enviada para os clientes que possuírem a tag contida nela. No primeiro caso o cliente é cadastrado (se ainda não for inscrito) e no segundo caso o cliente é descadastrado (se estiver inscrito). No terceiro, inicialmente é extraído da mensagem todas as tags, depois encontramos todos os clientes que possuem pelo menos uma dessas tags previamente cadastradas, aqui retornamos 1 para identificar como mensagem.

Por fim, enviamos o tmpbuf pela função handle_send, que mandará uma única mensagem para o cliente atual caso for inscrição/desinscrição, senão, significa que foi uma mensagem com tags, que será enviada para todos clientes encontrados na função anterior.

Algumas decisões que não estavam claras na especificação do trabalho foram tomadas, como: Um cliente que manda uma mensagem contendo uma tag em que ele mesmo possui interesse, não receberá esta mensagem; A inscrição e desinscrição em uma tag não pode conter nenhum espaço; Caso sejam enviadas duas ou mais mensagens em um único recv, se qualquer uma delas possuírem mensagens inválidas (ou seja, não está na ASCII), o cliente é desconectado e nenhuma mensagem é enviada, e caso o último caractere do buf não for '\n', o servidor irá esperar até que seja enviada uma quebra de linha para enviar todas as outras.

3. Conclusão

Dessa forma, o trabalho foi uma experiência de muito aprendizado, pois foi possível entender como é possível fazer uma comunicação cliente/servidor utilizando apenas a linguagem C/C++ com a biblioteca POSIX e a entender melhor sobre o funcionamento de multi-threads. Casos como múltiplas mensagens em um único recv e mensagens incompletas deram um pouco de trabalho, mas no fim foi possível implementá-las corretamente.