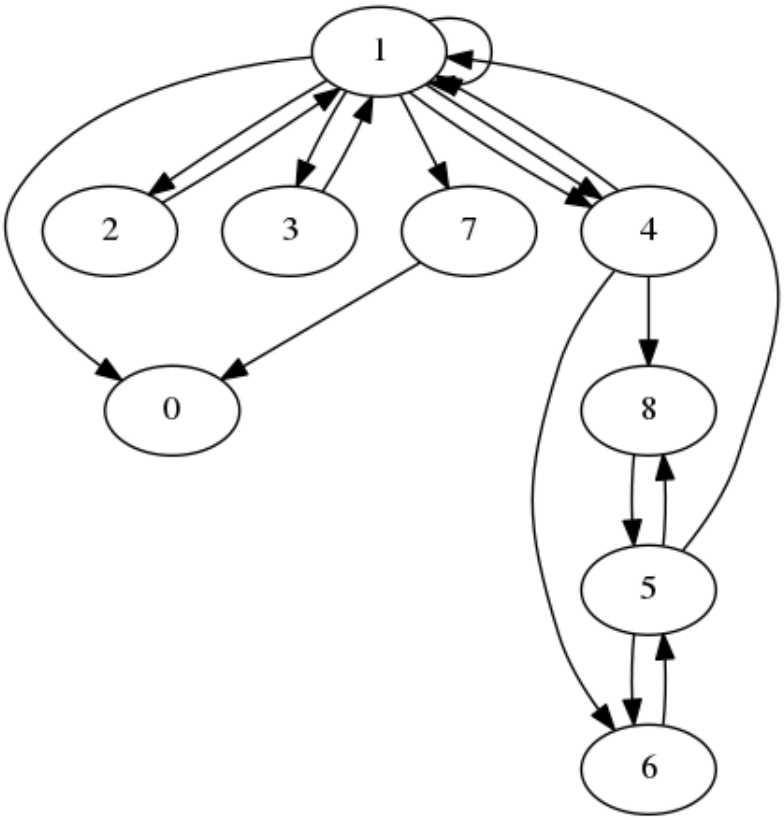
ep1redes

Yoshio Mori

September 2015



1 Introdução

O ep1irc pode ser visto como o autômato acima. Cada vértice representa um estado do servidor e cada aresta suas trasições.

Segue abaixo o significado de cada estado:

- 1 - Leitura das mensagens enviadas pelo cliente;
- 2,3,6,7,8 - Interpretação do comando nick, list join quit e part respectivamente;
- 4,5 - Leitura dos parametros do comando;

- 0 - Estado final;

2 ep1irc

O nome do servidor é ep1irc. Após a inicialização, o servidor entra no ciclo descrito por esse autômato. O estado inicial desse autômato é o 1, nesse estado é feita a leitura da mensagem do cliente e também a segmentação da mensagem em duas partes, comando e parâmetros. Inicialmente o único comando disponível é o nick, que faz o registro do nickname do cliente. Os demais comandos estarão disponíveis após o registro.

A interpretação do comando NICK, estado 2, só é possível com parâmetros válidos, um parâmetro válido para esse comando deve ser uma string de comprimento de no máximo 9 caracteres terminadas com null. Após a verificação de validade o nickname é associada ao cliente. Se o cliente já tem um nickname associado o servidor faz atualização do nickname. Cada cliente tem um único nickname associado a ele.

A interpretação do comando LIST, estado 3, é feita simplesmente enviando ao cliente a lista dos canais disponíveis, numa mensagem formatada.

A interpretação do comando JOIN, estado 6, consiste em pegar o parâmetro e distinguir se representa um canal válido, após isso a inserção do cliente nesse canal é feita. A inserção só é feita se o cliente já não foi inserido.

A interpretação do comando JOIN, estado 7, remove o cliente de todos os canais que o contém, após isso é feita a remoção do cliente do servidor. No automato podemos ver a transição do estado 7 para o 0, representando esse evento.

O comando PART, estado 8, é feita verificando se o parâmetro é válido e assim o cliente é removido do canal associado.

Os comandos JOIN e PART recebem uma lista de parâmetros, essa lista precisa ser interpretada, os estados responsáveis por fazer a leitura dos comandos são o 4 e 5. O servidor, ao interpretar um desses comandos, faz a transição para o estado 4, que é o estado onde é feita a validação dos parâmetros a leitura do primeiro parâmetro e a configuração das transições para possibilitar a leitura e interpretação de cada um dos parâmetros, onde a leitura do resto dos parâmetros é feita no estado 5. No Autômato podemos ver as transições (5,6) e (5,8) representando leitura e processamento do comandos JOIN e PART respectivamente. Após todos os parâmetros terem sido processados ocorre a transição (5,1).

3 Implementação

O servidor foi implementado usando múltiplos processos, cada processo faz o gerenciamento de um único cliente conectado ao servidor. O cliente se conecta ao servidor por meio de sockets associadas a conexões TCP/IP. O servidor faz uso do protocolo IRC.

Um vetor de strings representando os nicknames. Esse vetor deve estar em uma memória compartilhada para que todos os processos possam ter acesso. Para manipulação, inserção, busca e remoção, esse vetor tem uma interface de funções associadas a ele, que se encontra no arquivo clients.h.

O autômato é implementada fazendo uso de switch-case aninhada num laço. Cada estado possui um case associado e de mesmo número aos descritos aqui. As transições (1,2), (1,3), (1,4) e (1,7) é feita com uma sequencia de if-else onde cada if faz uma comparação de uma string de um comando válido com a string do comando envida na mensagem do usuário. O cliente registrado possui seu nickname associado ao processo que é uma string, esse nickname é usada para impedir que clients não registrados tenham acesso aos comandos específicos.

O case 4 do switch faz a validação do parâmetro, a leitura inicial da função strtok da biblioteca string.h e a configuração da variável send_to que indica o estado x da transição (5,x) para x = 6,8. A configuração da variável é necessária para indicar o comando, JOIN ou PART, associada aos parâmetros sendo lidos no estado 5.

Um vetor de clientes representa um canal. Cada canal tem um ponteiro indicando o primeiro endereço livre desse vetor. A cada novo cliente inserido esse ponteiro incrementa. Os vetores nesse servidor tem um limite máximo definido com MAXCLIENTS. Para inserir um cliente num canal é necessário fazer uma busca no canal por esse cliente para garantir que o canal não tenha o mesmo cliente inserido duas vezes.

4 Conclusão

Os itens:

- Aceitar a conexão de vários clientes simultaneamente;
- Definir o nickname do usuário;
- Listar os canais;
- Entrar em um canal;
- Sair do canal;
- Desconectar do servidor;

foram implementados. Os itens:

- Enviar mensagem para todos no canal;
- Transferir arquivo;
- 3 novos comandos;

ficaram pendentes. Além do controle de concorrência e liberação de memória que não deram tempo de implementar.