**ROTEADOR.H**

A biblioteca roteador.h contém as funções para manipular os roteadores do netmap. Esta é dividida em funções principais, auxiliares internas, que serão usadas somente dentro da própria biblioteca para deixar o código mais legível e para o reuso nas funções, e auxiliares externas, que serão usadas em outras bibliotecas, como na terminal.h.

**Estrutura do roteador (struct roteador)**

|  |  |
| --- | --- |
| char\* nome | LsRot\* rotConectados |
| char\* operadora |

**Estrutura da célula de uma lista de roteadores (struct celRot)**

Lista de roteadores conectados ao roteador (struct lsRot)

Nome da operadora

Nome do roteador

Em vários problemas é necessário buscar a célula de um roteador em uma lista, por exemplo, ao remover um roteador da lista do netmap e ao desconectar um roteador do outro, por isso, se faz útil uma função para realizar a busca e com isso é necessário saber qual a célula anterior ao roteador encontrado a fim de fazer o desencadeamento daquela célula, logo, para facilitar, foi pensado em uma lista duplamente encadeada.

Célula anterior

(struct celRot)

Célula seguinte (struct celRot)

|  |  |  |
| --- | --- | --- |
| CelRot\* ant | Roteador\* rot | CelRot\* prox |

Tipo roteador (struct roteador)

**Estrutura da lista de roteadores (struct lsRot)**

Foi pensada em uma lista com sentinela.

Última célula da lista

Primeira célula da lista

|  |
| --- |
| CelRot\* prim |
| CelRot\* ult |

**CadastraRoteador**

A função irá adicionar um roteador à lista de roteadores do netmap. Os argumentos são um ponteiro para o nome do roteador (char\*), um ponteiro para o nome da operadora do roteador(char\*) e o ponteiro pra lista de roteadores do netmap (LsRot\*). Não há retorno na função. Ao final, o roteador deve ter sido criado, e a célula que aponta para ele encadeada na lista de roteadores do netmap.

Primeiro, cria-se um tipo roteador, o qual armazena o nome e a operadora do roteador e inicializa a lista de roteadores conectados, sendo ela vazia. Utiliza-se uma função auxiliar interna para fazer isso, chamada criaRoteador.

Depois cria uma célula que apontará para o roteador criado. E então encadeia essa na célula na lista de roteadores do netmap. Os roteadores cadastrados no netmap serão adicionados sempre ao final da lista. Utiliza-se uma função auxiliar interna chamada EncadeiaRoteador para fazer isso, que será utilizada em outra função.

**RemoveRoteador**

A função irá remover um roteador existente da lista de roteadores do netmap. Não há retorno e os argumentos são o nome do roteador (char\*), a lista de roteadores do netmap (LsRot\*) e a lista de terminais do netmap, que é do tipo void\* (mas na verdade é CelTerm\*) como uma forma de não ter erros na compilação já que a roteador.h inclui a terminal.h e vice-versa. Ao final, o roteador deve ter sido removido do netmap e desconectado de todos os roteadores e terminais que antes estava.

Primeiro a função busca a célula do roteador na lista de roteadores do netmap. Caso encontre-o, é preciso desconectá-lo de todos os outros roteadores ao qual está conectado, nisso é feito um loop que percorre a lista de roteadores conectados chamando a função DesconectaRoteadores.

Depois disso, é necessário desconectar os terminais desse roteador, utiliza-se a função DesconectaRoteador que está definida na terminal.h. O roteador é desencadeado utilizando a função DesencadeiaRoteador da lista de roteadores do netmap e então liberado usando a função auxiliar LiberaTipoRoteador.

Caso o roteador não exista no netmap, uma mensagem de erro é escrita no log.txt e a função encerrada.

**ConectaRoteadores**

A função irá conectar dois roteadores existentes no netmap. Não há retorno e os argumentos são o nome de cada roteador (char\*) e a lista de roteadores do netmap (LsRot\*). Ao final, cada roteador deve estar na lista de roteadores conectados um do outro.

Primeiro a função busca as células dos roteadores na lista de roteadores do netmap para ter acesso às listas de roteadores conectados e o tipo roteador de cada um. Caso encontre-os, novas células são criadas e apontam para os roteadores, cada uma. Note que não é criado um novo tipo roteador.

Então a célula que aponta para um roteador será encadeada na lista de roteadores do outro roteador e vice-versa usando a função EncadeiaRoteador.

Caso um ou os dois roteadores não existam no netmap, uma mensagem de erro é escrita no log.txt e a função encerrada.

**DesconectaRoteadores**

A função irá desconectar dois roteadores existentes e conectados no netmap. Não há retorno e os argumentos são o nome de cada roteador (char\*) e a lista de roteadores do netmap. Ao final, a lista de roteadores conectados de cada roteador não deve conter uma célula apontando para o roteador um do outro.

Primeiro a função busca as células dos roteadores no netmap para ter acesso à lista de roteadores conectados de cada um.

Depois é preciso buscar o roteador1 na lista de roteadores conectados do roteador2 com a função auxiliar BuscaRoteador para poder chamar a função DesencadeiaRoteador passando como argumento a célula encontrada e a lista de roteadores conectados do roteador2 e vice-versa. Para finalizar, as células são liberadas.

Caso um ou os dois roteadores não existam no netmap ou eles não estejam conectados, uma mensagem de erro é escrita no arquivo log.txt e a função encerrada.

**FrequenciaOperadora**

A função contará quantos roteadores do netmap são da operadora que é passada como entrada. Não há retorno, já que a quantidade é escrita diretamente no arquivo saida.txt. Os argumentos de entrada são o nome da operadora (char\*) e a lista de roteadores do netmap.

Um ponteiro do tipo CelRot\* percorre a lista de roteadores do netmap verificando qual a operadora é cada roteador e se for igual à operadora de entrada, então o contador acrescenta +1.

**funcaoBusca**

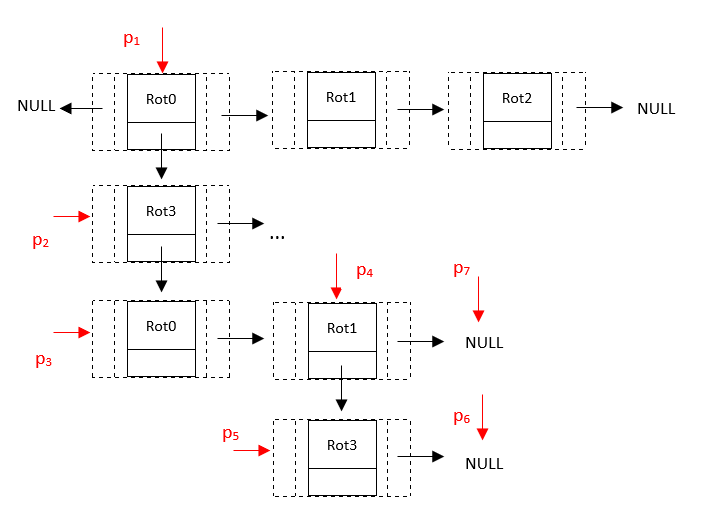
Outra função importante na roteador.h é a funcaoBusca que tem papel principal na função EnviarPacotesDados definida na terminal.h. Ela foi implementada no roteador.c porque é preciso manipular estruturas que são definidas somente na biblioteca roteador.h.

A função tem como argumento uma célula do tipo roteador (CelRot\*), o nome dor roteador que está buscando (char\*), um vetor vet que irá guardar o nome dos roteadores que já foram analisados e um contador i (int\*) que indica a primeira posição livre do vetor. O retorno é 1, caso o roteador seja encontrado e 0, caso contrário.

É usado um ponteiro p do tipo CelRot\* que irá apontar, inicialmente, para a primeira célula da lista de roteadores conectados do roteador de entrada.

Em um loop (irá parar quando p = NULL) são feitas verificações sobre o roteador que p aponta. Primeiro, se o nome do roteador está contido no vetor, usando uma função auxiliar chamada BuscaNomeVet. Caso esteja, p recebe o endereço da próxima célula, caso contrário, verifica se o nome do vetor é igual ao vetor que está buscando. Se não for o roteador, então guardará o nome desse no vetor e é chamada a funcaoBusca. Após a chamada, verifica se o retorno foi 1, se não, p recebe a próxima célula.

Um exemplo do funcionamento dessa função é dado pelo esquema a seguir. Ele mostra o caso em que o roteador é encontrado. O roteador que se busca é “Rot5”.



**p1:**

vet = {}

i = 0

**p2:**

vet = {“Rot0”}

i = 1

**p3, p4:**

vet = {“Rot0”, “Rot3”}

i = 2

**p5, p6, p7:**

vet = {“Rot0”, “Rot3”, “Rot1”}

i = 3

**p8:**

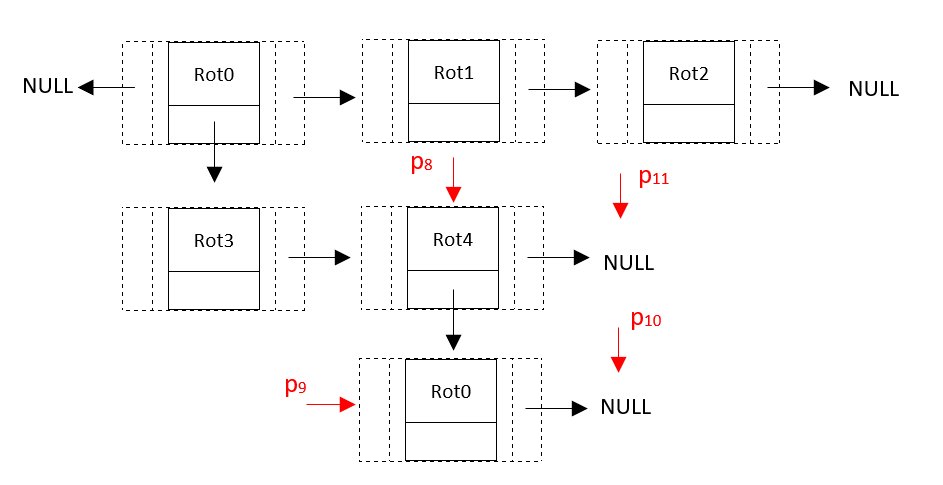
vet = {“Rot0”, “Rot3”, “Rot1”}

i = 3

**p9, p10, p11:**

vet = {“Rot0”, “Rot3”, “Rot1”, “Rot4”}

i = 4



**p12, p13:**

vet = {“Rot0”, “Rot3”, “Rot1”, “Rot4”}

i = 4

**p14, p15:**

vet = {“Rot0”, “Rot3”, “Rot1”, “Rot4”, “Rot2”}

i = 5

**p16:**

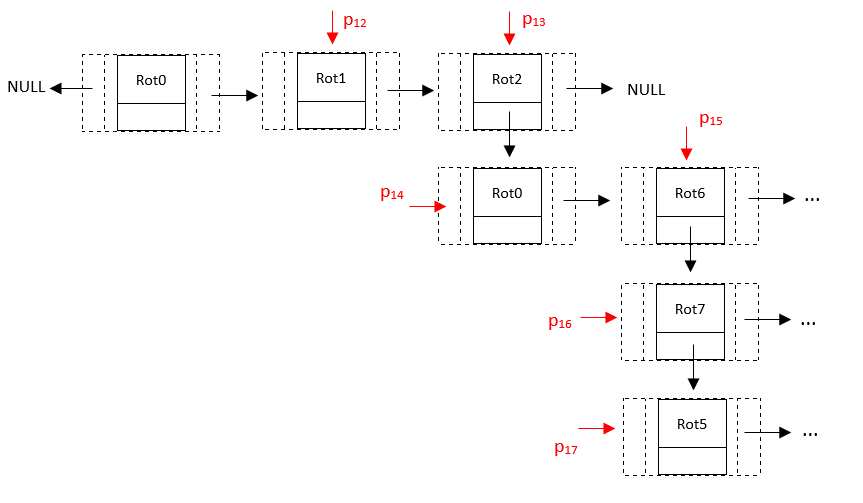
vet = {“Rot0”, “Rot3”, “Rot1”, “Rot4”, “Rot2”, “Rot6”}

i = 6

**p17:**

vet = {“Rot0”, “Rot3”, “Rot1”, “Rot4”, “Rot2”, “Rot6”, “Rot7”}

i = 7



***Obs:*** *desconsiderou-se as setas dos ponteiros ant e o tipo roteador foi colocado “dentro” da célula para facilitar a representação.*

Em p17, o roteador é encontrado, então a função encerra retornando 1. Observe que, mesmo se não encontrasse o roteador, a função pararia retornando 0. Foi pensado da forma a “varrer” todas as listas possíveis com o while e a chamada recursiva da função, mas evitando um loop infinito, utilizando o vetor.

**LiberaListaRot**

Função usada para liberar a lista de roteadores do netmap. Não há retorno e o argumento de entrada é uma lista de roteadores (LsRot\*). Ao final, toda a memória alocada para cada elemento e célula da lista deve ser liberada.

Em um loop, o roteador de cada célula é liberado usando a função LiberaTipoRoteador e então a célula que apontava pra ele também. Por fim, a estrutura da lista é liberada.