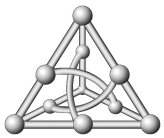

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Trabalho 08: Tabela de Roteamento IP

Redes de Computadores

1 Descrição

Você pode fazer este trabalho em dupla. Este trabalho pede para você implementar um tipo de tabela de encaminhamento IP para encontrar um roteador ou *host*. Essa busca é usada como parte do processo de encaminhamento (que você não precisa implementar). A tabela de roteamento deste trabalho é estática e é especificada como argumento da linha de comandos, logo, você não precisa implementar nenhum protocolo de roteamento, ou mesmo, a tabela de roteamento em si.

2 Tabela de Roteamento

Cada endereçõ IPv4 tem comprimento de 32 *bits*. Cada entrada na tabela de roteamento armazena:

1. um Destino, que é um endereço IP.
2. um Gateway ou Próximo *Hop*, que também é um endereço IP.
3. uma máscara, que é uma *string* de 32 *bits*, mas não é um endereço IP.
4. várias *flags*, que podem ser ignoradas neste trabalho.
5. a métrica, que para este trabalho pode ser ignorado.
6. uma interface, que é uma *string* arbitrária.

Vários roteadores usam métodos automatizados como os protocolos de roteamento ou DHCP para construir suas tabelas de roteamento. Neste trabalho, a tabela de roteamento será expecificada como uma série de argumentos de linha de comando no seguinte formato:

```
destino/gateway/máscara/interface
```

Cada destino, gateway e máscara são fornecidos na notação número-ponto suportada pelo `inet_aton`. A interface é uma *string* arbitrária.

Cada argumento desses é usado para criar uma entrada na tabela de roteamento na ordem que forem fornecidas na linha de comando.

Para procurar na tabela de roteamento, o IP usa um algoritmo chamado de *longest match*. De todas as entradas na tabela de roteamento que casariam com um padrão, aquele com a maior máscara é usado. Se existirem diversas entradas que casam com o padrão e tem máscara de mesmo tamanho, a primeira é usada.

Todos os *bits* '1' são uma máscara válida e estão a esquerda de todos os *bits* '0'. Logo, a maior máscara é também numericamente maior quando comparando inteiros não sinalizados de 32 *bits*.

Uma rota "casa" com um endereço IP quando seu destino E sua máscara são os mesmos que o endereço IP E a máscara, ou

```
(route[i].dest & route[i].mask) == (ip->dest & route[i].mask)
```

Ambas as máscaras totalmente com zero e totalmente com uns são legais. A primeira é a rota padrão, usada se nenhuma outra rota casar. A segunda é a rota para o *host*, usada somente se o destino do pacote é o mesmo destino que o da tabela de roteamento.

O primeiro parâmetro de todos é o endereço IP que vai casar (ser testado) com a tabela de roteamento.

3 Exemplos

```
$ ./rlookup 1.2.3.4 1.0.0.0/2.4.6.8/255.0.0.0/eth0 1.2.0.0/2.4.6.9/255.255.0.0/eth0
1.2.3.0/1.2.3.1/255.255.255.0/eth1
forwarding packet for 1.2.3.4 to next hop 1.2.3.1 over interface eth1
```

O endereço IP (1.2.3.4) casa com todas as três rotas que foram fornecidas, e aquele que tem a máscara mais longa é a terceira, que é a usada. Se o pacote fosse enviado para 1.2.5.6, nós teríamos:

```
$ ./rlookup 1.2.5.6 1.0.0.0/2.4.6.8/255.0.0.0/eth0 1.2.0.0/2.4.6.9/255.255.0.0/eth0
1.2.3.0/1.2.3.1/255.255.255.0/eth1
forwarding packet for 1.2.5.6 to next hop 2.4.6.9 over interface eth0
```

Agora, apenas as duas primeiras rotas casaram, logo, a segunda (a mais longa das duas) rota é a escolhida.

A ordem não importa se exister apenas uma única rota mais longa:

```
$ ./rlookup 1.2.5.6 1.2.0.0/2.4.6.9/255.255.0.0/eth0
1.0.0.0/2.4.6.8/255.0.0.0/eth0 1.2.3.0/1.2.3.1/255.255.255.0/eth1
forwarding packet for 1.2.5.6 to next hop 2.4.6.9 over interface eth0
```

A rota padrão é usada somente se não existir um casamento melhor

```
$ ./rlookup 1.2.5.6 0.0.0.0/1.2.3.1/0.0.0.0/eth1 1.2.0.0/2.4.6.9/255.255.0.0/eth0
forwarding packet for 1.2.5.6 to next hop 2.4.6.9 over interface eth0
```

```
$ ./rlookup 9.8.7.6 0.0.0.0/1.2.3.1/0.0.0.0/eth1 1.2.0.0/2.4.6.9/255.255.0.0/eth0
forwarding packet for 9.8.7.6 to next hop 1.2.3.1 over interface eth1
```

No caso de múltiplos casamentos mais longos, o primeiro deve ser usado

```
$ ./rlookup 1.2.5.6 1.2.0.0/2.4.6.9/255.255.0.0/eth0 1.2.0.0/1.2.3.1/255.255.0.0/eth1
forwarding packet for 1.2.5.6 to next hop 2.4.6.9 over interface eth0
```

Algumas vezes, não existe casamento nenhum:

```
$ ./rlookup 9.8.7.6 1.0.0.0/2.4.6.8/255.0.0.0/eth0 1.2.0.0/2.4.6.9/255.255.0.0/eth0
1.2.3.0/1.2.3.1/255.255.255.0/eth1
destination 9.8.7.6 not found in routing table, dropping packet
```

4 Resultados

Por favor siga os exemplos anteriores e relate os resultados do seu analisador de tabelas de roteamento (um pequeno texto explicativo).

Anexe o texto em .TXT ao seu código fonte com nome rlookup.c e envie tudo em um único .ZIP

5 Opções

Esse trabalho não precisa que você construa ou crie uma tabela de roteamento. Contudo, você é bem vindo a fazê-lo se assim desejar.

Másrcaras de rede podem ser especificadas de uma das seguintes duas formas: como descrito anteriormente (na notação decimal com pontos) ou usando a seguida pelo número de *bits* '1' mais significativos. Por exemplo, a rota padrão é 0, e a rota para um *host* é 32. Se você quiser, pode implementar o código para verificar se a entrada possui um caractere '.' (isso é, provavelmente, mais difícil do que você imagina) e se não (e o número estiver entre 0 e 32), usar o número correspondente como o número de *bits* 1 para ser colocado na máscara.

Se você fizer isso, tenha certeza que seu código funciona corretamente para os exemplos anteriores. Sua nota neste trabalho dependerá somente disso.