Trabalho 2 – Redes: Roteamento IP

Autores:

- Adauto Ferreira de Souza Neto – 2014.1905.055-9
- Vitor Santa Barbara Lira – 2016.1905.035-8

Proposta de trabalho:

Você desenvolverá um sistema simples de roteamento de mensagens em C, C++ ou Java, sem bibliotecas especiais, utilizando comunicação via protocolo UDP.

Dois programas devem ser desenvolvidos: um **emissor**, para envio de mensagens na rede, e um programa "**roteador**" que realiza o encaminhamento e das mensagens recebidas. Por simplicidade este último também fará o papel do destinatário, exibindo a mensagem recebida quando for o caso. Serão inicializados vários programas roteadores e emissores. (Obs: caso useJava com múltiplas janelas na interface gráfica, pode-se fazer apenas 1 programa).

Cada mensagem será enviada a algum roteador, que a repassa para o próximo roteador, e assim sucessivamente até ser entregue ao destinatário.

Solução encontrada:

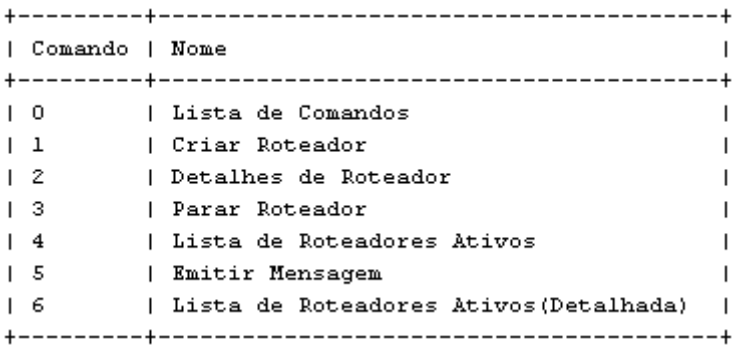
Foi desenvolvido um programa em Java, utilizando-se da IDE NetBeans8.2, com a funcionalidade dos programas **emissor** e **roteador**. Utilizando-se threads, o programa cria roteadores com o comando **roteador** e faz envio de mensagens no padrão UDP utilizando o comando **emissor**. Mais adiante neste relatório o formato completo do comando será explicitado.

Todos os pacotes são codificados para String e enviados da seguinte forma: **endereçoDeOrigem---endereçoDeDestino---PortaDoRoteador**. Ao ser recebido por um thread do tipo roteador, esse pacote é decodificado e processado.

Funcionalidades Implementadas:

Ao iniciar a execução do programa, é exibido um quadro contendo as 6 funcionalidades implementadas no sistema. São elas:

0. **Help:** Esse comando exibi o quadro contendo as funcionalidades do programa. Sempre que for digitado uma funcionalidade incorreta ele será exibido;
1. **Criar Roteador:** Esse comando cria um roteador seguindo os padrões pedidos no enunciado do trabalho. Esse comando irá pedir alguns dados e criará o roteador com eles;
2. **Detalhes do Roteador:** Comando que exibe a tabela de roteamento de determinado roteador, deve-se inserir a porta que deseja visualizar;
3. **Parar Roteador:** Comando que desabilita um roteador criado, deve-se inserir a porta que deseja parar;
4. **Lista de Roteadores Ativos:** É exibido uma lista com todos os roteadores em funcionamento;
5. **Emitir Mensagem:** Envia uma mensagem de uma origem até um destino, assim como pedido no trabalho.
6. **Lista de Detalhes dos Roteadores:** Lista a tabela do roteamento de todos os roteadores ativos.



Comando	Nome
0	Lista de Comandos
1	Criar Roteador
2	Detalhes de Roteador
3	Parar Roteador
4	Lista de Roteadores Ativos
5	Emitir Mensagem
6	Lista de Roteadores Ativos(Detalhada)

Para uma maior aproximação com o pedido no trabalho, também foi implementado o comando **emissor** e **roteador** que realização exatamente as funções dos programas solicitados no trabalho. Eles seguem o seguinte formato:

- Emissor: emissor edereçoRoteadorDefault portaRoteadorDefault endereçoOrigem endereçoDestino mensagem;
- Roteador: roteador portaRoteadorDefault enderecoEncaminhamento(Separados por espaço);

Arquitetura:

O programa foi construído usando-se alguns services e objetos, segue abaixo os mais relevantes para o objetivo final do programa:

- **CommandService**: Responsável por controlar a interação com o programa;
- **ConnectionService**: Responsável por controlar a criação das conexões;
- **EmitterService**: Responsável por emitir pacotes na rede;
- **RouterService**: Responsável por criar, listar, detalhar e parar roteadores;
- **Connection**: Reponsalve por armazenar uma coneção aberta;
- **Package**: Pacote enviado atraves da rede;
- **Redirection**: Caminho na tabela de roteamento de um roteador;
- **Router**: Objeto com o objetivo de representar um roteador;
- **RoutingTable**: Tabela de roteamento de um roteador.

Roteador e Emissor:

No roteador é possível se criar, listar, detalhar e parar um roteador com os comandos específicos. A criação pode ser feita de 2 formas, através do comando **roteador** ou colocar o comando 1 e ir colocando as informações separadamente de acordo com que é solicitado.

Ao se criar um roteador na porta especifica o programa inicia uma thread que fica aguardando para receber algum pacote.

Cada roteador tem uma tabela de roteamento que fica responsável por armazenar os caminhos para o redirecionamento dos pacotes e por processar o recebimento de um pacote para encontrar a rota para que ele seja encaminhado.

No emissor é possível enviar um pacote ao para o destino especificado, sendo possível fazer o envido de 2 formas, através do comando **emissor** ou colocar o comando 5 e ir colocando as informações separadamente de acordo com que é solicitado.

Execução:

Para a execução execute o método main da classe Main localizada em **src/main/java/br/com/neto/adauto/ApplicationMain.java.**