Relatório - Trabalho Prático 2

Redes de Computadores - 2017.2

Alunos:
Ana Flávia Maia Barbosa, Matheus Lima Machado, Nathaly Veneruchi Garcia.
Professora:
Hana Karina.

Descrição da arquitetura adotada, estruturas de dados utilizadas, decisões de implementações e detalhes gerais.

- Foi utilizada a linguagem JAVA para a implementação do programa com o objetivo de implementar um simulador de encaminhamento de pacotes em uma rede passando por vários roteadores.
- Foram criadas duas classes java, classe Emissor.java e classe Roteador.java
- Para atender o requisito de que uma mensagem deve ser do tipo UDP, foram utilizadas as classes DatagramSocket e DatagramPacket.
- Na classe Roteador.java, é simulada a função de um roteador real, como são executados vários roteadores, cada roteador possui a sua tabela de roteamento, com isso, são verificados os caminhos em que o pacote irá percorrer ou não, encaminhando-o até seu destino final.
- Na classe Emissor.java, é simulado a função de um host real, que envia algum dado para outro host (no caso, o roteador onde ocorreu a finalização).

Descrição dos métodos comuns às duas classes:

Método *trimAll*:

São tratados os casos em que há mais de um espaço (na entrada) entre as linhas da tabela de roteamento e transforma em apenas um.

Método *desencapsulamento*:

É identificado os dados recebidos da string, separando-os para cada posição do vetor. Observação: Na classe Emissor.java, o vetor tem tamanho 5 e na classe Roteador.java tem tamanho 3. Pois, no Emissor existem: Número IP, porta do roteador, endereço de origem, endereço de destino e a mensagem a ser enviada. E no Roteador: Endereço de origem, endereço de destino e a mensagem a ser enviada.

Descrição dos métodos presentes na classe Emissor.java:

Método *Main:*

1. É lido um vetor de dados, que serão os dados da mensagem a ser enviada: Número IP, porta do roteador, endereço de origem, destino e a mensagem a ser enviada.
2. Enquanto a mensagem for diferente de "SAIR", serão lidas todas as informações de criação de um pacote que o Emissor deseja enviar aos roteadores. Não sendo necessário assim, rodar novamente os programas para que informações de um novo pacote sejam passadas.

Descrição dos métodos presentes na classe Roteador.java:

Método *criaVetorBinarioAbreviado*:

É tratado o caso de a máscara vir no formato CIDR, convertendo-a em binário, alocando-a em um vetor de “32 bits” (vetor de inteiros com 32 posições).

Método *criaVetorBinario*:

É convertido para binário uma máscara no formato padrão, alocando-a em um vetor de “32 bits” (vetor de inteiros com 32 posições).

Método *verificaMascara*:

É um método que retorna um boolean. É verificado se a máscara passada como parâmetro é uma máscara abreviada (ex: "/8"), caso seja, retorna true.

Método *Main*:

1. Primeiro, é feita a leitura das entradas, são elas: porta e tabela.
2. É feita a contagem de quantas barras tem para separar os endereços, bem como saber onde começam e onde terminam e então realizamos a separação dos dados de entrada.
3. São inseridos os dados em uma matriz, para fácil manipulação dos dados em forma de tabela.
4. É criado o pacote, dizendo qual é o tamanho(b) da String que contém os dados que ele deve acomodar.
5. O socket então recebe o pacote.
6. É verificado o tipo de máscara (se está no formato padrão ou no formato CIDR).
7. Após isso, a tabela de roteamento é percorrida para a verificação de quais linhas da coluna Destino (da tabela de roteamento) casam com o destino do pacote.
8. É feito "AND lógico" bit a bit entre a rede de destino do pacote e a máscara da linha (da tabela de roteamento) em questão.
9. Nos atentamos ao seguinte caso: o pacote será recebido pelo roteador em que tiver a maior máscara e para isso, fazemos uma verificação e armazenamento conforme percorremos a tabela de roteamento.
10. Com a chegada na tabela candidata, é decidido qual será o resultado, ou seja a operação a ser realizada no pacote. (Encaminhamento, finalização e descarte).

a. Encaminhamento: É criado um pacote, colocando os dados no payload do mesmo e definindo a porta em que ele deve ser enviado.

b. Finalização: É informado os dados desencapsulados do pacote (ip origem, ip destino, conteudo da mensagem) relevantes para esta operação que o roteador decidiu executar
c. Descarte: É informado os dados desencapsulados do pacote (ip origem) relevantes para esta operação (descarte) que o roteador decidiu executar.

Funcionalidades não implementadas

Não se aplica.

Funcionalidades extras implementadas

A funcionalidade extra implementada permite que seja passada para o roteador, uma tabela, com máscaras no formato CIDR(ex:endereço/16) ou no formato padrão (ex: 255.255.0.0), podendo haver variação dos padrões dentro de uma mesma tabela.