Universidade Federal do Mato Grosso do Sul

Faculdade de Computação

Reteamento ip

Gian Fonseca && Jonathan Kinjo
Alunos

Campo Grande - MS

Conteúdo

1 Identificação

- Título do trabalho: Roteamento Ip-4.

- Matéria: Redes de Computadores – 2017.2

- Autores:

- Gian Fonseca - RGA:201519060378 – Do curso de Engenharia de Software.
- Jonathan H. Kinjo RGA:201519060360 – Do curso de Engenharia de Software.

2 Introdução

O presente relatório relata as atividades exercidas no trabalho de Roteamento IP-4. O trabalho pretende estudar o contexto da implementação do Cliente DNS. Assim, ele foi feito em linguagem java com a importação das biblioteca java.net e java.io.

3 Código em Java

3.1 Classe UDPClient

```
1  /*
2  *   Classe que representa o cliente ou emissor
3  *   essa classe herda de SendPackage
4  */
5  import java.io.*;
6  import java.net.*;
7  import java.util.*;
8  class UDPClient extends SendPackage {
9      /*
10     *  Metodo principal
11     *  Faz a comunicacao com o roteador, enviando um pacote
12     */
13     public static void main(String args[]) throws Exception {
14         input(args);
15         preparing();
16
17         forwarding(message.getBytes());
18         forwarding(address.getBytes());
```

```
19          forwarding(destiny.getBytes());
20
21          DatagramPacket receivePacket = new DatagramPacket(
               receiveData,
22                          receiveData.length);
23
24          socket.receive(receivePacket);
25          System.out.println("Pacote UDP recebido...");
26
27          String modifiedSentence = new String(receivePacket.
               getData());
28
29          System.out.println("Texto recebido do servidor:" +
               modifiedSentence);
30          socket.close();
31          System.out.println("Socket cliente fechado!");
32      }
33
34      /*
35      *   Metodo input, responsavel por trata do recebimento
           dos dados pelo terminal
36      */
37      static void input(String[] array) {
38          if(array.length == 0) {
39                  System.out
40                          .println("Obrigatorio, argumento\n
                               exemplo: (ex) java UDPClient.java
                               127.0.0.1  12345  10.0.0.5
                               1.2.3.4  Hello");
41                  System.exit(0);
42          }
43          router = array[0];
44          port = array[1];
45          address = array[2];
46          destiny = array[3];
47          message = array[4];
48      }
49  }
```

3.2 Classe UDPServer

```
1   /*
2   *   Classe que representa o roteador
```

```
3  *    essa classe estende de SendPackage
4  */
5  import java.io.*;
6  import java.net.*;
7
8  class UDPServer extends SendPackage {
9      static int port;
10     static Table table;
11     static DatagramPacket receivePacket;
12     /*
13     *    Metodo principal da classe
14     *
15     */
16     public static void main(String args[]) throws Exception {
17         input(args);
18
19         socket = new DatagramSocket(port);
20         started();
21     }
22
23     /*
24     *    Metodo started, responsavel por monitorar os pedidos
           de conexao
25     */
26     static void started () throws IOException {
27         while (true) {
28             receivePacket = new DatagramPacket(receiveData,
                   receiveData.length);
29             System.out.println("Esperando por datagrama UDP
                   na porta " + port);
30             receiving();
31
32             Route route = table.route(change(destiny));
33
34             if(route.network.equals("default"))
35                 System.out.println("destination " + destiny +
                       " not found in routing table, dropping
                       packet ");
36
37             resend();
38         }
39     }
40     /*
```

```
41      *   Metodo change, resposavel por recuperar dados
           recebidos
42      *   sempre que o servidor recebe um dado ele tem 1024
           bytes,
43      *   sendo assim, deve ser recuperado para o seu tamanho
           normal.
44      */
45      static String change (String value) {
46          String other = "";
47
48          for (int i=0, size = value.length(); i != size; i++)
               {
49              if((int) value.charAt(i) > 45 && (int) value.
                   charAt(i) < 58)
50                  other += value.charAt(i);
51          }
52
53          destiny = "";
54          for (int i=0, size = other.length(); i != size - 1; i
               ++) {
55              destiny += other.charAt(i);
56          }
57
58          return destiny;
59      }
60
61      /*
62      *   Metodo resend, responsavel por reenviar o pacote.
63      */
64      static void resend () throws IOException {
65          IPAddress = receivePacket.getAddress();
66
67          String capitalizedSentence = message.toUpperCase();
68
69          sendData = capitalizedSentence.getBytes();
70
71          DatagramPacket sendPacket = new DatagramPacket(
               sendData,
72                  sendData.length, IPAddress, receivePacket.
                       getPort());
73
74          System.out.print("Enviando " + capitalizedSentence +
               "...");
```

```
75
76          socket.send(sendPacket);
77          System.out.println("OK\n");
78      }
79
80      /*
81      *   Metodo receiving, responsavel por receber o pacote
            vindo do cliente
82      */
83      static void receiving () throws IOException {
84          socket.receive(receivePacket);
85          System.out.print("Datagrama UDP messagem  recebido...
                ");
86
87          message = new String(receivePacket.getData());
88          System.out.println(message);
89
90          socket.receive(receivePacket);
91          System.out.print("Datagrama UDP address  recebido..."
                );
92
93          address = new String(receivePacket.getData());
94          System.out.println(address);
95
96          socket.receive(receivePacket);
97          System.out.print("Datagrama UDP destiny  recebido..."
                );
98
99          destiny = new String(receivePacket.getData());
100         System.out.println(destiny);
101     }
102
103
104     /*
105     *   Metodo input, responsavel por trata do recebimento
            dos dados pelo terminal
106     */
107     static void input (String[] array) {
108         if(array.length == 0) {
109             System.out
110                     .println("Obrigatorio, argumento\n
                            exemplo: (ex) java UDPServer.java 12345
                            ");
```

```
111             System.exit(0);
112         }
113 
114         port = Integer.parseInt(array[0]);
115 
116         table = new Table(array.length - 1);
117 
118         for(String value : array)
119             table.add(value.split("/"));
120 
121     }
122 }
```

3.3 Classe Table

```
1  /*
2  *   Classe que representa a tabela
3  */
4 
5  public class Table {
6      Route[] route;
7      int index = 0;
8 
9      /*
10     *   Metodo construtor
11     */
12     Table (int size) {
13         route = new Route[size];
14     }
15 
16     /*
17     *   Metodo add, adiciona uma nova rota na tabela
18     */
19     public void add (String[] array) {
20         if(array.length == 1)
21             return;
22 
23         route[index] = new Route(array);
24         index = index + 1;
25     }
26 
27     /*
28     *   Metodo route, responsavel por decidir o roteamento
```

```
29      */
30      public Route route (String ip) {
31          Route choice = new Route("default");
32
33          for (Route value : route)
34          {
35
36
37              if((Long.parseLong(value.network.replace(".", "")
                    ) & Long.parseLong(value.mask.replace(".", ""))
                    ) == (Long.parseLong(ip.replace(".", "")) &
                    Long.parseLong(value.mask.replace(".", ""))))
38                  choice = choose(value, choice);
39          }
40
41
42          return choice;
43      }
44
45      /*
46      *   Metodo choose, define a plioridade entre rotas, como
            destino.
47      */
48      private Route choose (Route value, Route choice) {
49          System.out.println(value.network);
50          if(choice.network.equals("default"))
51              return value;
52
53          if(Long.parseLong(value.mask) > Long.parseLong(choice
                .mask))
54              return value;
55
56          return choice;
57      }
58  }
```

3.4 Classe Route

```
1   /*
2   *   Classe representante de uma rota
3   */
4
5   public class Route {
```

```
6       String network;
7       String mask;
8       String gateway;
9       String intface;
10      /*
11      *   Metodo construtor default
12      */
13      Route (String network) {
14          this.network = network;
15      }
16      /*
17      *   Metodo construtor
18      */
19      Route (String[] array) {
20          this.network = array[0];
21          this.mask = array[1];
22          this.gateway = array[2];
23          this.intface = array[3];
24      }
25  }
```

3.5 Classe SendPackage

```
1   /*
2   *   Classe SendPackage padroniza o mecanismo de envio de
        pacotes.
3   *   essa classe herda de pacote
4   */
5   import java.io.*;
6   import java.net.*;
7   import java.util.*;
8
9   class SendPackage extends Package {
10      static DatagramSocket socket;
11      static BufferedReader inFromUser;
12      static byte[] sendData = new byte[1024];
13      static byte[] receiveData = new byte[1024];
14      static InetAddress IPAddress;
15      /*
16      *   Metodo de preparacao de pacotes
17      */
18      public static void preparing () throws Exception {
19          inFromUser = new BufferedReader(new InputStreamReader
```

```
            (System.in));
20          socket = new DatagramSocket();
21          IPAddress = InetAddress.getByName(router);
22      }
23      /*
24      *   Metodo de envio de pacotes
25      */
26      static void forwarding (byte[] sendData) throws
           IOException {
27          DatagramPacket sendPacket = new DatagramPacket(
               sendData,
28                          sendData.length, InetAddress.
                               getByName(router), Integer.parseInt
                               (port));
29
30          System.out.println("Enviando pacote UDP para " +
               router + ":" + port);
31          socket.send(sendPacket);
32      }
33  }
```

3.6 Classe Package

```
1   /*
2   *   Classe que representa o tipo de dado de um pacote.
3   */
4   class Package {
5       static String address;
6       static String destiny;
7       static String router = "localhost";
8       static String message;
9       static String port;
10  }
```

Referências