Aplicativo para Eleição para Prefeito

Luiza Ávila Defranco Gonçalves - 587490
Stefany Gaspar Xavier França - 582931

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

1. Link do vídeo

Link: https://www.youtube.com/watch?v=RmsEPHQe9k8

2. Código

2.1. Servidor

```
1
2  import java.io.IOException;
3  import java.io.EOFException;
4  import java.io.ObjectInputStream;
5  import java.io.ObjectOutputStream;
6  import java.net.*;
7  import java.util.ArrayList;
8  import java.util.Scanner;
9
10
11 public class Server
12 {
13
14     private static final int PORTA = 6789;
15     private static final int MAXIMO_PESSOAS = 10;
16
17
18     // Manter listas das conexoes e dos fluxos
19     private ArrayList<Socket> todasConexoes = new ArrayList<>();
20     private ArrayList<ObjectInputStream> listaInput = new
           ArrayList<>();
21     private ArrayList<ObjectOutputStream> listaOutput = new
           ArrayList<>();
22
23     private int quantClientes;
24
25     private int quantidadeVotosA;
26     private int quantidadeVotosB;
27     private int quantidadeVotosC;
28
29     private ServerSocket servidor;
30
31
32     private void inicializarServidor() throws IOException
```

```
33      {
34          System.out.println("Servidor inicializado ...");
35
36
37              // Criar o socket do servidor para aceitar conexoes
                    de clientes
38              this.servidor = new ServerSocket(Server.PORTA,
                    Server.MAXIMO_PESSOAS);
39
40             /******************Parte de conexao ao servidor do
                   codigo. Fica parado ate conectar, quando um
                   cliente se conecta,
41              * adicionamos ele a lista, criamos uma stream para
                    troca de dados e criamos uma thread para ele
                    ************************/
42              while (true)
43              {
44                      try
45                      {
46                          Socket conexao = this.aguardarConexao();
47                          this.todasConexoes.add(conexao);
48                          this.quantClientes++;
49                          ObjectInputStream input = this.
                                criarInputStream(conexao);
50                          ObjectOutputStream output = this.
                                criarOutputStream(conexao);
51                          this.listaInput.add(input);
52                          this.listaOutput.add(output);
53
54                          new Thread(new Runnable()
55                              {
56                                  @Override
57                                  public void run() {
58                                      threadCliente(conexao, input
                                            , output);
59                                  }
60                              }).start();
61
62                      }catch(IOException ioe){
63                          ioe.printStackTrace();
64                      }
65              }
66      }
67
68      private Socket aguardarConexao() throws IOException
69      {
70          System.out.println("Esperando cliente...");
71          Socket conexao = this.servidor.accept();
```

```
72          System.out.println("Novo cliente conectado: " + conexao.
               getInetAddress().getHostName());
73          return conexao;
74      }
75
76
77      private void threadCliente(Socket conexao, ObjectInputStream
            input, ObjectOutputStream output)
78      {
79
80          this.msg(output, "Voce esta conectado!");
81          this.msg(output, "Vote 1 para Charlize Theron");
82          this.msg(output, "Vote 2 para Rita Lee");
83          this.msg(output, "Vote 3 para Dua Lipa");
84
85          while (true)
86          {
87              try
88              {
89          // Receber voto do cliente
90          int voto = (int)input.readObject();
91          System.out.println("Voto do cliente: " + voto);
92
93                  if(voto==1)
94                  {
95                    System.out.println("Voto contabilizado para o
                         Charlize Theron!");
96                    this.quantidadeVotosA++;
97                  }
98                  else if(voto==2)
99                  {
100                     System.out.println("Voto contabilizado para
                           o Rita Lee!");
101                   this.quantidadeVotosB++;
102                 }
103                 else if(voto==3)
104                 {
105                   System.out.println("Voto contabilizado para o
                         Dua Lipa!");
106                   this.quantidadeVotosC++;
107                 }
108                 System.out.println("Total de clientes
                       simultaneos: " + this.quantClientes);
109                 System.out.println("Quant. votos Charlize Theron
                       : " + quantidadeVotosA);
110                 System.out.println("Quant. votos Rita Lee: " +
                       quantidadeVotosB);
111                 System.out.println("Quant. votos Dua Lipa: " +
                       quantidadeVotosC);
```

```
112             }
113             catch(ClassNotFoundException classNotFoundException)
114             {
115                 System.out.println("Nao tratamos isso.");
116             }
117             catch (IOException ioe)
118             {
119               System.out.println("Algo inesperado aconteceu");
120               break;
121             }
122             break;
123         }
124 
125         try
126         {
127           // Tentar fechar a conexa e os fluxos de entrada e de
                saida com o cliente
128             this.fecharConexaoCliente(conexao);
129             System.out.println("Conexao finalizada!");
130         this.fecharFluxosCliente(input, output);
131         }
132         catch(EOFException eofe)
133         {
134           System.out.println("EOFException: " + eofe.getMessage
                ());
135         }
136         catch(IOException ioe)
137         {
138           System.out.println("IOException: " + ioe.getMessage())
                ;
139         }
140 
141         this.quantClientes--;
142     }
143 
144 /**Parte de enviar mensagens para o cliente */
145   private void msg(ObjectOutputStream output, String mensagem)
146     {
147         try
148         {
149           System.out.println("[SERVER] - Enviando mensagem para
                o cliente: " + mensagem);
150             output.writeObject(mensagem);
151             output.flush();
152         }
153         catch(IOException ioe)
154         {
155             System.out.println("ERRO: Nao consegui enviar a
                  mensagem");
```

```
156         }
157     }
158 
159 
160     private ObjectOutputStream criarOutputStream(Socket conexao)
            throws IOException
161     {
162         ObjectOutputStream output = new ObjectOutputStream(
               conexao.getOutputStream());
163 
164         // Limpar a stream antes
165         output.flush();
166         return output;
167     }
168 
169 
170     private ObjectInputStream criarInputStream(Socket conexao)
           throws IOException
171     {
172         return new ObjectInputStream(conexao.getInputStream());
173     }
174 
175 
176     private void fecharConexaoCliente(Socket conexao) throws
           IOException
177     {
178         System.out.println("Fechando conexao com o cliente...");
179         conexao.close();
180         this.todasConexoes.remove(conexao);
181     }
182 
183 
184     private void fecharFluxosCliente(ObjectInputStream input,
           ObjectOutputStream output) throws IOException
185     {
186       input.close();
187       output.close();
188     this.listaInput.remove(input);
189     this.listaOutput.remove(output);
190     }
191 
192 
193 // ------------------------------ METODO PRINCIPAL
    ------------------------------
194 
195     public static void main(String[] args) throws IOException
196     {
197         Server server = new Server();
198         server.inicializarServidor();
```

```
199     }
200
201 }
```

Listing 1. Servidor

2.2. Cliente

```
1
2  import java.io.IOException;
3  import java.io.EOFException;
4  import java.io.ObjectInputStream;
5  import java.io.ObjectOutputStream;
6  import java.net.*;
7  import java.util.Scanner;
8
9  public class Cliente
10 {
11
12   // Constantes
13     private static final String IP_SERVIDOR = "127.0.0.1";
14     private static final int PORTA_SERVIDOR = 6789;
15
16   // Variaveis de controle de conexao e fluxo
17     private Socket conexao;
18     private ObjectOutputStream output;
19     private ObjectInputStream input;
20
21     // Variaveis para trocar mensagens com o servidor
22     private String mensagem = "";
23
24   /**
25    * Metodo usado para iniciar o backend do Cliente.
26    */
27     public void inicializar()
28     {
29         try
30         {
31             /****Conectar ao servidor*****/
32             System.out.println("Tentando conectar ao servidor...
                   ");
33           this.conexao = new Socket(InetAddress.getByName(
                IP_SERVIDOR), PORTA_SERVIDOR);
34             System.out.println("Conectado a: " + this.conexao.
                   getInetAddress().getHostName());
35
36             /*****Criacao de fluxos****/
37             this.output = new ObjectOutputStream(this.conexao.
                   getOutputStream());
38             this.output.flush();
```

```
39              this.input = new ObjectInputStream(this.conexao.
                    getInputStream());
40              System.out.println("Fluxos de entrada e saida
                    prontos!");
41
42              /****Conectado****/
43              do
44              {
45                      Scanner scanner = new Scanner(System.in);
46                      // Obter a resposta do servidor
47                      try{
48                          // Obter a resposta do servidor
49                          for(int i=0;i<4;i++){
50                              String respostaServidor = (String)
                                    this.input.readObject();
51                              System.out.println("Mensagem
                                    recebida do servidor: " +
                                    respostaServidor);
52                          }
53                          int voto = scanner.nextInt();
54                          this.enviarMsgParaServidor(output,voto);
55                      }catch(ClassNotFoundException cnfe)
56                      {
57                          System.out.println("Algo deu errado");
58                      }
59                      catch(IOException ioe)
60                      {
61                          break;
62                      }
63
64              }while(!mensagem.equals("[SERVER] - END"));
65
66          }
67          catch(EOFException eofe)
68          {
69              System.out.println("Cliente finalizou a conexao!");
70          }
71          catch(IOException ioe)
72          {
73              ioe.printStackTrace();
74          }
75          finally
76          {
77              fechar();
78          }
79      }
80
81      /**
82       * Fechar a conexao e os fluxos com o servidor
```

```
83       */
84      private void fechar()
85      {
86          System.out.println("Fechando conexao...");
87          try
88          {
89              this.output.close();
90              this.input.close();
91              this.conexao.close();
92          }
93          catch(IOException ioe)
94          {
95              ioe.printStackTrace();
96          }
97      }
98
99
100     private void enviarMsgParaServidor(ObjectOutputStream output
            , int voto)
101     {
102         try
103         {
104             this.output.writeObject(voto);
105             this.output.flush();
106             System.out.println("[CLIENTE] - Mensagem enviada
                    para o servidor: " + voto);
107         }
108         catch(IOException ioe)
109         {
110             System.out.println("Algo deu errado ao enviar o voto
                    ");
111         }
112     }
113
114   public static void main(String args[])
115     {
116         Cliente cliente = new Cliente();
117
118         cliente.inicializar(); // Inicializando a conexao com o
                servidor
119
120     } // fim main()
121
122 } // fim Cliente
```

Listing 2. Cliente

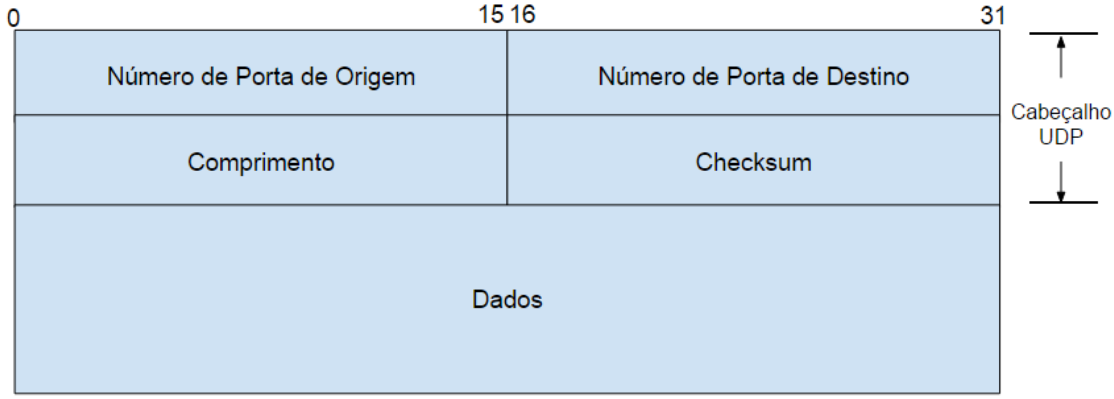
3. Características do protocolo de transferência do vídeo

3.1. Protocolo UDP

O ***User Datagram Protocol*** (UDP) é um protocolo da camada de transporte, que permite que a aplicação envie um datagrama encapsulado num pacote IPv4 ou IPv6 a um destino. Dentre as principais características do protocolo a que mais se destaca é que não existe qualquer tipo de garantia que o pacote chegue corretamente (ou de qualquer modo), ao contrário de seu "irmão" TCP. O protocolo também não remove qualquer tipo de verificação de erros, pode enviar pacotes fora de ordem e não é orientado a conexão, então não é necessário estabelecer conexão antes do envio do datagrama. O propósito dessa opção é acelerar o processo de envio de dados, visto que todas as etapas de comunicação necessárias para verificar a integridade de um pacote (e para reenviá-lo, se necessário) contribuem para deixá-lo mais lento.

Devido as suas características, só valerá a pena enviar pacotes utilizando o UDP quando este for pequeno, neste caso, menor que 512KB. Ele se baseia em portas para a troca de informações, desta forma, é atribuída uma porta ao destino e uma porta a origem, como pode ser visualizado a seguir:

Figure 1. Formato do pacote UDP



Fonte: http://www.bosontreinamentos.com.br/redes-computadores/curso-de-redes-protocolo-udp-user-datagram-protocol/

- Os campos porta de origem e porta de destino, especificam que portas que serão utilizadas na comunicação.
- O campo tamanho descreve quantos bytes terá o pacote completo.
- O campo checksum é opcional e faz uma soma verificadora para garantir que os dados estarão livres de erros.

As portas que podem ser adotadas pelo UDP:

- Porta origem: Valor de 1 a 65535
- Porta de destino: Valor de 1 a 65535

Como dito previamente, o UDP é diferente do protocolo TCP, e consegue ter algumas vantagens em determinadas situações. O UDP não se importa com criar ou destruir conexões, durante uma conexão, o UDP troca apenas 2 pacotes, enquanto no TCP

esse número é superior a 10, deixando-o mais rápido e mais propício à aplicações do tipo pergunta-resposta. A sua agilidez também pode ser melhor durante a transmissão de um vídeo ao vivo, por exemplo, pois é mais interessante que uma pessoa perca alguns trechos ou tenha que lidar com distorções de imagem e áudio do que esperar pelo recebimento de um pacote que se perdeu — o que pode acabar com o fator "tempo real". O UDP também é muito usado durante games online — caso você perca alguns pacotes, os personagens adversários podem se "teleportar" pela tela, no entanto não há motivos para que você receba os dados que foram perdidos, já que a partida continua mesmo sem que eles cheguem até você. Tudo o que importa é o que está acontecendo agora, não aquilo que ocorreu há alguns instantes, assim não faz sentido apostar em checagens de erro que só serviriam para aumentar a latência dos participantes. Ou seja, o protocolo UDP é admissível para fluxos de dados em tempo real, especialmente aqueles que admitem perda ou corrompimento de parte de seu conteúdo. Aplicações sensíveis a atrasos na rede, mas poucos sensíveis a perdas de pacotes, como jogos de computadores, também beneficiam do UDP. O UDP também suporta broadcasting e multicasting, devendo, necessariamente, ser utilizado.

Em suma, o protocolo UDP é utlizado quando a comunicação requere um pacote de ida e um de volta, comunicação em tempo real e a aplicação requer controle de retransmissões.

Devido a sua simplicidade e praticidade, alguns protocolos utilizam UDP, como:

- TFTP (*Trivial File Transfer Protocol*) Este protocolo é semelhante ao FTP (protocolo padrão/genérico independente de hardware sobre um modo de transferir arquivos/ficheiros e também é um programa de transferência)porém sem confirmação de recebimento pelo destino ou reenvio. É geralmente utilizado para transferir pequenos ficheiros entre hosts numa rede, tal como quando um terminal remoto ou um cliente inicia o seu funcionamento, a partir do servidor.
- SNMP (*Simple Network Management Protocol*) O SNMP ajuda o gestor da rede a localizar eventuais problemas e falhas em sua rede. Através de um gerente SNMP (SLAview, por exemplo), pode-se visualizar gráficos referentes a estatísticas de tráfego, nível do toner em impressoras, CPU e memória de diversos dispositivos. Até mesmo a quantidade de processos que estão sendo executados em um dado dispositivo pode ser analisada.
- DHCP (*Dynamic Host Control Protocol*) Trata-se de um protocolo utilizado em redes de computadores que permite a estes obterem um endereço IP automaticamente. É utilizado em redes que sofrem constantes alterações na topologia e o administrador não pode verificar o IP (Internet Protocol) de cada máquina devido a enorme quantidade, então o roteador distribui IPs automaticamente para as estações. Como esta atribuição é feita com a utilização do UDP, caso haja algum problema o usuário terá que pedir o reenvio ou reiniciar a máquina. O único problema técnico deste protocolo é que como os IPs são atribuídos aleatoriamente, fica mais difícil para o administrador ter controle sobre o que cada host está fazendo.
- DNS (*Domain Name Service*) É uma coleção de bancos de dados que traduz nomes de host para endereços únicos de IP. Neste caso, imaginemos que um usuário esteja acessando a internet e deseja ir para outra página. Ele digita o endereço no campo apropriado e entra. Se a página, por acaso, não abrir por não ter reconhecido o endereço, o problema poderá ter sido no envio ou resposta do servidor de nomes utilizando o UDP, e então o usuário tentará de novo acessar a

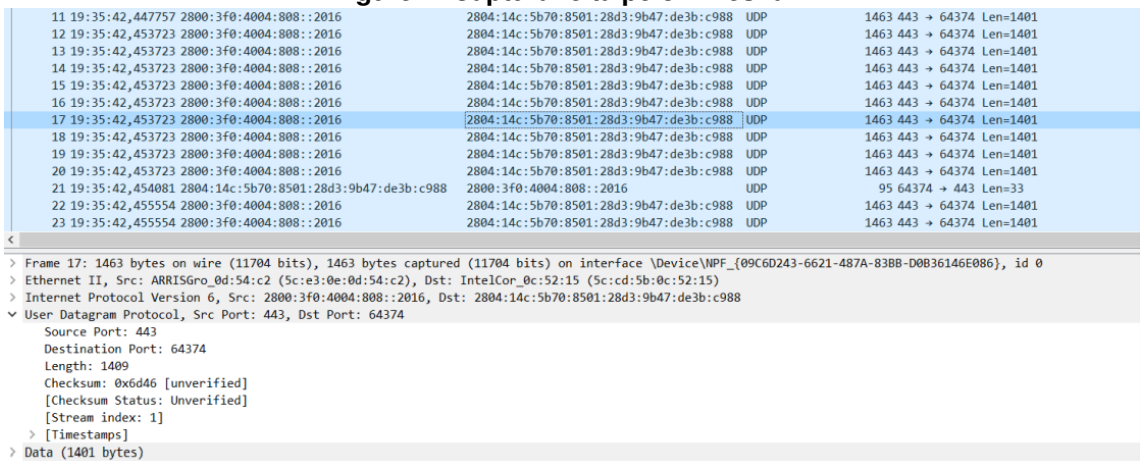
página e provavelmente conseguirá. Agora, imagine que isto fosse feito com o TCP, provavelmente esta falha não ocorreria, porém o tempo gasto para o computador saber qual IP se refere àquele nome seria inimaginável para as necessidades atuais.

Devido ao uso de portas, o UDP é vulnerável a ataques. Os mais comuns incluem: IP Spoofing - trocar o IP do host de origem por um outro qualquer; UDP flood - tipo de ataque Denial of Service (DoS) no qual o atacante sobrecarrega portas aleatórias no host alvo com pacotes IP contendo datagramas UDP; Fraggle - utiliza o UDP como protocolo de transporte. Ele causa uma chuva ao enviar um pacote UDP para uma lista de endereços de broadcast; e New Teardrop - diminui a parte de dados do pacote e o tamanho total do protocolo UDP é falsificado. Este ataque provoca o travamento das máquinas invadidas.

Enquanto uma proteção a ataques não pode ser 100% garantida, existem maneiras de suavizá-los. Geralmente, um firewall pode filtrar ou bloquear pacotes UDP maliciosos. Impedir a transmissão e/ou recepção de IPs inválidos, também ajuda. Uma possível solução para ataque tipo flood é o uso de proteção DDoS usando encaminhamento Anycast para balancear a quantidade de ataques sobre um processo de Deep Packet Inspection (DIP).

Na captura, conseguimos ver que o YouTube usa o protocolo UDP para envio de pacotes. Isso acontece pois live video streaming, como vimos antes, consegue se beneficiar da perda de alguns pacotes e não requere verificação. Caso TCP fosse usado, o buffering seria muito lento, vale mais a pena uma pequena perda de qualidade à uma grande lentidão por um vídeo de 10 minutos. Na nossa captura, conseguimos captar os pacotes de UDP do Youtube e vimos que usa a porta 443 (geralmente reservada para HTTPS) e o checksum (unverified). O UDP é colocado dentro de um IP que é envelopado ela Ethernet até a camada física mandar via Youtube.

Figure 2. Captura feita pelo Wireshark



Fonte: Autor.

References

[Canaltech 2020] Canaltech (2020). O que é dns? https://canaltech.com.br/internet/o-que-e-dns/. (accessed: 28.09.2020).

[cbpf 2020] cbpf (2020). Udp. http://www.cbpf.br/~sun/pdf/udp.pdf. (accessed: 28.09.2020).

[Cisco 2020] Cisco (2020). Definindo estratégias para proteção contra ataques de recusa de serviço de porta de diagnóstico de udp. https://www.cisco.com/c/pt_br/support/docs/security/ios-firewall/13367-3.html. (accessed: 28.09.2020).

[Quora 2020] Quora (2020). What is the reason behind youtube using tcp and not udp? https://www.quora.com/What-is-the-reason-behind-Youtube-using-TCP-and-not-UDP. (accessed: 28.09.2020).

[tecmundo 2020] tecmundo (2020). Internet: qual a diferença entre os protocolos udp e tcp? https://www.tecmundo.com.br/internet/57947-internet-diferenca-entre-protocolos-udp-tcp.htm. (accessed: 28.09.2020).

[TecMundo 2020] TecMundo (2020). O que é dhcp? https://www.tecmundo.com.br/internet/2079-o-que-e-dhcp-.htm. (accessed: 28.09.2020).

[Tecnoblog 2020] Tecnoblog (2020). O que é dns? https://tecnoblog.net/283932/o-que-e-dns/. (accessed: 28.09.2020).

[Telcomanager 2020] Telcomanager (2020). O que É snmp? https://www.telcomanager.com/blog/o-que-e-snmp/. (accessed: 28.09.2020).

[Wikipedia 2020a] Wikipedia (2020a). Lista de portas dos protocolos tcp e udp. https://pt.wikipedia.org/wiki/Lista_de_portas_dos_protocolos_TCP_e_UDP. (accessed: 28.09.2020).

[Wikipedia 2020b] Wikipedia (2020b). Protocolo udp. https://edisciplinas.usp.br/pluginfile.php/3257113/mod_resource/content/1/61-Revisao-udp-v5.pdf. (accessed: 28.09.2020).

[Wikipedia 2020c] Wikipedia (2020c). Trivial file transfer protocol? https://pt.wikipedia.org/wiki/Trivial_File_Transfer_Protocol. (accessed: 28.09.2020).

[Wikipedia 2020d] Wikipedia (2020d). User datagram protocol. https://pt.wikipedia.org/wiki/User_Datagram_Protocol. (accessed: 28.09.2020).

[]