MCZA031: Prática 1

Para Quarta-feira, 5 de Abril de 2017

João H. Kleinschmidt

Rodrigo Martins de Oliveira

Introdução

Nesta prática de laboratório de redes de computadores são exercitados os conceitos de comunicação entre aplicações através do protocolo de transporte não-orientado à conexão UDP. Os conceitos de endereço IP, porta e socket são cobertos.

Desenvolvimento

Exercício 1

a)

O servidor UDP abre um socket UDP em uma porta específica e "escuta" pacotes que chegam por esta porta, capturando-os e lendo informações nele contidas. Após receber um pacote o servidor capitaliza o conteúdo deste e o envia de volta para o endereço e porta de origem informados pelo pacote.
O cliente UDP abre um socket UDP em uma porta qualquer e envia um pacote para o endereço e porta esperados do servidor e espera pela resposta na mesma porta.

b)

Como não há quaisquer mecanismos mais complexos de verificação do status de rede do servidor e de confirmações de envio e recebimento de pacotes, o pacote enviado pelo cliente é perdido já que o servidor ainda não está online e escutando a porta designada, então o cliente fica "eternamente" aguardando a resposta do servidor. Mesmo que o servidor venha a ficar online posteriormente, o pacote enviado anteriormente pelo cliente já foi perdido e o servidor não saberá disso, portanto, continuará aguardando a chegada de um novo pacote.

c)

Se a porta para a qual o cliente envia os pacotes não for a mesma porta a qual o servidor está conectado qualquer pacote enviado pelo cliente será perdido (ou capturado por outra aplicação que detém controle sobre a porta) e o servidor nunca receberá os pacotes. O resultado é semelhante ao que foi visto no exercício 1.b)

Exercício 2

Desde que o endereço de IP e porta do servidor estejam corretamente configurados no cliente, a comunicação entre os dois processos acontece normalmente.

Exercício 3

Sim, o servidor receberá todas as mensagens. Pelo fato de ser uma comunicação não-orientada à conexão, todos os pacotes, de diferentes origens, são enfileirados pela camada de transporte e enviados sequencialmente para o servidor lê-los, portanto a concorrência de recebimento de múltiplos pacotes de diferentes endereços de origem é abstraída da aplicação.

Exercício 4

O servidor não precisa sofrer quaisquer alterações dada sua abstração sobre a origem dos pacotes, para o servidor, o endereço e porta de origem do pacote apenas são um parâmetro e não definem qualquer tratamento de sessão especial (como seria no caso de uma conexão TCP, em que uma conexão é estabelecida).

```
public class UDPClient {
    public static void main(String[] args) throws SocketException,
            UnknownHostException, IOException
    {
        BufferedReader inFromUser =
                new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("127.0.0.1");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence;
        while (true) {
            sentence = inFromUser.readLine();
            if ("sair".equals(sentence)) {
                break;
            }
            sendData = sentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
                sendData.length, IPAddress, 9876);
            clientSocket.send(sendPacket);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
                receiveData.length);
            clientSocket.receive(receivePacket);
            String modifiedSentence = new String(receivePacket.getData());
            System.out.println("Do servidor:" + modifiedSentence);
        }
        clientSocket.close();
    }
}
```

Exercício 5

```
public class WhatsappServer {
    public static void main(String[] args) throws SocketException,
       IOException {
        DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        List<String> blacklist = Arrays.asList("172.31.33.27",
            "172.31.33.29");
        while(true)
        {
            DatagramPacket receivePacket
```

Exercício 5 continued on next page...

```
                = new DatagramPacket(receiveData, receiveData.length);
            System.out.println("Servidor aguardando..." );
            serverSocket.receive(receivePacket);
            InetAddress IPAddress = receivePacket.getAddress();

            if (blacklist.contains(IPAddress.toString())) {
                continue;
            }

            String sentence = new String( receivePacket.getData());
            System.out.println("Mensagem recebida: " + sentence);
            int port = receivePacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket
                    = new DatagramPacket(sendData, sendData.length,
                        IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
```

Exercício 6

patients.xds

Exercício 7

A aplicação do Whatsapp foi criada contendo: (i) um servidor principal que é responsável por receber as mensagens de clientes e garantir que sejam recebidas aos respectivos clientes de destino; e (ii) aplicações cliente que enviam mensagens endereçadas a outros clientes para o servidor e recebem mensagens do servidor remetidas por outros clientes.
Cada aplicação cliente possui um identificador único obtido informado pelo servidor e pode enviar mensagens endereçando-as a outros clientes através de seus respectivos identificadores. O cliente espera receber informações sobre o recebimento da mensagem pelo destinatário e também sobre mensagens recebidas.
O servidor espera que cada cliente envie mensagens formatadas segundo um determinado padrão e envia respostas de confirmação de recebimento para os clientes e espera que eles também enviem respostas de confirmação de recebimento de mensagens para poder informar os remetentes das mensagens sobre sua entrega.
Cada cliente possui um mecanismo de *retry* para o envio de mensagens para garantir que elas sejam entregues ao servidor.
O servidor guarda representações de conversas entre clientes, nas quais enfileira mensagens a serem entregues. Quando as mensagens são entregues aos destinatários o servidor apaga suas cópias locais.

Conclusão

Nesta prática observamos que o protocolo UDP permite abstrair a origem dos pacotes de dados, já que não requer conexão, permitindo ao servidor lidar fácilmente com múltiplos clientes quando o tratamento *per* pacote é isolado e independente.

Também é notável que o protoloco UDP requer cuidados especiais para garantir o recebimento de pacotes.