

[Anterior](#) • [Trilha I](#) • [Próximo](#) • [comentar](#)

Busca:

Busca

Lição 6 - Utilizando Datagramas

- **Objetivo(s):** Ensinar como criar sockets UDP.
- **Direitos autorais e licença:** Veja notas de direitos autorais e licença no final da lição.

Conteúdo

- [6.1 - O que é um Datagrama?](#)
- [6.2 - Datagramas e Sockets](#)
- [6.3 - Aplicação de Criptografia](#)
- [6.4 - Direitos autorais e licença](#)
- [6.5 - Comentários](#)

6.1 - O que é um Datagrama?

Os exemplos que vimos, até agora, utilizaram comunicação TCP. Contudo, em alguns casos, esse protocolo não é recomendável devido à sobrecarga imposta pela segurança na entrega dos dados, o que não é recomendável para aplicações que transmitem um volume grande de dados. Nesses tipos de aplicações, o protocolo a ser utilizado deve ser o UDP, baseado em datagramas.

Um datagrama é uma estrutura unitária de transmissão de dados ou uma sequência de dados transmitida por uma rede ou linha de comunicação. A figura abaixo mostra o formato do datagrama IP.

| | | | | | | |
|------------------------|-------|--------------|-----------------|-----------------|---------|----|
| 0 | 4 | 8 | 16 | 19 | 24 | 31 |
| VERS | HL EN | SERVICE TYPE | TOTAL LENGTH | | | |
| IDENTIFICATION | | | FLAGS | FRAGMENT OFFSET | | |
| TIME TO LIVE | | PROTOCOL | HEADER CHECKSUM | | | |
| SOURCE IP ADDRESS | | | | | | |
| DESTINATION IP ADDRESS | | | | | | |
| IP OPTIONS (IF ANY) | | | | | PADDING | |
| DATA | | | | | | |

Quando se usa UDP não há "conexão" entre cliente e servidor: o remetente coloca explicitamente o endereço IP e a porta do destino, e o servidor deve extrair o endereço IP e a porta do remetente do datagrama recebido. Um problema é que ao usar este protocolo os dados transmitidos podem ser recebidos fora de ordem, ou perdidos. Todavia, o risco é aceitável devido à velocidade na entrega dos dados.

6.2 - Datagramas e Sockets

De maneira semelhante ao TCP, a comunicação com UDP se dá através de *sockets*. A diferença é que não há conexão nem *socket* servidor, ou seja, não é preciso liberar os recursos usando a primitiva *close*, e os sockets clientes e servidor são exatamente iguais.

A classe usada para criar *sockets* UDP é *java.net.DatagramSocket*. Ao criarmos um *socket* podemos estabelecer uma porta específica ou deixar o sistema selecionar uma porta disponível. A primeira alternativa é indicada para o servidor, uma vez que o cliente deve saber qual porta contatar. A segunda alternativa é mais indicada para os clientes, visto que a porta do cliente é irrelevante. Desse modo, para criarmos o *socket* do cliente e do servidor podemos usar o código abaixo:

```
DatagramSocket socketCliente = new DatagramSocket();
DatagramSocket socketServidor = new DatagramSocket(9876);
```

Apesar de um datagrama ser formado por vários campos, como vimos acima, o que realmente interessa na maioria dos casos são somente três desses campos: os dados enviados, o endereço do destino/remetente e a porta em que o *socket* destino/remetente está executando. Os dados devem sempre ser armazenados em um *array* de *bytes*; o endereço é representado por um objeto da classe *java.net.InetAddress*; e a porta é um inteiro. A classe usada para criar um datagrama é *java.net.DatagramPacket*.

A seguir, recriaremos a aplicação de criptografia, mas agora usando UDP.

6.3 - Aplicação de Criptografia

A primeira tarefa que devemos fazer é criar o buffer para armazenar os dados transmitidos e uma *string* que guardará a senha criptografada.

```
byte[] buffer = new byte[1024];
String senhaCriptografada;
```

Agora podemos criar o *socket* por onde os dados serão trafegados. Como estamos do lado servidor, podemos estabelecer uma porta específica para o *socket*.

```
DatagramSocket socketServidor = new DatagramSocket(PORTA);
```

Os próximos passos são:

1. receber um datagrama do cliente contendo a senha;
2. criptografar a senha;
3. enviar um datagrama para o cliente contendo a senha criptografada.

Na aplicação que usava TCP havia dois *sockets* no servidor: um que ficava esperando conexões e outro responsável pela troca de dados. Em UDP existe apenas um *socket* responsável pelas duas funções.

Precisamos criar um objeto para armazenar o datagrama enviado pelo cliente, o que pode ser feito por meio do código abaixo:

```
DatagramPacket pacoteRecebido = new DatagramPacket(buffer, buffer.length);
```

Note que passamos como parâmetro, no construtor, o *array* de *bytes* que guardará os dados do datagrama com seu respectivo tamanho. Outra opção seria criar o datagrama passando também o endereço e a porta de quem o enviou. Contudo, essas informações ainda são desconhecidas. Depois que o objeto for criado, podemos usá-lo para guardar o datagrama enviado pelo cliente. Para isso, usaremos o *socket* servidor e o método *receive*:

```
socketServidor.receive(pacoteRecebido);
```

Podemos realizar agora a criptografia da frase recebida. O código é idêntico ao exemplo mostrado anteriormente. Em seguida, devemos criar o datagrama para enviar a senha criptografada ao cliente. Antes, porém, devemos descobrir o endereço e a porta do cliente, o que pode ser feito usando o datagrama recebido:

```
InetAddress endereco = pacoteRecebido.getAddress();
int porta = pacoteRecebido.getPort();
```

Para finalizar, só resta transformar a senha criptografada em um *array* de *bytes*, criar o datagrama e enviá-lo ao cliente.

```
buffer = senhaCriptografada.getBytes();
DatagramPacket pacoteEnviado = new DatagramPacket(buffer, buffer.length, endereco, porta);
socketServidor.send(pacoteEnviado);
```

O código referente ao recebimento, tratamento e envio de pacotes deve ficar dentro de um laço, para garantir que o servidor atenda vários clientes. Dessa forma, o código completo fica como abaixo:

```
import java.math.BigInteger;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.security.MessageDigest;

public class ServidorCriptografiaUDP {
    public static void main(String args[]) {
        try {
            byte[] buffer = new byte[1024];
            String senhaCriptografada;

            DatagramSocket socketServidor = new DatagramSocket(9876);

            while (true) {
                DatagramPacket pacoteRecebido = new DatagramPacket(buffer, buffer.length);
                socketServidor.receive(pacoteRecebido);

                MessageDigest md = MessageDigest.getInstance("MD5");
                md.update(pacoteRecebido.getData());
                BigInteger hash = new BigInteger(1, md.digest());
                senhaCriptografada = hash.toString(16);

                InetAddress endereco = pacoteRecebido.getAddress();
                int porta = pacoteRecebido.getPort();

                buffer = senhaCriptografada.getBytes();
                DatagramPacket pacoteEnviado = new DatagramPacket(buffer, buffer.length, endereco, porta);
                socketServidor.send(pacoteEnviado);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Baixe o código-fonte acima neste link: <http://wiki.marceloakira.com/pub/GrupoJava/UtilizandoDatagramas/ServidorCriptografiaUDP.java>

A aplicação do lado cliente é bem semelhante, conforme é possível constatar no código mostrado abaixo:

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class ClienteCriptografiaUDP {
    public static void main(String args[]) throws Exception {
        String senha = "admin";
        String senhaCriptografada;
        byte[] buffer = new byte[1024];

        DatagramSocket socketCliente = new DatagramSocket();
        InetAddress endereco = InetAddress.getByName("localhost");

        buffer = senha.getBytes();
        DatagramPacket pacoteEnviado = new DatagramPacket(buffer, buffer.length, endereco, 9876);
        socketCliente.send(pacoteEnviado);

        DatagramPacket pacoteRecebido = new DatagramPacket(buffer, buffer.length);
        socketCliente.receive(pacoteRecebido);
        senhaCriptografada = new String(pacoteRecebido.getData());

        socketCliente.close();
    }
}
```

Baixe o código-fonte acima neste link: <http://wiki.marceloakira.com/pub/GrupoJava/UtilizandoDatagramas/ClienteCriptografiaUDP.java>

Para testar a aplicação baixe o [Servidor](#) e o [Cliente](#) e os execute na mesma máquina.

6.4 - Direitos autorais e licença

- **Autor(es):** Raphael de Aquino Gomes
- **Direito Autoral:** Copyright © Sistemas Abertos
- **Licença:** Esta obra está licenciada sob uma [Licença Creative Commons](#).



[← Anterior](#) • [Trilha I](#) [↑](#) • [Próximo →](#)

6.5 - Comentários

Adicionar