

# Curso Desenvolvedor Java



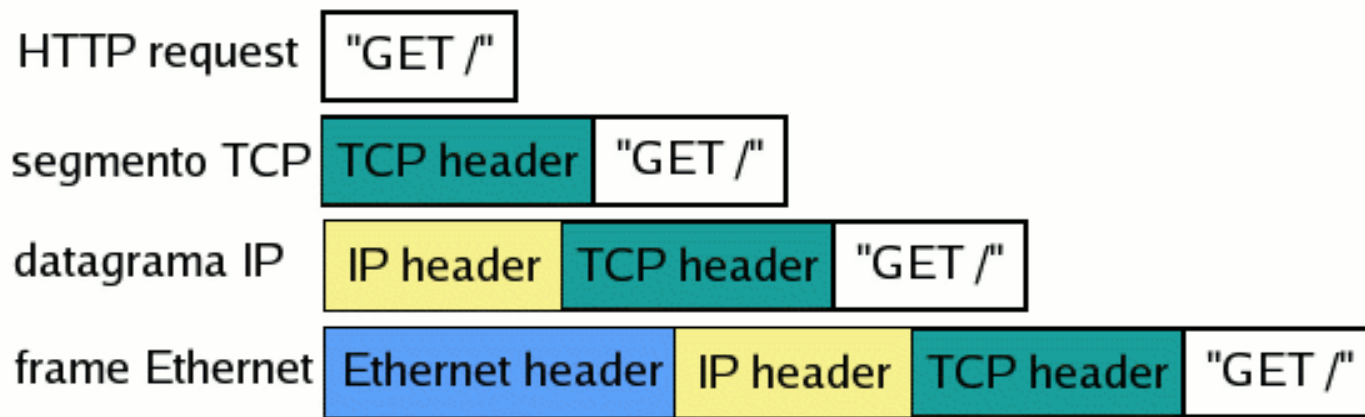
## Redes

- Conceitos

- Uma rede é simplesmente um conjunto de equipamentos de informática interconectados que se comunicam pelo mesmo protocolo de transmissão de dados.
- TCP/IP é a família de protocolos mais utilizada na internet e intranet para a comunicação entre computadores.
- Esta família de protocolos está subdividida em cinco camadas conceituais

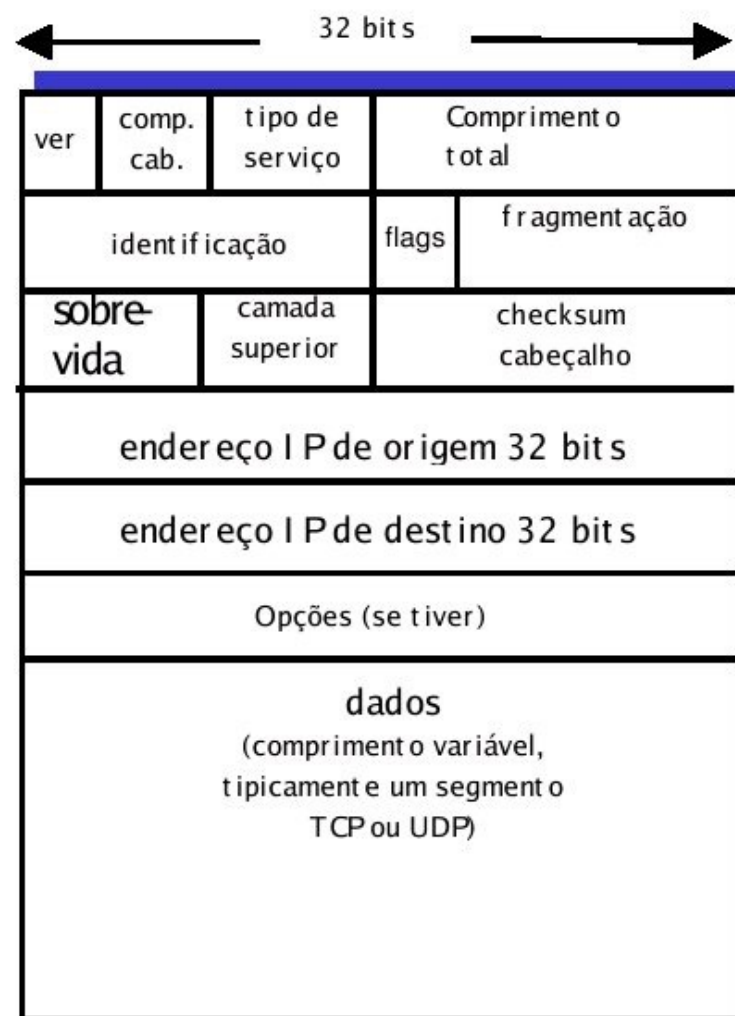


- Cada camada se comunica somente com as camadas diretamente acima e abaixo dela.
- Esta comunicação ocorre com a utilização do empacotamento / desempacotamento da informação.
- A medida que a informação é transmitida para a camada abaixo é empacotada e acrescida de um cabeçalho de controle.
- Quando a informação é transmitida para a camada acima esta é desempacotada e removido o cabeçalho de controle.



## • Protocolo IP

- IP é um protocolo de rede da camada 3 (rede) que permite endereçar hosts e rotear pacotes de dados (datagramas) entre eles.
- Seu endereçamento tem a forma x.x.x.x, onde cada x é um byte tal como 152.2.254.81. O número de hosts numa rede é determinado pela classe de endereçamento a qual ele pertence.



- Existem 5 classes de endereçamento.
  - Classe A: 1 – 126 ( 16M hosts cada) netmask 255.0.0.0
  - Classe B: 128 – 191 (65536 hosts cada) netmask 255.255.0.0
  - Classe C: 192 – 223 (256 hosts cada) netmask 255.255.255.0
  - Classe D: 224 – 239 (modo multicast)
  - Classe E: 240 – 255 (reservado para uso futuro)



- A Classe InetAddress
  - Representam um endereço IP
  - Possui métodos que:
    - Determinam o IP de um servidor quando informado seu nome de rede.
    - Retornam o nome do host local.

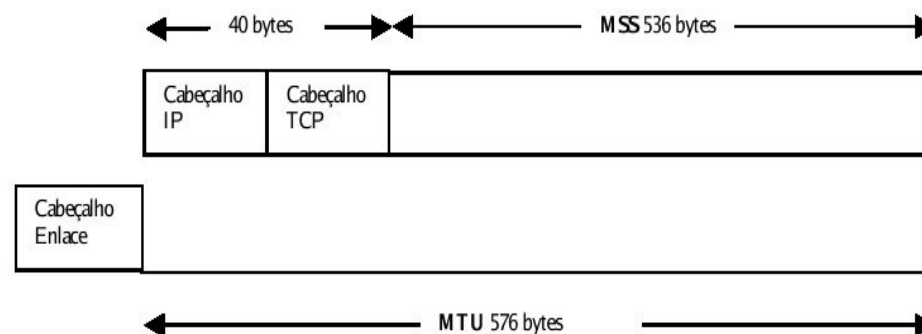
```
try {  
    InetAddress[] addr =  
        InetAddress.getAllByName("www.google.com.br");  
    for (InetAddress ip : addr) {  
        System.out.println(  
            "Host: " + ip.getCanonicalHostName() +  
            " IP: " + ip.getHostAddress());  
    }  
} catch (UnknownHostException e) {  
    e.printStackTrace();  
}
```

- Protocolo TCP

- TCP é um protocolo de transporte orientado a conexão que trabalha sobre o protocolo IP.
- Pacotes TCP, também chamados “segmentos”, são encapsulados em datagramas IP.
- A codificação / decodificação TCP ocorre nos hosts (origem / destino) que definem o canal de comunicação.

```
byte[] buffer = new byte[65509];  
DatagramPacket pkt = new DatagramPacket(buffer, buffer.length);  
MulticastSocket mSkt = new MulticastSocket(1234);  
mSkt.joinGroup(InetAddress.getByName("224.0.0.1"));  
  
while(true) {  
    mSkt.receive(pkt);  
    System.out.println(new String(pkt.getData(), 0, pkt.getLength()));  
}
```

- Ao longo do caminho entre estes hosts, os roteadores examinam o pacote IP apenas para o propósito de decisão de roteamento.
- Provê às aplicações seqüência de dados (data streams) que são transmitidas com garantia que os dados não estarão corrompidos e que os segmentos serão entregues na ordem correta.





- Protocolo UDP

- UDP é um protocolo “sem conexão” da camada de transporte que reside sobre o protocolo IP.
- Diferente do TCP, UDP não garante a integridade dos dados exceto o “checksum”.
- Se um pacote UDP estiver corrompido ele será simplesmente descartado.

```
byte[] ary = new byte[128];
```

```
DatagramPacket pack = new DatagramPacket(ary, 128);  
pack.setAddress(InetAddress.getByName(args[1]));  
pack.setData(args[2].getBytes());  
pack.setPort(1111);
```

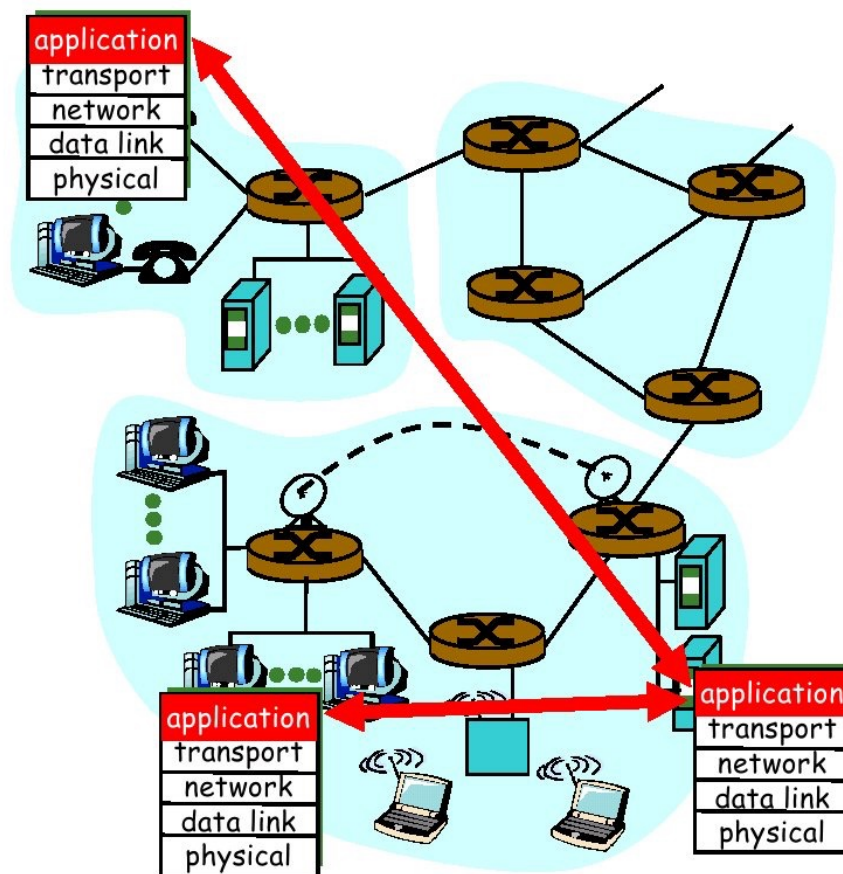
```
DatagramSocket sock = new DatagramSocket();  
sock.send(pack);  
sock.close();
```

- Aplicações

- Comunicação entre processos distribuídos
- Executam nos hosts no “espaço do usuário”
- Baseadas em troca de mensagens : e-mail, web, ftp

- Protocolos

- São parte de aplicação
- Definem o tipo de mensagens trocadas e as ações a serem tomadas pelos processos

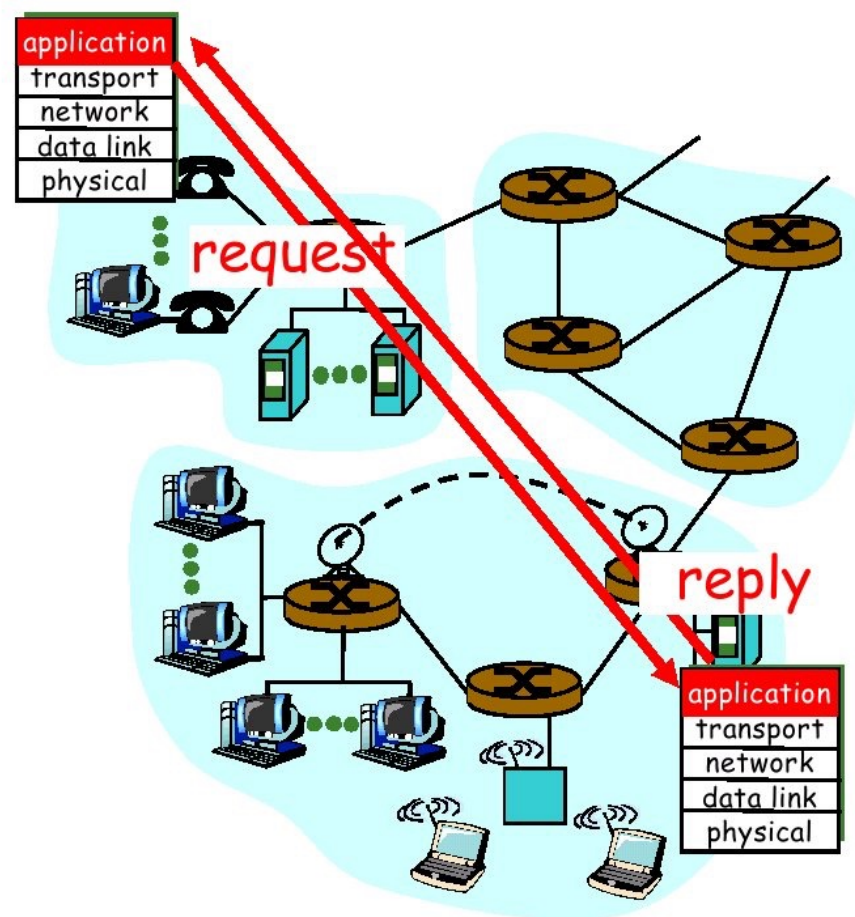


- Cliente

- Inicia o contato com o servidor
- Geralmente requisita serviços ao servidor
- Exemplo, Navegador Web – solicita páginas

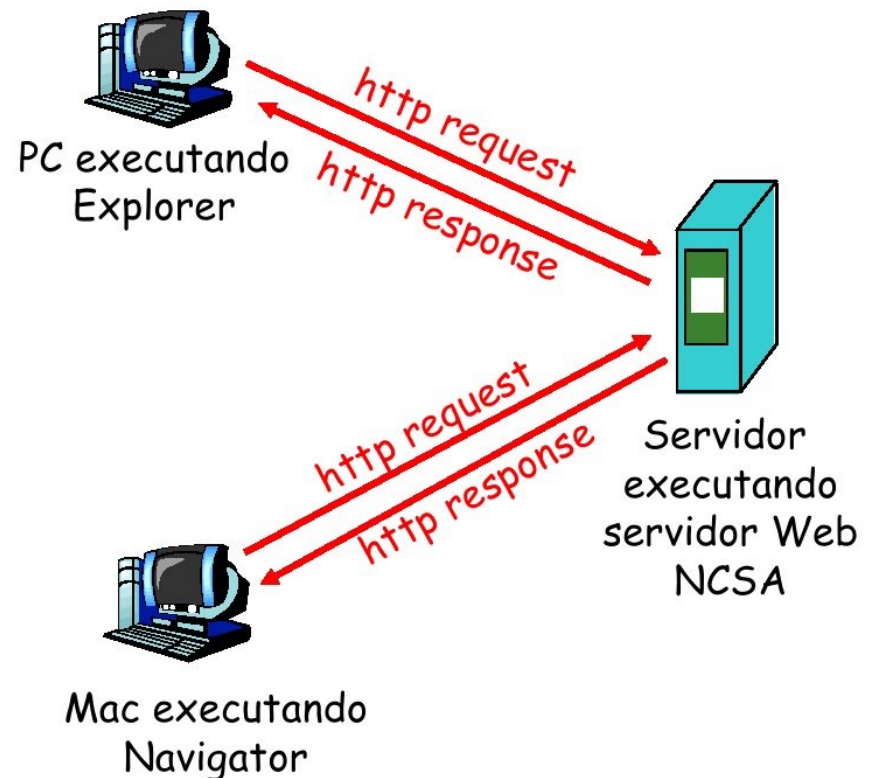
- Servidor

- Provê os serviços requisitados pelos clientes
- Exemplo, servidor Web – envia as páginas requisitadas



- Socket

- São a base de muitas aplicações baseada em rede.
- Representa uma conexão TCP entre duas aplicações através de uma rede TCP/IP
- É definida pelos endereços IP e os números das portas nas duas extremidades da conexão.
- Toda a vez que uma conexão é efetuada à uma porta num servidor, um TCP stream é criado entre o servidor e o cliente.



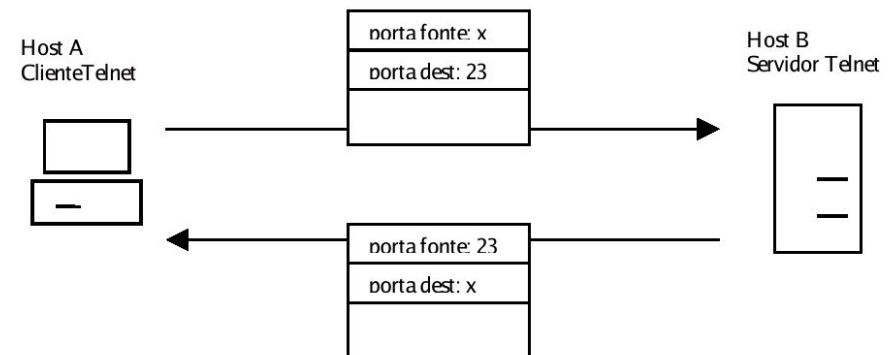
- Implementando um Cliente
  - Criar um Socket para um servidor em uma determinada porta
  - Obter os Streams de Leitura e/ou Gravação
  - Efetuar a Leitura/Gravação de dados
  - Fechar o Socket

```
Socket s = new Socket("localhost", 1234);  
InputStreamReader is = new InputStreamReader(s.getInputStream());  
BufferedReader in = new BufferedReader(is);  
PrintWriter out = new PrintWriter(s.getOutputStream());
```

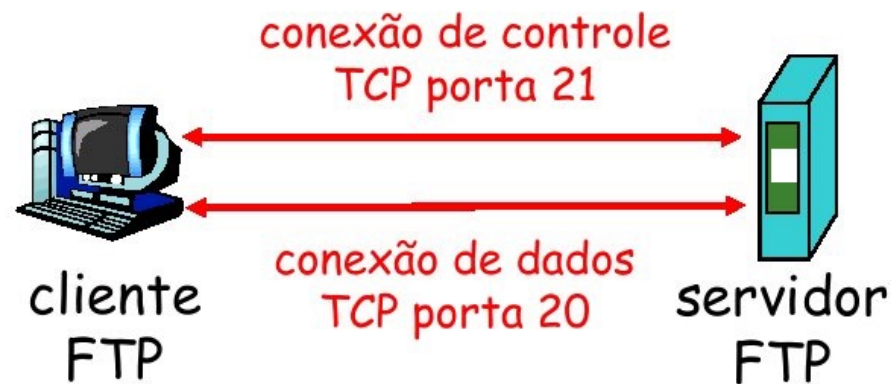
```
System.out.println(in.readLine());  
out.println(args[0]);  
out.flush();
```

```
s.close();
```

- A Classe Socket
  - Representa a conexão à um servidor em uma determinada porta
  - Permite a definição de tempo de Timeout
  - Oferece Streams de Input/Output (Entrada/Saída)



- Implementando um Servidor
  - Criar um ServerSocket
  - informando a porta a qual os clientes efetuarão conexões.
  - Utilizar o método accept() de SocketServer para aguardar as conexões dos clientes.
  - Ao aceitar a conexão de um cliente criar uma Thread para gerenciar o I/O entre o servidor e o cliente.
  - Ao final da Thread fechar o Socket criado pelo método accept().





```
ServerSocket server = new ServerSocket(1234);

Socket s = server.accept();
InputStreamReader is = new InputStreamReader(s.getInputStream());
BufferedReader in = new BufferedReader(is);
PrintWriter out = new PrintWriter(s.getOutputStream());

out.println("Bem vindo " + s.getInetAddress());
out.flush();

out.println("S:" + in.readLine());
out.flush();

s.close();
```