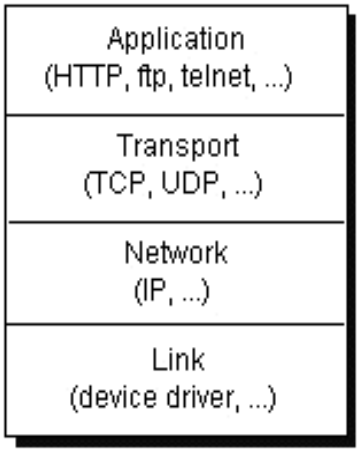
Comunicação em Rede no JAVA

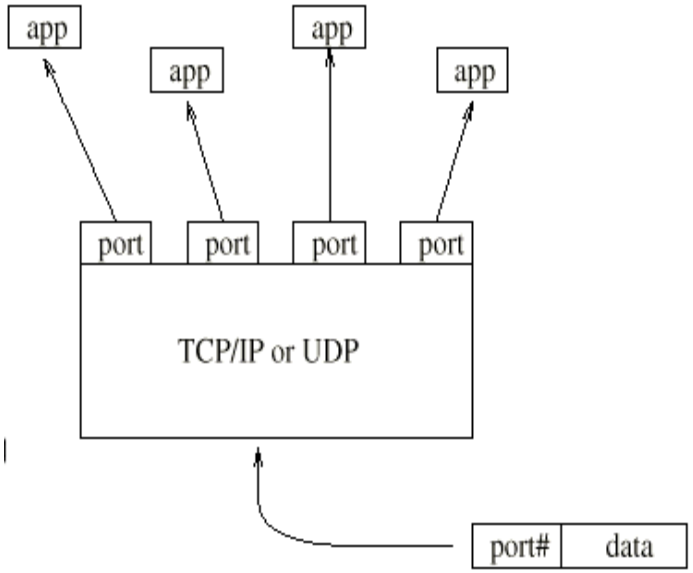
Aspectos básicos

Sockets

URL’s

Comunicação através de TCP/IP





- Todas as classes de comunicação em rede

estão definidas no *package* java.net.

- Classes para comunicação através de TCP: *Socket*, *SocketServer*, *URL*, *URLConnection*.
- Classes para comunicação através de UDP: *DatagramSocket*, e *DatagramPacket*.

Classe *Socket* - construtores

- Socket()
  - Cria um *socket* sem ligação.
- Socket(InetAddress addr, int port)
  - Cria um *stream socket* e liga-o a ao endereço *addr* e ao porto *port*.
- Socket(InetAddress rAddr, int rPort, InetAddress lAaddr, int lPort)
  - Cria um *stream socket* e liga-o a ao endereço *rAddr* e ao porto *rPort*. Associa o socket (bind) ao endereço local *lAddr* e ao porto local *lPort*.
- Socket(String host, int port)
  - Cria um *stream socket* e liga-o a host *host* e ao porto *port*.
- Socket(String rHost, int rPort, InetAddress lAddr, int lPort)
  - Cria um *stream socket* e liga-o a ao host *rHost* e ao porto *rPort*. Associa o socket (bind) ao endereço local *lAddr* e ao porto local *lPort*.

Classe *Socket* - métodos

- close()
  - Fecha o *socket*.
- InetAddress getInetAddress()
  - Retorna o endereço a que o *socket* está ligado.
- InputStream getInputStream()
  - Retorna um *InputStream* para o *socket*.
- InetAddress getLocalAddress()
  - Retorna o endereço local a que o *socket* está associado
- int getLocalPort()
  - Retorna o porto local a que o *socket* está associado.
- OutputStream getOutputStream()
  - Retorna um *OutputStream* para o *socket*.
- Int getPort()
  - Retorna o porto remoto a que o *socket* está ligado.
- setSoTimeout(int tout)
  - Activa (tout > 0)/ desactiva (tout = 0) a *flag* SO_TIMEOUT, com o *timeout* especificado em milisegundos.

Classe *Socket* - exemplo

```
import java.io.*;
import java.net.*;

public class EchoClient {
  public static void main(String[] args) throws IOException {
    Socket echoSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
      try {
         echoSocket = new Socket("taranis", 7);
         out = new PrintWriter(echoSocket.getOutputStream(), true);
         in = new BufferedReader(new InputStreamReader(echoSocket.getInputStream()));
      } catch (UnknownHostException e) {
         System.err.println("Don't know about host: taranis.");
         System.exit(1);
      } catch (IOException e) {
         System.err.println("Couldn't get I/O for the connection to: taranis.");
         System.exit(1);
      }
      BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
      String userInput;
      while ((userInput = stdIn.readLine()) != null) {
        out.println(userInput);
        System.out.println("echo: " + in.readLine());
      }
     out.close(); in.close(); stdIn.close(); echoSocket.close();
  }
}
```

Classe *ServerSocket* - construtores

- ServerSocket(int port)
  - Cria um ServerSocket associado ao porto *port*.
- ServerSocket(int port, int maxlog)
  - Cria um ServerSocket associado ao porto *port aceitando maxlog ligações*
- ServerSocket(int port, int backlog, InetAddress bindAddr)
  - Cria um *ServerSocket* associado ao porto *port* aceitando *maxlog* ligações e associado ao endereço *bindAddr*

Classe *ServerSocket* - métodos

- Socket accept()
  - Espera por ligações no socket. Retorna um novo socket quando a ligação é estabelecida.
- close()
  - Fecha um *ServerSocket*.
- InetAddress getInetAddress()
  - Retorna o endereço local a que o *socket* está associado.
- Int getLocalPort()
  - Retorna o porto local a que o *socket* está associado.
- Int getSoTimeout()
  - Retorna o estado da *flag* SO TIMEOUT.
- setSoTimeout(int)
  - Activa (tout > 0)/ desactiva (tout = 0) a flag SO_TIMEOUT, com o timeout especificado em milisegundos.

```
import java.net.*;
import java.io.*;
   class EchoTcpServer {
      public static void main(String[] args) {
          ServerSocket serverSocket = null;
          try { serverSocket = new ServerSocket(4444); }
           // ...
           Socket clientSocket = null;
           try {clientSocket = serverSocket.accept();}
           // ...
          try {
              BufferedReader is = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
              PrintWriter os = new PrintWriter(clientSocket.getOutputStream() ), false);
              String inputLine, outputLine;
              while ((inputLine = is.readLine()) != null) {
                  System.out.println("inputLine = " + inputLine);
                  outputLine = new String(inputLine);
                  os.println(outputLine);
                  os.flush();
                  // ...
              }
          // ...
        }
    // ...
  }
```