Seminář Java

Sokety

Radek Kočí

Fakulta informačních technologií VUT

Březen 2008

Porty

Připojení k síti

- obvykle jedno fyzické připojení
- všechna data procházejí tímto připojením
- problém přiřazení konkrétních dat konkrétní aplikaci

Porty

- protokoly TCP a UDP mapují data na jednotlivé procesy podle portů
- port je 16bitové číslo
- 0 – 1023 vyhraženo pro standardní služby (http, ssh, ...)
- adresace dat = adresa stroje (IP) + port

Java a sítě

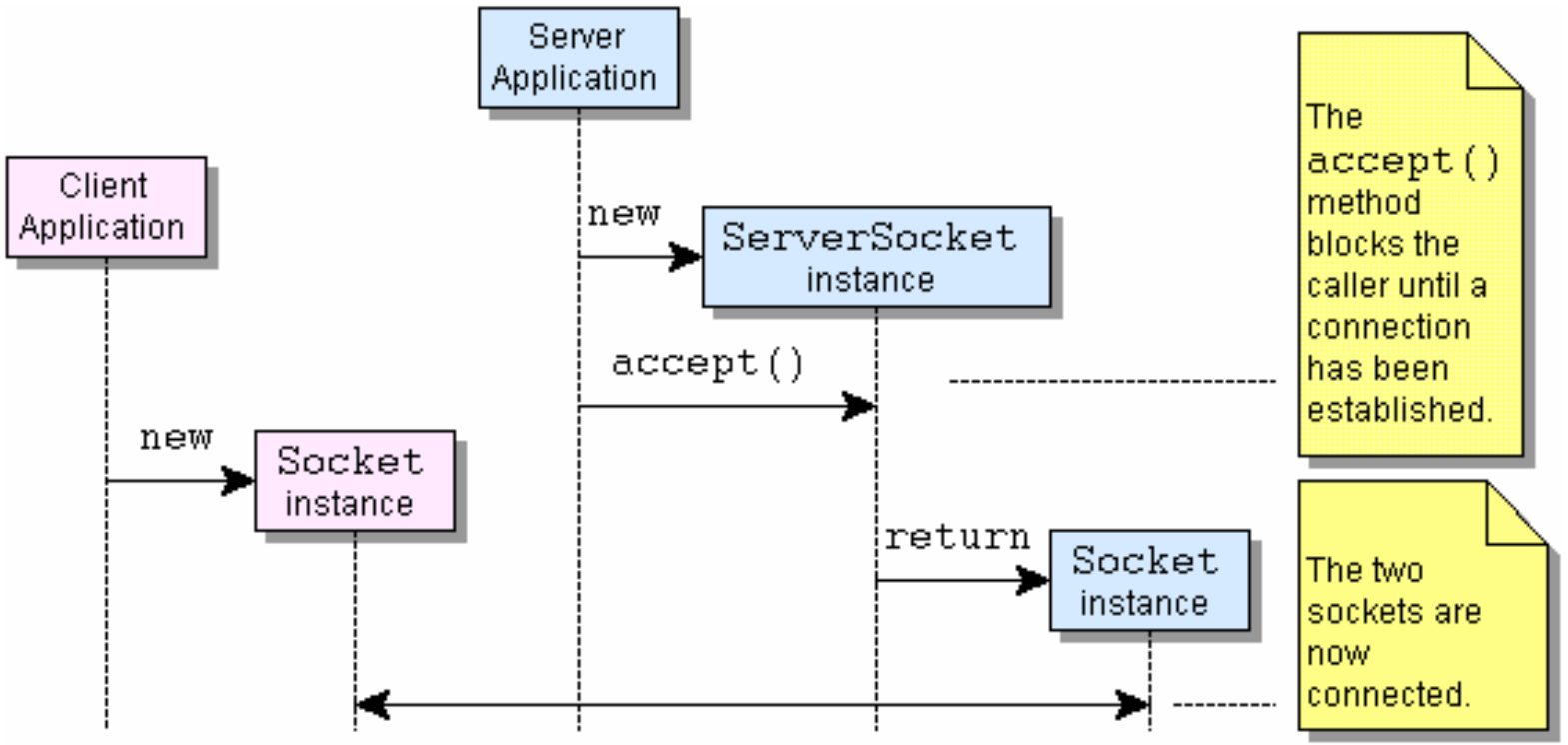
Balíček **`java.net`**

- Podpora komunikace s TCP
  - **`URL`**
  - **`URLConnection`**
  - **`Socket`**
  - **`ServerSocket`**
- Podpora komunikace s UDP
  - **`DatagramPacket`**
  - **`DatagramSocket`**
  - **`MulticastSocket`**

Sokety

- komunikace typu klient – server
  - server čeká na žádost (připojení) klienta
  - klienti se připojují na server
  - server vytváří spojení
  - klient i server zasílají/čtou data

- vytvořené spojení
  - soket na straně klienta
  - soket na straně serveru
  - komunikace prostřednictvím proudů


Server Application
Client Application
new
ServerSocket instance
accept()
new
Socket instance
return
Socket instance
The accept() method blocks the caller until a connection has been established.
The two sockets are now connected.

Třída `java.net.Socket`

`java.net.Socket`

- reprezentuje jednu stranu komunikace
- soket na straně klienta
  - žádost o vytvoření spojení se serverem
  - **`new Socket(String host, int port)`**
  - **`new Socket(InetAddress address, int port)`**
- čtení ze soketu
  - získání vstupního proudu
  - **`InputStream getInputStream()`**
- zápis do soketu
  - získání výstupního proudu
  - **`OutputStream getOutputStream()`**

Třída `java.net.ServerSocket`

`java.net.ServerSocket`

- reprezentuje server
- není potomek třídy **`Socket`**!
- po připojení klienta vytváří objekt třídy **`Socket`**
- vytvoření serveru
  - **`new ServerSocket(int port)`**
  - **`port`** na kterém server naslouchá
- naslouchání spojení
  - metoda **`accept()`**
  - blokující operace
  - vytváří objekt třídy **`Socket`**

Metody

- `getPort()`
- `getLocalPort()`
- `getInetAddress()`
- `getLocalAddress()`
- `...`

Sokety: ukázka aplikace (Server)

```
ServerSocket ss = new ServerSocket(9000);
Socket s = ss.accept();

BufferedReader in = new BufferedReader(
        new InputStreamReader(s.getInputStream()));
PrintStream out = new PrintStream(
        s.getOutputStream());

System.out.println(in.readLine());
out.print("server");
out.flush();

out.close(); in.close()
s.close(); ss.close();
```

Sokety: ukázka aplikace (Klient)

```
try {
  s = new Socket("localhost", 9000);
  System.out.println("Client: new socket port " +
                                    s.getLocalPort());

  out = new PrintStream(s.getOutputStream());
  out.print("klient");
  out.flush();

  out.close();
  s.close();
}
catch (UnknownHostException ex) { ... }
catch (java.io.IOException ex) { ... }
```

Sokety: ukázka aplikace

Problém uvedené ukázky

- obslouží pouze jednoho klienta a pak skončí
- $\Rightarrow$ nekonečná (podmíněná) smyčka

```
ServerSocket ss;
try {
  ss = new ServerSocket(9000);
  while(true) {
    Socket s = ss.accept();
    // obsluha klienta s
  }
  ss.close();
}
catch (java.io.IOException ex) { ... }
```

Sokety: ukázka aplikace

Problém uvedené ukázky

- při obsluze klienta nemůže obsloužit další
- souběžné obsloužení klientů
- $\Rightarrow$ vlákna

```
ServerSocket ss;
try {
  ss = new ServerSocket(9000);
  while(true) {
    Socket s = ss.accept();
    (new ClientThread(s)).start();
  }
  ss.close();
}
catch (java.io.IOException ex) { ... }
```

Sokety: ukázka aplikace

```
class ClientThread extends Thread {
  Socket s;

  public ClientThread(Socket s) {
    super();
    this.s = s;
  }

  public void run() {
    try {
      // obsluha klienta s
      s.close();
    } catch (java.io.IOException ex) { ... }
  }
}
```

Implementace jednoduchého WWW serveru

```
public class WebServer {
  public static void main(String[] argv) {
    new WebServer().start();
  }
  public void start() {
    try {
      ServerSocket ss = new ServerSocket(9000);
      for (int conns = 0; conns < 1; conns++) {
        Socket s = ss.accept();
        (new WebClientThread(s)).start();
      }
      ss.close();
    }
    catch (java.io.IOException ex) { ... }
  }
}
```

Implementace jednoduchého WWW serveru

```
class WebClientThread extends Thread {
  Socket s;
  public WebClientThread(Socket s) {
    super();
    this.s = s;
  }
  public void run() {
    try {
      PrintStream out = new PrintStream(
                          s.getOutputStream());
      out.print("<html>"); ...
      out.print("<p>Time: " +
         (new java.util.Date()).toString());
      ...
    } catch (java.io.IOException ex) { ... }
  }
}
```

Zdroje informací

http://java.sun.com/j2se/1.5.0/docs/guide/net/overview/overview.html

http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html

http://java.sun.com/docs/books/tutorial/rmi/TOC.html

http://www.javacoffeebreak.com/articles/rmi_corba/
http://my.execpc.com/ gopalan/misc/compare.html
http://www.webopedia.com/quick_ref/OSI_Layers.asp
http://www.javaworld.com