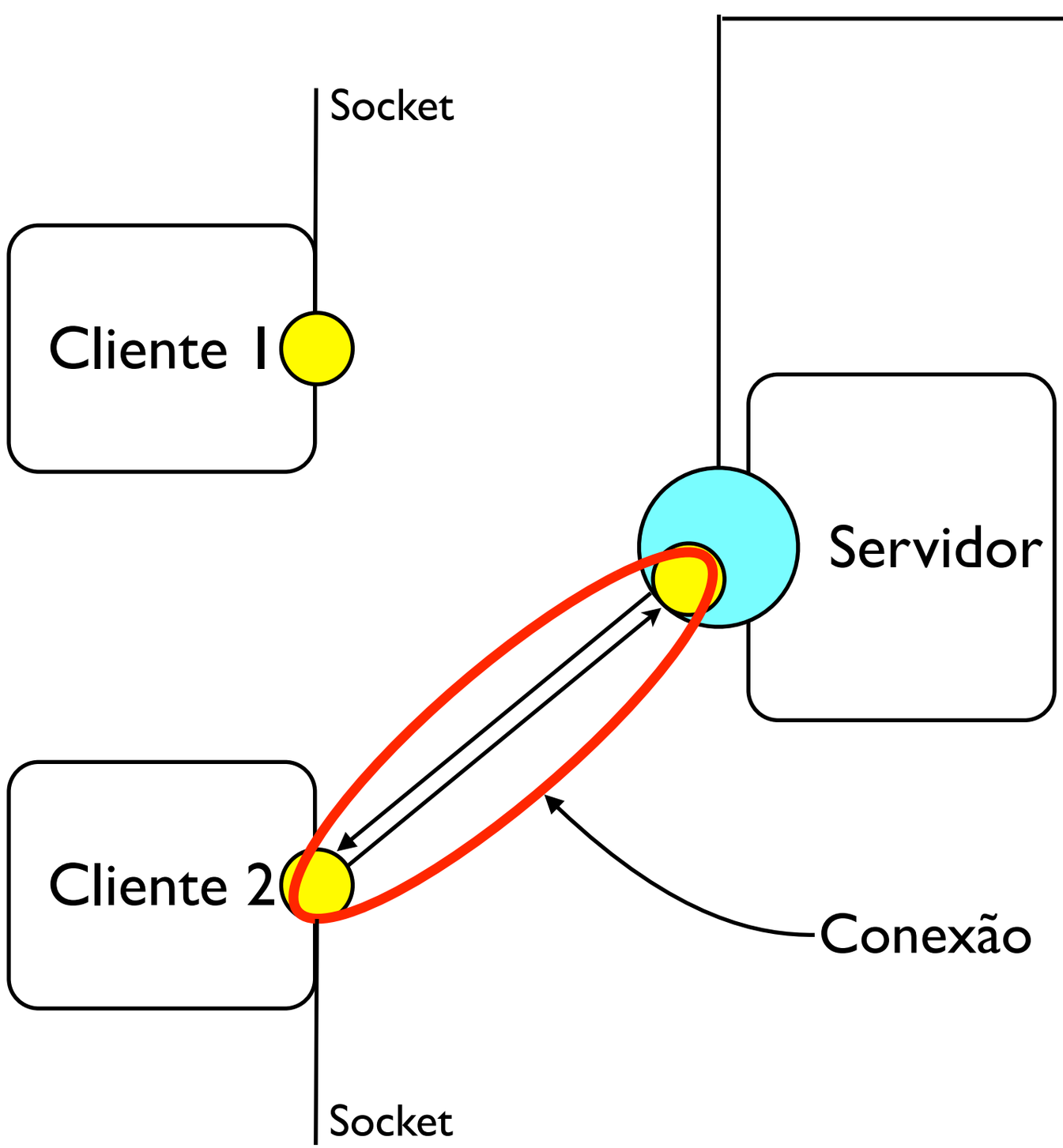
# Sistemas Distribuídos

Universidade do Minho

2011/2012




Server Socket TCP/IP:

- endereço (ip);
- porto 16 bits-> distinguem serviços na mesma máquina;
  - 0–1023 são standard ex: http 80
  - 1024–49151
  - 49152-65535 dinâmicos

- Servidor fica à espera de ligações;
- quando o cliente se liga é estabelecida uma conexão, bidireccional;
- Socket representa um extremo de uma conexão.

- Classes e métodos relevantes:
  - java.net.io.*, java.net.ServerSocket
  - – métodos relevantes: ServerSocket(), accept(), close()
  - – outros métodos: setReuseAddress(), bind()
  - java.net.Socket
    – métodos relevantes: Socket(), connect(), read(),write(),getInputStream(),getOutputStream()
  - – outros métodos: shutdowInput(), shutdownOutput()

# Aula 7: Socket JAVA

## Cliente

Esqueleto:

socket()
connect()

JAVA:

```
Socket socket = new Socket(remotehost,port);
```

while ()

write()
read()

```
out.write(...);
out.flush();
```

close()

```
socket.shutdownInput();
socket.shutdownOutput();
socket.close();
```

## Servidor

Esqueleto:

socket()
bind()

JAVA:

```
ServerSocket sSock = new
ServerSocket(porto);
```

listen()
while ()
accept()

```
while (true){ //para aceitar
conexões indefinidamente
Socket sock = sSock.accept()
//fica à escuta e bloqueia até que
uma conexão seja estabelecida
BufferedReader in=new BufferedReader(new
InputStreamReader(sock.getInputStream()))
BufferedWriter out = new
BufferedWriter(new
OutputStreamWriter(sock.getOutputStream()
))
...
while(...){
```

while ()

read()
write()

```
in.readLine();
out.write(...);
out.flush();
}
```

close()

```
sock.shutdownInput();
sock.shutdownOutput();
sock.close();
```

# Aula 8: ex1

- Implemente um servidor e um cliente da classe Banco desenvolvida anteriormente.
- Dica:
  - As classes BufferedReader e BufferedWriter são particularmente eficazes quando se pretende ler e escrever no canal de comunicação strings textuais.
  - Mas para serializar objectos no canal de comunicação de forma mais eficaz (exemplo: inteiros, doubles,...) usar o ObjectOutputStream e ObjectInputStream.

```
BufferedReader in=new BufferedReader(new
InputStreamReader(sock.getInputStream()))
BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(sock.getOutputStream()))
```

```
ObjectOutputStream oos =new ObjectOutputStream
(socket.getOutputStream());
ObjectInputStream ois =new ObjectInputStream
(socket.getInputStream());

oos.writeInt(1);
oos.writeChar("ola")
oos.flush();
...
ois.readDouble();
ois.readInt();
```

# Aula 8: ex1

~~Acesso implements Runnable~~

**Banco Client**

**Banco Server**

Banco

Contas

Contas

...

# Aula 8: ex extra

- Modificar o servidor de eco do exercício 1 de modo a ser multi-threaded, ou seja para aceitar várias conexões simultâneas de diferentes clientes. Teste, por exemplo, com 2 conexões telnet ou usando o cliente eco.