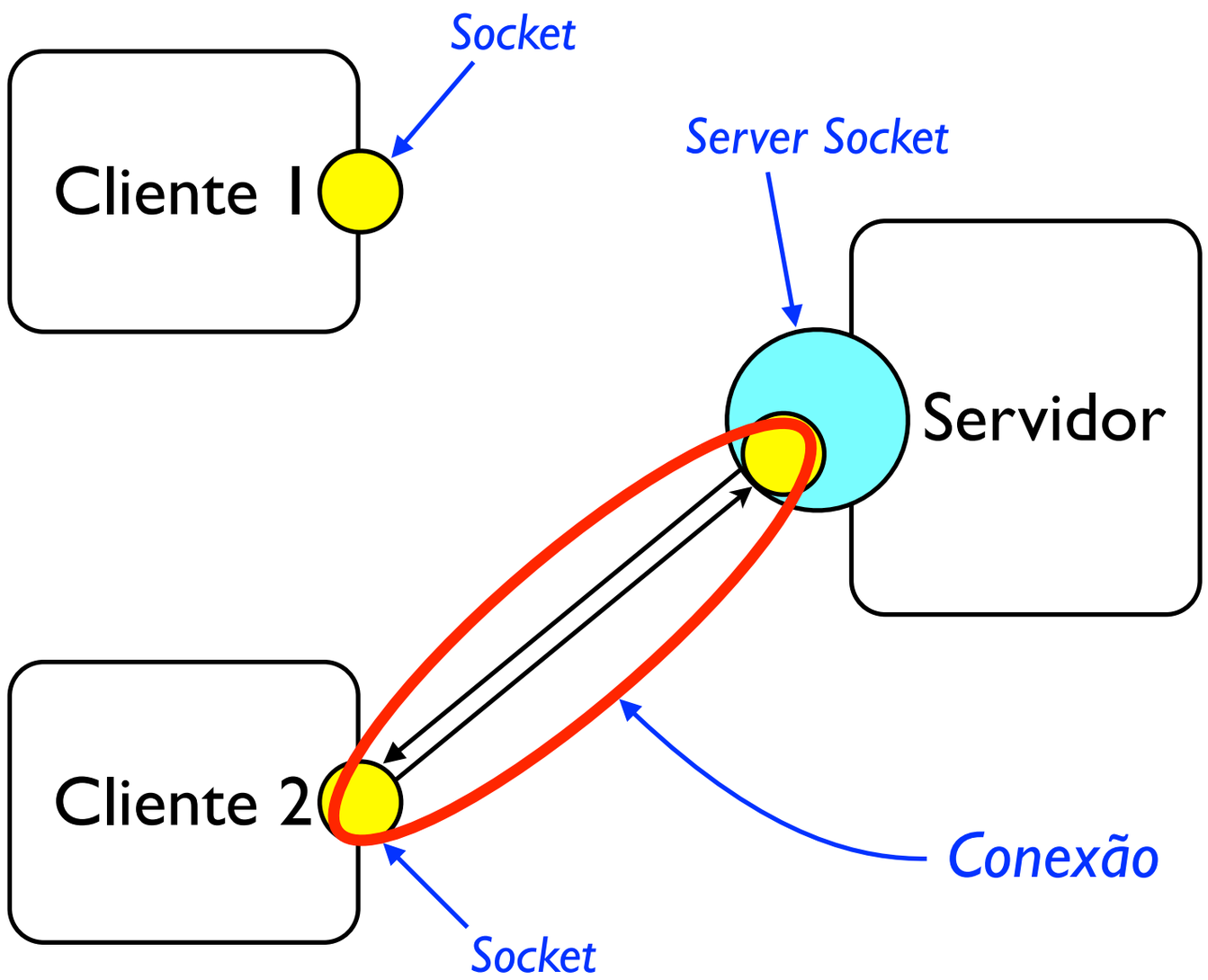
Cliente-Servidor com Sockets TCP

Sistemas Distribuídos

Paradigma cliente - servidor

- Servidor oferece um serviço ao cliente (e.g. servidor web envia html da página pedida, servidor de mail entrega novos emails, …)
- Cliente efetua pedidos ao servidor e trata as respostas (e.g. cliente do browser mostra a página web recebida do servidor, …)
  - O cliente tipicamente inicia o contacto com o servidor
- Comunicação deve de ser fiável (sem perda dados e com entrega ordenada de mensagens) → canais TCP

Paradigma cliente - servidor



- Servidor fica à espera de ligações num determinado porto;

- Quando o cliente se liga ao servidor é estabelecida uma nova conexão bidireccional;

- Socket representa um extremo de uma conexão.

- Numa ligação, existem dois extremos (socket)

- Classes e métodos relevantes em JAVA:
  - (servidor) java.net.ServerSocket:
    - construtor: ServerSocket(int port)
    - outros métodos: accept(), close()
  - (cliente) java.net.Socket:
    - construtor: Socket(String host, int port)
    - outros métodos: connect(), getInputStream(), getOutputStream()
- ler / escrever no socket: BufferedReader, InputStreamReader / PrintWriter

Cliente

Esqueleto:

criar socket e ligação com o servidor

abrir canais de escrita e leitura no socket

ler e escrever nos canais de acordo com o protocolo da aplicação

fechar socket e respectivos canais

JAVA:

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

BufferedReader in = new BufferedReader(new
 InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(
 socket.getOutputStream());

while(...) {
   out.println(…);
   out.flush();
   ... = in.readLine();
}

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

Servidor

Esqueleto:

criar novo server socket num dado porto

aceitar conexões indefinidamente

bloquear até que uma conexão seja estabelecida

abrir canais de escrita e leitura no socket

ler e escrever nos canais de acordo com o protocolo da aplicação

fechar socket e respectivos canais

JAVA:

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

while (true)
{
   Socket clSock = sSock.accept();

   BufferedReader in = new BufferedReader(new
    InputStreamReader(clSock.getInputStream()));

   PrintWriter out = new PrintWriter(
    clSock.getOutputStream());

   while(...) {
      ... = in.readLine();
      out.println(…);
      out.flush();
   }

   clSock.shutdownOutput();
   clSock.shutdownInput();
   clSock.close();
}
```

Exercícios

1) Implemente um servidor que aceite a ligação de um cliente de cada vez, e que devolva ao cliente cada linha de texto que este lhe envie até o cliente fechar a ligação.

2) Implemente um cliente para o servidor desenvolvido no exercício anterior.

- Sugestão:
  - Usar endereço "127.0.0.1" e porto 12345 na criação dos sockets
  - Fechar ligação quando se escrever "quit" no cliente

Exercícios

3) Modifique o servidor acima de modo a permitir tratar vários clientes concorrentemente, dedicando uma thread a cada cliente.