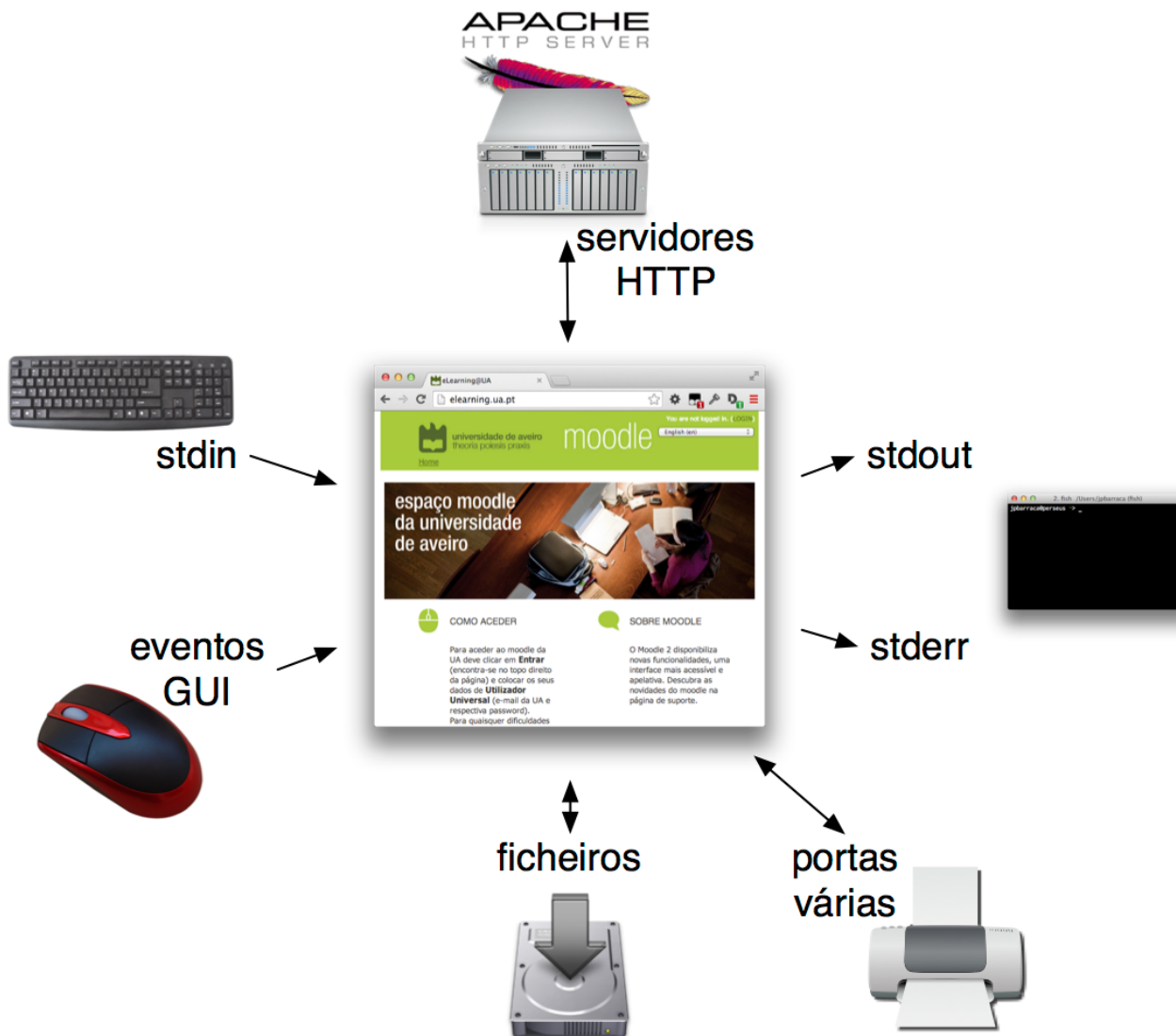


COMUNICAÇÃO ENTRE APLICAÇÕES

Comunicação

- Aplicações interagem de várias formas
 - ▣ com o utilizador (stdin, stdout, stderr)
 - ▣ com dispositivos (portas USB, Serie, etc..)
 - ▣ ficheiros
 - ▣ eventos de interação (apontadores)
 - ▣ **com outras aplicações**



Exemplo: Navegador Web

Comunicação entre Aplicações

- Mecanismo muito importante!
 - ▣ Aplicações na Internet
 - ▣ Aplicações e servidor gráfico (X11)
- Implementado de diferentes formas
 - ▣ PIPE – Redireccionamento através de stdin e stdout
 - ▣ Socket – Ligação de entre aplicações
 - ▣ Outros

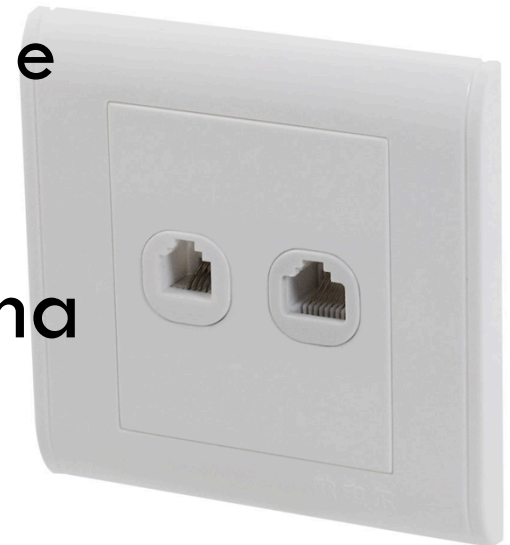
PIPE

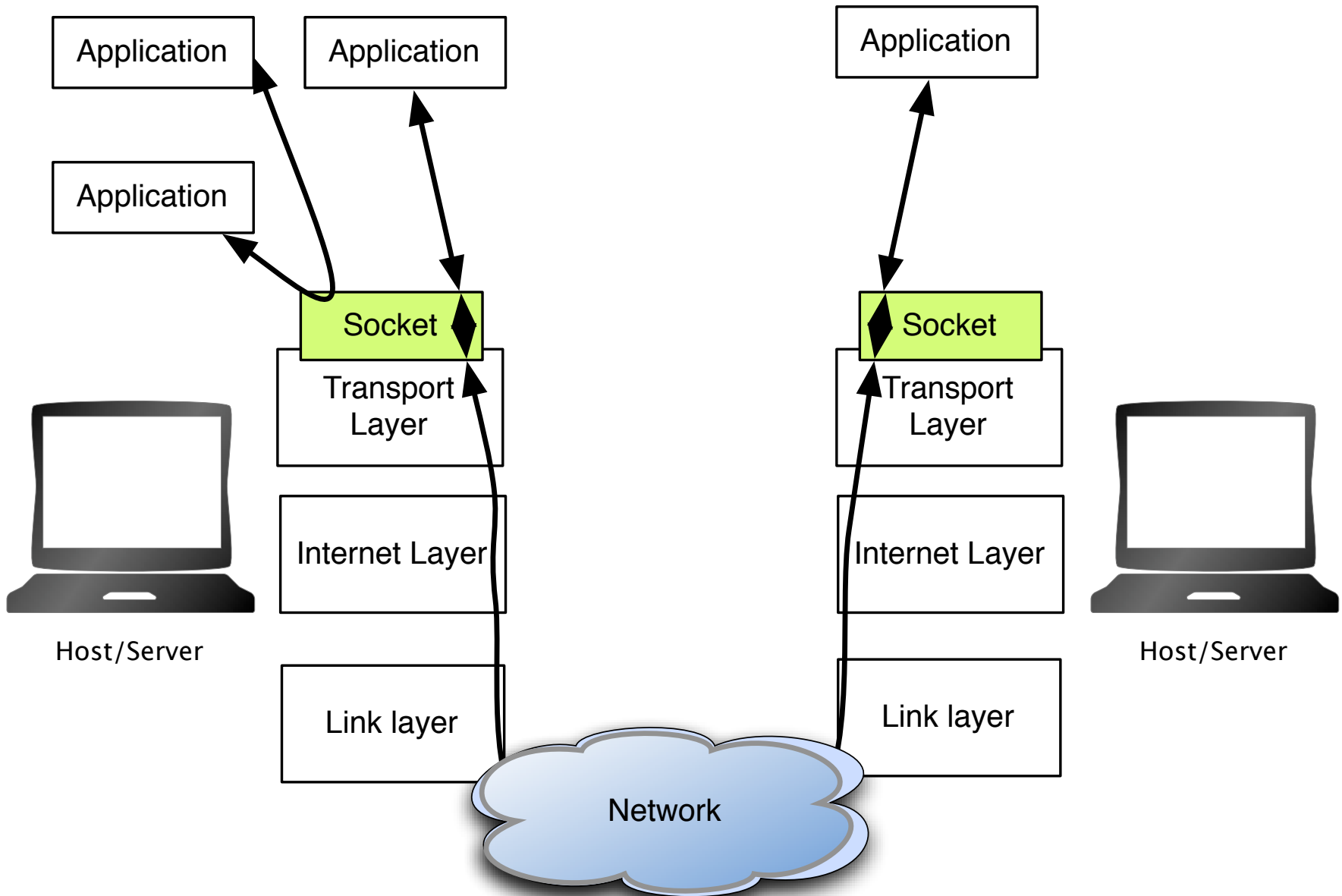
- ❑ Mecanismo básico
 - ❑ Pode ser transparente para aplicações
- ❑ Âmbito local (mesma máquina)
- ❑ Liga stdout/stderr a stdin de aplicações
- ❑ Exemplo: contar interfaces de rede

```
$ ifconfig -a | grep 'Link encap' | wc -l  
4
```

Socket

- Abstração semelhante a uma ficha de parede
- Aplicações definem características
 - ▣ ficam abertos para comunicação
 - ▣ tal como uma ficha tem formato e contatos específicos
- Comunicação interna ou externa





Socket: Características

1. **Família**
2. **Tipo**
3. **Nome**

□ Restrição: só 1 socket da mesma família, tipo e nome.

Socket: Família

- Indica qual o protocolo de rede a usar
- AF_UNIX: comunicações locais (sem IP)
- AF_INET/AF_INET6: usar IPv4/IPv6
 - ▣ Para comunicações locais ou remotas
- ... outros

Socket: Tipo

- Define o protocolo de transporte a usar
- SOCK_DGRAM: Não orientado à ligação
 - ▣ Usa UDP
- SOCK_STREAM: Orientado à ligação
 - ▣ Usa TCP

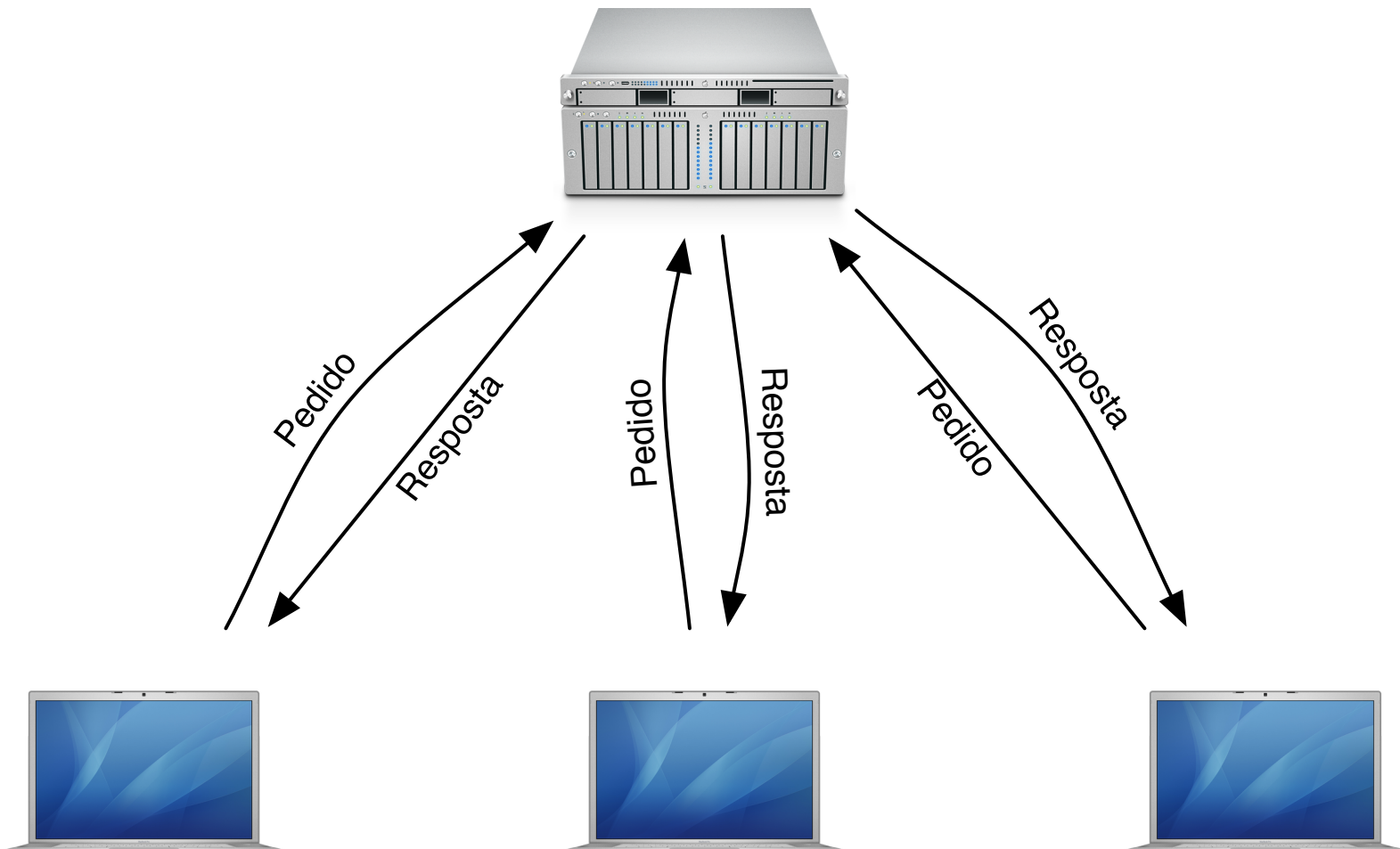
Socket: Nome

- Identifica qual o socket num sistema
 - ▣ Formato depende da família
- AF_UNIX: usa nome de ficheiro
 - ▣ Ex: /tmp/run.sock
- AF_INET/AF_INET6: usa endereço e porta
 - ▣ Um interface: 127.0.0.1:1234
 - ▣ Todos os interfaces: 0.0.0.0:1234

Modelo Cliente-Servidor

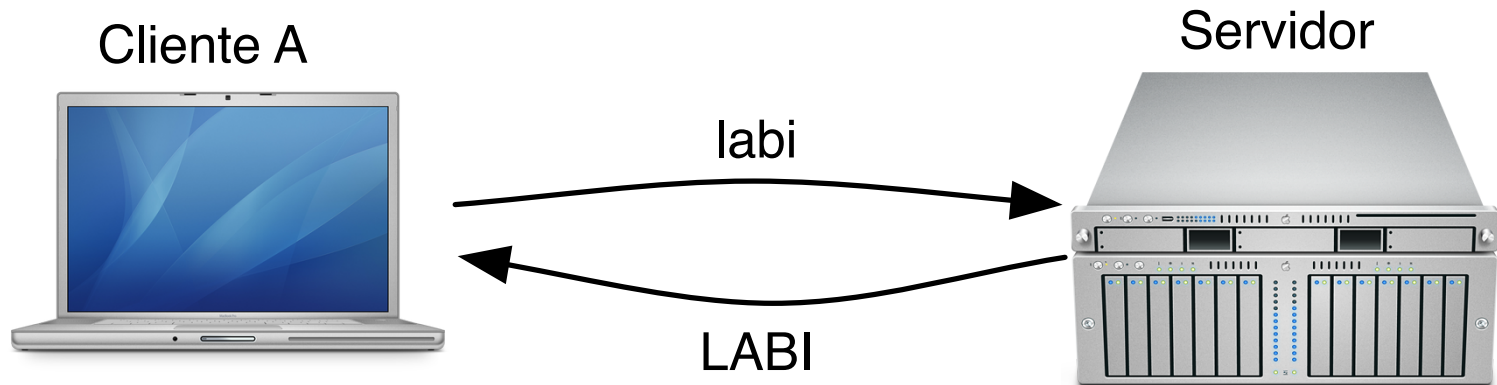
- Sockets assumem que existe separação de funções
- Cliente: efetua pedidos
 - ▣ Ex: navegador
- Servidor: responde a pedidos
 - ▣ Ex: servidor HTTP

Modelo Cliente-Servidor

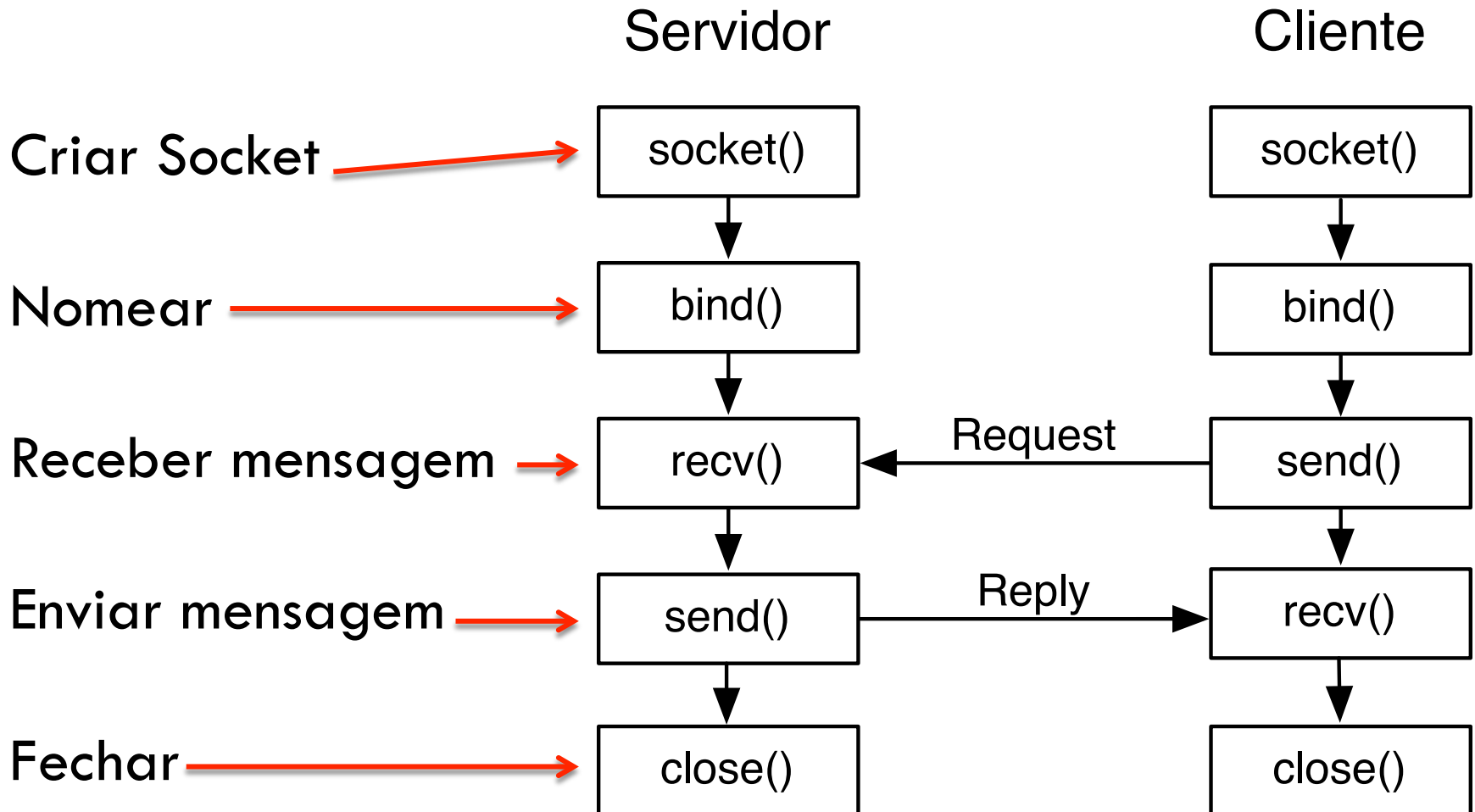


Exemplo: UDP Echo Server

- ❑ Cliente: envia mensagem para servidor
- ❑ Servidor: responde com mensagem em maiúsculas



Ações: UDP Echo



UDP: Criar sockets (servidor)

```
from socket import *
```



Importar módulo

```
udp_s = socket(AF_INET, SOCK_DGRAM)
```

```
udp_s.bind( ('0.0.0.0', 1234) )
```



Nomear Socket

(todos os interfaces, porta 1234)



Criar Socket

UDP: Transmitir informação (servidor)

Dados recebidos

Endereço de origem

**Receber mensagem
até 4096 bytes**

```
data, addr = udp_s.recvfrom(4096)
```

```
udp_s.sendto(data.upper(), addr)
```

```
udp_s.close()
```

Fechar Socket

Enviar Resposta

UDP: Criar sockets (cliente)

```
from socket import *
```



Importar módulo

```
udp_s = socket(AF_INET, SOCK_DGRAM)
```

```
udp_s.bind( ('0.0.0.0',0) )
```



Nomear Socket

(todos os interfaces, porta aleatória)



Criar Socket

UDP: Transmitir informação (cliente)

Dados recebidos

Endereço de destino

Enviar Mensagem

```
udp_s.sendto('labi', ('127.0.0.1', 1234) )
```

```
data, addr = udp_s.recvfrom(4096)
```

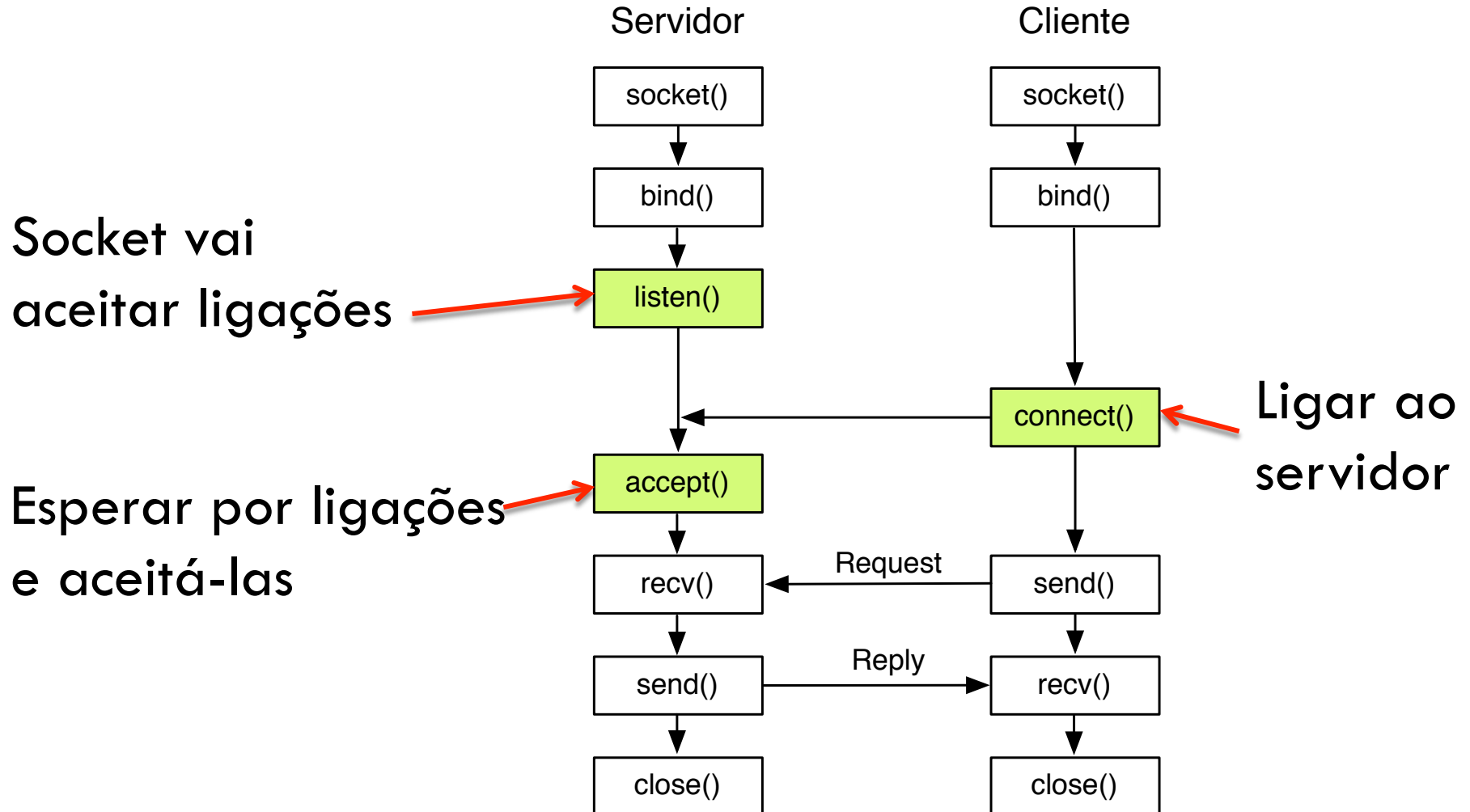
```
print data
```

```
udp_s.close()
```

**Receber resposta
até 4096 bytes**

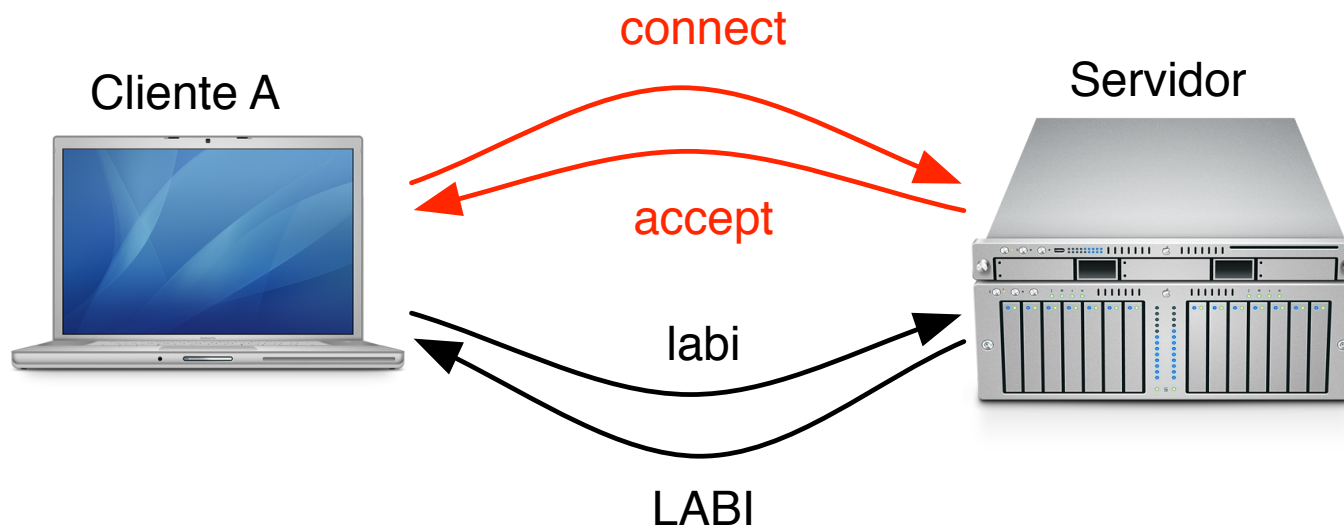
Fechar Socket

Ações: TCP



Exemplo: TCP Echo Server

- ❑ Cliente: estabelece ligação e envia mensagem para servidor
- ❑ Servidor: aceita ligação e responde com mensagem em maiúsculas



TCP: Criar sockets (servidor)

**Criar Socket
TCP**

**Aceitar
ligações**

```
...  
tcp_s = socket(AF_INET, SOCK_STREAM)  
tcp_s.bind( ('0.0.0.0', 1234) )  
tcp_s.listen(1)  
client_s = tcp_s.accept()
```

**Criado novo Socket
Identifica ligação**

**Nomear Socket
(todos os interfaces, porta 1234)**

TCP: Transmitir informação (servidor)

Dados recebidos

**Receber mensagem
até 4096 bytes**

```
data = client_s.recv(4096)
client_s.send(data.upper())
client_s.close()
tcp_s.close()
```

Enviar Resposta

client_s identifica univocamente ligação servidor <-> cliente

TCP: Criar sockets (cliente)

```
from socket import *
```

 **Importar módulo**

```
tcp_s = socket(AF_INET, SOCK_STREAM)
```

```
tcp_s.bind( ('0.0.0.0',0) )
```

```
tcp_s.connect( ('127.0.0.1', 1234))
```

 **Ligar ao servidor**

 **Criar Socket**

TCP: Transmitir informação (cliente)

Dados recebidos

Enviar Mensagem

**Receber resposta
até 4096 bytes**

```
tcp_s.send('labi')
```

```
data = tcp_s.recv(4096)
```

```
print data
```

```
tcp_s.close()
```

tcp_s identifica ligação cliente<->servidor