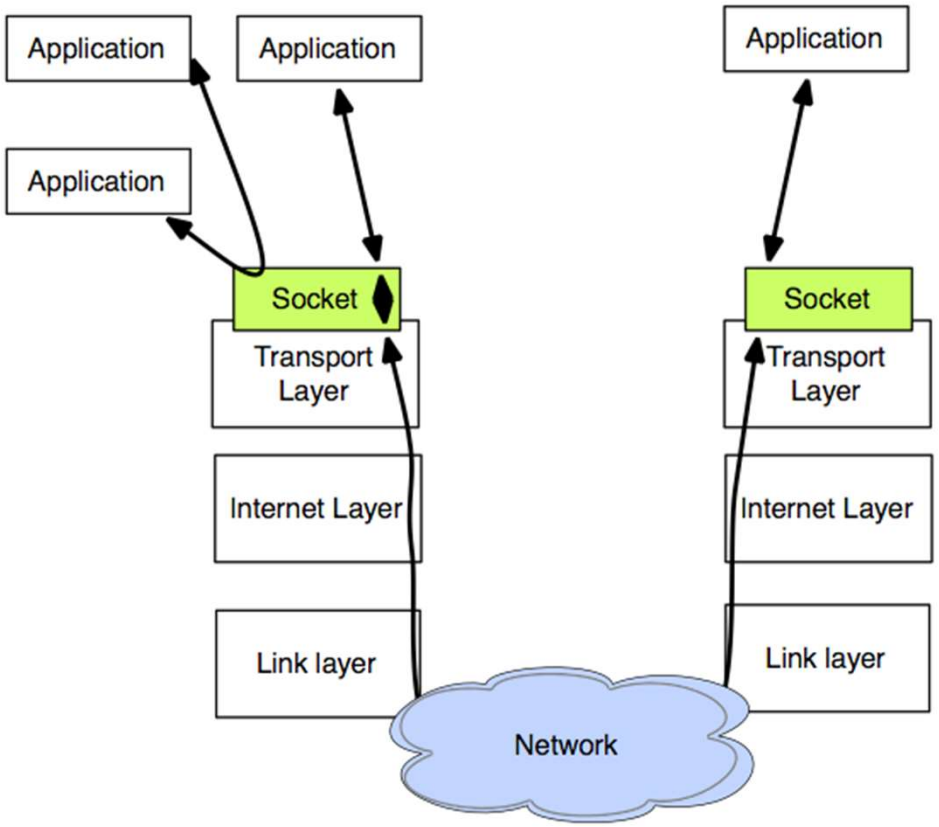


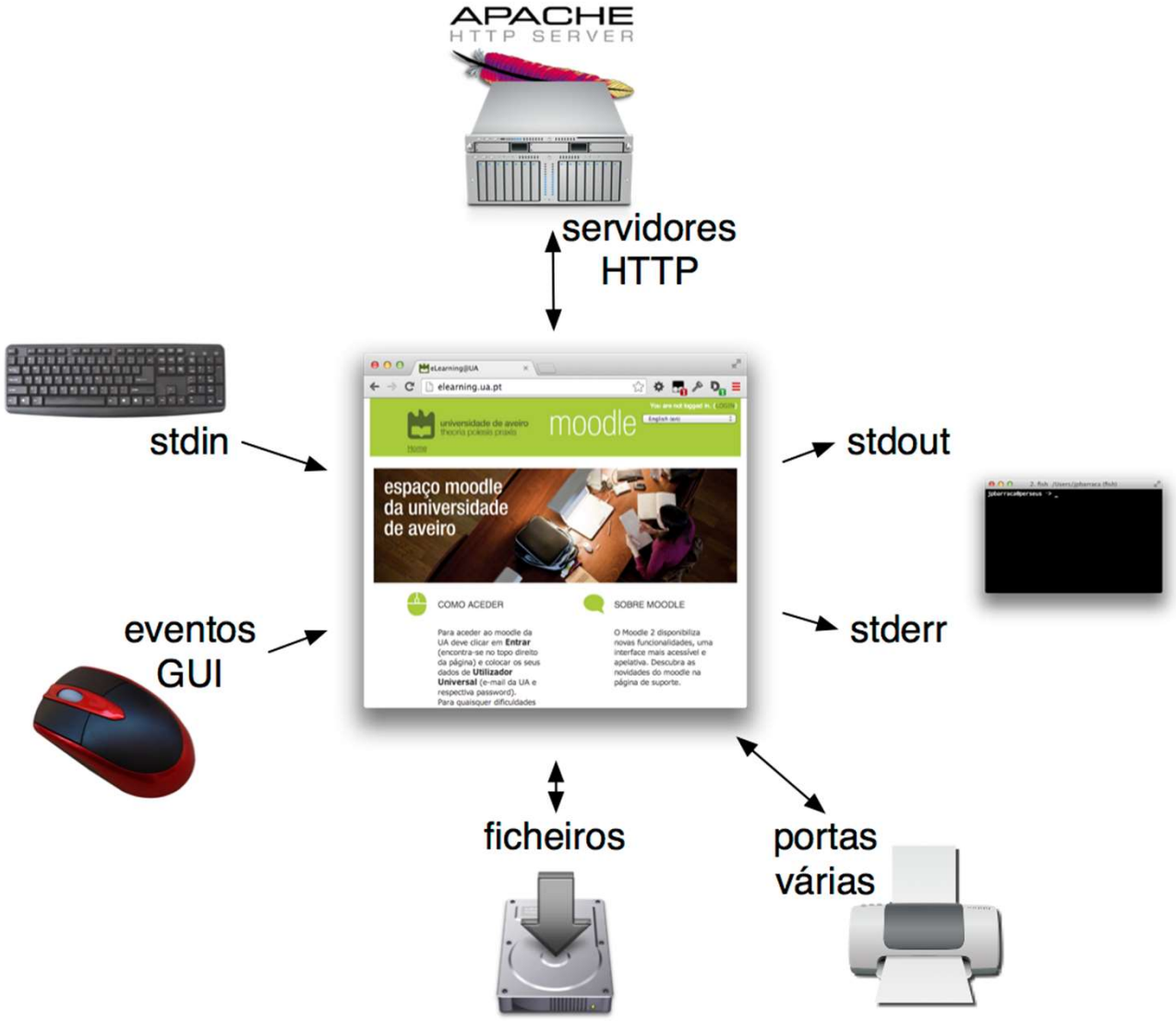
COMUNICAÇÃO ENTRE APLICAÇÕES

universidade de aveiro
departamento de eletrónica, telecomunicações e informática
deti

Laboratórios de Informática

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 ....

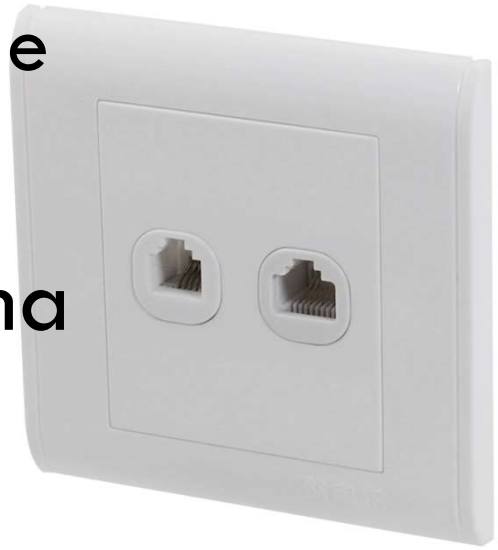
PIPE

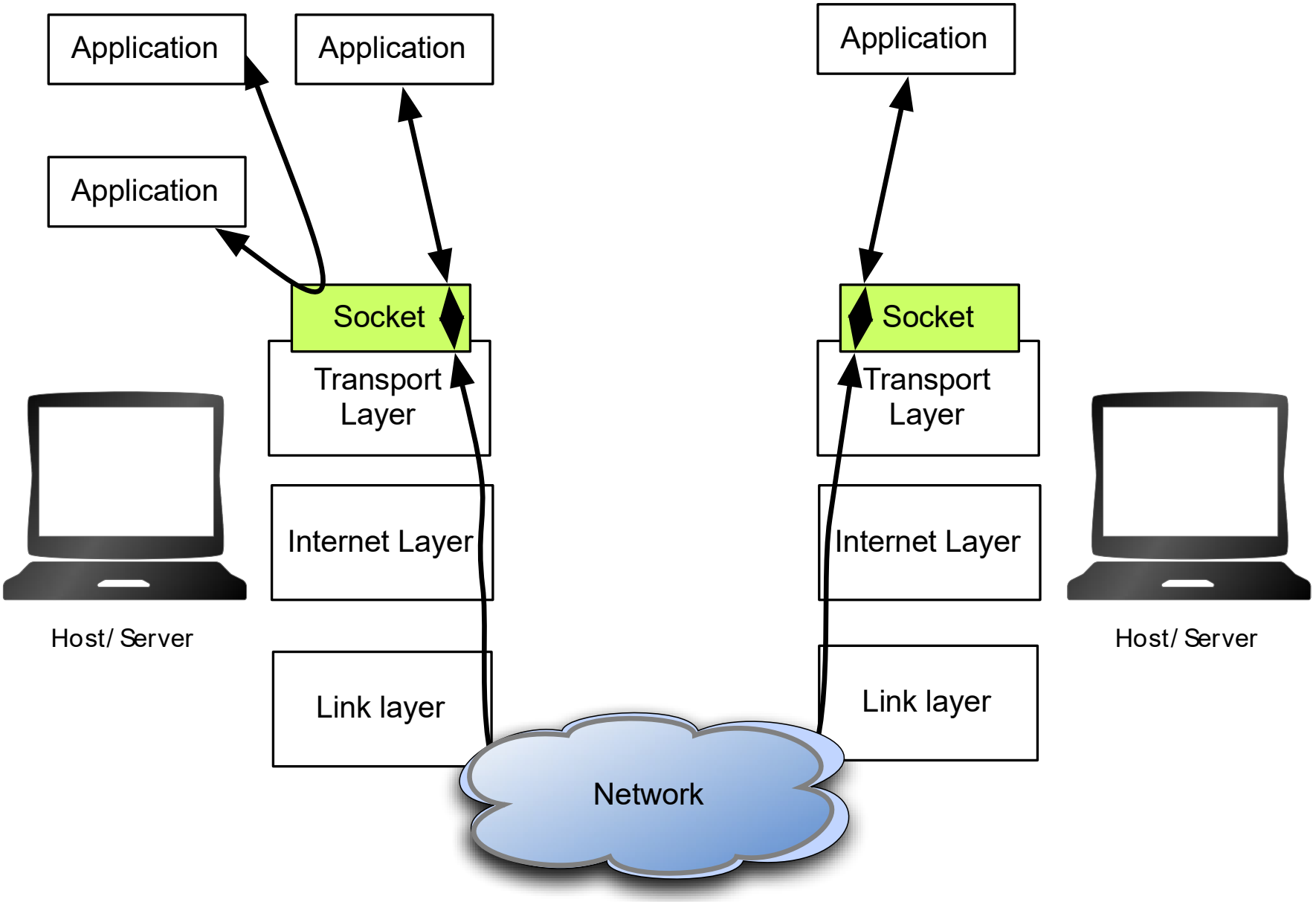
- Mecanismo básico
  - Pode ser transparente para aplicações
- Âmbito local (mesma máquina)
- Liga stdout/stderr a stdin de aplicações
- Exemplo: contar interfaces de rede

```
$ ifconfig -a | grep 'flags' | wc -l
      5
```

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




Application
Application
Application
Application
Socket
Socket
Transport Layer
Transport Layer
Internet Layer
Internet Layer
Link layer
Link layer
Network
Host/ Server
Host/ Server

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

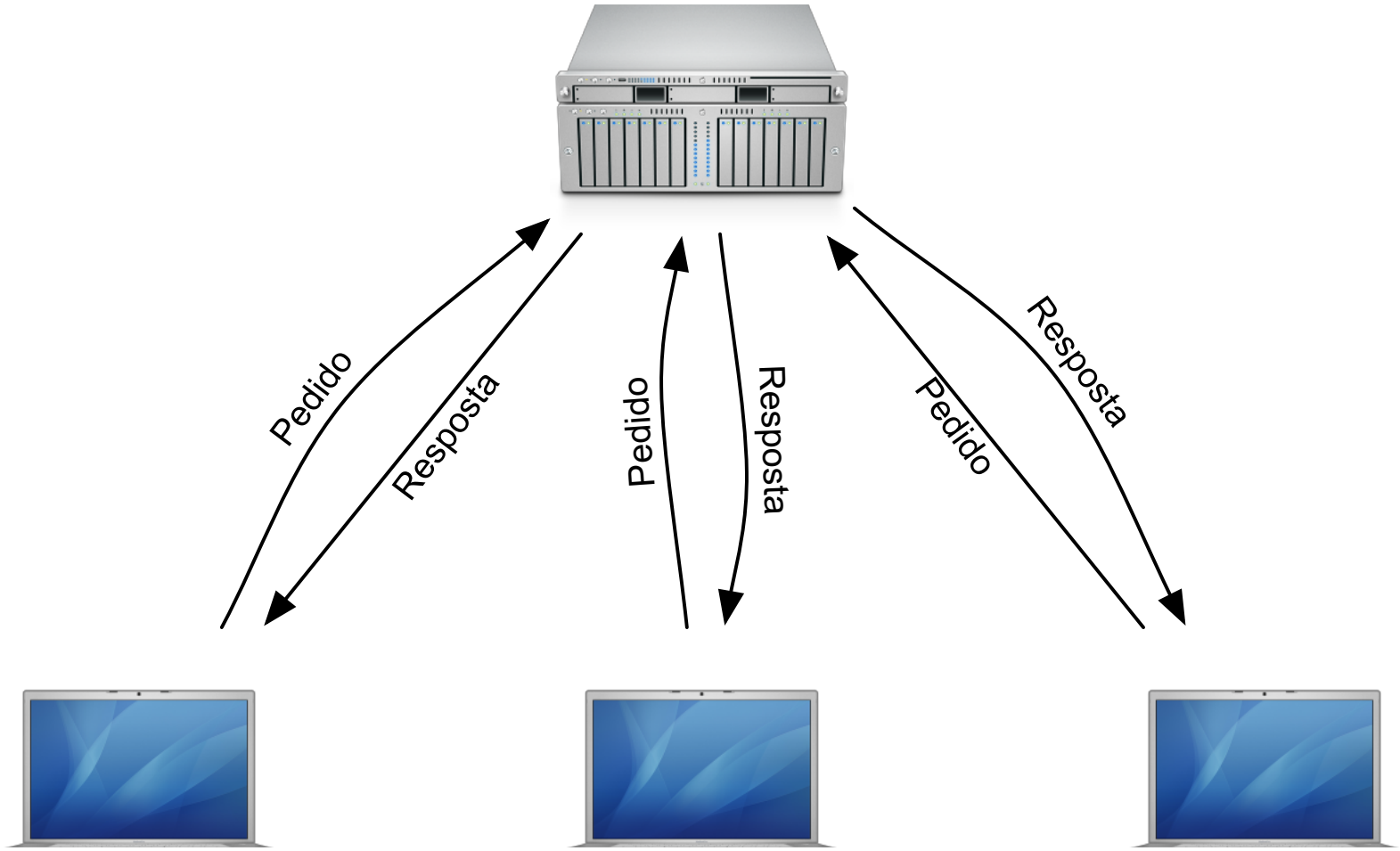
Socket: Nome

- 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

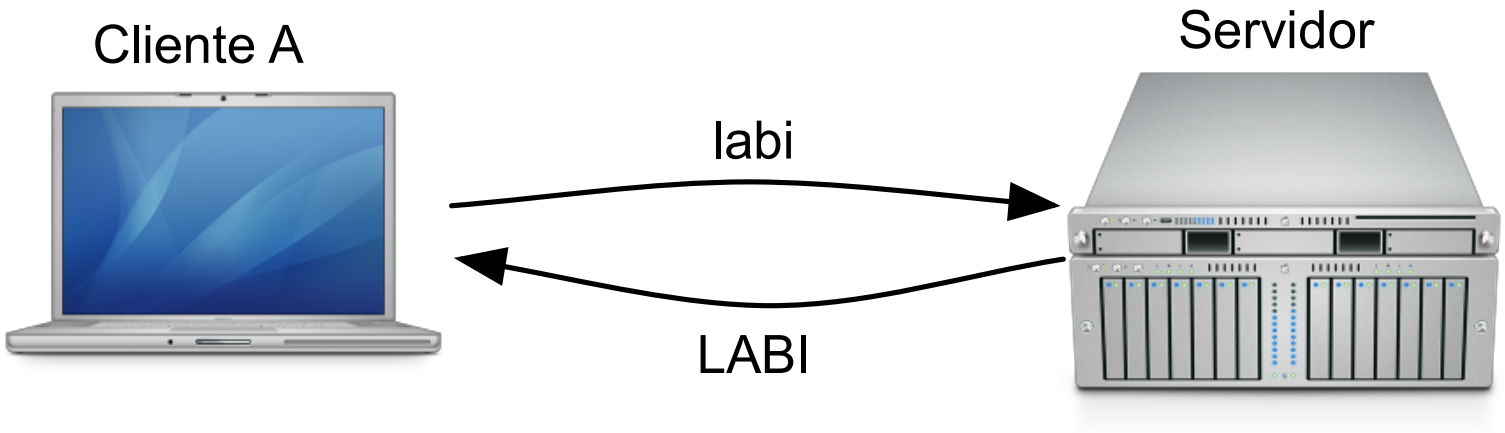
- Sockets assumem que existe separação de funções
- Cliente: efetua pedidos
  - Ex: navegador
- Servidor: responde a pedidos
  - Ex: servidor HTTP

Modelo Cliente-Servidor

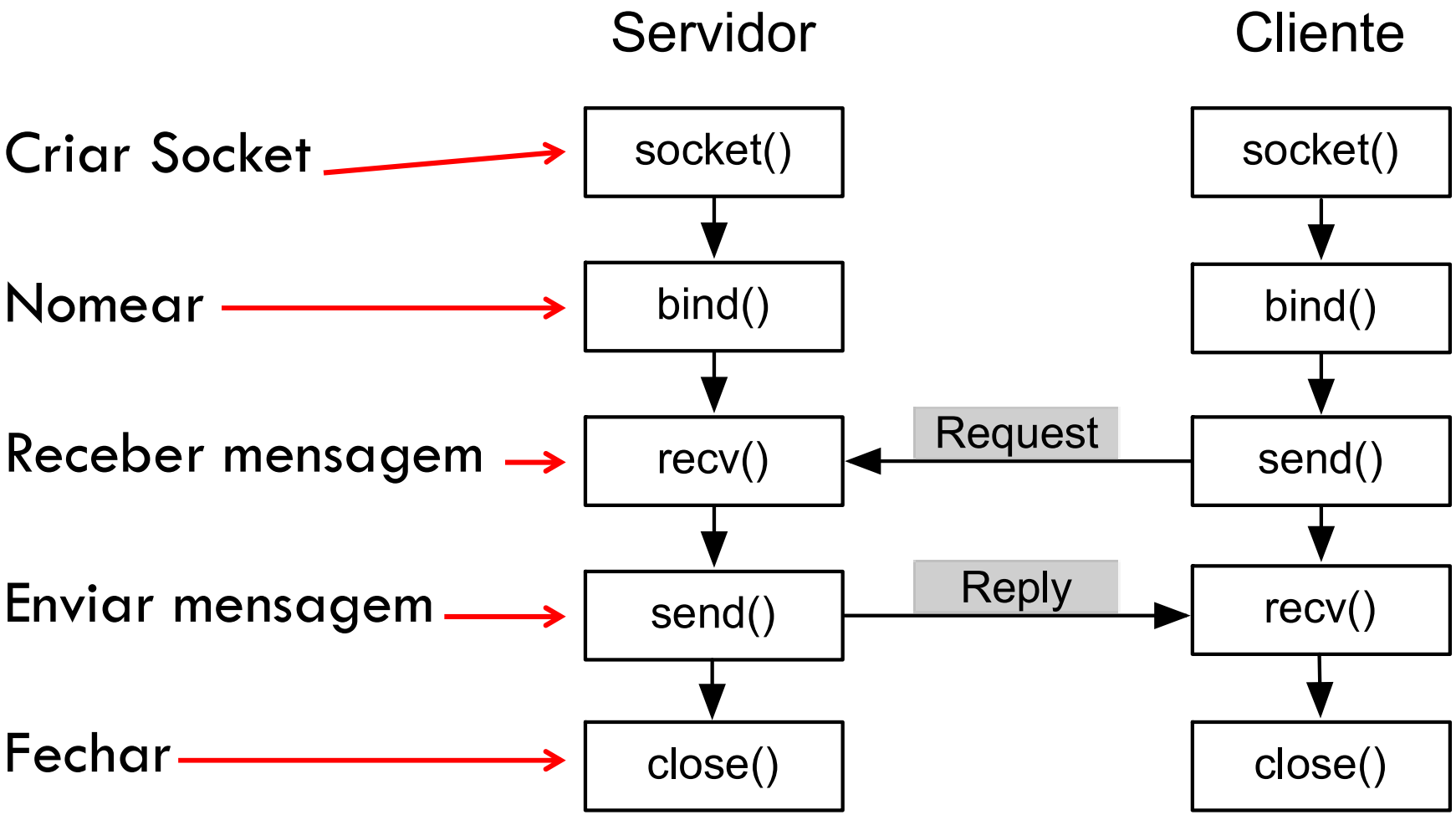


Exemplo: UDP Echo Server

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



Ações: UDP Echo Server



UDP: Criar sockets (servidor)

```
from socket import *


udp_s = socket(AF_INET, SOCK_DGRAM)

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

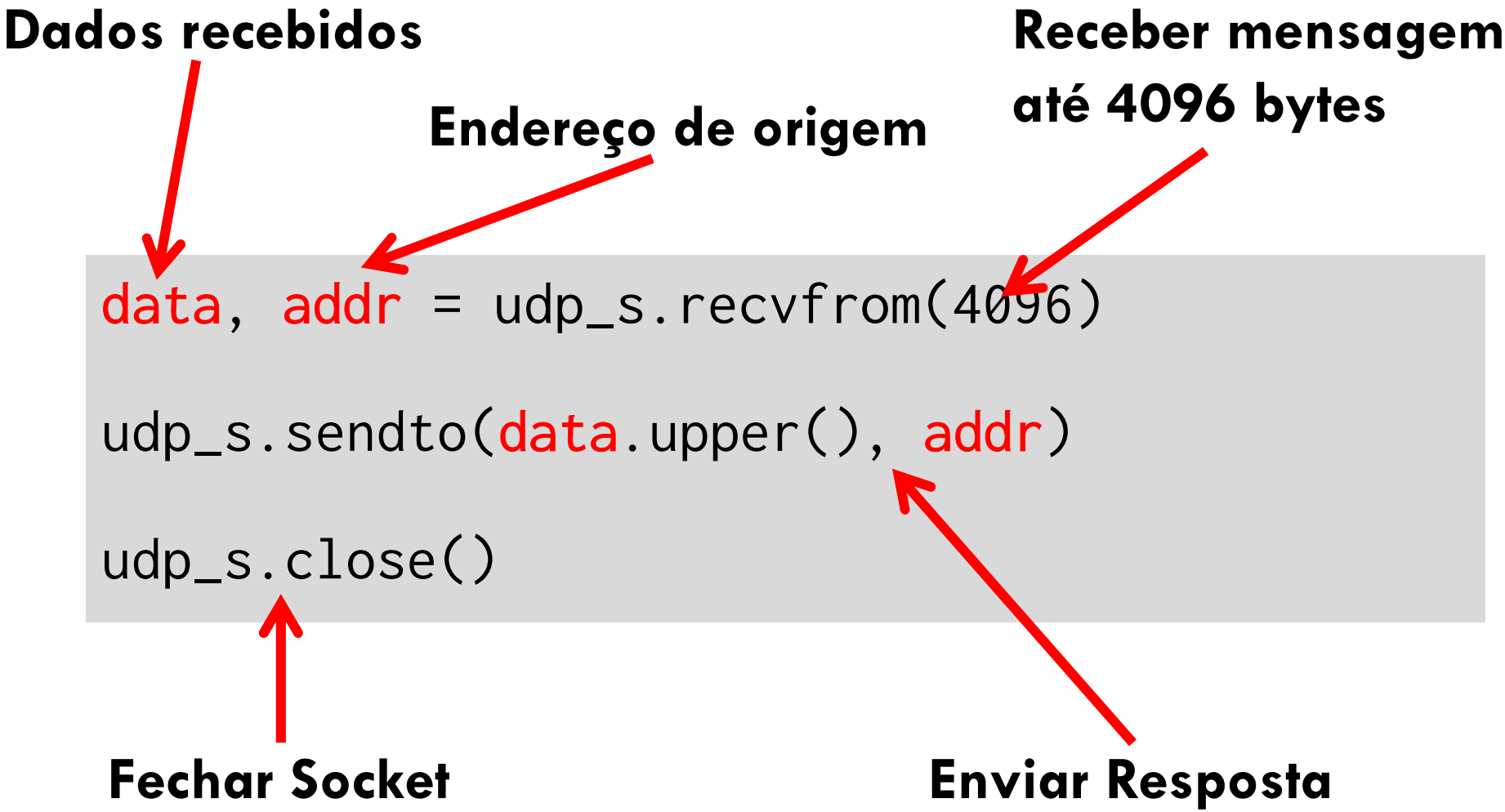
Importar módulo

Criar Socket

Nomear Socket

(todos os interfaces, porta 1234)

UDP: Transmitir informação (servidor)



UDP: Criar sockets (cliente)

```
from socket import *


udp_s = socket(AF_INET, SOCK_DGRAM)

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

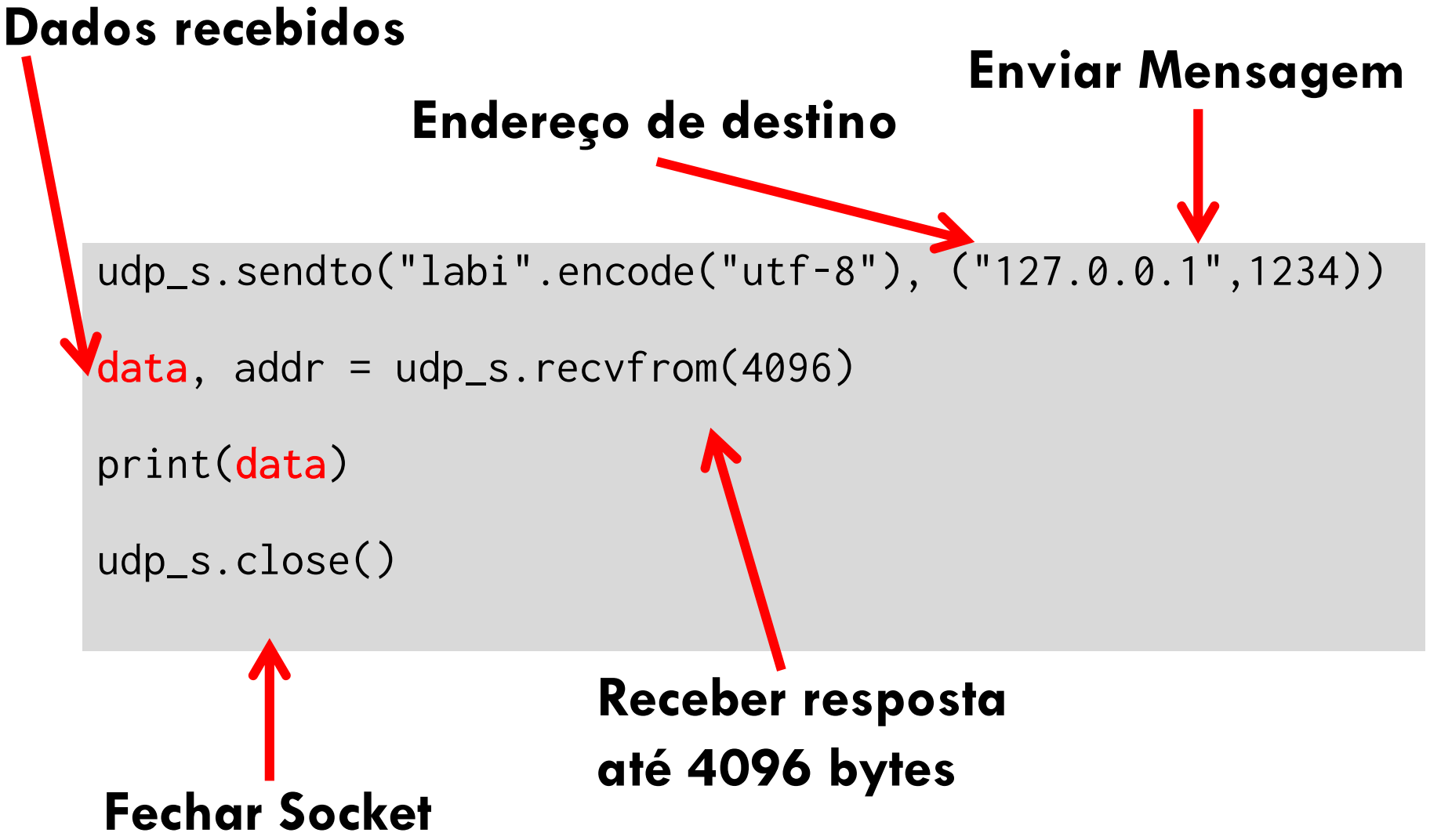
Importar módulo

Criar Socket UDP

Nomear Socket

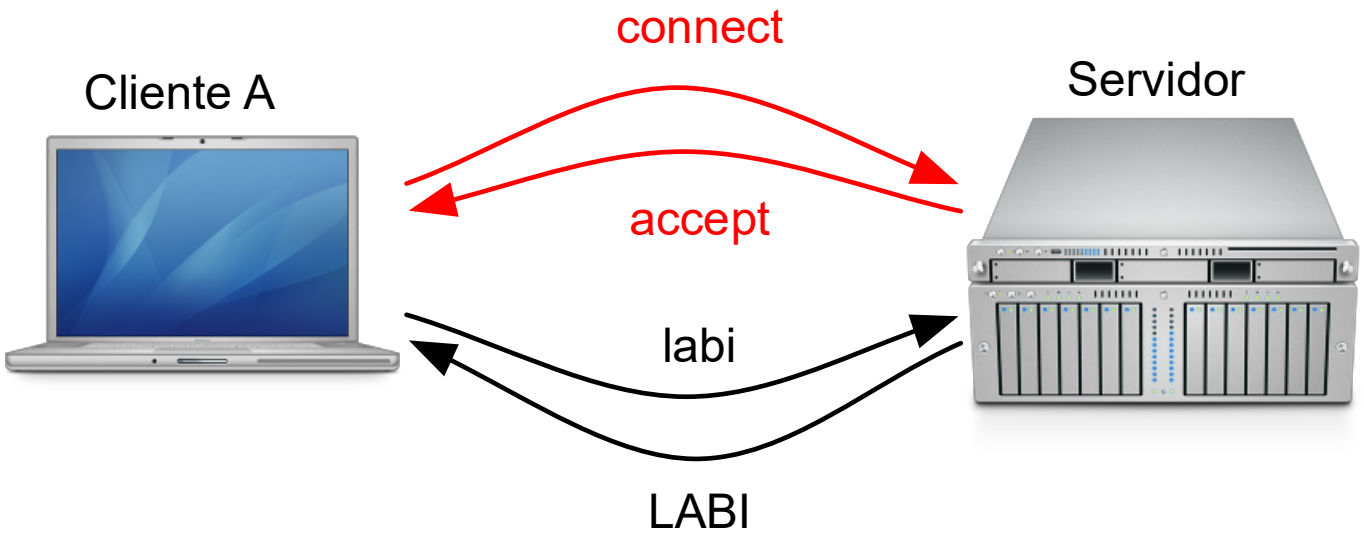
(todos os interfaces, porta aleatória)

UDP: Transmitir informação (cliente)

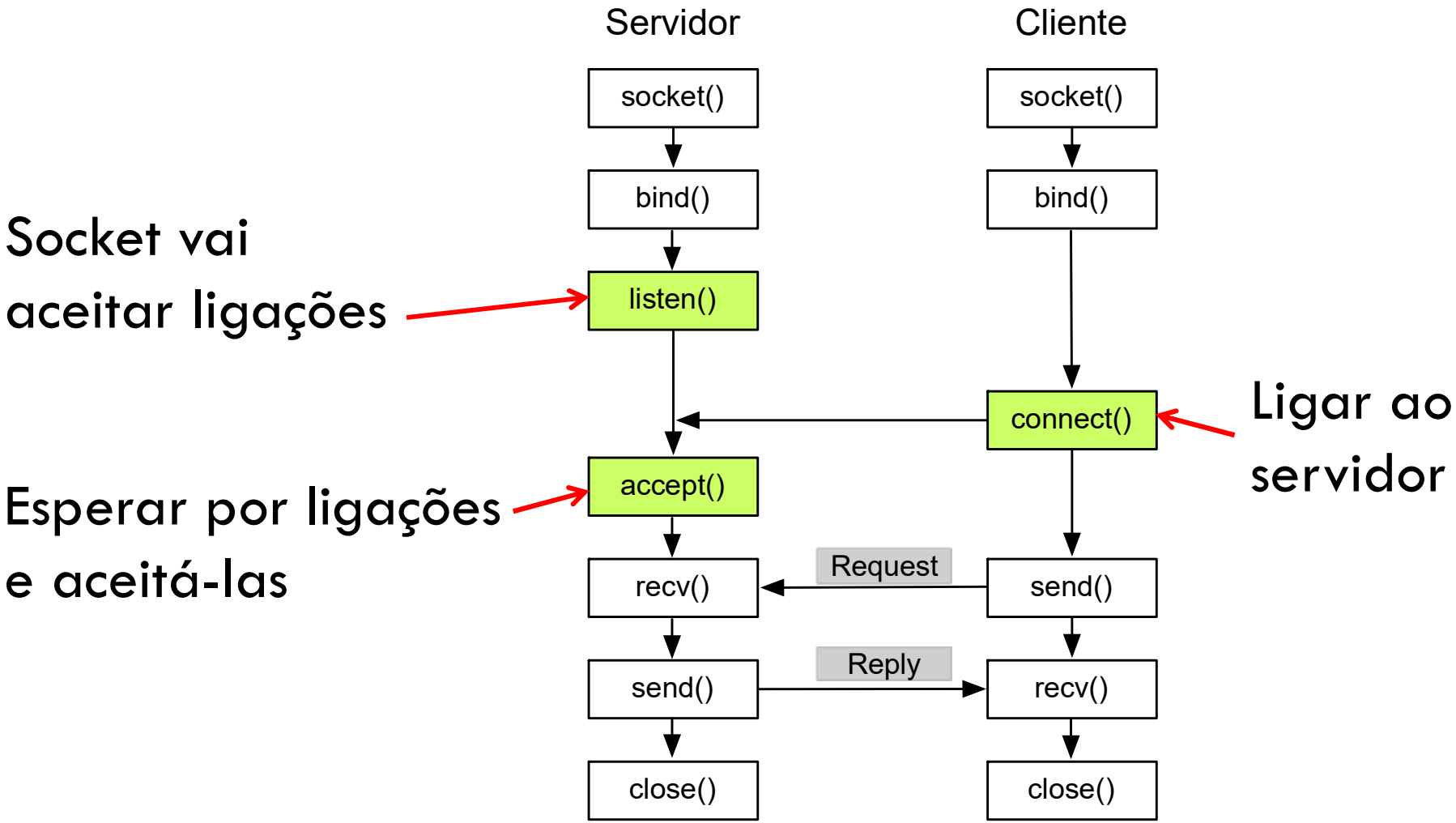


Exemplo: TCP Echo Server

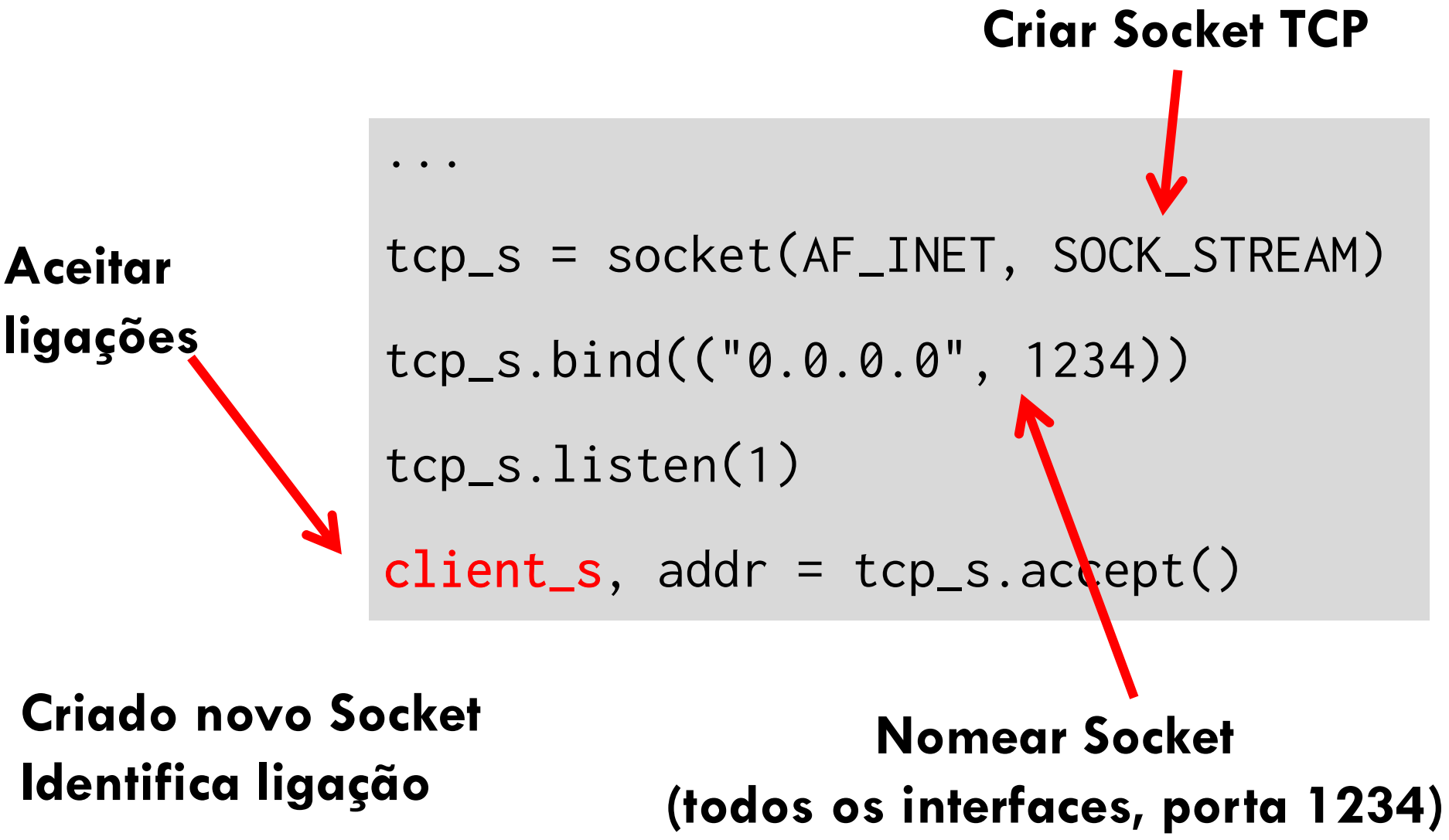
- Cliente: estabelece ligação e envia mensagem para servidor
- Servidor: aceita ligação e responde com mensagem em maiúsculas



Ações: TCP Echo Server



TCP: Criar sockets (servidor)



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 *


tcp_s = socket(AF_INET, SOCK_STREAM)

tcp_s.bind(("0.0.0.0",0))

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

Importar módulo

Criar Socket

Ligar ao servidor

TCP: Transmitir informação (cliente)

Dados recebidos

Enviar Mensagem

Receber resposta até 4096 bytes

```
tcp_s.send("labi".encode("utf-8"))

data = tcp_s.recv(4096)

print(data)

tcp_s.close()
```

tcp_s identifica ligação cliente<->servidor