Programação com Sockets (Python)

Pequena introdução

Conceitos básicos

Um socket é um terminal para comunicação entre processos numa rede de computadores. Correspondem a terminais de canais de comunicação bidirecional.

Um socket permite comunicar dentro de um processo, entre processos numa mesma máquina ou entre processos em máquinas diferentes.

A utilização de sockets é um típico uso da arquitetura por camadas, onde se usa o protocolo de comunicação em pilha ou camadas (*i.e. communication-protocol stacks*).

Conceitos básicos

O uso do conceito protocolo surge da necessidade de estabelecer regras que permitam comunicação entre processos.

Um protocolo é estabelecido para cada uma das camadas de comunicação OSI.

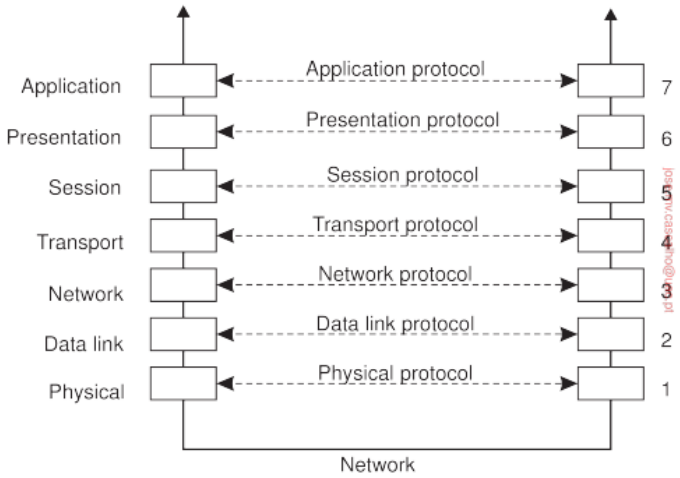

Figure 4.1: Layers, interfaces, and protocols in the OSI model.

Conceitos básicos

Na Internet, usa-se o protocolo da Internet (*i.e. Internet Protocol)*, que se situa ao nível do *Network Protocol*. Neste nível há a preocupação de enviar pacotes de informação na rede e garantir que estes chegam ao destino.

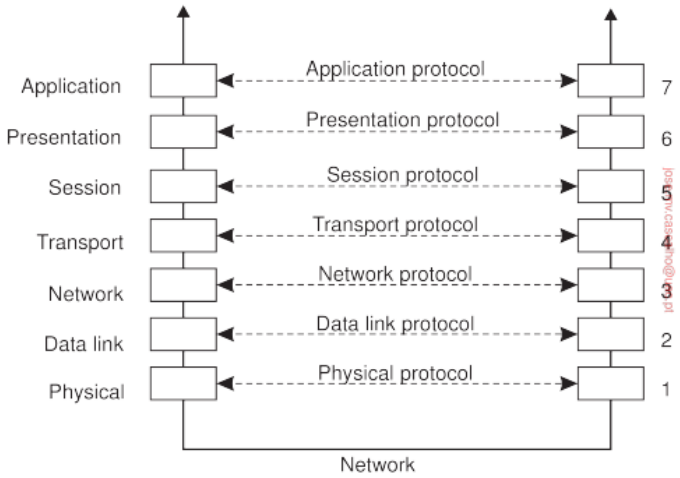

Figure 4.1: Layers, interfaces, and protocols in the OSI model.

Conceitos básicos

Na comunicação entre processos, na utilização dos socket, usa-se o TCP/IP ou UDP que está no nível de transporte (*i.e. transport protocol*).

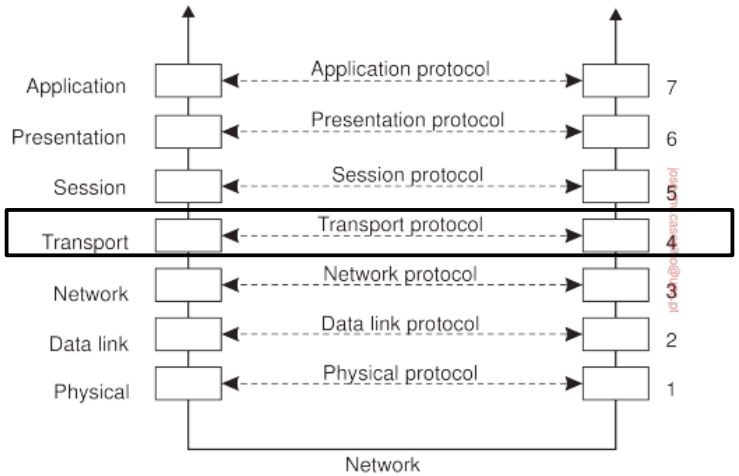

Figure 4.1: Layers, interfaces, and protocols in the OSI model.

Sockets no Python

No Python, os programas importam o módulo socket e criam um objeto socket. Chamam os métodos desse objeto para estabelecer conexões e enviar e receber dados.

O socket permite comunicação ao nível do transporte. Existem dois tipos de comunicação que se podem estabelecer:

– Comunicação com ligação orientada à conexão (*i.e connection-oriented protocol* ou *TCP/IP*).

– Comunicação sem conexão (*i.e. connectionless* ou *UDP*).

Caracterização de um socket

AF_INET é utilizado na Internet (Ipv4)

TCP versus UDP

Um nome do *host* (anfitrião) ou um endereço IP

Porto (ou porta)

Term	Description
domain	The family of protocols that will be used as the transport mechanism. These values are constants such as **`AF_INET`**, **`PF_INET`**, **`PF_UNIX`**, **`PF_X25`**, and so on.
type	The type of communications between the two endpoints, typically **`SOCK_STREAM`** for connection-oriented protocols and **`SOCK_DGRAM`** for connectionless protocols.
protocol	Typically zero, this may be used to identify a variant of a protocol within a domain and type.
hostname	The identifier of a network interface: • A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation • A string "**`<broadcast>`**", which specifies an **`INADDR_BROADCAST`** address. • A zero-length string, which specifies **`INADDR_ANY`**, or • An Integer, interpreted as a binary address in host byte order.
port	Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

Lado do servidor

Python

```python
# echo-server.py

import socket

HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

Lado do cliente

Python

```python
# echo-client.py

import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world")
    data = s.recv(1024)

print(f"Received {data!r}")
```

Verificar a conexão

Utilizando o comando *netstat -an*

Shell

```
$ netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  127.0.0.1.65432        *.*                    LISTEN
```

Passos na comunicação TCP/IP

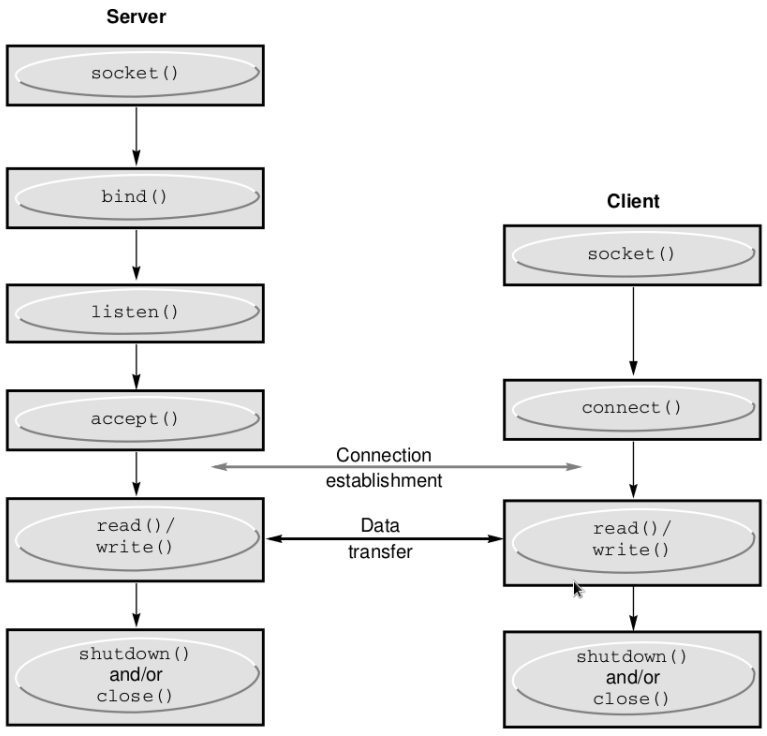

Figure 2-1 Connection-Oriented Communication Using Stream Sockets

Passos na comunicação UDP

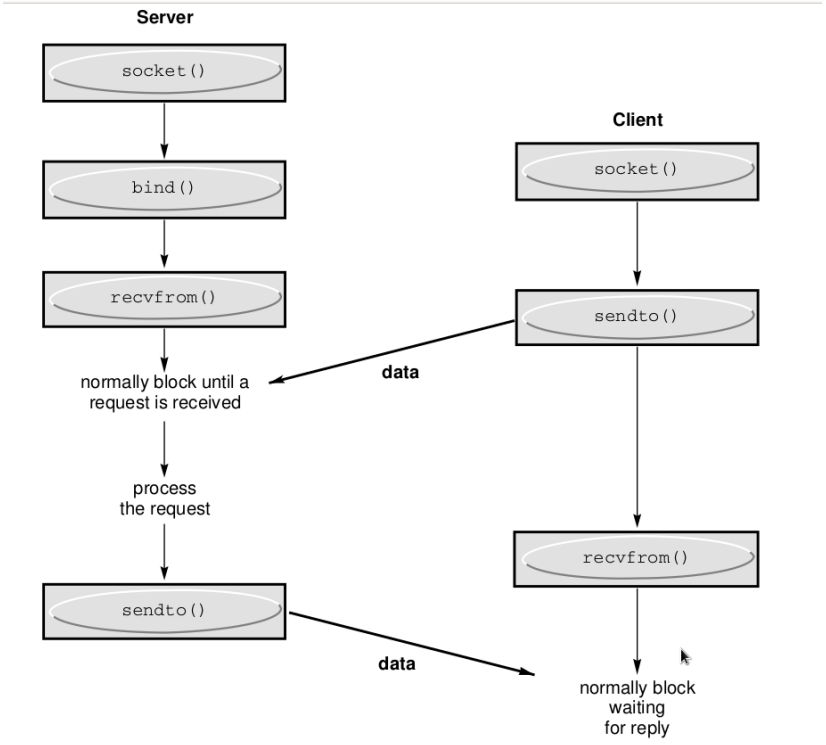

Figure 2-2 Connectionless Communication Using Datagram Sockets