# About Socket and The Python Socket Module

## Background

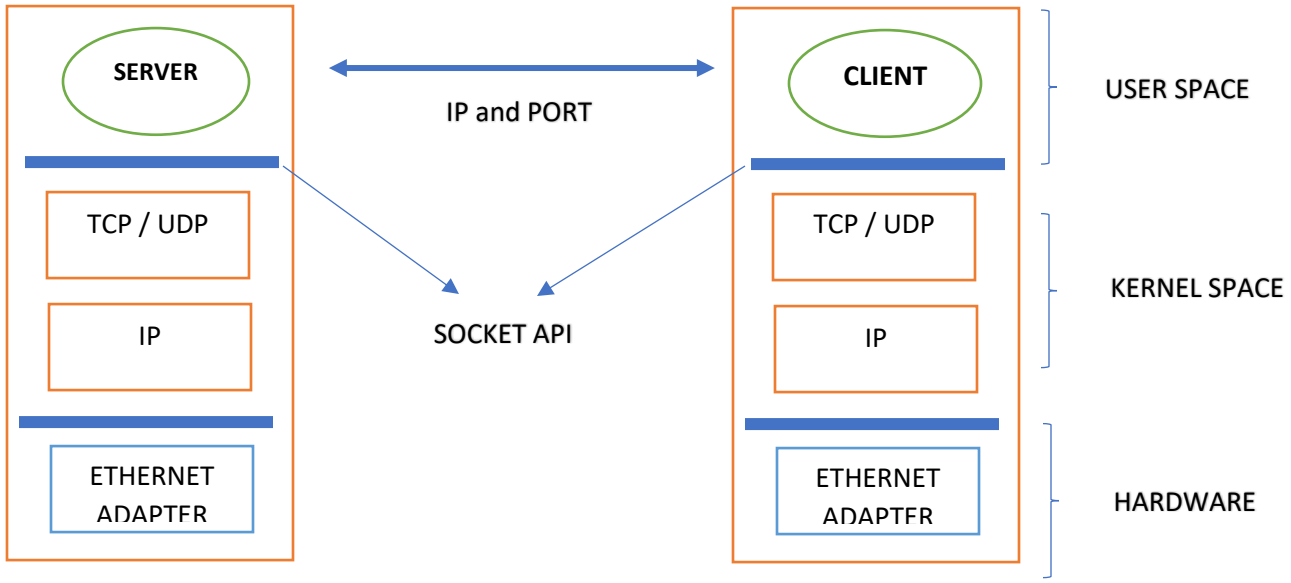
Sockets have a long history. Their use originated with ARPANET in 1971 and later became an API in the Berkeley Software Distribution (BSD) operating system released in 1983 called Berkeley sockets.

When the Internet took off in the 1990s with the World Wide Web, so did network programming. Web servers and browsers weren't the only applications taking advantage of newly connected networks and using sockets. Client-server applications of all types and sizes came into widespread use.

Today, although the underlying protocols used by the socket API have evolved over the years, and new ones have developed, the low-level API has remained the same.

The most common type of socket applications are client-server applications, where one side acts as the server and waits for connections from clients. This is the type of application that you'll be creating in this tutorial. More specifically, you'll focus on the socket API for Internet sockets, sometimes called Berkeley or BSD sockets. There are also Unix domain sockets, which can only be used to communicate between processes on the same host.

## Server and Client



## User Datagram Protocol (UDP):

**About UDP**

- **Signle Socket to receive messages**
- **Guarantee of delivery**
- **Not necessarily in-order delivery**
- **Datagram - independent packets**
- **Must address each packet**

**Example using UDP Applications**

- **Multimedia**
- **Voice over IP**

## Transmission Control Protocol (TCP):

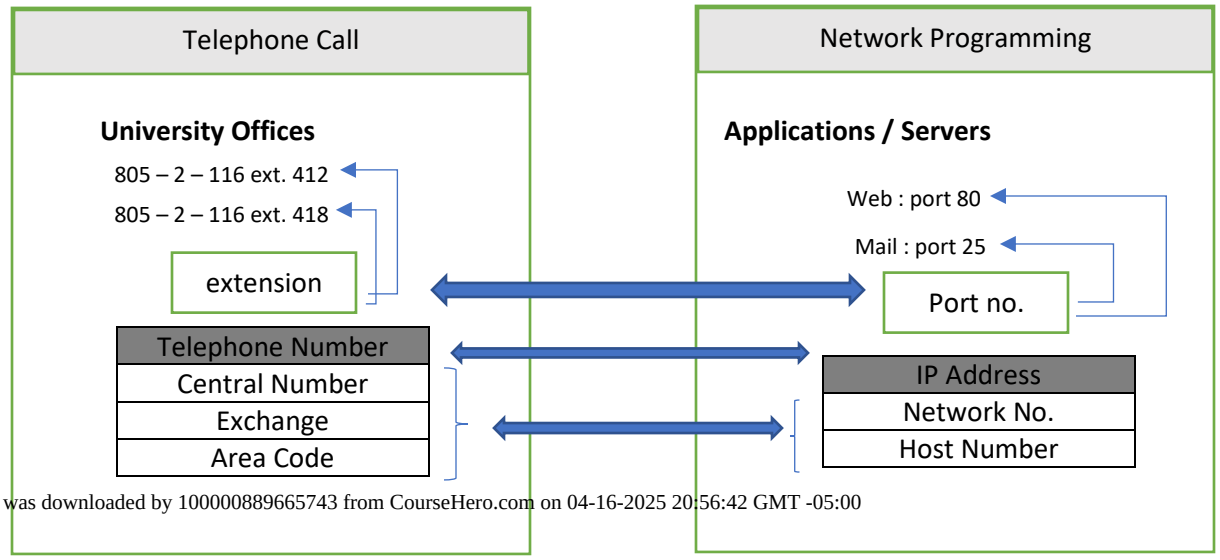
**About TCP**

- **Reliable – guarantee delivery**
- **Byte stream - in-order delivery**
- **Connection oriented – single socket per connection**
- **Setup connection followed by data transfer**

**Example using TCP Applications**

- **Web**
- **Email**
- **Telnet**
- **Others**

## Network Addressing Analogy



## Concept of Port Numbers

- Port numbers are used to indentify "entities" on a host

- Port numbers can be

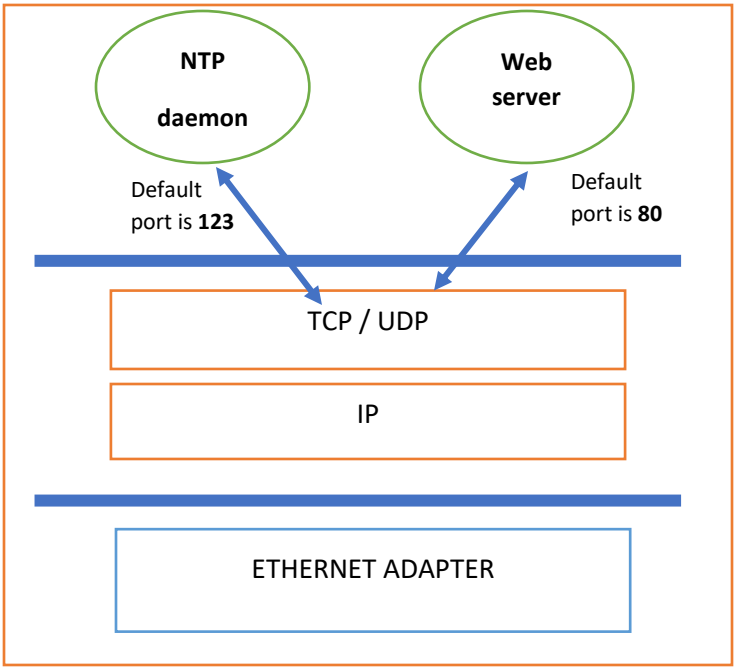
  - Well-known ( port 0-1023 )
  - Dynamic or Private ( port 1024 - 65535 )

- Servers / daemons usually use well-know ports

  - Any client can identify the server/service
  - HTTP is port 80
  - FTP is port 21
  - Telnet is port 23, and so on

- Clients usually use dynamic ports

  - Assigned by the kernel at run time



**Names and Addresses**

- Each attachment point on Internet is given unique address
    - Based on location within network - like phone numbers
- Humans prefer to deal with names not addresses
    - DNS provides mapping of name to address
    - Name based on administrative ownership of host

# What is a SOCKET?

***A socket is a file descriptor that lets an application read/write data from/to the network.***

## Socket API Overview

Python's socket module provides an interface to the Berkeley sockets API. This is the module that you'll use in this tutorial.

The primary socket API functions and methods in this module are:

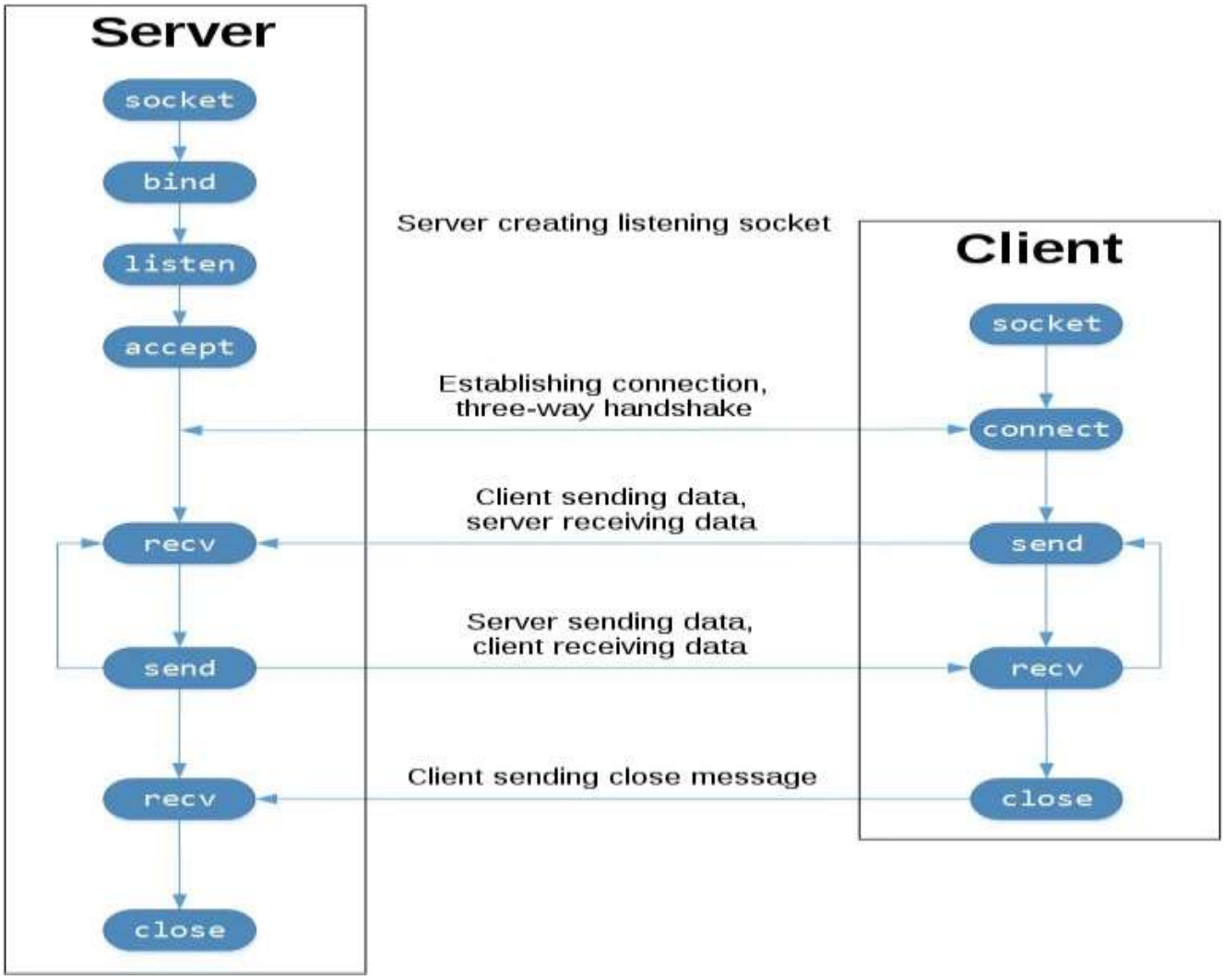
- `socket()`
- `.bind()`
- `.listen()`
- `.accept()`
- `.connect()`
- `.connect_ex()`
- `.send()`
- `.recv()`
- `.close()`

## TCP Sockets

You're going to create a socket object using `socket.socket()`, specifying the socket type as `socket.SOCK_STREAM`. When you do that, the default protocol that's used is the [Transmission Control Protocol (TCP)](). This is a good default and probably what you want.

Why should you use TCP? The Transmission Control Protocol (TCP):

- **Is reliable:** Packets dropped in the network are detected and retransmitted by the sender.
- **Has in-order data delivery:** Data is read by your application in the order it was written by the sender.




**TCP SOCKET FLOW DIAGRAM**

# Python Socket Module Server Code

```python
import socket


Host = "127.0.0.1"
Port = 65432

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 conn:
            data = conn.recv(1024)
            if not data:
                break
            print(f'the data is {data}')
            conn.send(b"welcome to my server")
```

## Server waiting a connection from client

```
(virt) C:\pythonCLASS\test>python mysocket.py
```

## Server data packets from client

```
(virt) C:\pythonCLASS\test>python mysocket.py
connected by ('127.0.0.1', 51927)
the data is b'Hello, world'
```

## Python Socket Module Client Code

```python
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}")
```

**Client establish connection to a server and received data packets from server.**

Command Prompt

```
(virt) C:\pythonCLASS\test>python mysocketclient.py
Received b'welcome to my server'

(virt) C:\pythonCLASS\test>
```

# https://realpython.com/python-sockets/