# 4.Python网络编程Socket

一. 概念:

1. 套接字:

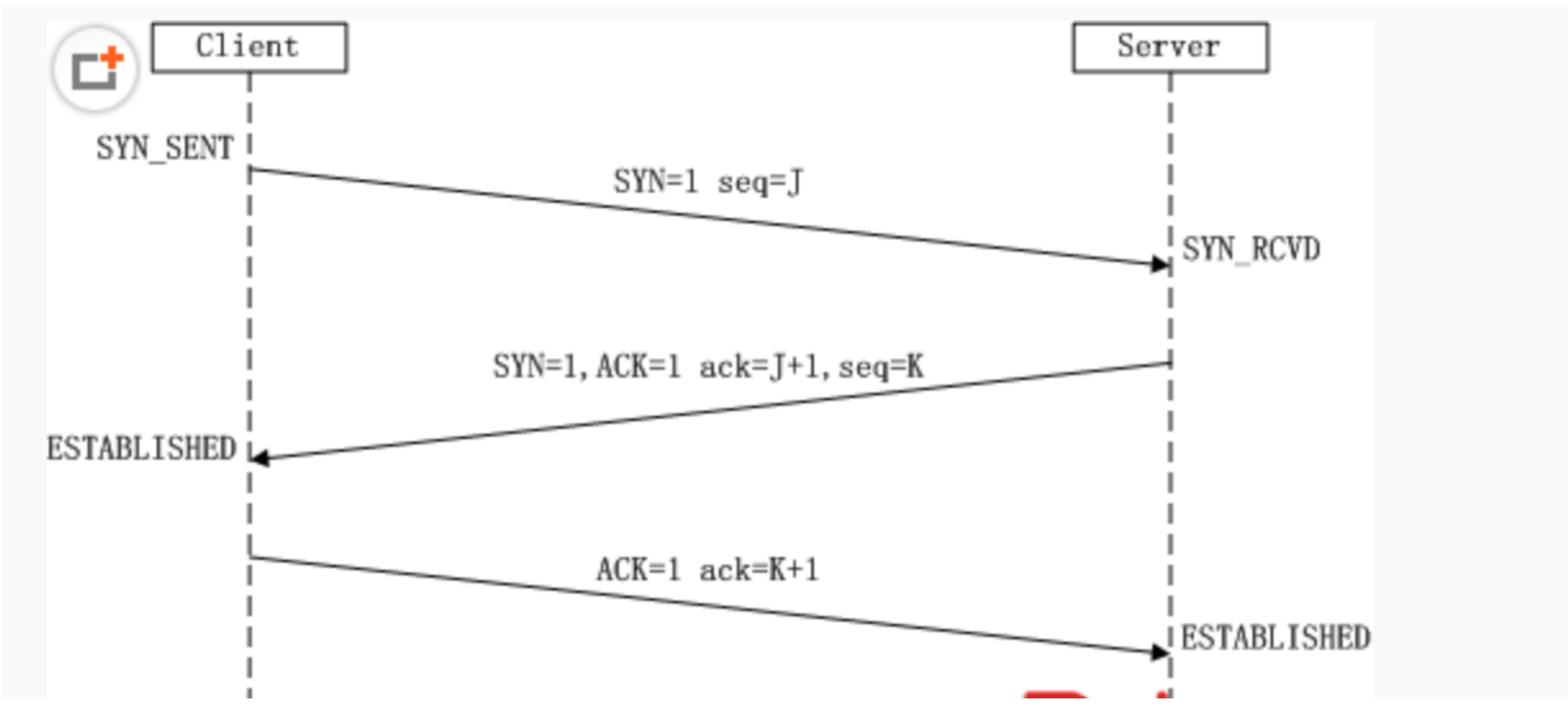
   套接字是一种网络数据结构, 网络化的应用程序在开始任何通信之前必须创建套接字, 就好像电话固定电话的插口一样, 没有它就不能通信; 最初套接字用在同一台主机上多个应用程序之间的通讯, 也就是进程之间的通信, 套接字有两种一种是基于文件型的一种是基于网络型的.
2. 分类:

   1) AF_UNIX: (AF是地址家族的意思); 本地通信实例用于进程之间的通信, 两个进程都运行在同一个机器上, 这些套接字是基于文件的; 所以它的底层结构是由文件系统来支持的, 因为在同一个电脑上, 文件系统的确是不同进程都可以访问的.

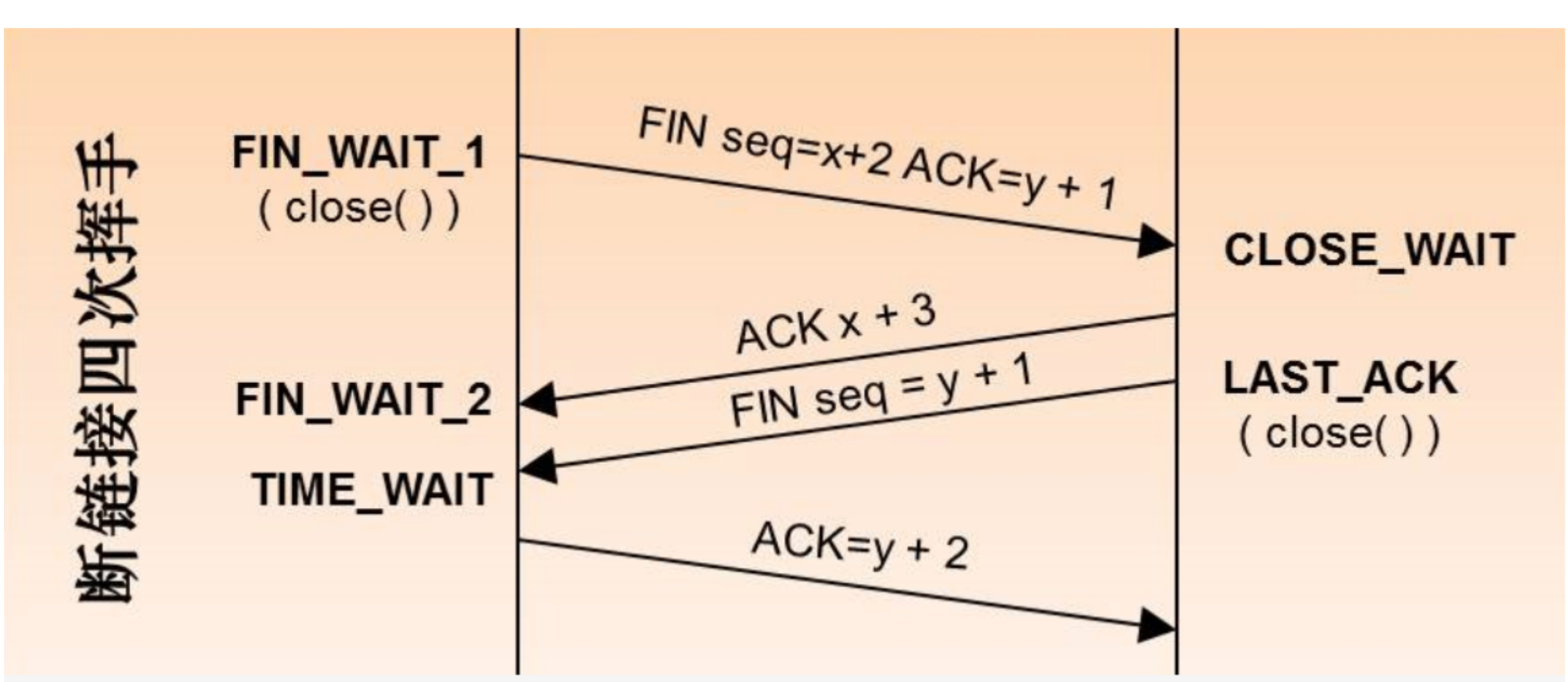
   2) AF_INET: 地址家族internet, 主要用于网络编程, 还有AF_INET6 用在IPv6的环境中.
3. 面向连接与无连接

   1) TCP的三次握手

   特点: 在通信之前一定要建立一条虚链路来保证每一份数据都可以顺序的, 可靠的传输到目的地; 创建tcp链接的套接字类型为SOCK_STREAM, 这个套接字使用IP来查找网络中的主机, 所以我们把他们组成的这个数据传输系统称之为TCP/IP生活中的实例如打电话.




2） TCP的四次断开




TCP四次断开过程：

当客户端和服务器通过三次握手建立了TCP连接以后， 当数据传送完毕， 需要四次断开， 关闭TCP连接.

1. 第一次分手：主机1(可以使客户端，也可以是服务器端)，设置Sequence Number和Acknowledgment Number，向主机2发送一个FIN报文段；此时，主机1进入FIN_WAIT_1状态，这表示主机1没有数据要发送给主机2了.
2. 第二次分手：主机2收到了主机1发送的FIN报文段，向主机1回一个ACK报文段，Acknowledgment Number为Sequence Number加1；主机1进入FIN_WAIT_2状态，主机2告诉主机1，我“同意”你的关闭请求.
3. 第三次分手：主机2向主机1发送FIN报文段，请求关闭连接，同时主机2进入LAST_ACK状态.
4. 第四次分手：主机1收到主机2发送的FIN报文段，向主机2发送ACK报文段，然后主机1进入TIME_WAIT状态.主机2收到主机1的ACK报文段以后，就关闭连接.

4. 无连接 UDP

   特点: 不需要建立连接就可以通信, 这样就使得数据到达的顺序与可靠性无法得到保证; 因为不需要建立虚链路, 所以数据传输速度相对于TCP来说比较快 , 创建UDP的套接字为SOCK_DGRAM, 生活中的实例如写信.

二. Python网络编程模块socket

In [5]:
```
#1.创建一个TCP socket
import socket
tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

In [7]:
```
#2.创建一个UDP socket
import socket
udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

TCP Socket常用方法:

1. s.bind() #绑定IP和端口号到套接字
2. s.listen() #开始TCP监听
3. s.accept() #被动接受TCP客户端的连接
4. s.connect() #主动初始化TCP的链接
5. s.recv() #接受TCP数据
6. s.send() #发送TCP数据
7. s.close() #关闭套接字

In [ ]:
```
#示例代码1.创建tcp server

import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.bind(('127.0.0.1', 5555))
s.listen(2)

while 1:
    cli, addr = s.accept()
    print(cli, addr)
    while 1:
        data =  cli.recv(1024)
        if data:
            print(data)
            cli.send('success')
            break
```

In [9]:
```
#示例代码2.创建tcp client
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 5555))
s.send('123')
print(s.recv(1024))
```

UDP Socket常用方法:

1. s.bind() #绑定IP和端口号到套接字
2. s.recvfrom() #接受UDP数据
3. s.sendto() #发送UDP数据
4. s.close() ##关闭套接字

In [ ]:
```
#示例代码3.创建UDP server
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('172.16.70.130', 5555))
while 1:
    data, addr =  s.recvfrom(1024)
    s.sendto('success', addr)
```

In [ ]:
```
#示例代码4.创建UDP client
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto('123', ('172.16.70.130', 5555))
print(s.recvfrom(1024))
```

In [ ]:
```

```