TCP编程

IP地址

ip地址是4组数字，每组数字最多三位，
中间用点隔开，每组数字的范围是0-255

192.168.0.1
172.16.22.19
8.8.8.8
123.22.122.1

私有IP范围 👇

10.0.0.0 – 10.255.255.255.
172.16.0.0 – 172.31.255.255.
192.168.0.0 – 192.168.255.255

端口号

每个程序都用了唯一的端口，端口是一个数字，从1-65535。

1024以下的端口号也叫作知名端口号，就是那些由

互联网名称与数字地址分配机构（ICANN）预留给传输控制协议（TCP），这些端口都代表了具体的约定的用法

，不应该滥用，而剩下的端口号叫动态端口号或私有端口号，比较自由，可以按自己的业务需要约定使用

IPv6

是网际协议（IP）的最新版本，用作互联网的网络层协议，用它来取代IPv4主要是为了解决IPv4地址枯竭等问题，IPv6的设计目的是取代IPv4

IPv6二进位制下为128位长度，以16位为一组，每组以冒号(:)隔开，可以分为8组，每组以4位十六进制方式表示

2001:0db8:85a3:08d3:1319:8a2e:0370:7344

DNS(domain name system)

由于ip地址是一堆数字，不方便记忆，所以常用域名来代替ip地址，用户只需要输入域名就能访问。保存域名和ip地址对应关系的网络服务就是DNS

路由

一旦应用程序请求操作系统向某一特定ip地址发送数据，操作系统就需要决定如何使用该机器连接的某一个物理网络来传输数据。这个决定，也就是根据目的ip地址选择将ip数据包发往何处，就是路由

套接字(Socket)

socket是一种操作系统提供的进程间通信机制

它是网络通信过程中端点的抽象表示，包含进行网络通信必需的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，对方主机的IP地址，对方进程的协议端口。

套接字类型

1. 流式套接字（SOCK_STREAM）
2. 数据报套接字（SOCK_DGRAM）
3. 原始套接字（SOCK_RAW）

套接字地址家族(Address Family)

1. AF_INET
2. AF_INET6
3. AF_UNIX

用socket模块

```
# 一般语法👇
socket(socket_family, socket_type, protocol=0)

# 其中，socket_family 是 AF_UNIX 、 AF_INET 、 AF_INET6
# socket_type 是 SOCK_STREAM 或 SOCK_DGRAM
# protocol 通常省略，默认为 0

# 创建 TCP 套接字：

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

服务端例子

```
import socket

HOST = '127.0.0.1'
PORT = 8001

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # 创建套接字
s.bind((HOST, PORT))  # 绑定套接字到本地IP与端口
s.listen(5)  # 监听连接

print(f'Server start at: {HOST}:{PORT}')

while 1:  # 服务器可以无限循环的接收内容
    conn, addr = s.accept()  # 接受客户端连接
    print(f'Connected by {addr}')

    while 1:  # 通信循环，因为一次发送的内容可能很大
        data = conn.recv(1024)  # 接收1024字节的内容
        print(data)

        # 给客户端也发送数据，send方法接收byte类型的数据，
        # 需要把字符串手动转换一下
        conn.send(bytes(f'Server received {data}', 'utf-8'))
    conn.close()  # 关闭客户端套接字
s.close() # 关闭服务器套接字
```

客户端例子

```
import socket

HOST = '127.0.0.1'
PORT = 8001

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))  # 尝试连接到服务器

while 1:
    cmd = raw_input('Please input msg:')
    s.send(bytes(cmd, 'utf-8'))
    data = s.recv(1024)
    print(data)
```

1. 启动服务端

```
❯ python tcp_server.py
Server start at: 127.0.0.1:8001
Connected by ('127.0.0.1', 64292) # 可以看到之后客户端启动后的请求的端口用的是64292
b'test'  # 这2行都是客户端发送来的内容
b'exit'
```

2. 启动客户端

```
❯ python tcp_client.py
Please input msg:test  # 输入 test 然后回车
Server received b'test'  # 这句是服务端返回的
Please input msg:exit  # 输入 exit 然后回车
Server received b'exit'  # 同样是服务端返回的
```

延伸阅读

1. 《CCNA学习指南中文版》 第2 - 3章
2. http://man7.org/linux/man-pages/man2/socket.2.html