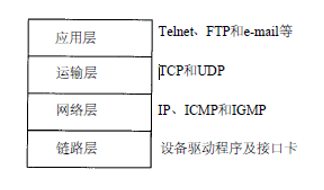


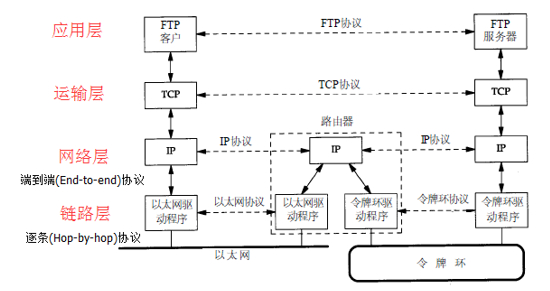
**TCP/IP四层网络模型：**

1. **链路层（数据链路层/网络接口层）：包括操作系统中的设备驱动程序、计算机中对应的网络接口卡**
2. **网络层（互联网层）：处理分组在网络中的活动，比如分组的选路**
3. **传输层：主要为两台主机上的应用提供端到端的通信**
4. **应用层：负责处理特定的应用程序细节**



**实例：**

**假设在一个局域网（LAN）如以太网中有两台主机，二者运行FTP协议：**



**IP地址**

**（1、IP地址被用来给Internet上的电脑一个编号。大家日常见到的情况是每台连网的PC上都需要有IP地址，才能正常通信。我们可以把‘个人电脑’比作‘一台电话’，那么‘IP地址’就相当于‘电话号码’，而Internet中的路由器，就相当于电信局的‘程控式交换机’。**

**（2、IP地址是一个32位的二进制数，通常被分割为4个‘8位二进制数’（也就是4个字节）。IP地址通常用‘点分十进制’表示成（a,b,c,d）的形式，其中，a,b,c,d都是0~255之间的十进制整数，比如：192.168.0.1**

**（3、windows系统在命令提示符下输入‘ipconfig’命令查看本机IP地址。**

**TCP协议：**

**（1、TCP协议则是建立在IP协议之上的。TCP协议负责在两台计算机之间建立可靠连接，保证数据包按顺序到达。TCP协议会通过握手建立连接，然后，对每个IP包编号，确保对方按顺序收到，如果包丢了，就自动重发。**

**（2、许多常用的更高级的协议都是建立在TCP协议基础上的，比如用于浏览器的HTTP协议、发送邮件的SMTP协议等。**

**（3、一个TCP报文除了包含要传输的数据外，还包含源IP地址和目标IP地址，源端口号和目标端口号。**

**端口：**

1. **端口有什么作用？在两台计算机通信时，只发IP地址是不够的，因为同一台计算机上跑着多个网络程序。一个TCP报文来了之后，到底是交给浏览器还是QQ,就需要端口号来区分。每个网络程序都向操作系统申请唯一的端口号，这样，两个进程在两台计算机之间建立网络连接就需要各自的IP地址和各自的端口号。**
2. **一个进程也可能同时与多个计算机建立链接，因此它会申请很多端口。**
3. **Windows系统下查看所有端口的命令是’netstat-ano’**
4. **🡪’netstat-an’(乌班图)**

**Udp编程-socket简介**

**（1、socket（简称 套接字）是进程间通信的一种方式，它与其他进程间通信的一个主要不同是：**

**它能实现不同主机间的进程间通信，我们网络上各种各样的服务大多都是基于Socket来完成通信的**

**（2、例如我们每天浏览网页、QQ聊天，收发email等等**

**创建socket**

**在python中使用socket模块的函数socket就可以完成：**

**import socket**

**socket.socket(AddressFamily,Type)**

**函数socket.socket创建一个socket，该函数带有两个参数：**

1. **Address Family：可以选择AF\_INET(用于Internet 进程间通信)或者AF\_UNIX（用于同一台机器进程间通信），实际工作中常用AF\_INET**
2. **Type:套接字类型**

**（1、可以是SOCK\_STREAM(流式套接字，主要用于TCP协议)**

**（2、或者SOCK\_DGRAM(数据报套接字，主要用于UDP协议)**

**创建一个tcp socket(tcp套接字)**

**Import socket**

**#创建tcp的套接字**

**s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)**

**#...这里是使用套接字的功能（省略）。。。**

**#不用的时候，关闭套接字**

**s.close()**

**创建一个udp socket(udp套接字)**

**Import socket**

**#创建udp的套接字01**

**s=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)**

**#..这里是使用套接字的功能（省略）。。。**

**#不用的时候，关闭套接字02**

**s.close()**

**套接字使用流程：**

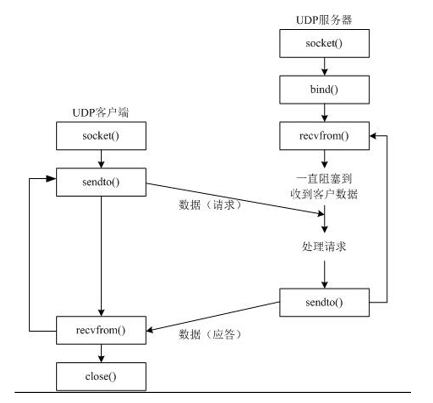
**套接字使用流程 与 文件的使用流程很类似**

1. **创建套接字**
2. **使用套接字收/发数据**
3. **关闭套接字**

**Udp网络程序-发送数据**

**创建一个基于udp的网络程序流程很简单，具体步骤如下：**

1. **创建客户端套接字**
2. **发送/接收数据**
3. **关闭套接字**



**示例代码**

**from socket import\***

**#1、创建udp套接字**

**udp\_socket=socket(AF\_INET,SOCK\_DGRAM)**

**#2、准备接收方的地址**

**#’192.168.43.85‘表示目的ip地址，这里我们给本机发数据，所以用本机的ip地址**

**#8080表示目的端口**

**dest\_addr=(‘192.168.43.85’,8080) #注意 是元祖，ip是字符串，端口是数字**

**#3、从键盘获取数据**

**send\_data=input(“请输入要发送的数据：”)**

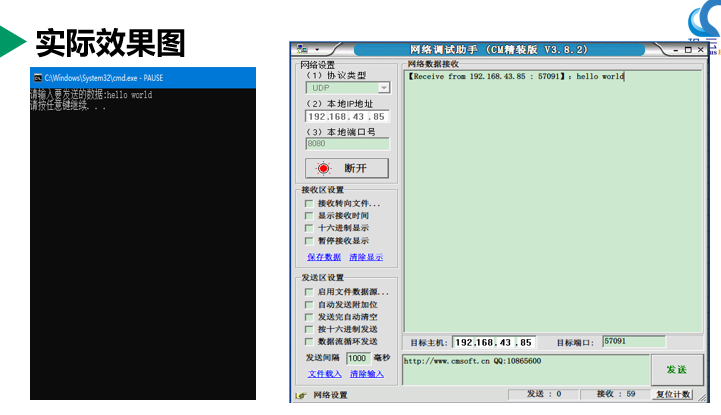
**#4、发送数据到指定的电脑上的指定程序中**

**udp\_socket.sendto(send\_data.encode(‘gb2312’),dest\_addr)**

**#5、关闭套接字**

**udp\_socket.close()**

**encode和decode后面的参数是文字编码，比如还有utf-8**



**Udp网络程序-发送、接收数据**

**from socket import\***

**#1.创建udp套接字**

**udp\_socket=socket(AF\_INET,SOCK\_DGRAM)**

**#2.准备接收方的地址**

**dest\_addr=(‘192.168.43.85’,8080)**

**#3.从键盘获取数据**

**send\_data=input(‘请输入要发送的数据:’)**

**#4.发送数据到指定的电脑上**

**udp\_socket.sendto(send\_data.encode(‘gb2312’),dest\_addr)**

**#5.等待接收对方发送的数据**

**recv\_data=udp\_socket.recvfrom(1024) #1024表示本次接收的最大字节数**

**#6.显示对方发送的数据**

**#接收到的数据recv\_data是一个元祖**

**#第1个元素是对方发送的数据 #第2个元素是对方的ip和端口**

**print(recv\_data[0].decode(‘gb2312’))**

**print(recv\_data[1])**

**#7.关闭套接字**

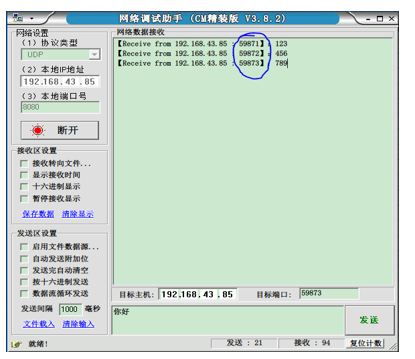
**udp\_socket.close()**

**udp网络程序-会变的端口号**

**重新运行多次之前的程序，然后再‘网络调试助手中’，每一个python进程所使用的端口号都不一样**

**为什么端口号会变？**

1. **每重新运行一次网络程序，上图圈中的数字，不一样的原因在于，这个数字标识这个网络程序，当重新运行时，如果没有确定到底用哪个，系统默认会随机分配**
2. **记住一点：这个网络程序在运行的过程中，这个就唯一标识这个程序，所以如果其他电脑上的网络程序如果想要向此程序发送数据，那么就需要向这个数字（即端口）标识的程序发送即可**



**服务器端需要绑定端口号**

1. **一般情况下，在一台电脑上运行的网络程序有很多，为了不与其他的网络程序占用同一个端口号，往往在编程中，upd的端口号一般不绑定**
2. **但是如果需要做成一个服务器端的程序的话，是需要绑定的，想想看这又是为什么呢？**
3. **如果报警电话每天都在变，想必世界就会乱了，所以一般服务性的程序，往往需要一个固定的端口号，这就是所谓的端口绑定**

**绑定实例**

**from socket import\***

**#1.创建套接字**

**udp\_socket=socket(AF\_INET，SOCK\_DGRAM)**

**#2.绑定本地的相关信息，如果一个网络程序不绑定，则系统会随机分配**

**local\_addr=(‘’,7788) # ip地址和端口号，ip一般不用写，表示本机的任何一个ip**

**udp\_socket.bind(local\_addr)**

**#3.等待接受对方发送的数据**

**recv\_data=udp\_socket.recvfrom(1024) #1024表示本次接收的最大字节数**

**#4.显示接收到的数据**

**print(recv\_data[0].decode(‘gb2312’))**

**#5 .关闭套接字**

**udp\_socket.close()**



**TCP简介**

**（1、TCP协议，传输控制协议是一种面向连接的、可靠的、基于字节流的传输层通信协议，有IETF的RFC 793定义。**

**（2、TCP通信需要经过创建连接、数据传送、终止连接三个步骤。**

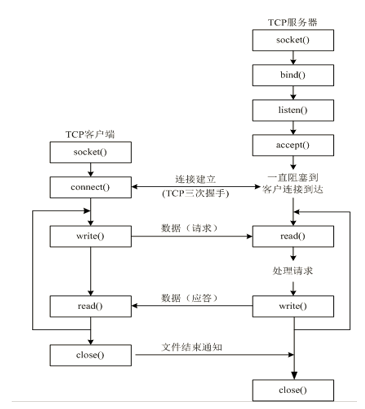
**（3、TCP通信模型中，在通信开始之前，一定要先建立相关的连接，才能发送数据，类似于生活中，‘打电话’**

**（4、这种连接是一对一的，因此TCP不适用于广播的应用程序，基于广播的应用程序请使用UDP协议。**

**TCP和UDP的区别**

1. **TCP面向连接（如打电话要先拨号建立连接）；UDP是无连接的，即发送数据之前不需要建立连接**
2. **TCP提供可靠的服务，也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达；UDP尽最大努力交付，即不保证可靠交付**
3. **UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信**
4. **每一条TCP连接只能是点到点的；UDP支持一对一，一对多，多对一和多对多的交互通信**
5. **TCP对系统资源要求较多，UDP对系统资源要求较少**

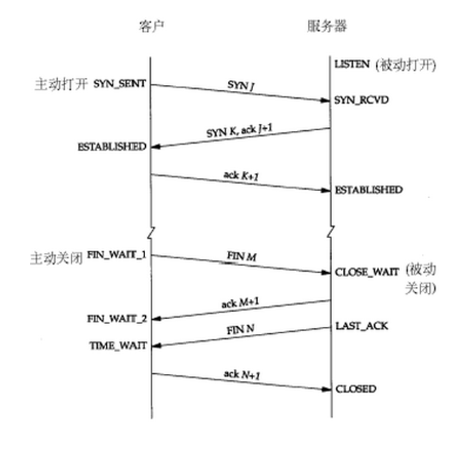
**TCP通信模型**



**Tcp通信之三次握手和四次挥手**

**TCP的三次握手（建立连接）和四次挥手（关闭连接）**

1. **TCP传输控制协议**
2. **三次握手**
3. **TCP是主机对主机层的传输控制协议，提供可靠的连接服务，采用三次握手确认建立一个连接；**
4. **位码即TCP标志位**
5. **有六种标示：**
6. **SYN(synchronous建立联机)**
7. **ACK（acknowledgement确认）**
8. **PSH(push传送)**
9. **FIN（finish结束）**
10. **RST(reset重置)**
11. **URG（urgent紧急）**



**Tcp通信之三次握手和四次挥手**

**各个状态的意义如下：**

* + **LISTEN - 侦听来自远方TCP端口的连接请求；**
  + **SYN-SENT -在发送连接请求后等待匹配的连接请求；**
  + **SYN-RECEIVED - 在收到和发送一个连接请求后等待对连接请求的确认；**
  + **ESTABLISHED- 代表一个打开的连接，数据可以传送给用户；**
  + **FIN-WAIT-1 - 等待远程TCP的连接中断请求，或先前的连接中断请求的确认；**
  + **CLOSE-WAIT - 等待从本地用户发来的连接中断请求；**
  + **FIN-WAIT-2 - 从远程TCP等待连接中断请求；**
  + **CLOSING -等待远程TCP对连接中断的确认；**
  + **LAST-ACK - 等待原来发向远程TCP的连接中断请求的确认；**
  + **TIME-WAIT -等待足够的时间以确保远程TCP接收到连接中断请求的确认；**
  + **CLOSED - 没有任何连接状态；**