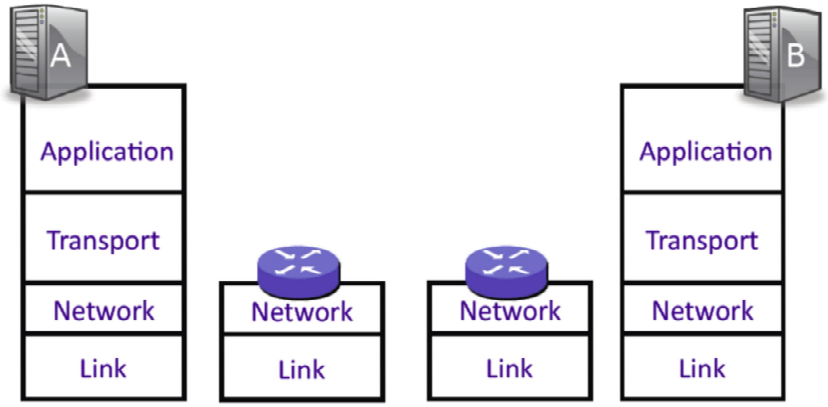
应用层：UDP

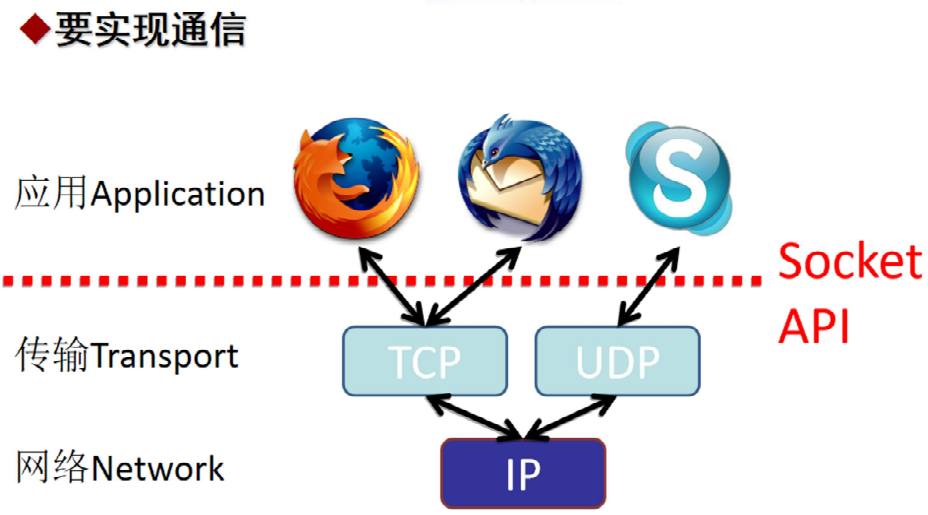
2021年10月17日 17:08

参考：[py网络编程之socket]
[py网络编程---socket编程]
[py网络编程基础]



1.有哪五层：

应用层，运输层，网络层，链路层，物理层



2.两个协议：TCP,UDP，特点和区别是什么?

数据是一段一段的发送。

IP协议负责把数据按块分割并以块为单位从一台计算机通过网络发送到另一台计算机。IP包的特点是按块发送，途径多个路由，但不保证能到达，也不保证顺序到达。

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的三次握手和四次拜拜就不再这里详细描述。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。

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，源端口和目标端口。
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
。
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。
TCP是建立可靠连接，并且通信双方都以流的形式发送数据。相对TCP，UDP则是面向无连接的协议。
使用UDP协议时，不需要建立连接，只需要知道对方的IP地址和端口号，就可以直接发数据包。但是不保证是否能到达。
虽然用UDP传输数据不可靠，但它的优点是和TCP比，速度快，对于不要求可靠到达的数据，就可以使用UDP协议。

先来看几个程序，思考以下几个问题：
(1)每一句话是什么含义?

什么是面向连接?
面向连接，即在通信之前建立一条通信线路。
无连接，系统自主选的一条路线传输。

服务器
1.socket:套接字，即把socket属性引用到命名空间，可以大大缩短代码，具体参考博客。
2.地址host不确定，21567指的是端口port(区分同一电脑不同程序)， '' 代表host,意思是获取客户机地址。为什么前面加了''，而后面'localhost'?客户端地址不确定

用UDP协议。

先来看几个程序，思考以下几个问题：
(1)每一句话是什么含义？
(2)层间(传输层与应用层)提供服务如何？位置+形式？
(3)层间接口携带哪些信息？传的什么，谁传的，传给谁
(4)谁根据哪些信息进行什么样的封装？

● 服务器端

```
from socket import *
ADDR = ('' , 21567)
udpSerSock = socket(AF_INET,SOCK_DGRAM)
udpSerSock.bind(ADDR)
data, addr = udpSerSock.recvfrom(1024)
print data
udpSerSock.close()
```

● 客户机端

```
from socket import *
ADDR = ('localhost', 21567)
udpCliSock = socket(AF_INET, SOCK_DGRAM)
data = raw_input('> ')
udpCliSock.sendto(data, ADDR)
udpCliSock.close()
```

为什么要求发送到的服务器地址固定，而客户机地址是随机的？
服务器上面是各种应用的端口，且UDP只需要知道对方的地址和端口，速度快但不能保证准度。

● 服务器端

```
from socket import *
ADDR = ('' , 21567)
udpSerSock = socket(AF_INET,SOCK_DGRAM)
udpSerSock.bind(ADDR)
while True:
    data, addrc = udpSerSock.recvfrom(1024)
    udpSerSock.sendto(data+'s',addrc)
udpSerSock.close()
```

● 客户机端

```
from socket import *
ADDR = ('localhost', 21567)
udpCliSock = socket(AF_INET, SOCK_DGRAM)
while True:
   data = raw_input('> ')
   udpCliSock.sendto(data, ADDR)
   data, ADDR = udpCliSock.recvfrom(1024)
   print data
udpCliSock.close()
```

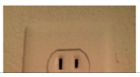
同一电脑不同程序)， '' 代表host,意思是获取客户机地址。为什么前面加了''，而后面'localhost'?客户端地址不确定
3.这句话意思是创建udp套接字udpSerSock，调用socket函数，第一个参数指socket_family，包括AF_INET网络，此外还有AF_UNIX本机。第二个参数指socket_type,包括SOCK_STREAM,SOCK_DGRAM（前者指要连接，后者指无连接）。第三个参数是protocol，(指定协议，如TCP,UDP)，默认值为0（此时第二个参数确定了第三个参数是什么）
4.主要看bind函数，作用是套接字与地址绑定
5.recvfrom值接收UDP函数，与TCP中recv函数类似，返回值是数据，地址。1024指bufsize,即要接受的最大数据量，单位bytes(又叫MSS，即Maxitum Segment Size,最大分段大小，因为数据是一段段传输的，另外还有一个是MTU，即Maxitum Transmission Unit,最大传输单元，其他详见MSS是如何确定的)此外还有一个参数flag,提供有关信息的其他信息，通常可以忽略
6.打印数据
7.关闭服务器套接字

客户机
1.同服务器
2.命名地址，前面代表服务器地址host(这里的localhost也可以写成127.0.0.1，0.0.0.0代表所有的网络地址)，(IP地址，4个8位，区分不同电脑)，后面代表端口
3.创建客户机套接字，网络+无需建立连接
4.意思是在屏幕上打出一个>，光标在后面闪动
5.sendto发送UDP数据，将数据发送到套接字，address形式是(ipaddr,port)的元组，指定远程地址，返回值是发送的字节数，该数量可能小于string的字节大小。
6.关闭客户机套接字

服务器
1,2,3,4同前面
5.一个while循环
6.同前面，1024指接收的数据最大值，单位是bytes，addrc指发送数据的套接字的地址(address+client)
7.sendto发送数据，在原有接收到的data后面加一个字符s，第二个参数指客户机的地址
8.关闭文件

客户机
全部同前面

```
from socket import *
udpCliSock = socket(AF_INET, SOCK_DGRAM)
ADDR = ('localhost', 21567)
while True:
    data = raw_input('> ')
    udpCliSock.sendto(data, ADDR)
udpCliSock.close()
```



```
from socket import *
ADDR = ('' , 21567)
udpSerSock = socket(AF_INET, SOCK_DGRAM)
udpSerSock.bind(ADDR)
while True:
    data, ADDR = udpSerSock.recvfrom(1024)
    print data
udpSerSock.close()
```

上面程序共同的部分（框架）：

from socket import*:

先看关键字socket,翻译过来是“套接字”，有什么作用?

socket的引入是问题2

分布式进程通信需要解决的问题

- 问题1：进程标示和寻址问题（服务用户）
- 问题2：传输层-应用层提供服务是如何（服务）
  - 位置：层间界面的SAP （TCP/IP：socket）
  - 形式：应用程序接口API （TCP/IP：socket API）
- 问题3：如何使用传输层提供的服务，实现应用进程之间的报文交换，实现应用（用户使用服务）
  - 定义应用层协议：报文格式，解释，时序等
  - 编制程序，使用OS提供的API，调用网络基础设施提供通信服务传报文，实现应用时序等；

首先,层间接口要携带如下信息。传的什么，谁传的，传给谁

- 层间接口必须要携带的信息
  - 要传输的报文（对于本层来说：SDU）
  - 谁传的：对方的应用进程的标示：IP+TCP(UDP) 端口
  - 传给谁：对方的应用进程的标示：对方的IP+TCP(UDP)端口号

谁根据哪些信息进行什么样的封装?

- 传输层实体（tcp或者udp实体）根据这些信息进行TCP报文段（UDP数据报）的封装
  - 源端口号，目标端口号，数据等
  - 将IP地址往下交IP实体，用于封装IP数据报：源IP,目标IP

问题2：传输层提供的服务-层间信息的代表

- 如果Socket API 每次传输报文，都携带如此多的信息，太繁琐易错，不便于管理
- 用个代号标示通信的双方或者单方：socket
- 就像OS打开文件返回的句柄一样
  - 对句柄的操作，就是对文件的操作
- TCP socket：
  - TCP服务，两个进程之间的通信需要之前要建立连接
    - 两个进程通信会持续一段时间，通信关系稳定
  - 可以用一个整数表示两个应用实体之间的通信关系，本地标示
  - 穿过层间接口的信息量最小
  - TCP socket：源IP,源端口，目标IP，目标IP,目标端口

- UDP socket：
  - UDP服务，两个进程之间的通信需要之前无需建立连接
    - 每个报文都是独立传输的
    - 前后报文可能给不同的分布式进程
  - 因此，只能用一个整数表示本应用实体的标示
    - 因为这个报文可能传给另外一个分布式进程 ·1
  - 穿过层间接口的信息大小最小
  - UDP socket：本IP,本端口
  - 但是传输 报文时：必须要提供对方IP，port
    - 接收报文时： 传输层需要上传对方的IP，port