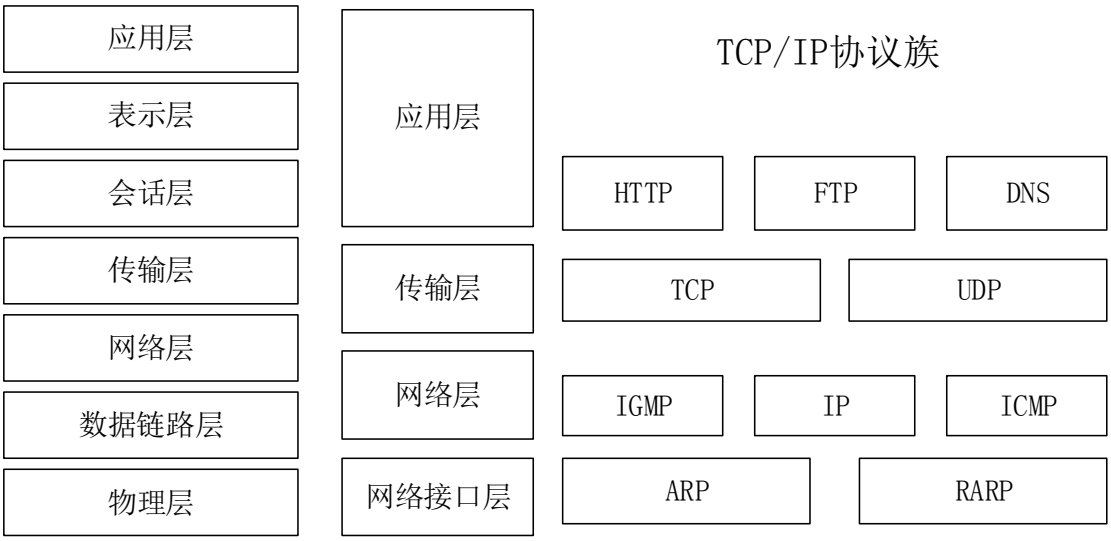
Socket 通信

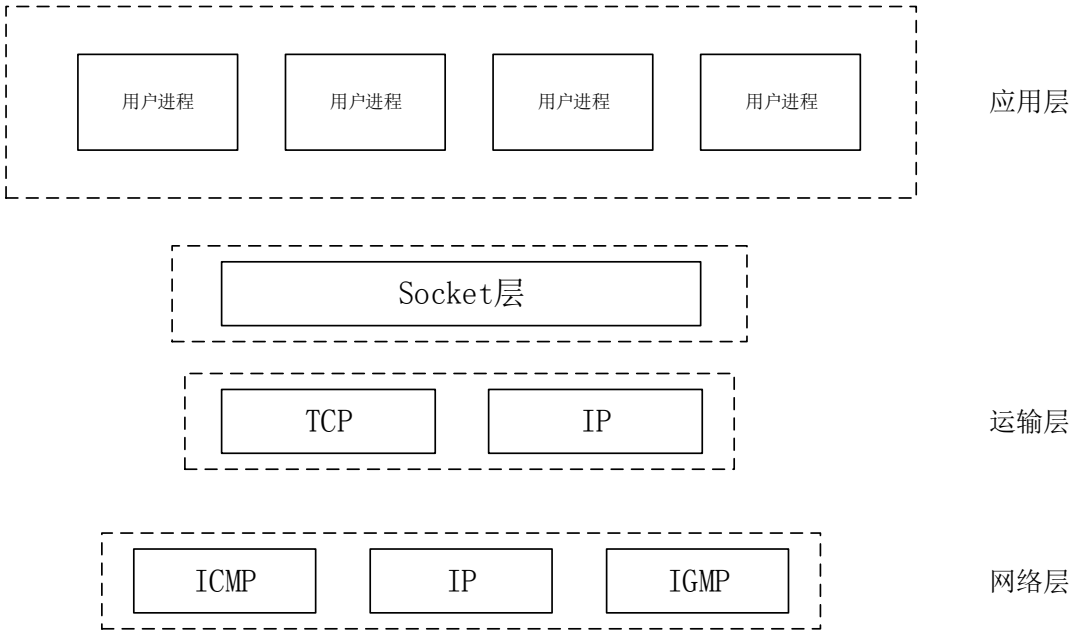
1. Socket

两个进程需要通信的一个基本前提是能够唯一标示一个进程，本地的进程通讯中采用 PID 来唯一标示一个进程。但是网络中两个进程的 PID 发生冲突的概率很大，而现有已知 IP 地址可以唯一标示网络上的主机，TCP 协议和端口号可以唯一标示主机的一个进程，因此可以利用 IP 地址、协议、端口号来唯一标示网络中的一个进程。

能够唯一标示网络中的进程后，两个网络进程之间就可以通过 socket 进行通信。Socket 直译为套接字，它是操作系统提供的在应用层和传输层之间的一个抽象层，它将 TCP/IP 层复杂的操作抽象为几个简单的接口供应用层调用，从而实现进程在网络中的通信。Socket 起源于 unix，本质上是文件，它实现了“打开-读/写-关闭”的模式，服务器和客户端各自维护一个文件进行通信，通讯结束时关闭文件。以下为 ISO 七层模型及 TCP/IP 协议族五层模型和 socket 所处的位置图。



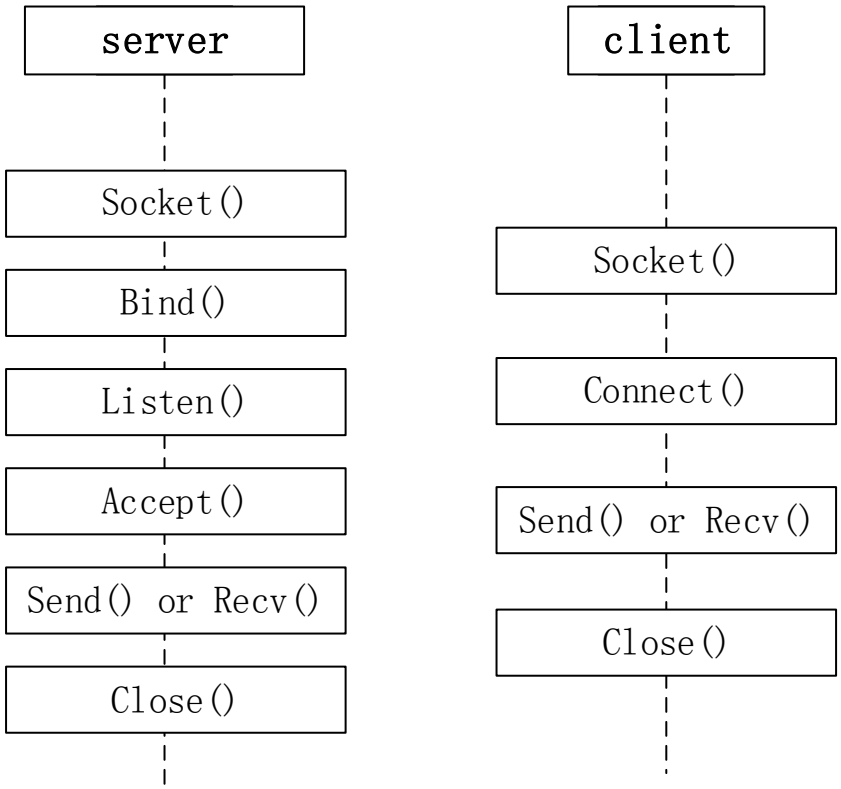
经典七层模型和 TCP/IP 协议族划分对比



socket 抽象层的位置

2. Socket 通信

以 TCP 协议通讯的 socket 为例，其交互流程如下图所示



socket 通信时序图

(1) 服务器根据地址类型、socket 类型、协议创建 socket

(2) 服务器为 socket 绑定 IP 地址和端口号

(3) 服务器 socket 监听端口号请求，随时准备接受客户端发来的连接请求，此时服务器端的 socket 并没有被打开

(4) 客户端创建 socket

(5) 客户端打开 socket，根据服务器 IP 地址和端口号尝试连接服务器 socket

(6) 服务器 socket 接收到客户端的 socket 请求，被动打开，接收客户端请求，直到客户端返回连接信息，这段时间 socket 处于阻塞状态，即 accept 方法一直等到客户端返回连接信息后才会返回一个对象，并开始接收下一个客户端的连接请求。

(7) 客户端连接成功，向服务器发送连接状态信息

(8) 服务器 accept 方法返回，此时双方已建立可靠连接

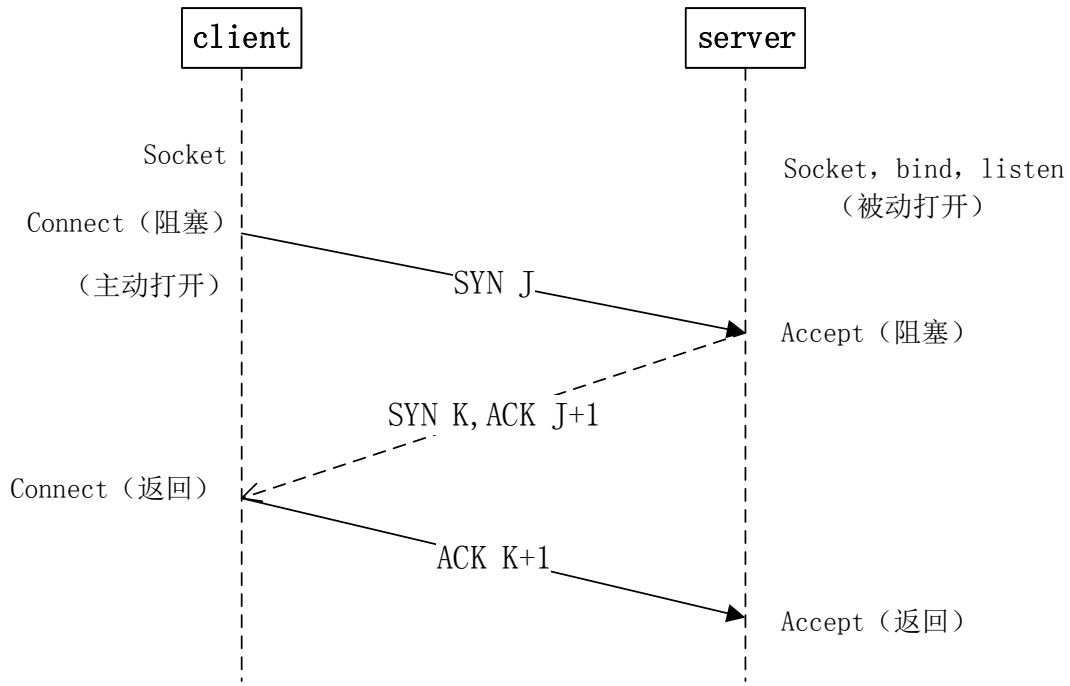
(9) 客户端向 socket 写入信息

(10) 服务器读取信息

(11) 客户端关闭

(12) 服务器关闭

其中 5、6、7、8 客户端与服务器建立连接的方式实质是利用 TCP 协议通过三次握手建立，具体过程如下：



客户端与服务器端的三次握手

3. Python socket 模块实例

(1) Socket_server.py

```
from socket import *
from time import ctime
import threading
HOST = ''
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST, PORT)
def msgHandler(conn_socket, addr):
    """
    对消息进行处理
    """
    print "connected from :", addr
    while True:
        data = conn_socket.recv(BUFSIZE)
        if not data:
            break
        conn_socket.send("[%s] %s"%(ctime(), data))
    conn_socket.close()
    print addr,"---closed"

def main():
    """
    tcpServerSocket Demo
    """
    tcpSocket = socket(AF_INET, SOCK_STREAM)
    tcpSocket.bind(ADDR)
    tcpSocket.listen(5)
    try:
        while True:
            print "waiting for connection..."
            connSock, addr = tcpSocket.accept()
            print connSock, addr
            thread = threading.Thread(target=msgHandler,args=(connSock,addr))
            thread.start()
    except Exception,e:
        tcpSocket.close()

if __name__ == "__main__":
    main()
```

(2) Socket_client.py

```
from socket import *
```

```
HOST = "10.67.0.134"
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST, PORT)
def main():
    """
    tcpClientSocket Demo
    """
    tcpSocket = socket(AF_INET, SOCK_STREAM)
    tcpSocket.connect(ADDR)
    while True:
        data = raw_input("请输入信息:")
        if not data:
            break
        tcpSocket.send(data)
        data = tcpSocket.recv(BUFSIZE)
        if not data:
            break
        print "from sever message:", (data)
    tcpSocket.close()

if __name__    == "__main__":
    main()
```