Python 网络编程介绍

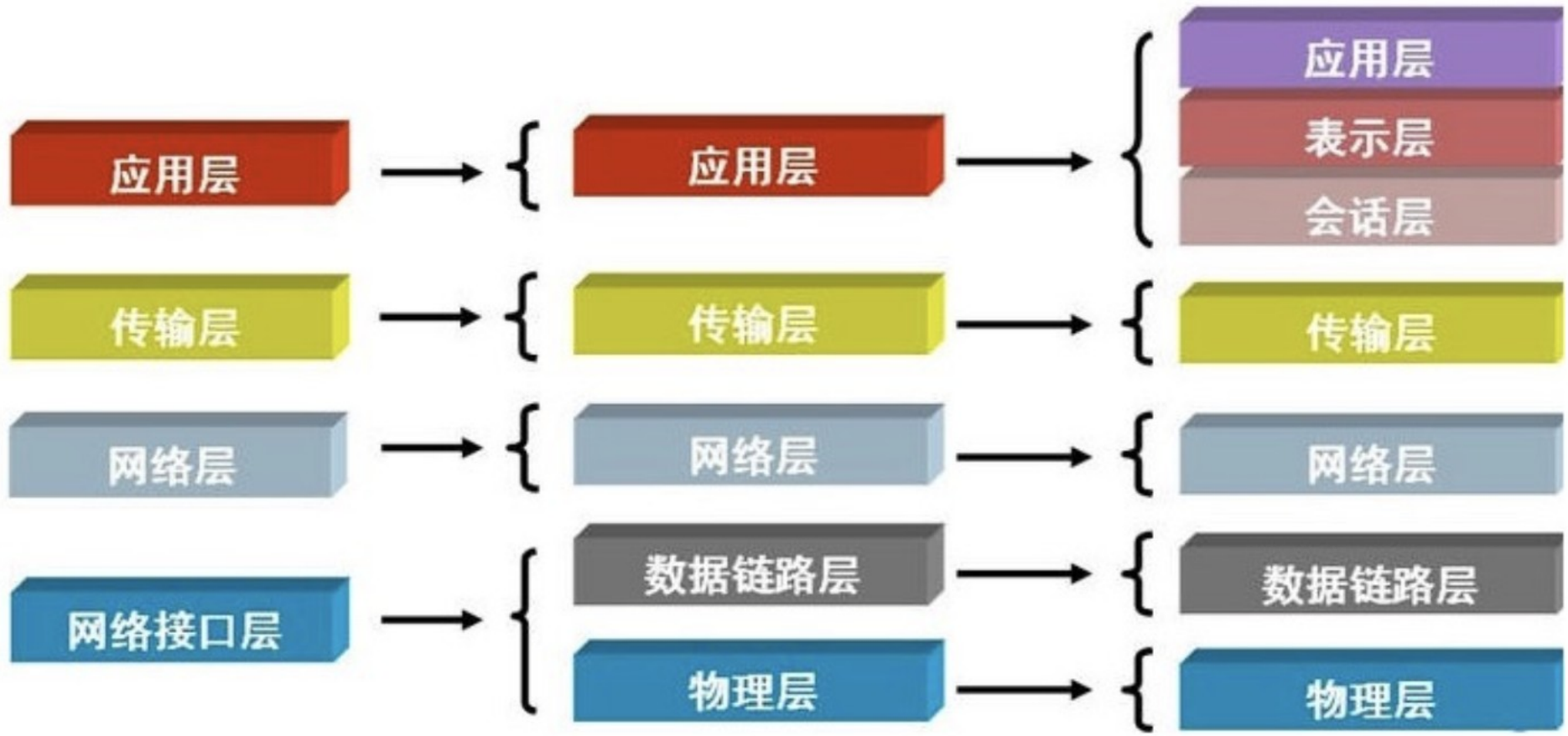
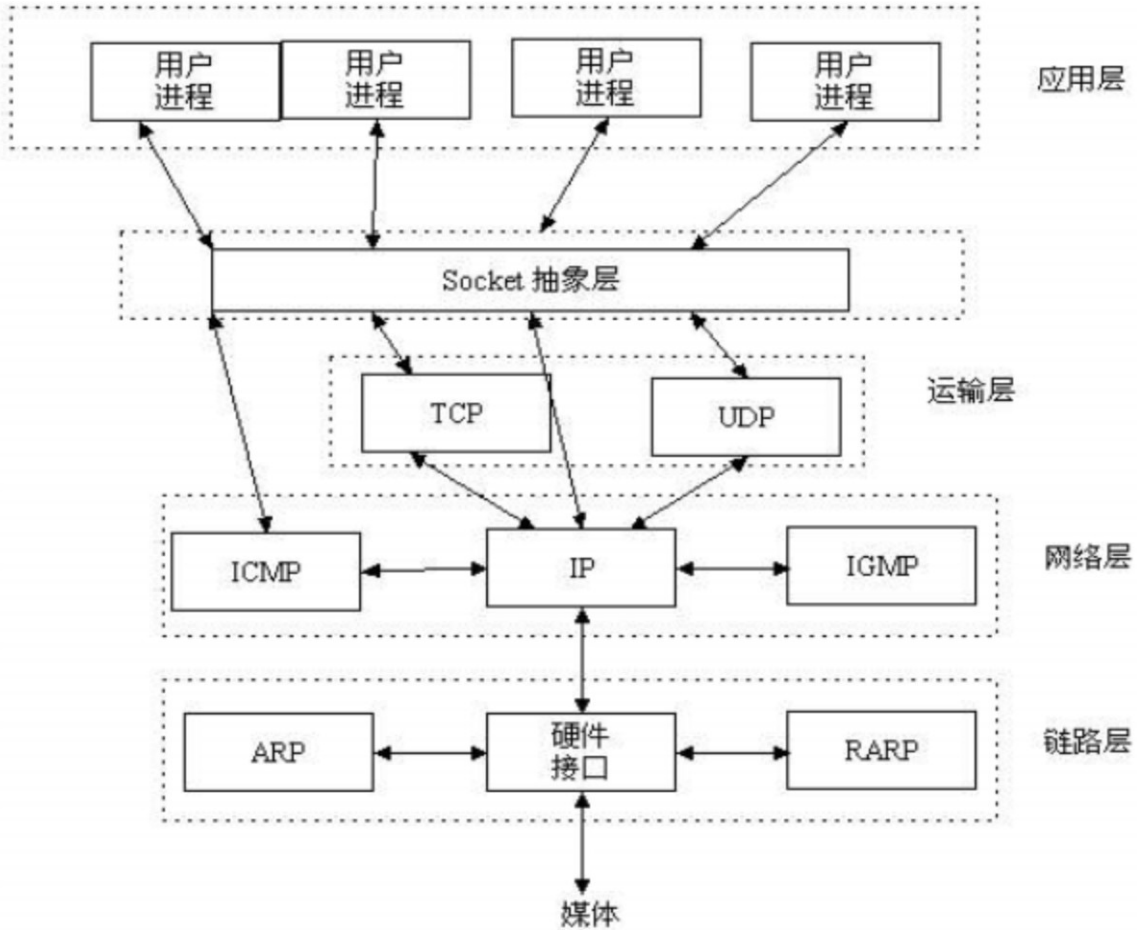


图 3.2 osi七层模型

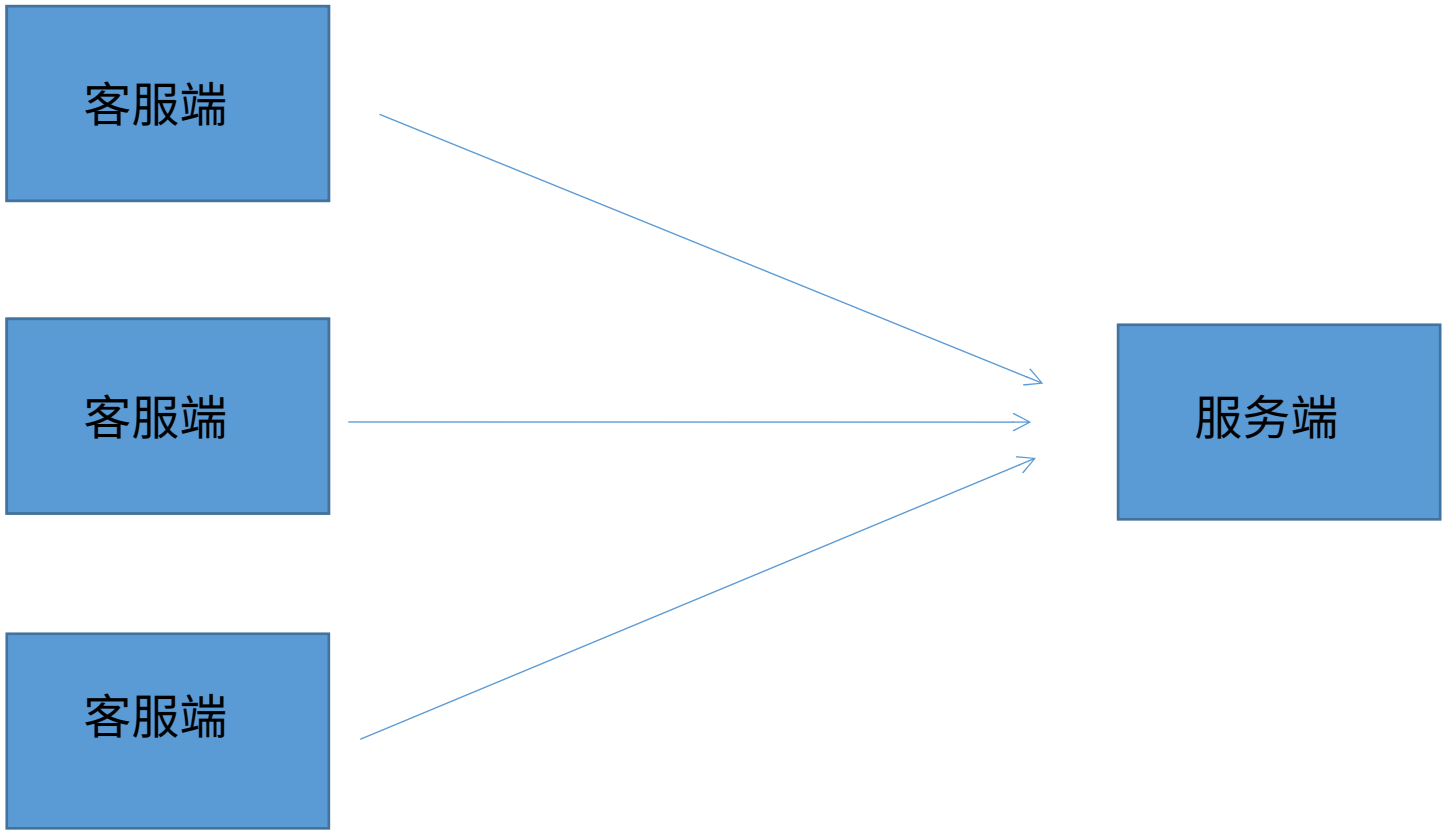

用户
进程
用户
进程
用户
进程
用户
进程
应用层
Socket 抽象层
运输层
TCP
UDP
网络层
ICMP
IP
IGMP
链路层
ARP
硬件
接口
RARP
媒体

每台主机有一个唯一的主机地址标识（ip），同时主机内还有标识服务的序号 id，称作端口（port）， socket 是两台主机之间信息传输的端点，绑定了相应的 ip+port，可以用（ip:port）的形式表示一个 socket

信息 : 需要寄的快递
ip: 小区
port: 门牌号 , 共有 65536 个端口
socket: 快递地址（小区 + 门牌号）
TCP， UCP 等协议 : 快递公司

利用 socket 发送消息 : 把快递（消息）放到门口（socket），由快递公司（TCP 等协议）负责送到对应的地址（对方 socket）

C/S 服务模式



以买火车票为例：
客户端：发出查询请求，如果有则购买一张票
服务端：维护余票情况，如果有余票则卖票给客户端，余票数量减一；没有则返回购买失败
注：以 12306APP 买票是 C/S 服务模式，用 12306 网页购票是 B/S 模式

socket类型	描述
socket.AF_UNIX	只能够用于单一的Unix系统进程间通信
socket.AF_INET	IPv4
socket.AF_INET6	IPv6
socket.SOCK_STREAM	流式socket , for TCP
socket.SOCK_DGRAM	数据报式socket , for UDP
socket.SOCK_RAW	原始套接字，普通的套接字无法处理ICMP、IGMP等网络报文，而SOCK_RAW可以；其次，SOCK_RAW也可以处理特殊的IPv4报文；此外，利用原始套接字，可以通过IP_HDRINCL套接字选项由用户构造IP头。
socket.SOCK_SEQPACKET	可靠的连续数据包服务
创建TCP Socket：	s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
创建UDP Socket：	s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

创建套接字 : socket.socket(family, type)
参数说明:

family: 套接字家族，可以使 AF_UNIX 或者 AF_INET, 一般是 AF_INET 。
type: 套接字类型，根据是面向连接的还是非连接分为 SOCK_STREAM 或 SOCK_DGRAM ，也就是 TCP 和 UDP 的区别 , 一般是 SOCK_STREAM 。

服务器端方法	
s.bind()	绑定地址（host,port）到套接字，在AF_INET下,以元组（host,port）的形式表示地址。
s.listen(backlog)	开始监听。backlog指定在拒绝连接之前，操作系统可以挂起的最大连接数量。该值至少为1，大部分应用程序设为5就可以了。
s.accept()	被动接受客户端连接,(阻塞式)等待连接的到来，并返回（conn,address）二元元组,其中conn是一个通信对象，可以用来接收和发送数据。address是连接客户端的地址。
客户端方法	
s.connect(address)	客户端向服务端发起连接。一般address的格式为元组（hostname,port），如果连接出错，返回socket.error错误。
s.connect_ex()	connect()函数的扩展版本,出错时返回出错码,而不是抛出异常

s.recv(bufsize)	接收数据，数据以bytes类型返回，bufsize指定要接收的最大数据量。
s.send()	发送数据。返回值是要发送的字节数量。
s.sendall()	完整发送数据。将数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回None，失败则抛出异常。
s.recvform()	接收UDP数据，与recv()类似，但返回值是（data,address）。其中data是包含接收的数据，address是发送数据的套接字地址。
s.sendto(data,address)	发送UDP数据，将数据data发送到套接字，address是形式为（ipaddr，port）的元组，指定远程地址。返回值是发送的字节数。
s.close()	关闭套接字，必须执行。
s.getpeername()	返回连接套接字的远程地址。返回值通常是元组（ipaddr,port）。
s.getsockname()	返回套接字自己的地址。通常是一个元组(ipaddr,port)
s.setsockopt(level,optname,value)	设置给定套接字选项的值。
s.getsockopt(level,optname[.buflen])	返回套接字选项的值。
s.settimeout(timeout)	设置套接字操作的超时期，timeout是一个浮点数，单位是秒。值为None表示没有超时期。一般，超时期应该在刚创建套接字时设置，因为它们可能用于连接的操作（如connect()）

```python
# 定义服务器信息
print('初始化服务器主机信息')
address = (host, port)
# 创建TCP服务socket对象
print("初始化服务器主机套接字对象.....")
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 关掉连接释放掉相应的端口
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# 绑定主机信息
print('绑定的主机信息.....')
server.bind(address)
# 启动服务器 一个只能接受一个客户端请求，可以有1个请求排队
print("开始启动服务器.....")
server.listen(5)
#等待连接
while True:
    # 等待来自客户端的连接
    print('等待客户端连接')
    conn, addr = server.accept()  # 等电话
    print('连接的客服端套接字对象为：{}\n客服端的IP地址（拨进电话号码）：{}'.format(conn, addr))
    buy_ticket(conn)
```

```python
#-*- coding: utf-8 -*-
import socket  # 导入 socket 模块
from socket_info import port,host

client = socket.socket()  # 创建 socket 对象
client.connect((host, port))
data = client.recv(100).decode('utf-8')
ticket_num,if_bought = int(data[:-1]),int(data[-1])
if not if_bought:
    print(f'现在还剩下{ticket_num}张票,客户端1没有买到票')
else:
    print(f'现在还剩下{ticket_num}张票,客户端1成功买到了一张票')
client.close()
```

单进程服务端

弊端：顺序，一个客户端堵塞会影响其余客户端

In [5]:
```python
# -*- coding: utf-8 -*-
import socket
import time
from socket_info import port,host

ticket_num = 1
def buy_ticket(conn):
    if_bought = 0
    global ticket_num
    if ticket_num > 0:
        ticket_num  -= 1
        if_bought = 1
    # 模拟信号传输时间
    time.sleep(5)
    conn.send((str(ticket_num) + str(if_bought)).encode('utf-8'))
    conn.close()

# 定义服务器信息
print('初始化服务器主机信息')
address = (host, port)
# 创建TCP服务socket对象
print("初始化服务器主机套接字对象......")
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 关掉连接释放掉相应的端口
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# 绑定主机信息
print('绑定的主机信息.......')
server.bind(address)
# 启动服务器 一个只能接受一个客户端请求，可以有1个请求排队
print("开始启动服务器......")
server.listen(5)
#等待连接
while True:
    # 等待来自客户端的连接
    print('等待客户端连接')
    conn, addr = server.accept()  # 等电话
    print('连接的客服端套接字对象为：{}\n客服端的IP地址（拨进电话号码）：{}'.format(conn, addr))
    buy_ticket(conn)
```

```
初始化服务器主机信息
初始化服务器主机套接字对象......
绑定的主机信息.......
开始启动服务器......
等待客户端连接
连接的客服端套接字对象为：<socket.socket fd=60, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr
=('127.0.0.1', 5002), raddr=('127.0.0.1', 58255)>
客服端的IP地址（拨进电话号码）：('127.0.0.1', 58255)
等待客户端连接
```

客户端

```
#-*- coding: utf-8 -*-
import socket  # 导入 socket 模块
from socket_info import port,host

client = socket.socket()  # 创建 socket 对象
client.connect((host, port))
data = client.recv(100).decode('utf-8')
ticket_num,if_bought = int(data[:-1]),int(data[-1])
if not if_bought:
    print(f'现在还剩下{ticket_num}张票,客户端1没有买到票')
else:
    print(f'现在还剩下{ticket_num}张票,客户端1成功买到了一张票')
client.close()
```

现在还剩下0张票,客户端1成功买到了一张票

多进程服务端

```python
# 定义服务器信息
print('初始化服务器主机信息')
address = (host, port)
# 创建TCP服务socket对象
print("初始化服务器主机套接字对象......")
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 关掉连接释放掉相应的端口
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# 绑定主机信息
print('绑定的主机信息......')
server.bind(address)
# 启动服务器 一个只能接受一个客户端请求，可以有1个请求排队
print("开始启动服务器......")
server.listen(5)
#等待连接
while True:
    # 等待来自客户端的连接
    print('等待客户端连接')
    conn, addr = server.accept()
    print('连接的客服端套接字对象为：{}\n客服端的IP地址（拨进电话号码）：{}'.format(conn, addr))
    p = Process(target=buy_ticket,args=(conn,))
    p.start()
```

```
初始化服务器主机信息
初始化服务器主机套接字对象......
绑定的主机信息......
开始启动服务器......
等待客户端连接
连接的客服端套接字对象为：<socket.socket fd=60, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr
=('127.0.0.1', 5002), raddr=('127.0.0.1', 58255)>
客服端的IP地址（拨进电话号码）：('127.0.0.1', 58255)
等待客户端连接
```

读取网页信息

```
建立连接的远程服务器地址: ('36.152.44.96', 80)
读取数据数量: 1024
HTTP/1.1 200 OK
Server:
Date: Sat, 14 Mar 2020 17:42:01 GMT
Content-Type: text/html
Content-Length: 14615
Connection: keep-alive
Accept-Ranges: bytes
P3p: CP=" OTI DSP COR IVA OUR IND COM "
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Pragma: no-cache
Set-Cookie: BAIDUID=4E5846B32DE116534794758CD9FE249A:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; p
ath=/; domain=.baidu.com
Set-Cookie: BIDUPSID=4E5846B32DE116534794758CD9FE249A; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path
=/; domain=.baidu.com
Set-Cookie: PSTM=1584207721; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BAIDUID=4E5846B32DE11653369D862BFA16C6CD:FG=1; max-age=31536000; expires=Sun, 14-Mar-21 17:42:01 GMT; dom
ain=.baidu.com; path=/; version=1; comment=bd
```

```python
import socket


def getDataBySocket(url):
    sock = socket.socket()
    ip_port = (url, 80)
    sock.connect(ip_port)

    print("建立连接的远程服务器地址:", sock.getpeername())

    sock.send(b"GET / HTTP/1.1\r\n")
    sock.send(("Host: " + url + "\r\n").encode("utf-8"))
    sock.send(("\n").encode("utf-8"))

    size = 1024
    while True:
        try:
            data = sock.recv(size)
            count = len(data)
            print("读取数据数量:", count)

            if count == 0:
                print("读数据完毕")
                break

            ret = str(data.decode("utf-8"))
            print(ret)
        except BaseException as exc:
            print("发生异常")
            break


url = "www.baidu.com"
getDataBySocket(url)
```

socket 是 python 用于网络编程的基础。

利用 socket 为基础在各个子领域构建起更灵活的框架。
以 web 服务为例 :
Flask:web 开发框架
Gunicorn: 多进程管理
nginx: 负载均衡