

## 第23讲 | 如何判断客户端的网络连接？

2018-07-17 蔡能

从0开始学游戏开发

[进入课程 >](#)



讲述：蔡能

时长 06:45 大小 3.10M



由于涉及到网络、脚本语言等等，这一节起，我要开始讲一些基础的、看起来比较枯燥的知识。我会尽力写得有趣生动，但是，知识的获取并不容易，即便我已经在努力去讲解，还是需要你用更多的时间去摸索和学习。

我们在前面说了 Pygame 的一些客户端知识，如果你想让这款游戏能够在网络上传输数据，接下来，那就需要编写服务器端，并且在客户端做客户端连接的准备。

前面我们已经用 Pygame 讲解了很多内容，那我们客户端的网络连接，我也继续使用 Python 来编写，当然这已经和 Pygame 没有关系了。因为网络连接是独立的代码，所以只需要直接写在游戏里就可以了。

在开始编写网络部分之前，我们需要整理一下网络的基础知识。如果我们一上来就编写网络代码，对于你来说，可能会有一些概念上的模糊。

对于网络编程，或许你已经接触到了，或许你只是有点概念，或许你一点都没接触过。但是你或许听说过 Socket 套接字、TCP/IP 协议、UDP 协议、HTTP、HTTPS 协议等等，那么这些协议是不是都属于网络编程范畴？还是这里面哪些部分是属于网络编程范畴呢？

网络，从模型上讲，一共分为七层，从底层到最上层，分别是：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

我来分别解释一下。

**物理层：**所谓的物理层，你可以理解为我们看到的各种网络线，也就是人肉眼能看到的物理线路，包括光纤、以前连接调制解调器的电话线等等。这些线路就是物理层。物理层有物理层的规范，比如电流、编码、帧、连接头等等。你只需要知道物理层也是存在规范的，就可以了。**物理层最主要的功能就是网络的物理连接。**

**数据链路层：**所谓的数据链路层，就是建立逻辑连接，然后进行硬件上的寻址操作、差错的校验，然后将二进制的字节进行组合，通过 MAC 地址进行访问，比如网卡、交换机等等。你需要记住的是，**在这一层，要通过 MAC 地址来进行访问，进行硬件寻址操作。**

**网络层：**网络层进行逻辑地址的寻址操作和数据链路层不同。数据链路是使用硬件寻址操作，而网络层是使用逻辑地址寻址，比如路由器、防火墙、多层交换机等等。我们最熟悉的 IPv4 (202.101.253.233)、IPv6、ARP 等等都属于这一层。你在这里需要记住的是，**网络层是逻辑寻址操作，会用到 ARP、IPv4 等等协议。**

**传输层：**在编程中最常用到的 TCP、UDP 等等协议，都在这一层进行操作，它首先定义了数据传输的协议端口号以及一些错误的检测。

**会话层：**会话层在传输层之上，它就在客户端和服务端。严谨地说，就是本地机器和远端机器之间的会话，比如要进行断点续传这些操作，就属于会话层的范畴。

**表示层：**表示层很容易理解，就是数据的传输，然后展现在电脑上。比如图片的传输和显示、网络地址的加密等等。

**应用层：**应用层就是提供给电脑用户的各种网络应用，比如你自己编写的网络软件、FTP 软件、Telnet、浏览器等等。

以上这些点你要硬性记住的话，会比较困难。我教给你一个方法。

首先，我们想象一段从网线过来一段数据，网线就是“物理层”，那么数据需要找到一个门牌号，这个门牌号是一个硬件地址，可能是你的电脑网卡，也可能是你公司的交换机。这些数据需要把这些门牌地址连接起来，这就是“数据链路层”。

随后，这些数据找到门牌号后，就需要分发到逻辑地址，比如路由器或者你的 IP 地址，这些逻辑地址就是网络地址，这就是“网络层”。

经过网络层后，就要看这是什么数据，是 TCP 协议的，还是 UDP 协议的。知道了协议后，才可以传输数据，所以这个是“传输层”。

那么在传输的过程中，可能会中断，所以我们需要登录服务器，断点续传进行重新传输，这些属于机器和机器之间的会话，所以是“会话层”。

传输完数据后，我们就会在电脑里显示这个内容，是一幅图片呢，还是一段电影？这个需要表示出来，所以是“表示层”。

最后，我们将这个一整套的东西，写成了一个应用，这就是“应用层”。


虽然这么表述起来，有许多不精确不严谨的地方，但是通过这段话能让你很快记住这个七层网络模型，对你将来的编程有很大的帮助。

Python 支持 Socket 编程，也就是支持 TCP、UDP 等协议的编程，也支持更上层的编程，比如 HTTP 协议等等。在今天的 content 中，我们要用到 TCP 编程。至于为什么要使用 TCP，有这样几个原因：

1. TCP 保证连接的**正确性**。在建立 TCP 连接的时候，需要经过三次握手，连接这一方发送 SYN 协议，被连接方返回 SYN+ACK 协议，最后连接方再返回 ACK 协议；
2. TCP 保证如果在一定时间内没有收到对方的信息，就重发消息，保证消息传输的**可靠性**；
3. TCP 可以进行**流量控制**。它拥有固定大小的缓冲池，TCP 的接收方只允许另一方发送缓冲池内所接纳的数据大小。

TCP 还有其他更多的保证传输可靠性的内容和标准，我在这里不做更多的阐述。另外，使用 TCP 可以进行长时间的连接，在客户端和服务端之间进行不停地交互。在交互过程中，服务端发送数据给客户端，客户端就能做出相应的回应。

在 Python 中编写 TCP 协议的代码比之使用 C/C++ 更为方便。因为 C/C++ 需要初始化一系列的内容，然后进行顺序的流程化绑定，设置网络参数，最后进行发送和接收操作，在结束的时候进行资源的回收。而在 Python 这里，只需要设置协议和 IP 地址就可以实现 TCP 协议编程。我们来看一段代码。

 复制代码

```
1 import socket
2 class go_sock(object):
3     ip = ""
4     port = 0
5     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     def __init__(self):
7         object.__init__(self)
8     def connect(self, ip, port):
9         self.ip = ip
10        self.port = port
11        self.sock.connect_ex((ip, port))
12    def close(self):
13        self.sock.close()
```

我在这里编写了一个类，这个类将 TCP 的内容封装在了类中，这样，我们的网络代码能在游戏中方便地初始化，使用起来就很方便。

首先，我们看到在类里面定义了 ip、port、sock 这三个变量，这三个变量分别是对应 IP 地址、端口号以及 socket 句柄。在类里，我们直接将 sock 初始化为 socket 类，其中 socket 类填写的内容中，参数 1 是服务器之间的网络通信，参数 2 是流 Socket，这里指的是 TCP 协议。

在初始化完成了之后，我们看到 connect 函数。在函数里面，我们看到参数对变量的初始化，其中 sock 句柄调用了标准 socket 函数 sock.connect\_ex，这个函数负责与对方进行一个连接。

最后的函数是 close 关闭操作，在任务完成之后，你可以调用 close 函数进行 socket 句柄的关闭。

我们可以这样使用这个类。

 复制代码

```
1 _inet = go_sock()  
2 _inet.connect("115.231.74.62", 21)  
3 _inet.sock.recv(100)
```

在这里，我们可以简单测试一下某些应用服务器，然后接收返回内容。这个类的封装工作到此就告一个段落，更多的网络服务和交互的编写，我将在下一节阐述。

## 小结

今天我们学习了网络的七层模型结构，以及我们将要在游戏中使用的 TCP 协议的编程。

我用了一个传输过程介绍了七层每一层做的事情，这个你一定要牢记。

我们使用 Python 封装了 Socket 库的细节内容，只需要直接编写 connect 代码就可以进行数据的接收和发送操作了。

选择 TCP 协议是因为它安全可靠，能保证游戏传输的过程中不出错。

现在，我给你留一个小问题吧。

如果我们要使用 UDP 来编写这个网络服务，该如何保证数据的准确性呢？选择 UDP 协议的优势在哪里？

欢迎留言说出你的看法。我在下一节的挑战中等你！

# 从0开始学游戏开发

你的游戏开发入门第一课

蔡能

原网易游戏引擎架构师  
资深游戏底层技术专家



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 第22讲 | 热点剖析（五）：如何选择移动端的游戏引擎？

下一篇 复习课 | 带你梳理客户端开发的三个重点

## 精选留言 (5)

写留言



淡看烟雨\_

2018-08-28

1

TCP属于长链接，规模上来以后必然很消耗资源，但是比较稳定，适合保持即时数据的传输，前提是得想办法尽量传输游戏过程中的关键数据。UDP不稳定，类似广播，可以用来做聊天，邮件，记录账户情况等，为保证UDP的完整性，可以加入重传机制，同步机制，单条数据可以使用CRC校验，增加同步请求标识，细致点可以自定义一个简单的通信会话模式。

展开



wusiration

2018-07-30

1

UDP传输层无法保证数据的可靠传输，只能靠应用层来实现可靠性传输，在应用层实现类

似TCP的确认、重传、窗口确认等机制。使用UDP的好处在于其资源消耗小，处理速度快。目前有RUDP、RTP、UDT等程序实现可靠传输。

展开 ▾



**王俊涛**

2018-07-18



这个类的类变量 实例变量 之间怎么操作的

展开 ▾

作者回复: 用点操作符连接



**王俊涛**

2018-07-18



这个类的类变量 实例变量 之间怎么操作的

展开 ▾



**Alan**

2018-07-17



以后会有深入一点课程吗

展开 ▾

作者回复: 会有的

