# 网络编程懒人入门(四)：快速理解TCP和UDP的差异-网络编程/专项技术区 - 即时通讯开发者社区!


原作者：MeloDev，本文由即时通讯网重新修订发布，感谢原作者的无私分享。

## 1、前言

对于即时通讯开者新手来说，在开始着手编写IM或消息推送系统的代码前，最头疼的问题莫过于到底该选TCP还是UDP作为传输层协议。本文延续《网络编程懒人入门》系列文章的风格，通过快速对比分析 TCP 和 UDP 的区别，来帮助即时通讯初学者快速了解这些基础的知识点，从而在IM、消息推送等网络通信应用场景中能准确地选择合适的传输层协议。

即时通讯网的另一篇文章《简述传输层协议TCP和UDP的

区别》也阐述了类似的内容，希望能为您提供更多的参考。

## 2、系列文章

**本文是系列文章中的第4篇，本系列文章的大纲如下：**

- 《网络编程懒人入门(一)：快速理解网络通信协议（上篇）》
- 《网络编程懒人入门(二)：快速理解网络通信协议（下篇）》
- 《网络编程懒人入门(三)：快速理解TCP协议一篇就够》
- 《网络编程懒人入门(四)：快速理解TCP和UDP的差异》（本文）
- 《网络编程懒人入门(五)：快速理解为什么说UDP有时比TCP更有优势》
- 《网络编程懒人入门(六)：史上最通俗的集线器、交换机、路由器功能原理入门》
- 《网络编程懒人入门(七)：深入浅出，全面理解HTTP协议》
- 《网络编程懒人入门(八)：手把手教你写基于TCP的Socket长连接》
- 《网络编程懒人入门(九)：通俗讲解，有了IP地址，为何还要用MAC地址？》

**本站的《脑残式网络编程入门》也适合入门学习，本系列**

大纲如下：

- 《脑残式网络编程入门(一)：跟着动画来学TCP三次握手和四次挥手》
- 《脑残式网络编程入门(二)：我们在读写Socket时，究竟在读写什么？》
- 《脑残式网络编程入门(三)：HTTP协议必知必会的一些知识》
- 《脑残式网络编程入门(四)：快速理解HTTP/2的服务器推送(Server Push)》

如果您觉得本系列文章过于基础，您可直接阅读《不为人知的网络编程》系列文章，该系列目录如下：

- 《不为人知的网络编程(一)：浅析TCP协议中的疑难杂症(上篇)》
- 《不为人知的网络编程(二)：浅析TCP协议中的疑难杂症(下篇)》
- 《不为人知的网络编程(三)：关闭TCP连接时为什么会TIME_WAIT、CLOSE_WAIT》
- 《不为人知的网络编程(四)：深入研究分析TCP的异常关闭》
- 《不为人知的网络编程(五)：UDP的连接性和负载均衡》
- 《不为人知的网络编程(六)：深入地理解UDP协议并用好它》
- 《不为人知的网络编程(七)：如何让不可靠的UDP变

的可靠？》

- 《不为人知的网络编程(八)：从数据传输层深度解密HTTP》
- 《不为人知的网络编程(九)：理论联系实际，全方位深入理解DNS》

**关于移动端网络特性及优化手段的总结性文章请见：**

- 《现代移动端网络短连接的优化手段总结：请求速度、弱网适应、安全保障》
- 《移动端IM开发者必读(一)：通俗易懂，理解移动网络的“弱”和“慢”》
- 《移动端IM开发者必读(二)：史上最全移动弱网络优化方法总结》

## 3、参考资料


《TCP/IP详解 - 第11章·UDP：用户数据报协议》
《TCP/IP详解 - 第17章·TCP：传输控制协议》
《TCP/IP详解 - 第18章·TCP连接的建立与终止》
《TCP/IP详解 - 第21章·TCP的超时与重传》
《通俗易懂-深入理解TCP协议（上）：理论基础》
《通俗易懂-深入理解TCP协议（下）：RTT、滑动窗口、拥塞处理》
《理论经典：TCP协议的3次握手与4次挥手过程详解》

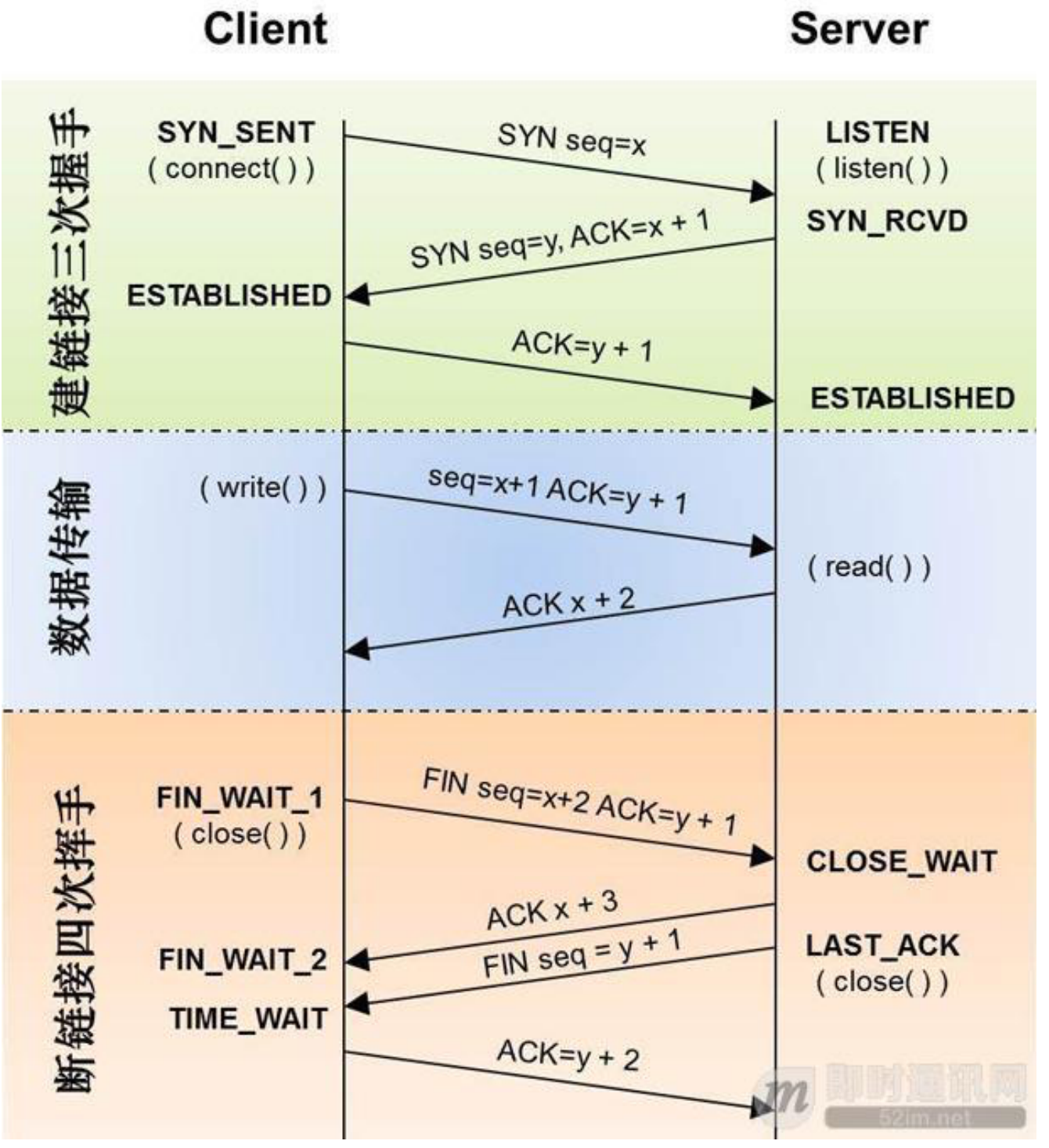
《理论联系实际：Wireshark抓包分析TCP 3次握手、4次挥手过程》
《技术往事：改变世界的TCP/IP协议（珍贵多图、手机慎点）》
《计算机网络通讯协议关系图（中文珍藏版）》
《高性能网络编程(一)：单台服务器并发TCP连接数到底可以有多少》
《高性能网络编程(二)：上一个10年，著名的C10K并发连接问题》
《高性能网络编程(三)：下一个10年，是时候考虑C10M并发问题了》
《高性能网络编程(四)：从C10K到C10M高性能网络应用的理论探索》
《简述传输层协议TCP和UDP的区别》
《UDP中一个包的大小最大能多大？》
《为什么QQ用的是UDP协议而不是TCP协议？》
《移动端即时通讯协议选择：UDP还是TCP？》

# 4、建立连接方式的差异

## 4.1TCP

说到 TCP 建立连接，相信大多数人脑海里肯定可以浮现出一个词，没错就是--“三次握手”。TCP 通过“三次握手”来建立连接，再通过“四次挥手”断开一个连接。在每次挥手中 TCP 做了哪些操作呢?

流程如下图所示（**TCP的三次握手和四次挥手**）：




上图就从客户端和服务端的角度，清楚的展示了 TCP 的三次握手和四次挥手。可以看到，当 TCP 试图建立连接时，三次握手指的是客户端主动触发了两次，服务端触发了一次。

**我们可以先明确一下 TCP 建立连接并且初始化的目标是**

什么呢?

- 1）初始化资源；
- 2）告诉对方我的序列号。

所以三次握手的次序是这样子的：

- 1）client端首先发送一个SYN包告诉Server端我的初始序列号是X；
- 2）Server端收到SYN包后回复给client一个ACK确认包，告诉client说我收到了；
- 3）接着Server端也需要告诉client端自己的初始序列号，于是Server也发送一个SYN包告诉client我的初始序列号是Y；
- 4）Client收到后，回复Server一个ACK确认包说我知道了。

其中的 2 、3 步骤可以简化为一步，也就是说将 ACK 确认包和 SYN 序列化包一同发送给 Client 端。到此我们就比较简单的解释了 TCP 建立连接的“三次握手”。

## 4.2UDP

我们都知道 TCP 是面向连接的、可靠的、有序的传输层协议，而 UDP 是面向数据报的、不可靠的、无序的传输

协议，所以 UDP 压根不会建立什么连接。

就好比发短信一样，UDP 只需要知道对方的 ip 地址，将数据报一份一份的发送过去就可以了，其他的作为发送方，都不需要关心。

（关于TCP的3次握手和4次挥手文章，可详见《理论经典：TCP协议的3次握手与4次挥手过程详解》、《理论联系实际：Wireshark抓包分析TCP 3次握手、4次挥手过程》）

## 5、数据发送方式的差异

**关于 TCP、UDP 之间数据发送的差异，可以体现二者最大的不同之处：**

- **TCP：**
  由于 TCP 是建立在两端连接之上的协议，所以理论上发送的数据流不存在大小的限制。但是由于缓冲区有大小限制，所以你如果用 TCP 发送一段很大的数据，可能会截断成好几段，接收方依次的接收。
- **UDP：**
  由于 UDP 本身发送的就是一份一份的数据报，所以自然而然的就有一个上限的大小。

**那么每次 UDP 发送的数据报大小由哪些因素共同决定呢？**

- UDP协议本身，UDP协议中有16位的UDP报文长度，那么UDP报文长度不能超过2^16=65536；
- 以太网(Ethernet)数据帧的长度，数据链路层的MTU(最大传输单元)；
- socket的UDP发送缓存区大小。

先来看第一个因素，UDP 本身协议的报文长度为 2^16 - 1，UDP 包头占 8 个字节，IP 协议本身封装后包头占 20 个字节，所以最终长度为： 2^16 - 1 - 20 - 8 = 65507 字节。

只看第一个因素有点理想化了，因为 UDP 属于不可靠协议，我们应该尽量避免在传输过程中，数据包被分割。所以这里有一个非常重要的概念 MTU -- 也就是最大传输单元。

在 Internet 下 MTU 的值为 576 字节，所以在 internet 下使用 UDP 协议，每个数据报最大的字节数为： 576 - 20 - 8 = 548

（有关UDP协议的最大包长限制，详见《UDP中一个包的大小最大能多大？》）

# 6、数据有序性的差异

我们再来谈谈数据的有序性。

## 6.1TCP

对于 TCP 来说，本身 TCP 有着超时重传、错误重传、还有等等一系列复杂的算法保证了 TCP 的数据是有序的，假设你发送了数据 1、2、3，则只要发送端和接收端保持连接时，接收端收到的数据始终都是 1、2、3。

## 6.2UDP

而 UDP 协议则要奔放的多，无论 server 端无论缓冲池的大小有多大，接收 client 端发来的消息总是一个一个的接收。并且由于 UDP 本身的不可靠性以及无序性，如果 client 发送了 1、2、3 这三个数据报过来，server 端接收到的可能是任意顺序、任意个数三个数据报的排列组合。

# 7、可靠性的差异

其实大家都知道 TCP 本身是可靠的协议，而 UDP 是不可靠的协议。

## 7.1TCP

TCP 内部的很多算法机制让他保持连接的过程中是很可靠的。比如：TCP 的超时重传、错误重传、TCP 的流量控制、阻塞控制、慢热启动算法、拥塞避免算法、快速恢复算法 等等。所以 TCP 是一个内部原理复杂，但是使用起来比较简单的这么一个协议。

## 7.2UDP

UDP 是一个面向非连接的协议，UDP 发送的每个数据报带有自己的 IP 地址和接收方的 IP 地址，它本身对这个数据报是否出错，是否到达不关心，只要发出去了就好了。

**所以来研究下，什么情况会导致 UDP 丢包：**

- **数据报分片重组丢失：**在文章之前我们就说过，UDP 的每个数据报大小多少最合适，事实上 UDP 协议本身规定的大小是 64kb，但是在数据链路层有 MTU 的限制，大小大概在 5kb，所以当你发送一个很大的 UDP 包的时候，这个包会在 IP 层进行分片，然后重组。这个过程就有可能导致分片的包丢失。UDP 本身有 CRC 检测机制，会抛弃掉丢失的 UDP 包；
- **UDP 缓冲区填满：**当 UDP 的缓冲区已经被填满的时候，接收方还没有处理这部分的 UDP 数据报，这个时候再过来的数据报就没有地方可以存了，自然就都被丢弃了。

# 8、使用场景总结

在文章最后的一部分，聊聊 TCP、UDP 使用场景。

先来说 UDP 的吧，有很多人都会觉得 UDP 与 TCP 相比，在性能速度上是占优势的。因为 UDP 并不用保持一个持续的连接，也不需要对收发包进行确认。但事实上经过这么多年的发展 TCP 已经拥有足够多的算法和优化，在网络状态不错的情况下，TCP 的整体性能是优于 UDP 的。

**那在什么时候我们非用 UDP 不可呢?**

- **对实时性要求高**：比如实时会议，实时视频这种情况下，如果使用 TCP，当网络不好发生重传时，画面肯定会有延时，甚至越堆越多。如果使用 UDP 的话，即使偶尔丢了几个包，但是也不会影响什么，这种情况下使用 UDP 比较好；
- **多点通信**：TCP 需要保持一个长连接，那么在涉及多点通讯的时候，肯定需要和多个通信节点建立其双向连接，然后有时在NAT环境下，两个通信节点建立其直接的 TCP 连接不是一个容易的事情，而 UDP 可以无需保持连接，直接发就可以了，所以成本会很低，而且穿透性好。这种情况下使用 UDP 也是没错的。

以上我们说了 UDP 的使用场景，在此之外的其他情况，使用 TCP 准没错。

**毕竟有一句话嘛：**

when in doubt，use TCP。

（原文链接：点此进入，有改动）

# 附录：更多网络编程资料

《Java新一代网络编程模型AIO原理及Linux系统AIO介绍》
《NIO框架入门(一)：服务端基于Netty4的UDP双向通信Demo演示》
《NIO框架入门(二)：服务端基于MINA2的UDP双向通信Demo演示》
《NIO框架入门(三)：iOS与MINA2、Netty4的跨平台UDP双向通信实战》
《NIO框架入门(四)：Android与MINA2、Netty4的跨平台UDP双向通信实战》
《P2P技术详解(一)：NAT详解——详细原理、P2P简介》
《P2P技术详解(二)：P2P中的NAT穿越(打洞)方案详解》
《P2P技术详解(三)：P2P技术之STUN、TURN、ICE详解》
《通俗易懂：快速理解P2P技术中的NAT穿透原理》
>> 更多同类文章 ……