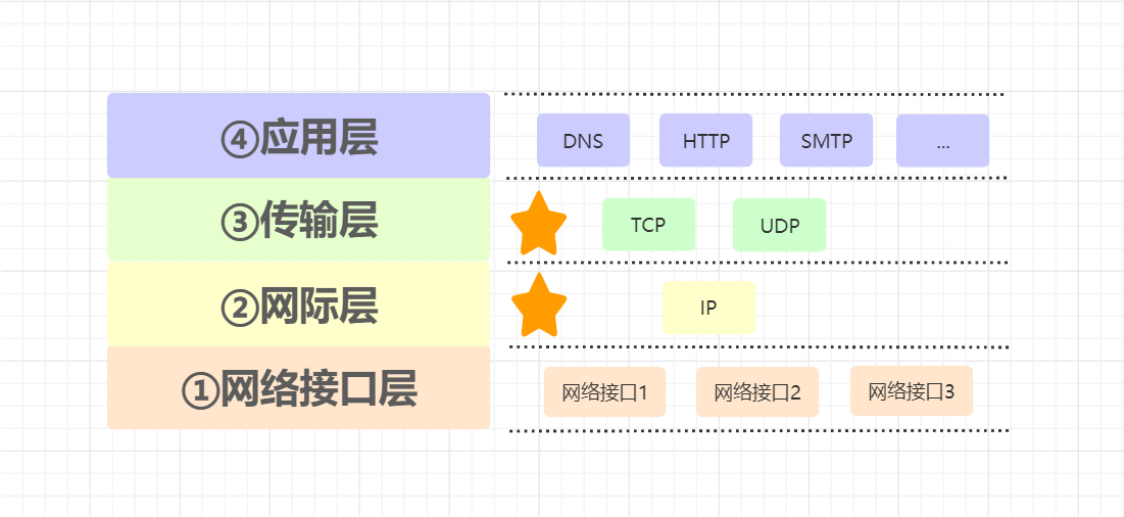
之前我们介绍TCP/IP四层体系结构的时候说过，在传输层中有两个最重要的协议：UDP和TCP协议。这两种协议的使用频率仅次于网络层的IP协议，因此这两个协议尤其是TCP协议将是传输层着重去探讨的对象。



一、为什么会出现TCP和UDP两种协议

首先我们要了解传输层中为什么要有两种协议，一种不行吗?

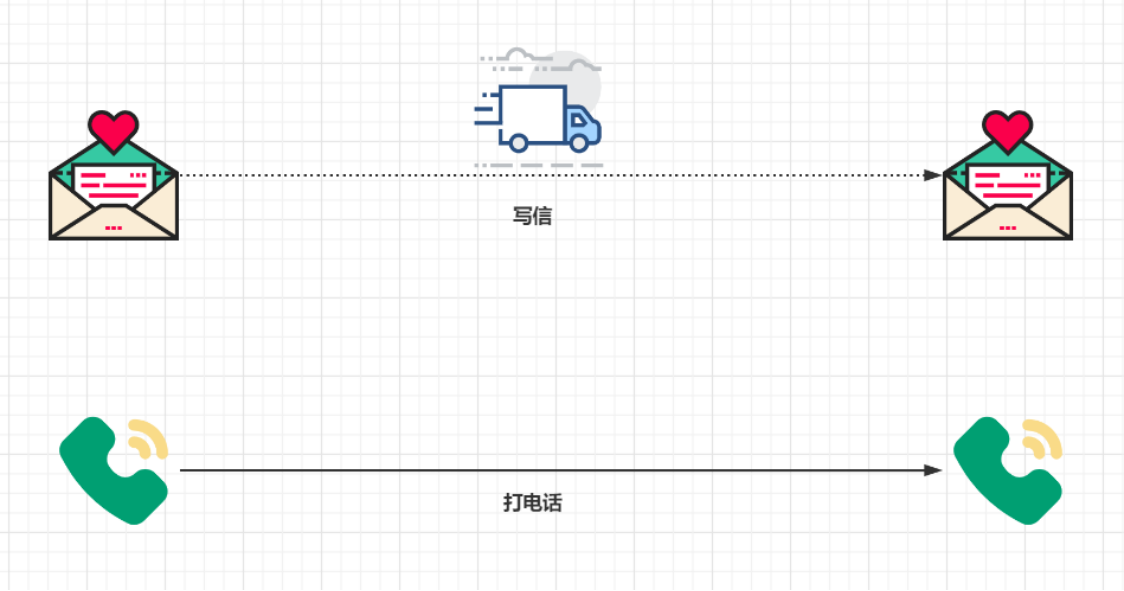
由于我们在网络上传输数据的时候有两类需求：

- **第一类：可靠传输数据，不对速度有太高要求**
- **第二类：尽最大努力、立即传输数据，允许传输时丢失一些信息**

关于第一类，就是对应着我们大多数的应用程序，比如网页，丢失一个数据包，页面可能就无法正确展示了；还有电子邮件、SSH软件、多数在线游戏等等，我们必须不惜一切代价正确接收每个发送的数据包。

至于第二类，相对第一类就少了很多，主要是实时应用，例如互联网广播、互联网电视等，此类场景用户可以忍受偶尔的断断续续，因此丢失一个或多个数据包，没有什么影响。

综上，我们的先辈发明了TCP协议，对应着第一类应用，TCP协议是基于连接的可靠的协议。对应第二类应用，发明了UDP协议，尽最大努力传递报文，不可靠。



TCP协议对应到生活中类似打电话，首先通过“您好/喂/Hello/米西米西”之类的词语确定对方听得到，对方也会“喂/您好/米西米西”确定我能听得到。

TCP本质上就是实现了以上类似的场景，不过机器不像我们人这么聪明，要保证这种可靠的通信，还是蛮复杂的，这也是TCP复杂的原因。

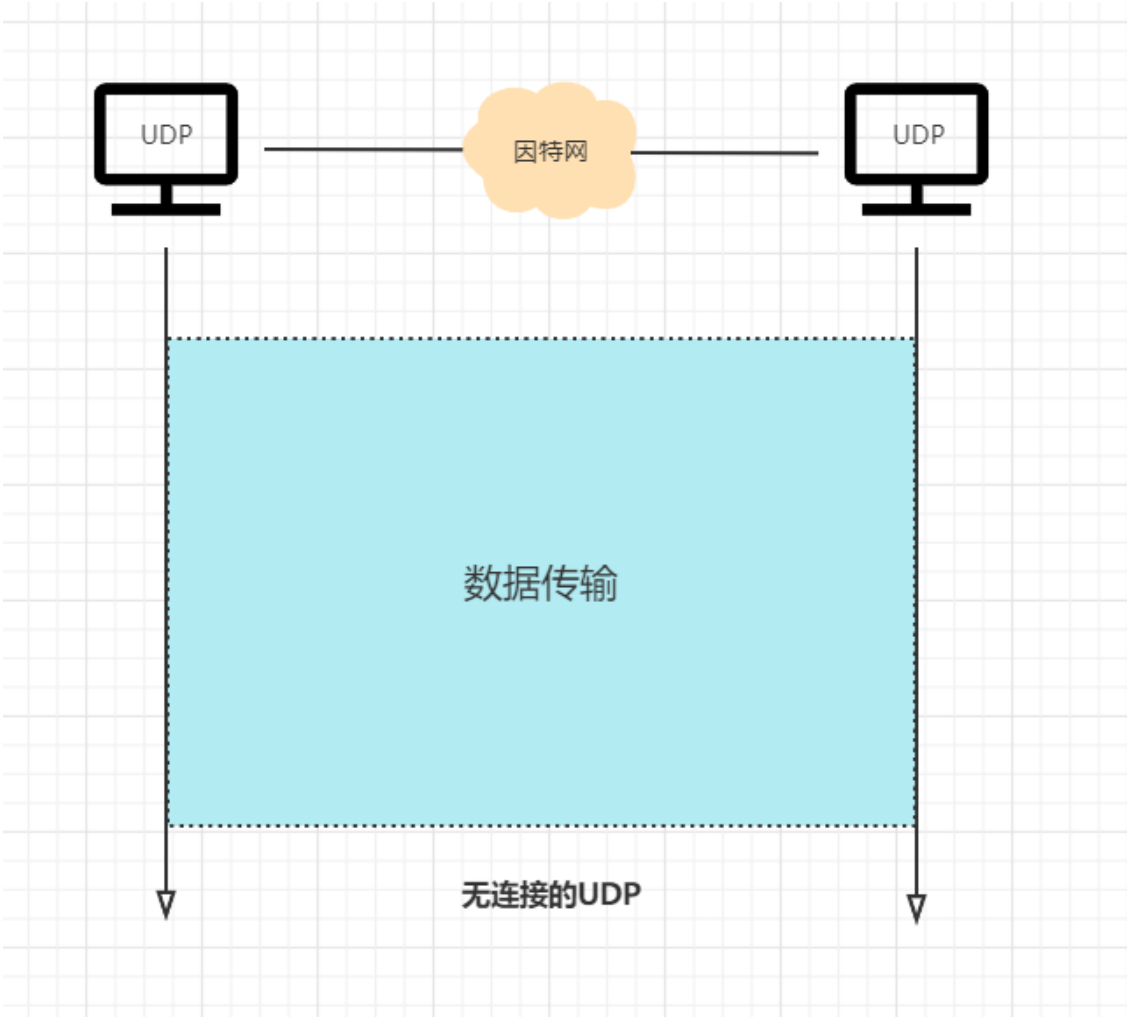
相对于TCP，UDP就简单了很多，因为不可靠，就像寄信，我哗哗哗寄出去几十封就不管了，剩下了就看邮局给力不给力了，正常情况下最多也就丢一两封信件，我们对此表示可以忍受。但是最坏的情况下就是发出去的信一封都没到对方手中。但是没办法啊，我们只管发，才不管对方有没有收到呢，这才简单嘛，这样我们寄信速度是很快的，因为根本没有确认的过程，发了就完事了！因此这也不可靠嘛！所以我们经常说UDP是一个无连接、快速但不可靠的协议。

二、面向连接和面向无连接

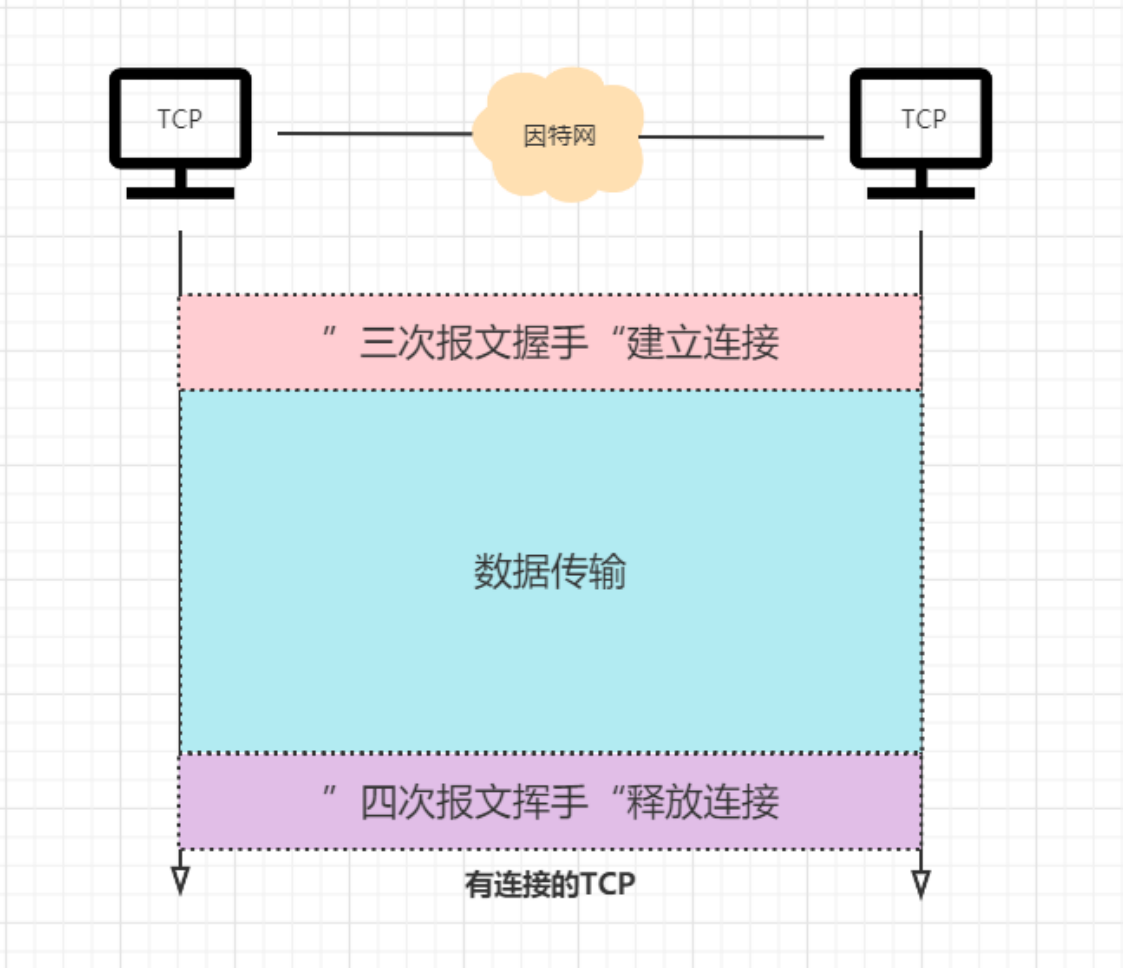
UDP的全称是："User Datagram Protocol"，即用户数据报协议。

而TCP的全称是："Transmission Control Protocol"，即传输控制协议，我们来简单看看他们之间的对比。

UDP是很简单的协议，目标是快速发出去，不需要知道信息是否被正确接收，因此发送的报文也极其简单，并且是随时就可以发送，下图表示两台使用UDP主机通过因特网的通信时大概的样子：



而使用TCP协议进行通信的双方主机，在进行数据传输之前，必须使用“三次报文握手”来建立TCP连接，TCP连接建立成功后才能进行数据传输，数据传输完成后必须使用“四次报文挥手”来释放TCP连接。



注意这里的连接属于逻辑连接，而不是物理连接。综上所述，UDP是面向无连接的，而TCP是面向有连接的。

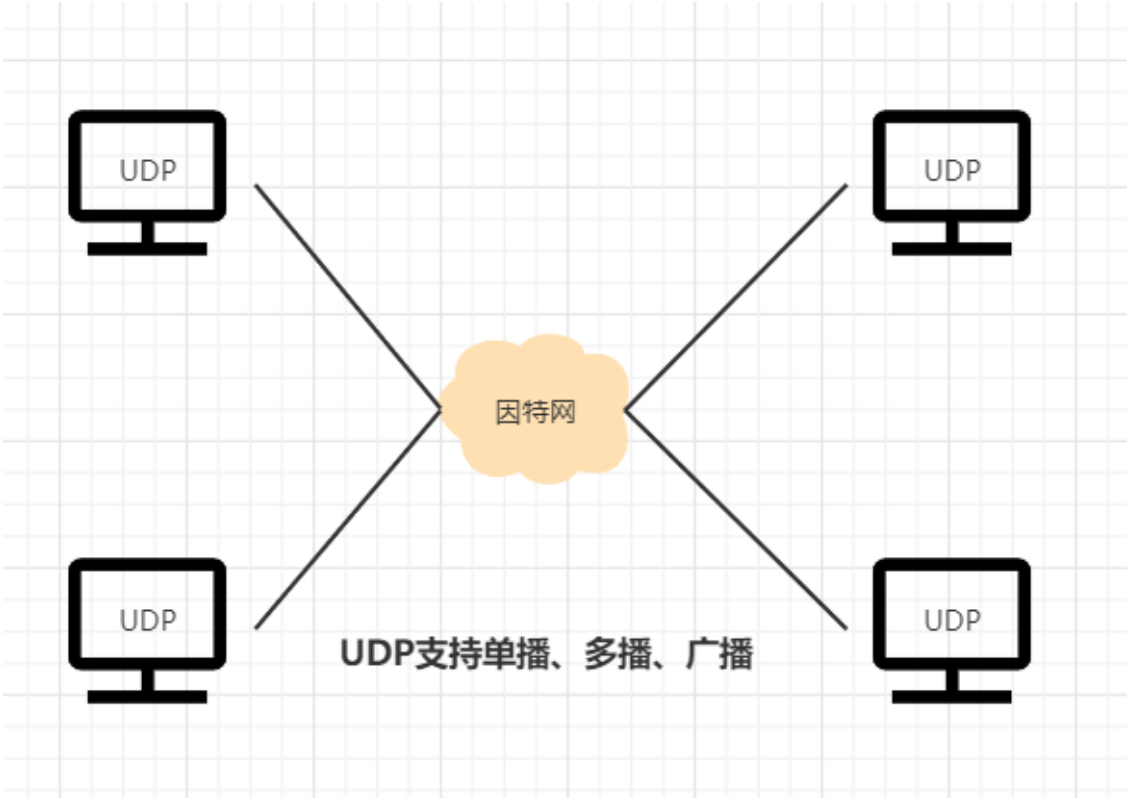
当别人问：到底什么是面向连接，什么是面向无连接呢？回答如下：

- **面向连接（connection-oriented）：面向连接的协议要求正式发送数据之前需要通过「握手」建立一个逻辑连接，结束通信时也是通过有序的四次「挥手」来断开连接。**
- **无连接（connectionless）：无连接的协议则不需要**

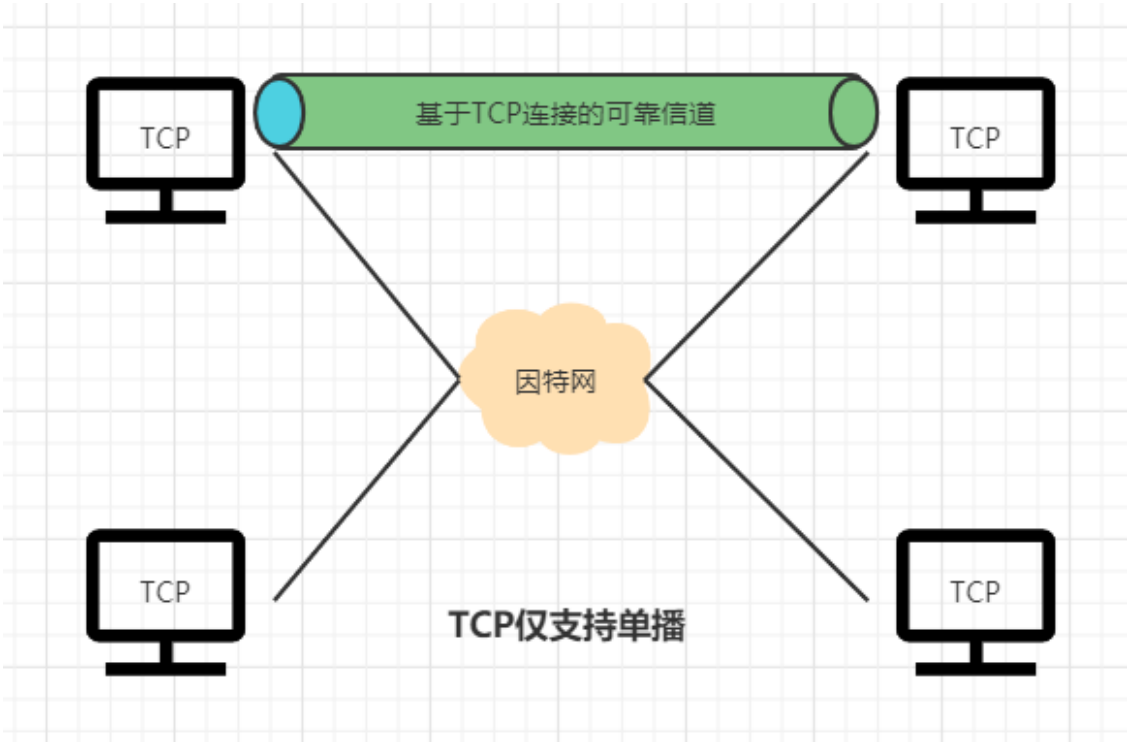
后续我们将深入学习三次握手和四次挥手，来了解其意义和机制。

三、TCP仅支持单播

UDP支持单播、多播和广播，换句话说，UDP支持一对一、一对多以及一对全的通信。



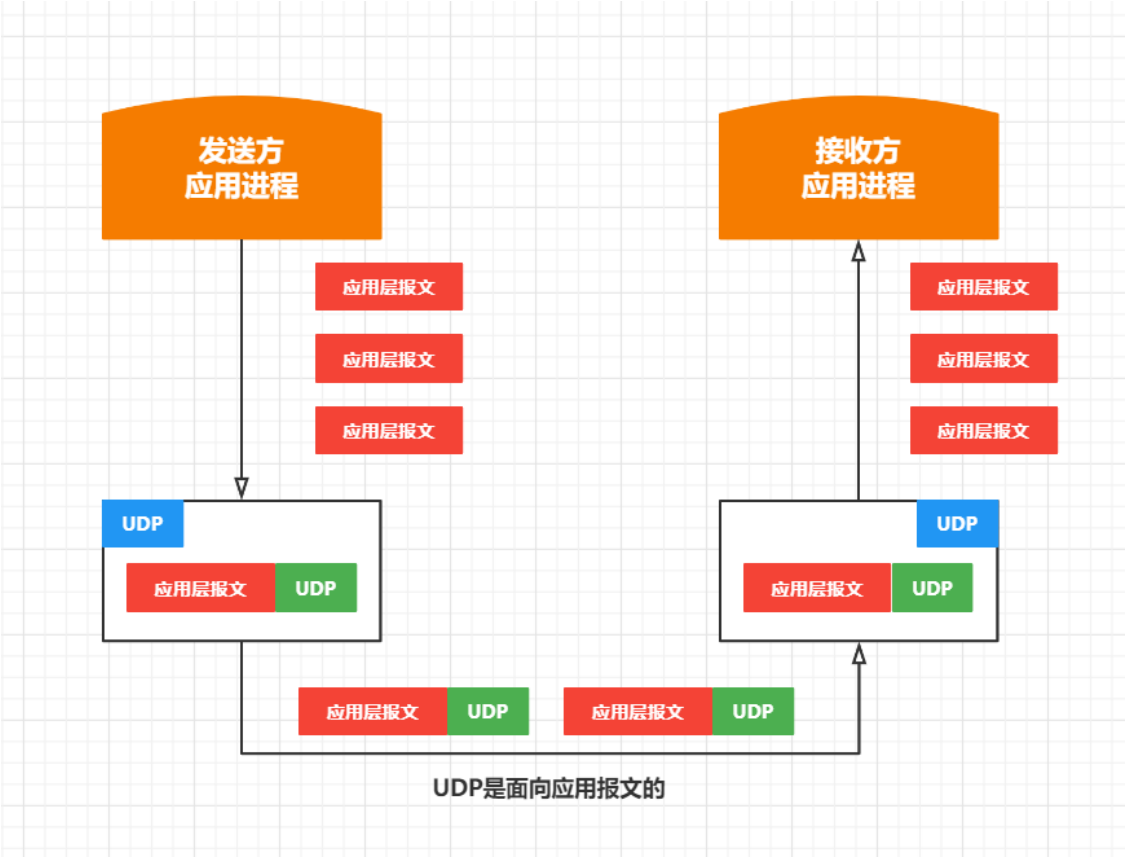
而使用TCP协议的通信双方，在进行数据传输之前，必须使用“三次报文握手”来建立TCP连接，TCP建立成功后，通信双方之间好像建立了一条基于TCP连接的可靠信道，通信双方即可通过这条可靠信道进行通信。很显然，这样做的话，**TCP仅支持单播，即一对一的通信。**



四、面向应用层报文和面向字节流

我们先来看看UDP协议是如何处理应用报文的。

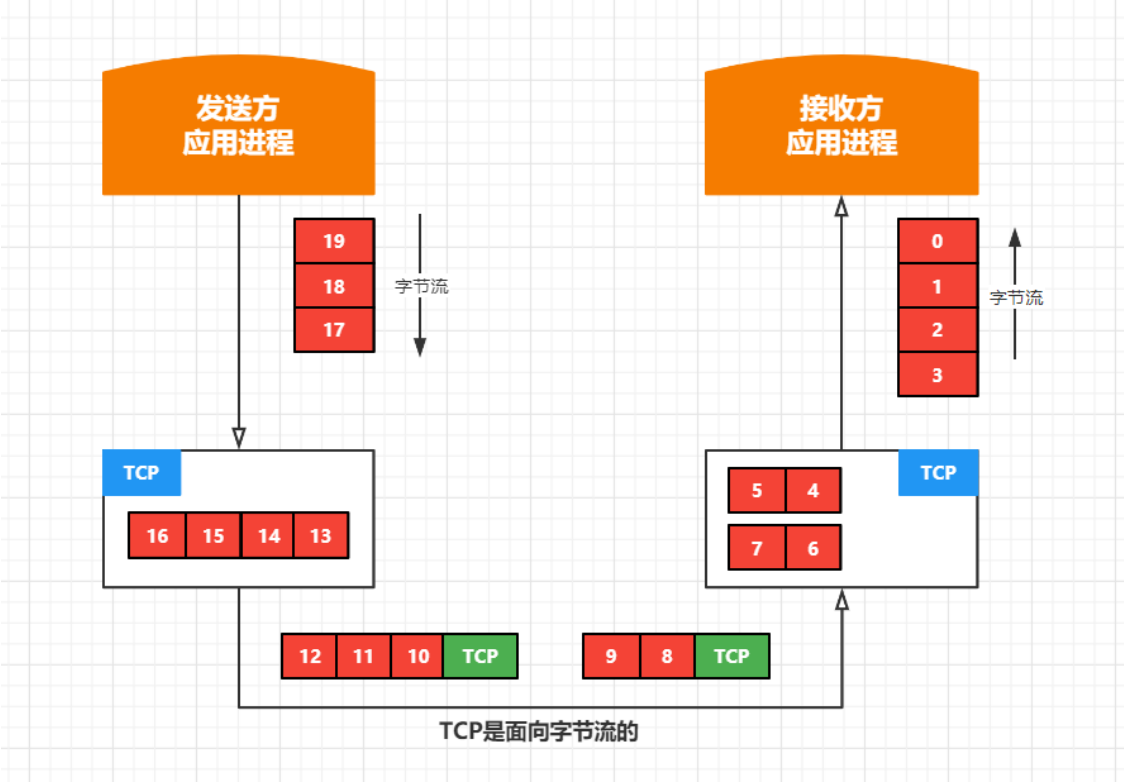
发送方将应用层报文交付给传输层UDP，UDP直接在应用层报文添加一个UDP首部，使之成为UDP数据报。接收方应用进程收到报文段后，直接去除UDP首部即可交付给上层应用，**也就是说，UDP对应用进程交付下来的报文既不合并也不拆分，而是保持报文的边界，仅仅加一个UDP首部即可，换句话说，UDP是面向应用报文的**，整体处理经过简化后如下图所示：



当然了，我们不要忘记数据链路层MTU的限制，如果超出MTU的值，IP层也会帮助我们做报文的分片。

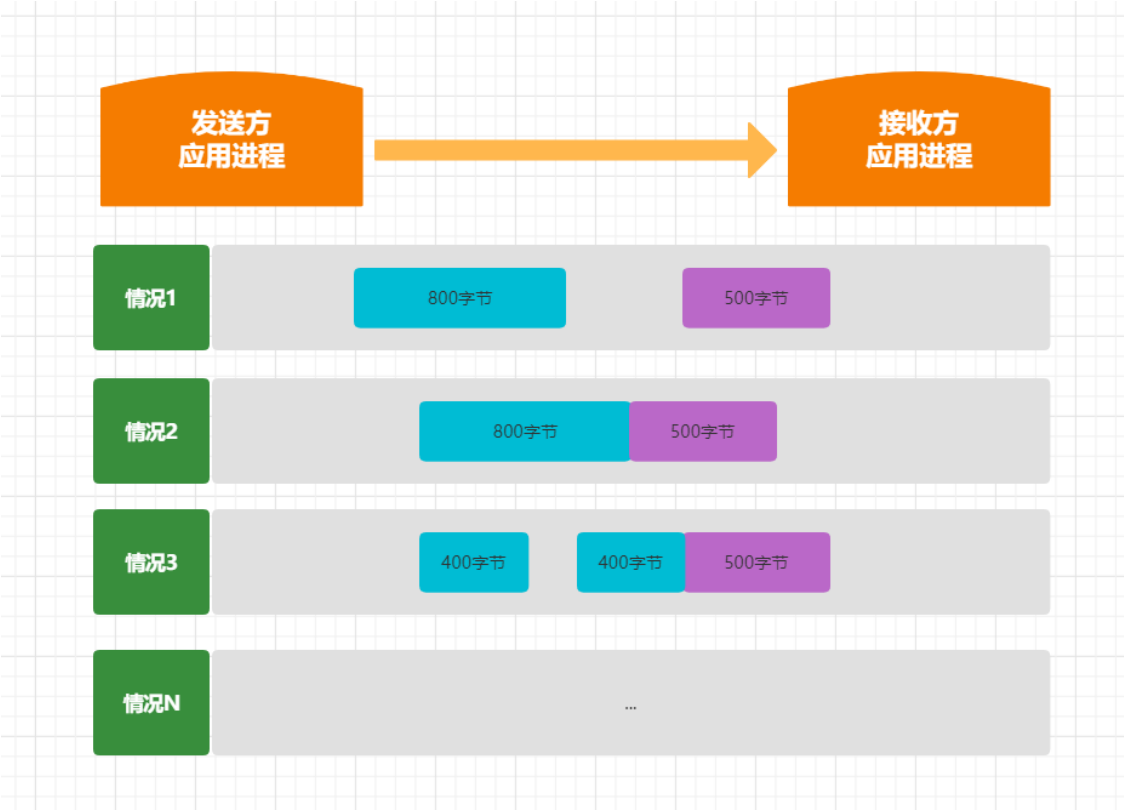
没有对比就没有切身感受，我们来看看TCP协议的情况。

发送方的TCP把应用进程交付下来的数据块仅仅看作是一连串的、无结构的字节流，TCP并不知道这些待传送的字节流的含义，仅将它们编号并存储在发送缓存中。



TCP根据发送策略，从发送缓存中提取一定数量的字节，构建成一个TCP报文段发送出去。

假设此时依次写入500 字节、800 字节，此时仅仅是把字节拷贝到缓存区中，最终发送出去多少是不确定的，比如可以分为两条报文依次发出去500字节和800字节数据；也可以把两部分数据合并为一个长度为 1300 字节的报文，一次发送；也可能第一次发送400个字节，第二次发送900个字节等等各种拆分组合。



上面出现的情况取决于诸多因素：路径最大传输单元MTU、发送窗口大小、拥塞窗口大小等。

接收方的TCP一方面从接收到的TCP报文段中取出数据载荷部分并存储在接收缓存中，一方面将接收缓存中的一些字节交付给应用进程。

当接收方从 TCP 套接字读数据时，它是没法得知对方每次写入的字节是多少的。接收端可能分2 次每次 650 字节读取，也有可能先分三次，一次 100 字节，一次 200 字节，一次 1000 字节进行读取。

可以看出，TCP不保证接收方应用进程所收到的数据块与发送方应用进程所发出的数据块具有对应大小的关系，例如，发送方应用进程交给发送方的TCP共10个数据块，但接收方的TCP可能只用了4个数据块，就把收到的字节流交付给了上层的应用进程。

其实这里说的就是大名鼎鼎的TCP的粘包和拆包问题。

TCP的粘包和拆包问题就是指：在进行TCP通信时，因为TCP是面向流的，所以发送方在传输数据时，可能会将多个小的数据包粘合在一起发送，而接收方则可能将这些数据包拆分为多个小的数据包进行接收，从而导致数据接收出现错误或者数据粘连问题。

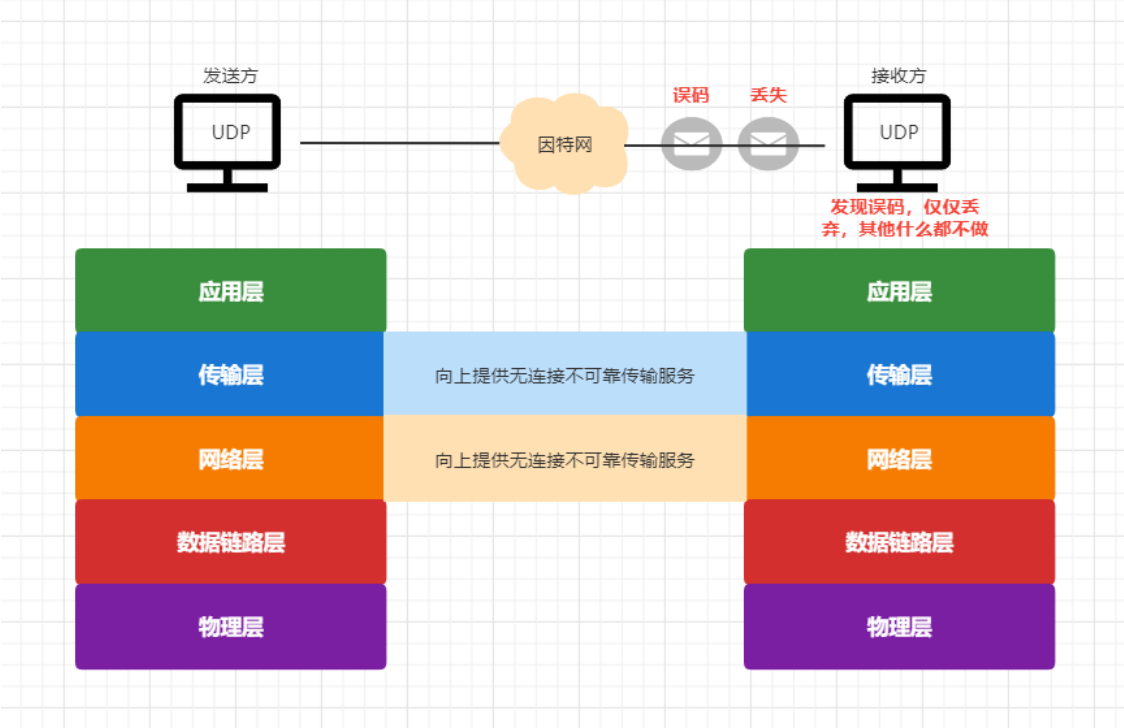
> 对于粘包和拆包问题，TCP的解决方案是将消息分为header和body，在head中保存有当前整个消息的长度，只有在读取到足够长度的消息之后才算是一个完整的消息。其他常见的解决方案还有：
>
> - 将业务层协议包的长度固定下来，每个包都是固定长度，比如512字节大小，如果客户端发送的数据长度不足512字节，则通过补充空格的方式补全到指定长度；
> - 在每个包的末尾使用固定的分隔符，比如换行符，如果一个包被拆分了，则等待下一个包发送过来之后找到其中的换行符，然后对其拆分后的头部部分与前一个包的剩余部分进行合并即可；
> - 不使用TCP协议，自定义协议避免粘包和拆包问题；

当然，接收方应用进程必须有能力识别收到的字节流，把他还原成有意义的应用层数据，TCP面向字节流，是实现可靠传输、流量控制、拥塞控制的基础。

五、可靠和不可靠

我们知道，网络层提供的是一种无连接、不可靠的协议：它尽最大可能将数据报从发送者传输给接收者，但并不保证包到达的顺序会与它们被传输的顺序一致，也不保证包是否重复，甚至都不保证包是否会达到接收者。

而UDP同样是一种向上提供无连接、不可靠的传输服务。



发送方给接收方发送UDP数据报，若传输过程中用户数据报受到干扰而产生误码，接收方UDP可以通过该数据报首部中的校验和字段的值，检查出产生误码的情况，**然后仅仅丢弃数据报，其他什么也不做**！

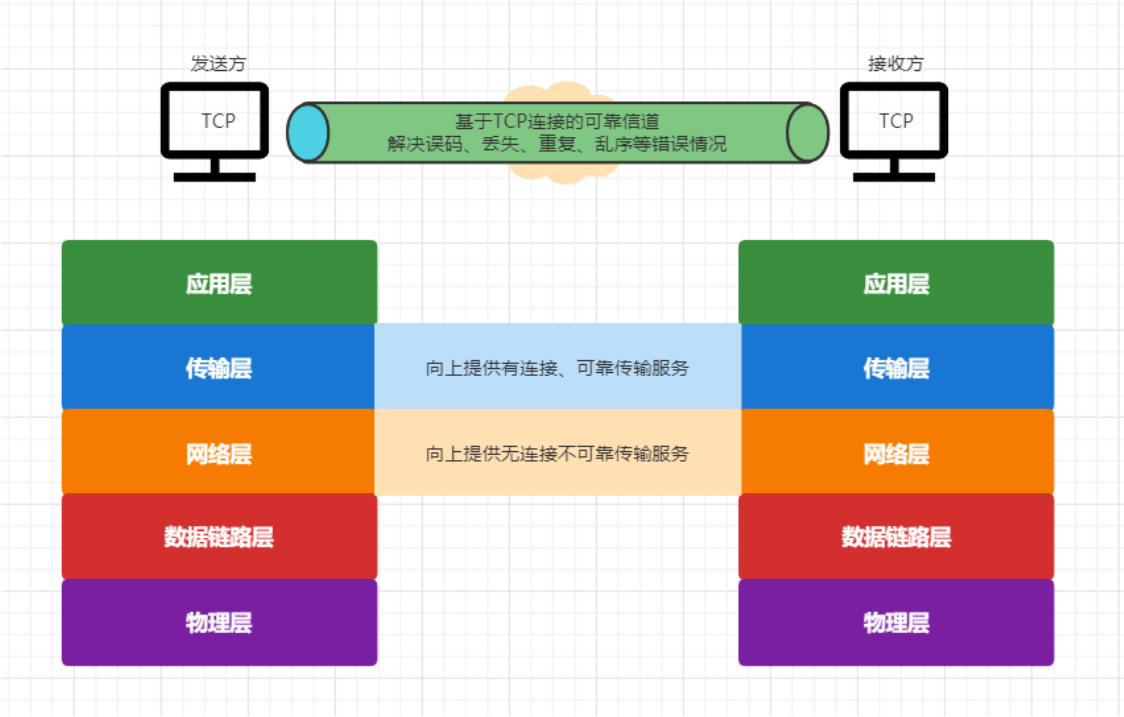
发送方给接收方发送UDP数据报，如果该数据报被因特网中的某个路由器丢弃了，发送方UDP不做任何处理。

因为UDP向上提供无连接、不可靠的传输服务，因此对于出现误码、丢失等问题，UDP并不关心。

基于UDP这个特点，UDP适用于实时应用，比如IP电话、视频会议等，这类场景可以容忍丢失一些数据报或小部分数据报有误码，就尽可能最大限度地传输数据报即可。

而在TCP中，这种情况得到解决，TCP向上提供有连接、可靠的传输服务，保障不会出现传输差错，或者说在出现差错时得到解决，**保证发送方的所有报文都能正确到达接收方**，而不是像UDP那样容忍误码或丢失等情况。

我们可以想象为建立TCP的双方之间有一根可靠信道：



因此TCP适用于要求可靠传输的应用，例如文件传输。总体来讲，TCP要想在IP基础上构建可靠的传输层协议，必须有一个复杂的机制来保障可靠性，主要有下面几个方面：

- 对每个包提供校验和
- 包的序列号解决了接收数据的乱序、重复问题
- 超时重传
- 流量控制、拥塞控制

这些实际上在之前的文章《10｜数据链路层篇：可靠传输问题（上）》和《11｜数据链路层篇：可靠传输问题（下）》探讨可靠传输问题时都有说明过，后续将结合TCP再来详细说明TCP的流量控制、拥塞控制、超时重传、可靠传输的具体实现机制。

六、报文首部有差别

最后，来看看UDP和TCP报文首部的区别。

UDP由于机制十分简单，因此首部构成也非常简单，包括源端口号、目的端口号、数据报总长度、校验和。

源端口号	目标端口号	总长度	Checksum	要发送的信息

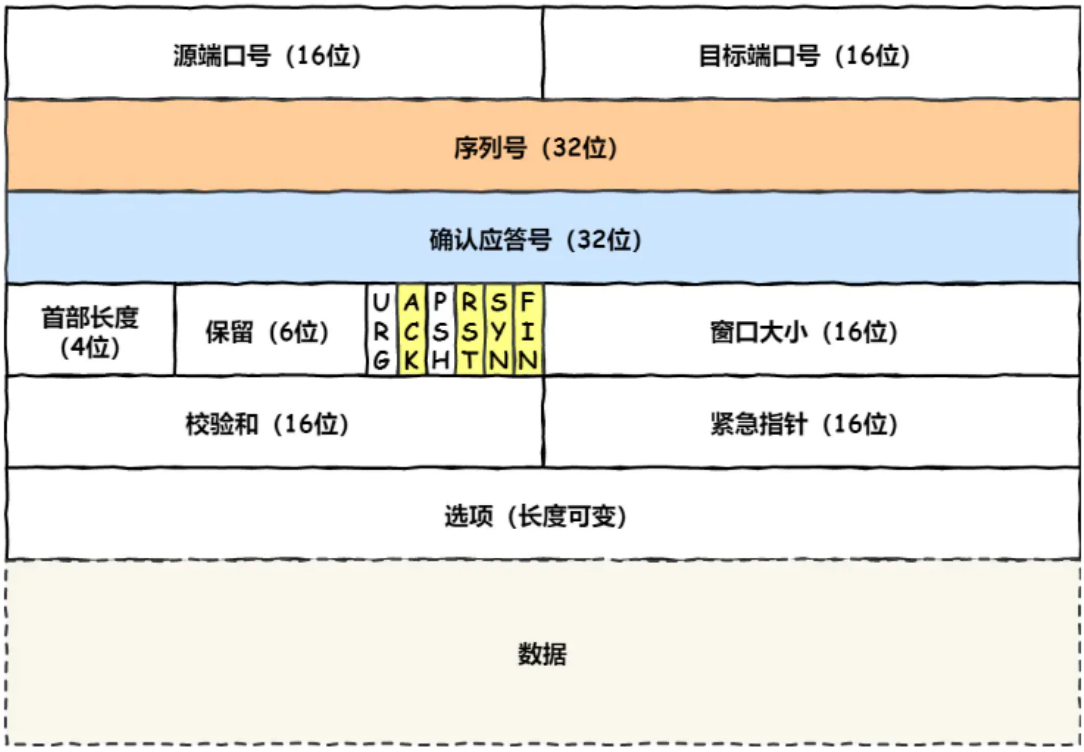
每个区域都占 2 个字节，所以头部只需要 8 个字节。这是我们迄今为止看到的最小的头部了，以后应该也看不到比这更小的头部了。

- 源端口号（Source Port）：这很简单，它是发送信息的应用程序的地址。
- 目标端口号（Destination Port）：也很简单，它是接收信息的应用程序的地址。
- 数据报的总长度（单位是 Byte）：此区域占 2 字节，这意味着一个数据报的最大长度是 2 的 16 次方（等于 65536）个字节。但现实中，很少见到大于 512 字节的 UDP 数据报。这主要是因为丢失一个较小的数据报是可以接受的，但是丢失较大的数据报则比较麻烦，因为 UDP 对于丢失数据包置若罔闻。
- Checksum：表示“校验和”，其原理与 OSI 第 2 层的以太网帧里的 CRC（循环冗余校验）类似，也是为了确保发送的数据和接收的数据是相同的。也许有些教材将这里的 Checksum 写成 CRC。CRC 和 Checksum 都属于冗余校验（Redundancy Check）。Checksum 是最简单的冗余校验。

> 对了，既然 OSI 第 2 层以太网协议已经有一个 CRC 校验了，为什么 UDP 协议还需要有一个 Checksum 校验呢?
>
> 原因是，OSI每一层都是独立的，校验了第二层并不意味着不需要校验第四层，第三层也同理。因为每一层都不应该知道另一层执行了一个校验，每个层应该实现自己的校验。此外，有时数据报到达第 4 层时，会带着从第 3 层跨越到第 4 层时产生的错误。因此，使用冗余校验是非常必要的。

而TCP的首部就相对来说比较复杂了，因为TCP要实现可靠传输、流量控制、拥塞控制等服务，所以字段也很多，其首部最小20字节，最大60字节，需要单独一个章节详细剖析其字段含义，这里不展开说明。

TCP 头部格式



七、总结

本文主要是对比了UDP协议和TCP协议，明白UDP是一个无连接、不可靠的协议，拼命传输上层应用报文达到最大传输速度即可，适合于实时应用，比如IP电话、视频会议等知识即可。

而TCP是实现了面向连接、可靠的传输服务，需要支持可靠传输、流量控制、拥塞控制等服务，因此TCP比较复杂，也是传输层中需要花费很多精力去学习和探讨的对象，后续文章将对TCP进行详细的学习。

最后用一张图来总结TCP和UDP对比：

UDP和TCP对比	UDP	TCP
是否面向连接	无连接	面向连接
单播、多播、广播的支持	支持一对一、一对多、多以一和多对多的交互通信	每一条TCP连接只能有两个端点，只能是一对一通信
面向报文还是字节流	对应用层交付的报文直接打包	面向字节流
是否可靠	尽最大努力交付，不可靠，不适用流量控制和拥塞控制	可靠传输，使用流量控制和拥塞控制
首部字段	首部开销小，仅8个字节	首部最小20字节，最大60字节