39 TCP 与 UDP

更新时间：2020-10-14 10:58:25

> 构成我们学习最大障碍的是已知的东西，而不是未知的东西。 —— 贝尔纳

计算机网络知识本身是庞大而高深的，不过对于前端面试来说，相关的高频考点确定性较高、且非常稳定。因此针对网络知识，我们的思路就是抓主要矛盾，重点问题重点解决。

TCP-三次握手与四次挥手

三次握手——激动人心的会面

TCP的三次握手和四次挥手，就像小情侣之间从见面到分手的过程一样，很腻歪，但是很有必要。

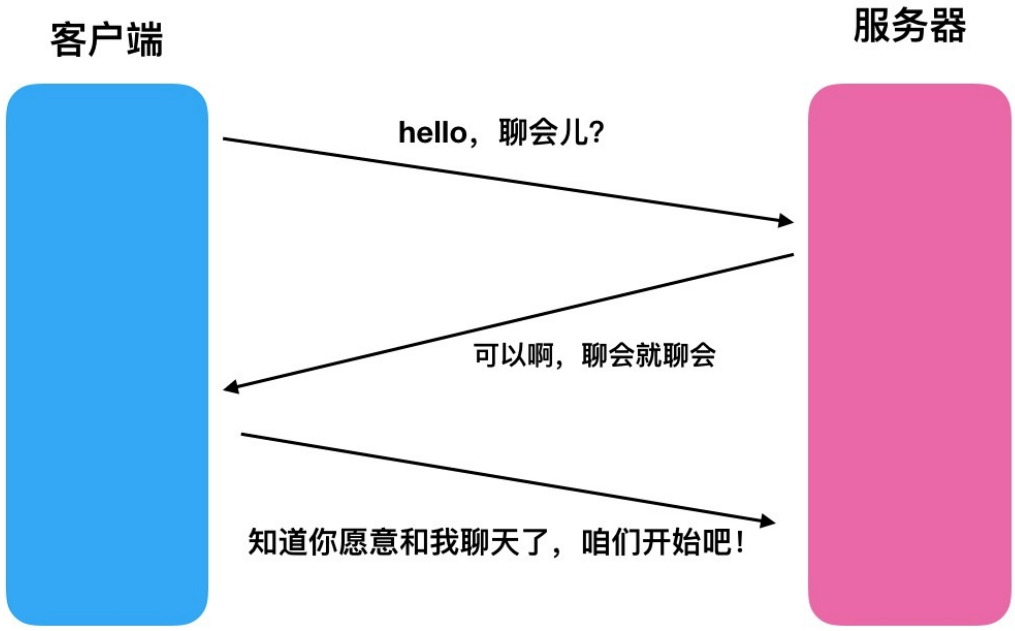
先说说三次握手。参与三次握手的两个主体分别是客户端和服务器。客户端大概率是个男孩子，比较主动，最初是由它来发起沟通，问道：

“服务器你好，我可以和你聊聊天吗？”

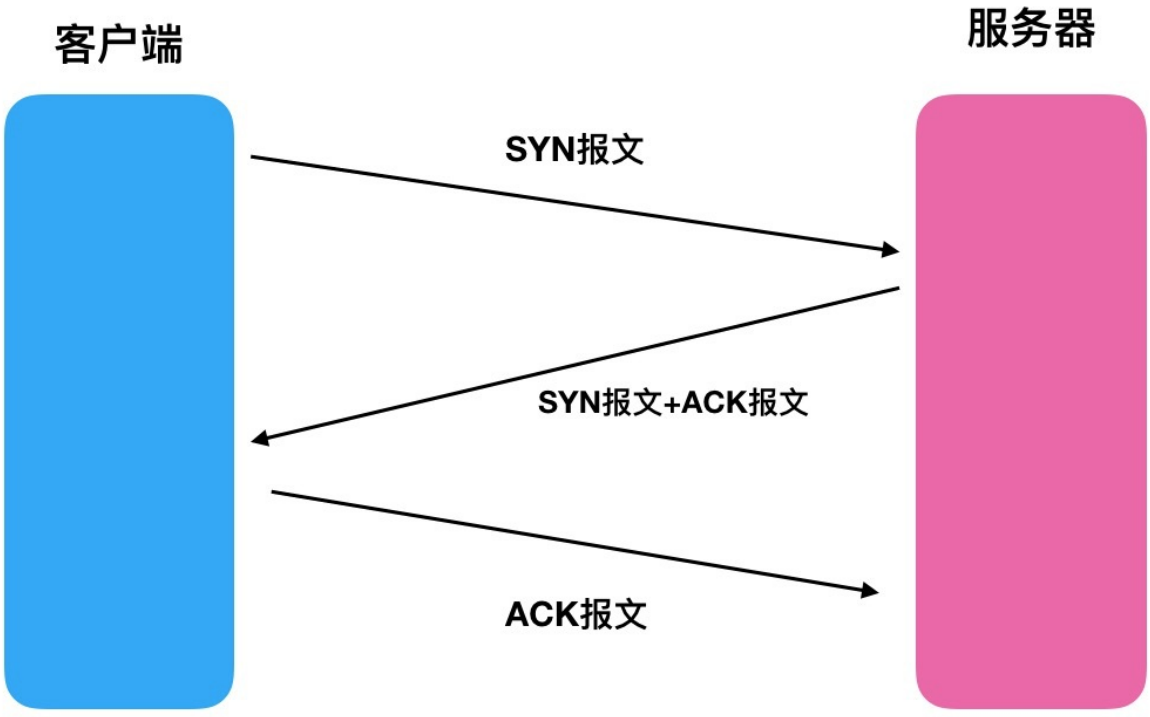
服务器如果觉得客户端这个请求没问题，就会抛出一个积极的回应：“可以呀，让我们来聊会天吧。”

客户端收到肯定的回复，开心得不行，赶紧说：“好的好的，我知道你答应我了，咱们现在可以开始聊天了！”

整个相亲过程如下图所示：



对应到实际的 TCP 连接建立过程，你会发现差异其实并不大，只需要把人类语言转换为计算机语言即可：



首先，大家要知道，在计算机的世界里，客户端和服务器通过“报文”这个东西互通心意。此处，登场的是 SYN 报文和 ACK 报文：

- SYN 报文： 起标识作用的家伙。当SYN=1而ACK=0时，表明这是一个连接请求报文。对方若同意建立连接，则应在响应报文中使SYN=1和ACK=1. 因此, SYN 置1就表示这是一个连接请求或连接接受报文。
- ACK 报文： TCP协议规定，只有ACK=1时有效，也规定连接建立后所有发送的报文的ACK必须为1。

计算机版的“相亲过程”如下：

由客户端发出请求连接，报文情况是： SYN=1，ACK=0，seq=x。

按照规定，SYN=1时不能携带数据，但要消耗一个序号，所以和 SYN 一起抵达战场的还有一个记录序号的 seq，其值为 x。

然后服务端进行回复确认，报文内容是：SYN=1，ACK=1，同时还有服务端为自己初始化的序号 seq=y, 以及确认号 ack=x+1,

再然后客户端再进行一次确认，这一步用不到 SYN 了，报文内容是：ACK=1, seq=x+1, ack=y+1。

如果你记不住 SYN、ACK 以及序列号变化的细节，那么也不必死磕。在面试的过程中，只要你能说出这个完整的过程，分数就能拿个八九不离十。更重要的，是要答好下面这个问题：

为什么 TCP 一定要握手三次？

这是一个比较经典的问题。按照我们常规的逻辑来理解，人与人之间建立联系，只需要两个步骤：

- A 向 B 发出邀请
- B 接受

为什么到了计算机世界里就变成了三步呢？

这是因为计算机和人不同，计算机之间还存在着一个叫“网络状态”的东西，这货很有可能会影响两台计算机的沟通。通过三次握手，客户端经历了一次请求和一次响应，服务端也经历了一次请求和一次响应，这时一方面确认了当前网络状态不错，另一方面又确认了自己这个沟通对象既能请求又能响应、确实没毛病。只有在这样安全、稳定的前提下，两台计算机之间才可以建立起 TCP 连接。

四次挥手——难舍难分的告别

话说客户端和服务器两人聊得正开心，突然客户端的手机响了——原来是他妈妈喊他回家吃饭。客户端不得不向服务端提出分别，对话如下：

- 客户端：我妈喊我回家吃饭，先走了
- 服务器：啥？你妈喊你回去你就回去，不跟我玩了是吧？
- 服务器：好，给老娘滚
- 客户端：我滚了

这里需要大家认识的一个新的报文类型就是 **FIN 报文**，它用来释放一个连接。 FIN=1 时，就表示此报文段的发送方的数据已经发送完毕，请求释放运输连接。

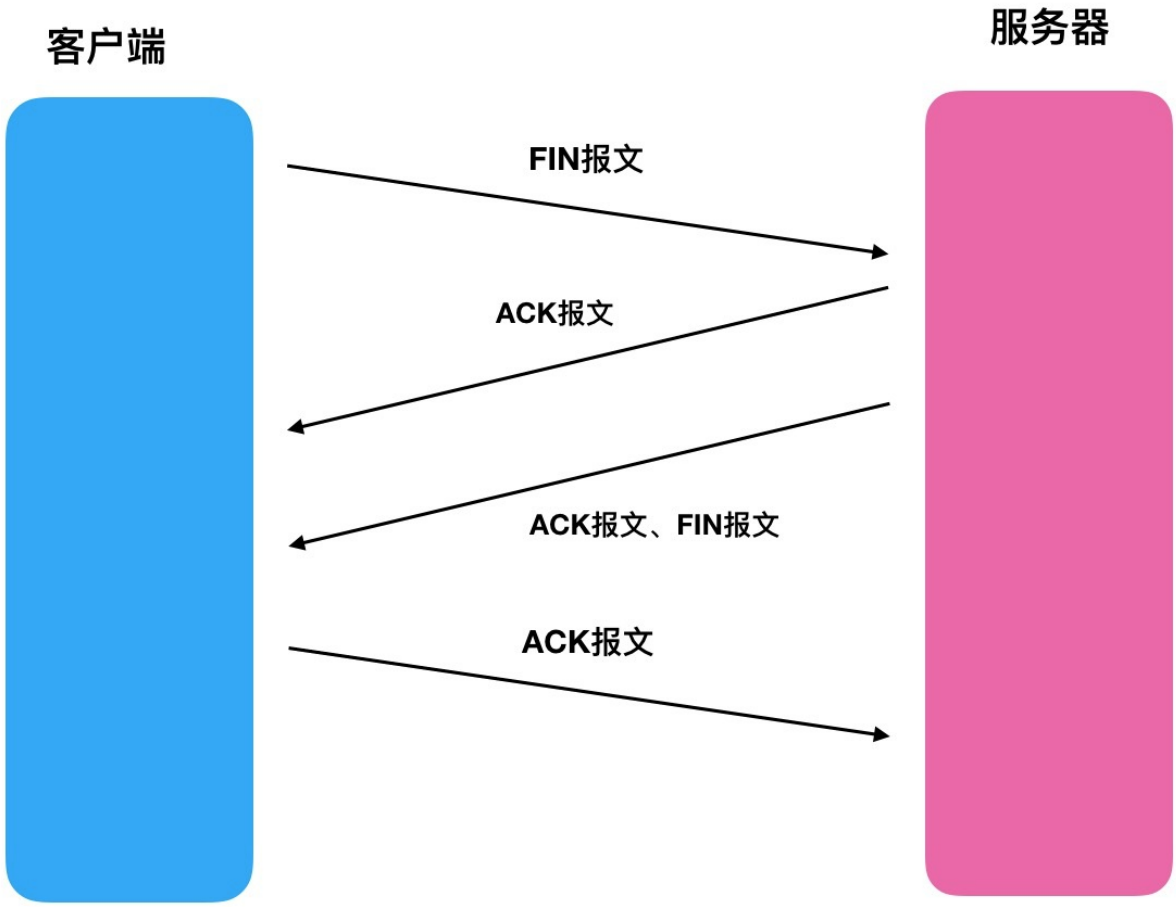
因此在客户端提分手的时候，就会向服务端抛出一个 FIN=1 报文。当然，一起过去的还有序列号 seq=x。

服务器接收到分手报文，痛不欲生，但还是接受了现实，回复了一个 ACK=1 的标识以示确认。当然，一起过去的还有确认码 ack=x+1，以及它自己的序列号 seq=y。

虽然接受了现实，但服务器还有话没说完（在上面的对话中，就是最后那句“滚”了）。作为主动提出分手的一方，客户端自知理亏，所以说会等服务器把想说的说完再走。服务器说完了剩下的话，觉得是时候结束了，于是反过来向客户端抛出一个 FIN=1 报文，请求结束关系。当然啦，一起过去的还有确认标识 ACK=1、确认码 ack=x+1，以及自己的新序列号 seq=z。

客户端收到了服务端的分手请求，求之不得，赶紧说好的好的：向服务端抛出一个 ACK=1 的报文。当然啦，一起过去的还有确认码 ack=z+1，确认标识 ACK=1，以及自己的新序列号 seq=x+1。

同样，这里不需大家强记每个过程中码和标识的细节变化。最重要的是，是掌握下图这个过程，以及四次挥手的必要性：



为什么TCP分手一定要挥手四次？

TCP连接是全双工协议，就是说双方都可以同时向对方发送或接收数据。

当客户端在想要断开连接时，只能确认自己没有数据要传输给服务器了，但并不能确认服务器是否还有数据要发送。

分手嘛，是两个人的事情。客户端不会因为自己没话说了就直接终止关系，而是会等服务器把话说完再走。因此，即便客户端抛出了分手请求，这时服务器还是可以传输数据过来的。

前两次挥手，只是对分手这件事做确认，但并不会立即行分手这件事。

第三次挥手前，服务器会把自己想说的话说完，然后再通知一次客户端。这时，双方才真正都为分开做好了准备。

第四次挥手，客户端接收到了来自服务端的分手请求，响应“接受”的信号，才最后给这一段关系画上了句号。

TCP与UDP的辨析

TCP 协议下，连接的建立需要三次握手，这是为了确保双方能够确实建立起稳定的传输通道。因此，TCP 又被称为“**面向连接的可靠传输**”。

UDP 则恰恰相反，它的世界里没有握手、没有知情同意，是一个非常随性的协议。

在 UDP 协议下，数据想发就发，随时可发。同时，它不关心对方到底有没有收到自己的数据、也不搞什么拥塞控制——就算现在是慢如蜗牛的 2G、3G 网络，它也不会为了保证不丢包来降低自己的速率。

因此，UDP 又被称为是“无连接的不可靠传输”。

UDP 的应用场景

乍一看，UDP 看上去这么不可靠，大家干脆协议选型的时候一股脑都用 TCP 算了，好像压根没有存在的意义呀！

实际上，存在即合理，UDP 虽然冒冒失失让人不放心，但也有它的过人之处，比如：

- 它可以面向多方提供服务：UDP 不止支持一对一的传输方式，同样支持一对多，多对多，多对一的方式。单播、多播、广播它都不在话下。
- 头部开销小：UDP 头部只有 8 个字节大，而 TCP 需要 20 个字节。同样的报文内容，UDP 会比 TCP 更高效。
- 随意也是优点：有时候，我们的连接是需要实时建立的，并没有那么多的资源去给你反复的三次握手和四次挥手，这种场景下，UDP “想发就发”反而更显灵活。

结合 UDP 的过人之处，相信大家不难想象它的应用场景：在一些对实时性要求比较强的场合，比如网络电话、视频会议、在线直播这些地方，UDP 比 TCP 更加合适。而像文件传输这样对可靠性和稳定性要求比较高、同时本身连接的确定性也比较强的需求，用 TCP 来解决会更稳妥。

}