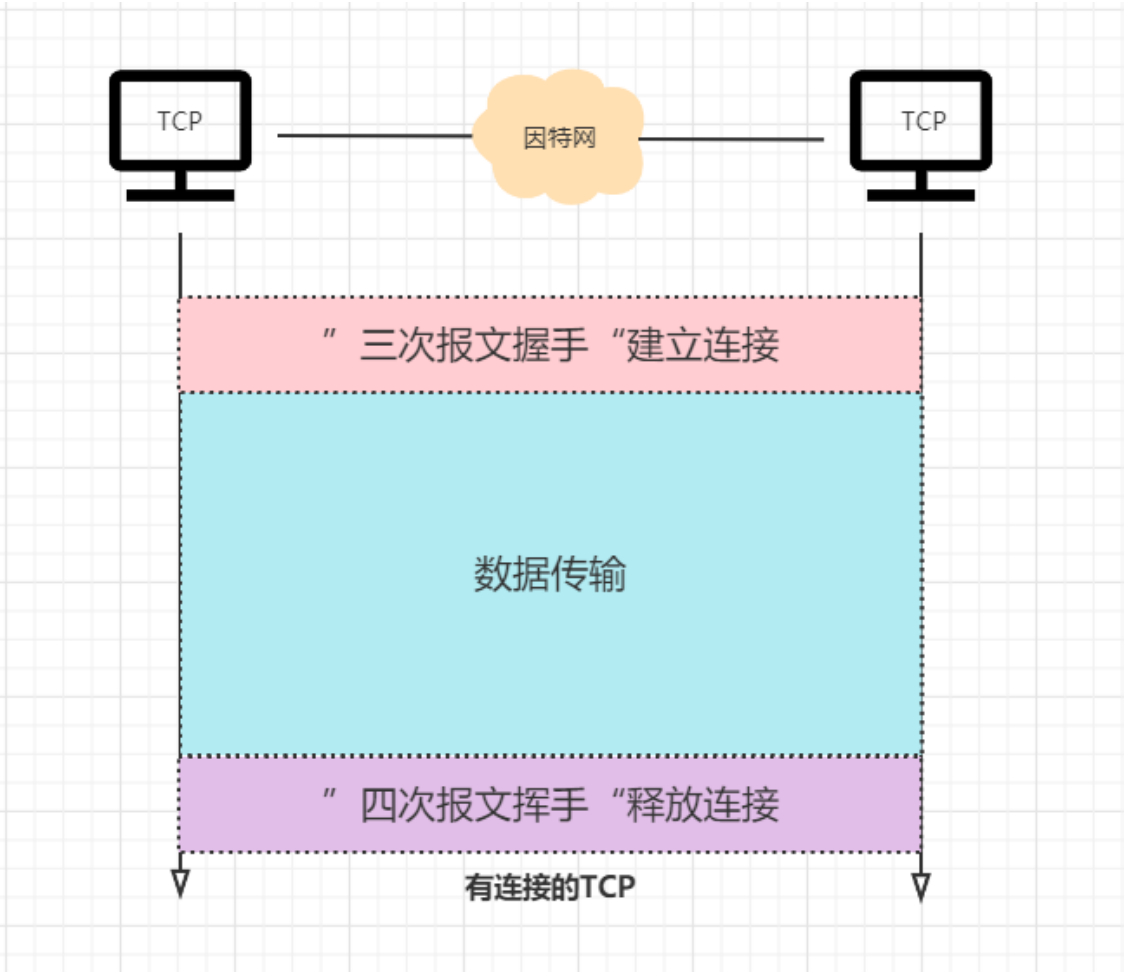
使用TCP协议进行通信的双方主机，在进行数据传输之前，必须使用“三次报文握手”来建立TCP连接，TCP连接建立成功后才能进行数据传输，数据传输完成后必须使用“四次报文挥手”来释放TCP连接。



本节来看看TCP的三次握手过程。

一、交流之前先建立连接

在 TCP 的世界里，对于发送的数据进行确认是它的最大特色，这样才能保证你发的我真的收到了，这种确认从第一步就开始了！

我们之前用写信来比喻UDP，用打电话来比喻TCP，假设 fossi 给好兄弟 stephen 打电话：

- fossi 进行拨号，嘟嘟嘟，等待 stephen 接听，接听成功，先喊一声：hello
- stephen 接起电话：hello，请问哪位？
- fossi 说：hello，我是 fossi 啊！
- stephen 说：奥，原来是 fossi 啊，啥事啊？
- fossi 巴拉巴拉...

也就是说，在正式巴拉巴拉之前，有一个建立连接的过程，不然对方有没有在听你说话都不能确定呢！这个过程对于TCP也是一样的道理。我们在发送正式数据之前，我们要先建立连接，就是发“hello”的过程。TCP的过程：

- 你想和我聊天吗?
- 是的，我已经准备好了！
- 好的收到，让我们开始聊天吧。

因此需要某些信息来标识此数据是一个连接请求（对应hello）？响应/确认？（对应好的收到），这个就是由 TCP 头上的标识位来做区分的。

二、TCP中建立连接到底要解决哪些问题?

打电话例子只是一个引子，我们回归到TCP中，实际上TCP的连接建立要解决以下问题：

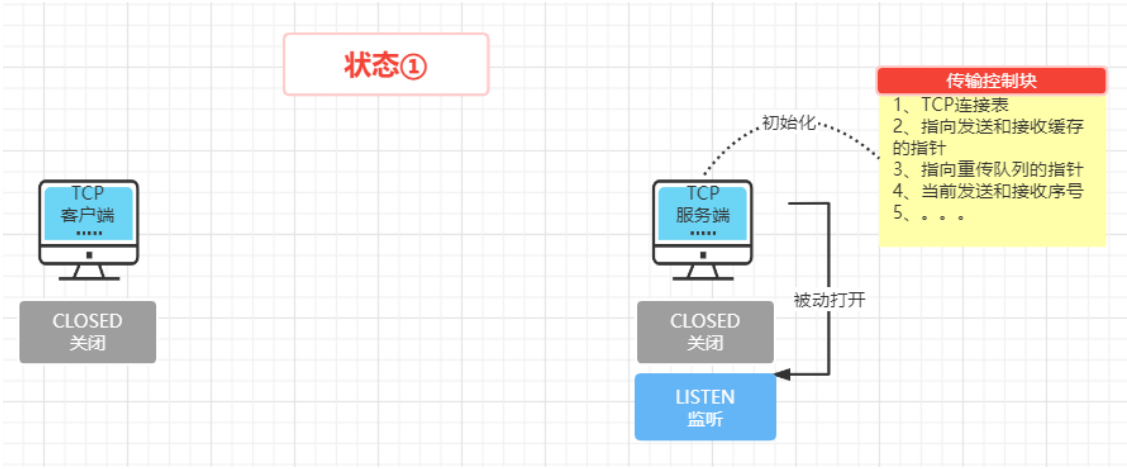
- 使TCP双方能够确知对方的存在；
- 使TCP双方能够协商一些参数，比如交换彼此的初始序列号、最大段大小（MSS）、窗口大小（Win）、窗口缩放因子（WS）、是否支持选择确认（SACK_PERM）等；
- 在交换完基础信息后，TCP双方就可以进行一些资源的初始化工作，比如TCP连接表、指向发送和接收缓存的指针、指向重传队列的指针、当前发送和接收序号等等；

上面所述的MSS、Win、WS等都将在后续章节中讨论到，我们一般会将TCP建立连接比喻为“握手”，而握手需要客户端和服务端交换三个报文段，因此我们一般都称之为三次握手，下面我们来具体看看TCP三次握手过程。

三、三次握手详细过程

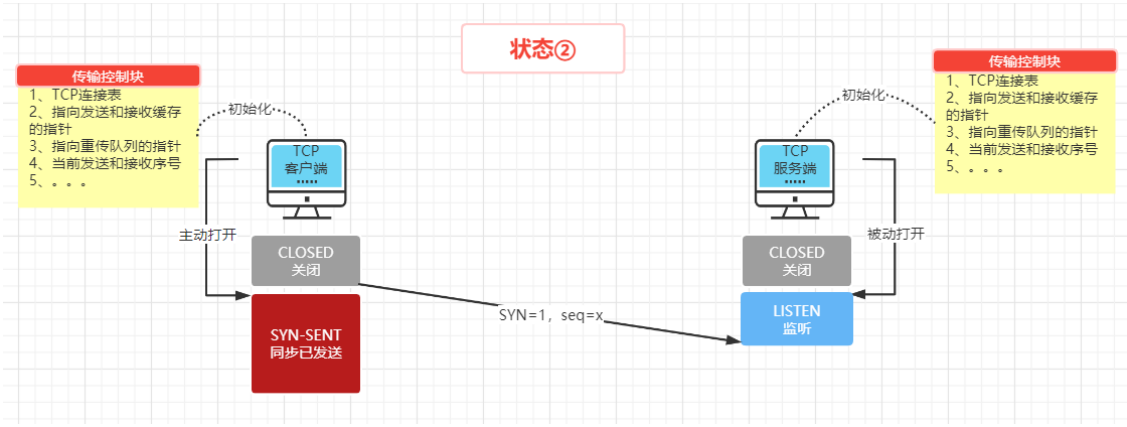
状态①：最初，两端的TCP进程都处于关闭状态，即**CLOSED状态**，CLOSED并不是一个真实的状态，而是一个假想的起点和终点。

由于TCP服务端进程需要提前进入监听状态，被动等待客户端的连接，因此TCP服务器进程首先会创建传输控制块，可以存储：TCP连接表、指向发送和接收缓存的指针、指向重传队列的指针、当前发送和接收序号等信息，之后，**TCP服务器进程就进入监听状态即LISTEN状态，准备接受TCP客户端进程的连接请求。**

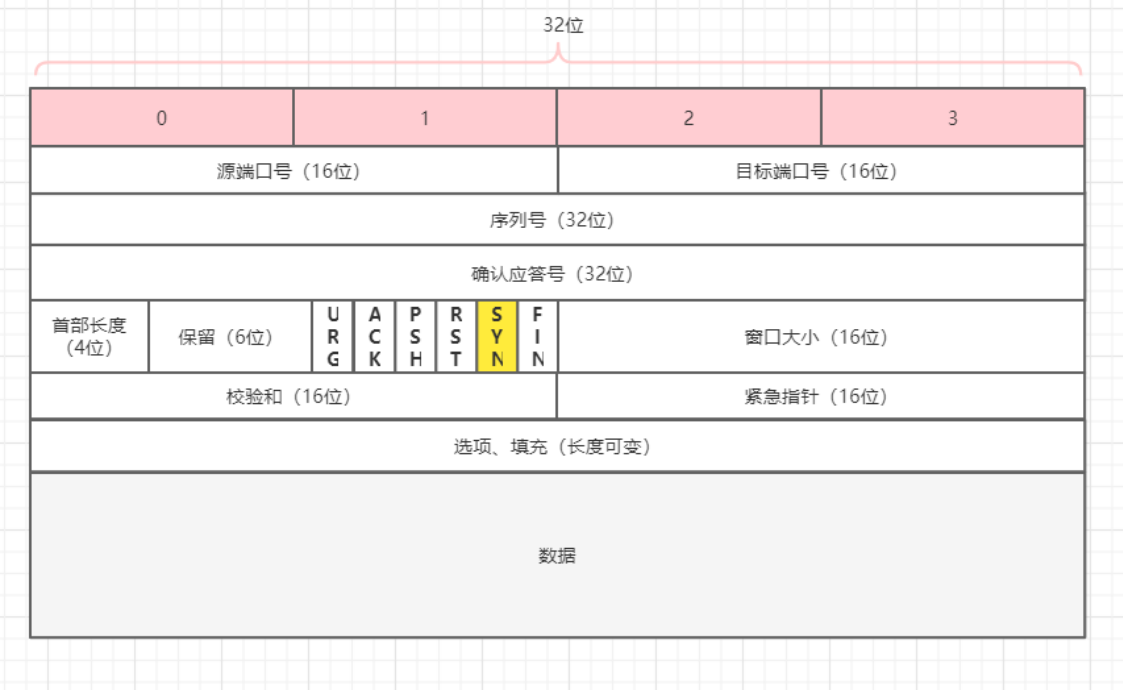


由于TCP服务器进程是被动等待客户端连接请求，而非主动发起，因此这个过程称为被动打开。

状态②：客户端此时要发起连接，首先也是创建传输控制块，然后向TCP服务端发起TCP连接请求报文段，并**进入同步已发送状态**。由于是TCP客户端主动发起连接，因此称之为主动打开。



TCP连接请求报文段首部中的同步位SYN被设置为1，表明这是一个TCP连接请求报文段，序号字段seq被随机设置了一个初始值x，即初始序列号ISN，作为客户端进程所选择的初始序号。



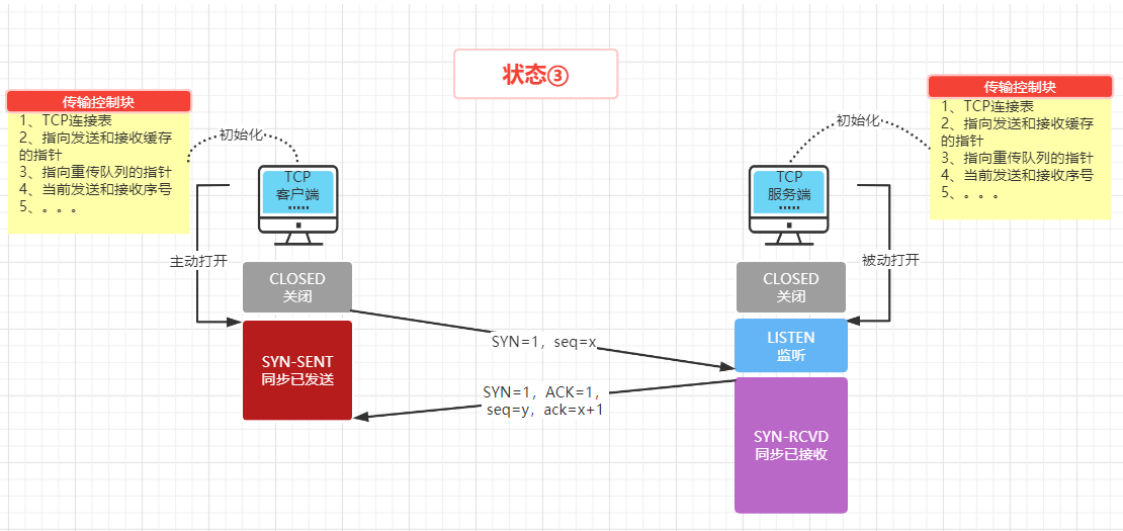
TCP规定SYN报文不携带数据，但是它占用一个序号，下次发送数据序列号要加一。

> 为什么 SYN 段不携带数据却要消耗一个序列号呢?

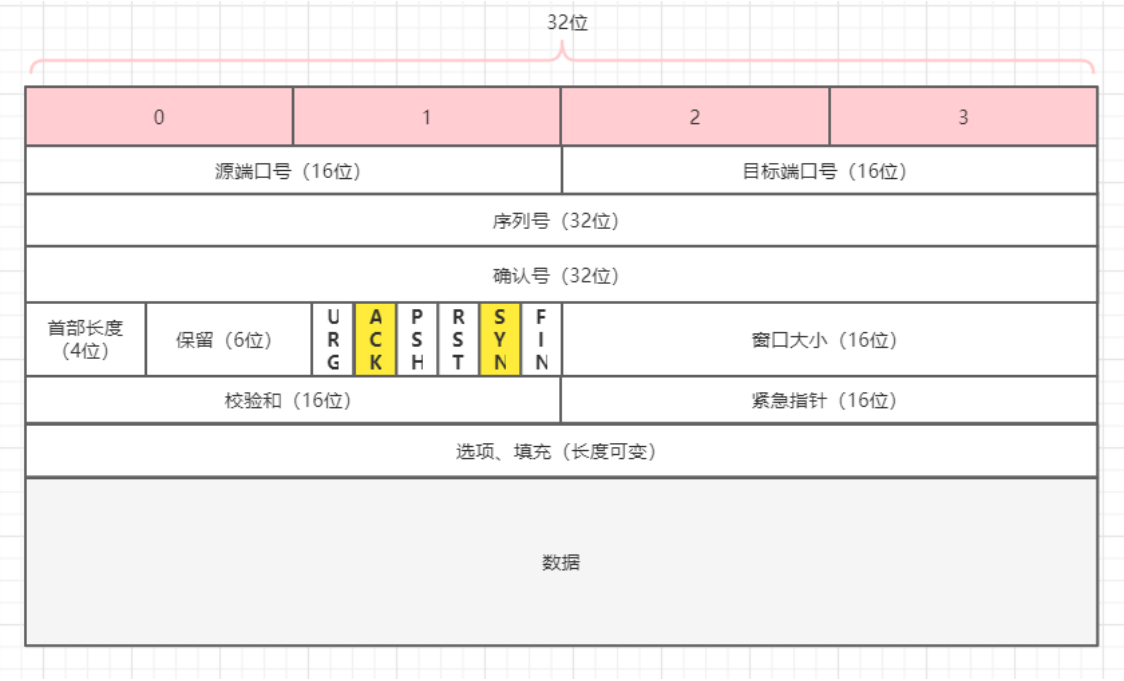
不占用序列号的段是不需要确认的（都没有内容确认个啥），比如 ACK 段。SYN 段需要对方的确认，需要占用一个序列号。

核心原则：凡是消耗序列号的TCP报文段，一定需要对端确认。如果这个段没有收到确认，会一直重传直到达到指定的次数为止。

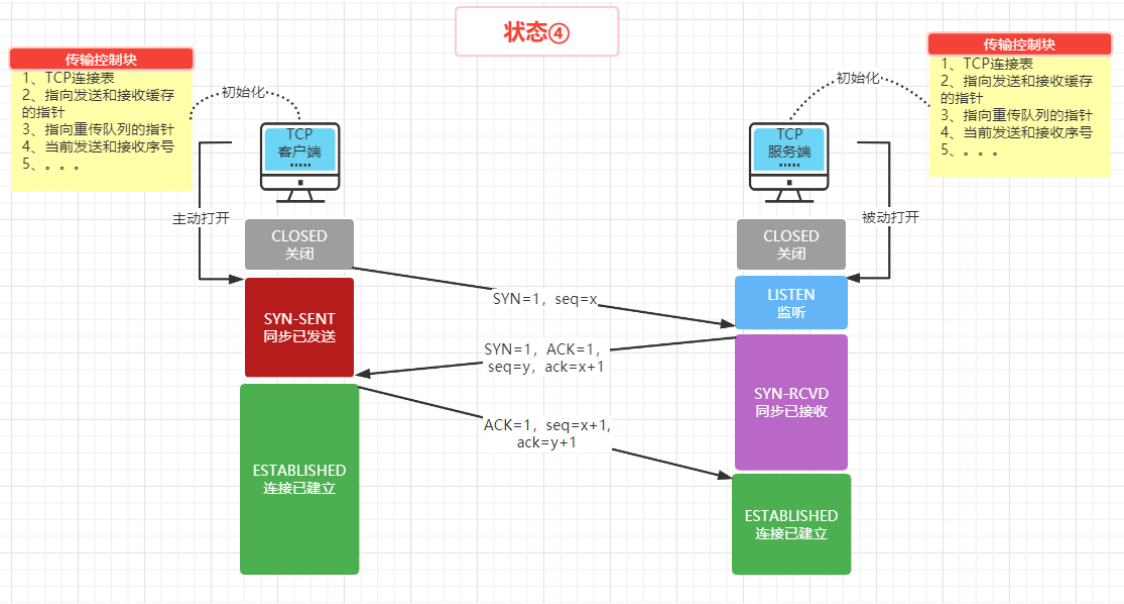
状态③：服务端收到来自客户端的TCP连接请求报文段后，如果同意建立连接，则向TCP客户端发送TCP连接请求确认报文段，并**进入同步已接收状态**。



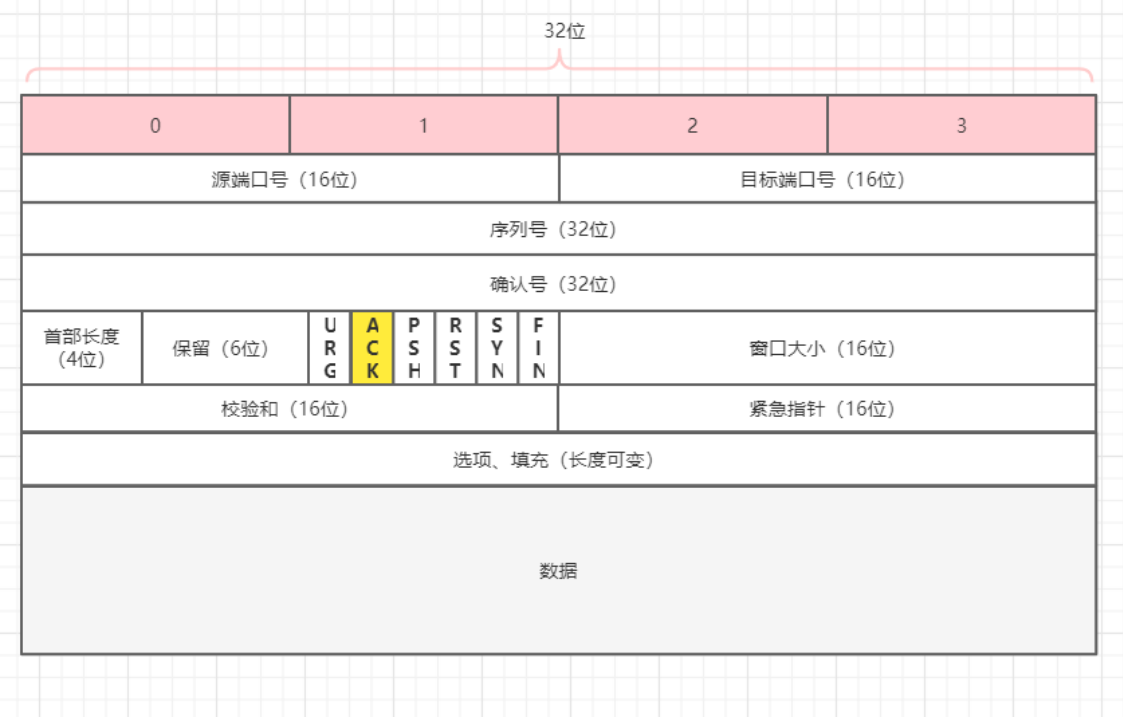
该报文段首部中的SYN和确认位ACK都设置为1，表明这是一个TCP连接请求确认报文段，序号字段seq被设置为一个随机初始值y，作为TCP服务端的初始序列号ISN，确认号字段ack的值设置为x+1，这是对客户端所选择的初始序列号的确认，注意，这个报文段也不携带数据，因为它是一个SYN被设置为1的报文段，不过同样要消耗一个序号，因为还需要客户端的一次确认。



状态④：客户端进程收到服务端进程的TCP连接请求确认报文后，此时**进入连接已建立状态**，此外还要向服务端发送一个普通的TCP确认报文段，服务端收到该ACK报文段后，也会**进入连接已建立状态**。



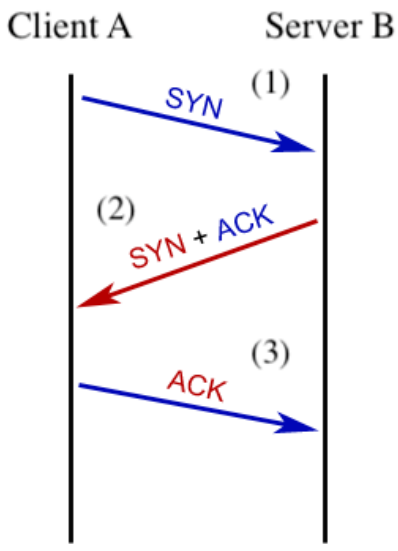
客户端发送三次握手最后一个 ACK 段，这个 ACK 段用来确认收到了服务端发送的 SYN 段。这个ACK段如果不携带任何数据，那么这个ACK段不消耗任何序列号，因为无需再确认；不过这个普通的ACK报文段实际上也可以携带数据，携带数据则会消耗序列号。



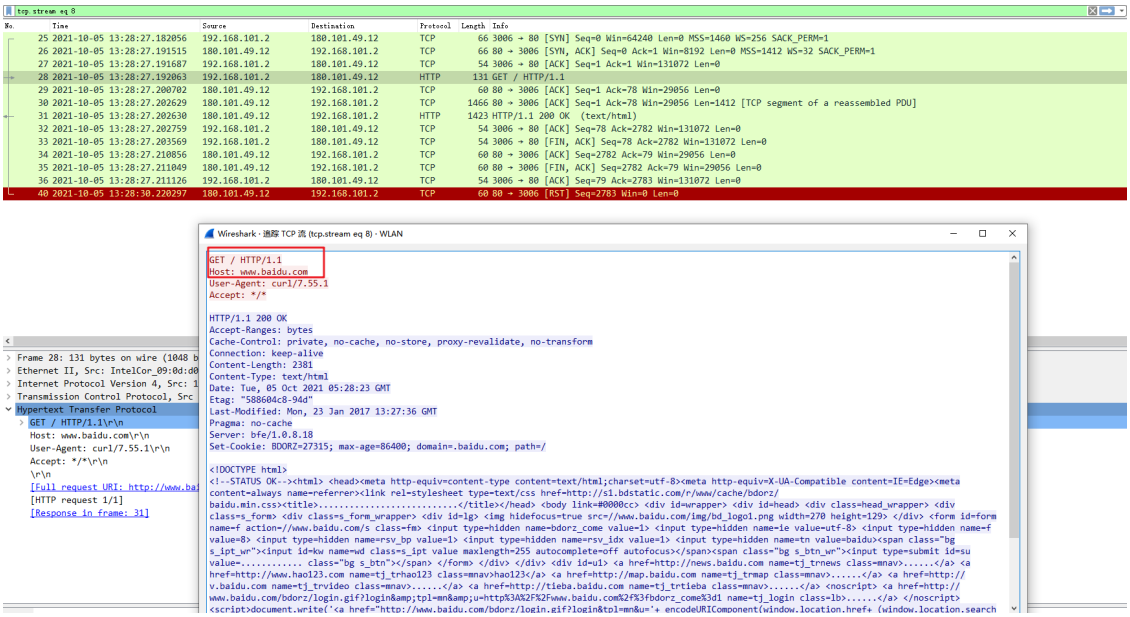
经过一轮的握手后，双方都进入了连接已建立状态，此时就可以基于已建立好的TCP连接进行可靠传输啦。

四、三次握手的实际抓包

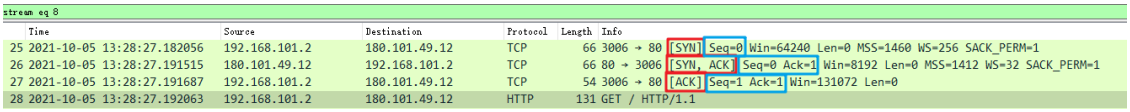
三次握手简化图就是：



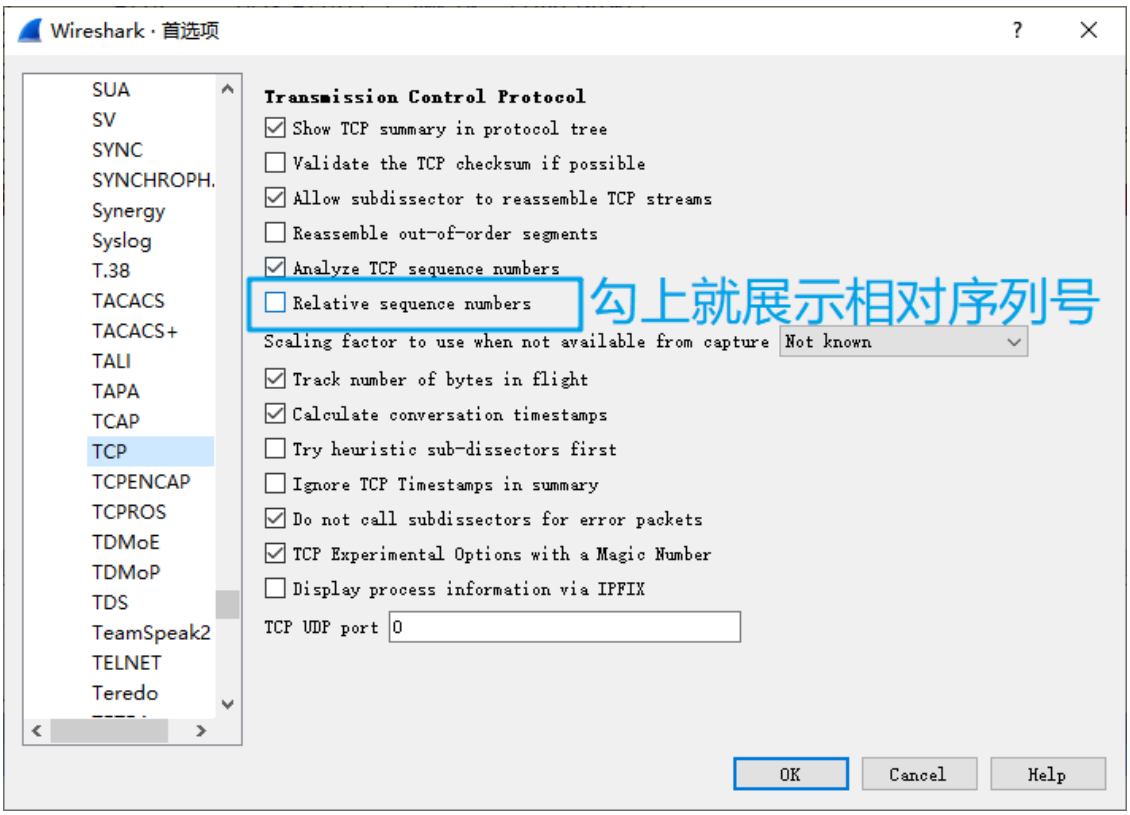
我们打开wireshark并开启抓包，执行命令`curl -v www.baidu.com`：



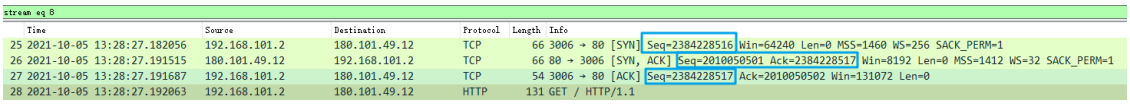
可以看到一开始的三行就是我们本篇所述的三次握手过程：



可以跟以上所学对应起来，当然了，这里的seq序列号被wireshark改为了**相对序列号**，这样更加方便观察，实际上序列号是随机生成的，我们改下设置：在wireshark 的"Edit->Preferences->Protocols->TCP"菜单里可以进行设置显示**绝对序列号**：



下面再来看下绝对序列号的三次握手：



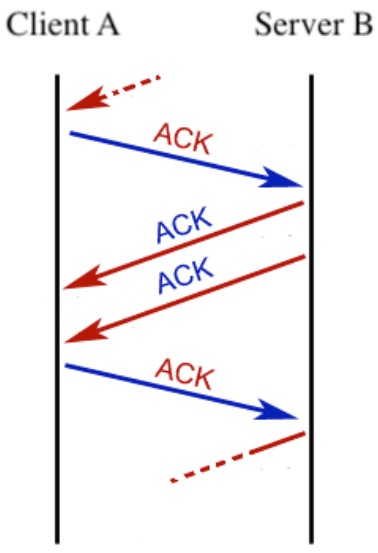
这样子更加符合我们的认识。

如果客户端发了 SYN 数据包，迟迟没有收到服务端的 ACK ，则客户端会进行定时重发多次 SYN 包。多次重试后还是无效，则放弃重试，如果在JAVA语言中会返回 java.net.ConnectException: Connection timed out异常。

若一切正常，双方都处于 ESTABLISHED 状态，此连接就已建立完成，客户端和服务端就可以相互发送数据了。在Linux系统下可以通过 netstat -napt 命令查看 TCP 连接状态：

五、连接的保持

经过 “三次握手” 的过程，双向通信已经建立起来了，应用程序之间可以互传数据包了。在互传数据的过程中，发送的所有数据包上都会设置 ACK 标志，以确认收到了先前的数据包：



当然了，不是每个数据包都要回复 ACK 的，比如客户端发了1，2，3，4，5五个数据包，如果服务端返回的 ACK 是5，说明前面四个数据包也都已成功接收到！