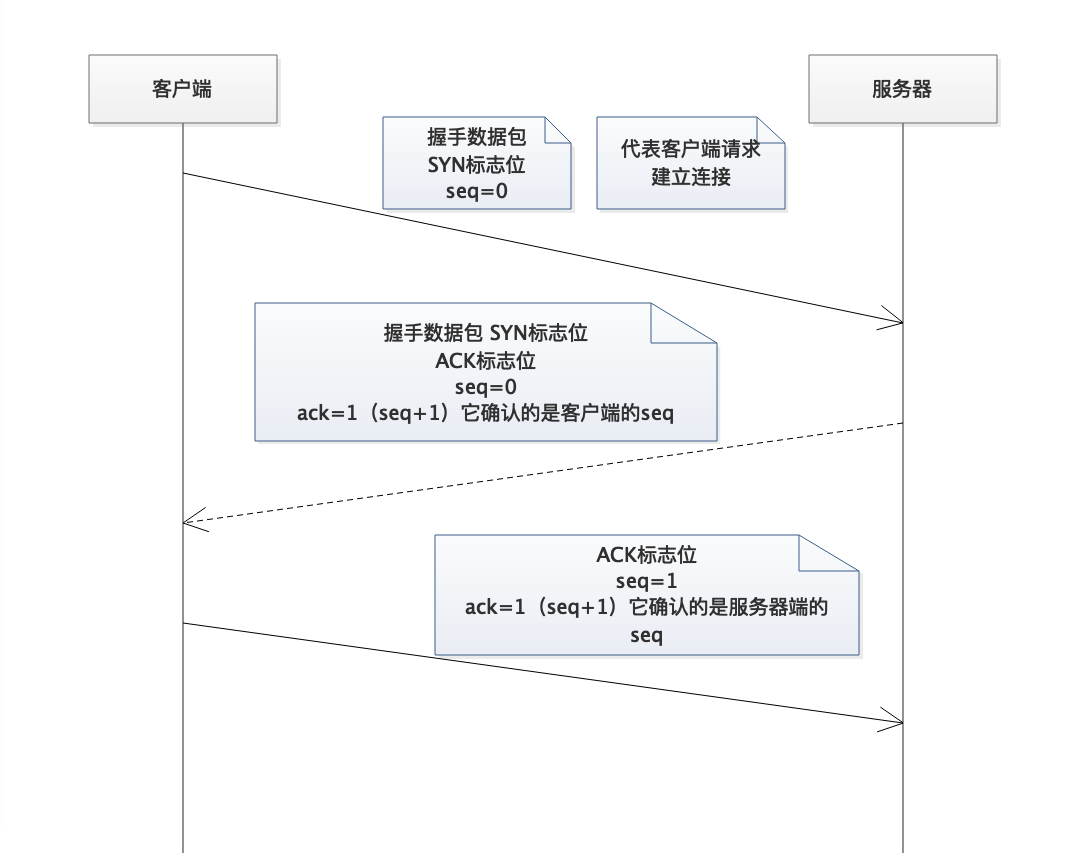
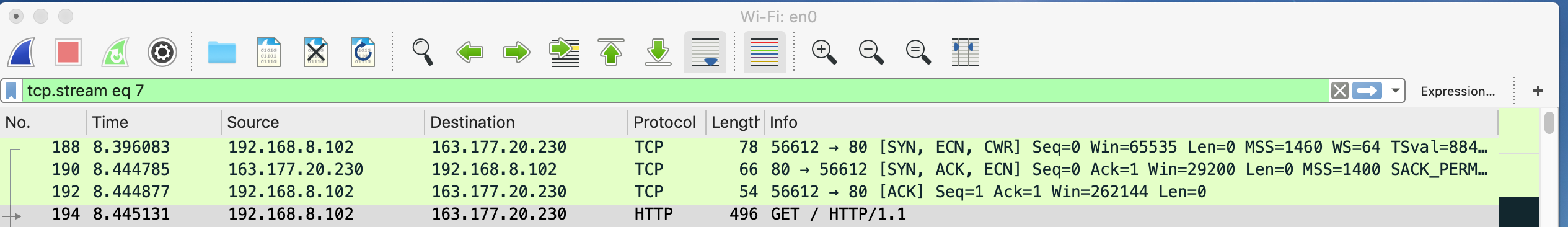
<https://cloud.tencent.com/developer/news/257281>

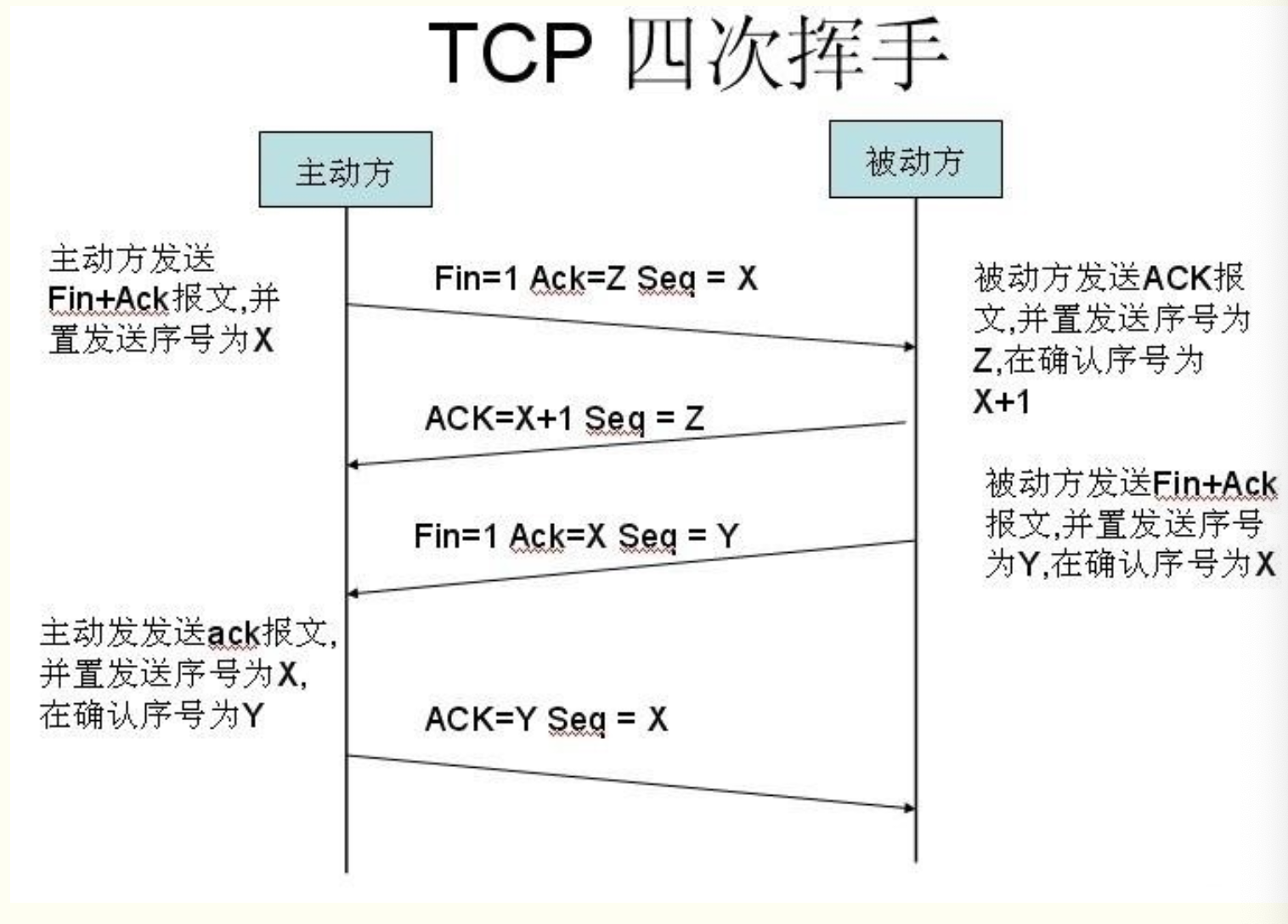
三次握手过程



下面是真实的数据抓包过程



四次挥手过程



我们需要记住的技巧是：

1、确认号(ack)总是对端的序列号+1（每个seq都会有一个确认号来确认收到）

2、每次发送数据都有一个序列号seq(也就是说每发一个tcp包都是需要序列号的)

3、握手挥手过程都有标志位的设置

4、SYN,ACK,FIN标志位组合运用。

这里的FIN=1和SYN=1以及ACK=1表示的是设置相应的标志位。

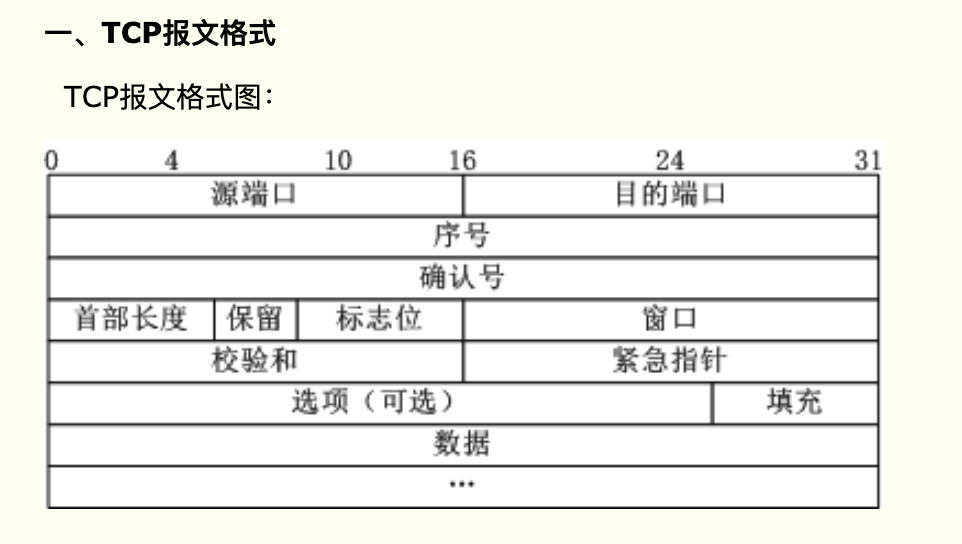
为什么要4次挥手？

确保数据能够完整传输。

当被动方收到主动方的FIN报文通知时，它仅仅表示主动方没有数据再发送给被动方了。

但未必被动方所有的数据都完整的发送给了主动方，所以被动方不会马上关闭SOCKET,它可能还需要发送一些数据给主动方后，

再发送FIN报文给主动方，告诉主动方同意关闭连接，所以这里的ACK报文和FIN报文多数情况下都是分开发送的。



上图中有几个字段需要重点介绍下：

（1）序号：Seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。

（2）确认序号：Ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。

（3）标志位：共6个，即URG、ACK、PSH、RST、SYN、FIN等，具体含义如下：

（A）URG：紧急指针（urgent pointer）有效。

（B）ACK：确认序号有效。

（C）PSH：接收方应该尽快将这个报文交给应用层。

（D）RST：重置连接。

（E）SYN：发起一个新连接。

（F）FIN：释放一个连接。

需要注意的是：

（A）不要将确认序号Ack与标志位中的ACK搞混了。

（B）确认方Ack=发起方Req+1，两端配对。

举例：

第一次握手时

000010

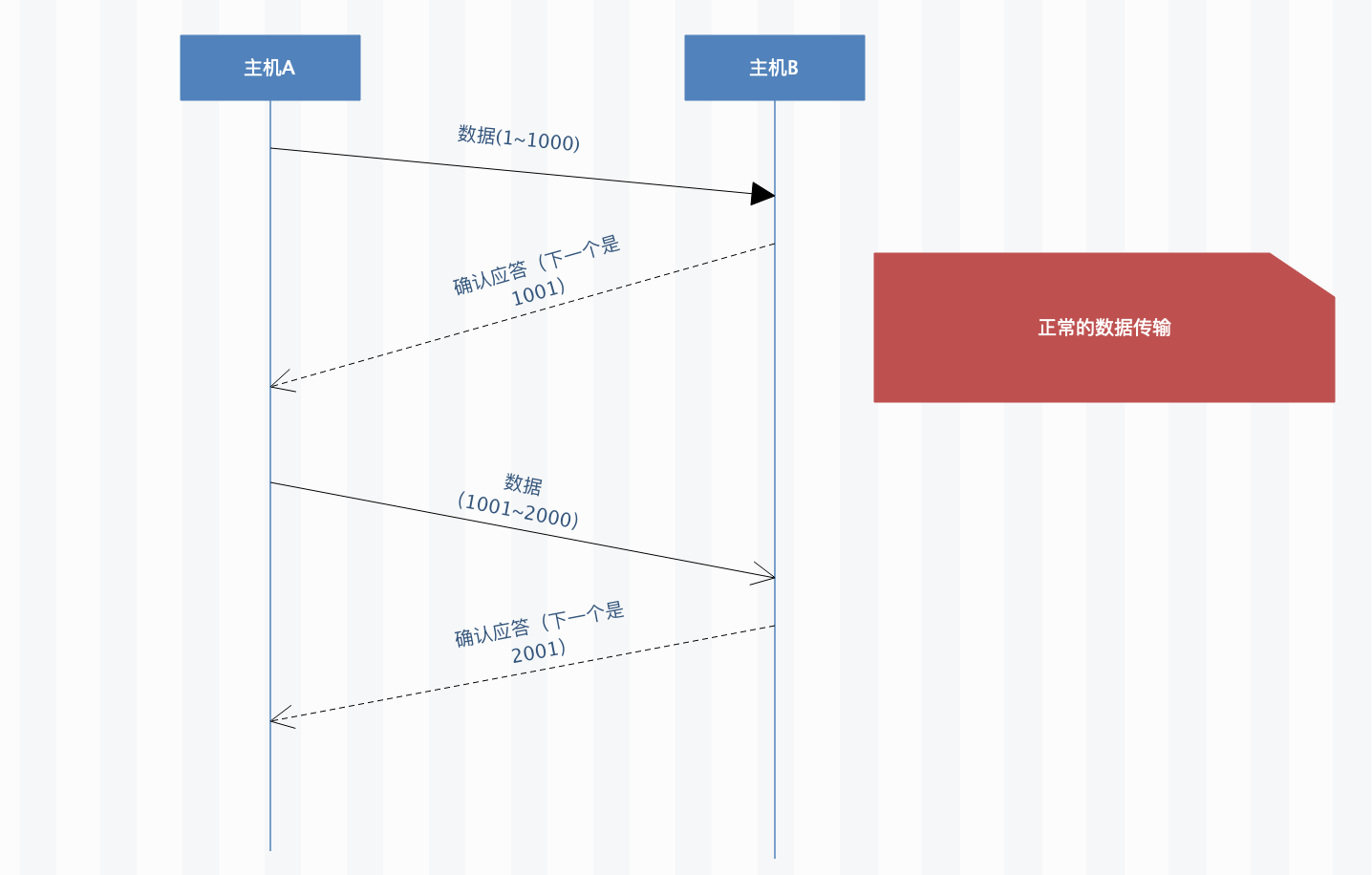
第二次握手时

010010

第三次握手时

010000

正常的数据传输过程



TCP通过肯定的确认应答(ACK)实现可靠的数据传输。当发送端将数据发出之后会等待对端的确认应答。如果有确认应答，说明数据已经成功到达对端。反之，则数据丢失的可能性很大。

在一定时间内没有等到确认应答，发送端就可以认为数据已经丢失，并进行重发。

未收到确认应答并不意味着数据一定丢失，也有可能是数据对方已经收到，只是返回的确认应答在途中丢失。

总结：

在一定时间内等不到确认应答，有可能：

（1）、数据丢失了

（2）、对端收到了数据，确认应答在途中丢失。

接收端查询接收数据TCP首部中的序列号和数据的长度，将自己下一步应该接收的序号作为确认应答返送回去，就这样，通过序列号和确认应答号，TCP可以实现可靠传输。