TCP握手协议

**1.建立连接-三次握手**

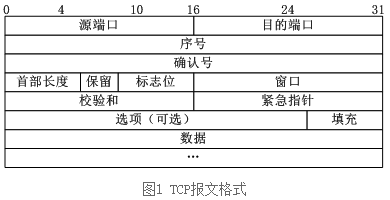
在TCP/IP协议中,TCP协议提供可靠的连接服务,采用**三次握手**建立一个连接.

第一次握手：建立连接时,客户端发送**syn包(seq=j)**到服务器,并进入**SYN\_SEND**状态,等待服务器确认；

**SYN：同步序列编号(Synchronize Sequence Numbers)**

第二次握手：服务器收到**syn包**,必须确认客户的SYN**（ack=j+1）**,同时自己也发送一个**SYN包（seq=k）**,即**SYN+ACK包**,此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN＋ACK包,向服务器发送确认包**ACK(ack=k+1)**,此包发送完毕,客户端和服务器进入**ESTABLISHED**状态,完成三次握手.



（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。

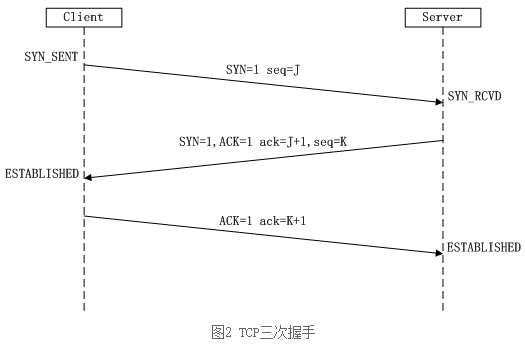
（E）SYN：发起一个新连接。

（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seq+1，两端配对。



第一次：客户端发送建立连接请求，SYN=1，随机生成seq=j，进入SYN\_SENT状态，等待服务器确认

第二次：服务器接收到建立连接请求，将SYN=1，ACK=1，回应ack=j + 1，随机产生seq=k，发送确认连接请求，进入SYN\_RCVD状态

第三次：客户端收到确认连接请求，检查ack是否是j+1，ACK是否为1；如果正确，发送ACK= 1，ack = k + 1到服务器，客户端状态变为ESTABLISHED，服务器接受到确认建立包后，检查ACK和ack，正确就连接成功，进入ESTABLISHED，完成三次握手，可以传输数据。

SYN攻击：在第一次握手的时候，伪造大量不存在的IP地址，不断向server发送SYN请求包，然后server就会不断处理请求，这些伪造的SYN包就会占用未连接的队列，导致正常的SYN请求因为队列满而被丢弃，可以通过指令指令检查，如果大量半连接的状态且源ip是随机的，那么就是遭到SYN攻击了。

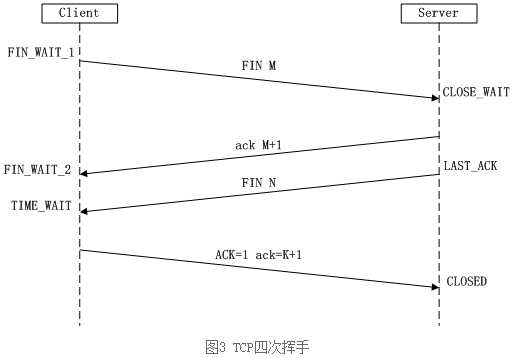
**为什么需要三次握手？**

如果是两次握手，即客户端发送请求，**SYN=1，seq = x**；但是由于网络延时，导致这个包超时没有送到服务器；因此超时重传，客户端再发送一个请求包**SYN=1， seq = y**，此时服务器收到了并建立连接，发送**SYN = 1，ACK =1，seq = j，ack =y + 1**；然后进行数据传输，并释放了连接。

就在这时，服务器收到了那个超时的请求包，以为客户端又要建立连接，就发送了**SYN = 1，ACK =1，seq = k，ack =x + 1**；此时是两次握手，连接建立了，因此服务器等待客户端发送数据，但是客户端此时并不想建立连接，即不会发送数据，导致服务器长时间等待，浪费资源。

如果使用三次握手，为什么能解决这个问题，因为如果客户端收到服务器确认连接的请求，并且**SYN = 1，ACK =1，seq = k，ack =x + 1**；说明服务器收到的请求包是客户端放弃的请求包，因此客户端可以不理睬这个确认连接请求，但是会发送一个复位报文段，RST =1，ACK=1，ack = k+1，表明拒绝TCP连接建立请求，并且告知服务器已经收到其发的确认连接包。

**2. 释放连接-四次挥手**



由于TCP全双工，因此每个方向都要单独进行关闭，当一方完成数据发送任务后，发送FIN来终止这一方向的连接，收到FiN只是意味着一个方向上没有数据流动了，即不会再收到数据了，但是TCP连接上依然能发送数据，知道这一方向也发送FIN。

首先是进行关闭的一方主动关闭，然后另一方则被动关闭。

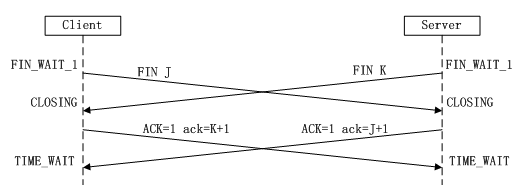
第一次：客户端发送一个FIN = 1，seq = m用来关闭客户端到服务器方向的数据传送，客户端进入FIN\_WAIT1状态

第二次：服务器收到FiN之后，发送一个ACK=1，ack = m +1，服务器进入CLOSE\_WAIT状态

第三次：服务器发送一个FIN=1，seq = n，用来关闭服务器到客户端这个方向上的数据传送，服务器进入LAST\_ACK状态。

第四次：客户端数到FIN之后，进入TIME\_WAIT状态，接着发送一个ACK=1，ack = n +1，进入Close状态，Server收到ACK和ack进行核对后，也进入close状态，完成四次挥手。

还有一种是同时主动关闭



为什么释放连接需要四次挥手？

因为是全双工的形式连接，所以为了避免客户端数据发送完毕后，向服务端发送FIN，然后服务端还有数据没有发送完毕的情况，如果服务端发送FIN，那么这些未发送的数据就不能发送了，因此需要先确认收到ACK，表明客户端已经关闭发送数据连接，等到服务端未发送的数据发送完了，那么再发送FIN请求，关闭服务端的发送连接。

关于TIME\_WAIT?

等待2MS的原因，因为服务器那边可能会在一段时间内，没有收到客户端的ACK信号，就会超时重传，这样客户端就知道上一条确认关闭请求没有发送成功，从而重新发送。如果没有等待2ms立即关闭，那么服务器就不知道客户端是否收到自己的关闭请求，因此会引起关闭异常；