返回目录:全网各大厂 iOS 面试题-题集大全

三次握手与四次挥手

在面试中，三次握手和四次挥手可以说是问的最频繁的一个知识点了，我相信大家也都看过很多关于三次握手与四次挥手的文章，今天的这篇文章，重点是围绕着面试，我们应该掌握哪些比较重要的点，哪些是比较被面试官给问到的，我觉得如果你能把我下面列举的一些点都记住、理解，我想就差不多了。

三次握手

当面试官问你为什么需要有三次握手、三次握手的作用、讲讲三次三次握手的时候，我想很多人会这样回答：

首先很多人会先讲下握手的过程：

1. **第一次握手：** 客户端给服务器发送一个 SYN 报文。
2. **第二次握手：** 服务器收到 SYN 报文之后，会应答一个 SYN+ACK 报文。
3. **第三次握手：** 客户端收到 SYN+ACK 报文之后，会回应一个 ACK 报文。
4. 服务器收到 ACK 报文之后，三次握手建立完成。

作用是为了确认双方的接收与发送能力是否正常。

这里我顺便解释一下为啥只有三次握手才能确认双方的接受与发送能力是否正常，而两次却不可以：

第一次握手： 客户端发送网络包，服务端收到了。这样服务端就能得出结论：客户端的发送能力、服务端的接收能力是正常的。

第二次握手： 服务端发包，客户端收到了。这样客户端就能得出结论：服务端的接收、发送能力，客户端的接收、发送能力是正常的。不过此时服务器并不能确认客户端的接收能力是否正常。

第三次握手： 客户端发包，服务端收到了。这样服务端就能得出结论：客户端的接收、发送能力正常，服务器自己的发送、接收能力也正常。

因此，需要三次握手才能确认双方的接收与发送能力是否正常。

这样回答其实也是可以的，但我觉得，这个过程的我们应该要描述的更详细一点，因为三次握手的过程中，双方是由很多状态的改变的，而这些状态，也是面试官可能会问的点。所以我觉得在回答三次握手的时候，我们应该要描述的详细一点，而且描述的详细一点意味着可以扯久一点。加分的描述我觉得应该是这样：

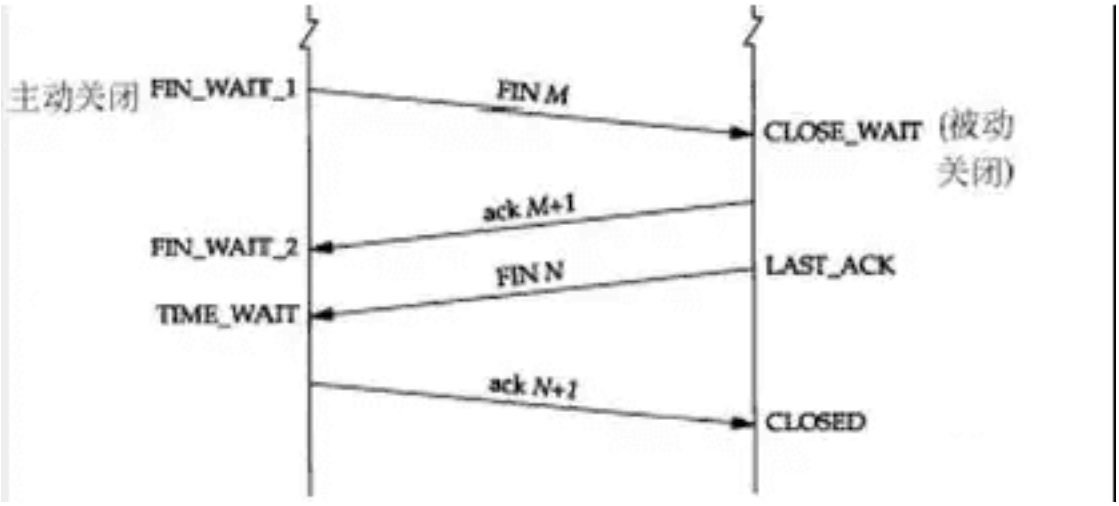
刚开始客户端处于 closed 的状态，服务端处于 listen 状态。然后

1. **第一次握手：** 客户端给服务端发一个 SYN 报文，并指明客户端的初始化序列号 ISN(c)。此时客户端处于 SYN_Send 状态。

2. **第二次握手：** 服务器收到客户端的 SYN 报文之后，会以自己的 SYN 报文作为应答，并且也是指定了自己的初始化序列号 ISN(s)，同时会把客户端的 ISN + 1 作为 ACK 的值，表示自己已经收到了客户端的 SYN，此时服务器处于 SYN_REVD 的状态。

3. **第三次握手：** 客户端收到 SYN 报文之后，会发送一个 ACK 报文，当然，也是一样把服务器的 ISN + 1 作为 ACK 的值，表示已经收到了服务端的 SYN 报文，此时客户端处于 establised 状态。

4. 服务器收到 ACK 报文之后，也处于 establised 状态，此时，双方以建立起了链接。



三次握手的作用

三次握手的作用也是有好多的，多记住几个，保证不亏。例如：

1. 确认双方的接受能力、发送能力是否正常。

2. 指定自己的初始化序列号，为后面的可靠传送做准备。

3. 如果是 https 协议的话，三次握手这个过程，还会进行数字证书的验证以及加密密钥的生成到。

单单这样还不足以应付三次握手，面试官可能还会问一些其他的问题，例如：

1、(ISN)是固定的吗

三次握手的一个重要功能是客户端和服务端交换 ISN(Initial Sequence Number), 以便让对方知道接下来接收数据的时候如何按序列号组装数据。

如果 ISN 是固定的，攻击者很容易猜出后续的确认号，因此 ISN 是动态生成的。

2、什么是半连接队列

服务器第一次收到客户端的 SYN 之后，就会处于 SYN_RCVD 状态，此时双方还没有完全建立其连接，服务器会把此种状态下请求连接放在一个队列里，我们把这种队列称之为半连接队列。当然还有一个全连接队列，就是已经完成三次握手，建立起连接的就会放在全连接队列中。如果队列满了就有可能会出现丢包现象。

这里在补充一点关于 SYN-ACK 重传次数的问题：服务器发送完 SYN-ACK 包，如果未收到客户确认包，服务器进行首次重传，等待一段时间仍未收到客户确认包，进行第二次重传，如果重传次数超 过系统规定的最大重传次数，系统将该连接信息从半连接队列中删除。注意，每次重传等待的时间不一定相同，一般会是指数增长，例如间隔时间为 1s, 2s, 4s, 8s, ….

3、三次握手过程中可以携带数据吗

很多人可能会认为三次握手都不能携带数据，其实第三次握手的时候，是可以携带数据的。也就是说，第一次、第二次握手不可以携带数据，而第三次握手是可以携带数据的。

为什么这样呢?大家可以想一个问题，假如第一次握手可以携带数据的话，如果有人要恶意攻击服务器，那他每次都在第一次握手中的 SYN 报文中放入大量的数据，因为攻击者根本就不理服务器的接收、发送能力是否正常，然后疯狂着重复发 SYN 报文的话，这会让服务器花费很多时间、内存空间来接收这些报文。也就是说，第一次握手可以放数据的话，其中一个简单的原因就是会让服务器更加容易受到攻击了。

而对于第三次的话，此时客户端已经处于 established 状态，也就是说，对于客户端来说，他已经建立起连接了，并且也已经知道服务器的接收、发送能力是正常的了，所以能携带数据页没啥毛病。

关于三次握手的，https 的认证过程能知道一下最好，不过我就不说了，留着写 http 面试相关时的文章再说。

四次挥手

四次挥手也一样，千万不要对方一个 FIN 报文，我方一个 ACK 报文，再我方一个 FIN 报文，我方一个 ACK 报文。然后结束，最好是说的详细一点，例如想下面这样就差不多了，要把每个阶段的状态记好，我上次面试就被问了几个了，呵呵。我答错了，还以为自己答对了，当时还解释的头头是道，呵呵。

刚开始双方都处于 estabilised 状态，假如是客户端先发起关闭请求，则：

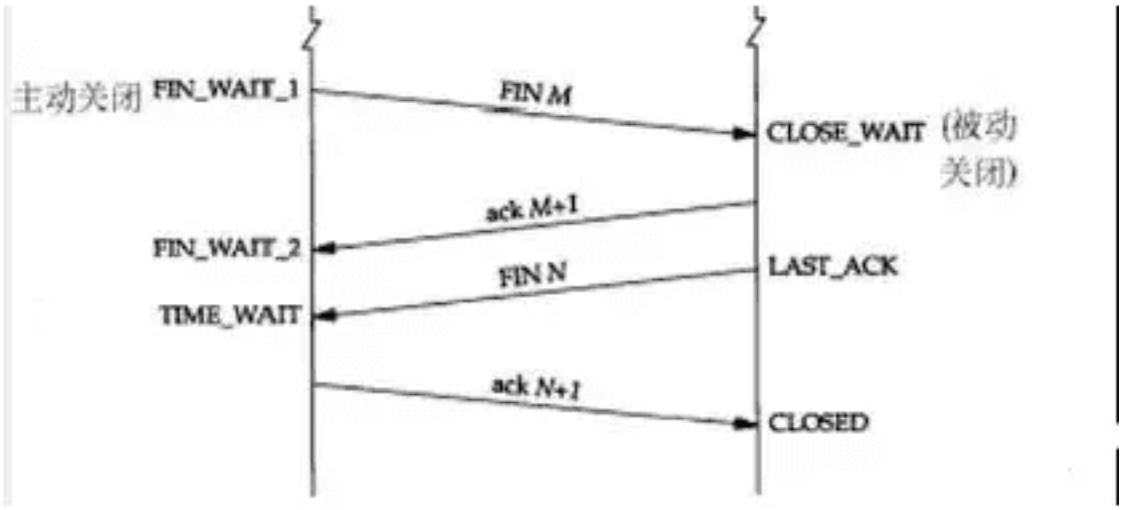
1. **第一次挥手：** 客户端发送一个 FIN 报文，报文中会指定一个序列号。此时客户端处于 CLOSED_WAIT1 状态。

2. **第二次握手：** 服务端收到 FIN 之后，会发送 ACK 报文，且把客户端的序列号值 + 1 作为 ACK 报文的序列号值，表明已经收到客户端的报文了，此时服务端处于 CLOSE_WAIT2 状态。

3. **第三次挥手：** 如果服务端也想断开连接了，和客户端的第一次挥手一样，发给 FIN 报文，且指定一个序列号。此时服务端处于 LAST_ACK 的状态。

4. **第四次挥手：** 客户端收到 FIN 之后，一样发送一个 ACK 报文作为应答，且把服务端的序列号值 + 1 作为自己 ACK 报文的序列号值，此时客户端处于 TIME_WAIT 状态。需要过一阵子以确保服务端收到自己的 ACK 报文之后才会进入 CLOSED 状态

5. 服务端收到 ACK 报文之后，就处于关闭连接了，处于 CLOSED 状态。



这里特别需要主要的就是 TIME_WAIT 这个状态了，这个是面试的高频考点，就是要理解，为什么客户端发送 ACK 之后不直接关闭，而是要等一阵子才关闭。这其中的原因就是，要确保服务器是否已经收到了我们的 ACK 报文，如果没有收到的话，服务器会重新发 FIN 报文给客户端，客户端再次收到 FIN 报文之后，就知道之前的 ACK 报文丢失了，然后再次发送 ACK 报文。

至于 TIME_WAIT 持续的时间至少是一个报文的来回时间。一般会设置一个计时，如果过了这个计时没有再次收到 FIN 报文，则代表对方成功就是 ACK 报文，此时处于 CLOSED 状态。

这里我给出每个状态所包含的含义，有兴趣的可以看看。

LISTEN - 侦听来自远方 TCP 端口的连接请求;

SYN-SENT -在发送连接请求后等待匹配的连接请求;

SYN-RECEIVED - 在收到和发送一个连接请求后等待对连接请求的确认;

ESTABLISHED - 代表一个打开的连接，数据可以传送给用户;

FIN-WAIT-1 - 等待远程 TCP 的连接中断请求，或先前的连接中断请求的确认;

FIN-WAIT-2 - 从远程 TCP 等待连接中断请求;

CLOSE-WAIT - 等待从本地用户发来的连接中断请求;

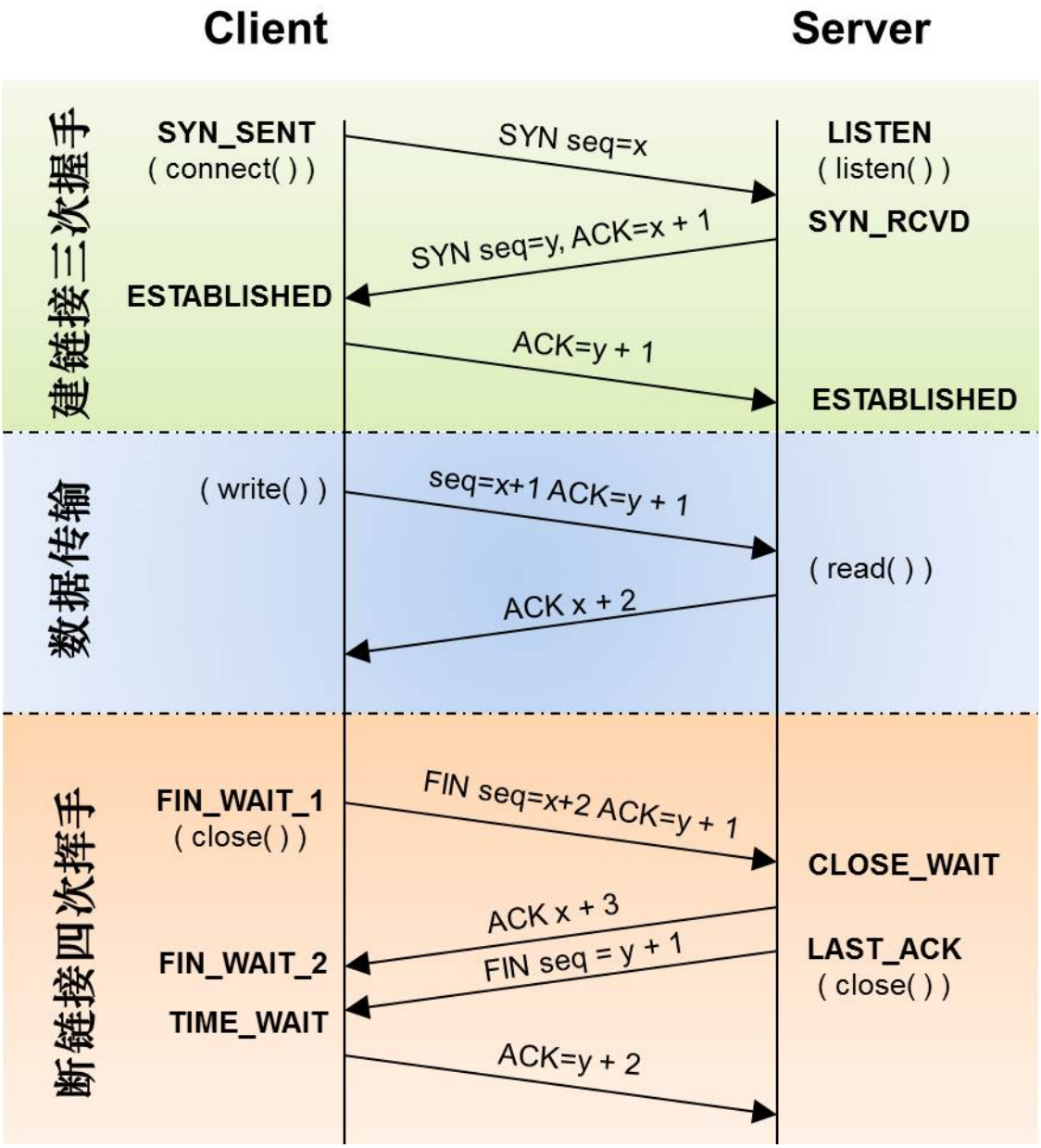
CLOSING -等待远程 TCP 对连接中断的确认;

LAST-ACK - 等待原来发向远程 TCP 的连接中断请求的确认;

TIME-WAIT -等待足够的时间以确保远程 TCP 接收到连接中断请求的确认;

CLOSED - 没有任何连接状态;

再放个三次握手与四次挥手的图



返回目录:全网各大厂 iOS 面试题-题集大全

更多精选大厂 · iOS 面试题答案 PDF 文集

*