# TCP协议部分知识梳理

- 特点
  - 面向连接，确认重传，流量控制，可靠交付的传输层协议。
  - 可靠传输的定义：接收端能够按照正确的顺序接收到正确的内容。
  - 可靠传输的实现：TCP的可靠性是通过顺序编号（seq）和确认（ACK/ack）来实现的。TCP在开始传送一个报文时，为准备重传而首先将该报文插入到发送队列之中，同时启动时钟。其后，如果收到了接受端对该报文的ACK信息，就将该报文从队列中删去。如果在时钟规定的时间内，ACK未返回，那么就从发送队列中再次发送这个报文。
  - 一次完整的TCP通信过程：建立连接（三次握手）、数据传输、断开连接（四次挥手）。
- TCP报文头部格式
  - 源端口，目的端口
  - 序列号 seq： 16位的二进制数，每个TCP报文都有，确保有序接收。
  - 确认号 ack： 16位的二进制数，每个TCP报文都有，向对方确认收到了多少数据，同时也表示期望的对方（接收方）的下一次发送的报文的sequence number是多少。
  - SYN同步标志位：代表是建立连接的请求报文，同步序列编号只有第一次和第二次握手时为1。

    同步双方的初始序列号
  - ACK确认标志位：向对端确认收到的报文序号时为1。
  - FIN结束标志位：四次挥手时第一次和第三次挥手时为1，代表通知对方，自己已经发送完了，准备断开。
  - 窗口大小：滑动窗口的大小，滑动窗口用于流量控制。
- TCP的"三次握手"
  - 握手过程
    - 请求方发起第一次握手，发送SYN=1，seq=x；
    - 接收方收到请求后，发送SYN=1，ACK=1,seq=y，ack=x+1;
    - 请求方收到后，发送ACK=1，seq=x+1 ack=y+1；A收到B的ACK，是不会对ACK再做确认的

      不会对纯粹的确认包再做确认
  - 🔴为什么需要三次握手?

    常见面试题
    - ⭐理解1：其实所谓的TCP三次握手请求连接，无非就是初始化一个序列号，保证后面的数据有序到达。

前两次握手成功后(两次SYN)，双方就已经确认了对方的序列号。这样双方都知道什么样的序列号是合法的，什么样的是不合法的。server端接收到不合法的就丢弃，接收到合法的就认为连接是建立成功的。

- 理解2：“三次握手”的目的是“为了防止已失效的连接请求报文段突然又传送到了服务端，因而产生错误”。（不能使用两次握手的原因）。

  假设不采用“三次握手”（“两次握手就能成功”），那么只要server发出确认，新的连接就建立了。由于现在client并没有发出建立连接的请求，因此不会理睬server的确认，也不会向server发送数据。但server却以为新的运输连接已经建立，并一直等待client发来数据。这样，server的很多资源就白白浪费掉了。采用“三次握手”的办法可以防止上述现象发生。

- 理解3：请求方发送同步信号，接收方收到后需要确认，TCP确认收到的包

  也可理解为TCP是全双工，为了保证数据的全双工，双方要通知对方自己的收发能力是OK的。

  - 1 A 发送同步信号SYN+A的初始序列号seq=x

    可以认为是A发B收的请求

  - 2 B 确认收到A的同步信号，发送ACK=1，ack=x+1
  - 3 B发送同步信号SYN + B的初始化序列号seq=y

    可以认为是B发A收的请求

  - 4 A确认收到B的同步信号，发送ACK=1，ack=y+1
  - 为了减少时延，避免资源的浪费，2和3两步合并发送

- 🔴握手时包丢了怎么办?
  - 第一个包，即A发给B的SYN 中途被丢，没有到达B
    - A会周期性超时重传，直到收到B的确认
  - 第二个包，即B发给A的SYN +ACK 中途被丢，没有到达A
    - B会周期性超时重传，直到收到A的确认

      A没有收到ACK，也会超时重传。
  - 第三个包，即A发给B的ACK 中途被丢，没有到达B
    - A发完ACK，单方面认为TCP为 Established状态，而B显然认为TCP为Active(SYN_REVD)状态
    - A会超时重传这个ACK吗？不会！TCP不会为没有数据的ACK超时重传
    - 1.双方都没有数据发送，B会周期性超时重传，直到收到A的确认，收到之后B的TCP 连接也为 Established状态，双向可以发包
    - 2. 假定此时A有数据发送，B收到A的 Data + ACK，自然会切换为Established 状态，并接受A的Data

      A发给B的第三次握手的seq=x+1,A下一次发给B的正常数据的seq也是x+1
    - 3. 假定B有数据发送，数据发送不了，会一直周期性超时重传SYN + ACK，直到收到A的确认才可以发送数据

TCP的“四次挥手”

- 半关闭
- 🔴在TIME_WAIT状态时的端口不能使用，要等到2MSL时间结束才可继续使用。当连接处于2MSL等待阶段时任何迟到的报文段都将被丢弃。
- 2MSL Maximum Segment Lifetime英文的缩写，中文可以译为“最大报文段生存时间”，他是任何报文在网络上存在的最长时间，超过这个时间报文将被丢弃。

▾ 为什么建立连接只需要三次握手？断开连接需要四次挥手？

- 因为TCP是全双工模式，接收到FIN时意味将没有数据再发来，但是还是可以继续发送数据。

  当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

▾ 🔴为什么TIME_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回CLOSED状态？

- 1. 保证TCP协议的全双工连接能够可靠关闭

  如果Client直接CLOSED了，那么由于IP协议的不可靠性或者是其它网络原因，导致Server没有收到Client最后回复的ACK。那么Server就会在超时之后继续发送FIN，此时由于Client已经CLOSED了，就找不到与重发的FIN对应的连接，最后Server就会收到RST而不是ACK，Server就会以为是连接错误把问题报告给高层。这样的情况虽然不会造成数据丢失，但是却导致TCP协议不符合可靠连接的要求。所以，Client不是直接进入CLOSED，而是要保持TIME_WAIT，当再次收到FIN的时候，能够保证对方收到ACK，最后正确的关闭连接。

- 2. 保证这次连接的重复数据段从网络中消失

  虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假设网络是不可靠的，有可能最后一个ACK丢失。所以TIME_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。
  A发给B的第四次挥手丢了，B发送第三次挥手之后的超时重传的计时器的时间加上数据包在网络中的传输时间一定小于2MSL的，所以能够保证第四次挥手的包丢了，仍然有足够的时间超时重传。