1. **三次握手**

TCP连接都是C/S方式的，握手都是客户端主动发起的。

服务器运行之后就会处于监听（Listen）状态，客户端主动给服务器发送连接请求（IP）数据报后会进入SYN-sent的状态，发送的请求数据报中，TCP首部标记位SYN=1，序号seq=x（随机）。服务器收到请求后会给客户端发回确认数据报然后进入SYN-rcvd的状态，发回数据包中首部标记位SYN也是1，同时还多了一个确认为ACK为1，还有一个它自己的序号y，确认号是请求序号加1。客户端收到服务器的确认数据包后也会给服务器发送一个确认数据报，然后处于established的状态。发回确认数据报，主要是为了告诉服务器我已经受到你的确认并且处于连接状态了。发送的确认数据报中没有SYN标记位了，有确认为ACK为1，序号是请求序号加1，确认号是刚刚收到的服务器确认数据包的序号加1。服务器收到确认数据报后也进入established的状态，此时双方就可以进行数据的交互了。

*三次握手和四次挥手都是tcp协议特有的，也有的人说http的三次握手和四次挥手，但其实http协议也是运行在tcp协议上的。*

1. **为什么不是两次握手**
2. 实现可靠通信。双方都会有一个序号，互相进行序号的确认就保证了通信的可靠。
3. 放置重复连接。当服务器端收到过期链接请求并回应后，客户端已经不需要了，就不会进行第三次握手，也就不会建立连接。
4. **四次挥手**

四次挥手本质上就是我释放连接，你确认；然后再到你释放连接，我确认的过程。通信结束后双方都可以主动释放连接，我们假设是客户端释放和服务器的连接

首先，双方都是在established的状态，客户端向服务器发送了一个释放报文段，然后进入FIN-wait状态，报文段首部标记位FIN为1，序号为u。服务器收到后发回对客户端的确认，然后进入CLOSE-wait的状态，确认报文段的确认标记位ACK为1，序号为v，确认号为刚刚请求释放报文段的序号加1（u+1）.服务器发送回第一个确认报文段后仍然可以单方向给客户端传数据，当数据都传完了之后，服务器也向客户端发送了一个释放报文段后就进入了LAST-ACK的状态，发回的释放报文段中有释放标记位FIN为1，确认标记ACK为1，它有自己的序号为w，确认号和上一次发的报文段的确认号一样。客户端收到后也会向服务器发回一个确认报文段，然后进入TIME-WAIT状态，之后在等待两倍的最大报文段生存时间就会真正释放连接，这个确认报文段中有确认位ACK为1，序号为第一次挥手的序号加1（u+1），确认号为上一次挥手的序号加1（w+1）。服务器收到这个确认数据报后就直接释放连接了。

1. **为什么是四次挥手**

四次挥手是两次释放连接的过程。假设是客户端不需要再想服务端发送数据了它就会发送释放报文段，服务器会回应一个确认报文段。但是还不能真正断开连接，要是服务器还有消息没有发送完怎么办，所以此时服务器仍然可以给客户端发送数据。等到发送完了，服务端也给客户端发送一个释放报文段，客户端会回应一个确认报文段，之后再等待两个最大报文段生存时间后才会真正释放连接。而服务端收到确认报文段后就直接释放连接了。

1. **为什么要等待两个最大报文段生存时间**
2. 保证最后一个确认报文段能够到达服务器，否则服务器收不到的话就会一直超时重发释放报文段，直到被确认。
3. 确保失效的连接请求都消失在本次连接中。
4. **TCP和UDP的区别**

TCP是面向连接的可靠传输，UDP无连接不可靠；

TCP传输数据有序，UDP不保证数据的有序性;

TCP面向字节流，把数据看成一连串字节流，UDP是面向报文的

TCP传输速度相对UDP较慢;

TCP有流量控制和拥塞控制，UDP没有;

TCP是重量级协议，UDP是轻量级协议;

TCP首部20字节，UDP首部8字节;

TCP连接只能是一对一的（端到端）；UDP支持一对一、一对多、多对一和多对多的通信方式。

1. **TCP如何保证可靠传输**
   1. 对数据报的校验、响应、重传（举个例子）。
   2. 对失序数据包重新排序，然后丢弃重复的数据。
   3. 流量控制，主要是为了让接收方来得及接收数据。
2. **流量控制和拥塞控制**

拥塞控制和流量控制不同，拥塞控制是一个全局性的过程，而流量控制指点对点通信量的控制。

流量控制是为了控制发送方的发送速率，保证接收方来得及接收。TCP会话的双方分别维护一个发送窗口和一个接收窗口，接收方送回的响应报文段中有可以控制发送窗口大小的window字段。发送窗口的大小一般约等于接收窗口的大小。

拥塞控制就是为了防止过多的数据注入到网络中，防止链路过载。为了进行拥塞控制，TCP 发送方要维持一个拥塞窗口(cwnd)的状态变量。拥塞控制窗口的大小取决于网络的拥塞程度，并且动态变化。发送方让自己的发送窗口取为拥塞窗口和接收方的接受窗口中较小的一个。

TCP的拥塞控制采用了四种算法，即：慢开始、拥塞避免、快重传和快恢复。

慢开始：逐渐增大拥塞窗口，cwnd初始值为 1，每经过一个传播轮次，cwnd 加倍。

拥塞避免：让拥塞窗口 cwnd 缓慢增大，即每经过一个往返时间 RTT 就把发送方的 cwnd 加 1。

快重传与快恢复：简称FRR是一种拥塞控制算法，它能快速恢复丢失的数据包。

1. **适合使用TCP协议的场景：**
   1. 可靠传输（如文件，邮件）；
   2. 保证有序（如视频，音频）
   3. 数据流量较大的场景，因为TCP提供了流量控制和拥塞控制（如Web应用和数据库）
2. **适合使用UDP协议的场景：**
   1. 实时性要求高的场景：UDP没有TCP的重传机制，数据传输速度更快。（如视频会议、在线游戏等）
   2. 数据量较小的场景：没有流量控制和拥塞控制机制，不会调整数据发送的速度（如DNS查询、NTP协议等）
   3. 需要广播和多播的场景：可以在网络中同时向多个主机传输数据。（如视频流媒体等）

网**络七层模型及其作用**

* 1. 应用层 为应用提供交互服务
  2. 表示层 数据转化
  3. 会话层 建立或解除联系
  4. 运输层 为不同主机的进程间的数据传输提供服务
  5. 网络层 为数据报选择路由
  6. 数据链路层 把IP数据报组装成帧和差错检测
  7. 物理层 以二进制形式传输数据

1. **常见的网络协议**
2. **[应用层](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_24" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**[协议](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_24" \t "https://blog.csdn.net/m0_46233999/article/details/_self)：DNS,HTTP,HTTPS,TFTP,FTP,SNMP,SMTP

**[DNS](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "DNS_27" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：53 UDP/TCP 是将人类可读的域名 (如，www.baidu.com) 转换为机器可读的 IP 地址 (如，192.0.2.44)。

**[HTTP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "HTTP_45" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：80 TCP 指定了客户端可能发送给服务器什么样的消息以及得到什么样的响应。

**[HTTPS](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "HTTPS_109" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：443 HTTP的升级版，增加了传输数据的安全性，HTTPS协议是在HTTP的基础上增加了一个SSL外壳

**DHCP：**自动分配IP地址

1. **[传输层](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_143" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**[协议](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_143" \t "https://blog.csdn.net/m0_46233999/article/details/_self)

**[UDP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "UDP_145" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：无需建立连接就可以发送封装的 IP 数据包

**[TCP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "TCP_161" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：是一种面向连接、可靠的、基于字节流的传输层协议

1. **[网络层](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_183" \t "https://blog.csdn.net/m0_46233999/article/details/_self) IP,ICMP,IGMP,RIP,OSPF,BGP**

**[IP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "IP_188" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：寻址和路由；分片与重组

**[ICMP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "ICMP_223" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：放在IP数据包里 检测网络通信故障和实现链路追踪

1. **[数据链路层](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "_231" \t "https://blog.csdn.net/m0_46233999/article/details/_self) ARP,PPP,SLIP,CSLIP,MTU**

**[ARP](https://blog.csdn.net/m0_46233999/article/details/119455352?ops_request_misc=%7B%22request%5Fid%22%3A%22167221315216800180613894%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=167221315216800180613894&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-119455352-null-null.142^v68^control,201^v4^add_ask,213^v2^t3_control2&utm_term=%E5%B8%B8%E8%A7%81%E7%9A%84%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE%E6%9C%89%E5%93%AA%E4%BA%9B&spm=1018.2226.3001.4187" \l "ARP_234" \t "https://blog.csdn.net/m0_46233999/article/details/_self)**：负责把目的主机的IP 地址解析成目的MAC地址

1. **HTTP和HTTPS的区别**

首先最大的区别就是HTTP是明文传输，而HTTPS是HTTP的升级加密版，被加密协议封装起来，然后加密协议再运行于TCP协议之上，所以他们使用的端口是不同的，而且这个这个加密协议需要缴费使用。另外因为HTTPS传输过程需要加密和解密处理，需要消耗更多的CPU资源。

1. **HTTP1.0 --> HTTP1.1 --> HTTP2**

HTTP1.0只能保持短暂连接

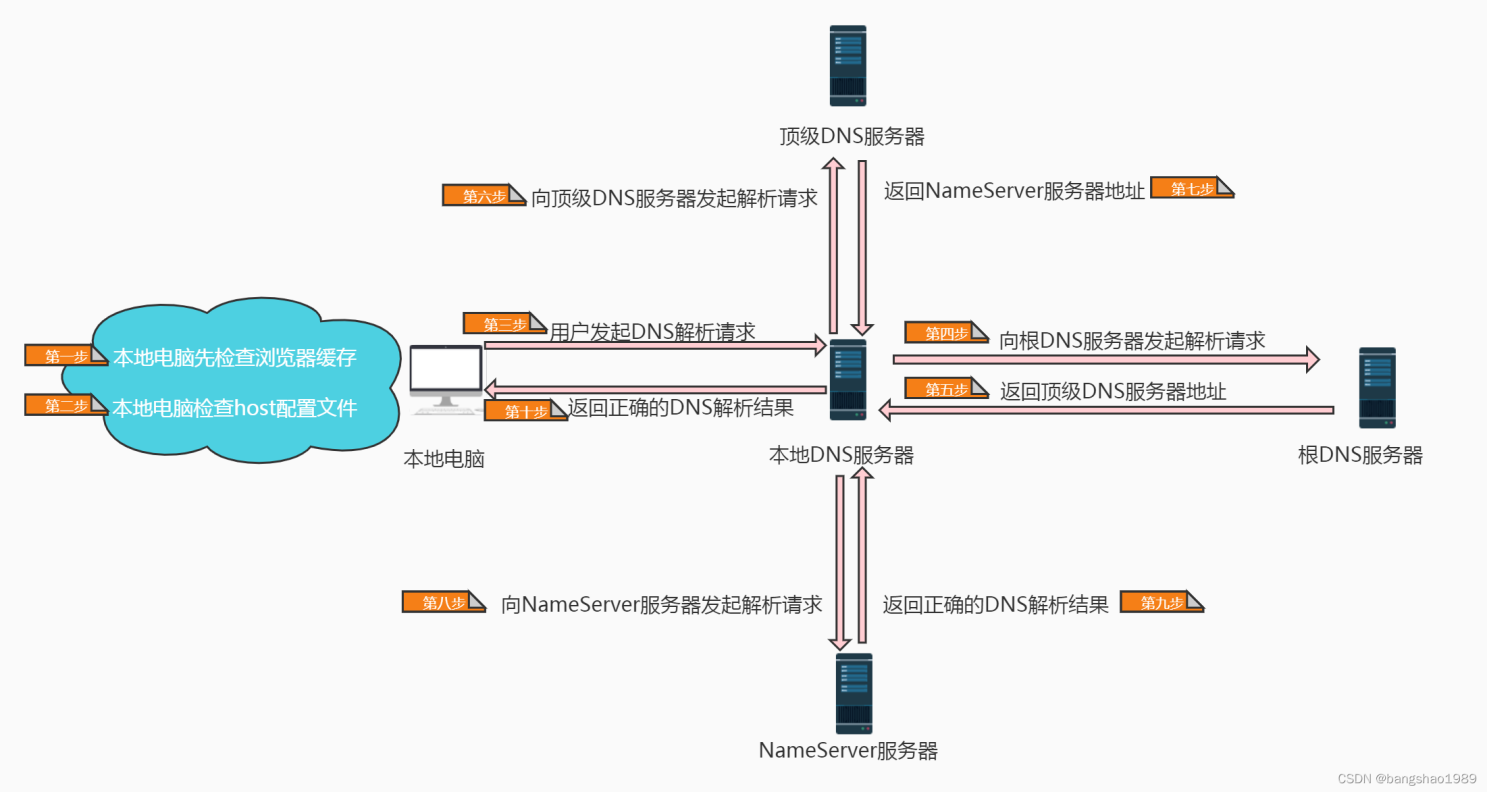
HTTP1.1的TCP连接默认不关闭，可以被多个请求复用；引入管道机制，客户端可以同时发送多个请求。

HTTP2是二进制协议，数据从文本传输进化为二进制传输且对头部（Header）进行压缩，二进制更符合机器的特性所以传输性能更高，HTTP2还支持多路复用和服务端推送。

多路复用：一个连接可以发送多个请求和回应，并且可以无序

服务端推送：客户端将服务端可能需要的数据事先发送到服务端

1. **DNS过程**



1. **浏览器访问www.baidu.com经历了什么**

（1）浏览器向DNS服务器发出解析域名的请求；

（2）DNS服务器将"www.baidu.com"域名解析为对应的IP地址，并返回给浏览器；

（3）浏览器与百度服务器进行三次握手，建立TCP连接；

（4）浏览器发出HTTP请求报文；

（5）服务器回复HTTP响应报文；

（6）浏览器解析响应报文，渲染HTML内容，并显示在页面上；

（7）收发报文结束，执行四次挥手，释放TCP连接。

1. **几种常用的序列化协议**

序列化：将对象序列化为二进制形式放进字节数组里，用于网络传输、数据持久化等。

反序列化：对网络传输对象的解码，和将磁盘上的字节数组还原为原始对象。

* 1. XML 适用于配置文件存储和实时数据转换
  2. JSON 适用于传输数据量相对较小的情况，可以跨防火墙
  3. Fastjson 适用于web和Android
  4. Thrift 适用于分布式系统的RPC解决方案
  5. Avro
  6. Protobuf

1. **互联网是如何工作的**
2. **浏览器是如何运作的**