[TCP协议中的三次握手和四次挥手(图解)](https://www.cnblogs.com/thrillerz/p/6464203.html)

ACK：**发送方根据收到ACK当中的期望收到的下一个字节的序号n以及窗口m，还有当前已经发送的字节序号x，算出还可以发送的字节数。**

建立连接/三次握手：

首先Client端发送连接请求报文，Server段接受连接后回复ACK报文，并为这次连接分配资源。Client端接收到ACK报文后也向Server段发生ACK报文，并分配资源，这样TCP连接就建立了。

断开连接/四次挥手：

**中断连接端可以是Client端，也可以是Server端。**

假设Client端发起中断连接请求，也就是发送FIN报文。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据。所以你先发送ACK，"告诉Client端，你的请求我收到了，但是我还没准备好，请继续你等我的消息"。这个时候Client端就进入FIN\_WAIT状态，继续等待Server端的FIN报文。当Server端确定数据已发送完成，则向Client端发送FIN报文，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。Client端收到FIN报文后，"就知道可以关闭连接了，但是他还是不相信网络，怕Server端不知道要关闭，所以发送ACK后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。“，Server端收到ACK后，"就知道可以断开连接了"。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。



# TCP流量控制与拥塞控制浅析

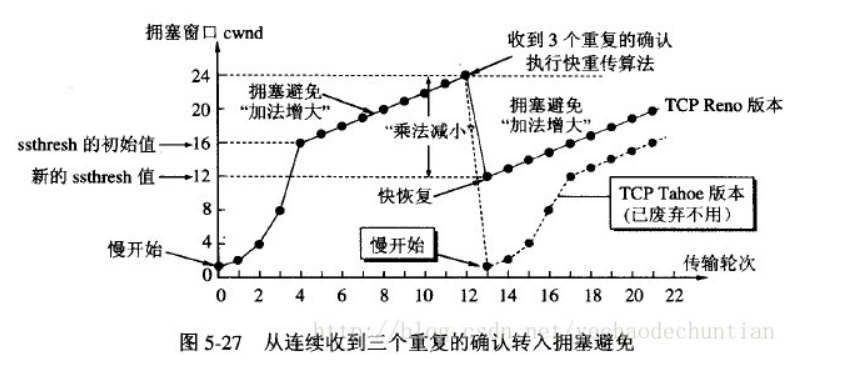
发送方窗口的上限值 = Min [ rwnd, cwnd ]

滑动窗口：rwnd

滑动窗口是接受数据端使用的窗口大小，用来告知发送端接收端的缓存大小，以此可以控制发送端发送数据的大小，从而达到**流量控制**的目的

拥塞窗口：cwnd

那么对于数据的发送端就是拥塞窗口了，拥塞窗口不代表缓存，拥塞窗口指某一源端数据流在一个RTT内可以最多发送的[数据包](http://baike.baidu.com/view/25880.htm)数,cwnd:发送端窗口( congestion window )。



从**慢开始**，流量很低开始，流量每次\*2增加，试探网络是否能承受。当到了ssthresh时，开始**拥塞避免**，改为加性增。当遇到丢包/拥塞情况（收到3个重复的确认），执行**快重传**，不用等到重传即计时到期，马上重新传递未被确认报文段。网络拥塞时，**快恢复**，乘性减少cwnd和ssthresh，然后再加性增。（以前是ssthresh减半，从慢开始再重新开始，）

# 详解DNS域名解析全过程

域名解析：为了找到域名对应的IP地址。

1. 浏览器缓存
2. 操作系统缓存（windows的hosts）
3. 本都域名解析（LDNS，离你很近的附近）缓存
4. LDNS向根域名服务器请求，返回顶级域名服务器ip
5. LDNS向顶级域名服务器（com/cn/org等）请求，返回权限域名服务器
6. LDNS向权限域名服务器请求，返回结果或下一个权限域名服务器
7. LDNS向用户返回域名ip

OSI七层协议/ TCP/IP四层协议/五层协议



HTTP协议：

HTTP是一个应用层协议。

HTTP是一个无状态的协议，同一个客户端的这次请求和上次请求是没有对应关系。

一次HTTP操作称为一个事务，其工作过程可分为四步：

1）首先客户机与服务器需要建立连接。只要单击某个超级链接，HTTP的工作开始。

2）建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可能的内容

3）服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME信息包括服务器信息、实体信息和可能的内容。

4）客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。如果在以上过程中的某一步出现错误，那么产生错误的信息将返回到客户端，有显示屏输出。对于用户来说，这些过程是由HTTP自己完成的，用户只要用鼠标点击，等待信息显示就可以了。

**常用的请求方式**

常用的请求方式是GET和POST.

GET方式：请求URL所指定资源的信息，如果请求URL只是一个数据产生过程，那么最终要在响应实体中返回的是处理过程的结果所指向的资源，而不是处理过程的描述。  
POST方式：用来向目的服务器发出请求，要求它接受被附在请求后的实体，并把它当作请求队列中请求URI所指定资源的附加新子项。

Get是向服务器索取数据的一种请求；而Post是向服务器提交数据的一种请求，要提交的数据位于信息头后面的实体中。

**GET与POST方法有以下区别：**

（1）   在客户端，Get方式在通过URL提交数据，数据在URL中可以看到；POST方式，数据放置在HTML HEADER内提交。

（2）   GET方式提交的数据最多只能有1024字节，而POST则没有此限制。

（3）   安全性问题。正如在（1）中提到，使用 Get 的时候，参数会显示在地址栏上，而 Post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好。

（4）   安全的和幂等的。所谓安全的意味着该操作用于获取信息而非修改信息。幂等的意味着对同一 URL 的多个请求应该返回同样的结果。

**Cookie和Session**

Cookie和Session都为了用来保存状态信息，都是保存客户端状态的机制，它们都是为了解决HTTP无状态的问题而所做的努力。

Session可以用Cookie来实现，也可以用URL回写的机制来实现。用Cookie来实现的Session可以认为是对Cookie更高级的应用。

**Cookie和Session有以下明显的不同点**：

1）Cookie将状态保存在客户端，Session将状态保存在服务器端；

2）Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一个服务器。网络服务器用HTTP头向客户端发送cookies，在客户终端，浏览器解析这些cookies并将它们保存为一个本地文件，它会自动将同一服务器的任何请求缚上这些cookies。Session并没有在HTTP的协议中定义；

3）Session是针对每一个用户的，变量的值保存在服务器上，用一个sessionID来区分是哪个用户session变量,这个值是通过用户的浏览器在访问的时候返回给服务器，当客户禁用cookie时，这个值也可能设置为由get来返回给服务器；

4）就安全性来说：当你访问一个使用session 的站点，同时在自己机子上建立一个cookie，建议在服务器端的SESSION机制更安全些.因为它不会任意读取客户存储的信息。

**ARP是地址解析协议，简单语言解释一下工作原理。**

(1)首先，每个主机都会在自己的ARP缓冲区中建立一个ARP列表，以表示IP地址和MAC地址之间的对应关系。

（2）当源主机要发送数据时，首先检查ARP列表中是否有对应IP地址的目的主机的MAC地址，如果有，则直接发送数据，如果没有，就向本网段的所有主机发送ARP数据包，该数据包包括的内容有：源主机IP地址，源主机MAC地址，目的主机的IP地址。

（3）当本网络的所有主机收到该ARP数据包时，首先检查数据包中的IP地址是否是自己的IP地址，如果不是，则忽略该数据包，如果是，则首先从数据包中取出源主机的IP和MAC地址写入到ARP列表中，如果已经存在，则覆盖，然后将自己的MAC地址写入ARP响应包中，告诉源主机自己是它想要找的MAC地址。

（4）源主机收到ARP响应包后。将目的主机的IP和MAC地址写入ARP列表，并利用此信息发送数据。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

广播发送ARP请求，单播发送ARP响应。

**长连接**：

       所谓长连接，指在一个TCP连接上可以连续发送多个数据包，在TCP连接保持期间，如果没有数据包发送，需要双方发检测包以维持此连接，一般需要自己做在线维持（不发生RST包和四次挥手）。

连接→数据传输→保持连接(心跳)→数据传输→保持连接(心跳)→……→关闭连接（一个TCP连接通道多个读写通信）；      这就要求长连接在没有数据通信时，定时发送数据包(心跳)，以维持连接状态；

      TCP保活功能，保活功能主要为服务器应用提供，服务器应用希望知道客户主机是否崩溃，从而可以代表客户使用资源。如果客户已经消失，使得服务器上保留一个半开放的连接，而服务器又在等待来自客户端的数据，则服务器将应远等待客户端的数据，保活功能就是试图在服务器端检测到这种半开放的连接。

       如果一个给定的连接在两小时内没有任何的动作，则服务器就向客户发一个探测报文段，客户主机必须处于以下4个状态之一：

1. 客户主机依然正常运行，并从服务器可达。客户的TCP响应正常，而服务器也知道对方是正常的，服务器在两小时后将保活定时器复位。
2. 客户主机已经崩溃，并且关闭或者正在重新启动。在任何一种情况下，客户的TCP都没有响应。服务端将不能收到对探测的响应，并在75秒后超时。服务器总共发送10个这样的探测 ，每个间隔75秒。如果服务器没有收到一个响应，它就认为客户主机已经关闭并终止连接。
3. 客户主机崩溃并已经重新启动。服务器将收到一个对其保活探测的响应，这个响应是一个复位，使得服务器终止这个连接。
4. 客户机正常运行，但是服务器不可达，这种情况与2类似，TCP能发现的就是没有收到探查的响应。

**短连接**：

      短连接是指通信双方有数据交互时，就建立一个TCP连接，数据发送完成后，则断开此TCP连接（管理起来比较简单，存在的连接都是有用的连接，不需要额外的控制手段）；

     连接→数据传输→关闭连接；

**ICMP协议**

ICMP是InternetControl Message Protocol，因特网控制报文协议。它是TCP/IP协议族的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由器是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。ICMP报文有两种：差错报告报文和询问报文。

**15、运输层协议与网络层协议的区别？**

网络层协议负责的是提供主机间的逻辑通信

       运输层协议负责的是提供进程间的逻辑通信

# HTTP1.0 HTTP 1.1 HTTP 2.0主要区别

影响一个 HTTP 网络请求的因素主要有两个：**带宽和延迟。**

* **带宽：**如果说我们还停留在拨号上网的阶段，带宽可能会成为一个比较严重影响请求的问题，但是现在网络基础建设已经使得带宽得到极大的提升，我们不再会担心由带宽而影响网速，那么就只剩下延迟了。
* **延迟：**
  + 浏览器阻塞（HOL blocking）：浏览器会因为一些原因阻塞请求。浏览器对于同一个域名，同时只能有 4 个连接（这个根据浏览器内核不同可能会有所差异），超过浏览器最大连接数限制，后续请求就会被阻塞。
  + DNS 查询（DNS Lookup）：浏览器需要知道目标服务器的 IP 才能建立连接。将域名解析为 IP 的这个系统就是 DNS。这个通常可以利用DNS缓存结果来达到减少这个时间的目的。
  + 建立连接（Initial connection）：HTTP 是基于 TCP 协议的，浏览器最快也要在第三次握手时才能捎带 HTTP 请求报文，达到真正的建立连接，但是这些连接无法复用会导致每次请求都经历三次握手和慢启动。三次握手在高延迟的场景下影响较明显，慢启动则对文件类大请求影响较大。

**HTTP1.0和HTTP1.1的一些区别**

* **缓存处理**，在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略例如Entity tag，If-Unmodified-Since, If-Match, If-None-Match等更多可供选择的缓存头来控制缓存策略。
* **带宽优化及网络连接的使用/允许传对象一部分**，HTTP1.0中，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点续传功能，HTTP1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是206（Partial Content），这样就方便了开发者自由的选择以便于充分利用带宽和连接。
* **错误通知的管理**，在HTTP1.1中新增了24个错误状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示服务器上的某个资源被永久性的删除。
* **Host头处理**，在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）。
* **长连接（即tcp长连接）**，HTTP 1.1支持长连接（PersistentConnection）和请求的流水线（Pipelining）处理，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟，在HTTP1.1中默认开启Connection： keep-alive，一定程度上弥补了HTTP1.0每次请求都要创建连接的缺点。

**四、HTTPS与HTTP的一些区别**

* HTTPS协议需要到CA申请证书，一般免费证书很少，需要交费。
* HTTP协议运行在TCP之上，所有传输的内容都是明文，HTTPS运行在SSL/TLS之上，SSL/TLS运行在TCP之上，所有传输的内容都经过加密的。
* HTTP和HTTPS使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
* HTTPS可以有效的防止运营商劫持，解决了防劫持的一个大问题。

**八、HTTP2.0和HTTP1.X相比的新特性**

* **新的二进制格式**（Binary Format），HTTP1.x的解析是基于文本。基于文本协议的格式解析存在天然缺陷，文本的表现形式有多样性，要做到健壮性考虑的场景必然很多，二进制则不同，只认0和1的组合。基于这种考虑HTTP2.0的协议解析决定采用二进制格式，实现方便且健壮。
* **多路复用**（MultiPlexing），即连接共享，即每一个request都是是用作连接共享机制的。一个request对应一个id，这样一个连接上可以有多个request，每个连接的request可以随机的混杂在一起，接收方可以根据request的 id将request再归属到各自不同的服务端请求里面。
* **header压缩**，如上文中所言，对前面提到过HTTP1.x的header带有大量信息，而且每次都要重复发送，HTTP2.0使用encoder来减少需要传输的header大小，通讯双方各自cache一份header fields表，既避免了重复header的传输，又减小了需要传输的大小。
* **服务端推送**（server push），同SPDY一样，HTTP2.0也具有server push功能。

**HTTP2.0的多路复用和HTTP1.X中的长连接复用有什么区别？**

* HTTP/1.\* 一次请求-响应，建立一个连接，用完关闭；每一个请求都要建立一个连接；
* HTTP/1.1 Pipeling解决方式为，若干个请求排队串行化单线程处理，后面的请求等待前面请求的返回才能获得执行机会，一旦有某请求超时等，后续请求只能被阻塞，毫无办法，也就是人们常说的线头阻塞；
* HTTP/2多个请求可同时在一个连接上并行执行。某个请求任务耗时严重，不会影响到其它连接的正常执行；

**2开头 （请求成功）表示成功处理了请求的状态代码。**

**3开头 （请求被重定向）表示要完成请求，需要进一步操作。 通常，这些状态代码用来重定向。**

**4开头 （请求错误）这些状态代码表示请求可能出错，妨碍了服务器的处理。**

**5开头（服务器错误）这些状态代码表示服务器在尝试处理请求时发生内部错误。 这些错误可能是服务器本身的错误，而不是请求出错。**