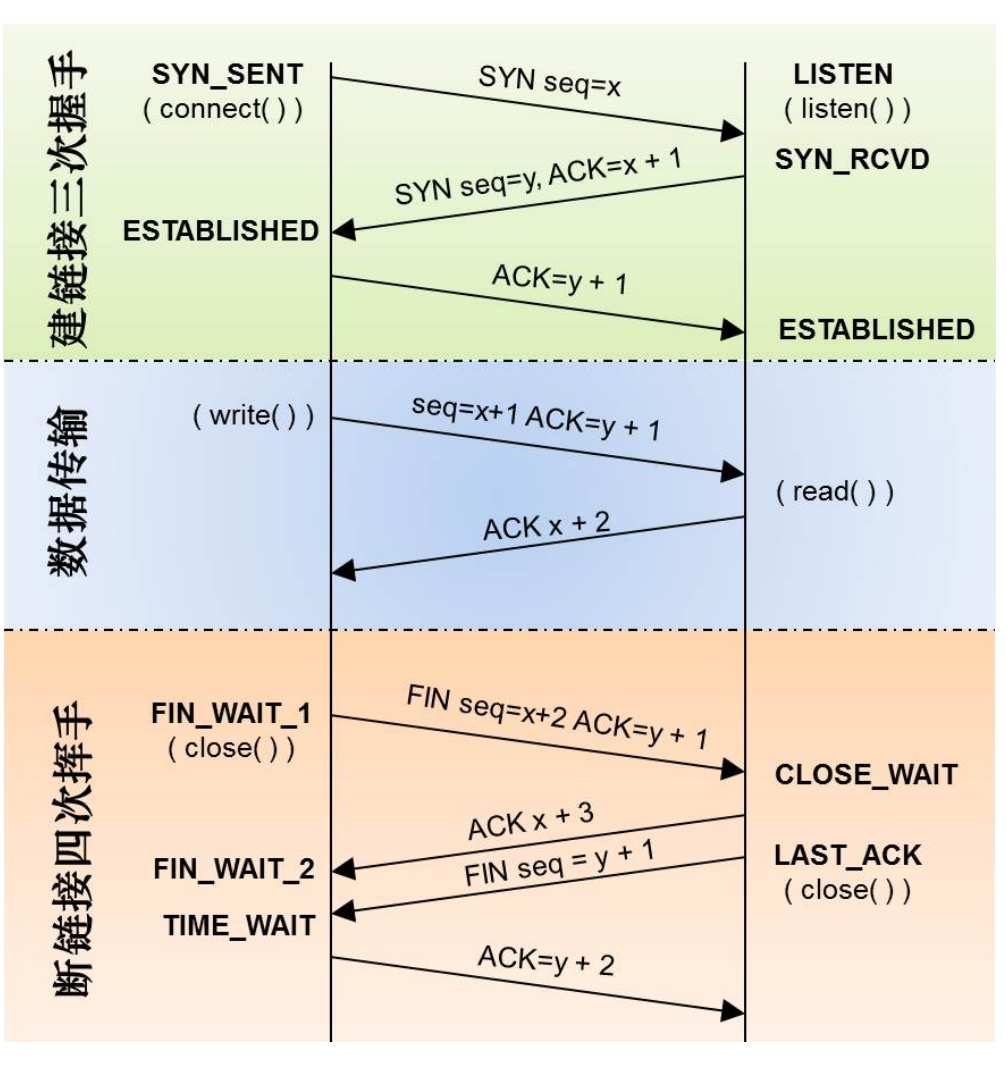
1.http

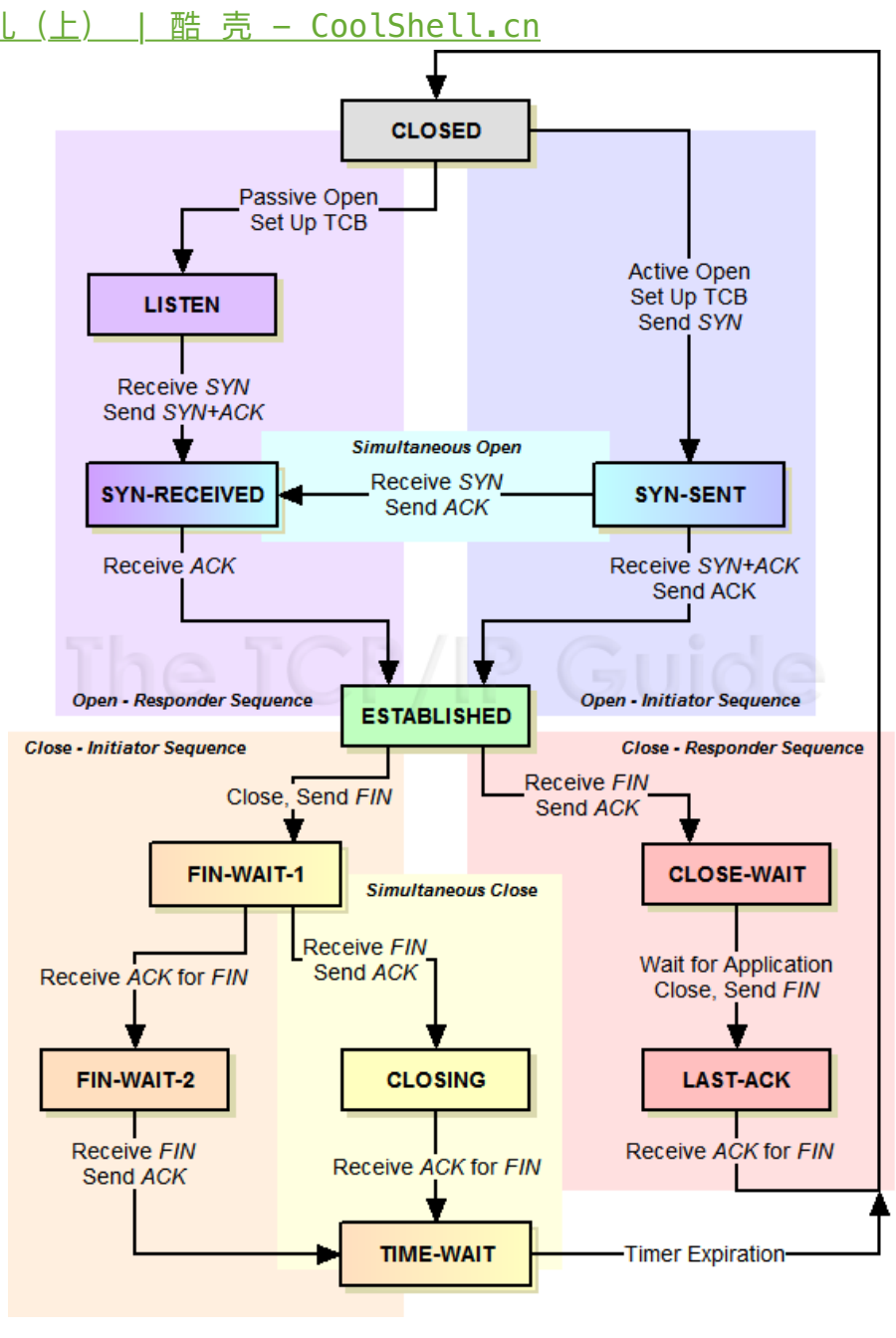
HTTP协议（七）Cookie – 小坦克 – 博客园 HTTP协议（五）代理 – 小坦克 – 博客园
HTTP协议（四）缓存 – 小坦克 – 博客园 HTTP协议（三）压缩 – 小坦克 – 博客园
HTTP协议（二）基本认证 – 小坦克 – 博客园 HTTP协议详解 – 小坦克 – 博客园

HTTP协议小结 – xsfelvis – 博客频道 – CSDN.NET

2.tcp

TCP三次握手及其背后的缺陷 – xsfelvis – 博客频道 – CSDN.NET (2)TCP的流量控制和拥塞控制_简单_新浪博客
TCP 的那些事儿（下） | 酷 壳 – CoolShell.cn TCP 的那些事儿（上） | 酷 壳 – CoolShell.cn





可靠传输，重传：超时重传，快速重传

TCP标志位

网络

OSI七层模型：将服务、接口和协议这三个概念区分开来，通过七个层次化的结构模型使不同的系统不同的网络之间实现可靠的通讯。

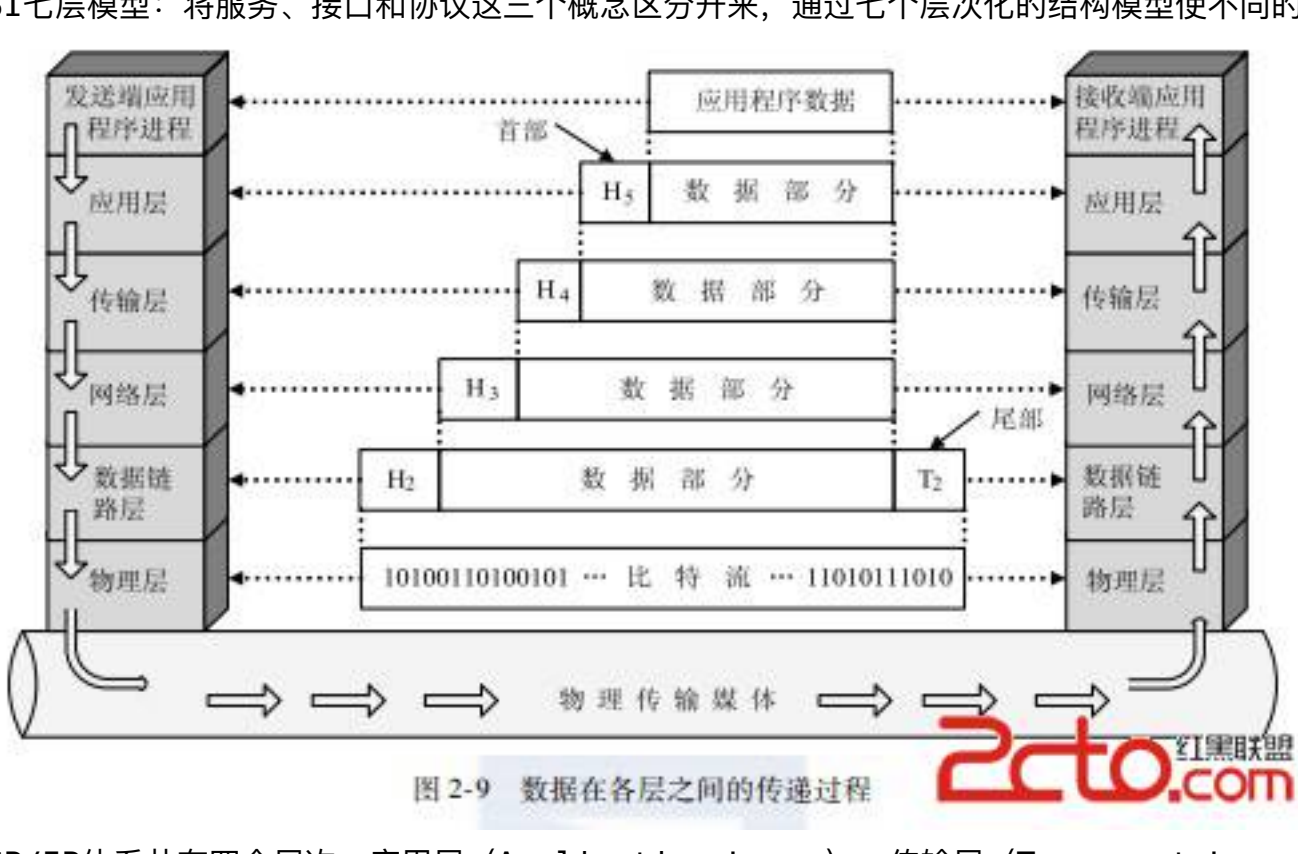


图 2-9 数据在各层之间的传递过程

TCP/IP体系共有四个层次：应用层（Application Layer）、传输层（Transport Layer）、网络互连层（Internet Layer）和网络接口层（Host-to-Network Layer）

（1）网络接口层（Host-to-Network Layer）–> 接收和发送数据报

网络接口层主要负责将数据报发送到网络传输介质上以及从网络上接收TCP/IP数据报，相当于OSI参考模型中的物理层和数据链路层。在实际中，先后流行的以太网、令牌环网、ATM、帧中继等都可视为其底层协议。它将发送的信息组帧，并通过物理层向选定网络发送，或者从网络上接收物理帧，将去除数据控制信息的IP数据报交给网络互连层。

（2）网络互连层（Internet Layer）–> 数据报封装和路由寻址功能

网络互连层的主要功能是寻址和对数据报的封装以及重要的路由选择功能。这些功能大部分都是由IP协议来完成的，再加上地址解析协议（Address Resolution Protocol，ARP）、因特网控制报文协议（Internet Control Message Protocol，ICMP）等协议从旁协助，所以IP协议是本层众多实体中的核心。下面简单介绍这几个协议。

网际协议（Internet Protocol，IP）。该协议是一个无连接的协议，主要负责将数据报从源结点转发到目的结点。也就是说，IP协议通过对每个数据报中都有的源地址和目的地址进行分析，然后进行路由选择（即选择一条到达目标的最佳路径），最后再转发到目的地。需要注意的是：IP协议只是负责对数据进行转发，并不对数据进行检查。也就是说，它不负责数据的可靠性，这样设计的主要目的是提高IP协议传送和转发数据的效率。

地址解析协议（Address Resolution Protocol，ARP）。该协议主要负责将TCP/IP网络中的IP地址解析和转换成计算机的物理地址，以便于物理设备（如网卡）按该地址来接收数据。

反向地址解析协议（Reverse Address Resolution Protocol，RARP）。该协议的作用与ARP的作用相反，它主要负责将设备的物理地址解析和转换成IP地址。

因特网控制报文协议（Internet Control Message Protocol，ICMP）。该协议主要负责发送和传递包含控制信息的数据报，这些控制信息包括哪台计算机出了什么错误、网络路由出现了什么错误等内容。

（3）传输层（Transport Layer）–> 应用进程间的端到端通信

传输层主要负责在应用进程之间的"端到端"通信，即从某个应用进程传输到另外一个应用进程。它与OSI参考模型的传输层功能类似，也对高层屏蔽了底层通信网络的实现细节。

传输层在某一时刻可能要同时为多个不同的应用进程服务，因此为了识别不同的应用进程，传输层在每一个分组中必须增加用于识别信源和信宿的应用程序的标识，同时，每一个分组还要附带校验和，以保证接收端能校验分组的正确性，这样可以将数据报发送到合适的应用进程。这个增加的标识称为端口（port）或者端口号（port ID）。

TCP/IP体系结构的传输层包含两个主要协议，即传输控制协议（Transport Control Protocol，TCP）和用户数据报协议（User Datagram Protocol，UDP）。这两个协议分别应用于有不同要求的应用进程。

TCP协议是一种可靠的、面向连接的协议，保证通信主机之间有可靠的字节流传输，完成流量控制功能，协调收发双方的发送与接收速度，达到正确传输的目的。

UDP是一种不可靠、无连接的协议，其特点是协议简单、额外开销小、效率较高，但是不能保证传输是否正确。

（4）应用层（Application Layer）–> 不同协议

应用层是TCP/IP的最高层，它包括了多种高层协议，并且总有新的协议加入。与OSI的应用层类似，它是直接为应用进程服务的一层。即当不同的应用进程数据通信或者数据交换时，就去调用应用层的不同协议实体，让这些实体去调用TCP或者UDP层服务来进行网络传输。

与OSI不同，TCP/IP中包含了许多具体的应用层协议。

简单邮件传输协议（Simple Mail Transportation Protocol，SMTP）：该协议主要用于在电子邮件服务器之间传输电子邮件。

域名系统（Domain Name System，DNS）：该协议用于域名与IP地址之间的转换。

超文本传输协议（Hypertext Transportation Protocol，HTTP）：该协议是为因特网上传输和处理超文本或者WWW（World Wide Web）页面而服务的应用层协议。当用户浏览网页时，就需要用到HTTP协议。

文件传输协议（File Transportation Protocol，FTP）。该协议是为网络中传输文件进程服务的。所谓传输文件，是指将文件从一台计算机通过网络复制到另一台计算机中。

远程终端协议（Telnet）。该协议是用于远程登录网络主机的一个应用层协议。

简单网络管理协议（Simple Network Management Protocol，SNMP）。该协议用于在控制台与网络设备（如路由器、交换机等）之间交换网络管理信息。

网络拓扑结构：总线、星型、环型、树型、网状结构和混合型结构
按照覆盖范围分为：局域网，城域网，广域网
按照传播方式分为：广播网络和点对点网络
广播式网络是指网络中的计算机或者设备使用一个共享的通信介质进行数据传播，网络中的所有结点都能收到任一结点发出的数据信息。
单播：采用一对一的发送形式，将数据发送给网络所有目的节点。
组播：采用一对一组的发送形式，将数据发送给网络中的某一组主机。
广播：采用一对所有的发送形式，将数据发送给网络中所有目的节点。 （与单播区别？）
点对点式网络是两个结点之间的通信方式是点对点的。如果两台计算机之间没有直接连接的线路，那么它们之间的分组传输就要通过中间结点的接收、存储、转发，直至目的结点。点-点传播方式主要应用于WAN中，通常采用的拓扑结构有：星型、环型、树型、网状型。

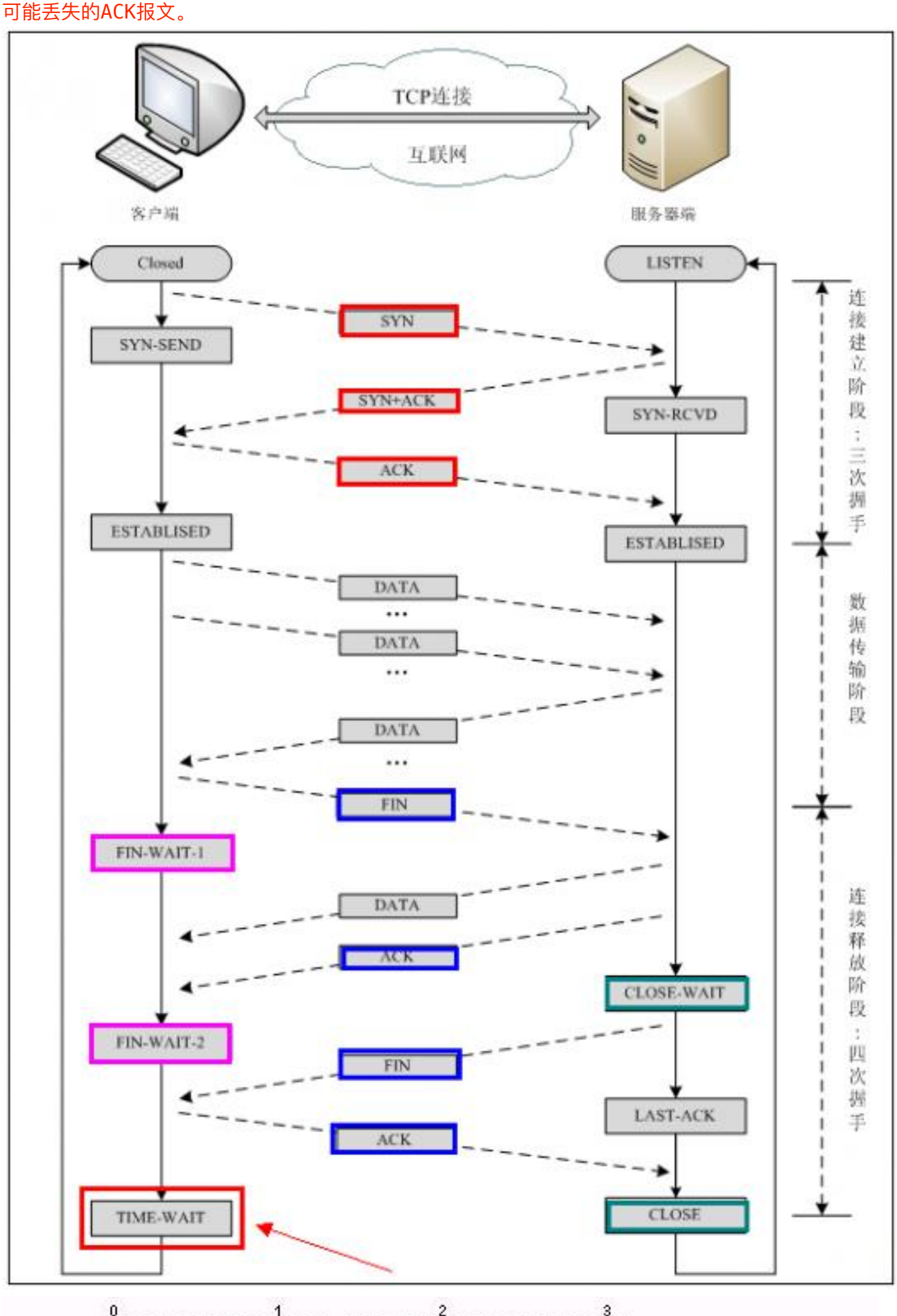
tcp（传输控制协议）三次握手和四次挥手过程 TCP三次握手及其背后的缺陷 – xsfelvis – 博客频道 – CSDN.NET

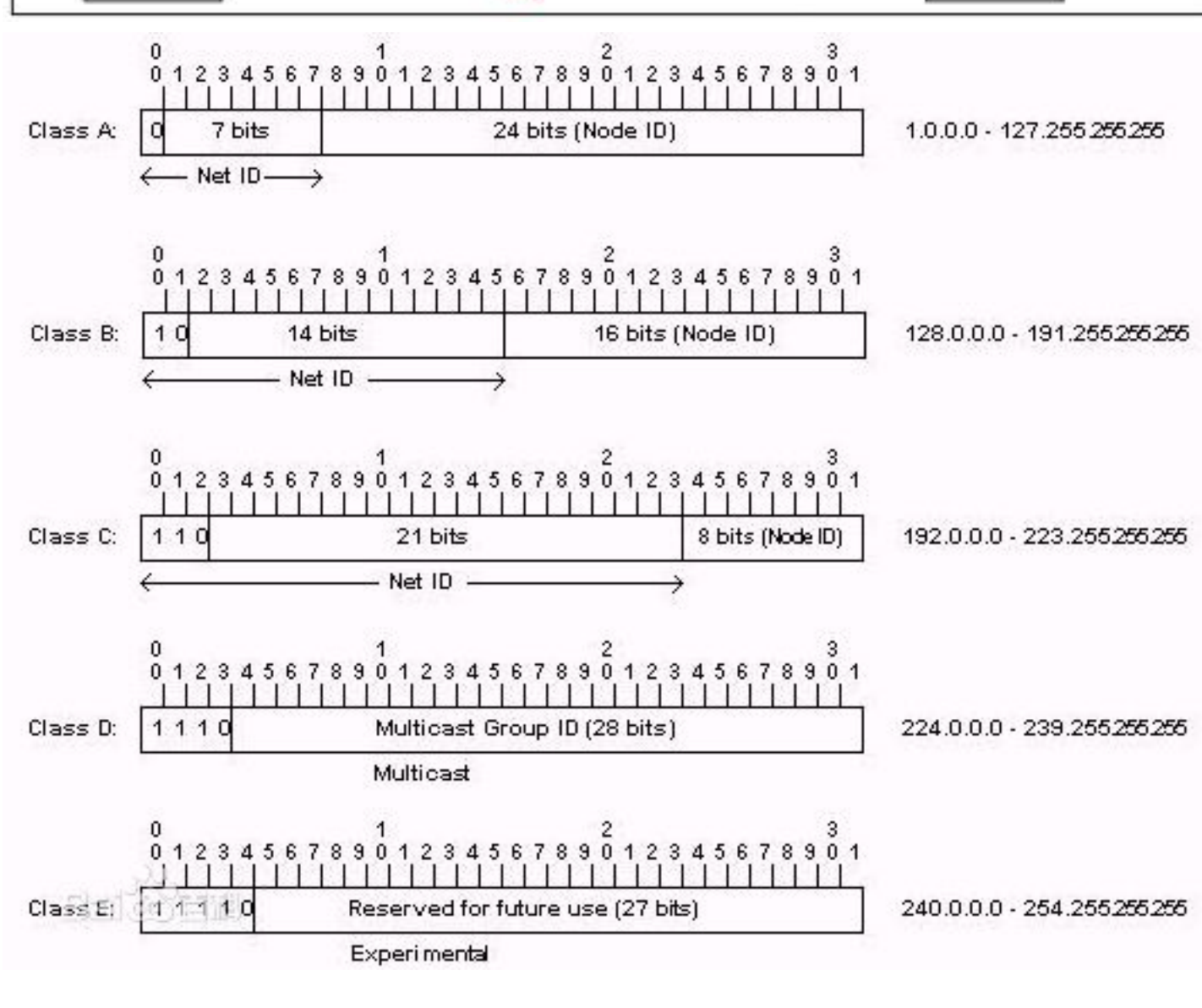
建立：首先Client端发送连接请求报文，Server端接收连接请求后回复ACK报文，并为这次连接分配资源。Client端接收到ACK报文后也向Server端发生ACK报文，并分配资源，这样TCP连接就建立了。

断开：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连接就这样关闭了！

**为什么连接的时候是三次握手，关闭的时候却是四次握手?
因为当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)才能返回到CLOSE状态?
这是因为虽然双方都同意关闭连接了，而且握手的4个报文也都协调和发送完毕，按理可以直接回到CLOSED状态（就好比从SYN_SEND状态到ESTABLISH状态那样）；但是因为我们必须要假想网络是不可靠的，你无法保证你最后发送的ACK报文会一定被对方收到，因此对方处于LAST_ACK状态下的SOCKET可能会因为超时未收到ACK报文，而重发FIN报文，所以这个TIME_WAIT状态的作用就是用来重发可能丢失的ACK报文。



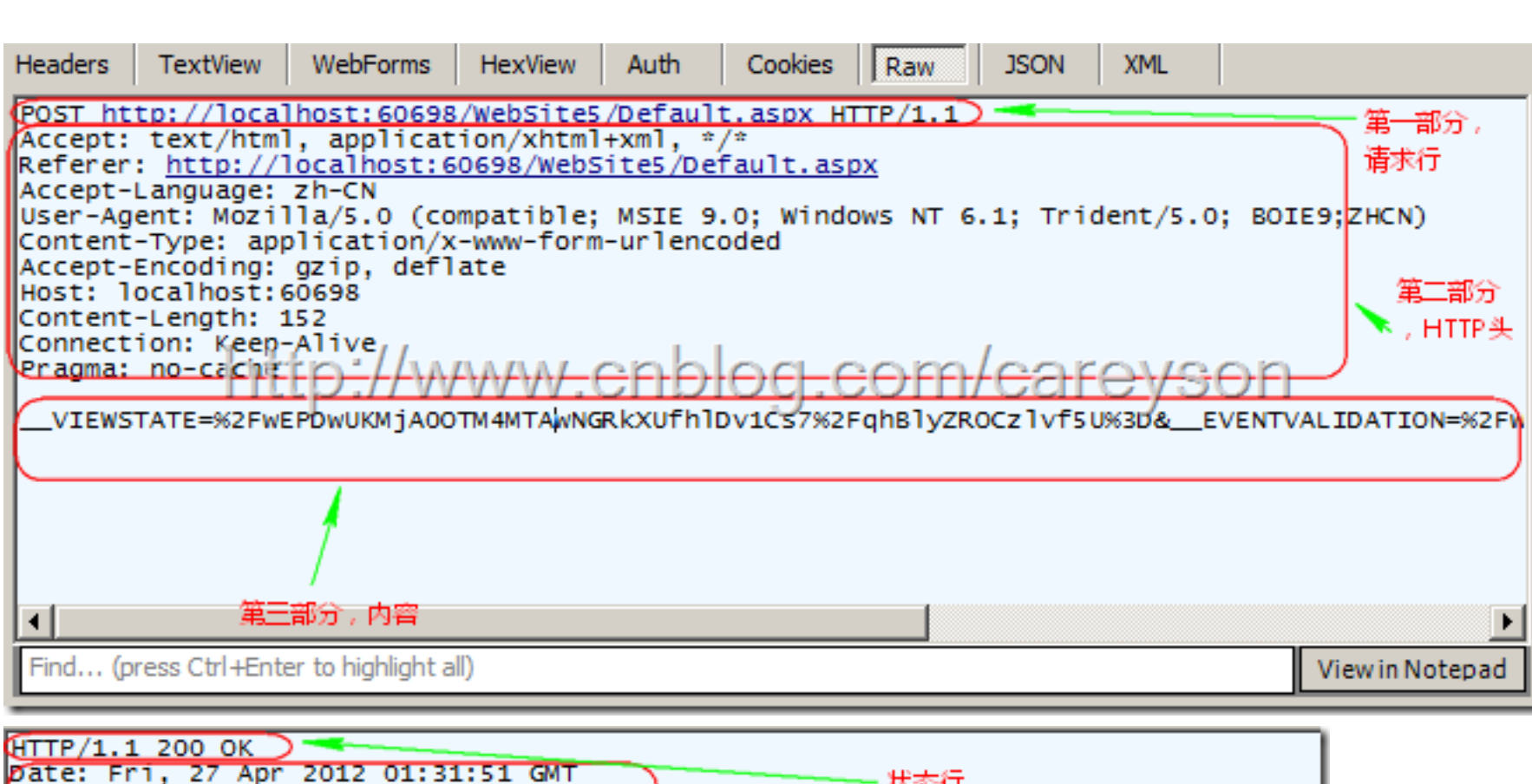


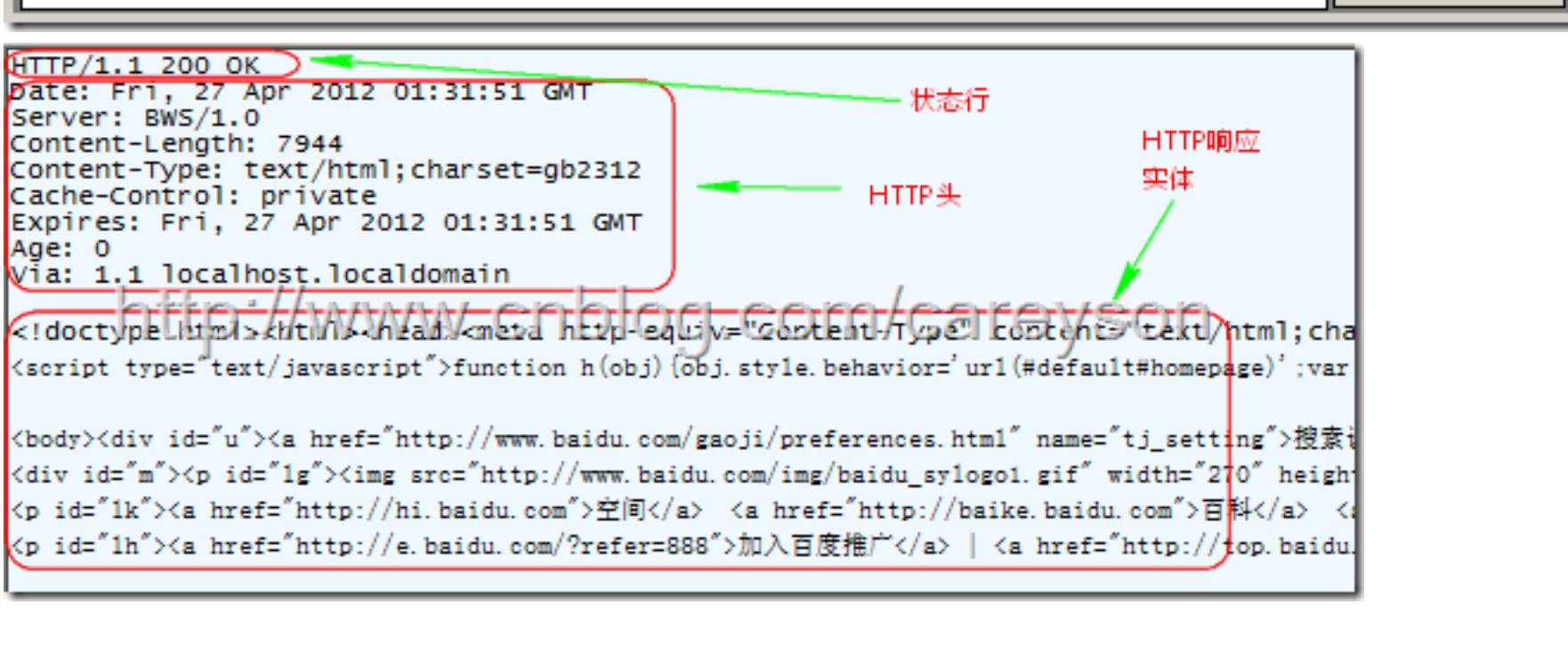
**TCP滑动窗口：TCP滑动窗口用来暂存两台主机间要传送的数据，有点类似CACHE。
每个TCP/IP主机有两个滑动窗口：一个用于接收数据，另一个用于发送数据。
通过每个TCP传输的字段指定顺序号，以获得可靠性。如果一个分段被分解成几个小段，接收主机会知道是否所有小段都已收到。通过发送应答，用以确认别的主机收到了数据。对于发送的每一个小段，接收主机必须在一个指定的时间返回一个确认。如果发送者未收到确认，数据会被重新发送；如果收到的数据包损坏，接收主机会舍弃它，因为确认未被发送，发送者会重新发送分段。

IP地址由网络号和主机号组成。IP地址有两种表示形式：二进制表示(1和0太多了就搞不清)和点分十进制表示。
每个IP地址的长度为4字节，由四个8位域组成，我们通常称之为八位体。八位体由句点.分开，表示为一个0-255之间的十进制数。
在分配网络号和主机号时应遵守以下几条准则:
(1)网络号不能为127。大家知道该标识号被保留作回路及诊断功能，还记得平时ping 127.0.0.1?
(2)不能将网络号和主机号的各位均置1。如果每一位都是1的话，该地址会被解释为网内广播而不是一个主机号(TCP/IP是一个可广播的协议嘛)
(3)相应于上面一条，各位不能均置0，否则该地址被解释为"就是本网络"。
(4)对于本网络来说，主机号应该是唯一。(否则会出现IP地址已分配或有冲突之类的错误)

Http协议 格式 HTTP协议漫谈–程序猿 HTTP协议小结 – xsfelvis – 博客频道 – CSDN.NET
http请求包含的内容：1.请求行 2.HTTP头 3.内容
第一部分请求行写法是固定的，由三部分组成，第一部分是请求方法，第二部分是请求网址，第三部分是HTTP版本。
第二部分HTTP头在HTTP请求可以是3种HTTP头：1.请求头(request header) 2.普通头(general header) 3.实体头(entity header)。通常来说，由于Get请求往往不包含内容实体，因此也不会有实体头。
第三部分内容只在POST请求中存在，因为GET请求并不包含任何实体。

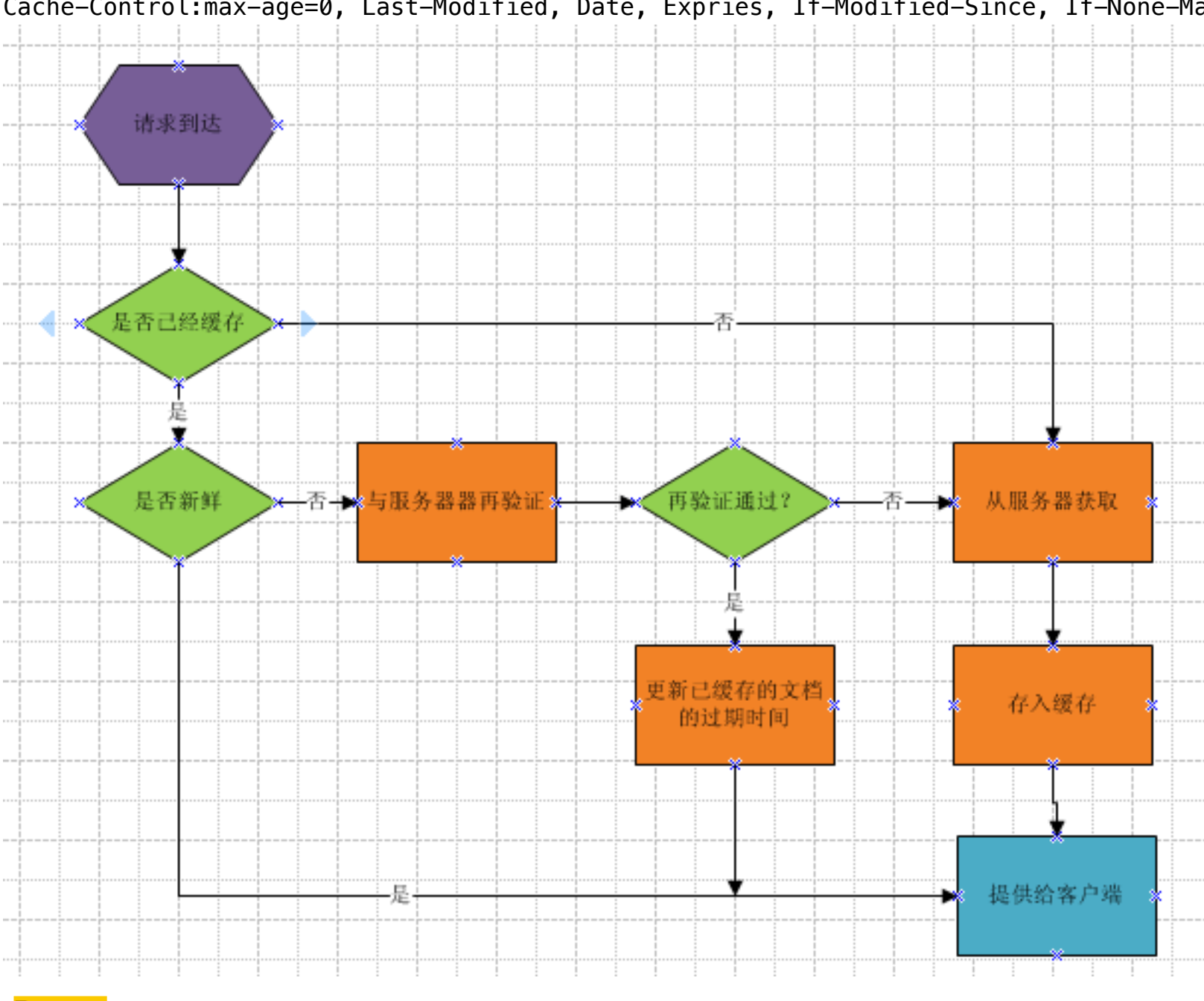
http响应包含的内容：1.状态行 2.HTTP头 3.返回内容
第一部分是HTTP版本，第二部分是响应状态码，第三部分是状态码的描述，因此也可以把第二和第三部分看成一个部分。
信息类（100-199）响应成功（200-299）重定向类（300-399）客户端错误类（400-499）服务端错误类（500-599）
第二部分包含的头包括：1.响应头(response header) 2.普通头(general header) 3.实体头(entity header)。
第三部分HTTP响应内容就是HTTP请求所请求的信息。这个信息可以是一个HTML，也可以是一个图片。





http请求头
HTTP头并不是严格要求的，仅仅是一个标签，如果浏览器可以解析就会按照某些标准（比如浏览器自身标准，W3C的标准）去解释这个头，否则不识别的头就会被浏览器无视。对服务器也是同理。
通用头(General header)
通用头即可以包含在HTTP请求中，也可以包含在HTTP响应中。通用头的作用是描述HTTP协议本身。比如描述HTTP是否持久连接的Connection头，HTTP发送日期的Date头，描述HTTP所在TCP连接时间的Keep-Alive头，用于缓存控制的Cache-Control头等。
实体头(Entity header)
实体头是那些描述HTTP信息的头。既可以出现在HTTP POST方法的请求中，也可以出现在HTTP响应中。比如Content-Type和Content-length都是描述实体的类型和大小的头都属于实体头。其它还有用于描述实体的Content-Language，Content-MD5，Content-Encoding以及控制实体缓存的Expires和Last-Modifies头等。
请求头(HTTP Request Header)
请求头是那些由客户端发往服务端以便帮助服务端更好的满足客户端请求的头。请求头只能出现在HTTP请求中。比如告诉服务器只接收某种响应内容的Accept头，发送Cookies的Cookie头，显示请求主机域的HOST头，用于缓存的If-Match，If-Match-Since，If-None-Match头，用于只取HTTP响应信息中部分信息的Range头，用于附属HTML相关请求引用的Referer头等。
响应头(HTTP Response Header)
HTTP响应头是那些描述HTTP响应本身的头，这里面并不包含描述HTTP响应中第三部分也就是HTTP信息的头（这部分由实体头负责）。比如说定时刷新的Refresh头，当遇到503错误时自动重试的Retry-After头，显示服务器信息的Server头，设置COOKIE的Set-Cookie头，告诉客户端可以部分请求的Accept-Ranges头等。

http缓存机制的实现HTTP协议：缓存_知识库_博客园
Cache-Control:max-age=0, Last-Modified, Date, Expries, If-Modified-Since, If-None-Match



Request

Cache-Control: max-age=0	以秒为单位
If-Modified-Since: Mon, 19 Nov 2012 08:38:01 GMT	缓存文件的最后修改时间。
If-None-Match: "0693f67a67cc1:0"	缓存文件的Etag值
Cache-Control: no-cache	不使用缓存
Pragma: no-cache	不使用缓存

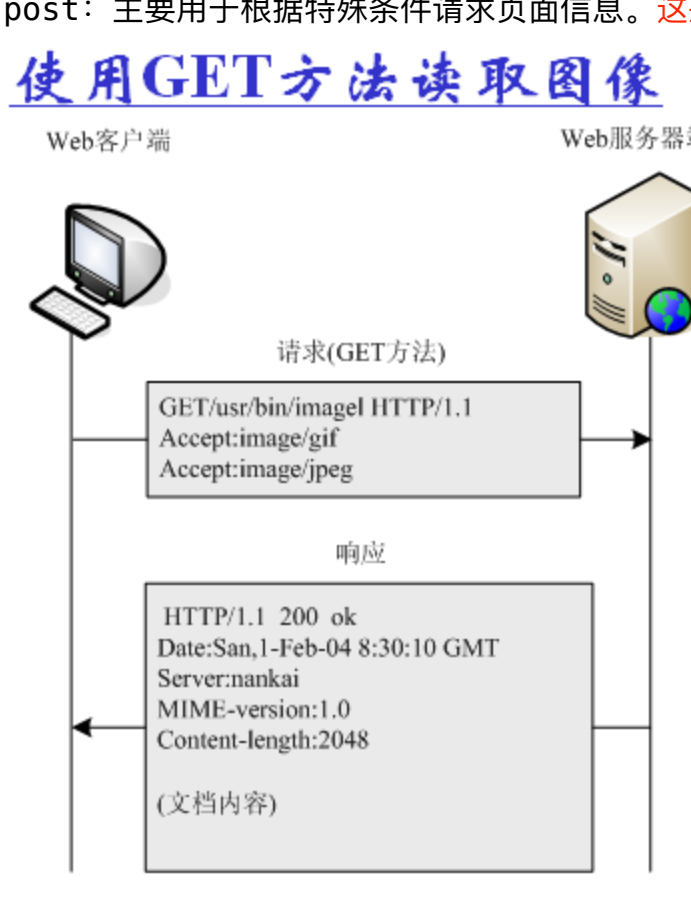
Response

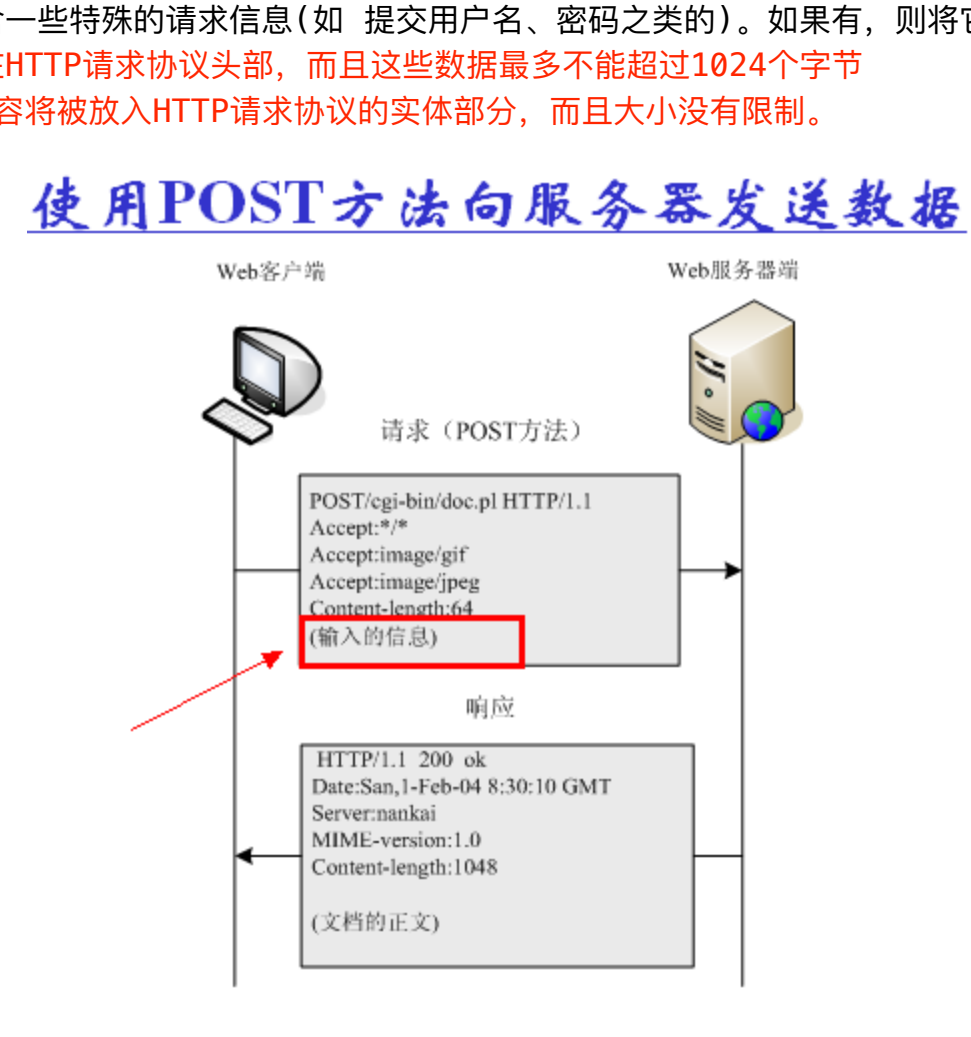
Cache-Control: public	响应被缓存，并且在多用户间共享， （公有缓存和私有缓存的区别，请看另一节）
Cache-Control: private	响应只能作为私有缓存，不能在用户之间共享
Cache-Control:no-cache	提醒浏览器要从服务器提取文档进行验证
Cache-Control:no-store	绝对禁止缓存（用于机密，敏感文件）
Cache-Control: max-age=60	60秒之后缓存过期（相对时间）
Date: Mon, 19 Nov 2012 08:39:00 GMT	当前response发送的时间
Expires: Mon, 19 Nov 2012 08:40:01 GMT	缓存过期的时间（绝对时间）
Last-Modified: Mon, 19 Nov 2012 08:38:01 GMT	服务器端文件的最后修改时间
ETag: "20b1add7ec1cd1:0"	服务器端文件的Etag值

http和https的区别 HTTP协议小结 – xsfelvis – 博客频道 – CSDN.NET
HTTPS(Secure Hypertext Transfer Protocol)安全超文本传输协议 它是一个安全通信通道，它基于HTTP开发，用于在客户计算机和服务器之间交换信息。它使用安全套接字层(SSL)进行信息交换，简单来说它是HTTP的安全版。
https协议需要到ca申请证书，一般免费证书很少，需要交费。
http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议。
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。
http的连接很简单，是无状态的。
HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全

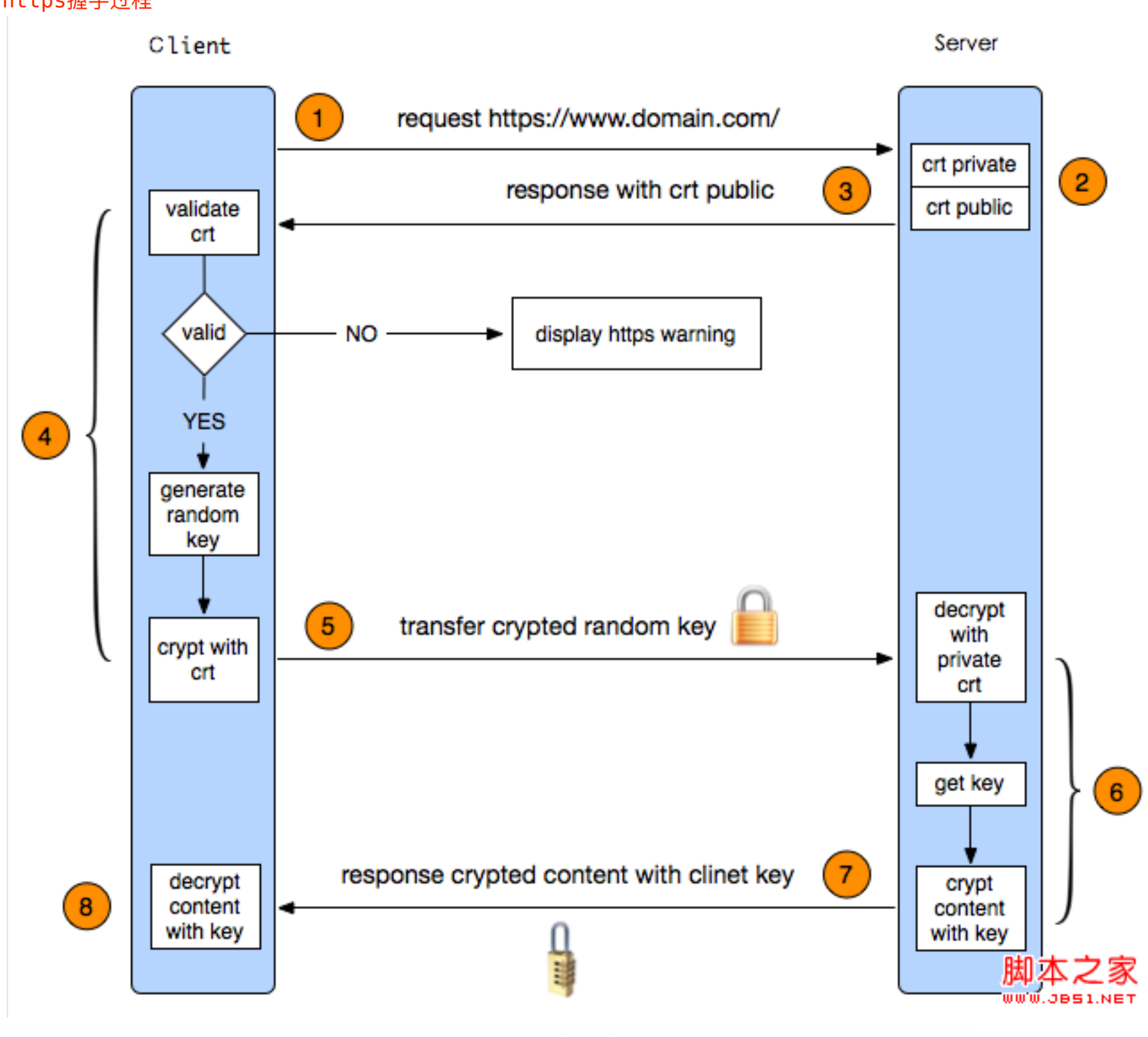
SPDY：Google开发的基于TCP的应用层协议，用以最小化网络延迟，提升网络速度，优化用户的网络使用体验。SPDY并不是一种用于替代HTTP的协议，而是对HTTP协议的增强。新协议的功能包括数据流的多路复用、请求优先级以及HTTP报头压缩。
HTTP –> SPDY –> SSL –> TCP
1.单个TCP连接支持并发的HTTP请求。
2.压缩报头和去掉不必要的头部来减少当前HTTP使用的带宽。
3.定义一个容易实现，在服务器端高效率的协议。通过减少边缘情况、定义易解析的消息格式来减少HTTP的复杂性。
4.强制使用SSL，让SSL协议在现存的网络设施下有更好的安全性和兼容性。
5.允许服务器在需要时发起对客户端的连接并推送数据。
(1)复用流：SPDY允许在一个连接上无限制并发流。因为请求在一个通道上，TCP效率更高：更少的网络连接，发出更少更密集的数据包。
(2)请求优先级：虽然无限的并发流解决了序列化的问题，但他们引入了另一个问题：如果带宽通道受限制， 客户端可能会因防止堵塞通道而阻止请求。为了克服这个问题，SPDY实行请求优先级：客户端从服务器端请求它希望的项目数量，并为每个请求分配一个优先级。这可以防止在网络通道被非关键资源堵塞时，高优先级的请求被挂起。
(3)HTTP报头压缩：SPDY压缩请求和响应HTTP报头，从而减少传输的数据包数量和字节数。

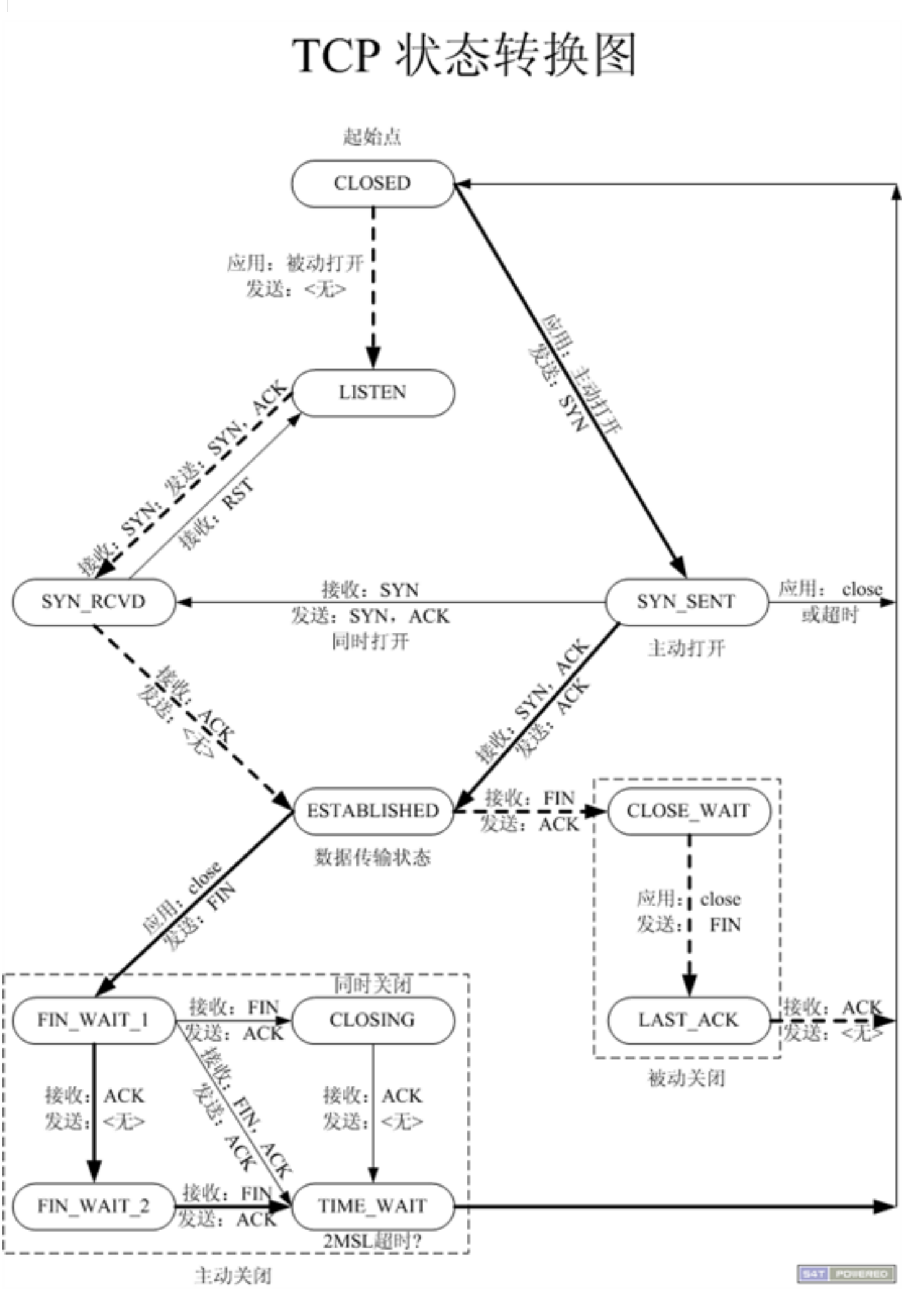
http六种请求方法：get，post，update，delete，head，options
get和post的区别
get：主要用于简单的获取页面信息，同时也可以包含一些特殊的请求信息(如 提交用户名、密码之类的)。如果有，则将它附着在URL尾部，以？隔开，空格转化为"+"。实际上是将其放在HTTP请求协议头部，而且这些数据最多不能超过1024个字节
post：主要用于根据特殊条件请求页面信息。这些内容将被放入HTTP请求协议的实体部分，而且大小没有限制。





聊聊HTTPS与Android安全 – 北漂周的专栏（微信：stchou_zst） – 博客频道 – CSDN.NET
聊聊HTTPS与Android安全（二） – 北漂周的专栏（微信：stchou_zst） – 博客频道 – CSDN.NET
https握手过程





中间人攻击：中间人获取server发给client的公钥，自己伪造一对公私钥，然后将伪造自己让client以为它是server，然后将伪造的公钥发给client，并拦截client发给server的密文，用伪造的私钥即可得到client发出去的内容，最后用真实的公钥对内容进行加密发给server。
解决办法：数字证书，可信任的中间人 证书链

TCP状态转换图TCP状态转换图 – 李大嘴 – 博客园
1、建立连接协议（三次握手）
（1）客户端发送一个带SYN标志的TCP报文到服务器。这是三次握手过程中的报文1。
（2） 服务器端回应客户端的，这是三次握手中的第2个报文，这个报文同时带ACK标志和SYN标志。因此它表示对刚才客户端SYN报文的回应；同时又标志SYN给客户端，询问客户端是否准备好进行数据通讯。
（3） 客户必须再次回应服务段一个ACK报文，这是报文段3。
2、连接终止协议（四次握手）
由于TCP连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务后就能发送一个FIN来终止这个方向的连接。收到一个 FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。
（1） TCP客户端发送一个FIN，用来关闭客户到服务器的数据传送（报文段4）。
（2） 服务器收到这个FIN，它发回一个ACK，确认序号为收到的序号加1（报文段5）。和SYN一样，一个FIN将占用一个序号。
（3） 服务器关闭客户端的连接，发送一个FIN给客户端（报文段6）。
（4） 客户段发回ACK报文确认，并将确认序号设置为收到序号加1（报文段7）。
CLOSED: 这个没什么好说的了，表示初始状态。
LISTEN: 这个也是非常容易理解的一个状态，表示服务器端的某个SOCKET处于监听状态，可以接受连接了。
SYN_RCVD: 这个状态表示接受到了SYN报文，在正常情况下，这个状态是服务器端的SOCKET在建立TCP连接时的三次握手会话过程中的一个中间状态，很短暂，基本 上用netstat你是很难看到这种状态的，除非你特意写了一个客户端测试程序，故意将三次TCP握手过程中最后一个ACK报文不予发送。因此这种状态 时，当收到客户端的ACK报文后，它会进入到ESTABLISHED状态。
SYN_SENT: 这个状态与SYN_RCVD遥想呼应，当客户端SOCKET执行CONNECT连接时，它首先发送SYN报文，因此也随即它会进入到了SYN_SENT状 态，并等待服务端的发送三次握手中的第2个报文。SYN_SENT状态表示客户端已发送SYN报文。
ESTABLISHED：这个容易理解了，表示连接已经建立了。
FIN_WAIT_1: 这个状态要好好解释一下，其实FIN_WAIT_1和FIN_WAIT_2状态的真正含义都是表示等待对方的FIN报文。而这两种状态的区别 是：FIN_WAIT_1状态实际上是当SOCKET在ESTABLISHED状态时，它想主动关闭连接，向对方发送了FIN报文，此时该SOCKET即 进入到FIN_WAIT_1状态。而当对方回应ACK报文后，则进入到FIN_WAIT_2状态，当然在实际的正常情况下，无论对方何种情况下，都应该马 上回应ACK报文，所以FIN_WAIT_1状态一般是比较难见到的，而FIN_WAIT_2状态还有时常常可以用netstat看到。
FIN_WAIT_2：上面已经详细解释了这种状态，实际上FIN_WAIT_2状态下的SOCKET，表示半连接，也即有一方要求close连接，但另外还告诉对方，我暂时还有点数据需要传送给你，稍后再关闭连接。
TIME_WAIT: 表示收到了对方的FIN报文，并发送出了ACK报文，就等2MSL后即可回到CLOSED可用状态了。如果FIN_WAIT_1状态下，收到了对方同时带 FIN标志和ACK标志的报文时，可以直接进入到TIME_WAIT状态，而无须经过FIN_WAIT_2状态。
CLOSING: 这种状态比较特殊，实际情况中应该是很少见，属于一种比较罕见的例外状态。正常情况下，当你发送FIN报文后，按理来说是应该先收到（或同时收到）对方的 ACK报文，再收到对方的FIN报文。但是CLOSING状态表示你发送FIN报文后，并没有收到对方的ACK报文，反而却也收到了对方的FIN报文。什 么情况下会出现此种情况呢？其实细想一下，也不难得出结论：那就是如果双方几乎在同时close一个SOCKET的话，那么就出现了双方同时发送FIN报 文的情况，也即会出现CLOSING状态，表示双方都正在关闭SOCKET连接。
CLOSE_WAIT: 这种状态的含义其实是表示在等待关闭。怎么理解呢？当对方close一个SOCKET后发送FIN报文给自己，你系统毫无疑问地会回应一个ACK报文给对 方，此时则进入到CLOSE_WAIT状态。接下来呢，实际上你真正需要考虑的事情是察看你是否还有数据发送给对方，如果没有的话，那么你也就可以 close这个SOCKET，发送FIN报文给对方，也即关闭连接。所以你在CLOSE_WAIT状态下，需要完成的事情是等待你去关闭连接。
LAST_ACK: 这个状态还是比较容易好理解的，它是被动关闭一方在发送FIN报文后，最后等待对方的ACK报文。当收到ACK报文后，也即可以进入到CLOSED可用状态了。

浏览器输入一个url到服务器处理整个过程
1）浏览器向DNS服务器请求解析该URL中的域名所对应的IP地址；
2）解析出IP地址后，根据该IP地址和默认端口80，和服务器建立TCP连接；
3）浏览器发出HTTP请求，该请求报文作为TCP三次握手的第三个报文的数据发送给服务器；
4）服务器给出相应，把对应的html文本发送给浏览器；
5）释放TCP连接；
6）浏览器将该文本显示出来。