HTTP与TCP

网络 `http` `tcp`

Http

HTTP是一个适用于分布式超媒体信息系统的应用层协议。
主要特点

- 支持C/S(客户端/服务端)模式。
- 简单快速。客户向服务器请求服务时，只需传入请求方式和路径。请求方法常用的的有GET、HEAD、POST。每种方法规定了客户与服务器的联系的类型不同。因为HTTP协议比较简单，HTTP服务器的程序规模较小，因而通信速度很快。
- 灵活。HTTP允许传输任意类型的数据对象。传输的类型由Content-Type加以标记。
- 无连接。限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。
- 无状态。HTTP协议是无状态协议，即协议对事务的处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则必须重传，这样可能导致每次传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

HTTP的URL的一般形式`http://host [":"port] [ads_path]` host为主机域名或IP地址；port为端口号；abs_path指定请求资源的URI通用资源标志符

HTTP的报文是面向文本的，报文中的每个字段都是一些ASCII码串，各个字段的长度时不确定的。
HTTP有两类报文：请求报文和响应报文。

- HTTP请求报文，由请求行、请求报头、空行和请求数据4部分组成，请求报文的一般格式如下：

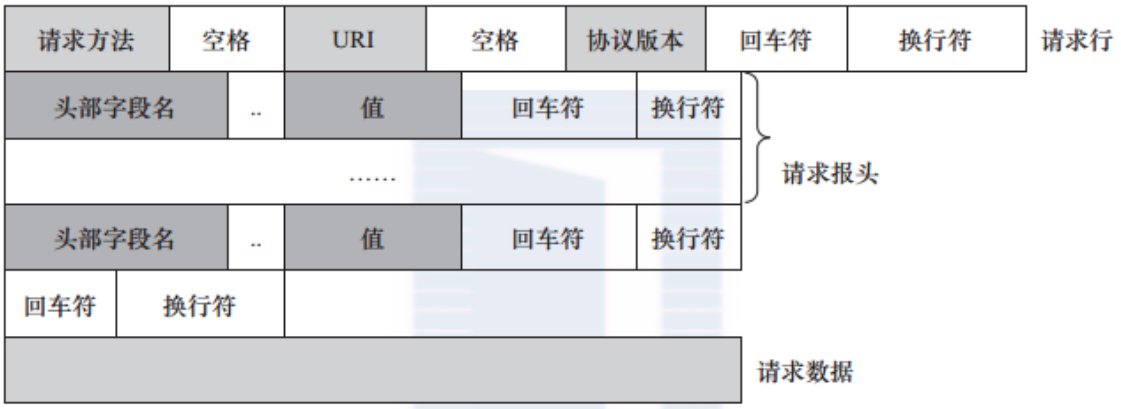


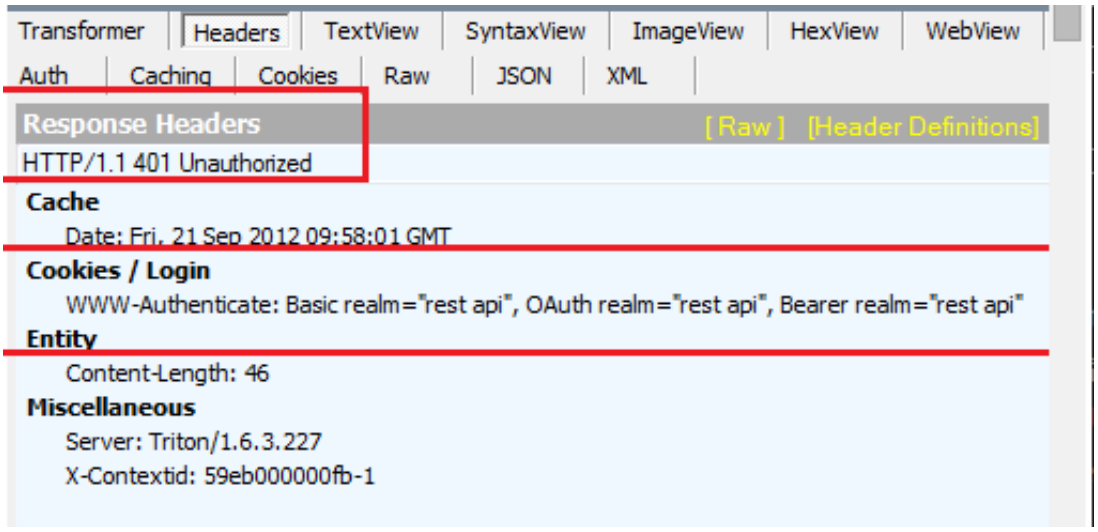
图 3-1 HTTP 请求报文的一般格式

(1)请求行：由请求方法字段、URI字段和HTTP协议版本字段组成，格式为：`Method Request-URI HTTP-Version CRLF` 例如：GET /form.html HTTP/1.1 (CRLF)
(2)HTTP头:a.请求头（requst header）,b.普通头（geneal header）,Get请求不包含内容实体，因此没有实体头
(3)空行
(4)请求数据：在Post请求中存在

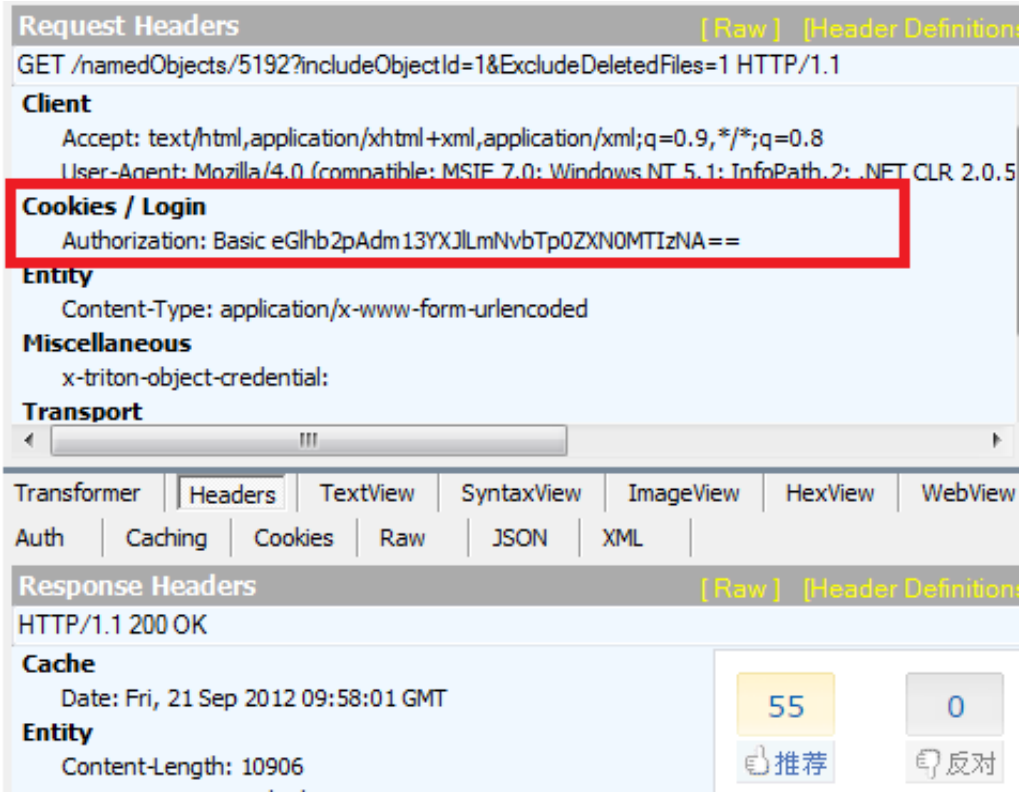
- HTTP响应：1.状态行，2.HTTP头，3.返回内容
(1)状态行：HTTP版本、响应状态码、状态码描述。100-199(指示信息)，200-299(响应成功)，300重定向，400客户端，500服务端。
(2)HTTP头：a.请求头（requst header）,b.普通头（geneal header），c.实体头（ebtity header）
(3)返回内容：返回的信息可以是HTML或者图片
基本认证
http协议是无状态的， 浏览器和web服务器之间可以通过cookie来身份识别。

- HTTP基本认证？
桌面应用程序也通过HTTP协议跟Web服务器交互， 桌面应用程序一般不会使用cookie, 而是把 “用户名+冒号+密码”用BASE64编码的字符串放在http request 中的header Authorization中发送给服务端， 这种方式叫HTTP基本认证(Basic Authentication)，当访问基本认证时会提示输入 用户名和密码。
当用户名及密码输入错误时会返回401界面

- HTTP基本认证过程
  1.第一步：客户端发送http requst给服务器
  2.第二步：因为request中没有包含Authorization header, 服务器会返回一个401 Unauthozied 给客户端，并且在Response 的 header “WWW-Authenticate” 中添加信息。



  3.第三步：客户端把用户名用和密码用BASE64编码后，放在Authorization header中发送给服务器， 认证成功。
  4.第四步：服务器将Authorization header中的用户名密码取出，进行验证， 如果验证通过，将根据请求，发送资源给客户端

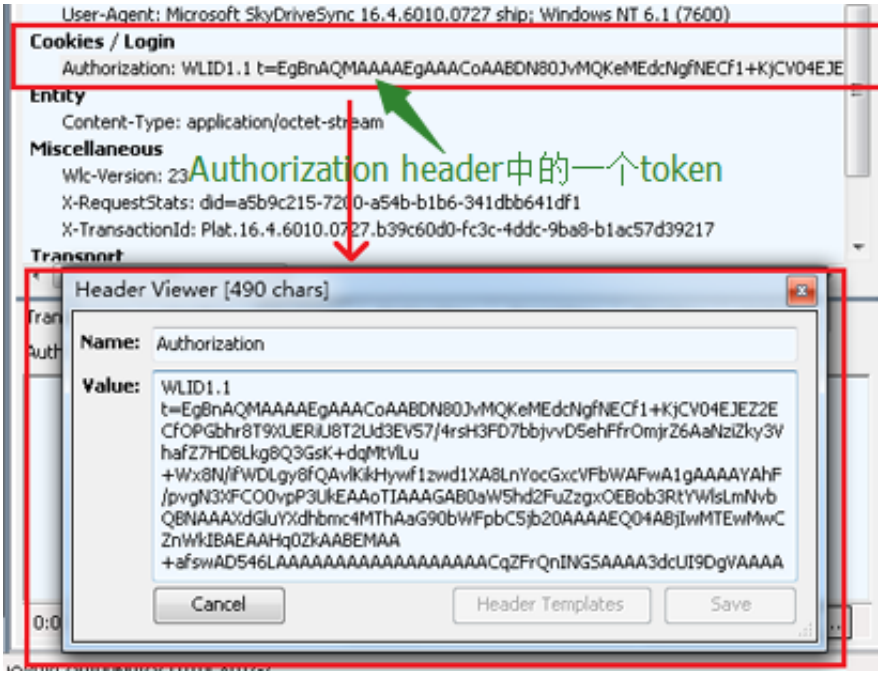


- HTTP认证的优点
  简洁明了，Rest API就是经常使用基本认证
  Http协议是无状态的，同一个客户端对服务器的每个请求都要求认证

- HTTP基本认证和 HTTPS
  把 “用户名+冒号+密码” 用BASE64编码后的string虽然用肉眼看不出来， 但用程序很容易解密，用Fiddler就直接给解密了。 所以这样的http request 在网络上，如果用

HTTP传输是很不安全的。 一般都是会用HTTPS传输, HTTPS是加密的, 所以比较安全.

- HTTP OAuth认证
  OAuth 对于Http来说，就是放在Authorization header中的不是用户名密码， 而是一个token.
  如下图

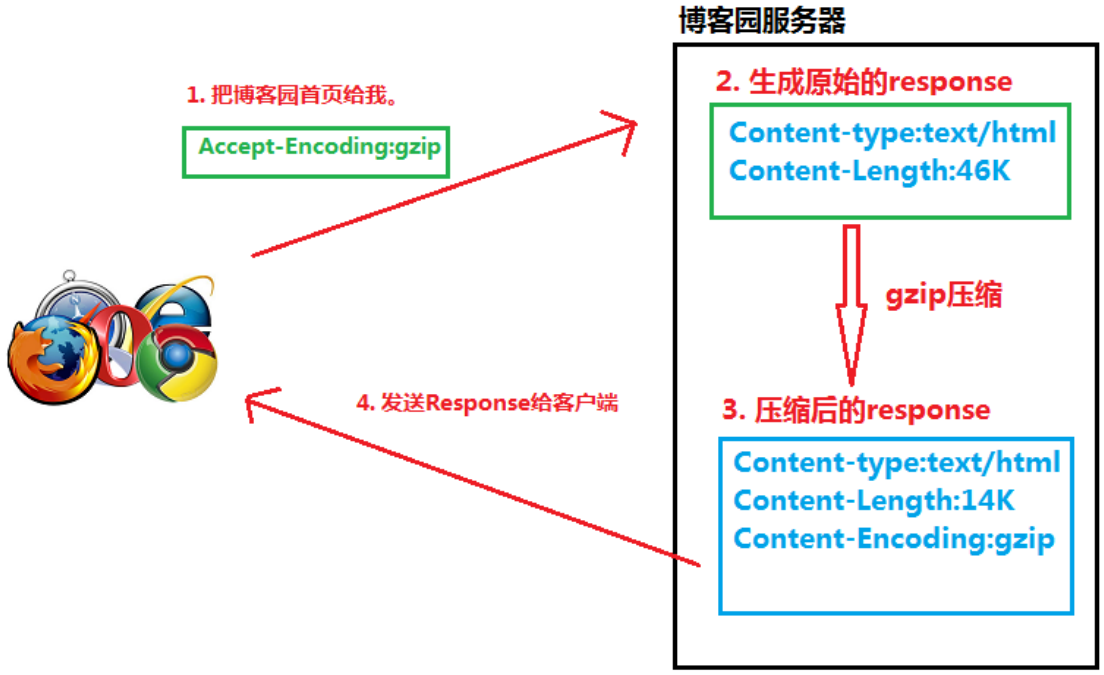


压缩

HTTP压缩： Web服务器和浏览器之间压缩传输的”文本内容“的方法。 HTTP采用通用的压缩算法，比如gzip来压缩HTML,Javascript, CSS文件。 能大大减少网络传输的数据量，提高了用户显示网页的速度。当然，同时会增加一点点服务器的开销。

- HTTP内容编码与HTTP压缩
  1.HTTP内容编码：通常是指对body内容进行编码，采用gzip这样的编码从而达到压缩的目的，也可以采用其他搅乱内容或加密的方式来加密，以此来防止未授权或是第三方来查看内容
  2.HTTP压缩：其实也是内容编码的一种形式。

- HTTP压缩的过程
  1.浏览器发送Http request 给Web服务器, request 中有Accept-Encoding: gzip, deflate。 (告诉服务器， 浏览器支持gzip压缩)–>客户端请求
  2.Web服务器接到request后， 生成原始的Response, 其中有原始的Content-Type和Content-Length。–>服务端生成原生的response
  3.Web服务器通过Gzip，来对Response进行编码， 编码后header中有Content-Type和Content-Length(压缩后的大小)， 并且增加了Content-Encoding:gzip. 然后把Response发送给浏览器。–>服务端对response进行压缩处理，并在header中添加Content-Encodeing:gzip,然后将其发送给客户端。
  4.浏览器接到Response后，根据Contenfuwut-Encoding:gzip来对Response 进行解码。 获取到原始response后， 然后显示出网页。–>客户端对服务端返回的gzip进行解码来获得原生的response来展示内容。



- 内容编码类型
  Content-Encoding值
  gzip：表明实体采用GNU zip编码
  compress ：表明实体采用Unix的文件压缩程序
  deflate：表明实体是用zlib的格式压缩的
  identity：表明没有对实体进行编码。当没有Content-Encoding header时， 就默认为这种情况
  gzip, compress, 以及deflate编码都是无损压缩算法，用于减少传输报文的大小，不会导致信息损失。 其中gzip通常效率最高， 使用最为广泛。

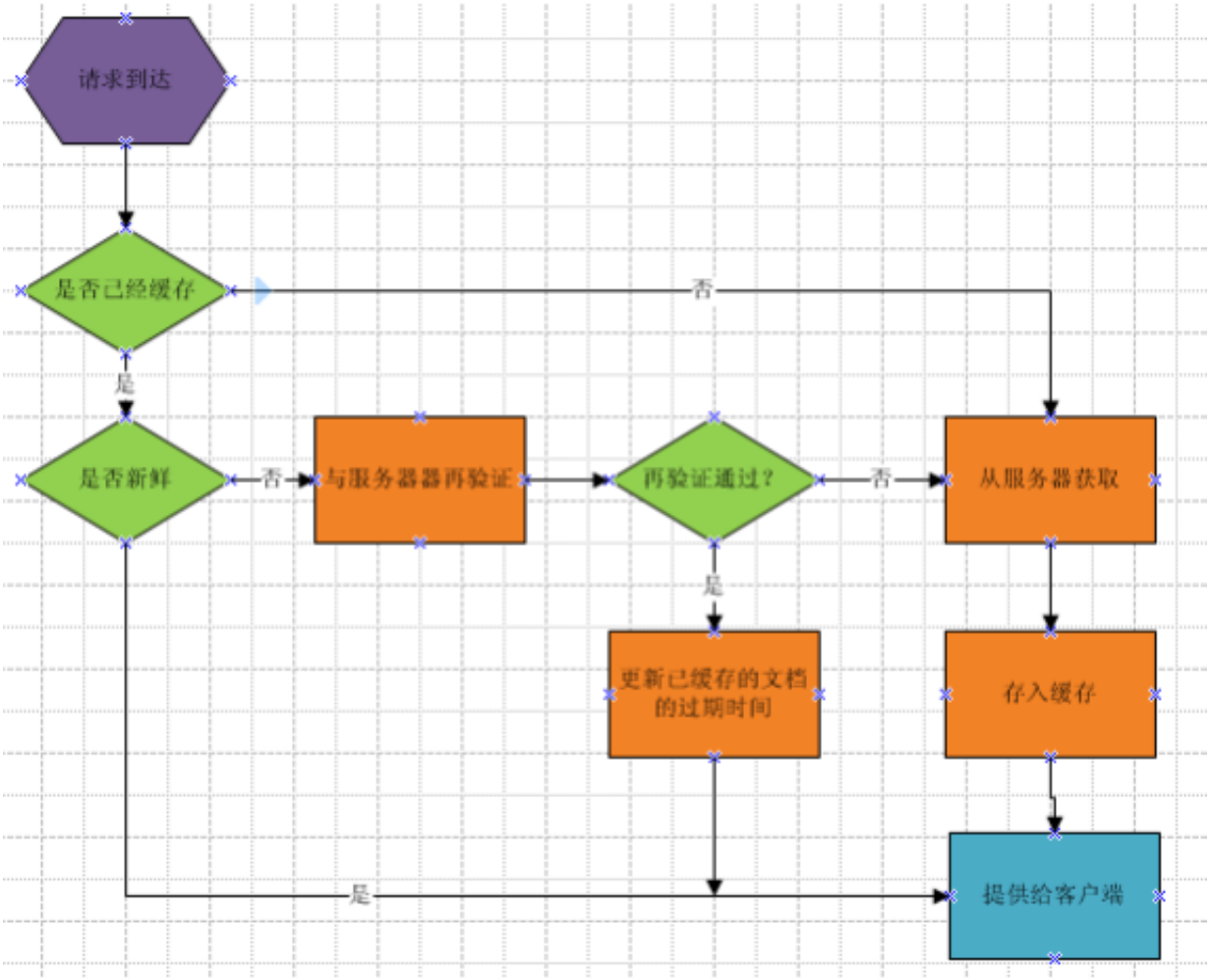
- 压缩的好处
  http压缩对纯文本可以压缩至原内容的40%, 从而节省了60%的数据传输。

- Gzip的缺点、如何压缩
  Gzip对JPEG文件压缩不够好
  Gzip压缩：对文本文件内容中的类似字符串进行临时替换，从而是文件的体积变小，这种方式对web来说非常有用，因为HTML和CSS文件中会用大量的重复字符串资源，如标签、空格等。

- HTTP Response能压缩，HTTP Request也可以压缩。
  Response在被服务端可以被压缩，Request也可以被压缩，但不是在服务度，而是在HTTP程序。在发送Reques时才对其进行编码

缓存

HTTP协议中提供了强大的缓存机制，利用这些缓存机制可有效大大提高性能



- 概念
  HTTP中的缓存功能是指浏览器缓存和缓存代理服务器。
- 优点
  1.减少冗余的数据传输
  2.减少服务器的负担，从而大大提高网站性能
  3.加快客户端对数据页的加载
- 最新缓存
  1.根据最新的修改时间 header ”If-Modified-Since“来告诉Web服务器。 a .客户端向服务端求求你一个文档时，首先进行本地文件的检索，得到缓存文档中的最后修改时间，然后通过If-Modified-Since， 发送Request给Web服务器。 b .服务器收到Request后将服务器文档的修改时间与请求传过来的时间进行比对，如果一样，则缓

存有效，服务端返回403告诉客户端加载本地缓存内容。`c`.倘若文档有所更新，则缓存无效，服务端将发送最新的数据给客户端
2.浏览器把缓存文件的ETag（标签），通过header “If-None-Match”, 来告诉Web服务器。当用Last-Modified 无法解决的一些问题时（如时间改变了，但是内容没有改变），就需要根据实体内容生成的一段hash字符串（类似于MD5或者SHA1之后的结果），可以标识资源的状态。 当资源发送改变时，ETag也随之发生变化。

代理：未写
Cookie：未写
HTTP与HTTPS
HTTPS是基于HTTP的一个安全超文本传输协议,是由SSL+HTTP协议构建的可进行加密传输、身份认证的网路协议，使用的是SSL进行信息交换，该协议需要ca申请证书

- 区别
  1.http传输信息是明文传输，而https传输信息为SSL加密传输。
  2.https需要ca申请证书（付费）
  3.http连接简单属于无状态协议
  4.https由SSL+HTTP进行构建传输协议，因此比http更安全

- HTTPS握手过程
  1.客户端请求https连接
  2.服务端返回证书（公钥）
  3.客户端(浏览器)产生一个随机的对称密钥
  4.使用证书中服务端的公钥加密此对称密钥
  5.将此已经加密过的对称密钥发给server。至此6.client和server都通过可靠的手段拥有对称密钥.
  7.通过对称密钥加密Http的通信内容。

- 中间人攻击
  中间人获取s发给c的公钥，然后自己伪造一对公私钥，再将自己的公钥发送给c,拦截c发给s的密文，用伪造的密钥得到c发出的内容，再用真实的公钥发送给s,解决办法：数字证书、证书链

HTTP的请求方法
http的请求方法主要有：get,post,head,delete,update,options

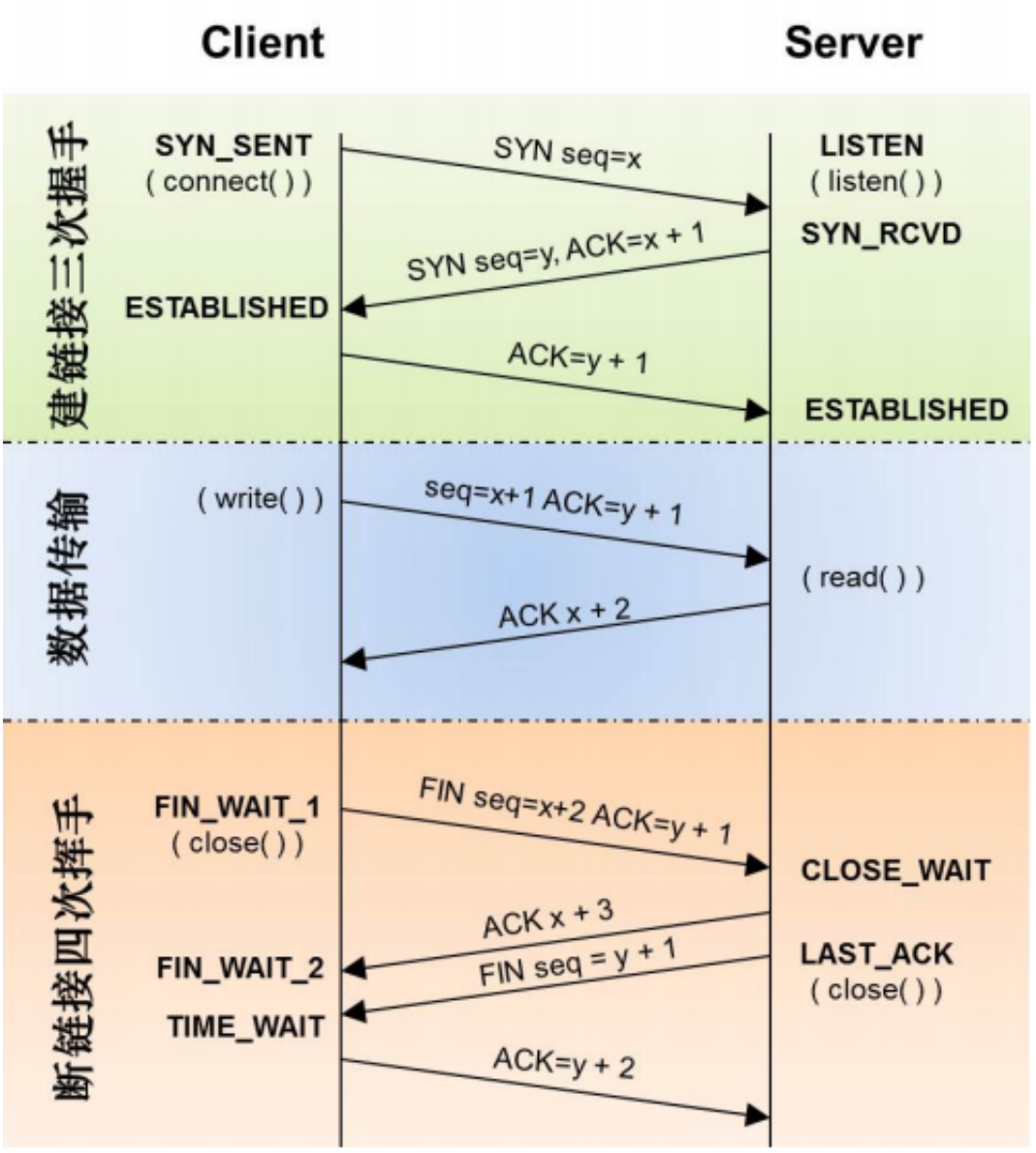
- get请求:主要用于获取页面信息，同时可以包含一些特殊的信息（如提交用户名和密码）格式如http://zzk.cnblogs.com/s?t=b&w=http%E4%BB%A3%E7%90%86，在实际中HTTP协议的头部数据不可超过1M
- post请求：主要用于按某些条件来请求数，且协议实体部分没有数据大小限制

Tcp

Tcp建立连接的三次握手以及四次挥手

建立：首先C端发送请求报文，S端接受连接请求后恢复ACK报文，并为这次连接分配资源，C端接受ACK报文后也向S端发送ACK报文分配资源，这样TCP连接就建立成功了

断开：C端发起中断请求（发送FIN报文）；S端接受报文后（C端已没有数据发送给你了）；S端向C端发送ACK(使C端处于FIN_WAIT,等待S端数据发送完成)，当S端数据发送完成后会向C端FIN报文；C端向S端发送ACK报文（处于TIME_WAIT,确保网络）；S端接受到ACK就表明可以断开连接了，若未接受到则可以重传。



建立链接的3次握手：

当S端接收到C端的SYN连接请求报文后，S端直接发送SYN+ACK报文。ACK报文是用来应答的，SYN报文是用来同步的。

主要是要初始化Sequence Number 的初始值。通信的双方要互相通知对方自己的初始化的Sequence Number（缩写为ISN：Inital Sequence Number）——所以叫SYN，全称Synchronize Sequence Numbers。也就上图中的 x 和 y。这个号要作为以后的数据通信的序号，以保证应用层接收到的数据不会因为网络上的传输的问题而乱序（TCP会用这个序号来拼接数据）。

断开连接时的4次挥手：
关闭连接时，C端主动向S端发送FIN报文，当S端接收到FIN报文后，并不会立即关闭SCOET连接，而是先要发送给一个ACK给C端告诉我已收到，等到S端将数据全部传输完后再向C端发送FIN报文，C端收到FIn后再向S端发送ACK应答。
其实你仔细看是2次，因为TCP是全双工的，所以，发送方和接收方都需要Fin和Ack。只不过，有一方是被动的，所以看上去就成了所谓的4次挥手。如果两边同时断连接，那就会就进入到CLOSING状态，然后到达TIME_WAIT状态。

可靠传输，重传：超时重传，快速重传。
在断开连接时，当S端向C端发送FIN报文后，C端向S端发送应答ACK报文，当S端接收到ACK后就可以断开连接，此时C端等待2MSL。后若依然没有收到回复，说明S端已关闭，我C端也可以关闭了，Tcp连接关闭

TIME_WAIT状态需要经过2MLS:当双方都同意关闭连接时，并且挥手的4次报文也都发送完毕，由于网络不可靠的原因，可能不能保证ACK报文S端接受不到，进而导致S端重发FIN报文

网络 OSI七层模型
将服务、接口和协议区分开来，通过该结构模型使不同的系统在不同的网络之间实现可靠的通讯.端应用程序进程->应用层->传输层->网络层->数据链路层->物理层。

TCP/IP体系
应用层、传输层、网络互联层和网络接口层。

- 网络接口层：接受和发送数据报
  主要负责将数据包发送到网络传输介质上以及从网络上接受TCP/IP协议。
- 网络互连层：数据包封装和路由寻址
  1.IP(网际协议)：主要负责将数据从源节点发送到目的节点，并不对数据进行检查
  2.ARP(地址解析协议)：将IP地址解析为计算机的物理地址，便于物理设备进行接收数据
  3.RARP(反向地址解析协议)：将设备的物理地址转化为IP地址。
  4.ICMP(因特控制报文协议)：主要负责发送和传递包含控制信息的数据包。
- 传输层：应用进程间端到端的通信
  将某个应用进程传输到另外的一个应用进程，通过标识端口或者端口号来识别不同的进程。最主要的两个协议TCP和UDP
  1.TCP:可靠、面向连接、可靠。
  2.UDP:不可靠、面向无连接、效率高、不安全、协议简单。

- 应用层：不同协议
  直接为应用进程服务的，其中包含包含多种应用协议：SMTP、HTTP、FTP、Telent(远程终端协议)、SNMP(简单网络管理协议)