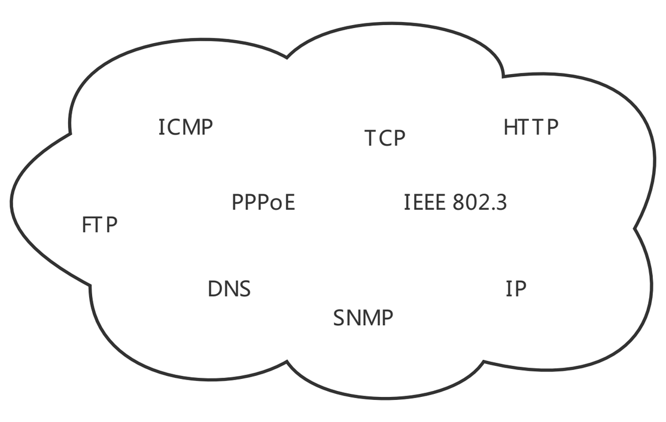
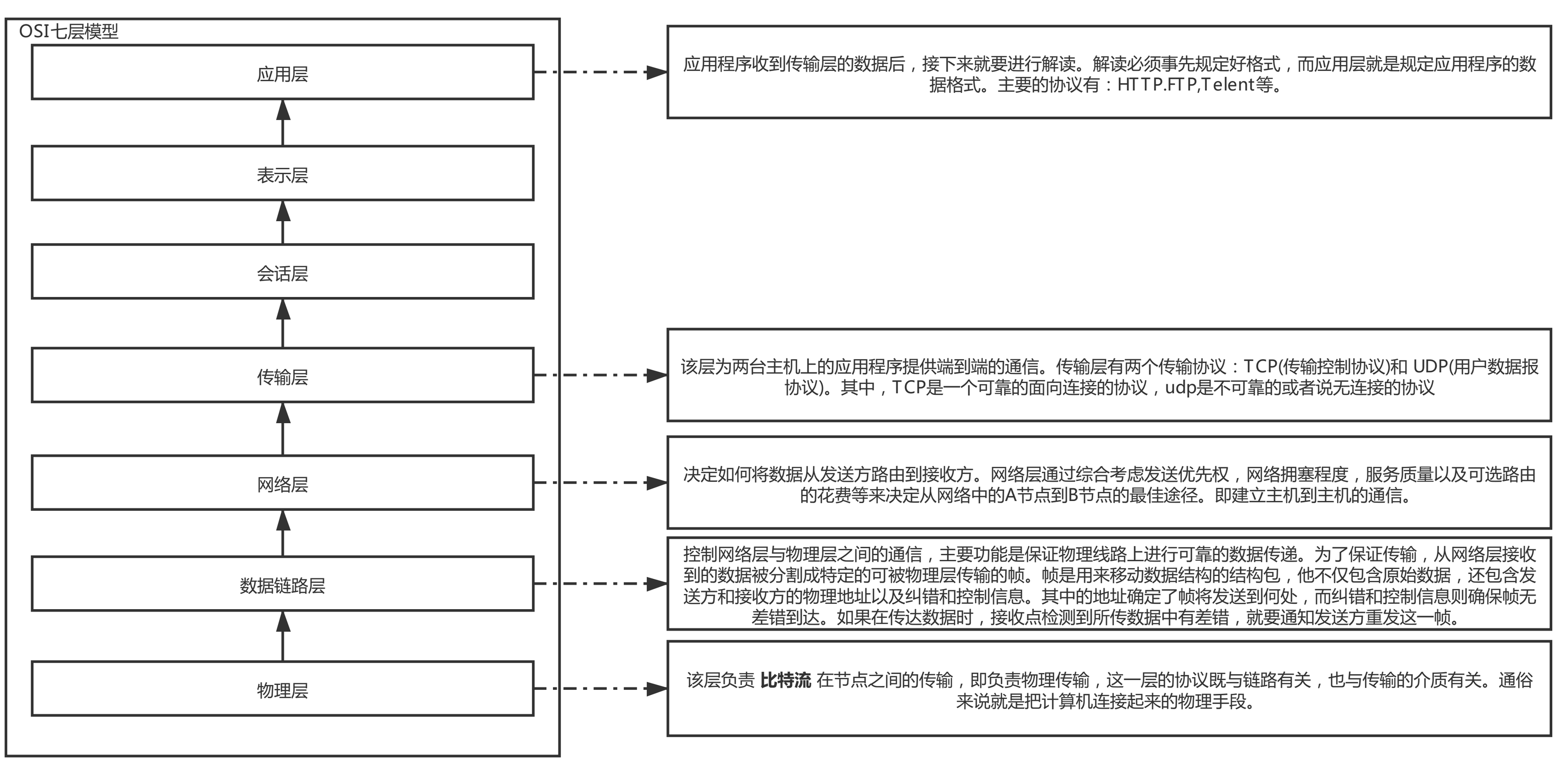
什么是TCP/IP协议?

通俗地讲，就是计算机与网络设备之间如果要相互通信，双方就必须基于相同的方法或规则，即如何探测到通信目标，使用哪种语言通信，怎样开始和结束通信等等，这些规则都需要事先确定，而我们就将这种规则称为协议（protocol）。

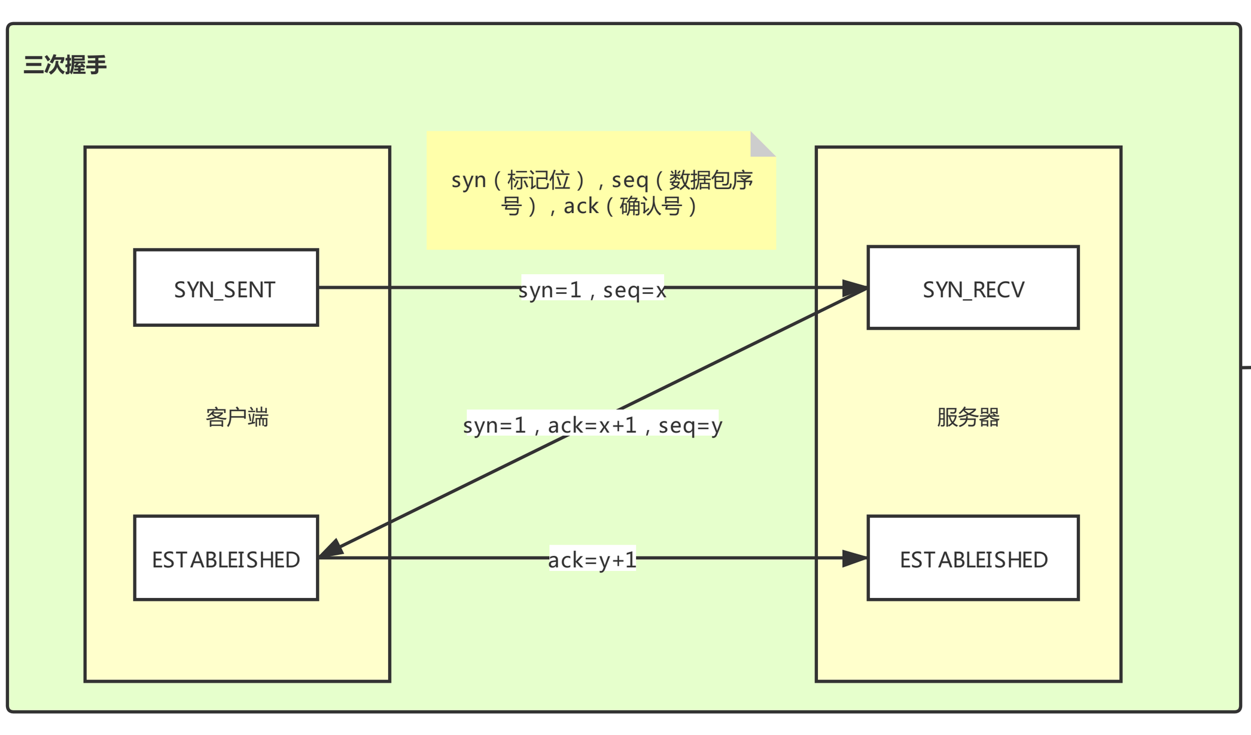


TCP/IP是互联网相关各类协议族的总称



TCP与UDP的区别：

	TCP	UDP
可靠性	可靠	不可靠
连接性	面向连接	无连接
报文	面向字节流	面向报文
效率	传输效率低	传输效率高
双工性	全双工	一对一、一对多、多对一、多对多
流量控制	滑动窗口	无
拥塞控制	慢开始、拥塞避免、快重传、快恢复	无
传输速度	慢	快
应用场景	对效率要求低，对准确性要求高或者要求有连接的场景	对效率要求高，对准确性要求低



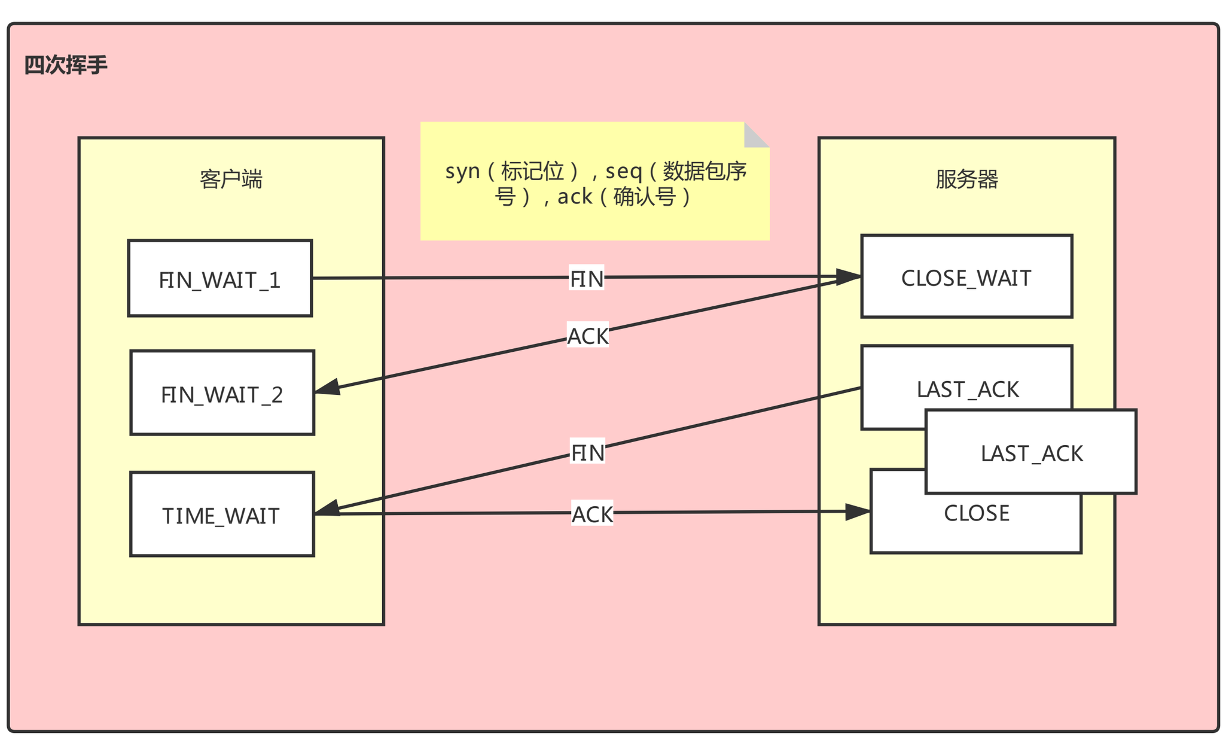
具体过程如下：

1建立连接。客户端发送连接请求报文段，并将syn(标记位)设置为1，Squence Number(数据包序号)(seq)为x,接下来等待服务端确认，客户端进入SYN_SENT状态(请求连接)；

2服务端收到客户端的 SYN 报文段，对 SYN 报文段进行确认，设置 ack(确认号)为 x+1(即seq+1；同时自己还要发送 SYN 请求信息，将 SYN 设置为1, seq为 y。服务端将上述所有信息放到 SYN+ACK 报文段中，一并发送给客户端，此时服务器进入 SYN_RECV状态。

SYN_RECV是指,服务端被动打开后,接收到了客户端的SYN并且发送了ACK时的状态。再进一步接收到客户端的ACK就进入ESTABLISHED状态。

3客户端收到服务端的 SYN+ACK(确认符) 报文段；然后将 ACK 设置为 y+1,向服务端发送ACK报文段，这个报文段发送完毕后，客户端和服务端都进入ESTABLISHED(连接成功)状态，完成TCP 的三次握手。



当客户端和服务端通过三次握手建立了 TCP 连接以后,当数据传送完毕,断开连接就需要进行TCP的四次挥手。其四次挥手如下所示：

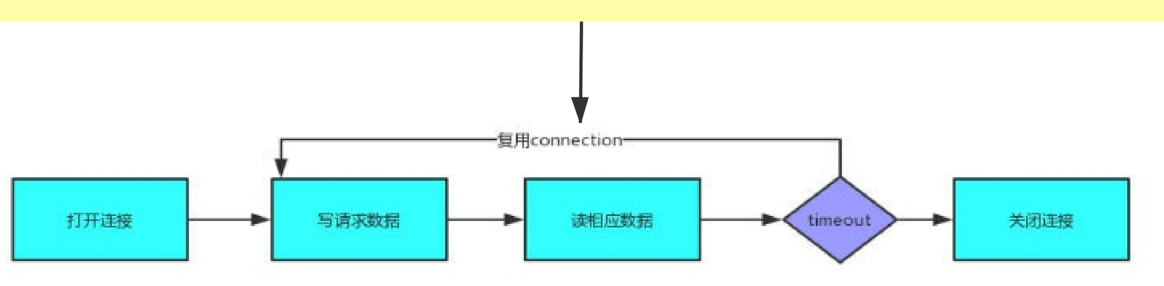
1客户端设置seq和 ACK ,向服务器发送一个 FIN(终结)报文段。此时，客户端进入 FIN_WAIT_1 状态，表示客户端没有数据要发送给服务端了。

2服务端收到了客户端发送的 FIN 报文段，向客户端回了一个 ACK 报文段。

3服务端向客户端发送FIN 报文段，请求关闭连接，同时服务端进入 LAST_ACK 状态。

4客户端收到服务端发送的 FIN 报文段后，向服务端发送 ACK 报文段,然后客户端进入 TIME_WAIT 状态。服务端收到客户端的 ACK 报文段以后，就关闭连接。此时，客户端等待 2MSL（指一个片段在网络中最大的存活时间）后依然没有收到回复，则说明服务端已经正常关闭，这样客户端就可以关闭连接了。

如果有大量的连接，每次在连接，关闭都要经历三次握手，四次挥手，这显然会造成性能低下。因此。Http 有一种叫做 长连接（keepalive connections） 的机制。它可以在传输数据后仍保持连接，当客户端需要再次获取数据时，直接使用刚刚空闲下来的连接而无需再次握手。



为什么要三次握手？
为了防止已失效的连接请求报文突然又传送到了服务端，因为产生错误。
具体解释：“已失效的连接请求报文段”产生情况：
client 发出的第一个连接请求报文段并没有丢失，而是在某个网络节点长时间滞留，因此导致延误到连接释放以后的某个时间才到达 service。如果没有三次握手，那么此时server收到此失效的连接请求报文段，就误认为是 client再次发出的一个新的连接请求，于是向 client 发出确认报文段，同意建立连接，而此时 client 并没有发出建立连接的情况，因此并不会理会服务端的响应，而service将会一直等待client发送数据，因此就会导致这条连接线路白白浪费。
如果此时变成两次挥手行不行？
这个时候需要明白全双工与半双工，再进行回答。比如：
第一次握手： A给B打电话说，你可以听到我说话吗？
第二次握手： B收到了A的信息，然后对A说： 我可以听得到你说话啊，你能听得到我说话吗？
第三次握手： A收到了B的信息，然后说可以的，我要给你发信息啦！
在三次握手之后，A和B都能确定这么一件事： 我说的话，你能听到； 你说的话，我也能听到。 这样，就可以开始正常通信了，如果是两次，那将无法确定。

为什么要四次挥手？
TCP 协议是一种面向连接，可靠，基于字节流的传输层通信协议。TCP 是全双工模式(同一时刻可以同时发送和接收)，这就意味着，当主机1发出 FIN 报文段时，只是表示主机1已经没有数据要发送了，主机1告诉主机2，它的数据已经全部发送完毕；但是，这个时候主机1还是可以接受来自主机2的数据；当主机2返回 ACK报文段时，这个时候就表示主机2也没有数据要发送了，就会告诉主机1，我也没有数据要发送了，之后彼此就会中断这次TCP连接。

为什么要等待 2MSL?
MSL：报文段最大生存时间，它是任何报文段被丢弃前在网络内的最长时间
原因如下：
保证TCP协议的全双工连接能够可靠关闭
保证这次连接的重复数据从网络中消失
第一点： 如果主机1直接 关闭，由于IP协议的不可靠性或者其他网络原因，导致主机2没有收到主机1最后回复的 ACK。那么主机2就会在超时之后继续发送 FIN，此时由于主机1已经关闭，就找不到与重发的 FIN 对应的连接。所以，主机1 不是直接进入 关闭，而是TIME_WAIT 状态。当再次收到 FIN 的时候，能够保证对方收到 ACK ，最后正确关闭连接。
第二点：如果主机1直接 关闭，然后又再向主机 2 发起一个新连接，我们不能保证这个新连接与刚才关闭的连接端口是不同的。也就是说有可能新连接和老连接的端口号是相同的。一般来说不会发生什么问题，但还是有特殊情况出现；假设新连接和已经关闭的老连接端口号是一样的，如果前一次连接的某些数据仍然滞留在网络中(Lost Duplicate)，那些延迟数据在建立新连接之后才到达主机2，由于新连接和老连接的端口号是一样的，TCP 协议就认为那个延迟的数据时属于新连接的，这样就和真正的新连接的数据包发生混淆了。所以TCP连接要在 TIME_WAIT 状态等待两倍 MSL ，保证本次连接的所有数据都从网络中消失。

HTTP

概念：HTTP协议，即超文本传输协议(Hypertext transfer protocol)。是一种详细规定了浏览器和万维网(WWW = World Wide Web)服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。

HTTP协议是用于从WWW服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP是一个应用层协议，由请求和响应构成，是一个标准的客户端服务器模型。HTTP是一个无状态的协议。

HTTP与HTTPS

HTTP协议通常承载于TCP协议之上，有时也承载于TLS或SSL协议层之上，这个时候，就成了我们常说的HTTPS。

HTTP默认的端口号为80，HTTPS的端口号为443。

HTTP的连接

HTTP协议是建立在TCP协议基础之上的，当浏览器需要从服务器获取网页数据的时候，会发出一次HTTP请求。HTTP会通过TCP建立起一个到服务器的连接通道，当本次请求需要的数据完毕后，HTTP会立即将TCP连接断开，这个过程是很短的。所以HTTP连接是一种短连接，是一种无状态的连接。

所谓的无状态，是指浏览器每次向服务器发起请求的时候，不是通过一个连接，而是每次都建立一个新的连接。如果是一个连接的话，服务器进程中就能保持住这个连接并且在内存中记住一些信息状态。而每次请求结束后，连接就关闭，相关的内容就释放了，所以记不住任何状态，成为无状态连接。

HTTP的无状态性

所谓HTTP协议的无状态性是指服务器的协议层无需为不同的请求之间建立任何相关关系，它特指的是协议层的无状态性。但是这并不代表建立在HTTP协议之上的应用程序就无法维持状态。

应用层可以通过会话Session来跟踪用户请求之间的相关性，服务器会为每个会话对象绑定一个唯一的会话ID，浏览器可以将会话ID记录在本地缓存LocalStorage或者Cookie，在后续的请求都带上这个会话ID，服务器就可以为每个请求找到相应的会话状态。

无状态是指协议对于事务处理没有记忆能力，服务器不知道客户端是什么状态。从另一方面讲，打开一个服务器上的网页和你之前打开这个服务器上的网页之间没有任何联系。

HTTP的协议格式

HTTP的请求和响应的消息协议是一样的，分为三个部分，起始行、消息报头和消息体。这三个部分以CRLF作为分隔符。最后一个消息头有两个CRLF，用来表示消息头部的结束。
发出的请求信息格式如下：

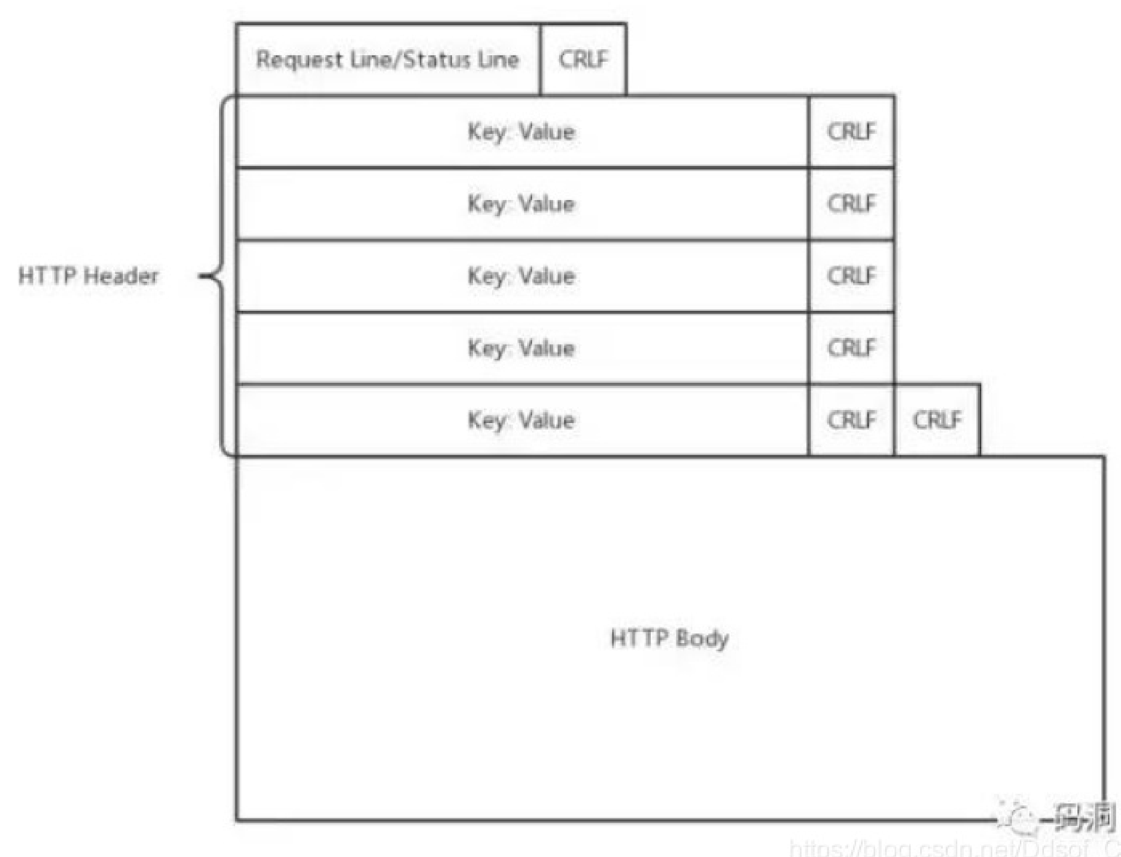
●请求行，例如GET /images/logo.gif HTTP/1.1，表示从/images目录下请求logo.gif这个文件。

●（请求/消息报）头，例如Accept-Language: en

●空行

●可选的消息体　请求行和标题必须以作为结尾（也就是，回车然后换行）。空行内必须只有而无其他空格。在HTTP/1.1协议中，所有的请求头，除post外，都是可选的。

三个部分分别是：请求行/起始行、消息报头/请求头、消息体/请求正文。



GET和POST的区别：
1.GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456. POST方法是把提交的数据放在HTTP包的Body中。

2.GET提交的数据大小有限制，最多只能有1024字节（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制。

3.GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值。

4.GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码。

HTTP请求方法

HTTP/1.1协议中共定义了八种方法（有时也叫“动作”）来表明Request-URI指定的资源的不同操作方式：

OPTIONS - 返回服务器针对特定资源所支持的HTTP请求方法。也可以利用向Web服务器发送’*'的请求来测试服务器的功能性。

HEAD - 向服务器索要与GET请求相一致的响应，只不过响应体将不会被返回。这一方法可以在不必传输整个响应内容的情况下，就可以获取包含在响应消息头中的元信息。该方法常用于测试超链接的有效性，是否可以访问，以及最近是否更新。

GET - 向特定的资源发出请求。注意：GET方法不应当被用于产生“副作用”的操作中，例如在web app.中。其中一个原因是GET可能会被网络蜘蛛等随意访问。

POST - 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。

PUT - 向指定资源位置上传其最新内容。

DELETE - 请求服务器删除Request-URI所标识的资源。

TRACE - 回显服务器收到的请求，主要用于测试或诊断。

CONNECT - HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

方法名称是区分大小写的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码405（Method Not Allowed）；当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码501（Not Implemented）。

HTTP服务器至少应该实现GET和HEAD方法，其他方法都是可选的。此外，除了上述方法，特定的HTTP服务器还能够扩展自定义的方法。

HTTP状态码

2XX 成功

200 OK，表示从客户端发来的请求在服务器端被正确处理

204 No content，表示请求成功，但响应报文不含实体的主体部分

206 Partial Content，进行范围请求

3XX 重定向

301 moved permanently，永久性重定向，表示资源已被分配了新的 URL

302 found，临时性重定向，表示资源临时被分配了新的 URL

303 see other，表示资源存在着另一个 URL，应使用 GET 方法定向获取资源

304 not modified，表示服务器允许访问资源，但因发生请求未满足条件的情况

307 temporary redirect，临时重定向，和302含义相同

4XX 客户端错误

400 bad request，请求报文存在语法错误

401 unauthorized，表示发送的请求需要有通过 HTTP 认证的认证信息

403 forbidden，表示对请求资源的访问被服务器拒绝

404 not found，表示在服务器上没有找到请求的资源

5XX 服务器错误

500 internal sever error，表示服务器端在执行请求时发生了错误

503 service unavailable，表明服务器暂时处于超负载或正在停机维护，无法处理请求