H4 HTTP报文格式

请求报文

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

第一行叫做请求行（request line），其后继的行叫做首部行（header line）。

请求行有3个字段：方法字段、URL字段和HTTP版本字段。方法字段可以几种不同的值，包括GET、POST、HEAD、PUT和DELETE。

响应报文

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GET
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
```

它有三个部分：一个初始状态行（status line），6个首部行（header line），然后是实体体（entity body）。实体体是报文的主要部分。

H4 常见的状态码

- 200 OK：请求成功，信息在返回的响应报文中。
- 301 Moved Permanently：请求的对象已经被永久转移了，新的URL定义在响应报文的Location：首部行中。客户软件将自动获取新的URL。
- 400 Bad Request：一个通用的差错代码，指示该请求不能被服务器理解。
- 404 Not Found：被请求的文档不在服务器上。
- 404 HTTP Version Not Supported：服务器不支持请求报文使用的HTTP协议版本。

H4 连接大量处于time-wait是因为什么，大量处于closed-wait是因为什么

在**高并发短连接**的TCP服务器上，当服务器处理完请求后立刻主动正常关闭连接。这个场景下会出现大量socket处于TIME_WAIT状态。如果客户端的并发量持续很高，此时部分客户端就会显示连接不上。我来解释下这个场景。主动正常关闭TCP连接，都会出现TIMEWAIT。

为什么我们要关注这个高并发短连接呢？有两个方面需要注意：

1. **高并发可以让服务器在短时间范围内同时占用大量端口**，而端口有个0~65535的范围，并不是很多，刨除系统和其他服务要用的，剩下的就更少了。
2. 在这个场景中，**短连接表示“业务处理+传输数据的时间远远小于TIMEWAIT超时的时间”的连接。**

> 这里有个相对长短的概念，比如取一个web页面，1秒钟的http短连接处理完业务，在关闭连接之后，这个业务用过的端口会停留在TIMEWAIT状态几分钟，而这几分钟，其他HTTP请求来临的时候是无法占用此端口的(占着茅坑不拉翔)。

单用这个业务计算服务器的利用率会发现，服务器干正经事的时间和端口（资源）被挂着无法被使用的时间的比例是 1：几百，服务器资源严重浪费。（说个题外话，从这个意义出发来考虑服务器性能调优的话，长连接业务的服务就不需要考虑TIMEWAIT状态。同时，假如你对服务器业务场景非常熟悉，你会发现，在实际业务场景中，一般**长连接对应的业务的并发量并不会很高**。

综合这两个方面，持续的到达一定量的高并发短连接，会使服务器因端口资源不足而拒绝为一部分客户服务。同时，这些端口都是服务器临时分配，无法用SO_REUSEADDR选项解决这个问题

参考

解决TIME_WAIT过多造成的问题

H4 get和post的本质区别是什么

get产生一个TCP数据包，post产生两个TCP数据包。即使用get方法时，浏览器一次性将首部和数据发送给服务端，而使用post方法时会将首部和数据分两次发送。注意，这才是本质区别，还有一些区别入下：

- GET在浏览器回退时是无害的，而POST会再次提交请求。
- GET产生的URL地址可以被Bookmark，而POST不可以。
- GET请求会被浏览器主动cache，而POST不会，除非手动设置。
- GET请求只能进行url编码，而POST支持多种编码方式。
- GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
- GET请求在URL中传送的参数是有长度限制的，而POST么有。
- 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。
- GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。
- GET参数通过URL传递，POST放在Request body中。

参考

http请求中get和post的区别

GET和POST两种基本请求方法的区别

H4 http与https的区别

https和http的区别是https在应用层与传输层之间使用安全套接字层进行加密。

参考

HTTP与HTTPS的区别

HTTPS 与 SSL 证书概要

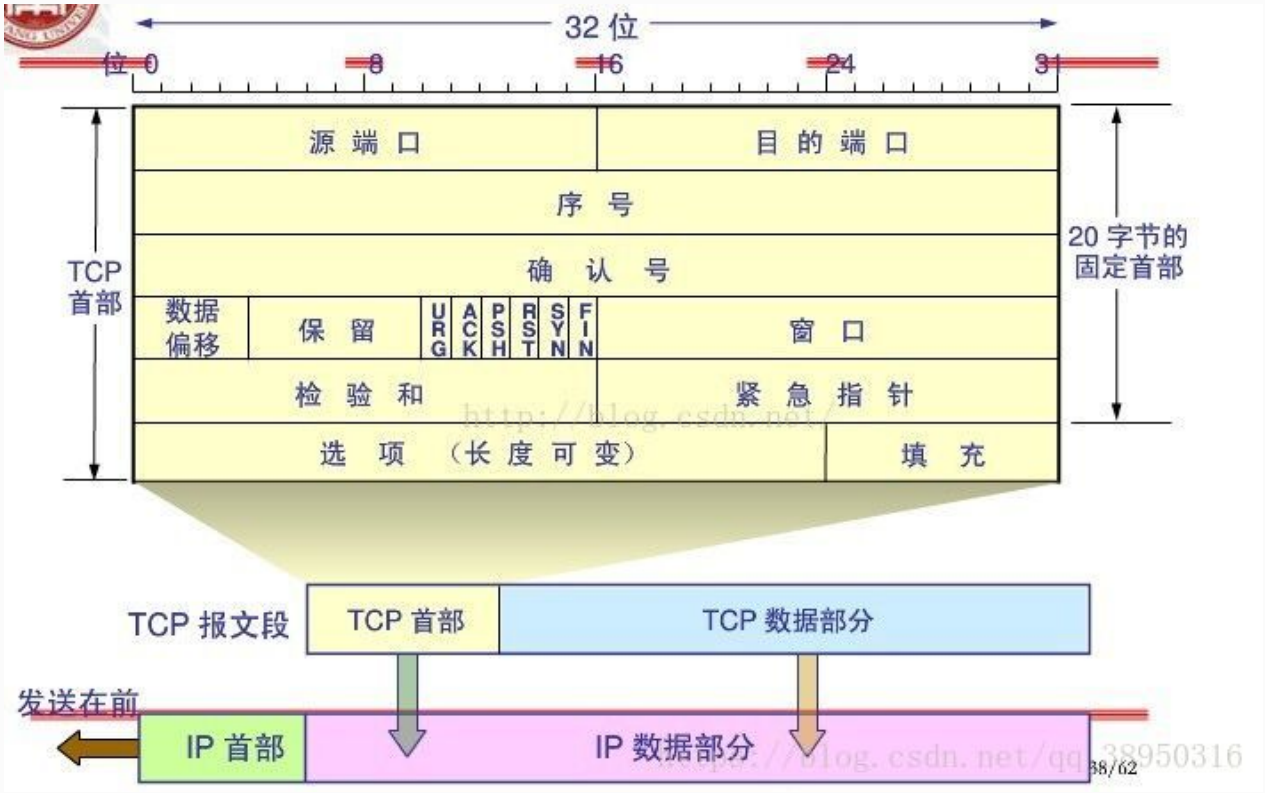
H4 https中对称加密与非对称加密用在哪?

对称加密用于加密传送数据，非对称加密用于加密对称加密的密钥。为什么这样设计，原因是对称加密算法需要耗费较高的算力，而对成加密需要的算力较低，所以这样设计。

H4 服务器如果有大量的半连接状态的客户端连接，如何处理

有两种原因，一是遭受到DOS攻击，二是

H4 TCP报文结构和UDP报文结构



序列号seq：占4个字节，用来标记数据段的顺序，TCP把连接中发送的所有数据字节都编上一个序号，第一个字节的编号由本地随机产生；给字节编上序号后，就给每一个报文段指派一个序号；序列号seq就是这个报文段中的第一个字节的数据编号。

确认号ack：占4个字节，期待收到对方下一个报文段的第一个数据字节的序号；序列号表示报文段携带数据的第一个字节的编号；而确认号指的是期望接收到下一个字节的编号；因此当前报文段最后一个字节的编号+1即为确认号。

确认ACK：占1位，仅当ACK=1时，确认号字段才有效。ACK=0时，确认号无效

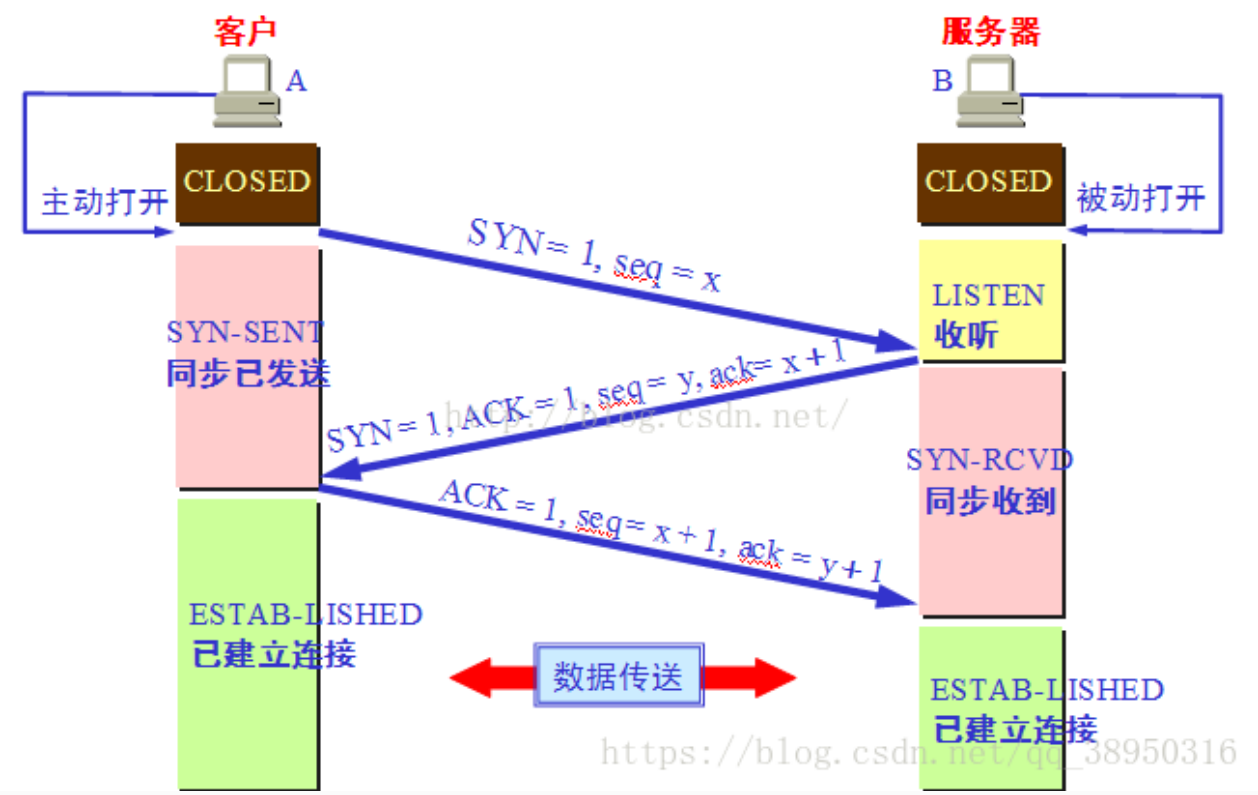
同步SYN：连接建立时用于同步序号。当SYN=1，ACK=0时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得SYN=1，ACK=1。因此，SYN=1表示这是一个连接请求，或连接接受报文。SYN这个标志位只有在TCP建产连接时才会被置1，握手完成后SYN标志位被置0。

终止FIN：用来释放一个连接。FIN=1表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接

PS：ACK、SYN和FIN这些大写的单词表示标志位，其值要么是1，要么是0；ack、seq小写的单词表示序号。

H4 三次握手和四次挥手

三次握手

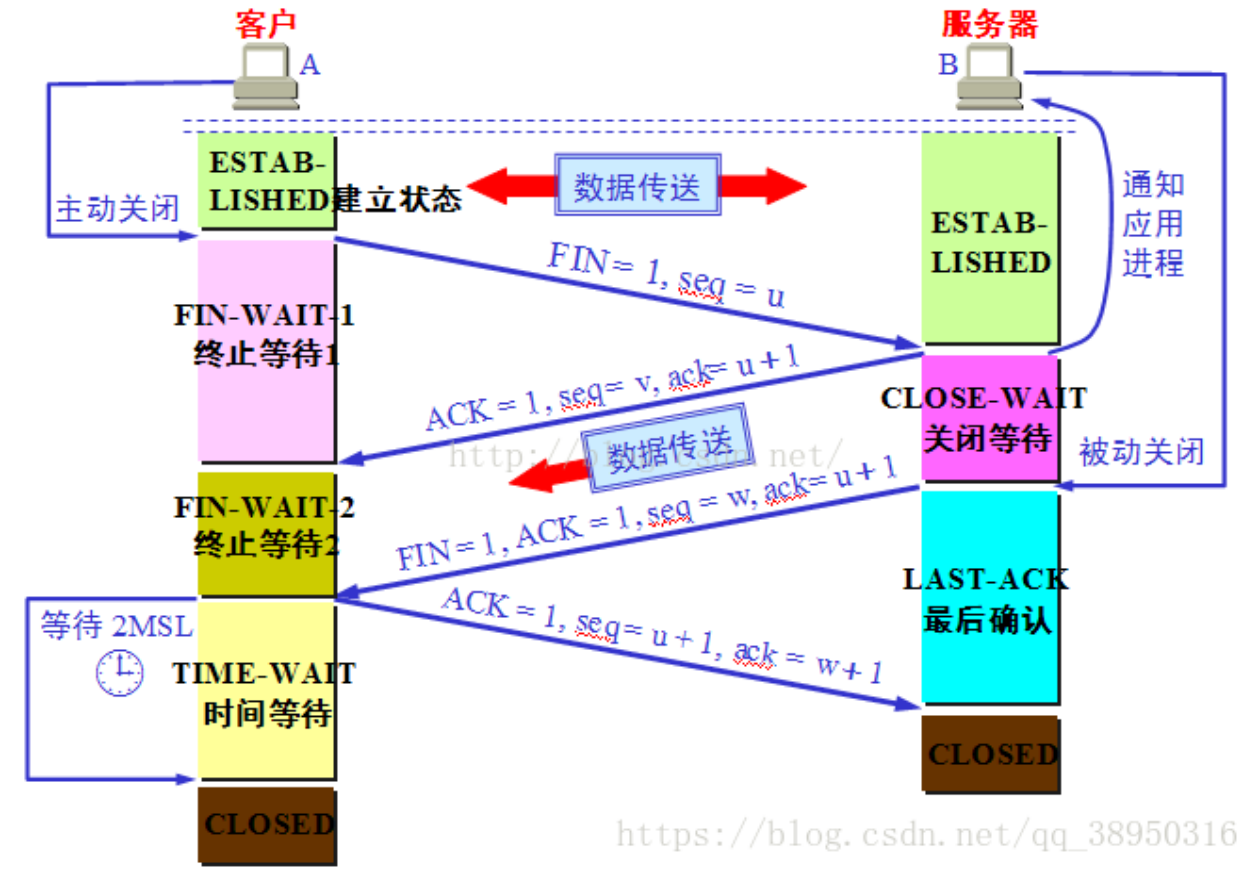


第一次握手：建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

四次挥手过程理解



1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过去的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。 2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。 3）客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。 4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。 5）客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2*MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。 6）服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

参考

TCP的三次握手与四次挥手理解及面试题（很全面）

为什么握手是三次挥手是四次?

https里用户跟服务器发送请求除了你刚刚说的random1外还有哪些东西?（这个当时没答出来回来wireshark抓包看了下是版本号和客户端支持的加密套件）

url请求过程?具体在说下dns解析的过程?

dns本地服务器如果被黑客黑了怎么办?

TCP为什么不能两次握手?

答:

为什么不采用两次握手? 如果是两次握手的情景: 客户端在发送一个连接建立请求之后进入等待状态，等到服务端确认之后就进入established状态。服务端在发送一个确认连接建立请求报文之后(不管客户端是否有回应)也进入established状态。 这就好比， A给B打电话， A:你听得到我说话吗? B:我听得到啊 A和B就都以为对方都能听得到自己了。 但有一种情况是，B的麦是坏的，A根本就听不到B说话，结果A没收到B的回应，但B却以为A能听得到他，B就一直等着A说点什么...这样让B身心俱疲。

三次握手: 客户端在发送一个连接建立请求报文之后进入等待状态，等到服务端返回确认建立连接的通知; 服务端发送确认建立连接请求报文，同时向客户端发送连接建立请求报文，进入等待状态。

客户端接受到服务端发送的确认请求报文。进入established状态。客户端接受到来自服务端的连接建立请求报文，发送确认连接建立请求报文。

服务端接受到来自客户端的确认建立连接报文，进入established。

如果是三次握手，则会是这样: A:你能听得到我说话吗? B:我听得到啊，你能不能听得到我说话? A:我也听得到你啊。(established) B:(established)

这样子A和B都能明确知道对方都能听到自己说话了。 这样A和B就能安心煲电话粥了。从此过上幸福的快乐的生活。

三次握手是因为，作为连接的一方，都要让对方明白自己知道对方的意见。

第一次握手，a主动请求建立，b收到。

第二次握手，a收到来自b的应答，此刻a知道了b同意了。

第三次握手，b收到来自a的应答，此刻b知道了a知道它发送给a的请求。

至此，连接双方a、b都知道了对方知道自己的话被知道了，连接可以建立了。

4次挥手的过程?

答：客户和服务器都可以首先发送关闭连接命令。例如当客户首先发起关闭连接命令时，它会发送FIN报文，然后服务器发送ACK报文。接着服务器发送FIN报文，然后客户发送ACK确认报文。客户机等待TIMEWAIT时间，过后关闭连接。

H4 TIMEWAIT的存在意义?

答：虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

TCP片段

H4 11.OSI7层协议是哪7层?MAC地址在那一层?

答：应用层、表示层、会话层、传输层、网络层、数据链路层、物理层。MAC地址是在数据链路层。