前言

这个文档的内容**纯手打**，如果想要看更多的干货文章，关注我的公众号：**Java3y**。有更多的原创技术文章和干货！

目前疯狂处于**疯狂**更新PDF中，只要是Java后端的知识，都会有！**欢迎来我公众号催更！**微信搜索：**Java3y**

如果文档中有任何的不懂的问题，都可以直接来找我询问，我乐意帮助你们！公众号有我的**联系方式**

- 🔥**Java精美脑图**
- 🔥**Java学习路线**
- 🔥**开发常用工具**
- 🔥**精美原创电子书**

在公众号下回复「**888**」即可获取！！

学习不能盲目，跟着我，会让你事半功倍

电子书的整理也是挺不容易，如果你觉得有帮助，想要打赏作者，那么可以通过这个收款码打赏我，**金额不重要，心意最重要**。主要是我可以通过这个打赏情况来预计大家对这本电子书的评价，嘻嘻

HTTP入门

1. 为什么要学HTTP?

我们绝大多数的Web应用都是基于HTTP来进行开发的。我们对Web的操作都是通过HTTP协议来进行传输数据的。

简单来说，**HTTP协议就是客户端和服务器交互的一种通迅的格式**。

HTTP的诞生主要是为了能够**让文档之间相互关联，形成超文本可以互相传阅**

可以说，Http就是Web通信的基础，这是我们必学的。

2. HTTP基础概念

我们学计算机网络的时候就知道，我们把计算机网络分层了5层，一般我们现在用的都是TCP/IP这么一个分层结构。

虽然官方的是ISO 提出的7层结构，但是仅仅是理论基础，在实际上大多人都是使用TCP/IP的分层结构

首先，我们先得知道，为什么我们要在计算机网络中分层次？？？

因为如果两台计算机能够相互通信的话，实际实现起来是非常困难操作的...我们**分层的目的就是为了将困难的问题简单化**，并且如果我们分层了，我们在使用的时候就可**以仅仅关注我们需要关注的层次，而不用理会其他层**。

如果需要改动设计的时候，我们只需要把变动的层替换即可，并不用涉及到其他的层次。这与我们程序设计中的低耦合是一个概念。

而我们的HTTP协议是在最上层，也就是应用层。这是最贴近我们的程序员的层次。

3.网站通信粗略过程

我们知道HTTP是在应用层中的，显然，**我们在Web通信的过程中，不仅仅是需要HTTP协议的，还会涉及到其他的协议的**。

DNS：负责解析域名

- 我们访问一个网页的时候，往往是通过域名来访问的`www.zhongfucheng.site`,而计算机通信只认的是我们的主机地址(192.168.xxx.xxx)，因此，当我们输入域名的时候，需要DNS把域名解析成主机来进行访问。

HTTP：产生请求报文数据

- 当我们对Web页面进行操作的时候，就会产生HTTP报文数据，请求对应的服务端进行响应。

TCP协议：分割HTTP数据，保证数据运输

- TCP协议**采用了三次握手的方式来保证数据的准确运输**，在运输的数据的时候，发送标识过去给服务器，服务器也返回标识给客户端，而客户端收到消息后再次返回标识给服务器。这样一来就保证了数据运输是可靠的。

IP协议：传输数据包，找到通信目的地地址。

- IP协议把我们的产生的数据包发送给对方，IP地址指明了节点被分配的地址，但IP地址可能会变换，**我们可以使用ARP协议来将IP地址反射为MAC地址**。MAC地址是不会更改的，是网卡所属的固定地址。
- 在找到通信目的地之前，**我们是需要不断的中转的，这过程我们称作为："路由中转"**，我们并不知道路由中转了多少次的。因此是不能全面了解到互联网中的传输状况的。

接下来就离我们比较远了，属于硬件相关的了，也就是链路层和物理层。以后复习到计算机网络的时候再来补充吧！

我们网页上请求数据就是上边这么一个流程。

4. 告知服务器请求的意图

我们如果开发过Web程序的话，我们知道常用的提交方式有POST和GET方法

我们也知道GET是用来获取数据的，POST是用来提交数据的。

其实HTTP协议中还支持着其他的方法，比如：Input、Delete、OPTIONS很多这样的方法。而由于常用，于是我们也可能仅仅知道GET和POST方法了。

HTTP提供方法的目的就是为了告知服务器该客户端想进行什么操作。当HTTP是OPTIONS方法的时候，服务器端就会返回它支持什么HTTP方法。

当然了，**现在RESTful盛行，也就是充分利用了HTTP协议的这些方法。**

5. HTTP是不保存状态的协议

HTTP是无状态的，也就是说，**它是不对通信状态进行保存的。它并不知道之前通信的对方是谁**。这样设计的目的就是为了让HTTP简单化，能够快速处理大量的事务！

但是，我们经常是需要知道访问的人是谁，于是就有了Cookie技术了。

- 要是服务器端想要记住客户端是谁，那么就颁发一个cookie给客户端
- 客户端把Cookie保存在硬盘中，当下次访问服务器的时候，浏览器会自动把客户端的cookie带过去。
- 就这样，服务器就能够知道这家伙是谁了。

6.持久连接

在HTTP1.0的时候，每一次进行HTTP通信就会断开一次连接。如果容量很少的文本传输是没有问题的。但是如果我们访问一个网页，该网页有非常多的图片。一个图片就算上一个HTTP请求了。那么在中途中就不断地建立TCP连接、获取图片、断开TCP连接。

这样是非常浪费资源的，因此在HTTP1.1版本，就是持久连接了。**一次HTTP连接能够处理多个请求**。

持久连接为“管线化”方式发送成为了可能：**在一次HTTP连接里面，不需要等待服务器响应请求，就能够继续发送第二次请求**。

7.提升传输效率

在说明之前，首先我们要知道什么是实体主体

- **实体主体就是作为数据在HTTP中传输的数据**。

一般地，**实体主体可以等价为报文主体，报文主体是HTTP中的一部分**。

我们如果不使用任何手段，服务器返回的数据实体主体是原样返回的。我们可以使用两种方式来提高传输效率

- **使用压缩技术把实体主体压小，在客户端再把数据解析**
- **使用分块传输编码，将实体主体分块传输，当浏览器解析到实体主体就能够显示了。**

我们如果在下载东西的过程中断了，按照以前我们是需要重新下载的，但是现在可以在中断中继续下载。我们可以**使用到获取范围数据，这种叫做范围请求**！

这种请求只会下载资源的一部分。

- 比如我的图片下载到一半了，我们只需要下载另一半就可以组成一张完整的图片了。那么**请求的时候请求没有下载的一部分即可。**

如果文档中有任何的不懂的问题，都可以直接来找我询问，我乐意帮助你们！微信搜**Java3y**公众号有我的联系方式。更多**原创**技术文章可关注我的GitHub：https://github.com/ZhongFuCheng3y/3y

8.常用的状态码简述

2XX

一般是请求成功

200 正常处理

204 成功处理，但服务器没有新数据返回，显示页面不更新

206 对服务器进行范围请求，只返回一部分数据

3XX

一般表示重定向

301 请求的资源已分配了新的URI中，URL地址改变了。【永久重定向】

302 请求的资源临时分配了新的URI中，URL地址没变【临时重定向】

303 与302相同的功能，但明确客户端应该采用GET方式来获取资源

304 发送了附带请求，但不符合条件【返回未过期的缓存数据】

307 与302相同，但不会把POST请求变成GET

4XX

表示客户端出错了。

400 请求报文语法错误了

401 需要认证身份

403 没有权限访问

404 服务器没有这个资源

5XX

服务器出错了

500 内部资源出错了

503 服务器正忙

9.服务器与客户端之间的应用程序

首先要说的是，一个HTTP服务器可以拥有多个站点，也就是说：**HTTP下可以配置多个虚拟主机。当用户访问不同主机的时候，实际上都是访问同一台HTTP服务器。**

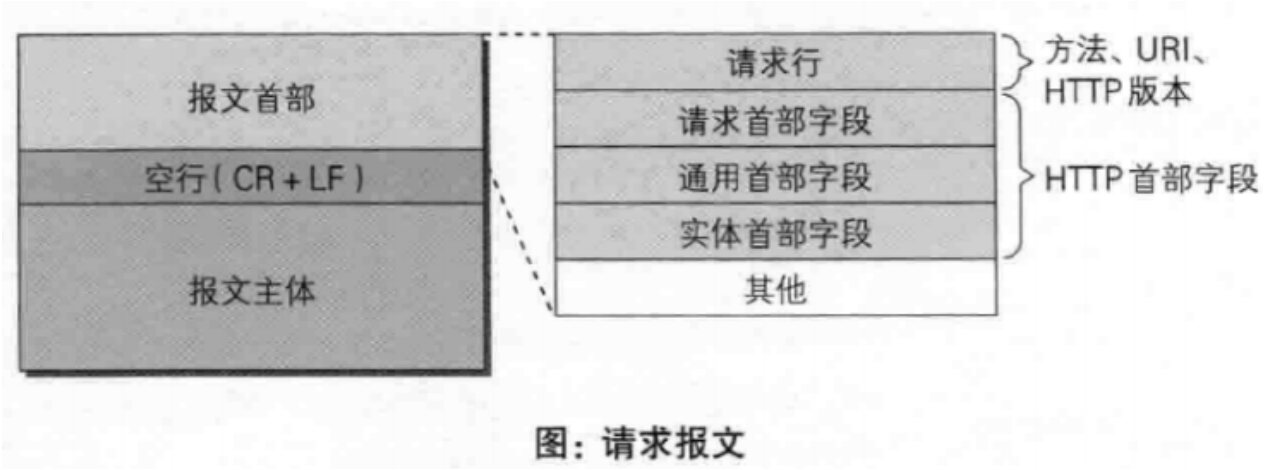
在客户端和服务器中还有一些用于**通信数据转发的应用程序**：

- 代理
  - 可以用来缓存数据，当代理缓存了数据以后，客户端就可以直接用代理获取数据
  - 可以用来对网站进行访问控制，获取访问日志记录
- 网关
  - 能够提供非HTTP请求的操作，访问数据库什么的
- 隧道
  - 建立一条安全的通信路径，可以使用SSL等加密手段进行通信。

10. HTTP首部简述

10.1 HTTP请求报文

HTTP请求报文：在请求中，HTTP报文由方法、URI、HTTP版本、HTTP首部字段等部分组成。



图：请求报文

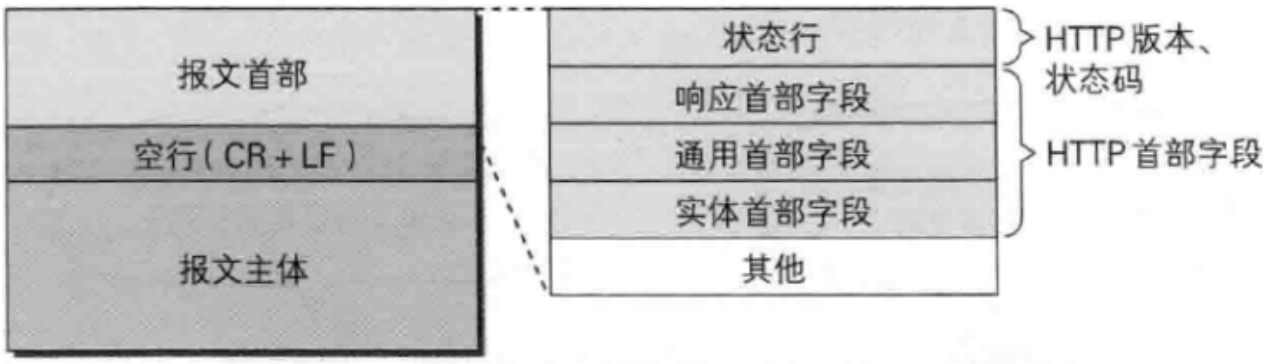
1. 请求行【描述客户端的**请求方式**、**请求的资源名称**，以及使用的**HTTP协议版本号**】
2. 首部字段【描述客户端请求哪台主机，以及**客户端的一些环境信息**等】
3. 一个空行

首部字段例子：

- Accept: text/html,image/*　【浏览器告诉服务器，它支持的数据类型】
- Accept-Charset: ISO-8859-1　【浏览器告诉服务器，它支持哪种**字符集**】
- Accept-Encoding: gzip,compress 【浏览器告诉服务器，它支持的**压缩格式**】
- Accept-Language: en-us,zh-cn 【浏览器告诉服务器，它的语言环境】
- Host: www.it315.org:80【浏览器告诉服务器，它的想访问哪台主机】
- If-Modified-Since: Tue, 11 Jul 2000 18:23:51 GMT【浏览器告诉服务器，缓存数据的时间】
- Referer: http://www.it315.org/index.jsp【浏览器告诉服务器，客户机是从那个页面来的---**反盗链**】
- 8.User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)【浏览器告诉服务器，浏览器的内核是什么】
- Cookie【浏览器告诉服务器，**带来的Cookie是什么**】
- Connection: close/Keep-Alive　【浏览器告诉服务器，请求完后是断开链接还是保持链接】
- Date: Tue, 11 Jul 2000 18:23:51 GMT【浏览器告诉服务器，请求的时间】

10.2 HTTP响应报文

HTTP响应报文：在响应中，HTTP报文由HTTP版本、状态码（数字和原因短语）、HTTP首部字段3部分组成。



图：响应报文

1. 一个状态行【用于描述**服务器对请求的处理结果。**】

2. 首部字段【用于描述**服务器的基本信息**，以及**数据的描述**，**服务器通过这些数据的描述信息，可以通知客户端如何处理等一会儿它回送的数据】**
3. 一个空行
4. 实体内容**【服务器向客户端回送的数据】**

状态行：

- 格式： HTTP版本号　状态码　原因叙述
- 状态行：HTTP/1.1　200　OK
- 状态码用于表示**服务器对请求的处理结果**，它是一个**三位的十进制数**。响应状态码分为5类

状态码	含义
100～199	表示成功接收请求，要求客户端继续提交下一次请求才能完成整个处理过程
200～299	表示成功接收请求并已完成整个处理过程，常用200
300～399	为完成请求，客户需进一步细化请求。例如，请求的资源已经移动一个新地址，常用302、307和304
400～499	客户端的请求有错误，常用404
500～599	服务器端出现错误，常用 500

首部字段例子：

- Location: http://www.it315.org/index.jsp 【服务器告诉浏览器**要跳转到哪个页面】**
- Server:apache tomcat【服务器告诉浏览器，服务器的型号是什么】
- Content-Encoding: gzip 【服务器告诉浏览器**数据压缩的格式】**
- Content-Length: 80 【服务器告诉浏览器回送数据的长度】
- Content-Language: zh-cn 【服务器告诉浏览器，服务器的语言环境】
- Content-Type: text/html; charset=GB2312 【服务器告诉浏览器，**回送数据的类型】**
- Last-Modified: Tue, 11 Jul 2000 18:23:51 GMT【服务器告诉浏览器该资源上次更新时间】
- Refresh: 1;url=http://www.it315.org【服务器告诉浏览器要**定时刷新】**
- Content-Disposition: attachment; filename=aaa.zip【服务器告诉浏览器**以下载方式打开数据】**
- Transfer-Encoding: chunked 【服务器告诉浏览器数据以分块方式回送】
- Set-Cookie:SS=Q0=5Lb_nQ; path=/search【服务器告诉浏览器要**保存Cookie】**
- Expires: -1【服务器告诉浏览器**不要设置缓存】**
- Cache-Control: no-cache 【服务器告诉浏览器**不要设置缓存】**
- Pragma: no-cache 【服务器告诉浏览器**不要设置缓存】**
- Connection: close/Keep-Alive 【服务器告诉浏览器连接方式】
- Date: Tue, 11 Jul 2000 18:23:51 GMT【服务器告诉浏览器回送数据的时间】

11.HTTPS简述

HTTP在安全上是不足的

- 通信使用明文【没有加密过内容的】
- 不验证通信方身份，无论是客户端和服务器，都是随意通信的
- 无法证明报文的完整性【别人监听后，可以篡改】

我们一般在上网时，使用抓包工具就很容易获取到HTTP请求的信息了，这是TCP/IP在网络通信中无法避免的。

假设我们对HTTP报文进行加密了， 那也仅仅是是内容的加密。别人获取到了HTTP内容了，即使无法破解HTTP内容，还是能够篡改的。

我们最好就是**使用SSL建立安全的通信线路**，就可以在这条线路上进行HTTP通信了。

其实HTTPS就是披着SSL的HTTP...

HTTPS使用的是共享密钥和公开私有密钥混合来进行加密的。由于公开私有密钥需要太多的资源，不可能一直以公开私有密钥进行通信。因此，**HTTP在建立通信线路的时候使用公开私有密钥，当建立完连接后，随后就使用共享密钥进行加密和解密了**

对于认证方面，HTTPS是**基于第三方的认证机构来获取认受认可的证书**、因此，可以从中认证该服务器是否是合法的。

而客户端方面则需要自己购买认证证书、这实施起来难度是很大的【认证证书需要钱】。

所以，一般的网站都是使用表单认证就算了，这是用得最广泛的客户端认证了。

如果文档中有任何的不懂的问题，都可以直接来找我询问，我乐意帮助你们！微信搜**Java3y**公众号有我的联系方式。更多**原创**技术文章可关注我的GitHub：https://github.com/ZhongFuCheng3y/3y

HTTP/2 && HTTPS

1. HTTP协议的今生来世

到现在为止，HTTP协议已经有四个版本了：

- HTTP1.0
- HTTP1.1
- HTTP/2
- HTTP/3

下面就简单聊聊他们三者的区别，以及整理一些必要的额外知识点。

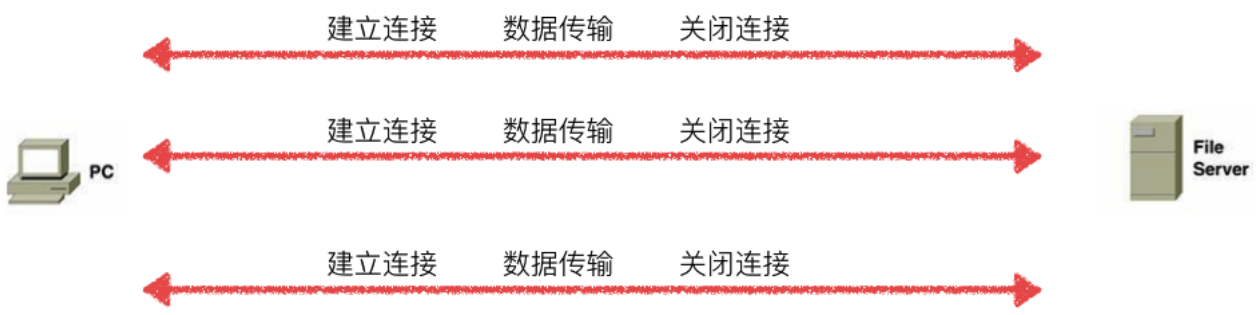
(HTTP/3暂不聊，有兴趣的可以去搜搜)

1.1 HTTP1.0和HTTP1.1区别

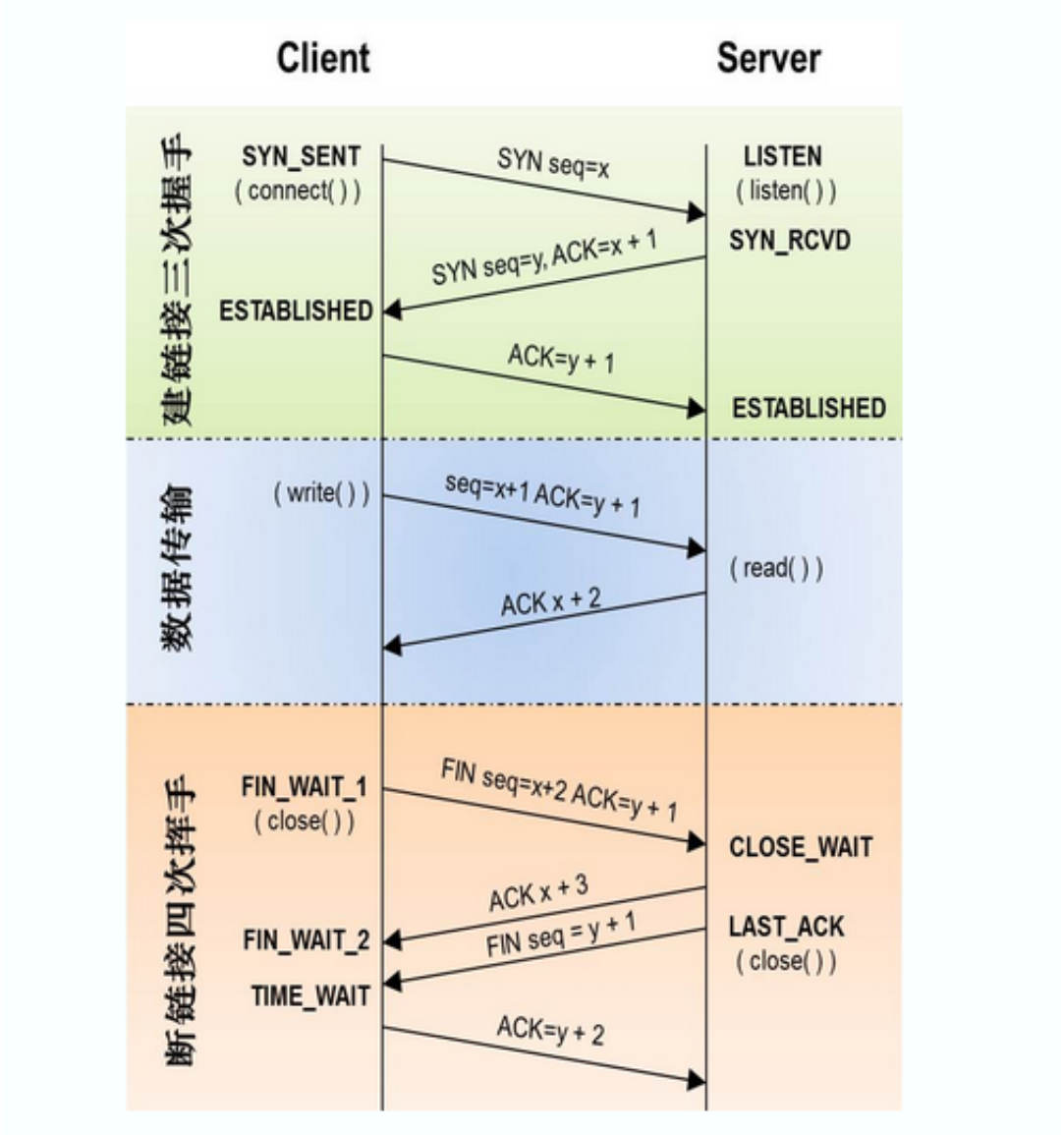
HTTP1.0和HTTP1.1最主要的区别就是：

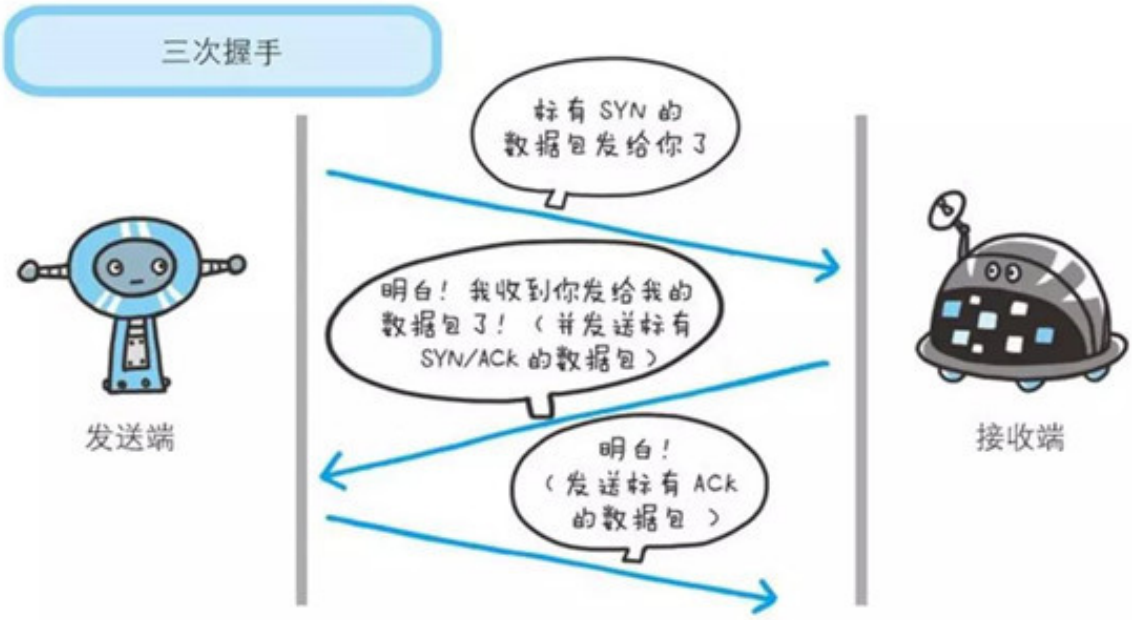
- HTTP1.1默认是**持久化连接**！

在HTTP1.0默认是短连接：



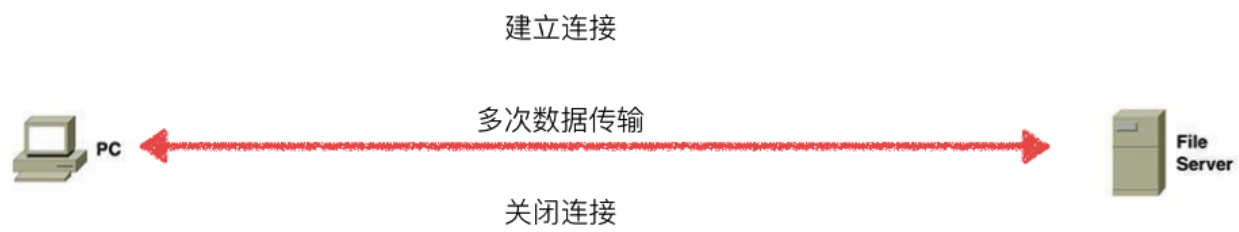
简单来说就是：**每次与服务器交互，都需要新开一个连接**！





试想一下：请求一张图片，新开一个连接，请求一个CSS文件，新开一个连接，请求一个JS文件，新开一个连接。HTTP协议是基于TCP的，TCP每次都要经过**三次握手，四次挥手，慢启动**...这都需要去消耗我们非常多的资源的！

在HTTP1.1中默认就使用持久化连接来解决：**建立一次连接，多次请求均由这个连接完成**！(如果阻塞了，还是会开新的TCP连接的)



相对于持久化连接还有另外比较重要的改动：

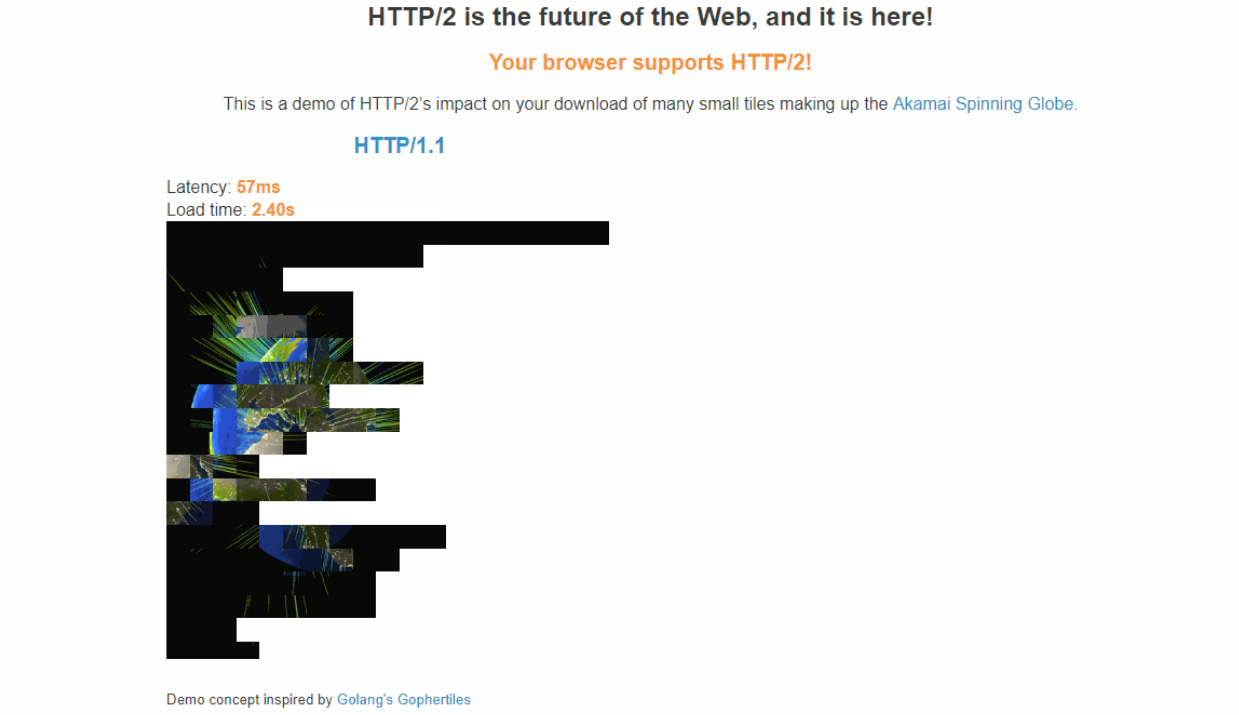
- HTTP 1.1增加host字段
- HTTP 1.1中引入了`Chunked transfer-coding`，范围请求，实现断点续传(实际上就是利用HTTP消息头使用分块传输编码，将实体主体分块传输)
- HTTP 1.1管线化(pipelining)理论，客户端可以同时发出多个HTTP请求，而不用一个个等待响应之后再请求
  - 注意：这个pipelining仅仅是**限于理论场景下**，大部分桌面浏览器仍然会**选择默认关闭**HTTP pipelining！
  - 所以现在使用HTTP1.1协议的应用，都是有**可能会开多个TCP连接**的！

参考资料：https://www.cnblogs.com/gofighting/p/5421890.html

1.2 HTTP2基础

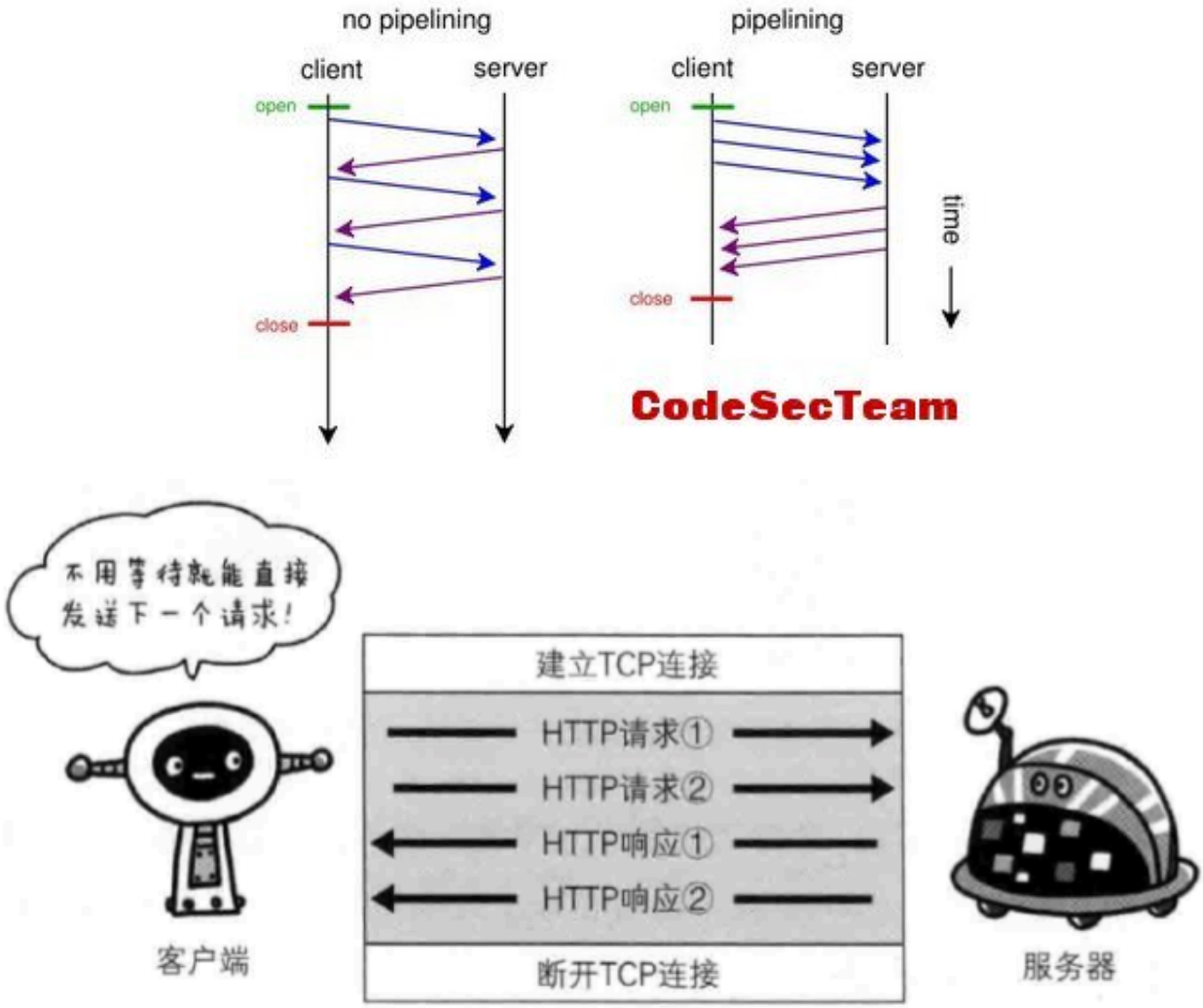
在说HTTP2之前，不如先直观比较一下HTTP2和HTTP1.1的区别：

- https://http2.akamai.com/demo



上面也已经说了，HTTP 1.1提出了管线化(pipelining)理论，但是仅仅是限于理论的阶段上，这个功能默认还是关闭了的。

管线化(pipelining)和非管线化的**区别**：



图：不等待响应，直接发送下一个请求

> HTTP Pipelining其实是把多个HTTP请求放到一个TCP连接中一一发送，而在发送过程中不需要等待服务器对前一个请求的响应；只不过，**客户端还是要按照发送请求的顺序来接收响应！**

> 就像在超市收银台或者银行柜台排队时一样，你并不知道前面的**顾客**是干脆利索的还是会跟收银员/柜员磨蹭到世界末日（不管怎么说，服务器（即收银员/柜员）是要按照顺序处理请求的，如果**前一个请求非常耗时（顾客磨蹭）**，那么后续请求都会受到影响。

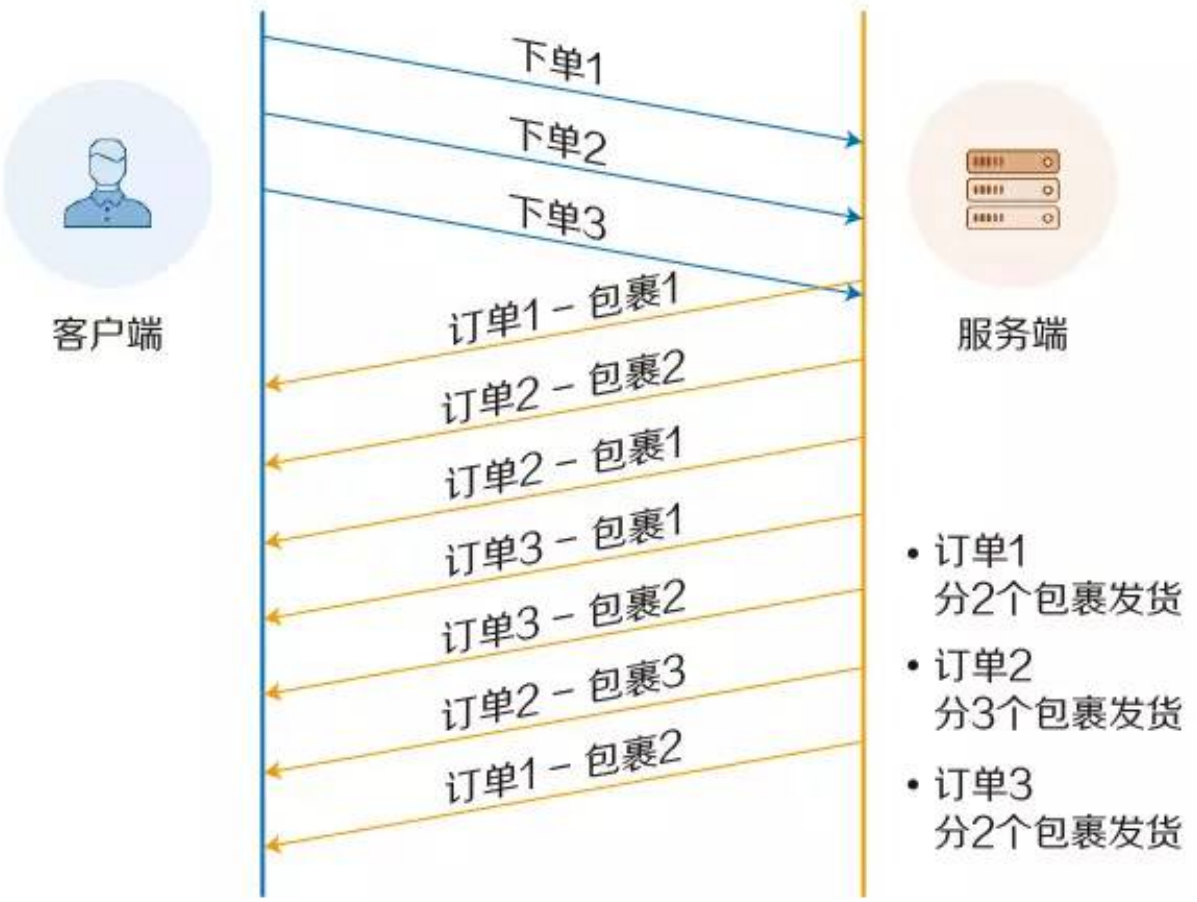
- 在HTTP1.0中，发送一次请求时，需要**等待服务端响应了**才可以继续发送请求。
- 在HTTP1.1中，发送一次请求时，不需要等待服务端响应了就可以发送请求了，但是回送数据给客户端的时候，客户端还是需要按照**响应的顺序**来一一接收
- 所以说，无论是HTTP1.0还是HTTP1.1提出了Pipelining理论，还是会出现**阻塞**的情况。从专业的名词上说这种情况，叫做**线头阻塞**（Head of line blocking）简称：HOLB

1.3 HTTP1.1和HTTP2区别

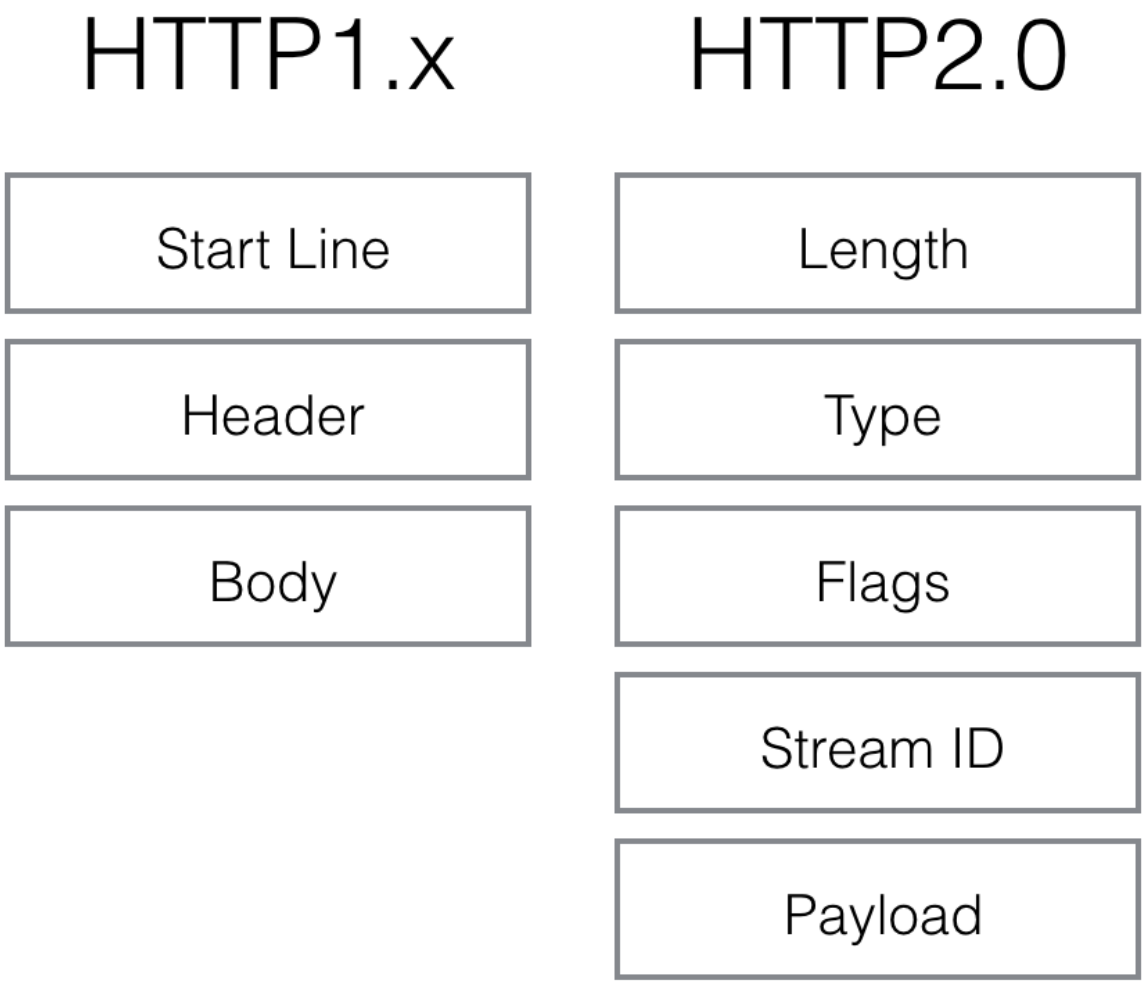
HTTP2与HTTP1.1最重要的区别就是**解决了线头阻塞的**问题！其中最重要的改动是：**多路复用(Multiplexing)**

- 多路复用意味着线头阻塞将不在是一个问题，允许同时通过单一的 HTTP/2 连接发起**多重的请求-响应消息**，合并多个请求为一个的优化将不再适用。
    - (我们知道：HTTP1.1中的Pipelining是没有付诸于实际的)，之前为了**减少**HTTP请求，有很多操作将多个请求合并，比如：Spriting(多个图片合成一个图片)，内联Inlining(将图片的原始数据嵌入在CSS文件里面的URL里），拼接Concatenation(一个请求就将其下载完多个JS文件)，分片Sharding(将请求分配到各个主机上)……

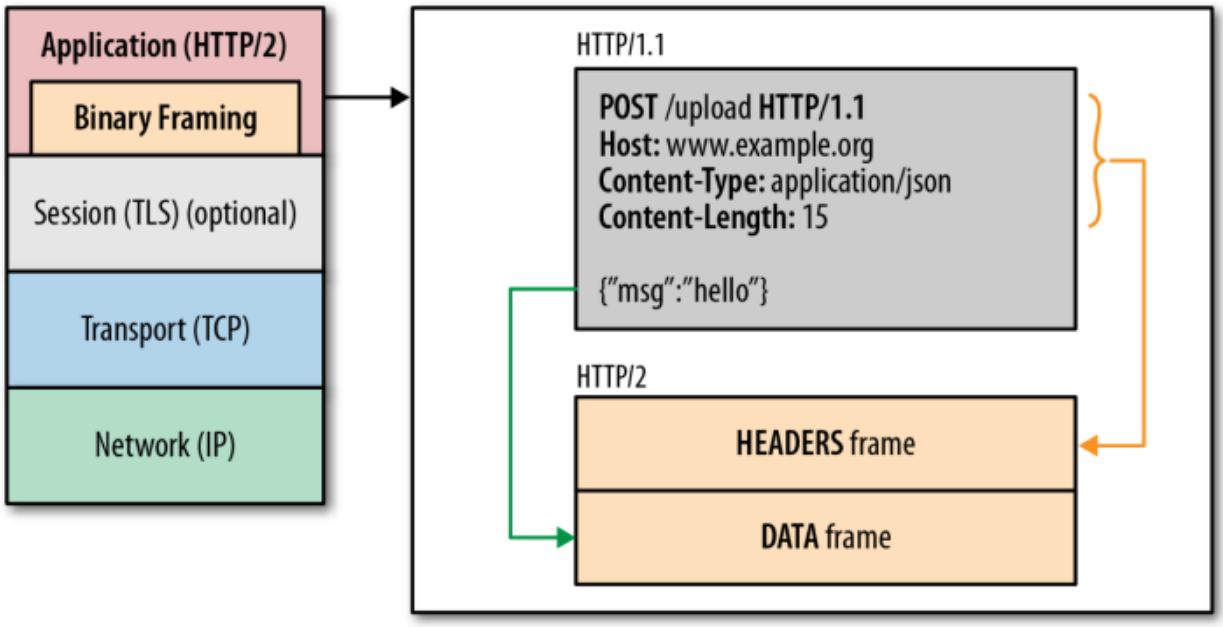
使用了HTTP2可能是这样子的：



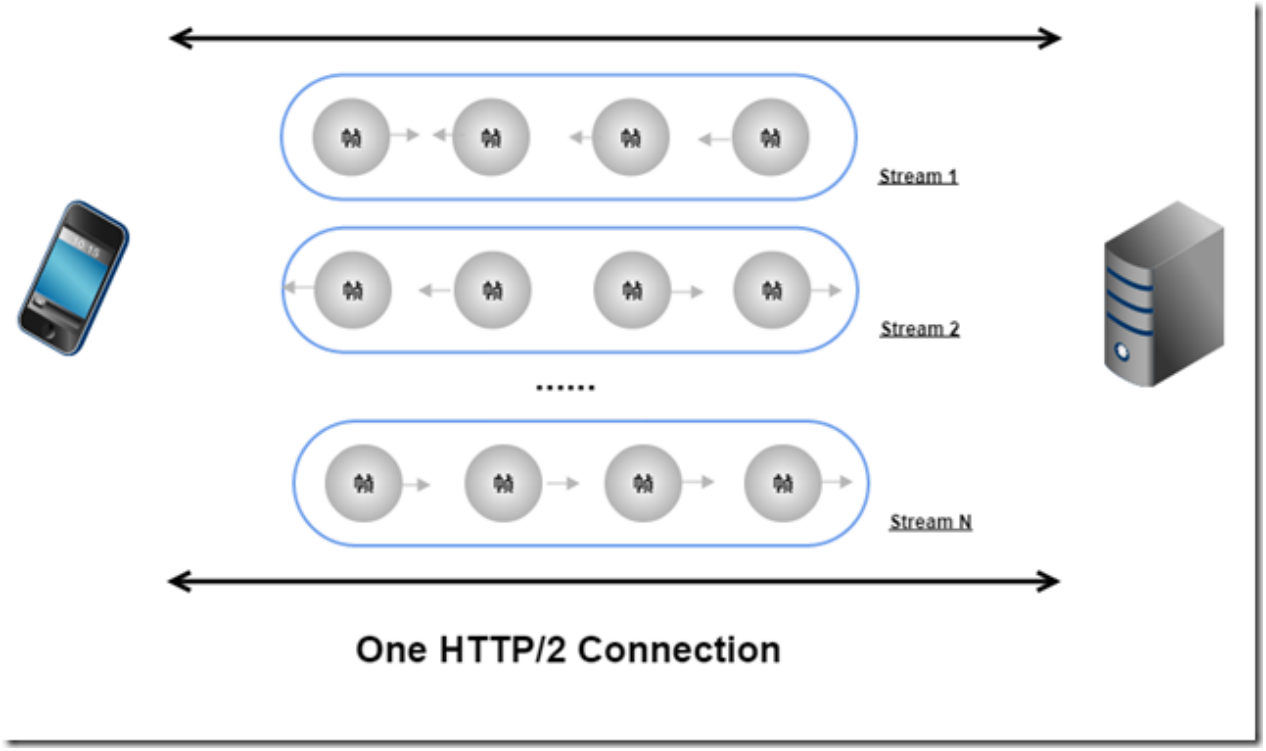
HTTP2所有性能增强的核心在于**新的二进制分帧层**(不再以文本格式来传输了)，它定义了如何封装http消息并在客户端与服务器之间传输。



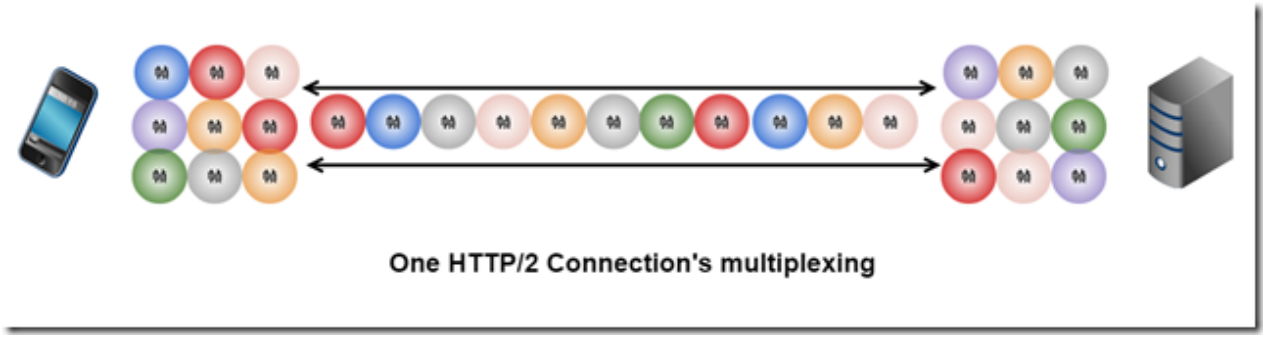
看上去协议的格式和HTTP1.x完全不同了，**实际上HTTP2并没有改变HTTP1.x的语义**，只是把原来HTTP1.x的header和body部分用**frame重新封装了一层**而已



HTTP2连接上**传输的每个帧都关联到一个"流"**。流是一个独立的，双向的帧序列可以通过一个HTTP2的连接在服务端与客户端之间不断的交换数据。



实际上运输时：



HTTP2还有一些比较重要的改动：

- 使用HPACK对HTTP/2头部压缩
- 服务器推送
  - HTTP2推送资料：https://segmentfault.com/a/1190000015773338
- 流量控制
  - 针对传输中的**流**进行控制(TCP默认的粒度是针对连接)
- 流优先级（Stream Priority）它被用来告诉**对端哪个流更重要**。

2.2 HTTP2总结

HTTP1.1新改动：

- **持久连接**
- 请求管道化
- 增加缓存处理（新的字段如cache-control）
- 增加Host字段、支持断点传输等

HTTP2新改动：

- 二进制分帧
- **多路复用**
- 头部压缩
- 服务器推送

2.3HTTPS再次回顾

之前在面试的时候被问到了HTTPS，SSL这样的知识点，也没答上来，这里也简单整理一下。

首先还是来解释一下基础的东东：

- 对称加密：
  - 加密和解密都是用同一个密钥
- 非对称加密：
  - 加密用公开的密钥，解密用私钥
  - (私钥只有自己知道，公开的密钥大家都知道)
- 数字签名：
  - 验证传输的内容**是对方发送的数据**
  - 发送的数据**没有被篡改过**
- 数字证书（Certificate Authority）简称CA
  - 认证机构证明是**真实的服务器发送的数据**。

3y的通讯之路：

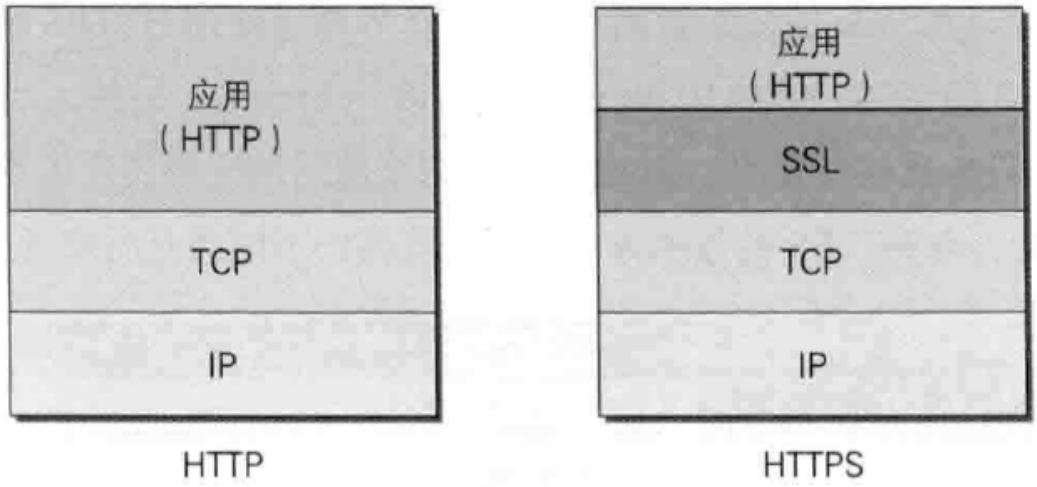
- 远古时代：3y和女朋友聊天传输数据之间没有任何的加密，直接传输
  - 内容被看得一清二楚，毫无隐私可言
- 上古时期：使用对称加密的方式来保证传输的数据只有两个人知道
  - 此时有个问题：**密钥不能通过网络传输**(因为没有加密之前，都是不安全的)，所以3y和女朋友先约见面一次，告诉对方密码是多少，再对话聊天。
- 中古时期：3y不单单要跟女朋友聊天，还要跟爸妈聊天的哇(同样不想泄漏了自己的通讯信息)。那有那么多人，难道每一次都要约来见面一次吗？(说明维护多个对称密钥是麻烦的！)--->所以用到了非对称加密
  - 3y自己保留一份密码，独一无二的(私钥)。告诉3y女朋友，爸妈一份密码(这份密码是公开的，谁都可以拿--->公钥)。让他们给我发消息之前，先用那份我告诉他们的密码加密一下，再发送给我。我收到信息之后，用自己独一无二的私钥解密就可以了！
- 近代：此时又出现一个问题：虽然别人不知道私钥是什么，拿不到你**原始传输**的数据，但是可以拿到加密后的数据，他们可以**改掉**某部分的数据再发送给服务器，这样服务器拿到的数据就**不是完整的**了。
  - 3y女朋友给3y发了一条信息”3y我喜欢你“，然后用3y给的公钥加密，发给3y了。此时不怀好意的人截取到这条加密的信息，他**破解不了原信息**。但是他可以**修改加密后的数据**再传给3y。可能3y拿到收到的数据就是”3y你今晚跪键盘吧“

- 现代：拿到的数据可能被篡改了，我们可以使用数字签名来解决被篡改的问题。数字签名其实也可以看做是**非对称加密的手段一种**，具体是这样的：得到原信息hash值，用**私钥**对hash值加密，**另一端**用**公钥**解密，最后比对hash值是否变了。如果变了就说明被篡改了。(一端用私钥加密，另一端用公钥解密，也确保了来源)
- 目前现在：好像使用了数字签名就万无一失了，其实还有问题。我们使用非对称加密的时候，是使用**公钥进行加密的**。如果**公钥被伪造了**，后面的数字签名其实就毫无意义了。讲到底：**还是可能会被中间人攻击**~此时我们就有了**CA认证机构来确认公钥的真实性**！

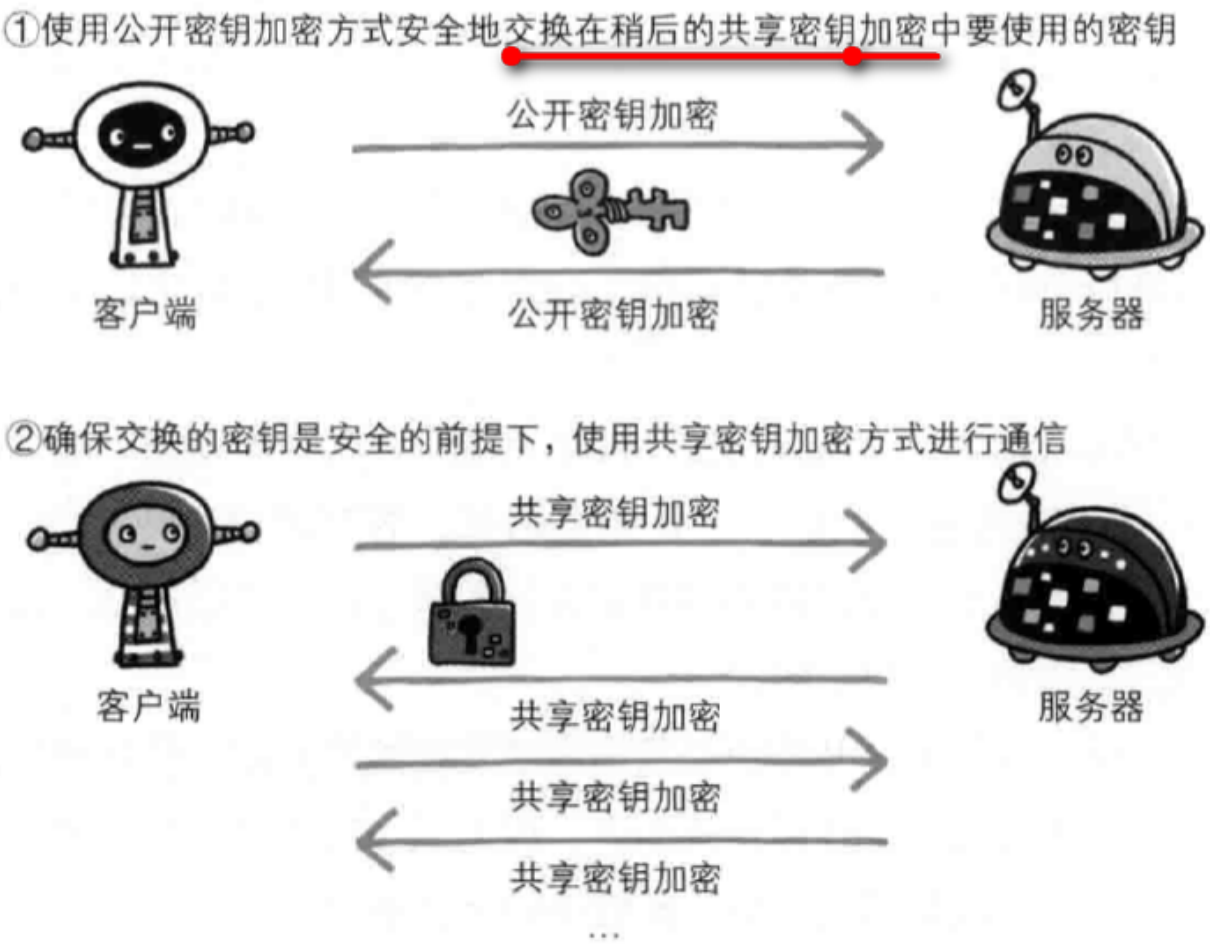
对于数字签名和CA认证还是不太了解参考一下

- 阮一峰：http://www.ruanyifeng.com/blog/2011/08/what_is_a_digital_signature.html
- 什么是数字签名和证书？https://www.jianshu.com/p/9db57e761255

回到我们的HTTPS，HTTPS其实就是在HTTP协议下多加了一层SSL协议(ps:现在都用TLS协议了)

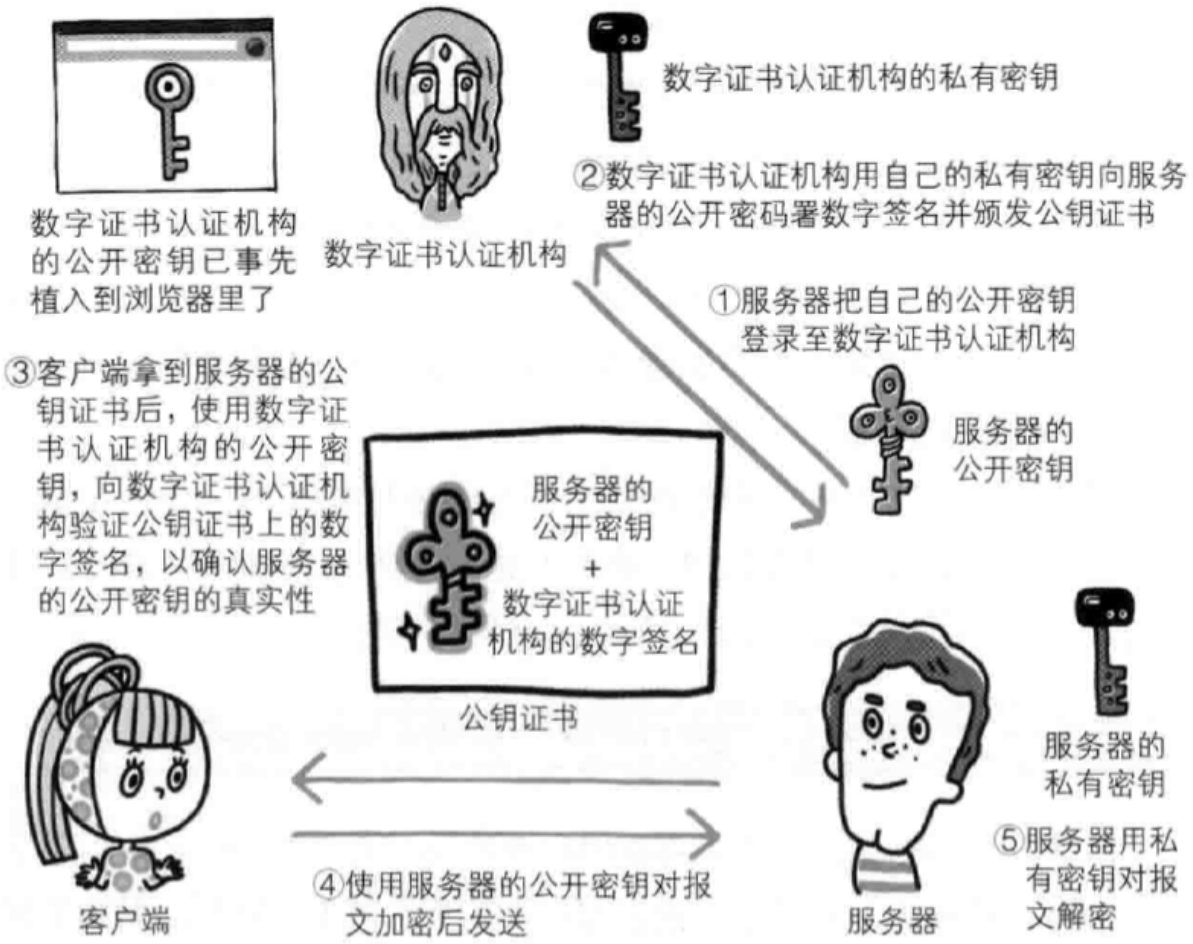


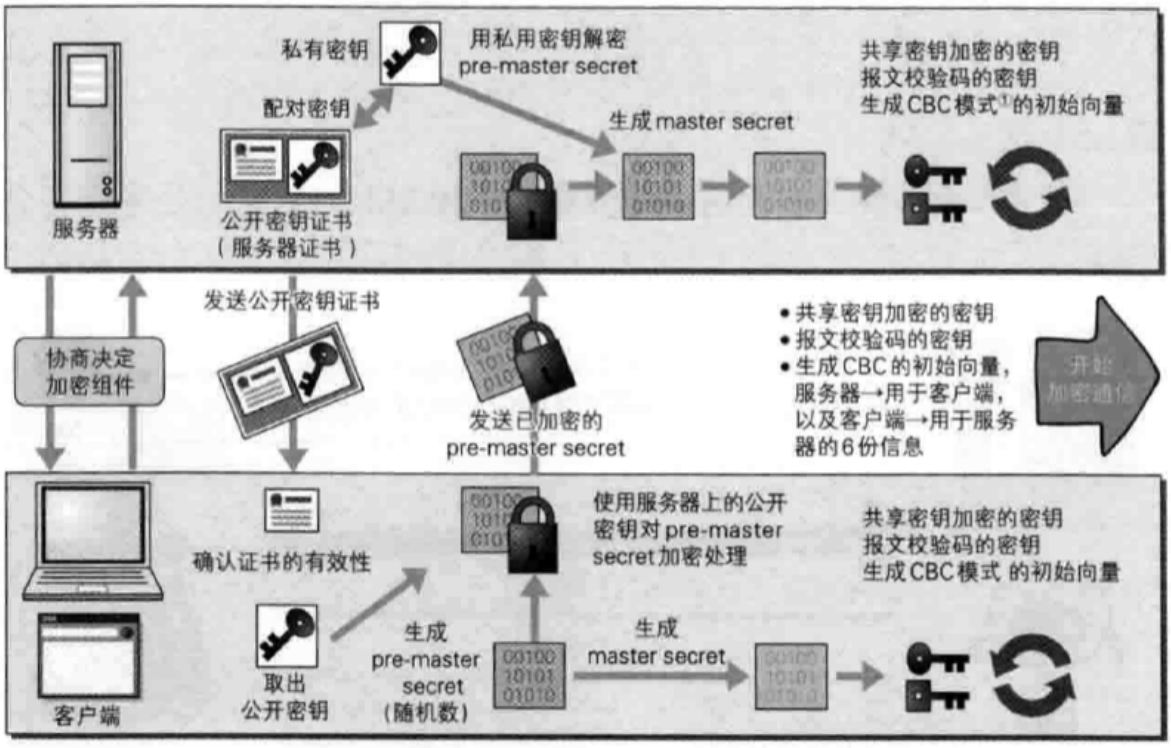
HTTPS采用的是**混合方式加密**：



图：混合加密机制

过程是这样子的：





- 用户向web服务器发起一个安全连接的请求
- 服务器返回经过CA认证的数字证书，证书里面包含了服务器的public key(公钥)

- 用户拿到数字证书，用自己浏览器内置的CA证书解密得到服务器的public key
- 用户用服务器的public key加密一个用于接下来的对称加密算法的密钥，传给web服务器
  - 因为只有服务器有private key可以解密，所以**不用担心中间人拦截这个加密的密钥**
- 服务器拿到这个加密的密钥，解密获取密钥，再使用对称加密算法，和用户完成接下来的网络通信

所以相比HTTP，HTTPS 传输更加安全

- （1） 所有信息都是加密传播，黑客无法窃听。
- （2） 具有校验机制，一旦被篡改，通信双方会立刻发现。
- （3） 配备身份证书，防止身份被冒充。

参考资料：

- 数字签名、数字证书、SSL、https是什么关系？https://www.zhihu.com/question/52493697/answer/131015846
- 浅谈SSL/TLS工作原理：https://zhuanlan.zhihu.com/p/36981565
- HTTPS:https://tech.upyun.com/article/192/HTTPS%E7%B3%BB%E5%88%97%E5%B9%B2%E8%B4%A7%EF%BC%88%E4%B8%80%EF%BC%89%EF%BC%9AHTTPS%20%E5%8E%9F%E7%90%86%E8%AF%A6%E8%A7%A3.html
- 网站HTTP升级HTTPS完全配置手册：https://www.cnblogs.com/powertoolsteam/p/http2https.html

如果文档中有任何的不懂的问题，都可以直接来找我询问，我乐意帮助你们！微信搜**Java3y**公众号有我的联系方式。更多**原创**技术文章可关注我的GitHub：https://github.com/ZhongFuCheng3y/3y

HTTP常见面试题

1.Http与Https的区别：

> Http与Https的区别：

1. HTTP 的URL 以http:// 开头，而HTTPS 的URL 以https:// 开头
2. HTTP 是不安全的，而 HTTPS 是安全的
3. HTTP 标准端口是80 ，而 HTTPS 的标准端口是443
4. 在OSI 网络模型中，HTTP工作于应用层，而HTTPS 的安全传输机制工作在传输层
5. HTTP 无法加密，而HTTPS 对传输的数据进行加密
6. HTTP无需证书，而HTTPS 需要CA机构颁发的SSL证书

2. 什么是Http协议无状态协议?怎么解决Http协议无状态协议?

- **无状态协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息**
  - **也就是说，当客户端一次HTTP请求完成以后，客户端再发送一次HTTP请求，HTTP并不知道当前客户端是一个"老用户"。**
- **可以使用Cookie来解决无状态的问题，Cookie就相当于一个通行证，第一次访问的时候给客户端发送一个Cookie，当客户端再次来的时候，拿着Cookie(通行证)，那么服务器就知道这个是"老用户"。**

3.URI和URL的区别

> URI和URL的区别

URI，是uniform resource identifier，统一资源标识符，用来唯一的标识一个资源。

- Web上可用的每种资源如HTML文档、图像、视频片段、程序等都是一个来URI来定位的
- URI一般由三部组成：
- ①访问资源的命名机制
- ②存放资源的主机名
- ③资源自身的名称，由路径表示，着重强调于资源。

URL是uniform resource locator，统一资源定位器，它是一种具体的URI，即URL可以用来标识一个资源，而且还指明了如何locate这个资源。

- URL是Internet上用来描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上，特别是著名的Mosaic。
- 采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL一般由三部组成：
- ①协议(或称为服务方式)
- ②存有该资源的主机IP地址(有时也包括端口号)
- ③主机资源的具体地址。如目录和文件名等

URN，uniform resource name，统一资源命名，是通过名字来标识资源，比如mailto:java-net@java.sun.com。

- URI是以一种抽象的，高层次概念定义统一资源标识，而URL和URN则是具体的资源标识的方式。URL和URN都是一种URI。笼统地说，每个 URL 都是 URI，但不一定每个 URI 都是 URL。这是因为 URI 还包括一个子类，即统一资源名称 (URN)，它命名资源但不指定如何定位资源。上面的 mailto、news 和 isbn URI 都是 URN 的示例。

在Java的URI中，**一个URI实例可以代表绝对的，也可以是相对的，只要它符合URI的语法规则。而URL类则不仅符合语义，还包含了定位该资源的信息，因此它不能是相对的。**

在Java类库中，URI类不包含任何访问资源的方法，它唯一的作用就是解析。

相反的是，URL类可以打开一个到达资源的流。

4. 常用的HTTP方法有哪些?

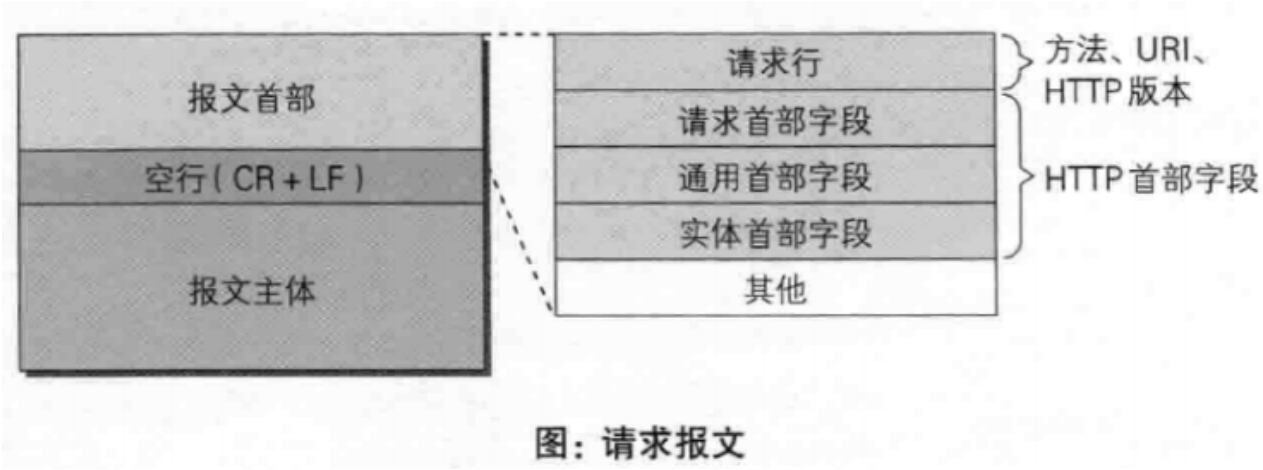
> 常用的HTTP方法有哪些?

- GET： 用于请求访问已经被URI（统一资源标识符）识别的资源，可以通过URL传参给服务器
- POST：用于传输信息给服务器，主要功能与GET方法类似，但一般推荐使用POST方式。
- PUT： 传输文件，报文主体中包含文件内容，保存到对应URI位置。
- HEAD： 获得报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URI是否有效。
- DELETE：删除文件，与PUT方法相反，删除对应URI位置的文件。
- OPTIONS：查询相应URI支持的HTTP方法。

5. HTTP请求报文与响应报文格式

> HTTP请求报文与响应报文格式

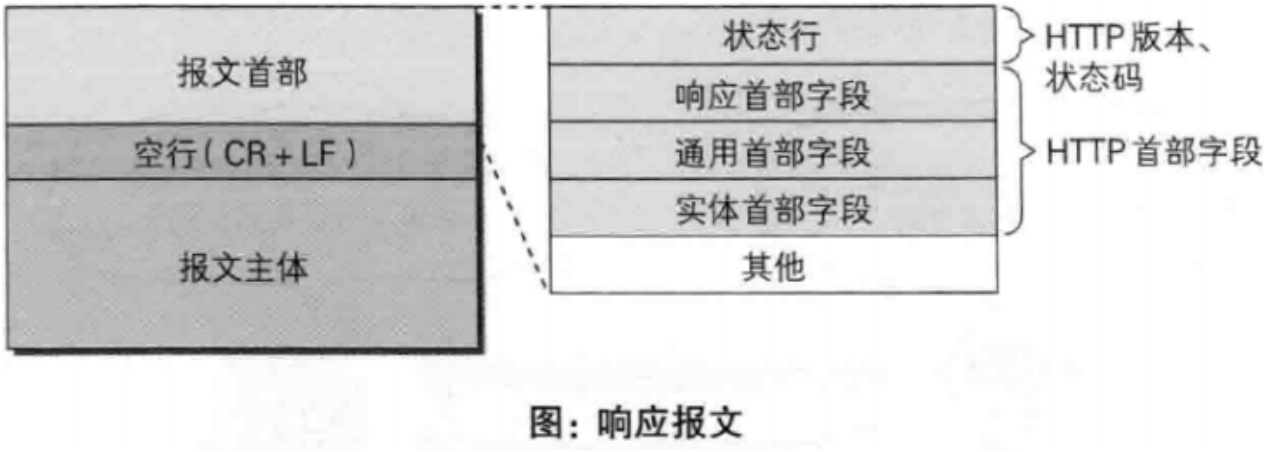
请求报文包含四部分：



图：请求报文

- a、请求行：包含请求方法、URI、HTTP版本信息
- b、请求首部字段
- c、请求内容实体
- d、空行

响应报文包含四部分：



图：响应报文

- a、状态行：包含HTTP版本、状态码、状态码的原因短语

- b、响应首部字段
- c、响应内容实体
- d、空行

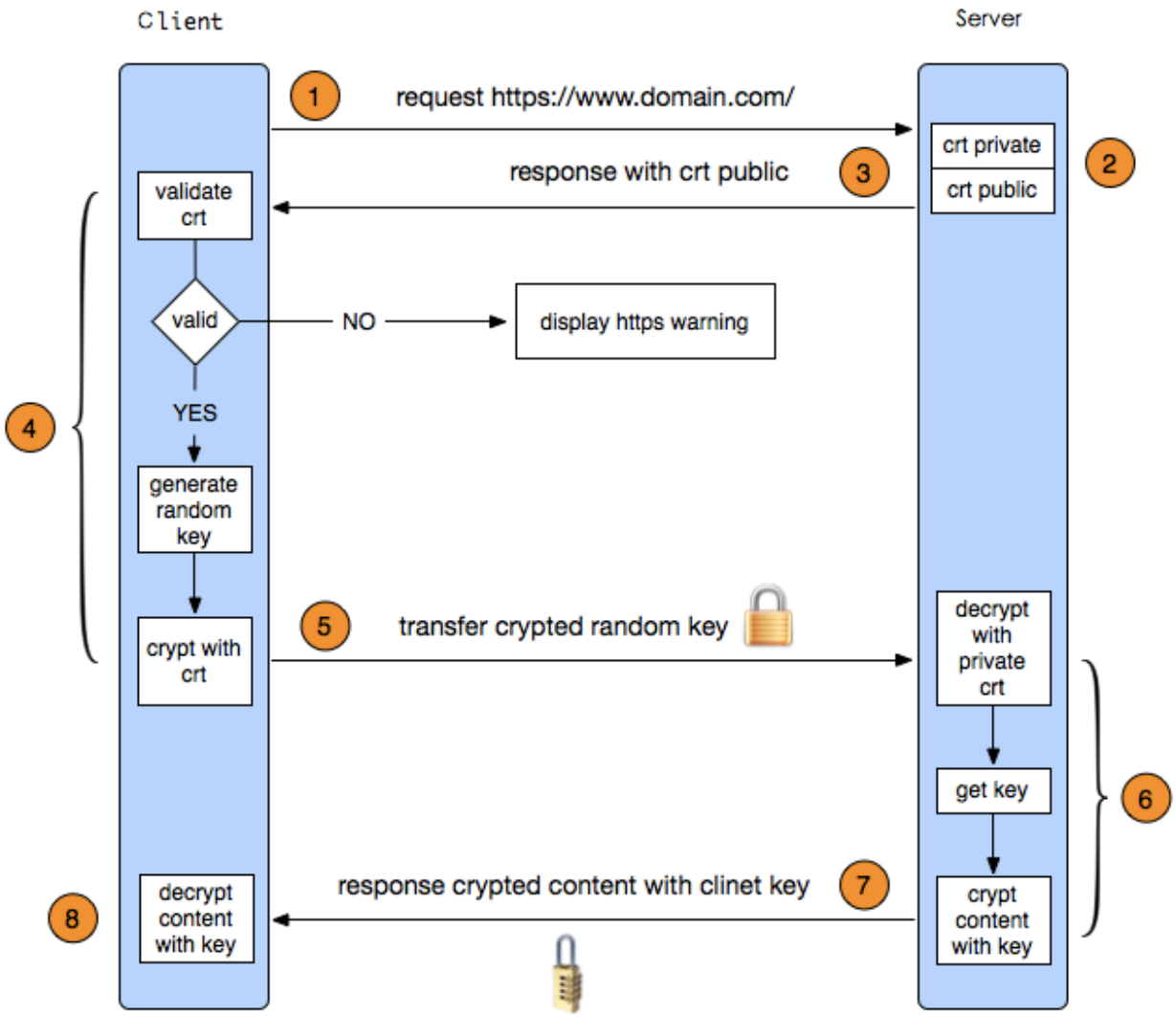
常见的首部：

- **通用首部字段（请求报文与响应报文都会使用的首部字段）**
  - Date：创建报文时间
  - Connection：连接的管理
  - Cache-Control：缓存的控制
  - Transfer-Encoding：报文主体的传输编码方式
- **请求首部字段（请求报文会使用的首部字段）**
  - Host：请求资源所在服务器
  - Accept：可处理的媒体类型
  - Accept-Charset：可接收的字符集
  - Accept-Encoding：可接受的内容编码
  - Accept-Language：可接受的自然语言
- **响应首部字段（响应报文会使用的首部字段）**
  - Accept-Ranges：可接受的字节范围
  - Location：令客户端重新定向到的URI
  - Server：HTTP服务器的安装信息
- **实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）**
  - Allow：资源可支持的HTTP方法
  - Content-Type：实体主类的类型
  - Content-Encoding：实体主体适用的编码方式
  - Content-Language：实体主体的自然语言
  - Content-Length：实体主体的的字节数
  - Content-Range：实体主体的位置范围，一般用于发出部分请求时使用

6. HTTPS工作原理

> HTTPS工作原理

- 一、首先HTTP请求服务端生成证书，客户端对证书的有效期、合法性、域名是否与请求的域名一致、证书的公钥（RSA加密）等进行校验；
- 二、客户端如果校验通过后，就根据证书的公钥的有效， 生成随机数，随机数使用公钥进行加密（RSA加密）；
- 三、消息体产生的后，对它的摘要进行MD5（或者SHA1）算法加密，此时就得到了RSA签名；
- 四、发送给服务端，此时只有服务端（RSA私钥）能解密。
- 五、解密得到的随机数，再用AES加密，作为密钥（此时的密钥只有客户端和服务端知道）。



具体的参考链接：http://blog.csdn.net/sean_cd/article/details/6966130

6. 一次完整的HTTP请求所经历的7个步骤

> 一次完整的HTTP请求所经历的7个步骤

HTTP通信机制是在一次完整的HTTP通信过程中，Web浏览器与Web服务器之间将完成下列7个步骤：

- 建立TCP连接

在HTTP工作开始之前，Web浏览器首先要通过网络与Web服务器建立连接，该连接是通过TCP来完成的，该协议与IP协议共同构建 Internet，即著名的TCP/IP协议族，因此Internet又被称作是TCP/IP网络。**HTTP是比TCP更高层次的应用层协议，根据规则， 只有低层协议建立之后才能，才能进行更层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。**

- Web浏览器向Web服务器发送请求行

一旦建立了TCP连接，**Web浏览器就会向Web服务器发送请求命令**。例如：GET /sample/hello.jsp HTTP/1.1。

- Web浏览器发送请求头
  - 浏览器发送其请求命令之后，还要以头信息的形式向Web服务器发送一些别的信息，**之后浏览器发送了一空白行来通知服务器**，它已经结束了该头信息的发送。

- Web服务器应答
    - 客户机向服务器发出请求后，服务器会客户机回送应答， **HTTP/1.1 200 OK ，应答的第一部分是协议的版本号和应答状态码。**
- Web服务器发送应答头
    - 正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档。
- Web服务器向浏览器发送数据
    - Web服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，**它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据**。
- Web服务器关闭TCP连接
    - 一般情况下，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接，然后如果浏览器或者服务器在其头信息加入了这行代码：

`Connection:keep-alive`

TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽。

建立TCP连接->发送请求行->发送请求头->（到达服务器）发送状态行->发送响应头->发送响应数据->断TCP连接

最具体的HTTP请求过程：http://blog.51cto.com/linux5588/1351007

7. 常见的HTTP相应状态码

> 常见的HTTP相应状态码

- 200：请求被正常处理
- 204：请求被受理但没有资源可以返回
- 206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。
- 301：永久性重定向
- 302：临时重定向
- 303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上
- 304：发送附带条件的请求时，条件不满足时返回，与重定向无关
- 307：临时重定向，与302类似，只是强制要求使用POST方法
- 400：请求报文语法有误，服务器无法识别
- 401：请求需要认证
- 403：请求的对应资源禁止被访问
- 404：服务器无法找到对应资源
- 500：服务器内部错误
- 503：服务器正忙

8. HTTP1.1版本新特性

> HTTP1.1版本新特性

- a、**默认持久连接节省通信量**，只要客户端服务端任意一端没有明确提出断开TCP连接，就一直保持连接，可以发送多次HTTP请求
- b、**管线化，客户端可以同时发出多个HTTP请求，而不用一个个等待响应**
- c、**断点续传**
    - **实际上就是利用HTTP消息头使用分块传输编码，将实体主体分块传输。**

9. HTTP优化方案

我下面就简要概括一下：

- **TCP复用：TCP连接复用是将多个客户端的HTTP请求复用到一个服务器端TCP连接上，而HTTP复用则是一个客户端的多个HTTP请求通过一个TCP连接进行处理。前者是负载均衡设备的独特功能；而后者是HTTP 1.1协议所支持的新功能**
- **内容缓存：将经常用到的内容进行缓存起来，那么客户端就可以直接在内存中获取相应的数据了。**
- **压缩：将文本数据进行压缩，减少带宽**
- **SSL加速（SSL Acceleration）：使用SSL协议对HTTP协议进行加密，在通道内加密并加速**
- **TCP缓冲：通过采用TCP缓冲技术，可以提高服务器端响应时间和处理效率，减少由于通信链路问题给服务器造成的连接负担。**

详情参考：

- http://blog.51cto.com/virtualadc/580832
- http://www.cnblogs.com/cocowool/archive/2011/08/22/2149929.html

如果文档中有任何的不懂的问题，都可以直接来找我询问，我乐意帮助你们！微信搜**Java3y**公众号有我的联系方式。更多**原创**技术文章可关注我的GitHub：https://github.com/ZhongFuCheng3y/3y