（重要）Java面试中常问的计算机网络方面问题

GET 和 POST 的区别

(GET) 请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：/test/demo_form.asp?name1=value1&name2=value2

1. GET 请求可被缓存
2. GET 请求保留在浏览器历史记录中
3. GET 请求可被收藏为书签
4. GET 请求不应在处理敏感数据时使用
5. GET 请求有长度限制
6. GET 请求只应当用于取回数据POST 方法 **(POST)** 请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：POST /test/demo_form.asp HTTP/1.1Host: w3schools.comname1=value1&name2=value2
7. POST 请求不会被缓存
8. POST 请求不会保留在浏览器历史记录中
9. POST 不能被收藏为书签
10. POST 请求对数据长度没有要求

dns使用的协议

既使用TCP又使用UDP

- 首先了解一下TCP与UDP传送字节的长度限制：

1. UDP报文的最大长度为512字节，而TCP则允许报文长度超过512字节。当DNS查询超过512字节时，协议的TC标志出现删除标志，这时则使用TCP发送。通常传统的UDP报文一般不会大于512字节。

- 区域传送时使用TCP，主要有一下两点考虑：

1. 辅域名服务器会定时（一般时3小时）向主域名服务器进行查询以便了解数据是否有变动。如有变动，则会执行一次区域传送，进行数据同步。区域传送将使用TCP而不是UDP，因为数据同步传送的数据量比一个请求和应答的数据量要多得多。
2. TCP是一种可靠的连接，保证了数据的准确性。

- 域名解析时使用UDP协议：

1. 客户端向DNS服务器查询域名，一般返回的内容都不超过512字节，用UDP传输即可。不用经过TCP三次握手，这样DNS服务器负载更低，响应更快。虽然从理论上说，客户端也可以指定向DNS服务器查询的时候使用TCP，但事实上，很多DNS服务器进行配置的时候，仅支持UDP查询包。

幂等

一个幂等操作的特点是其任意多次执行所产生的影响均与一次执行的影响相同。幂等函数，或幂等方法，是指可以使用相同参数重复执行，并能获得相同结果的函数。这些函数不会影响系统状态，也不用担心重复执行会对系统造成改变。例如，“getUsername()和setTrue()”函数就是一个幂等函数.

Cookies和session区别

1. Cookies是一种能够让网站服务器把少量数据储存到客户端的硬盘或内存，或是从客户端的硬盘读取数据的一种技术。Cookies是当你浏览某网站时，由Web服务器置于你硬盘上的一个非常小的文本文件，它可以记录你的用户ID、密码、浏览过的网页、停留的时间等信息。session: 当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。cookie机制：采用的是在客户端保持状态的方案，而session机制采用的是在服务端保持状态的方案。同时我们看到由于服务器端保持状态的方案在客户端也需要保存一个标识，所以session机制可能需要借助cookie机制来达到保存标识的目的。

1. Session是服务器用来跟踪用户的一种手段，每个Session都有一个唯一标识：session ID。当服务器创建了Session时，给客户端发送的响应报文包含了Set-cookie字段，其中有一个名为sid的键值对，这个键值Session ID。客户端收到后就把Cookie保存浏览器，并且之后发送的请求报表都包含SessionID。HTTP就是通过Session和Cookie这两个发送一起合作来实现跟踪用户状态，Session用于服务端，Cookie用于客户端

TCP粘包和拆包产生的原因

1. 应用程序写入数据的字节大小大于套接字发送缓冲区的大小
2. 进行MSS大小的TCP分段。MSS是最大报文段长度的缩写。MSS是TCP报文段中的数据字段的最大长度。数据字段加上TCP首部才等于整个的TCP报文段。所以MSS并不是TCP报文段的最大长度，而是：MSS=TCP报文段长度-TCP首部长度
3. 以太网的payload大于MTU进行IP分片。MTU指：一种通信协议的某一层上面所能通过的最大数据包大小。如果IP层有一个数据包要传，而且数据的长度比链路层的MTU大，那么IP层就会进行分片，把数据包分成托干片，让每一片都不超过MTU。注意，IP分片可以发生在原始发送端主机上，也可以发生在中间路由器上。

TCP粘包和拆包的解决策略

1. 消息定长。例如100字节。
2. 在包尾部增加回车或者空格符等特殊字符进行分割，典型的如FTP协议

3. 将消息分为消息头和消息尾。
4. 其它复杂的协议，如RTMP协议等。

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_SEND状态，等待服务器确认；

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状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据

四次挥手

1. 客户端先发送FIN，进入FIN_WAIT1状态
2. 服务端收到FIN，发送ACK，进入CLOSE_WAIT状态，客户端收到这个ACK，进入FIN_WAIT2状态
3. 服务端发送FIN，进入LAST_ACK状态
4. 客户端收到FIN，发送ACK，进入TIME_WAIT状态，服务端收到ACK，进入CLOSE状态

TIME_WAIT的状态就是主动断开的一方（这里是客户端），发送完最后一次ACK之后进入的状态。并且持续时间还挺长的。客户端TIME_WAIT持续2倍MSL时长，在linux体系中大概是60s，转换成CLOSE状态

TIME_WAIT

TIME_WAIT 是主动关闭链接时形成的，等待2MSL时间，约4分钟。主要是防止最后一个ACK丢失。 由于TIME_WAIT 的时间会非常长，因此server端应尽量减少主动关闭连接

CLOSE_WAIT

CLOSE_WAIT是被动关闭连接是形成的。根据TCP状态机，服务器端收到客户端发送的FIN，则按照TCP实现发送ACK，因此进入CLOSE_WAIT状态。但如果服务器端不执行close()，就不能由CLOSE_WAIT迁移到LAST_ACK，则系统中会存在很多CLOSE_WAIT状态的连接。此时，可能是系统忙于处理读、写操作，而未将已收到FIN的连接，进行close。此时，recv/read已收到FIN的连接socket，会返回0。

为什么需要 TIME_WAIT 状态?

假设最终的ACK丢失，server将重发FIN，client必须维护TCP状态信息以便可以重发最终的ACK，否则会发送RST，结果server认为发生错误。TCP实现必须可靠地终止连接的两个方向(全双工关闭)，client必须进入 TIME_WAIT 状态，因为client可能面 临重发最终ACK的情形。

为什么 TIME_WAIT 状态需要保持 2MSL 这么长的时间?

如果 TIME_WAIT 状态保持时间不足够长(比如小于2MSL)，第一个连接就正常终止了。第二个拥有相同相关五元组的连接出现，而第一个连接的重复报文到达，干扰了第二个连接。TCP实现必须防止某个连接的重复报文在连接终止后出现，所以让TIME_WAIT状态保持时间足够长(2MSL)，连接相应方向上的TCP报文要么完全响应完毕，要么被 丢弃。建立第二个连接的时候，不会混淆。

TIME_WAIT 和CLOSE_WAIT状态socket过多

如果服务器出了异常，百分之八九十都是下面两种情况：

1.服务器保持了大量TIME_WAIT状态

2.服务器保持了大量CLOSE_WAIT状态，简单来说CLOSE_WAIT数目过大是由于被动关闭连接处理不当导致的。

一次完整的HTTP请求过程

域名解析 --> 发起TCP的3次握手 --> 建立TCP连接后发起http请求 --> 服务器响应http请求，浏览器得到html代码 --> 浏览器解析html代码，并请求html代码中的资源（如js、css、图片等） --> 浏览器对页面进行渲染呈现给用户

讲一下长连接

一、基于http协议的长连接

1. 在HTTP1.0和HTTP1.1协议中都有对长连接的支持。其中HTTP1.0需要在request中增加"Connection： keep-alive" header才能够支持，而HTTP1.1默认支持.

- http1.0请求与服务端的交互过程:

1. 客户端发出带有包含一个header："Connection： keep-alive"的请求
2. 服务端接收到这个请求后,根据http1.0和"Connection： keep-alive"判断出这是一个长连接,就会在response的header中也增加"Connection： keep-alive",同是不会关闭已建立的tcp连接.
3. 客户端收到服务端的response后,发现其中包含"Connection： keep-alive"，就认为是一个长连接，不关闭这个连接。并用该连接再发送request.转到a)

二、发心跳包。每隔几秒就发一个数据包过去

TCP如何保证可靠传输?

1. 三次握手。
2. 将数据截断为合理的长度。应用数据被分割成 TCP 认为最适合发送的数据块（按字节编号，合理分片）
3. 超时重发。当 TCP 发出一个段后，它启动一个定时器，如果不能及时收到一个确认就重发

4. 对于收到的请求，给出确认响应
5. 校验出包有错，丢弃报文段，不给出响应
6. 对失序数据进行重新排序，然后才交给应用层
7. 对于重复数据 ， 能够丢弃重复数据
8. 流量控制。TCP 连接的每一方都有固定大小的缓冲空间。TCP 的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。
9. 拥塞控制。当网络拥塞时，减少数据的发送。

详细介绍http

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

特点

1. 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

2. 灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

3. 无连接：无连接的含义是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。采用这种方式可以节省传输时间。

4. 无状态：HTTP协议是无状态协议。无状态是指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，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

5. 支持B/S及C/S模式。

请求消息Request

1. 请求行，用来说明请求类型,要访问的资源以及所使用的HTTP版本.
2. 请求头部，紧接着请求行（即第一行）之后的部分，用来说明服务器要使用的附加信息从第二行起为请求头部，HOST将指出请求的目的地.User-Agent,服务器端和客户端脚本都能访问它,它是浏览器类型检测逻辑的重要基础.该信息由你的浏览器来定义,并且在每个请求中自动发送等等
3. 空行，请求头部后面的空行是必须的
4. 请求数据也叫主体，可以添加任意的其他数据。

响应消息Response

1. 状态行，由HTTP协议版本号， 状态码， 状态消息 三部分组成。
2. 消息报头，用来说明客户端要使用的一些附加信息
3. 空行，消息报头后面的空行是必须的
4. 响应正文，服务器返回给客户端的文本信息。

状态码

- 200 OK //客户端请求成功
- 301 Moved Permanently //永久重定向,使用域名跳转
- 302 Found // 临时重定向,未登陆的用户访问用户中心重定向到登录页面
- 400 Bad Request //客户端请求有语法错误，不能被服务器所理解
- 401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
- 403 Forbidden //服务器收到请求，但是拒绝提供服务
- 404 Not Found //请求资源不存在，eg：输入了错误的URL
- 500 Internal Server Error //服务器发生不可预期的错误
- 503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

http的方法

1. get:客户端向服务端发起请求，获得资源。请求获得URL处所在的资源。
2. post:向服务端提交新的请求字段。请求URL的资源后添加新的数据。
3. head:请求获取URL资源的响应报告，即获得URL资源的头部
4. patch：请求局部修改URL所在资源的数据项
5. put：请求修改URL所在资源的数据元素。
6. delete：请求删除url资源的数据

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

HTTPS和HTTP的区别

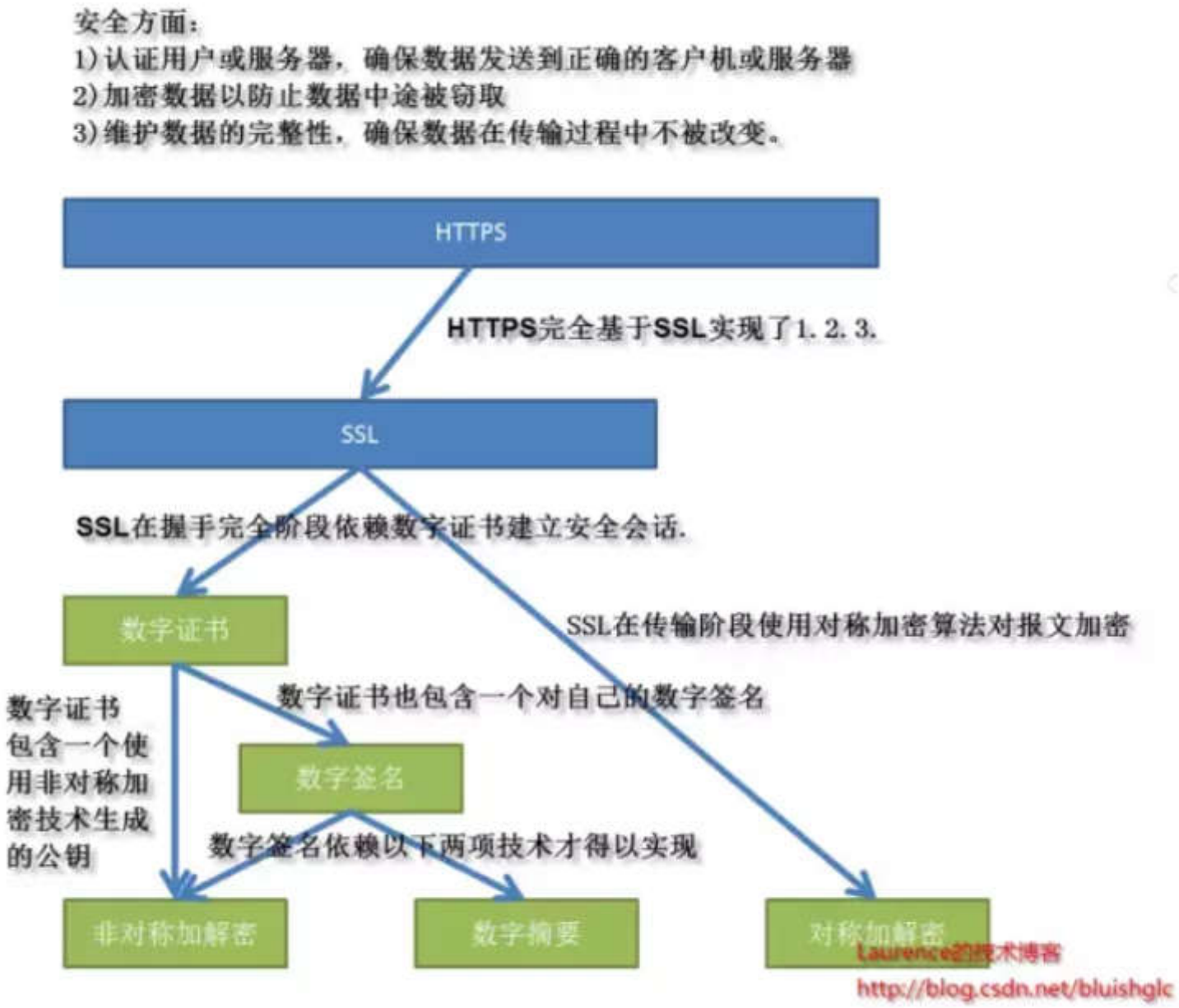
1. https协议需要到CA申请证书，一般免费证书很少，需要交费。
2. http是超文本传输协议，信息是明文传输；https 则是具有安全性的ssl加密传输协 议。
3. http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
4. http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。
5. http默认使用80端口，https默认使用443端口

https是如何保证数据传输的安全

https实际就是在TCP层与http层之间加入了SSL/TLS来为上层的安全保驾护航，主要用到对称加密、非对称加密、证书，等技术进行客户端与服务器的数据加密传输，最终达到保证整个通信的安全性。

- SSL/TLS协议作用：

1. 认证用户和服务器，确保数据发送到正确的客户机和服务器；
2. 加密数据以防止数据中途被窃取；
3. 维护数据的完整性，确保数据在传输过程中不被改变。


安全方面：
1）认证用户或服务器，确保数据发送到正确的客户机或服务器
2）加密数据以防止数据中途被窃取
3）维护数据的完整性，确保数据在传输过程中不被改变。
HTTPS
HTTPS完全基于SSL实现了1. 2. 3.
SSL
SSL在握手完全阶段依赖数字证书建立安全会话.
数字证书
SSL在传输阶段使用对称加密算法对报文加密
数字证书也包含一个对自己的数字签名
数字证书包含一个使用非对称加密技术生成的公钥
数字签名
数字签名依赖以下两项技术才得以实现
非对称加解密
数字摘要
对称加解密
Laurence的技术博客
http://blog.csdn.net/bluishglc