**1、 OSI，TCP/IP，五层协议的体系结构**

OSI（Open System Interconnect）：开放系统互联，是一个七层的计算机网络模型，分别为：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。

TCP/IP（Transmission Control Protocol/Internet Protocol）：传输控制协议/因特网互联协议，是一个四层的计算机网络模型，分别为：网络接口层、网络层、传输层和应用层。

结合OSI和TCP/IP产生了一个五层结构，分别为：物理层、数据链路层、网络层、传输层和应用层。Internet就是采用的TCP/IP协议。

**物理层**：激活、维持、关闭通信端点之间的机械特性、电气特性、功能特性以及过程特性。该层为上层协议提供了一个传输数据的物理媒体。

**数据链路层**：数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。

**网络层**：网络层负责对子网间的数据包进行路由选择。此外，网络层还可以实现拥塞控制、网际互连等功能。

**传输层**：第一个端到端，即主机到主机的层次。传输层负责将上层数据分段并提供端到端的、可靠的或不可靠的传输。此外，传输层还要处理端到端的差错控制和流量控制问题。

**会话层**：会话层管理主机之间的会话进程，即负责建立、管理、终止进程之间的会话。会话层还利用在数据中插入校验点来实现数据的同步。

**表示层**：表示层对上层数据或信息进行变换以保证一个主机应用层信息可以被另一个主机的应用程序理解。表示层的数据转换包括数据的加密、压缩、格式转换等。

**应用层**：为操作系统或网络应用程序提供访问网络服务的接口。

**2、NAT是什么？**

（1）网络地址转换，是一种将私有地址转换为合法IP地址的转换技术，这种技术可以解决现在IP地址不够的问题。

　（2）NAT的实现方式：静态转换；动态转换；端口多路复用（即 内部IP+端口号——外部IP+端口号，这种方式改变外出数据包的源端口并进行端口转换，内部网络的所有主机都可共享一个合法外部IP地址实现对Internet的访问，从而节约IP资源，同时隐藏网络内部的所有主机，有效避免来自Internet的攻击）。

（3）缺点：由于需要将IP包头中的IP地址进行转换，因此不能进行加密操作。

**3、ARP是地址解析协议，简单语言解释一下工作原理。**

（1）首先，每个主机都会在自己的ARP缓冲区中建立一个ARP列表，以表示IP地址和MAC地址之间的对应关系。

（2）当源主机要发送数据时，首先检查ARP列表中是否有对应IP地址的目的主机的MAC地址，如果有，则直接发送数据，如果没有，就向本网段的所有主机发送ARP数据包，该数据包包括的内容有：源主机IP地址，源主机MAC地址，目的主机的IP地址。

（3）当本网络的所有主机收到该ARP数据包时，首先检查数据包中的IP地址是否是自己的IP地址，如果不是，则忽略该数据包，如果是，则首先从数据包中取出源主机的IP和MAC地址写入到ARP列表中，如果已经存在，则覆盖，然后将自己的MAC地址写入ARP响应包中，告诉源主机自己是它想要找的MAC地址。

（4）源主机收到ARP响应包后。将目的主机的IP和MAC地址写入ARP列表，并利用此信息发送数据。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

广播发送ARP请求，单播发送ARP响应。

补充：RARP

RARP逆地址解析协议，作用是完成硬件地址到IP地址的映射，主要用于无盘工作站，因为给无盘工作站配置的IP地址不能保存。工作流程：在网络中配置一台RARP服务器，里面保存着IP地址和MAC地址的映射关系，当无盘工作站启动后，就封装一个RARP数据包，里面有其MAC地址，然后广播到网络上去，当服务器收到请求包后，就查找对应的MAC地址的IP地址装入响应报文中发回给请求者。因为需要广播请求报文，因此RARP只能用于具有广播能力的网络。

**4、DNS（Domain Name System）域名系统，简单描述其工作原理。**

答：当DNS客户机需要在程序中使用名称时，它会查询DNS服务器来解析该名称。客户机发送的每条查询信息包括三条信息：包括：指定的DNS域名，指定的查询类型，DNS域名的指定类别。基于UDP服务，端口53. 该应用一般不直接为用户使用，而是为其他应用服务，如HTTP，SMTP等在其中需要完成主机名到IP地址的转换。

**5、面向连接和非面向连接的服务的特点是什么？**

答：面向连接的服务，通信双方在进行通信之前，要先在双方建立起一个完整的可以彼此沟通的通道，在通信过程中，整个连接的情况一直可以被实时地监控和管理。

非面向连接的服务，不需要预先建立一个联络两个通信节点的连接，需要通信的时候，发送节点就可以往网络上发送信息，让信息自主地在网络上去传，一般在传输的过程中不再加以监控。

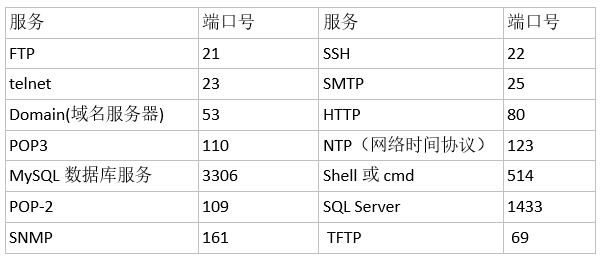
**6、网关的作用？**

答：通过它可以访问外网。

**7、ipconfig的作用是什么？**

答：显示当前TCP/IP配置的信息。

**8、端口及对应的服务？**



**9、ICMP协议？**

答：ICMP是Internet Control Message Protocol，因特网控制报文协议。它是TCP/IP协议族的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由器是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。ICMP报文有两种：差错报告报文和询问报文。

**10、TFTP协议？**

答：Trivial File Transfer Protocol，端口是69。是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。

**11、HTTP协议？**

答：HTTP超文本传输协议，是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。

**12、DHCP协议？**

答：动态主机配置协议，是一种让系统得以连接到网络上，并获取所需要的配置参数手段。

**13、运输层的协议？**

答：TCP，传输单位称为：TCP报文段

UDP，传输单位称为：用户数据报

其端口的作用是识别那个应用程序在使用该协议。

**14、TCP和UDP的区别？**

a、TCP面向连接（如打电话要先拨号建立连接），TCP建立连接要进行3次握手,而断开连接要进行4次挥手; UDP是无连接的，即发送数据之前不需要建立连接

b、TCP提供可靠的服务，是通过顺序编号和确认（ACK）来实现的。通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付。传输途中如果出现了丢包，UDP也不负责重发。甚至当出现包的到达顺序乱掉时也没有纠正的功能。如果需要这些细节控制，可以交给采用UDP的应用程序去处理。

c、TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的，发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给IP层

UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

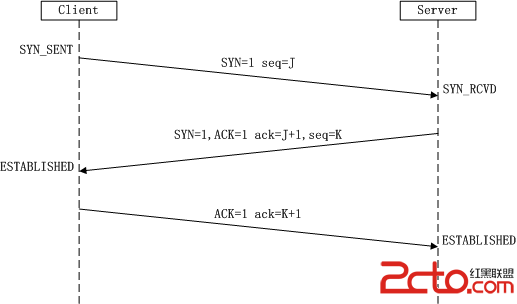
d、每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

e、TCP首部开销最小20字节;UDP的首部开销小，只有8个字节

f、TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

**15、TCP的三次握手过程？为什么会采用三次握手，若采用二次握手可以吗？**

答：建立连接的过程是利用客户服务器模式，假设主机A为客户端，主机B为服务器端。

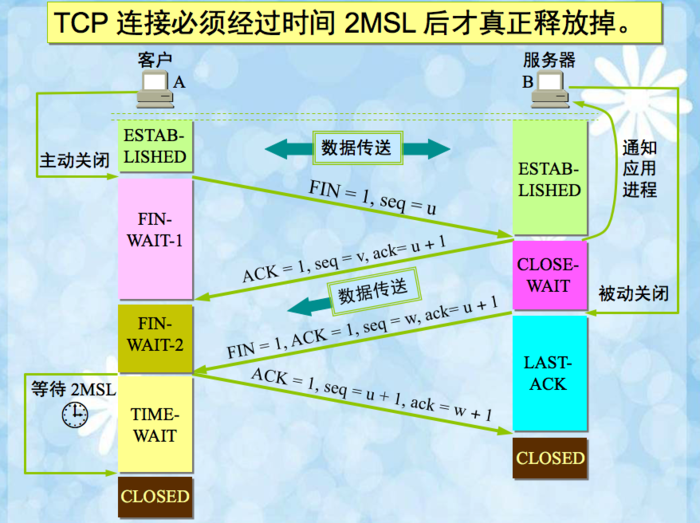


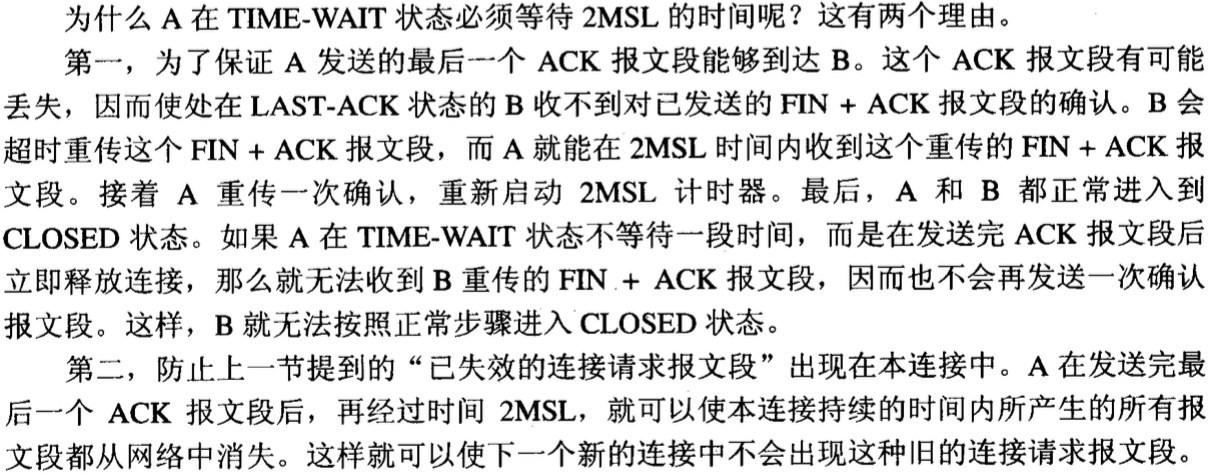
1. TCP的三次握手过程：主机A向B发送连接请求；主机B对收到的主机A的报文段进行确认；主机A再次对主机B的确认进行确认。
2. 采用三次握手是为了防止失效的连接请求报文段突然又传送到主机B，因而产生错误。失效的连接请求报文段是指：主机A发出的连接请求没有收到主机B的确认，于是经过一段时间后，主机A又重新向主机B发送连接请求，且建立成功，顺序完成数据传输。考虑这样一种特殊情况，主机A第一次发送的连接请求并没有丢失，而是因为网络节点导致延迟达到主机B，主机B以为是主机A又发起的新连接，于是主机B同意连接，并向主机A发回确认，但是此时主机A根本不会理会，主机B就一直在等待主机A发送数据，导致主机B的资源浪费。

(3)为什么连接的时候是三次握手，关闭的时候却是四次握手？

因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

（4）为什么TIME\_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？





注意： TIME\_WAIT状态的时间可能为30秒、2分钟不等，Linux是60秒。在内核源码/usr/src/linux/include/net/tcp.h中有 #define TCP\_TIMEWAIT\_LEN (60\*HZ)。

在实际socket编程中，服务程序停止后想立即重启，而新套接字依旧使用同一端口可以通过设置 SO\_REUSEADDR选项达到不必等待2MSL时间结束再使用此端口，当然也可以修改内核的代码，重新编译内核，从而修改TIME\_WAIT的时间。

在Linux下使用统计命令：

netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

发现存在大量TIME\_WAIT状态的连接，造成资源浪费

一个灵活的思路：客户端向服务端发起HTTP请求，服务端响应后主动关闭连接，于是TIME\_WAIT便留在了服务端。这里的关键在于主动关闭连接的是服务端！在关闭TCP连接的时候，先发起的一方注定逃不开TIME\_WAIT的宿命。如果客户端可控的话，那么在服务端打开KeepAlive，尽可能不让服务端主动关闭连接，而让客户端主动关闭连接，如此一来问题便迎刃而解了。

**16、详细解释一下IP协议的定义，在哪个层上面，主要有什么作用？TCP和UDP呢？**

答：IP协议是网络层的协议，它是为了实现相互连接的计算机进行通信设计的协议，它实现了自动路由功能，即自动寻径功能。TCP是传输层的协议，它向下屏蔽IP协议的不可靠传输的特性，向上提供一种面向连接的、可靠的点到点数据传输。TCP在可靠性和安全性上等更有保证。UDP也是传输层协议，它提供的是一种非面向连接的，不可靠的数据传输，这主要是有些应用需要更快速的数据传输，比如局域网内的大多数文件传输都是基于UDP的。UDP在传输速率上更快，开销更小。

**17、运输层协议与网络层协议的区别？**

网络层协议负责的是提供主机间的逻辑通信 运输层协议负责的是提供进程间的逻辑通信

**18、在浏览器中输入www.baidu.com后执行的全部过程**

　　现在假设如果我们在客户端（客户端）浏览器中输入http://www.baidu.com,而baidu.com为要访问的服务器（服务器），下面详细分析客户端为了访问服务器而执行的一系列关于协议的操作：

　　1、客户端浏览器通过DNS解析到www.baidu.com的IP地址220.181.27.48，通过这个IP地址找到客户端到服务器的路径。客户端浏览器发起一个HTTP会话到220.161.27.48，然后通过TCP进行封装数据包，输入到网络层。

　　2、在客户端的传输层，把HTTP会话请求分成报文段，添加源和目的端口，如服务器使用80端口监听客户端的请求，客户端由系统随机选择一个端口如5000，与服务器进行交换，服务器把相应的响应返回给客户端的5000端口。然后使用IP层的IP地址查找目的端。

3、客户端的网络层不用关系应用层或者传输层的东西，主要做的是通过查找路由表确定如何到达服务器，期间可能经过多个路由器，这些都是由路由器来完成的工作，我不作过多的描述，无非就是通过查找路由表决定通过那个路径到达服务器。

　　4、客户端的链路层，包通过链路层发送到路由器，通过邻居协议查找给定IP地址的MAC地址，然后发送ARP请求查找目的地址，如果得到回应后就可以使用ARP的请求应答交换的IP数据包现在就可以传输了，然后发送IP数据包到达服务器的地址。

**19、传输层是UDP的应用层协议有哪些？**

（1） DNS：用于域名解析服务，将域名地址转换为IP地址。DNS用的是53号端口。

（2） SNMP：简单网络管理协议，使用161号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。

（3） TFTP(Trival File Transfer Protocal)，简单文件传输协议，该协议在熟知端口69上使用UDP服务。

**20、SESSION与COOKIE的区别?**

1）COOKIE保存在客户端，而SESSION则保存服务器端。

2）从安全性的角度来讲，SESSION的安全性要高。

3）从保存内容的类型的角度来讲，COOKIE只保存ASCII字符串（及能够自动转换成字符串），而session则可以保存所有的数据类型。

4）从保存内容的大小的角度来讲，COOKIE保存的内容是有限制的，而SESSION基本上没有这个限制。

5）从存储位置的角度来讲，cookie保存在内存或者硬盘里，session可以保存本地文件里，也可以保存在redis，MySQL里面，session\_set\_save\_handler() 可以设置存储句柄，可以把session统一保存在redis服务器里面

6）从性能的角度来讲，用SESSION的话，对服务器的压力会更大一些。

7）Cookie支持跨域名访问，例如将domain属性设置为“.biaodianfu.com”，则以“.biaodianfu.com”为后缀的一切域名均能够访问该Cookie。

**21、常见的状态码？**

200 - 请求成功

301 - 资源（网页等）被永久转移到其它URL，今后任何新的请求都应使用新的URI代替

302 - 临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

304 - 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源

403 - 服务器理解请求客户端的请求，但是拒绝执行此请求

404 - 请求的资源（网页等）不存在

500 - 内部服务器错误

503 - 由于超载或系统维护，服务器暂时的无法处理客户端的请求

**22、GET和POST的区别**

GET和POST本质上就是TCP连接，你要给GET加上request body，给POST带上url参数，技术上是完全行的通的。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

1. GET在浏览器回退时是无害的，而POST会再次提交请求。

2. GET产生的URL地址可以被Bookmark，而POST不可以。

3. GET请求会被浏览器主动cache，而POST不会，除非手动设置。

4. GET请求只能进行url编码，而POST支持多种编码方式。

5. GET请求参数会被保留在浏览器历史记录里，而POST中的参数不会被保留。

6. 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。

7. 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。

8. GET请求在URL中传送的参数是有长度限制的，而POST没有。 GET参数通过URL传递，POST放在Request body中。(大多数)浏览器通常都会限制url长度在2K个字节，而(大多数)服务器最多处理64K大小的url。

GET和POST还有一个重大区别，简单的说：

**GET产生一个TCP数据包;POST产生两个TCP数据包**。

长的说：

对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200(返回数据);

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok(返回数据)。

也就是说，GET只需要汽车跑一趟就把货送到了，而POST得跑两趟，第一趟，先去和服务器打个招呼“嗨，我等下要送一批货来，你们打开门迎接我”，然后再回头把货送过去。

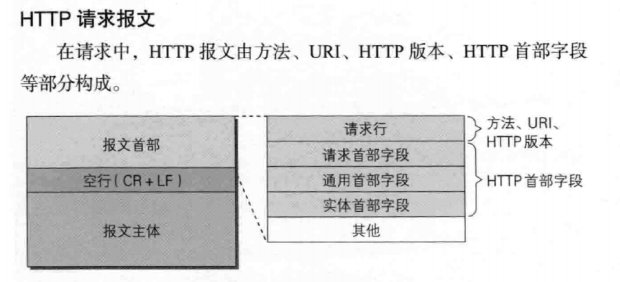
因为POST需要两步，时间上消耗的要多一点，看起来GET比POST更有效。因此Yahoo团队有推荐用GET替换POST来优化网站性能。但这是一个坑!跳入需谨慎。为什么?

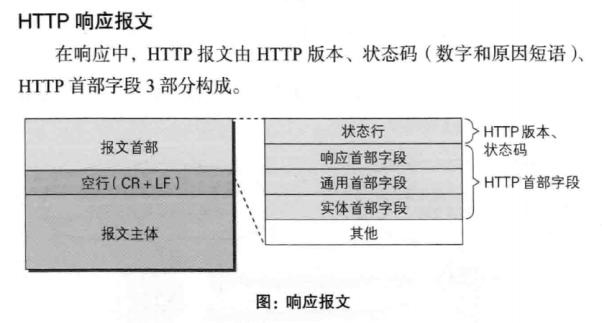
1. GET与POST都有自己的语义，不能随便混用。

2. 据研究，在网络环境好的情况下，发一次包的时间和发两次包的时间差别基本可以无视。而在网络环境差的情况下，两次包的TCP在验证数据包完整性上，有非常大的优点。

3. 并不是所有浏览器都会在POST中发送两次包，Firefox就只发送一次。

**23.HTTP请求报文和相应报文的格式？**





HTTP的头域包括通用头、请求头、响应头和实体头四个部分