[**计算机网络面试问题集锦**](https://blog.csdn.net/justloveyou_/article/details/78303617)

###### **1.Http和Https的区别**

　Http协议运行在TCP之上，明文传输，客户端与服务器端都无法验证对方的身份；Https是身披SSL(Secure Socket Layer)外壳的Http，运行于SSL上，SSL运行于TCP之上，是添加了加密和认证机制的HTTP。二者之间存在如下不同：

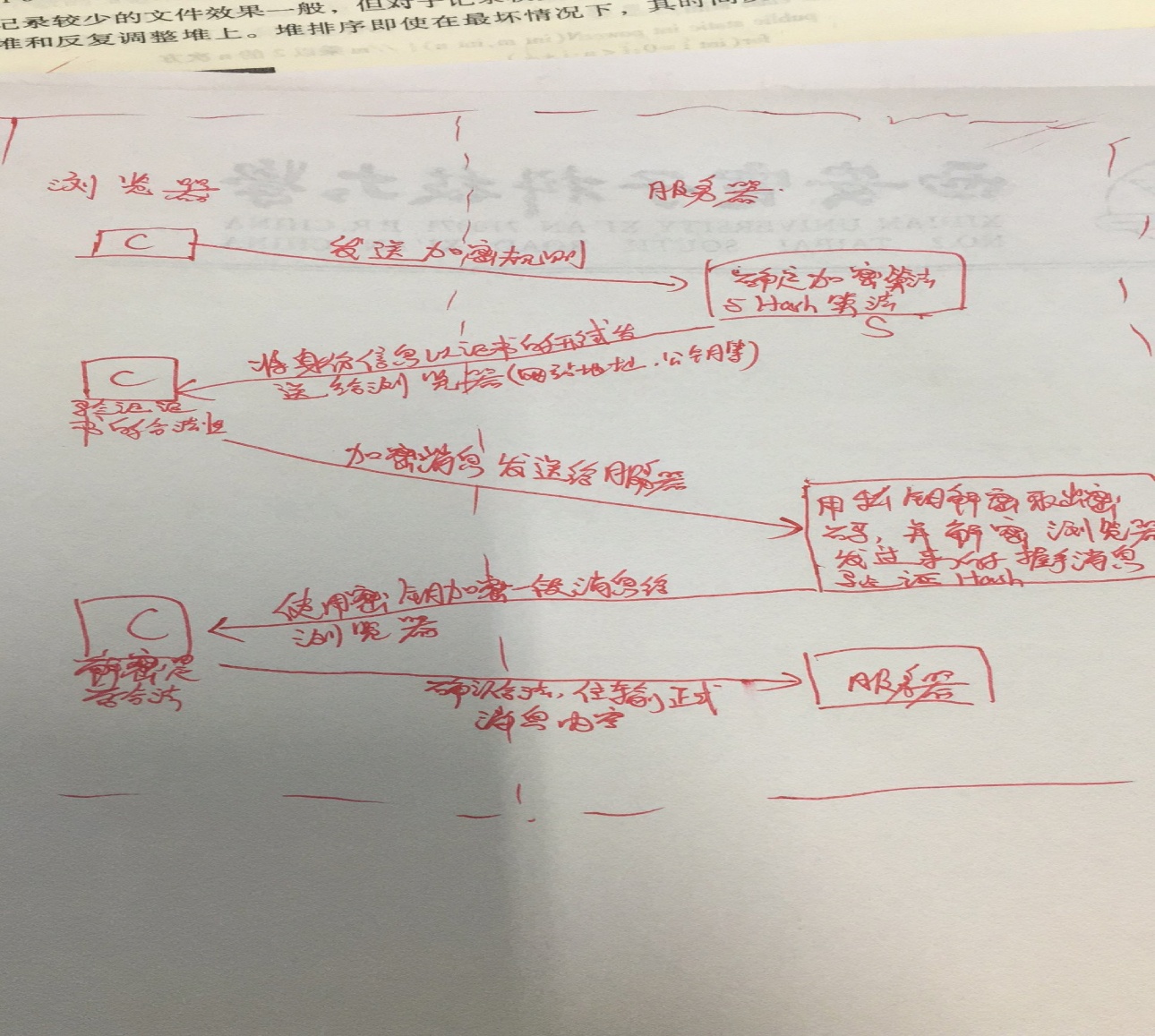
* 端口不同：Http与Https使用不同的连接方式，用的端口也不一样，前者是80，后者是443；
* 资源消耗：和HTTP通信相比，Https通信会由于加减密处理消耗更多的CPU和内存资源；
* 开销：Https通信需要证书，而证书一般需要向认证机构购买； Https的加密机制是一种共享密钥加密和公开密钥加密并用的混合加密机制。

###### **2.HTTPs协议的握手过程**

第一步: 浏览器将自己支持的一套加密规则发送给网站。

第二步: 网站从中选出一组加密算法与HASH算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥，以及证书的颁发机构等信息.

第三步: 获得网站证书之后浏览器要做以下工作：



a) 验证证书的合法性（颁发证书的机构是否合法，证书中包含的网站地址是否与正在访问的地址一致等），如果证书受信任，则浏览器栏里面会显示一个小锁头，否则会给出证书不受信的提示。

b) 如果证书受信任，或者是用户接受了不受信的证书，浏览器会生成一串随机数的密码，并用证书中提供的公钥加密。

c) 使用约定好的HASH计算握手消息，并使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给网站。

第四步:网站接收浏览器发来的数据之后要做以下的操作:

a) 使用自己的私钥将信息解密取出密码，使用密码解密浏览器发来的握手消息，并验证HASH是否与浏览器发来的一致。

b) 使用密码加密一段握手消息，发送给浏览器。

第五步: 浏览器解密并计算握手消息的HASH，如果与服务端发来的HASH一致，此时握手过程结束，之后所有的通信数据将由之前浏览器生成的随机密码并利用对称加密算法进行加密。

这里浏览器与网站互相发送加密的握手消息并验证，目的是为了保证双方都获得了一致的密码，并且可以正常的加密解密数据，为后续真正数据的传输做一次测试。

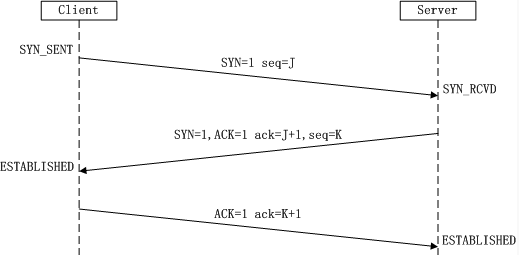
###### **3.对称加密与非对称加密**

对称密钥加密是指加密和解密使用同一个密钥的方式，这种方式存在的最大问题就是密钥发送问题，即如何安全地将密钥发给对方；而非对称加密是指使用一对非对称密钥，即公钥和私钥，公钥可以随意发布，但私钥只有自己知道。发送密文的一方使用对方的公钥进行加密处理，对方接收到加密信息后，使用自己的私钥进行解密。由于非对称加密的方式不需要发送用来解密的私钥，所以可以保证安全性；但是和对称加密比起来，它非常的慢，所以我们还是要用对称加密来传送消息，但对称加密所使用的密钥我们可以通过非对称加密的方式发送出去。

###### **4、三次握手与四次挥手**

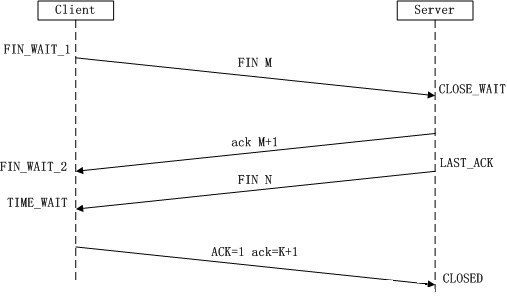
(1). 三次握手(我要和你建立链接，你真的要和我建立链接么，我真的要和你建立链接，成功)：

* 第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。
* 第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。
* 第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。



(2). 四次挥手(我要和你断开链接；好的，断吧。我也要和你断开链接；好的，断吧)：

* 第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。
* 第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态。此时TCP链接处于半关闭状态，即客户端已经没有要发送的数据了，但服务端若发送数据，则客户端仍要接收。
* 第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。
* 第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。



###### **5、为什么TCP链接需要三次握手，两次不可以么，为什么？**

为了防止 已失效的链接请求报文突然又传送到了服务端，因而产生错误。客户端发出的连接请求报文并未丢失，而是在某个网络节点长时间滞留了，以致延误到链接释放以后的某个时间才到达Server。这是，Server误以为这是Client发出的一个新的链接请求，于是就向客户端发送确认数据包，同意建立链接。若不采用“三次握手”，那么只要Server发出确认数据包，新的链接就建立了。由于client此时并未发出建立链接的请求，所以其不会理睬Server的确认，也不与Server通信；而这时Server一直在等待Client的请求，这样Server就白白浪费了一定的资源。若采用“三次握手”，在这种情况下，由于Server端没有收到来自客户端的确认，则就会知道Client并没有要求建立请求，就不会建立链接。

###### **6、TCP协议如何来保证传输的可靠性**

TCP提供一种面向连接的、可靠的字节流服务。其中，面向连接意味着两个使用TCP的应用（通常是一个客户和一个服务器）在彼此交换数据之前必须先建立一个TCP连接。在一个TCP连接中，仅有两方进行彼此通信；而字节流服务意味着两个应用程序通过TCP链接交换8bit字节构成的字节流，TCP不在字节流中插入记录标识符。

对于可靠性，TCP通过以下方式进行保证：

* 数据包校验：目的是检测数据在传输过程中的任何变化，若校验出包有错，则丢弃报文段并且不给出响应，这时TCP发送数据端超时后会重发数据；
* 对失序数据包重排序：既然TCP报文段作为IP数据报来传输，而IP数据报的到达可能会失序，因此TCP报文段的到达也可能会失序。TCP将对失序数据进行重新排序，然后才交给应用层；
* 丢弃重复数据：对于重复数据，能够丢弃重复数据；
* 应答机制：当TCP收到发自TCP连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒；
* 超时重发：当TCP发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段；
* 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，这可以防止较快主机致使较慢主机的缓冲区溢出，这就是流量控制。TCP使用的流量控制协议是可变大小的滑动窗口协议。

###### **7、客户端不断进行请求链接会怎样？DDos(Distributed Denial of Service)攻击？**

　服务器端会为每个请求创建一个链接，并向其发送确认报文，然后等待客户端进行确认，利用TCP的缺陷，通过向网络服务所在端口发送大量的伪造原地址的攻击保温，造成目标服务器的半开链接队列被占满，从而阻止其他合法用户进行访问。

1)、DDos 攻击

* 客户端向服务端发送请求链接数据包
* 服务端向客户端发送确认数据包
* 客户端不向服务端发送确认数据包，服务器一直等待来自客户端的确认

2)、DDos 预防 ( 没有彻底根治的办法，除非不使用TCP )

* 限制同时打开SYN半链接的数目
* 缩短SYN半链接的Time out 时间
* 关闭不必要的服务

###### **8、Get与POST的区别**

REST（表现层状态转移）：在网络中client与server的一种交互形式；通俗来讲就是URL定位资源，用HTTP动词（GET，POST，DELETE，DETC）描述操作。

GET与POST是我们常用的两种HTTP Method，二者之间的区别主要包括如下五个方面：

(1). 从功能上讲，GET一般用来从服务器上获取资源，POST一般用来更新服务器上的资源；

(2). 从REST服务角度上说，GET是幂等的，即读取同一个资源，总是得到相同的数据，而POST不是幂等的，因为每次请求对资源的改变并不是相同的；进一步地，GET不会改变服务器上的资源，而POST会对服务器资源进行改变；

(3). 从请求参数形式上看，GET请求的数据会附在URL之后，即将请求数据放置在HTTP报文的 请求头 中，以?分割URL和传输数据，参数之间以&相连。特别地，如果数据是英文字母/数字，原样发送；否则，会将其编码为 application/x-www-form-urlencoded MIME 字符串(如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如：%E4%BD%A0%E5%A5%BD，其中％XX中的XX为该符号以16进制表示的ASCII)；而POST请求会把提交的数据则放置在是HTTP请求报文的 请求体 中。

(4). 就安全性而言，POST的安全性要比GET的安全性高，因为GET请求提交的数据将明文出现在URL上，而且POST请求参数则被包装到请求体中，相对更安全。

(5). 从请求的大小看，GET请求的长度受限于浏览器或服务器对URL长度的限制，允许发送的数据量比较小，而POST请求则是没有大小限制的。

GET请求中URL编码的意义

　　我们知道，在GET请求中会对URL中非西文字符进行编码，这样做的目的就是为了 避免歧义。看下面的例子，

　　针对“name1=value1&name2=value2”的例子，我们来谈一下数据从客户端到服务端的解析过程。首先，上述字符串在计算机中用ASCII吗表示为：

6E616D6531 3D 76616C756531 26 6E616D6532 3D 76616C756532

6E616D6531：name1

3D：=

76616C756531：value1

26：&

6E616D6532：name2

3D：=

76616C756532：value2

　　服务端在接收到该数据后就可以遍历该字节流，一个字节一个字节的吃，当吃到3D这字节后，服务端就知道前面吃得字节表示一个key，再往后吃，如果遇到26，说明从刚才吃的3D到26子节之间的是上一个key的value，以此类推就可以解析出客户端传过来的参数。

　　现在考虑这样一个问题，如果我们的参数值中就包含=或&这种特殊字符的时候该怎么办？比如，“name1=value1”，其中value1的值是“va&lu=e1”字符串，那么实际在传输过程中就会变成这样“name1=va&lu=e1”。这样，我们的本意是只有一个键值对，但是服务端却会解析成两个键值对，这样就产生了歧义。

　　那么，如何解决上述问题带来的歧义呢？解决的办法就是对参数进行URL编码：例如，我们对上述会产生歧义的字符进行URL编码后结果：“name1=va%26lu%3D”，这样服务端会把紧跟在“%”后的字节当成普通的字节，就是不会把它当成各个参数或键值对的分隔符。

###### **9、TCP与UDP的区别**

　 TCP (Transmission Control Protocol)和UDP(User Datagram Protocol)协议属于传输层协议，它们之间的区别包括：

* TCP是面向连接的，UDP是无连接的；
* TCP是可靠的，UDP是不可靠的；
* TCP只支持点对点通信，UDP支持一对一、一对多、多对一、多对多的通信模式；
* TCP是面向字节流的，UDP是面向报文的；
* TCP有拥塞控制机制;UDP没有拥塞控制，适合媒体通信；
* TCP首部开销(20个字节)比UDP的首部开销(8个字节)要大；

###### **10、UDP如何实现可靠传输**

UDP不属于连接协议，具有资源消耗少，处理速度快的优点，所以通常音频，视频和普通数据在传送时，使用UDP较多，因为即使丢失少量的包，也不会对接受结果产生较大的影响。

传输层无法保证数据的可靠传输，只能通过应用层来实现了。实现的方式可以参照tcp可靠性传输的方式，只是实现不在传输层，实现转移到了应用层。

最简单的方式是在应用层模仿传输层TCP的可靠性传输。下面不考虑拥塞处理，可靠UDP的简单设计。

* 1）、添加seq/ack机制，确保数据发送到对端
* 2）、添加发送和接收缓冲区，主要是用户超时重传。
* 3）、添加超时重传机制。

详细说明：送端发送数据时，生成一个随机seq=x，然后每一片按照数据大小分配seq。数据到达接收端后接收端放入缓存，并发送一个ack=x的包，表示对方已经收到了数据。发送端收到了ack包后，删除缓冲区对应的数据。时间到后，定时任务检查是否需要重传数据。

**RUDP（Reliable User Datagram Protocol）：RUDP 提供一组数据服务质量增强机制，如拥塞控制的改进、重发机制及淡化服务器算法等**，从而在包丢失和网络拥塞的情况下， RTP 客户机（实时位置）面前呈现的就是一个高质量的 RTP 流。在不干扰协议的实时特性的同时，可靠 UDP 的拥塞控制机制允许 TCP 方式下的流控制行为。

**RTP（Real Time Protocol）：***RTP为数据提供了具有实时特征的端对端传送服务*，如在组播或单播网络服务下的交互式视频音频或模拟数据。应用程序通常在 UDP 上运行 RTP 以便使用其多路结点和校验服务；这两种协议都提供了传输层协议的功能。但是 RTP 可以与其它适合的底层网络或传输协议一起使用。如果底层网络提供组播方式，那么 RTP 可以使用该组播表传输数据到多个目的地。RTP 本身并没有提供按时发送机制或其它服务质量（QoS）保证，它依赖于底层服务去实现这一过程。 RTP 并不保证传送或防止无序传送，也不确定底层网络的可靠性。 RTP 实行有序传送， RTP 中的序列号允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如：在视频解码中，就不需要顺序解码。

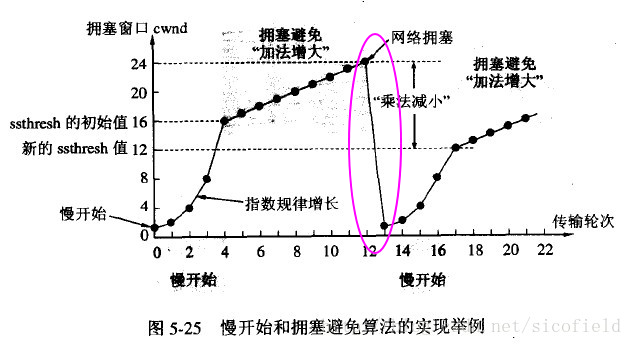
**UDT（UDP-based Data Transfer Protocol）：**基于UDP的数据传输协议（UDP-basedData Transfer Protocol，简称UDT）是一种互联网数据传输协议。UDT的主要目的是支持高速广域网上的海量数据传输，而互联网上的标准数据传输协议TCP在高带宽长距离网络上性能很差。顾名思义，UDT建于UDP之上，并引入新的拥塞控制和数据可靠性控制机制。UDT是面向连接的双向的应用层协议。它同时支持可靠的数据流传输和部分可靠的数据报传输。由于UDT完全在UDP上实现，它也可以应用在除了高速数据传输之外的其它应用领域，例如点到点技术（P2P），防火墙穿透，多媒体数据传输等等。

###### **11、TCP的拥塞处理**

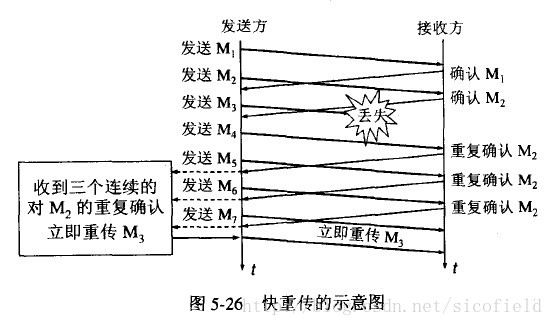
计算机网络中的带宽、交换结点中的缓存及处理机等都是网络的资源。在某段时间，若对网络中某一资源的需求超过了该资源所能提供的可用部分，网络的性能就会变坏，这种情况就叫做拥塞。拥塞控制就是 防止过多的数据注入网络中，这样可以使网络中的路由器或链路不致过载。注意，拥塞控制和流量控制不同，前者是一个全局性的过程，而后者指点对点通信量的控制。拥塞控制的方法主要有以下四种：

1). 慢启动：不要一开始就发送大量的数据，先探测一下网络的拥塞程度，也就是说由小到大逐渐增加拥塞窗口的大小;

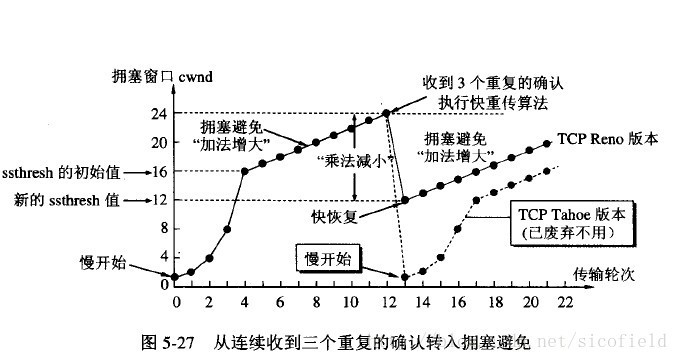
2). 拥塞避免：拥塞避免算法让拥塞窗口缓慢增长，即每经过一个往返时间RTT就把发送方的拥塞窗口cwnd加1，而不是加倍，这样拥塞窗口按线性规律缓慢增长。



3). 快重传：快重传要求接收方在收到一个 失序的报文段 后就立即发出 重复确认（为的是使发送方及早知道有报文段没有到达对方）而不要等到自己发送数据时捎带确认。快重传算法规定，发送方只要一连收到三个重复确认就应当立即重传对方尚未收到的报文段，而不必继续等待设置的重传计时器时间到期。



4). 快恢复：快重传配合使用的还有快恢复算法，当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把ssthresh门限减半，但是接下去并不执行慢开始算法：因为如果网络出现拥塞的话就不会收到好几个重复的确认，所以发送方现在认为网络可能没有出现拥塞。所以此时不执行慢开始算法，而是将cwnd设置为ssthresh的大小，然后执行拥塞避免算法。



###### **12、从输入网址到获得页面的过程**

(1). 浏览器查询 DNS，获取域名对应的IP地址:具体过程包括浏览器搜索自身的DNS缓存、搜索操作系统的DNS缓存、读取本地的Host文件和向本地DNS服务器进行查询等。对于向本地DNS服务器进行查询，如果要查询的域名包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析(此解析具有权威性)；如果要查询的域名不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析（此解析不具有权威性）。如果本地域名服务器并未缓存该网址映射关系，那么将根据其设置发起递归查询或者迭代查询；DNS服务器有两种，一种是主DNS服务器，一个是辅助DNS服务器，在一个区中的主DNS服务器从自己的本级的数据文件中读取DNS数据信息，而辅助DNS服务器则从区的主DNS服务器中读取DNS数据信息，辅助DNS服务器一般会定时向主域名服务器进行查询以便了解数据是否有发生变动，如果有变动则会进行一次区域传送（辅助DNS服务器与主DNS服务器通信，加载数据信息），进行数据同步，而区域传送使用的是TCP协议，保证数据的准确性；而客户端向DNS服务器查询域名时，一般返回的内容都不超过512字节，用UDP传输即可。

　　(2). 浏览器获得域名对应的IP地址以后，浏览器向服务器请求建立链接，发起三次握手；

　　(3). TCP/IP链接建立起来后，浏览器向服务器发送HTTP请求；

　　(4). 服务器接收到这个请求，并根据路径参数映射到特定的请求处理器进行处理，并将处理结果及相应的视图返回给浏览器；

　　(5). 浏览器解析并渲染视图，若遇到对js文件、css文件及图片等静态资源的引用，则重复上述步骤并向服务器请求这些资源；

　　(6). 浏览器根据其请求到的资源、数据渲染页面，最终向用户呈现一个完整的页面。

###### **13、Session、Cookie 与 Application**

　　Cookie和Session都是客户端与服务器之间保持状态的解决方案，具体来说，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器端保持状态的方案。

(1). Cookie及其相关API

　　Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie，而客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务器，服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。

(2). Session及其相关API

　　同样地，会话状态也可以保存在服务器端。客户端请求服务器，如果服务器记录该用户状态，就获取Session来保存状态，这时，如果服务器已经为此客户端创建过session，服务器就按照sessionid把这个session检索出来使用；如果客户端请求不包含sessionid，则为此客户端创建一个session并且生成一个与此session相关联的sessionid，并将这个sessionid在本次响应中返回给客户端保存。保存这个sessionid的方式可以采用 cookie机制 ，这样在交互过程中浏览器可以自动的按照规则把这个标识发挥给服务器；若浏览器禁用Cookie的话，可以通过 URL重写机制 将sessionid传回服务器。

(3). Session 与 Cookie 的对比

* 实现机制：Session的实现常常依赖于Cookie机制，通过Cookie机制回传SessionID；
* 大小限制：Cookie有大小限制并且浏览器对每个站点也有cookie的个数限制，Session没有大小限制，理论上只与服务器的内存大小有关；
* 安全性：Cookie存在安全隐患，通过拦截或本地文件找得到cookie后可以进行攻击，而Session由于保存在服务器端，相对更加安全；
* 服务器资源消耗：Session是保存在服务器端上会存在一段时间才会消失，如果session过多会增加服务器的压力。

(4). Application：Application（Java Web中的ServletContext）与一个Web应用程序相对应，为应用程序提供了一个全局的状态，所有客户都可以使用该状态。

###### **14、SQL 注入**

　　SQL注入就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

1). SQL注入攻击的总体思路

　　(1). 寻找到SQL注入的位置   
　　(2). 判断服务器类型和后台数据库类型   
　　(3). 针对不通的服务器和数据库特点进行SQL注入攻击

2). SQL注入攻击实例

　　比如，在一个登录界面，要求输入用户名和密码，可以这样输入实现免帐号登录：

用户名： ‘or 1 = 1 --

密 码：

　　用户一旦点击登录，如若没有做特殊处理，那么这个非法用户就很得意的登陆进去了。这是为什么呢?下面我们分析一下：从理论上说，后台认证程序中会有如下的SQL语句：String sql = “select \* from user\_table where username=’ “+userName+” ’ and password=’ “+password+” ‘”; 因此，当输入了上面的用户名和密码，上面的SQL语句变成：SELECT \* FROM user\_table WHERE username=’’or 1 = 1 – and password=’’。分析上述SQL语句我们知道，   
username=‘ or 1=1 这个语句一定会成功；然后后面加两个-，这意味着注释，它将后面的语句注释，让他们不起作用。这样，上述语句永远都能正确执行，用户轻易骗过系统，获取合法身份。

3). 应对方法

(1). 参数绑定

　　使用预编译手段，绑定参数是最好的防SQL注入的方法。目前许多的ORM框架及JDBC等都实现了SQL预编译和参数绑定功能，攻击者的恶意SQL会被当做SQL的参数而不是SQL命令被执行。在mybatis的mapper文件中，对于传递的参数我们一般是使用#和$来获取参数值。当使用#时，变量是占位符，就是一般我们使用javajdbc的PrepareStatement时的占位符，所有可以防止sql注入；当使用$时，变量就是直接追加在sql中，一般会有sql注入问题。

(2). 使用正则表达式过滤传入的参数

###### **15、 XSS 攻击**

XSS是一种经常出现在web应用中的计算机安全漏洞，与SQL注入一起成为web中最主流的攻击方式。XSS是指恶意攻击者利用网站没有对用户提交数据进行转义处理或者过滤不足的缺点，进而添加一些脚本代码嵌入到web页面中去，使别的用户访问都会执行相应的嵌入代码，从而盗取用户资料、利用用户身份进行某种动作或者对访问者进行病毒侵害的一种攻击方式。

1). XSS攻击的危害

* 盗取各类用户帐号，如机器登录帐号、用户网银帐号、各类管理员帐号
* 控制企业数据，包括读取、篡改、添加、删除企业敏感数据的能力
* 盗窃企业重要的具有商业价值的资料
* 非法转账
* 强制发送电子邮件
* 网站挂马
* 控制受害者机器向其它网站发起攻击

2). 原因解析

　　主要原因：过于信任客户端提交的数据！

　　解决办法：不信任任何客户端提交的数据，只要是客户端提交的数据就应该先进行相应的过滤处理然后方可进行下一步的操作。

　　进一步分析细节：客户端提交的数据本来就是应用所需要的，但是恶意攻击者利用网站对客户端提交数据的信任，在数据中插入一些符号以及javascript代码，那么这些数据将会成为应用代码中的一部分了，那么攻击者就可以肆无忌惮地展开攻击啦，因此我们绝不可以信任任何客户端提交的数据！！！

3). XSS 攻击分类

(1). 反射性XSS攻击 (非持久性XSS攻击)

　　漏洞产生的原因是攻击者注入的数据反映在响应中。一个典型的非持久性XSS攻击包含一个带XSS攻击向量的链接(即每次攻击需要用户的点击)，例如，正常发送消息：

http://www.test.com/message.php?send=Hello,World！

接收者将会接收信息并显示Hello,World；但是，非正常发送消息：

http://www.test.com/message.php?send=<script>alert(‘foolish!’)</script>！

接收者接收消息显示的时候将会弹出警告窗口！

(2). 持久性XSS攻击 (留言板场景)

　　XSS攻击向量(一般指XSS攻击代码)存储在网站数据库，当一个页面被用户打开的时候执行。也就是说，每当用户使用浏览器打开指定页面时，脚本便执行。与非持久性XSS攻击相比，持久性XSS攻击危害性更大。从名字就可以了解到，持久性XSS攻击就是将攻击代码存入数据库中，然后客户端打开时就执行这些攻击代码。

例如，留言板表单中的表单域：

<input type=“text” name=“content” value=“这里是用户填写的数据”>

正常操作流程是：用户是提交相应留言信息 —— 将数据存储到数据库 —— 其他用户访问留言板，应用去数据并显示；而非正常操作流程是攻击者在value填写:

<script>alert(‘foolish!’)；</script> <!--或者html其他标签（破坏样式。。。）、一段攻击型代码-->

并将数据提交、存储到数据库中；当其他用户取出数据显示的时候，将会执行这些攻击性代码。

4). 修复漏洞方针

　　漏洞产生的根本原因是 太相信用户提交的数据，对用户所提交的数据过滤不足所导致的，因此解决方案也应该从这个方面入手，具体方案包括：

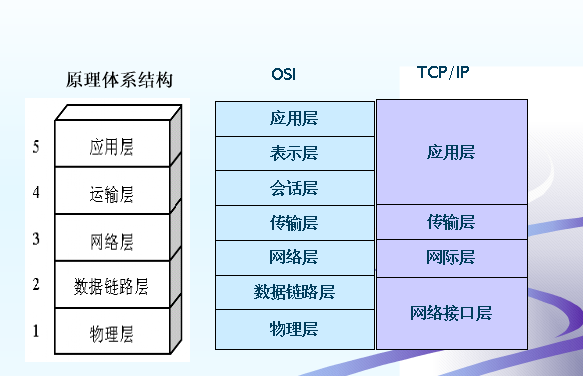
* 将重要的cookie标记为http only, 这样的话Javascript 中的document.cookie语句就不能   
  获取到cookie了（如果在cookie中设置了HttpOnly属性，那么通过js脚本将无法读取到cookie信息，这样能有效的防止XSS攻击）；
* 表单数据规定值的类型，例如：年龄应为只能为int、name只能为字母数字组合。。。。
* 对数据进行Html Encode 处理
* 过滤或移除特殊的Html标签，例如: <script>, <iframe> , < for <, > for>, &quot for
* 过滤JavaScript 事件的标签，例如 “onclick=”, “onfocus” 等等。

　　需要注意的是，在有些应用中是允许html标签出现的，甚至是javascript代码出现。因此，我们在过滤数据的时候需要仔细分析哪些数据是有特殊要求（例如输出需要html代码、javascript代码拼接、或者此表单直接允许使用等等），然后区别处理！

###### **16、OSI网络体系结构与TCP/IP协议模型**

为了更好地了解计算机网络体系结构，笔者以两篇博客的篇幅来介绍这个计算机网络中最为重要的知识点，具体见[《计算机网络体系结构综述（上）》](http://blog.csdn.net/justloveyou_/article/details/69611328) 和 [《计算机网络体系结构综述（下）》](http://blog.csdn.net/justloveyou_/article/details/69612153)。下面只做简要的总结。

　　在[《计算机网络体系结构综述（下）》](http://blog.csdn.net/justloveyou_/article/details/69612153)一文中，我们知道TCP/IP与OSI最大的不同在于：OSI是一个理论上的网络通信模型，而TCP/IP则是实际上的网络通信标准。但是，它们的初衷是一样的，都是为了使得两台计算机能够像两个知心朋友那样能够互相准确理解对方的意思并做出优雅的回应。现在，我们对OSI七层模型的各层进行简要的介绍：



1). 物理层

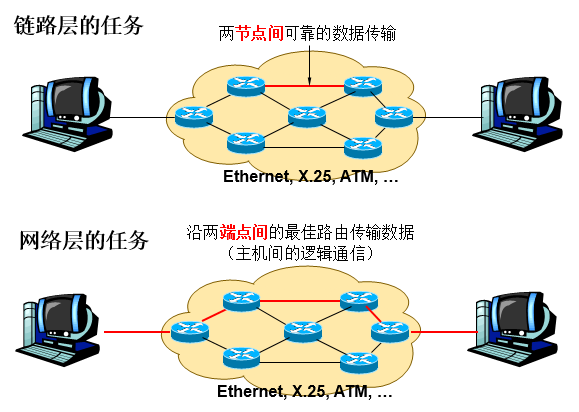
　　参考模型的最低层，也是OSI模型的第一层，实现了相邻计算机节点之间比特流的透明传送，并尽可能地屏蔽掉具体传输介质和物理设备的差异，使其上层(数据链路层)不必关心网络的具体传输介质。

2). 数据链路层（data link layer）

　　接收来自物理层的位流形式的数据，并封装成帧，传送到上一层；同样，也将来自上层的数据帧，拆装为位流形式的数据转发到物理层。这一层在物理层提供的比特流的基础上，通过差错控制、流量控制方法，使有差错的物理线路变为无差错的数据链路，即提供可靠的通过物理介质传输数据的方法。

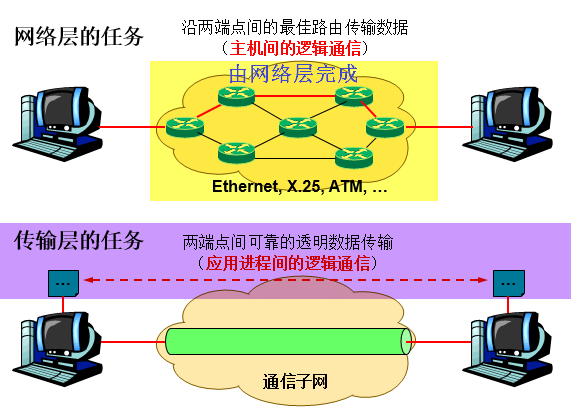
3). 网络层

　　将网络地址翻译成对应的物理地址，并通过路由选择算法为分组通过通信子网选择最适当的路径。



4). 传输层（transport layer）

　　在源端与目的端之间提供可靠的透明数据传输，使上层服务用户不必关系通信子网的实现细节。在协议栈中，传输层位于网络层之上，传输层协议为不同主机上运行的进程提供逻辑通信，而网络层协议为不同主机提供逻辑通信，如下图所示。



实际上，网络层可以看作是传输层的一部分，其为传输层提供服务。但对于终端系统而言，网络层对它们而言是透明的，它们知道传输层的存在，也就是说，在逻辑上它们认为是传输层为它们提供了端对端的通信，这也是分层思想的妙处。

5). 会话层（Session Layer）

会话层是OSI模型的第五层，是用户应用程序和网络之间的接口，负责在网络中的两节点之间建立、维持和终止通信。

6). 表示层（Presentation Layer）：数据的编码，压缩和解压缩，数据的加密和解密

表示层是OSI模型的第六层，它对来自应用层的命令和数据进行解释，以确保一个系统的应用层所发送的信息可以被另一个系统的应用层读取。

7). 应用层（Application layer）：为用户的应用进程提供网络通信服务

###### **17.TCP和UDP分别对应的常见应用层协议**

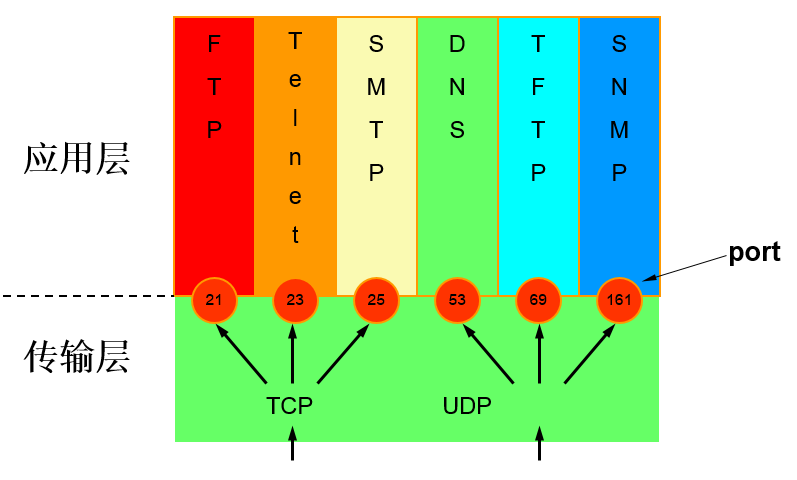
1). TCP对应的应用层协议

* FTP：定义了文件传输协议，使用21端口。常说某某计算机开了FTP服务便是启动了文件传输服务。下载文件，上传主页，都要用到FTP服务。
* Telnet：它是一种用于远程登陆的端口，用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于DOS模式下的通信服务。如以前的BBS是-纯字符界面的，支持BBS的服务器将23端口打开，对外提供服务。
* SMTP：定义了简单邮件传送协议，现在很多邮件服务器都用的是这个协议，用于发送邮件。如常见的免费邮件服务中用的就是这个邮件服务端口，所以在电子邮件设置-中常看到有这么SMTP端口设置这个栏，服务器开放的是25号端口。
* POP3：它是和SMTP对应，POP3用于接收邮件。通常情况下，POP3协议所用的是110端口。也是说，只要你有相应的使用POP3协议的程序（例如Fo-xmail或Outlook），就可以不以Web方式登陆进邮箱界面，直接用邮件程序就可以收到邮件（如是163邮箱就没有必要先进入网易网站，再进入自己的邮-箱来收信）。
* HTTP：从Web服务器传输超文本到本地浏览器的传送协议。

2). UDP对应的应用层协议

* 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。
* 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。
* 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。

3). 图示



###### **18、网络层的ARP协议工作原理**

　　网络层的ARP协议（在OSI模型中属于链路层，在TCP/IP中属于网络层）完成了IP地址与物理地址的映射。首先，每台主机都会在自己的ARP缓冲区中建立一个ARP列表，以表示IP地址和MAC地址的对应关系。当源主机需要将一个数据包要发送到目的主机时，会首先检查自己ARP列表中是否存在该IP地址对应的MAC地址：如果有，就直接将数据包发送到这个MAC地址；如果没有，就向本地网段发起一个ARP请求的广播包，查询此目的主机对应的MAC地址。此ARP请求数据包里包括源主机的IP地址、硬件地址、以及目的主机的IP地址。网络中所有的主机收到这个ARP请求后，会检查数据包中的目的IP是否和自己的IP地址一致。如果不相同就忽略此数据包；如果相同，该主机首先将发送端的MAC地址和IP地址添加到自己的ARP列表中，如果ARP表中已经存在该IP的信息，则将其覆盖，然后给源主机发送一个ARP响应数据包，告诉对方自己是它需要查找的MAC地址；源主机收到这个ARP响应数据包后，将得到的目的主机的IP地址和MAC地址添加到自己的ARP列表中，并利用此信息开始数据的传输。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

###### **19、IP地址的分类**

　　IP地址是指互联网协议地址，是IP协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。IP地址编址方案将IP地址空间划分为A、B、C、D、E五类，其中A、B、C是基本类，D、E类作为多播和保留使用，为特殊地址。

　　每个IP地址包括两个标识码（ID），即网络ID和主机ID。同一个物理网络上的所有主机都使用同一个网络ID，网络上的一个主机（包括网络上工作站，服务器和路由器等）有一个主机ID与其对应。A~E类地址的特点如下：

* A类地址：以0开头，第一个字节范围：0~127；
* B类地址：以10开头，第一个字节范围：128~191；
* C类地址：以110开头，第一个字节范围：192~223；
* D类地址：以1110开头，第一个字节范围为224~239；
* E类地址：以1111开头，保留地址

1). A类地址：1字节的网络地址 + 3字节主机地址，网络地址的最高位必须是“0”

　　一个A类IP地址是指， 在IP地址的四段号码中，第一段号码为网络号码，剩下的三段号码为本地计算机的号码。如果用二进制表示IP地址的话，A类IP地址就由1字节的网络地址和3字节主机地址组成，网络地址的最高位必须是“0”。A类IP地址中网络的标识长度为8位，主机标识的长度为24位，A类网络地址数量较少，有126个网络，每个网络可以容纳主机数达1600多万台。

　　A类IP地址的地址范围1.0.0.0到127.255.255.255（二进制表示为：00000001 00000000 00000000 00000000 - 01111110 11111111 11111111 11111111），最后一个是广播地址。A类IP地址的子网掩码为255.0.0.0，每个网络支持的最大主机数为256的3次方-2=16777214台。

2). B类地址: 2字节的网络地址 + 2字节主机地址，网络地址的最高位必须是“10”

　　一个B类IP地址是指，在IP地址的四段号码中，前两段号码为网络号码。如果用二进制表示IP地址的话，B类IP地址就由2字节的网络地址和2字节主机地址组成，网络地址的最高位必须是“10”。B类IP地址中网络的标识长度为16位，主机标识的长度为16位，B类网络地址适用于中等规模的网络，有16384个网络，每个网络所能容纳的计算机数为6万多台。

　　B类IP地址地址范围128.0.0.0-191.255.255.255（二进制表示为：10000000 00000000 00000000 00000000—-10111111 11111111 11111111 11111111），最后一个是广播地址。B类IP地址的子网掩码为255.255.0.0，每个网络支持的最大主机数为256的2次方-2=65534台。

3). C类地址: 3字节的网络地址 + 1字节主机地址，网络地址的最高位必须是“110”

　　一个C类IP地址是指，在IP地址的四段号码中，前三段号码为网络号码，剩下的一段号码为本地计算机的号码。如果用二进制表示IP地址的话，C类IP地址就由3字节的网络地址和1字节主机地址组成，网络地址的最高位必须是“110”。C类IP地址中网络的标识长度为24位，主机标识的长度为8位，C类网络地址数量较多，有209万余个网络。适用于小规模的局域网络，每个网络最多只能包含254台计算机。

　　C类IP地址范围192.0.0.0-223.255.255.255（二进制表示为: 11000000 00000000 00000000 00000000 - 11011111 11111111 11111111 11111111）。C类IP地址的子网掩码为255.255.255.0，每个网络支持的最大主机数为256-2=254台。

4). D类地址:多播地址，用于1对多通信，最高位必须是“1110”

　　D类IP地址在历史上被叫做多播地址(multicast address)，即组播地址。在以太网中，多播地址命名了一组应该在这个网络中应用接收到一个分组的站点。多播地址的最高位必须是“1110”，范围从224.0.0.0到239.255.255.255。

5). E类地址:为保留地址，最高位必须是“1111”

###### **20、IP地址与物理地址**

　　物理地址是数据链路层和物理层使用的地址，IP地址是网络层和以上各层使用的地址，是一种逻辑地址，其中ARP协议用于IP地址与物理地址的对应。

###### **21、 常见状态码及原因短语**

　　HTTP请求结构： 请求方式 + 请求URI + 协议及其版本   
　　HTTP响应结构： 状态码 + 原因短语 + 协议及其版本

* 1×× : 请求处理中，请求已被接受，正在处理
* 2×× : 请求成功，请求被成功处理   
  200 OK
* 3×× : 重定向，要完成请求必须进行进一步处理   
  301 : 永久性转移   
  302 ：暂时性转移   
  304 ： 已缓存
* 4×× : 客户端错误，请求不合法   
  400：Bad Request,请求有语法问题   
  403：拒绝请求   
  404：客户端所访问的页面不存在
* 5×× : 服务器端错误，服务器不能处理合法请求   
  500 ：服务器内部错误   
  503 ： 服务不可用，稍等

##### **22、TCP/IP四层模型** 　　ISO制定的OSI参考模型的过于庞大、复杂招致了许多批评。与此对照，由技术人员自己开发的TCP/IP协议栈获得了更为广泛的应用。如图2-1所示，是TCP/IP参考模型和OSI参考模型的对比示意图。 https://images.cnblogs.com/cnblogs_com/bluetzar/OSIVSTCP.jpg 　　TCP/IP协议栈是美国国防部高级研究计划局计算机网（Advanced Research Projects Agency Network，ARPANET）和其后继因特网使用的参考模型。ARPANET是由美国国防部（U.S．Department of Defense，DoD）赞助的研究网络。最初，它只连接了美国境内的四所大学。随后的几年中，它通过租用的电话线连接了数百所大学和政府部门。最终ARPANET发展成为全球规模最大的互连网络-因特网。最初的ARPANET于1990年永久性地关闭。　　 　　TCP/IP参考模型分为四个层次：应用层、传输层、网络互连层和主机到网络层。如图所示。 https://images.cnblogs.com/cnblogs_com/bluetzar/TCP4.jpg 　　在TCP/IP参考模型中，去掉了OSI参考模型中的会话层和表示层（这两层的功能被合并到应用层实现）。同时将OSI参考模型中的数据链路层和物理层合并为主机到网络层。下面，分别介绍各层的主要功能。 　　主机到网络层　　 　　实际上TCP/IP参考模型没有真正描述这一层的实现，只是要求能够提供给其上层-网络互连层一个访问接口，以便在其上传递IP分组。由于这一层次未被定义，所以其具体的实现方法将随着网络类型的不同而不同。　　 　　网络互连层　　 　　网络互连层是整个TCP/IP协议栈的核心。它的功能是把分组发往目标网络或主机。同时，为了尽快地发送分组，可能需要沿不同的路径同时进行分组传递。因此，分组到达的顺序和发送的顺序可能不同，这就需要上层必须对分组进行排序。　　 　　网络互连层定义了分组格式和协议，即IP协议（Internet Protocol）。　　 　　网络互连层除了需要完成路由的功能外，也可以完成将不同类型的网络（异构网）互连的任务。除此之外，网络互连层还需要完成拥塞控制的功能。　　 　　传输层　　 　　在TCP/IP模型中，传输层的功能是使源端主机和目标端主机上的对等实体可以进行会话。在传输层定义了两种服务质量不同的协议。即：传输控制协议TCP（transmission control protocol）和用户数据报协议UDP（user datagram protocol）。　　 　　TCP协议是一个面向连接的、可靠的协议。它将一台主机发出的字节流无差错地发往互联网上的其他主机。在发送端，它负责把上层传送下来的字节流分成报文段并传递给下层。在接收端，它负责把收到的报文进行重组后递交给上层。TCP协议还要处理端到端的流量控制，以避免缓慢接收的接收方没有足够的缓冲区接收发送方发送的大量数据。　　 　　UDP协议是一个不可靠的、无连接协议，主要适用于不需要对报文进行排序和流量控制的场合。　　 　　应用层　　 　　TCP/IP模型将OSI参考模型中的会话层和表示层的功能合并到应用层实现。　　 　　应用层面向不同的网络应用引入了不同的应用层协议。其中，有基于TCP协议的，如文件传输协议（File Transfer Protocol，FTP）、虚拟终端协议（TELNET）、超文本链接协议（Hyper Text Transfer Protocol，HTTP），也有基于UDP协议的。

###### **23、TCP/IP模型相关联的协议**

**1).ARP协议**

OSI模型有七层，TCP在第4层传输层，IP在第3层网络层，而ARP在第2层数据链路层。高层对低层是有强依赖的，所以TCP的建立前要进行ARP的请求和应答。

要了解ARP的作用，首先要分清两个“地址”：

　　（1）TCP/IP的32bit IP地址。仅知道主机的IP地址不能让内核发送数据帧给主机。

　　（2）网络接口的硬件地址，它是一个48bit的值，用来标识不同的以太网或令牌环网络接口。在硬件层次上，进行数据交换必须有正确的接口地址，内核必须知道目的端的硬件地址才能发送数据。

　　简言之，就是在以太网中，一台主机要把数据帧发送到同一局域网上的另一台主机时，设备驱动程序必须知道以太网地址(**以太网地址就是指MAC地址**)才能发送数据。而我们只知道IP地址，这时就需要采用ARP协议将IP地址映射为以太网地址。

　　要注意一点，一般认为ARP协议只使适用于局域网。

1) 本地主机在局域网中广播ARP请求，ARP请求数据帧中包含目的主机的IP地址。意思是“如果你是这个IP地址的拥有者，请回答你的硬件地址”。

2) 目的主机的ARP层解析这份广播报文，识别出是询问其硬件地址。于是发送ARP应答包，里面包含IP地址及其对应的硬件地址。

3) 本地主机收到ARP应答后，知道了目的地址的硬件地址，之后的数据报就可以传送了

**2).RARP协议(逆地址解析协议)**

将局域网中某个主机的物理地址转换为IP地址，比如局域网中有一台主机只知道物理地址而不知道IP地址，那么可以通过RARP协议发出征求自身IP地址的广播请求，然后由RARP服务器负责回答。RARP协议广泛应用于无盘工作站引导时获取IP地址。

RARP允许局域网的物理机器从网管服务器ARP表或者缓存上请求其IP地址。

**3).ICMP协议**

ICMP（Internet Control Message Protocol）因特网控制报文协议。它是IPv4协议族中的一个子协议，用于IP主机、路由器之间传递控制消息。控制消息是在网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然不传输用户数据，但是对于用户数据的传递起着重要的作用。   
ICMP协议与ARP协议不同，ICMP靠IP协议来完成任务，所以ICMP报文中要封装IP头部。它与传输层协议（如TCP和UDP）的目的不同，一般不用来在端系统之间传送数据，不被用户网络程序直接使用，除了想Ping和Tracert这样的诊断程序。

假定主机A的IP地址是192.168.1.1，主机B的IP地址是192.168.1.2，都在同一子网内，则当你在主机A上运行“**Ping** 192.168.1.2”后，都发生了些什么呢?  
  
首先，**Ping**命令会构建一个固定格式的ICMP请求数据包，然后由ICMP协议将这个数据包连同地址“192.168.1.2”一起交给IP层协议（和ICMP一样，实际上是一组后台运行的进程），IP层协议将以地址“192.168.1.2”作为目的地址，本机IP地址作为源地址，加上一些其他的控制信息，构建一个IP数据包，并在一个映射表中查找出IP地址192.168.1.2所对应的物理地址（也叫MAC地址，熟悉网卡配置的朋友不会陌生，这是数据链路层协议构建数据链路层的传输单元——帧所必需的），一并交给数据链路层。后者构建一个数据帧，目的地址是IP层传过来的物理地址，源地址则是本机的物理地址，还要附加上一些控制信息，依据以太网的介质访问规则，将它们传送出去。

其中映射表由ARP实现。ARP(Address Resolution Protocol)是地址解析协议,是一种将IP地址转化成物理地址的协议。ARP具体说来就是将网络层（IP层，也就是相当于OSI的第三层）地址解析为数据连接层（MAC层，也就是相当于OSI的第二层）的MAC地址。  
    主机B收到这个数据帧后，先检查它的目的地址，并和本机的物理地址对比，如符合，则接收；否则丢弃。接收后检查该数据帧，将IP数据包从帧中提取出来，交给本机的IP层协议。同样，IP层检查后，将有用的信息提取后交给ICMP协议，后者处理后，马上构建一个ICMP应答包，发送给主机A，其过程和主机A发送ICMP请求包到主机B一模一样。

即先由IP地址，在网络层传输，然后再根据mac地址由数据链路层传送到目的主机.

Traceroute是用来侦测主机到目的主机之间所经路由情况的重要工具，也是最便利的工具。前面说到，尽管**ping**工具也可以进行侦测，但是，因为ip头的限制，**ping**不能完全的记录下所经过的路由器。所以Traceroute正好就填补了这个缺憾.

**4).IGMP协议(组管理协议)**

该协议运行于主机和与主机直接相连的组播路由器之间，是IP主机用来报告多址广播组成员身份的协议。通过[IGMP协议](https://www.baidu.com/s?wd=IGMP%E5%8D%8F%E8%AE%AE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmvuBm1RsnWT1nH0vmvD30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnvrj6vrj0z)，一方面可以通过[IGMP协议](https://www.baidu.com/s?wd=IGMP%E5%8D%8F%E8%AE%AE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmvuBm1RsnWT1nH0vmvD30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnvrj6vrj0z)主机通知本地路由器希望加入并接收某个特定组播组的信息；另一方面，路由器通过[IGMP协议](https://www.baidu.com/s?wd=IGMP%E5%8D%8F%E8%AE%AE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmvuBm1RsnWT1nH0vmvD30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnvrj6vrj0z)周期性地查询局域网内某个已知组的成员是否处于活动状态。(通过IP数据包传输).

**5).DNS协议(域名解析协议)—UDP和TCP**

本机向本地域名服务器发出一个DNS请求报文，报文里携带需要查询的域名；第二个步骤是本地域名服务器向本机回应一个DNS响应报文，里面包含域名对应的IP地址。

DNS在进行区域传输的时候使用TCP协议，其它时候则使用UDP协议；   
    DNS的规范规定了2种类型的DNS服务器，一个叫主DNS服务器，一个叫辅助DNS服务器。在一个区中主DNS服务器从自己本机的数据文件中读取该区的DNS数据信息，而辅助DNS服务器则从区的主DNS服务器中读取该区的DNS数据信息。当一个辅助DNS服务器启动时，它需要与主DNS服务器通信，并加载数据信息，这就叫做区传送（zone transfer）。   
**为什么既使用TCP又使用UDP？**首先了解一下TCP与UDP传送字节的长度限制：   
   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：**i).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。   
ii).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：   
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。

**6).TFTP协议(域名解析协议)—UDP**

在项目中udp环境下需要进行文件传输，一种高效的文件传输。文件传输一般会想到FTP 和TFTP，但考虑限制条件，所以采用TFTP。

TFTP是什么？

TFTP是一种简单的文件传输协议。目标是在UDP之上上建立一个类似于FTP的但仅支持文件上传和下载功能的传输协议，所以它不包含FTP协议中的目录操作和用户权限等内容。

**7).BOOTP协议(引导程序协议)--UDP**

i).引导程序协议（BOOTP）是一个基于TCP/IP协议的协议，它可以让无盘站从一个中心服务器上获得IP地址，为局域网中的无盘工作站分配动态IP地址，并不需要每个用户去设置静态IP地址。  
        使用BOOTP协议的时候，一般包括Bootstrap Protocol Server（自举协议服务端）和Bootstrap Protocol Client（自举协议客户端）两部分。  
    ii).客户端获取IP地址的过程如下：  
        首先，由BOOTP启动代码启动客户端，这个时候客户端还没有IP地址，使用广播形式以IP地址255.255.255.255向网络中发出IP地址查询要求。  
        接着，运行BOOTP协议的服务器接收到这个请求，会根据请求中提供的MAC地址找到客户端，并发送一个含有IP地址、服务器IP地址、网关等信息的FOUND帧。  
        最后，客户端会根据该FOUND帧来通过专用TFTP服务器下载启动镜像文件，模拟成磁盘启动。

功能类似RRAP协议,但是有区别:

RA P使用链路层广播， RARP请求就不会被路由器转发

BOOTP使用UDP，且通常需与TFTP协同工作

**8).SNMP协议(简单网络管理协议)—UDP**

简单网络管理协议（SNMP：Simple Network Management Protocol）是由互联网工程任务组（IETF：Internet Engineering Task Force ）定义的一套网络管理协议。该协议基于简单网关监视协议（SGMP：Simple Gateway Monitor Protocol）。利用SNMP，一个管理工作站可以远程管理所有支持这种协议的网络设备，包括监视网络状态、修改网络设备配置、接收网络事件警告等。虽然SNMP开始是面向基于IP的网络管理，但作为一个工业标准也被成功用于电话网络管理。

采用UDP传输数据,因为它的格式与TCP相比少了很多的字段，简单了很多，这也是传输数据时效率高、SNMP采用的一个主要原因。因为在网络雍堵时，TCP会断线而UDP不会。

**9).FTP协议(文件传输协议)—UDP**

FTP 使用 2 个端口，一个数据端口和一个命令端口（也叫做控制端口）。这两个端口一般是21 （命令端口）和 20 （数据端口）。控制 Socket 用来传送命令，数据 Socket 是用于传送数据。每一个 FTP 命令发送之后，FTP 服务器都会返回一个字符串，其中包括一个响应代码和一些说明信息。其中的返回码主要是用于判断命令是否被成功执行了。

**命令端口**

一般来说，客户端有一个 Socket 用来连接 FTP 服务器的相关端口，它负责 FTP 命令的发送和接收返回的响应信息。一些操作如“登录”、“改变目录”、“删除文件”，依靠这个连接发送命令就可完成。

**数据端口**

对于有数据传输的操作，主要是显示目录列表，上传、下载文件，我们需要依靠另一个 Socket来完成。

如果使用被动模式，通常服务器端会返回一个端口号。客户端需要用另开一个 Socket 来连接这个端口，然后我们可根据操作来发送命令，数据会通过新开的一个端口传输。

如果使用主动模式，通常客户端会发送一个端口号给服务器端，并在这个端口监听。服务器需要连接到客户端开启的这个数据端口，并进行数据的传输。

**10).SMTP协议(简单邮件传输协议)—UDP**

SMTP称为简单邮件传输协议（Simple Mail Transfer Protocal），目标是向用户提供高效、可靠的邮件传输。它的一个重要特点是它能够在传送中接力传送邮件，即邮件可以通过不同网络上的主机接力式传送。通常它工作在两种情况下：一是邮件从客户机传输到服务器；二是从某一个服务器传输到另一个服务器。SMTP是一个请求/响应协议，它监听25号端口，用于接收用户的Mail请求，并与远端Mail服务器建立SMTP连接。

**11). 常用的数据传输速率单位有Kb/s、Mb/s、Gb/s，它们之间的换算关系为：1Kb/s=10^3b/s，1Mb/s=10^6b/s，1Gb/s=10^9b/s， 所以1Gb/s=10＾9b/s=10＾3Mb/s。**

###### ****24.**** [****HTTP长连接和短连接****](http://www.cnblogs.com/0201zcr/p/4694945.html)

1). HTTP协议与TCP/IP协议的关系

　　HTTP的长连接和短连接本质上是TCP长连接和短连接。HTTP属于应用层协议，在传输层使用TCP协议，在网络层使用IP协议。IP协议主要解决网络路由和寻址问题，TCP协议主要解决如何在IP层之上可靠的传递数据包，使在网络上的另一端收到发端发出的所有包，并且顺序与发出顺序一致。TCP有可靠，面向连接的特点

2). 如何理解HTTP协议是无状态的

　　HTTP协议是无状态的，指的是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。也就是说，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是一个无状态的面向连接的协议，无状态不代表HTTP不能保持TCP连接，更不能代表HTTP使用的是UDP协议（无连接）。

3). 什么是长连接、短连接？

　　在HTTP/1.0中，默认使用的是短连接。也就是说，浏览器和服务器每进行一次HTTP操作，就建立一次连接，但任务结束就中断连接。如果客户端浏览器访问的某个HTML或其他类型的 Web页中包含有其他的Web资源，如JavaScript文件、图像文件、CSS文件等；当浏览器每遇到这样一个Web资源，就会建立一个HTTP会话。

但从 HTTP/1.1起，默认使用长连接，用以保持连接特性。使用长连接的HTTP协议，会在响应头有加入这行代码：

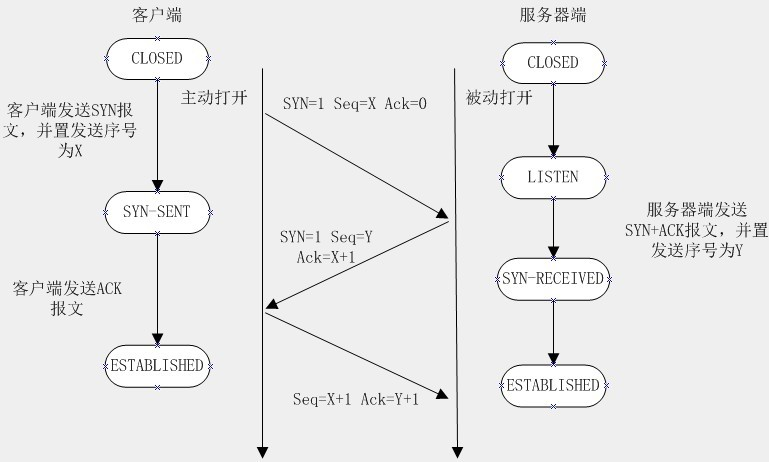
|  |
| --- |
| Connection:keep-alive |

　　在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的 TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现长连接要客户端和服务端都支持长连接。

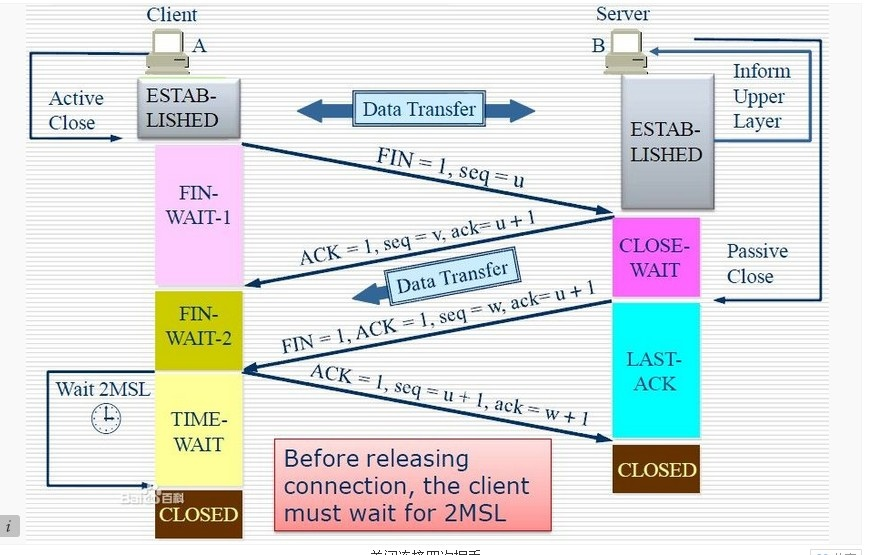
HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。  
3.1) TCP连接

　　当网络通信时采用TCP协议时，在真正的读写操作之前，server与client之间必须建立一个连接，当读写操作完成后，双方不再需要这个连接 时它们可以释放这个连接，连接的建立是需要三次握手的，而释放则需要4次握手，所以说每个连接的建立都是需要资源消耗和时间消耗的

经典的三次握手示意图：



经典的四次握手关闭图：



3.2) TCP短连接

　　我们模拟一下TCP短连接的情况，client向server发起连接请求，server接到请求，然后双方建立连接。client向server 发送消息，server回应client，然后一次读写就完成了，这时候双方任何一个都可以发起close操作，不过一般都是client先发起 close操作。为什么呢，一般的server不会回复完client后立即关闭连接的，当然不排除有特殊的情况。从上面的描述看，短连接一般只会在 client/server间传递一次读写操作

短连接的优点是：管理起来比较简单，存在的连接都是有用的连接，不需要额外的控制手段

3.3) TCP长连接

　　接下来我们再模拟一下长连接的情况，client向server发起连接，server接受client连接，双方建立连接。Client与server完成一次读写之后，它们之间的连接并不会主动关闭，后续的读写操作会继续使用这个连接。

首先说一下TCP/IP详解上讲到的TCP保活功能，保活功能主要为服务器应用提供，服务器应用希望知道客户主机是否崩溃，从而可以代表客户使用资源。如果客户已经消失，使得服务器上保留一个半开放的连接，而服务器又在等待来自客户端的数据，则服务器将应远等待客户端的数据，保活功能就是试图在服务 器端检测到这种半开放的连接。

如果一个给定的连接在两小时内没有任何的动作，则服务器就向客户发一个探测报文段，客户主机必须处于以下4个状态之一：

1. 客户主机依然正常运行，并从服务器可达。客户的TCP响应正常，而服务器也知道对方是正常的，服务器在两小时后将保活定时器复位。
2. 客户主机已经崩溃，并且关闭或者正在重新启动。在任何一种情况下，客户的TCP都没有响应。服务端将不能收到对探测的响应，并在75秒后超时。服务器总共发送10个这样的探测 ，每个间隔75秒。如果服务器没有收到一个响应，它就认为客户主机已经关闭并终止连接。
3. 客户主机崩溃并已经重新启动。服务器将收到一个对其保活探测的响应，这个响应是一个复位，使得服务器终止这个连接。
4. 客户机正常运行，但是服务器不可达，这种情况与2类似，TCP能发现的就是没有收到探查的响应。

3.4) 长连接短连接操作过程

短连接的操作步骤是：

建立连接——数据传输——关闭连接...建立连接——数据传输——关闭连接

长连接的操作步骤是：

建立连接——数据传输...（保持连接）...数据传输——关闭连接

4). 长连接和短连接的优点和缺点

　　由上可以看出，长连接可以省去较多的TCP建立和关闭的操作，减少浪费，节约时间。对于频繁请求资源的客户来说，较适用长连接。不过这里存在一个问题，存活功能的探测周期太长，还有就是它只是探测TCP连接的存活，属于比较斯文的做法，遇到恶意的连接时，保活功能就不够使了。在长连接的应用场景下，client端一般不会主动关闭它们之间的连接，Client与server之间的连接如果一直不关闭的话，会存在一个问题，随着客户端连接越来越多，server早晚有扛不住的时候，这时候server端需要采取一些策略，如关闭一些长时间没有读写事件发生的连接，这样可 以避免一些恶意连接导致server端服务受损；如果条件再允许就可以以客户端机器为颗粒度，限制每个客户端的最大长连接数，这样可以完全避免某个蛋疼的客户端连累后端服务。

短连接对于服务器来说管理较为简单，存在的连接都是有用的连接，不需要额外的控制手段。但如果客户请求频繁，将在TCP的建立和关闭操作上浪费时间和带宽。

长连接和短连接的产生在于client和server采取的关闭策略，具体的应用场景采用具体的策略，没有十全十美的选择，只有合适的选择。

5). 什么时候用长连接，短连接？   
 　　长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况，。每个TCP连接都需要三步握手，这需要时间，如果每个操作都是先连接，再操作的话那么处理速度会降低很多，所以每个操作完后都不断开，次处理时直接发送数据包就OK了，不用建立TCP连接。例如：数据库的连接用长连接， 如果用短连接频繁的通信会造成socket错误，而且频繁的socket 创建也是对资源的浪费。   
　　而像WEB网站的http服务一般都用短链接，因为长连接对于服务端来说会耗费一定的资源，而像WEB网站这么频繁的成千上万甚至上亿客户端的连接用短连接会更省一些资源，如果用长连接，而且同时有成千上万的用户，如果每个用户都占用一个连接的话，那可想而知吧。所以并发量大，但每个用户无需频繁操作情况下需用短连好。

###### ****25.http/0.9, http/1.0, http/1.1, http/2.0的区别****

HTTP/0.9是第一个版本的HTTP协议，已过时。它的组成极其简单，只允许客户端发送GET这一种请求，且不支持请求头。由于没有协议头，造成了HTTP/0.9协议只支持一种内容，即纯文本。不过网页仍然支持用HTML语言格式化，同时无法插入图片。

HTTP/0.9具有典型的无状态性，每个事务独立进行处理，事务结束时就释放这个连接。由此可见，HTTP协议的无状态特点在其第一个版本0.9中已经成型。一次HTTP/0.9的传输首先要建立一个由客户端到Web服务器的TCP连接，由客户端发起一个请求，然后由Web服务器返回页面内容，然后连接会关闭。如果请求的页面不存在，也不会返回任何错误码。

HTTP/1.0相对于HTTP/0.9增加了如下主要特性：

* 请求与响应支持头域
* 响应对象以一个响应状态行开始
* 响应对象不只限于超文本
* 开始支持客户端通过POST方法向Web服务器提交数据，支持GET、HEAD、POST方法
* 支持长连接（但默认还是使用短连接），缓存机制，以及身份认证

HTTP/1.1是目前主流的HTTP协议版本，相对于HTTP/1.0新增了以下内容：

1)默认为长连接

HTTP 1.1支持长连接（PersistentConnection）和请求的流水线（Pipelining）处理，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟，在HTTP1.1中默认开启Connection：keep-alive，一定程度上弥补了HTTP1.0每次请求都要创建连接的缺点。

2)提供了范围请求功能(宽带优化)

HTTP1.0中，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点续传功能，HTTP1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是206（Partial Content），这样就方便了开发者自由的选择以便于充分利用带宽和连接。这是支持文件断点续传的基础。

3)提供了虚拟主机的功能(HOST域)

在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）。

4)多了一些缓存处理字段

HTTP/1.1在1.0的基础上加入了一些cache的新特性，引入了实体标签，一般被称为e-tags，新增更为强大的Cache-Control头。

5)错误通知的管理

在HTTP1.1中新增了24个错误状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示服务器上的某个资源被永久性的删除。

HTTP/2.0，相对于HTTP/1.1新增了以下内容：

1)二进制分帧

HTTP 2.0 的所有帧都采用二进制编码

* + 帧：客户端与服务器通过交换帧来通信，帧是基于这个新协议通信的最小单位。
  + 消息：是指逻辑上的 HTTP 消息，比如请求、响应等，由一或多个帧组成。
  + 流：流是连接中的一个虚拟信道，可以承载双向的消息；每个流都有一个唯一的整数标识符（1、2 … N）；

2)多路复用

多路复用允许同时通过单一的HTTP/2.0 连接发起多重的请求-响应消息。有了新的分帧机制后，HTTP/2.0不再依赖多个TCP 连接去处理更多并发的请求。每个数据流都拆分成很多互不依赖的帧，而这些帧可以交错（乱序发送），还可以分优先级。最后再在另一端根据每个帧首部的流标识符把它们重新组合起来。HTTP 2.0 连接都是持久化的，而且客户端与服务器之间也只需要一个连接（每个域名一个连接）即可。

3)头部压缩

HTTP/1.1 的首部带有大量信息，而且每次都要重复发送。HTTP/2.0 要求通讯双方各自缓存一份首部字段表，从而避免了重复传输。

4)请求优先级

浏览器可以在发现资源时立即分派请求，指定每个流的优先级，让服务器决定最优的响应次序。这样请求就不必排队了，既节省了时间，也最大限度地利用了每个连接。

5)服务端推送

服务端推送能把客户端所需要的资源伴随着index.html一起发送到客户端，省去了客户端重复请求的步骤。正因为没有发起请求，建立连接等操作，所以静态资源通过服务端推送的方式可以极大地提升速度。