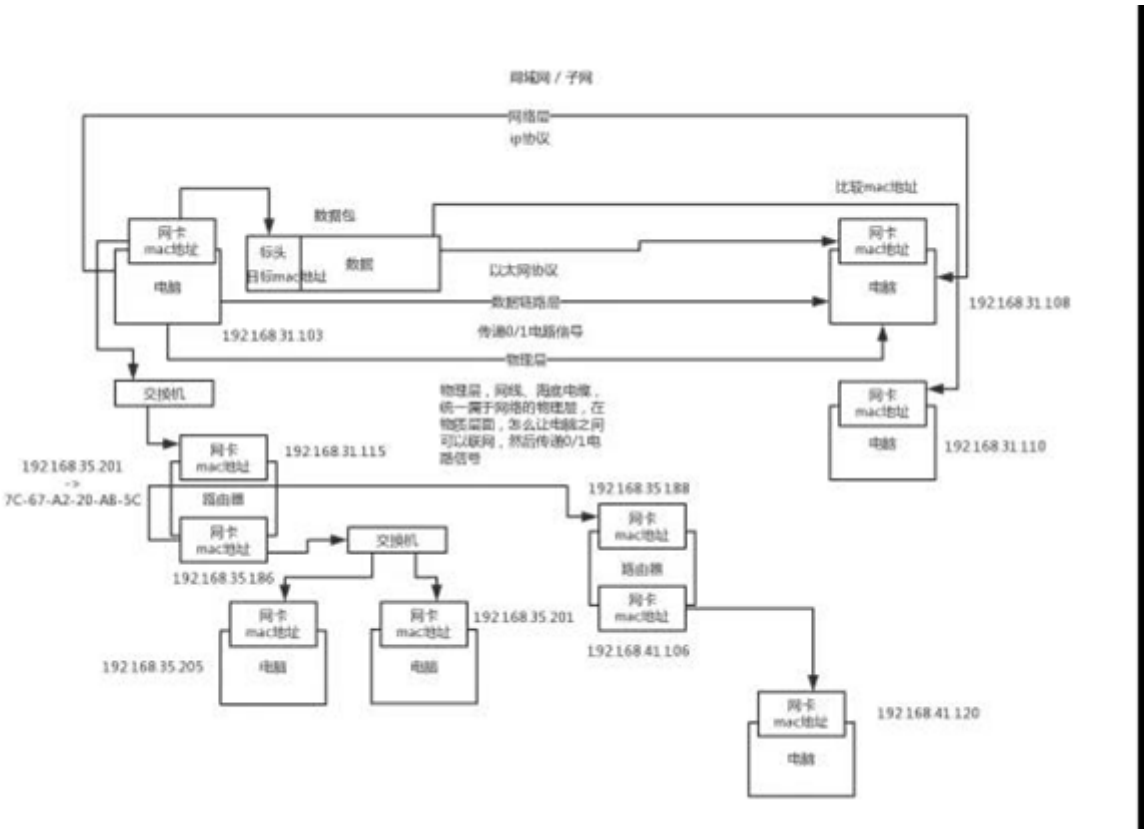
你能聊聊TCP/IP四层网络模型吗？OSI七层网络模型也说一下！

TCP/IP四层：数据链路层（以太网协议）、网络层（IP协议）、传输层（TCP协议）、应用层（HTTP协议）。



数据链路层(以太网协议**)**：

一堆0/1电路信号，封装成数据包，包含头(head)和数据(data)，head里包含从哪儿来到哪儿去，从一台电脑的一个网卡，发送到另外一台电脑的一个网卡，以太网发送的数据包指定目标电脑网卡的mac地址。

网络层(IP协议**)**：

IPv4和IPv6，IPv4由32个二进制组成，一般用4个十进制数字表示，从0.0.0.0到255.255.255.255之间。IP地址前24位(前3个十进制数字)代表网络，后8位(最后一个十进制数字)代表主机。

前3个十进制数字相同表示同一个子网的。子网掩码：用于判断两个IP地址是不是同一个子网。判断方法就是分别把两个IP地址和自己的子网掩码进行二进制与运算，比较代表网络那部分是否相同。

路由器，负载将多个子网进行连接，实际就是配置了多个网卡的设备，可以通过不同网卡接入不同的网络。**网关**，其实是路由器的一种，运作在网络层。可以就把路由器上的ip地址认为是网关，路由器上每个网卡都有mac地址和对应的ip地址。路由器虽然有mac地址，但是不能通过mac地址寻址的，必须通过ip地址寻址，所以路由器其实是工作在网络层的设备。

网络交换机，是通过mac地址来寻址和传输数据包的；但是路由器是通过ip地址寻址和传输数据包的。网络交换机主要用在**局域网的通信**，一般你架设一个局域网，里面的电脑通信是通过数据链路层发送数据包，通过mac地址来广播的，广播的时候就是通过网络交换机这个设备来把数据广播到局域网内的其他机器上去的。

一个子网内的机器之间通信，就是在数据包里写上对方的mac地址，然后交换机广播出去ok了；但是如果是跨子网的通信，就是写上对方的ip地址，然后先通过mac地址广播到路由器，让路由器再根据另外一个子网的ip地址转换为mac地址，通过另外一个子网的交换机广播过去。

传输层(TCP协议)：

端口，发送数据包到某个机器的一个网卡的某个端口上去，然后那个机器上监听那个端口的程序，就可以提取发送到这个端口的数据，知道是自己的数据。端口号是0~65536的范围内，0~1023被系统占用了，别的应用程序就用1024以上的端口。

网络层，是基于ip协议，进行主机和主机间的寻址和通信的，然后传输层，其实是建立某个主机的某个端口，到另外一个主机的某个端口的连接和通信的。

这个通信，就是通过socket来实现的，通过socket就可以基于tcp/ip协议完成刚才上面说的一系列的比如基于ip地址和mac地址转换和寻址啊，通过路由器通信啊之类的，而且会建立一个端口到另外一个端口的连接。

应用层(HTTP协议)：

针对各种不同的应用，邮件、网页之类的，都是定义不同的应用层协议的。

比如最常见的，应用层的协议就是http协议，进行网络通信。电子邮件协议，SMTP、POP3、IMAP4等。

GET http://localhost:8080/　http/1.1

key:value

用浏览器请求一个链接的时候，经历了哪些过程?

假设我们电脑设置了几个东西如下：

ip地址：192.168.31.37

子网掩码：255.255.255.0

网关地址：192.168.31.1

DNS地址：8.8.8.8

1.请求www.baidu.com地址，这个时候找DNS服务器，DNS服务器解析域名之后，返回一个ip地址，比如172.194.26.108。

2. 判断本机和访问的IP是不是同一个子网，用子网掩码坐与运算。通常不在一个子网，就发送数据包给网关，也就是路由器。

通过浏览器访问网站，走应用层的http协议，把浏览器发出的请求打包为数据包。

把http协议请求，有请求头，空行，请求数据，就构成http请求报文，封装到一个应用层数据包里。

3. 接下来走到传输层，按照TCP协议，TCP协议会让你设置端口，发送方端口随机选择，接收端一般默认80端口。这时会把应用层数据包封装到TCP数据包，再加一个TCP头，TCP头放了端口信息。

4. 走网络层，会把TCP头和TCP数据包，放到IP数据包里，再做了一个IP头，IP头包括本机和目标机器的IP地址。通过IP 协议，判断两个IP不在一个子网内，则将数据包通过以太网协议广播到网关，通过网关在发生出去。

5. 最后走到数据链路层，把IP头和IP数据包封装到以太网数据包，再增加一个以太网数据包头，头里放了本机网卡MAC地址和网关MAC地址。但以太网数据包有1500字节的限制，超过要切分为多个数据包，每个数据包包含以太网头、IP头和切割后的IP数据包。

以太网数据包通过交换机发送到网关，然后通过路由器转发到别的子网或者别的路由器，以此类推，通过N个路由器或网关转发，最终到达目标服务器，比如172.194.26.108。

目标服务器接收到以太网数据包后，从IP数据包，拿出TCP数据包，再从TCP数据包取出HTTP数据包，读取出HTTP数据包里各种协议内容，比如html页面，或者业务处理，然后再把响应结果封装成http响应报文，封装到http数据包里，再封装TCP数据包，封装IP数据包，封装以太网数据包，再通过网关发送回去，完成整个请求过程。

画一下TCP三次握手流程图？为啥是三次而不是二次或者四次呢？

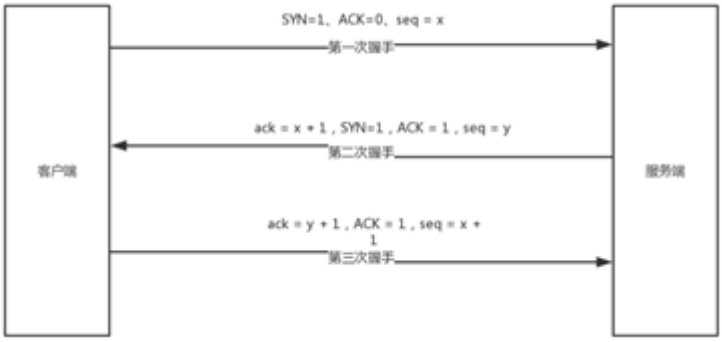
三次握手：

建立三次握手的时候，TCP报头用到了下面几个东西，ACK、SYN、FIN。

第一次握手，客户端发送连接请求报文，此时SYN=1、ACK=0，这就是说这是个连接请求，seq = x，接着客户端处于SYN_SENT状态，等待服务器响应。

第二次握手，服务端收到SYN=1的请求报文，需要返回一个确认报文，ack = x + 1，SYN=1，ACK = 1，seq = y，发送给客户端，自己处于SYN_RECV状态。

第三次握手，客户端收到了报文，将ack = y + 1，ACK = 1，seq = x + 1



重要的问题，为什么不是二次或四次？

二次握手的问题：

假设两次握手就ok了，要是客户端第一次握手过去，结果卡在某个地方了，没到服务端；完了客户端再次重试发送了第一次握手过去，服务端收到了，ok了，大家来回来去，三次握手建立了连接。

结果，尴尬的是，后来那个卡在哪儿的老的第一次握手发到了服务器，服务器直接就返回一个第二次握手，这个时候服务器开辟了资源准备客户端发送数据啥的，结果呢？客户端根本就不会理睬这个发回去的二次握手，因为之前都通信过了。

但是如果是三次握手，那个二次握手发回去，客户端发现根本不对，就会发送个复位的报文过去，让服务器撤销开辟的资源，别等着了。

四次挥手：

第一次挥手，客户端发送报文，FIN=1，seq=u，此时进入FIN-WAIT-1状态。

第二次挥手，服务端收到报文，这时候进入CLOSE_WATI状态，返回一个报文，ACK=1，ack=u+1，seq=v。客户端收到这个报文之后，直接进入FIN-WAIT-2状态，此时客户端到服务端

的连接就释放了。

第三次挥手，服务端发送连接释放报文，FIN=1，ack=u+1，seq=w，服务端进入LAST-ACK状态。

第四次挥手，客户端收到连接释放报文之后，发应答报文，ACK=1，ack=w+1，seq=u+1，进入TIME_WAIT状态，等待一会儿客户端进入CLOSED状态，服务端收到报文之后就进入CLOSED状态。

聊聊HTTP协议的工作原理！

说一下http的工作流程?http1.0、http1.1、http2.0具体有哪些区别?

http请求封装到应用层数据包，封装在tcp数据包，封装在ip数据包，封装在以太网数据包，如果过大，可能会拆成几个包，走以太网协议+交换机 -> 广播 -> 网关 -> 多个网关 -> 目标的机器 -> 一层一层拆包 -> http请求报文 -> 传递给tomcat -> spring mvc -> http响应 -> 一样的路径会去。

浏览器 -> 网站，互相之间是先要通过tcp三次握手，建立一个连接，浏览器和网站互相都给对方留出一份资源，浏览器发起http请求 -> tcp -> ip -> 以太网，网站上面去，网站返回一个响应，连接关闭，tcp四次挥手。释放掉浏览器和网站各自给对方保持的一份资源。

http 1.0要指定keep-alive来开启持久连接，默认是短连接，就是浏览器每次请求都要重新建立一次tcp连接，完事儿了就释放tcp连接。早期的网页都很low，没啥东西，就一点文字，就用这个没问题。

2000年之前，那个时候网页，都很low，当时你打开一个网页，就是说现场底层tcp三次握手，跟网站建立一个tcp连接，然后通过这个tcp连接，发送一次http请求，网站返回一个http响应（网页的html，里面有一大段文字），浏览器收到html渲染成网页，浏览器就走tcp四次挥手，跟网站断开连接了。

2000之后，2010之后更不用说了，网页发展很迅猛，一个网页包含着大量的css、js、图片等资源。比如你请求一个网页，这个网页的html先过来，过来之后，浏览器再次发起大量的请求去加载css、js、图片，打开一个网页可能浏览器要对网站服务器发送几十次请求。再频繁建立和释放tcp连接，会很慢很慢。

http 1.1默认支持长连接，就是说，浏览器打开一个网页之后，底层的tcp连接就保持着，不会立马断开，之后加载css、js之类的请求，都会基于这个tcp连接来走。http 1.1还支持host头，也就可以支持虚拟主机；而且对断点续传有支持。

浏览器，第一次请求去一个网站的一个页面的时候，就会打开一个tcp连接，接着就在一段时间内都不关闭了，然后接下来这个网页加载css、js、图片大量的请求全部走同一个tcp连接，频繁的发送请求获取响应，最后过了一段时间，这些事儿都完了，然后才会去释放那一个tcp连接。大幅度的提升复杂网页的打开的速度，性能。

未来http 2.0，支持多路复用，基于一个tcp连接并行发送多个请求以及接收响应，解决了http 1.1对同一时间同一个域名的请求有限制的问题。二进制分帧，将传输数据拆分为更小的帧（数据包），frame（数据包，帧），提高了性能，实现低延迟高吞吐。

聊聊HTTPS的工作原理？为啥用HTTPS就可以加密通信？

http协议都是明文的，是没有加密的，所以其实现在一般大部分应用都是用https协议的。之前是基于SSL协议对http进行加密，后来又升级到了TSL协议来加密，现在称之为SSL / TSL。

https的工作原理大概是这样的：

（1）浏览器把自己支持的加密规则发送给网站。

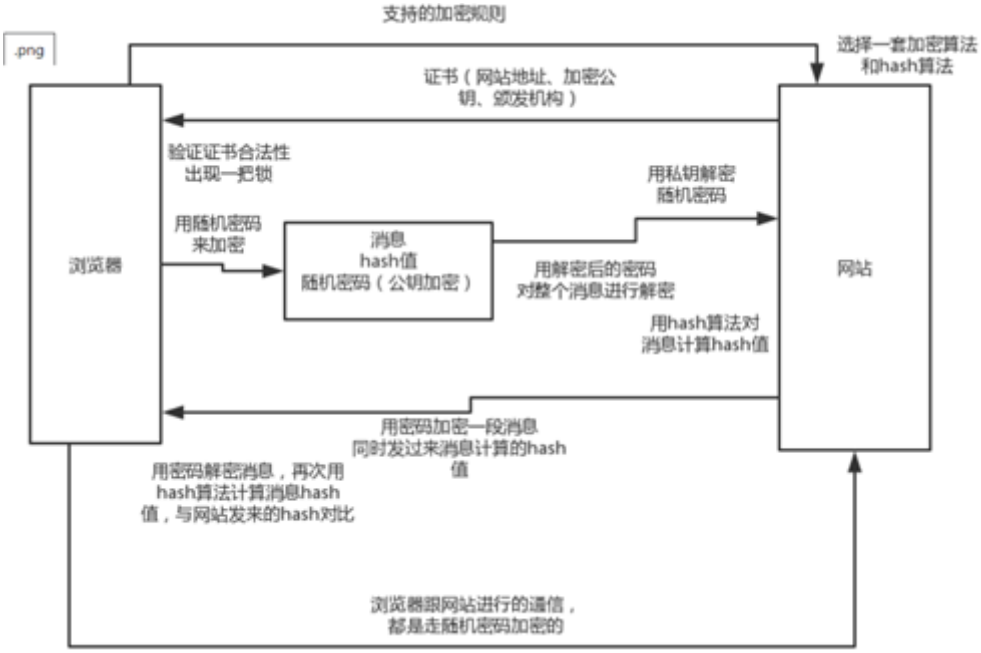
（2）网站从这套加密规则里选出来一套加密算法和hash算法，然后把自己的身份信息用证书的方式发回给浏览器，证书里有网站地址、加密公钥、证书颁发机构。

（3）浏览器验证证书的合法性，然后浏览器地址栏上会出现一把小锁；浏览器接着生成一串随机数密码，然后用证书里的公钥进行加密，这块走的非对称加密；用约定好的hash算法生成握手消息的hash值，然后用密码对消息进行加密，然后把所有东西都发给网站，这块走的是对称加密。

（4）网站，从消息里面可以取出来公钥加密后的随机密码，用本地的私钥对消息解密取出来密码，然后用密码解密浏览器发来的握手消息，计算消息的hash值，并验证与浏览器发送过来的hash值是否一致，最后用密码加密一段握手消息，发给浏览器。

（5）浏览器解密握手消息，然后计算消息的hash值，如果跟网站发来的hash一样，握手就结束，之后所有的数据都会由之前浏览器生成的随机密码，然后用对称加密来进行进行加密。

常用的非对称加密是RSA算法，对称加密是AES、RC4等，hash算法就是MD5。



聊聊http的长连接的工作原理到底是啥?

http本身没什么所谓的长连接短连接之说，其实说白了都是http下层的tcp连接是长连接还是短连接，tcp连接保持长连接，那么多个http请求和响应都可以通过一个链接来走。其实http 1.1之后，默认都是走长连接了，就是底层都是一个网页一个tcp连接，一个网页的所有图片、css、js的资源加载，都走底层一个tcp连接，来多次http请求即可。

http 1.0的时候，底层的tcp是短连接，一个网页发起的请求，每个请求都是先tcp三次握手，然后发送请求，获取响应，然后tcp四次挥手断开连接；每个请求，都会先连接再断开。短连接，建立连接之后，发送个请求，直接连接就给断开了。

http 1.1，tcp长连接，tcp三次握手，建立了连接，无论有多少次请求都是走一个tcp连接的，走了n多次请求之后，然后tcp连接被释放掉了。