**HTTP/HTTPS协议**

1. **OSI（Open System Interconnection）模型**

参考模型是国际标准化组织（ISO）制定的一个用于计算机或通信系统间互联的标准体系，一般称为OSI参考模型或七层模型。

****

**应用层：**

网络服务与最终用户的一个接口。

协议有：HTTP、HTTPS、DNS、FTP等

**表示层：**

数据的表示、安全、压缩。（在五层模型里面已经合并到了应用层）

格式有，JPEG、ASCll、DECOIC、加密格式等

**会话层：**

建立、管理、终止会话。（在五层模型里面已经合并到了应用层）

对应主机进程，指本地主机与远程主机正在进行的会话

**传输层（数据段 Segment）：**

定义传输数据的协议端口号，以及流控和差错校验。

协议有：TCP UDP，数据包一旦离开网卡即进入网络传输层

**网络层 （数据包 Packet）：**

进行逻辑地址寻址，实现不同网络之间的路径选择。

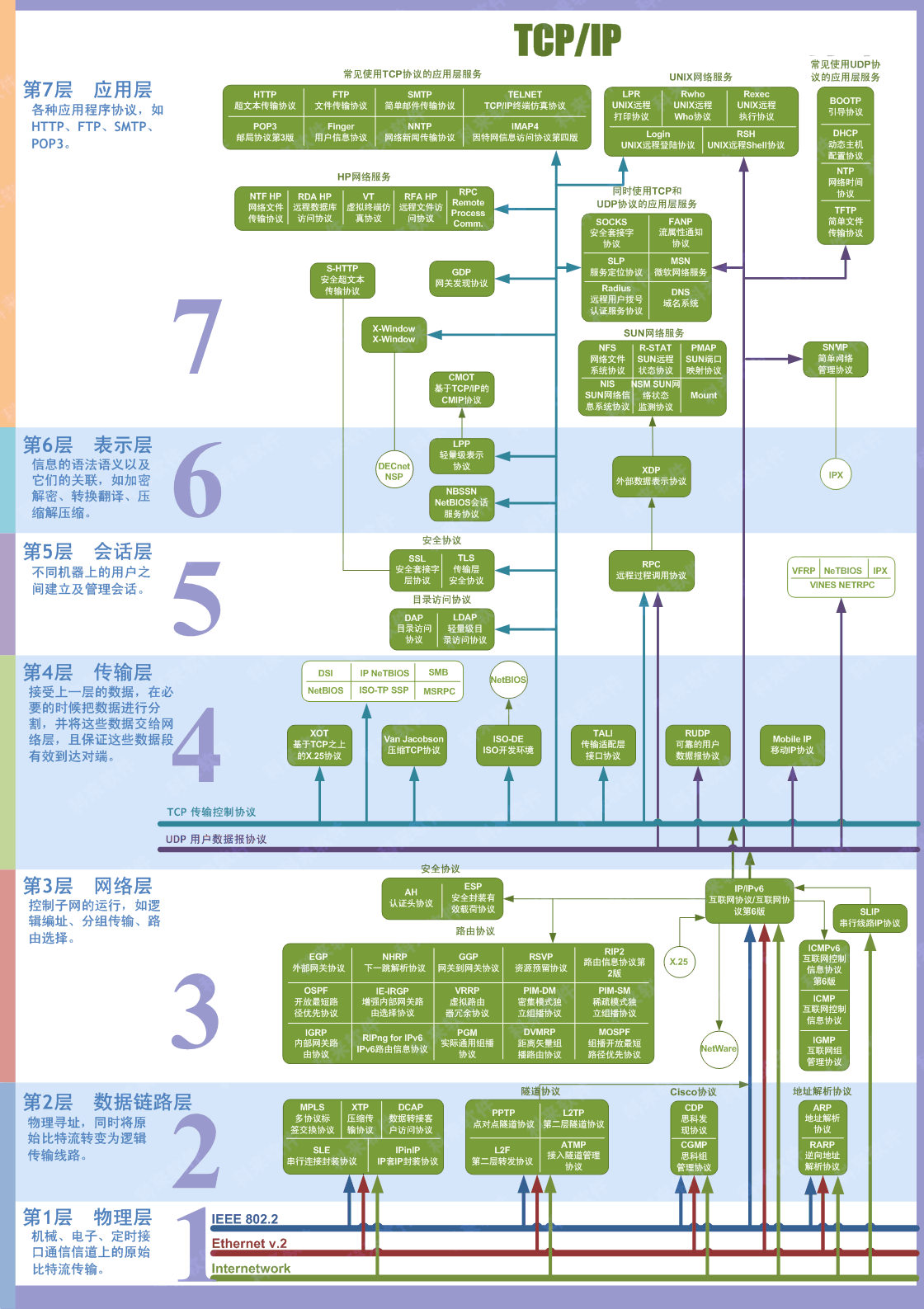
协议有：ICMP IGMP IP（IPV4 IPV6）

**数据链路层 （帧 Frame）：**

建立逻辑连接、进行硬件地址寻址、差错校验等功能。（由底层网络定义协议）。

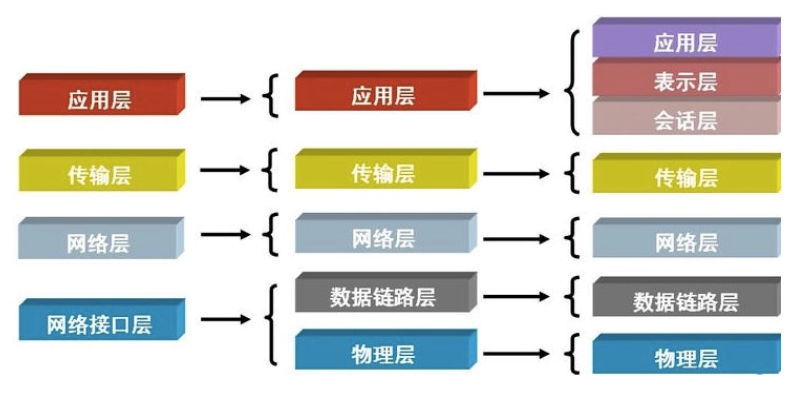
**物理层 （比特 bit）：**

建立、维护、断开物理连接。（由底层网络定义协议）



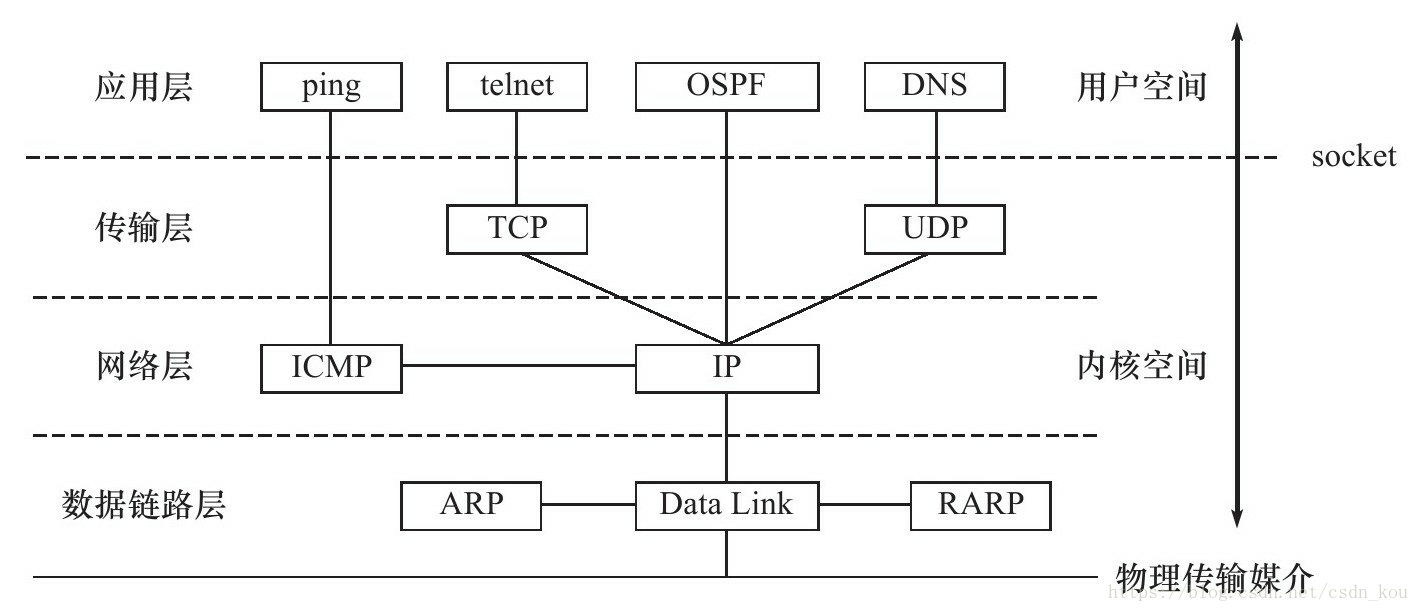
1. **TCP/IP四层模型**

TCP/IP四层协议和OSI的七层协议对应关系如下



应用层对应OSI的应用层、表示层、会话层合并

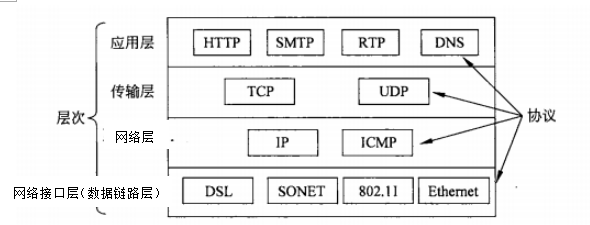
网络接口层对应OSI的物理层和数据链路层



1. **TCP/IP协议簇**

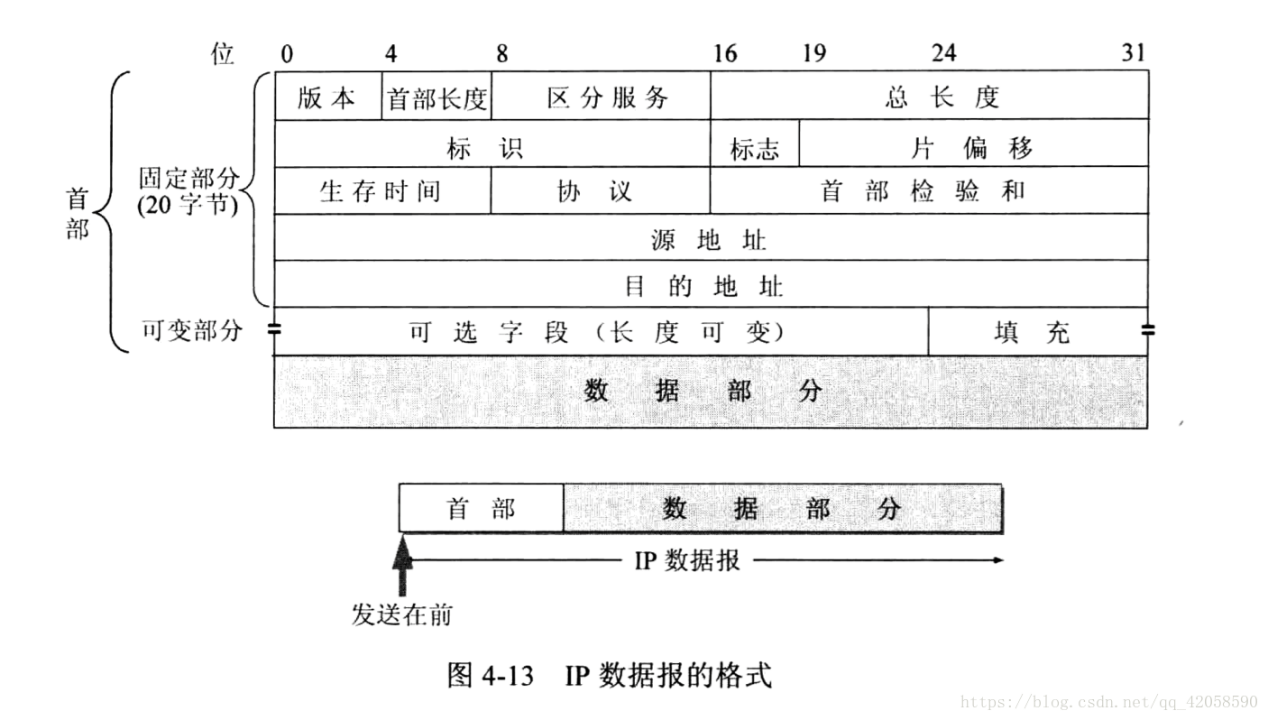
TCP/IP协议不仅仅指的是TCP 和IP两个协议，而是指一个由FTP、SMTP、TCP、UDP、IP等协议构成的协议簇， 只是因为在TCP/IP协议中TCP协议和IP协议最具代表性，所以被称为TCP/IP协议。是如今最基本、使用最广泛的网络通信协议。

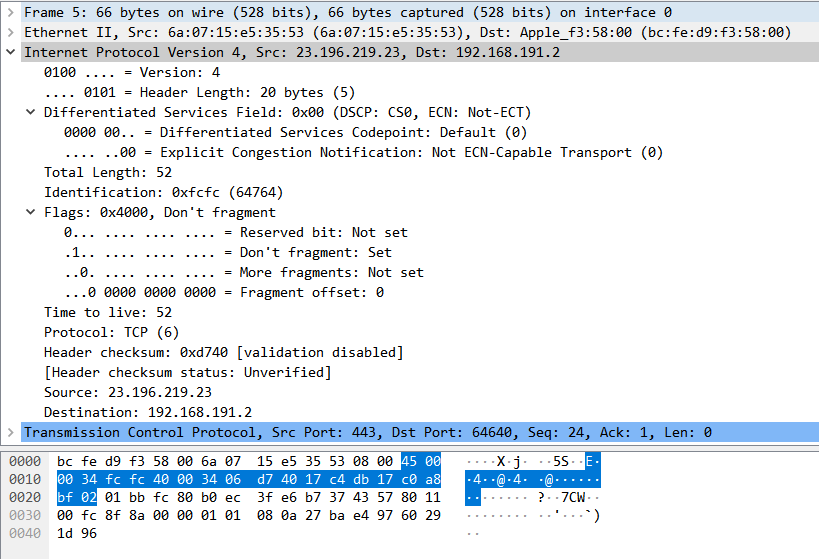
TCP/IP协议簇是一个四层协议系统，自底而上分别是网络接口层、网络层、传输层和应用层。每一层完成不同的功能，且通过若干协议来实现，上层协议使用下层协议提供的服务。



1. **IP协议 （Internet Protocol）（IPv4/IPv6）：**

TCP/IP协议的主要部分IP协议处于网络层。网络层的主要作用是“实现端对端的通信”。在网络世界里，所有的主机都具有“IP地址”。





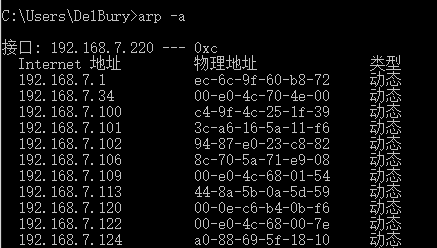
**MAC地址与IP地址**

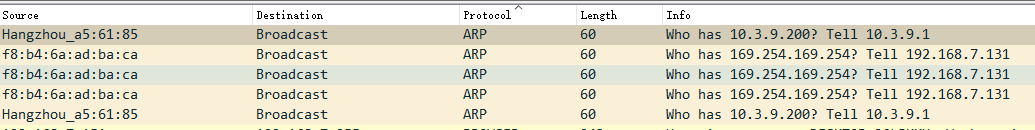
MAC地址（Media Access Control Address），直译为媒体访问控制地址，也称为局域网地址（LAN Address），以太网地址（Ethernet Address）或物理地址（Physical Address），它是一个用来确认网上设备位置的地址。在OSI模型中，第三层网络层负责IP地址，第二层数据链接层则负责MAC地址。MAC地址用于在网络中唯一标示一个网卡，一台设备若有一或多个网卡，则每个网卡都需要并会有一个唯一的MAC地址。

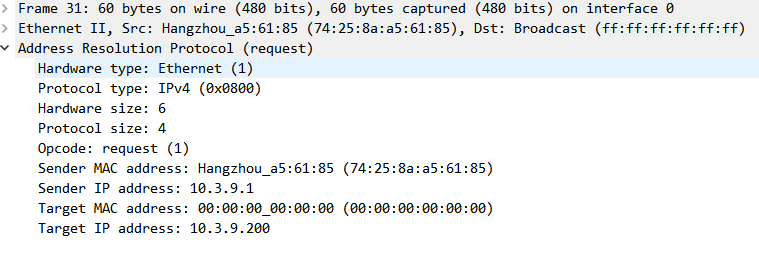
局域网采用了用MAC地址来标识具体用户的方法。具体实现：在交换机内部通过“表”的方式把MAC地址和IP地址一一对应，也就是所说的IP、MAC绑定。

具体的通信方式：接收过程，当有发给本地局域网内一台主机的数据包时，交换机接收下来，然后把数据包中的IP地址按照“表”中的对应关系映射成MAC地址，转发到对应的MAC地址的主机上，这样一来，即使某台主机盗用了这个IP地址，但由于他没有这个MAC地址，因此也不会收到数据包。

只有MAC地址也能够传输数据，前提是处于同一个局域网内，进行跨网络间的通信则需要IP地址与MAC地址

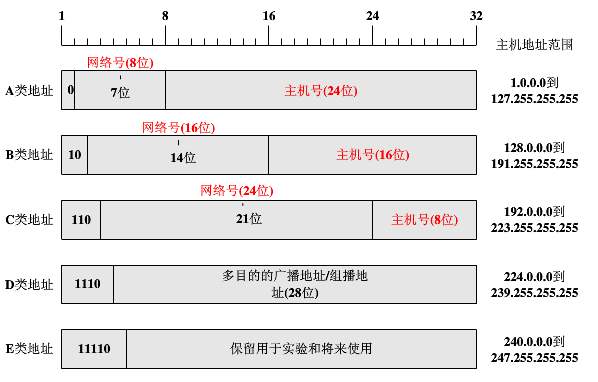






**IP分类：**

32位的IP地址被分成了以下下五类，又大大的减少了IP地址数量。A、B、C是有网络号和主机号组成的，使用于不同大小的网络规模。这样是为了区别是不是在同一个局域网内。



全0和全1的地址都保留不用

在IP地址3种主要类型里，各保留了3个区域作为私有地址，其地址范围如下：

A类地址：10.0.0.0～10.255.255.255

B类地址：172.16.0.0～172.31.255.255

C类地址：192.168.0.0～192.168.255.255

127.x.x.x段地址空间是被保留的回环地址（不会被发送到网络上）。也是本机地址，等效于localhost或本机IP。一般用于测试使用。实际，127.0.0.0 和 127.255.255.255 不可使用

**子网掩码：**

子网掩码(subnet mask)又叫网络掩码、地址掩码、子网络遮罩，它是一种用来指明一个IP地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机的位掩码。

子网掩码不能单独存在，它必须结合IP地址一起使用。子网掩码只有一个作用，就是将某个IP地址划分成网络地址和主机地址两部分。

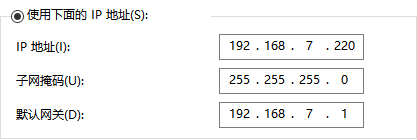
子网掩码是一个32位地址，用于屏蔽IP地址的一部分以区别网络标识和主机标识，并说明该IP地址是在局域网上，还是在远程网上。

子网掩码与IP地址按位与可得网络号

可以这么简单的理解：A主机要与B主机通信，A和B各自的IP地址与A主机的子网掩码进行And与运算，看得出的结果：

1、结果如果相同，则说明这两台主机是处于同一个网段，这样A可以通过ARP广播发现B的MAC地址，B也可以发现A的MAC地址来实现正常通信。

2、如果结果不同，ARP广播会在本地网关终结，这时候A会把发给B的数据包先发给本地网关，网关再根据B主机的IP地址来查询路由表，再将数据包继续传递转发，最终送达到目的地B。



**广播地址：**

在使用TCP/IP 协议的网络中，主机标识段host ID 为全1 的IP 地址为广播地址，广播的分组传送给host ID段所涉及的所有计算机。例如，对于10.1.1.0 （255.255.255.0 ）网段，其广播地址为10.1.1.255 （255 即为2 进制的11111111 ），当发出一个目的地址为10.1.1.255 的分组（封包）时，它将被分发给该网段上的所有计算机。

1. **TCP协议 （Transmission Control Protocol）：**

**TCP和UDP：**

TCP和UDP都是传输层的协议，基于IP协议，区别如下：

UDP（User Datagram Protocol用户数据报协议）：

1. 无连接
2. 尽最大努力交付
3. 单个数据报
4. 会丢包
5. 会乱序
6. 开销小

TCP（Transmission Control Protocol传输控制协议）：

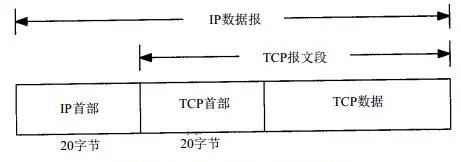
1. 面向连接
2. 可靠传输
3. 面向字节流
4. 有序传输
5. 开销大

TCP用于实现可靠传输的情况，对网络拥堵有较高的要求的情况

UDP用于高速传输和实时性较高的场合（如即时通信，网络游戏等），出现丢包，会出现短暂的卡顿

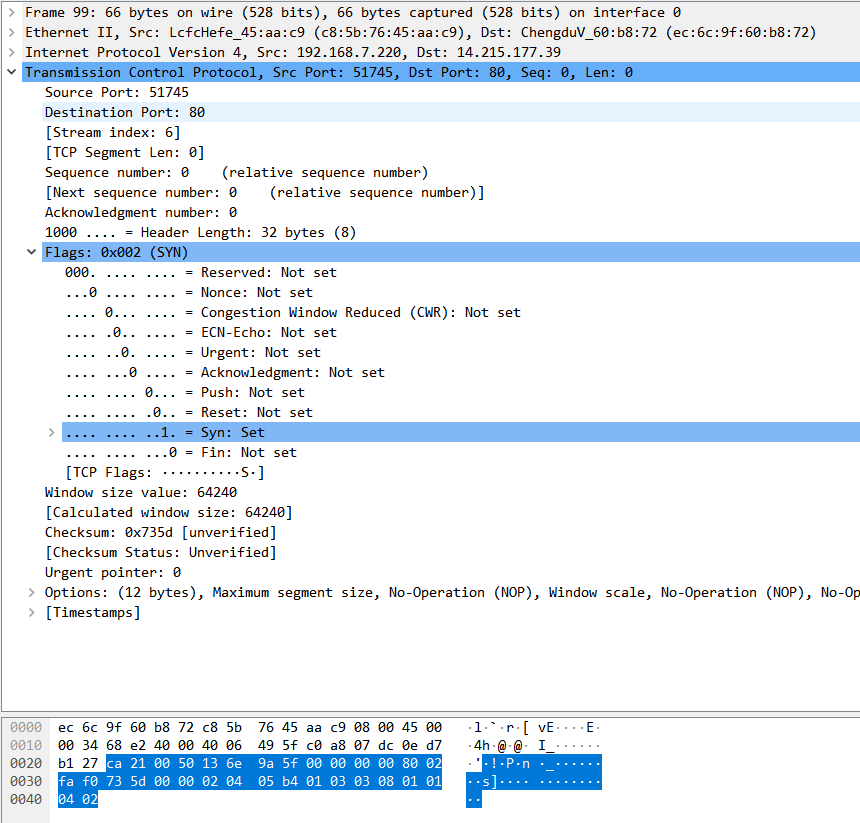
**TCP报文数据格式：**

TCP数据封装在一个IP数据报中：



下图是TCP报文数据格式。TCP首部如果不计选项和填充字段，它通常是20个字节。





**sequence number：**表示的是我方（发送方）这边，这个packet的数据部分的第一位应该在整个data stream中所在的位置。（注意这里使用的是“应该”。因为对于没有数据的传输，如ACK，虽然它有一个seq，但是这次传输在整个data stream中是不占位置的。所以下一个实际有数据的传输，会依旧从上一次发送ACK的数据包的seq开始）

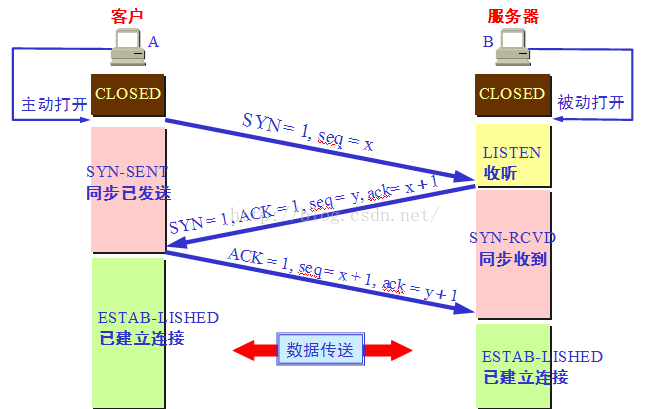
**acknowledge number：**表示的是期望的对方（接收方）的下一次sequence number是多少。

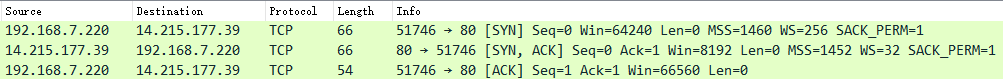
注意，SYN/FIN的传输虽然没有data，但是会让下一次传输的packet seq增加一，但是，ACK的传输，不会让下一次的传输packet加一

**TCP协议的三次握手：**

TCP提供可靠的、面向连接的数据传输服务。使用TCP通信之前，需要进行“三次握手”建立连接，通信结束后还要使用“四次挥手”断开连接。

三次握手过程：





第一次握手：建立连接时，客户端发送syn包（syn=x）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

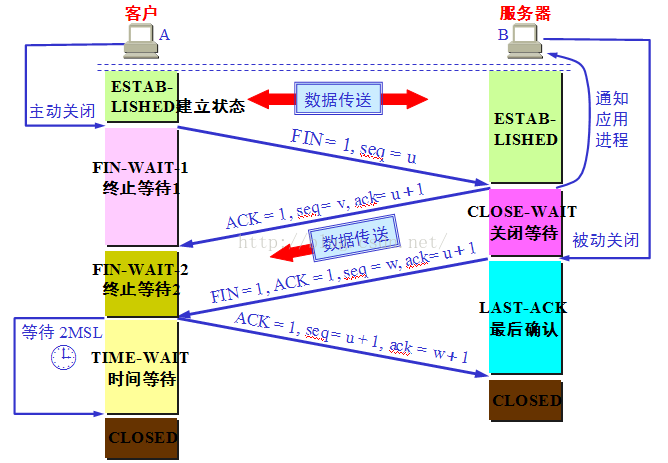
第二次握手：服务器收到syn包，必须确认客户的SYN（ack=x+1），同时自己也发送一个SYN包（syn=y），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

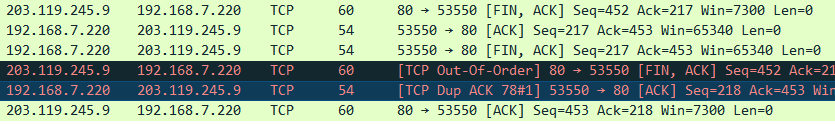
第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=y+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

为什么需要三次握手？

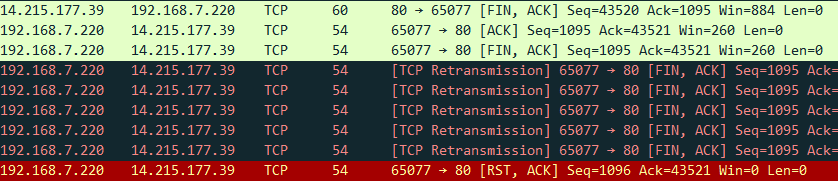
首先我们要知道信道是不可靠的，但是我们要建立可靠的连接发送可靠的数据，也就是数据传输是需要可靠的。在这个时候三次握手是一个理论上的最小值，并不是说是tcp协议要求的，而是为了满足在不可靠的信道上传输可靠的数据所要求的。三次握手的目的是“为了防止已经失效的连接请求报文段突然又传到服务端，因而产生错误”。

**四次挥手：**



发起断开连接请求可以是客户端也可以是服务器，如下图为淘宝的TCP关闭过程（connection: close）:

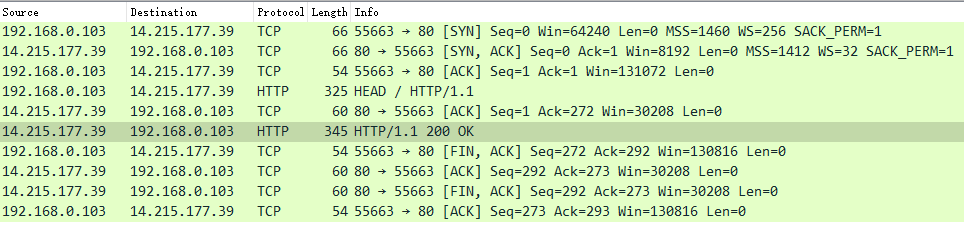
下图为百度的TCP关闭过程（connection: keep-alive）：



百度的TCP关闭过程（connection: close）：



HEAD请求，一次完整的建立连接和关闭连接



1. **HTTP协议 （Hypertext Transfer Protocol）：**

超文本传输协议（HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。

**HTTP版本：**

HTTP/0.9：

HTTP协议的最初版本，功能简陋，仅支持请求方式GET，并且仅能请求访问HTML格式的资源。

HTTP/1.0：

在0.9版本上做了进步，增加了请求方式POST和HEAD

根据Content-Type可以支持多种数据格式

开始支持cache

每次TCP连接只能发送一个请求，当服务器响应后就会关闭这次连接，下一个请求需要再次建立TCP连接，就是不支持keepalive

HTTP/1.1：

引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive。解决了1.0版本的keepalive问题，1.1版本加入了持久连接，一个TCP连接可以允许多个HTTP请求

加入了管道机制，在同一个TCP连接里，允许多个请求同时发送，增加了并发性，进一步改善了HTTP协议的效率

新增了请求方式PUT、PATCH、OPTIONS、DELETE等

客户端请求的头信息新增了Host字段，用来指定服务器的域名

加入了一个新的状态码100（Continue）、支持只发送header信息、身份认证机制、支持传送内容的一部分、支持文件断点续传（range）、新增了24个错误状态响应码

HTTP/2.0：

新增二进制协议、多工、数据流、头信息压缩、服务器推送等功能

**HTTP协议概述：**

通常，由HTTP客户端发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。HTTP服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。

基于 请求-响应 的模式、无状态保存（cookie）

**工作原理：**

在浏览器地址栏键入URL，按下回车之后会经历以下流程：

1.浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;

2.解析出 IP 地址后，根据该 IP 地址和端口号，和服务器建立TCP连接;

3.浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求;

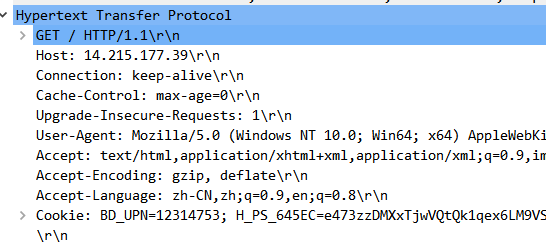
4.服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;

5.释放 TCP连接;

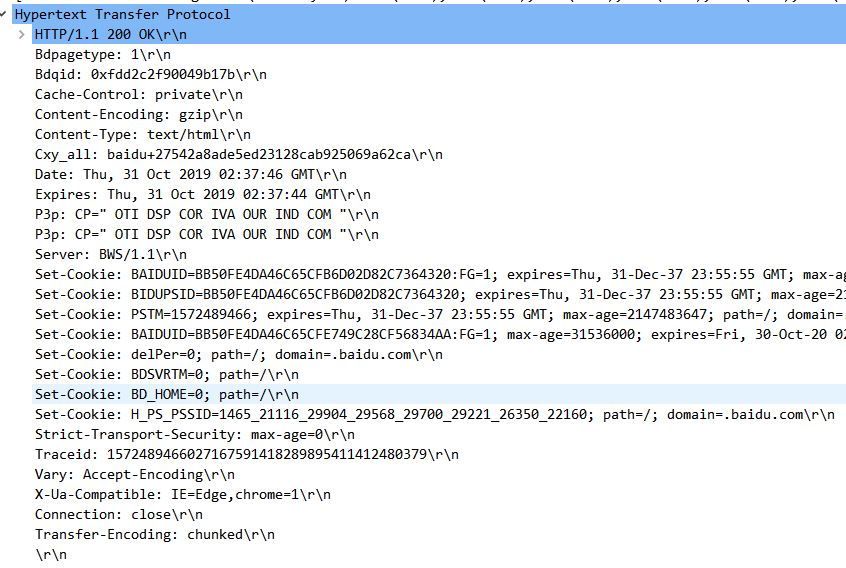
6.浏览器将该 html 文本并显示内容;

下图为百度的HTTP Header信息：

请求头：



响应头：



**常用请求方法：**

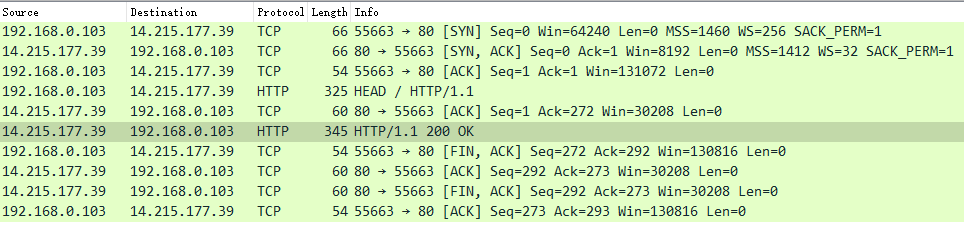
**GET**：用来请求指定资源的读取，参数只能放在URL中

**POST**：向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有

**PUT**：把一个资源存放在指定的位置上。本质上来讲， PUT和POST极为相似，都是向服务器发送数据，但它们之间有一个重要区别，PUT通常指定了资源的存放位置，而POST则没有，POST的数据存放位置由服务器自己决定

**DELETE**：删除某一个资源

**HEAD**：HEAD跟GET相似，不过服务端接收到HEAD请求时只返回响应头，不发送响应内容。所以，如果只需要查看某个页面的状态时，用HEAD更高效，因为省去了传输页面内容的时间



**OPTIONS**：OPTIONS方法是用于请求获得由Request-URI标识的资源在请求/响应的通信过程中可以使用的功能选项。通过这个方法，客户端可以在采取具体资源请求之前，决定对该资源采取何种必要措施，或者了解服务器的性能。说白了就是，在发生正式的请求之前，先进行一次预检请求。看服务端返回一些信息，浏览器拿到之后，看后台是否允许进行访问。

产生options请求的原因包括以下几条：

1、获取目的资源所支持的通信方式

2、跨域请求中，options请求是浏览器自发起的preflight request(预检请求)，以检测实际请求是否可以被浏览器接受。

preflight request请求报文中有两个需要关注的首部字段：

（1）Access-Control-Request-Method：告知服务器实际请求所使用的HTTP方法

（2）Access-Control-Request-Headers：告知服务器实际请求所携带的自定义首部字段

同时浏览器也会添加origin header,告知服务器实际请求的客户端的地址。服务器基于从预检请求获得的信息来判断，是否接受接下来的实际请求。

当跨域请求是简单请求时不会进行preflight request,只有复杂请求才会进行preflight request。跨域请求分两种：简单请求、复杂请求；

符合以下任一情况的就是复杂请求：

1.使用方法put或者delete;

2.发送json格式的数据（content-type: application/json）

3.请求中带有自定义头部；

**CONNECT**：connect的作用就是将服务器作为代理，让服务器代替用户去访问其他网页，网页开发中一般不会用到

**常见请求头：**

Referer (referrer)：表示这个请求是从哪个url访问的，一般用来防止外链，流量统计等

Accept：

例如：Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8

告诉服务端,该请求所能支持的响应MIME 类型（MIME格式：大类型/小类型[;参数]）（q是权重系数，范围 0 =< q <= 1，q 值越大，请求越倾向于获得其“;”之前的类型表示的内容。若没有指定q值，则默认为1，按从左到右排序顺序；若被赋值为0，则用于表示浏览器不接受此内容类型。）

Cookie：传送客户端的cookie值

User-Agent：浏览器通知服务器，客户端浏览器与操作系统相关信息

Connection：表示客户端与服务连接类型；Keep-Alive表示持久连接，close端连接

Host：请求的服务器主机名

Content-Length：请求体的长度

Content-Type：请求的与实体对应的MIME信息。如果是post请求,会有这个头,默认值为application/x-www-form-urlencoded，表示请求体内容使用url编码

Accept-Encoding：浏览器通知服务器，浏览器支持的数据压缩格式。

Accept-Language：浏览器通知服务器，浏览器支持的语言。

Cache-Control / Pragma / Expires：指定请求和响应遵循的缓存机制。

Range：

例如：Range: bytes=start-end

请求资源的部分内容（不包括响应头的大小），单位是byte，即字节，从0开始

**常见响应头：**

Last-Modified：表示资源最近修改的时间

ETag：这个响应头表示资源版本的标识符，通常是消息摘要(类似MD5一样）

Location：指定响应的路径，需要与状态码302配合使用，完成跳转

Content-Type：响应正文的类型（MIME类型）

Content-Disposition：通过浏览器以下载方式解析正文

Set-Cookie：与会话相关技术。服务器向浏览器写入cookie

Server：指的是服务器名称

Access-Control-Allow-\*：跨域相关的头

1. **HTTPS协议**

相较于HTTP协议以明文的方式传输数据，HTTPS（HTTP over SSL/TLS）协议通过SSL/TLS对数据进行加密，保护数据的安全性。有以下特点：

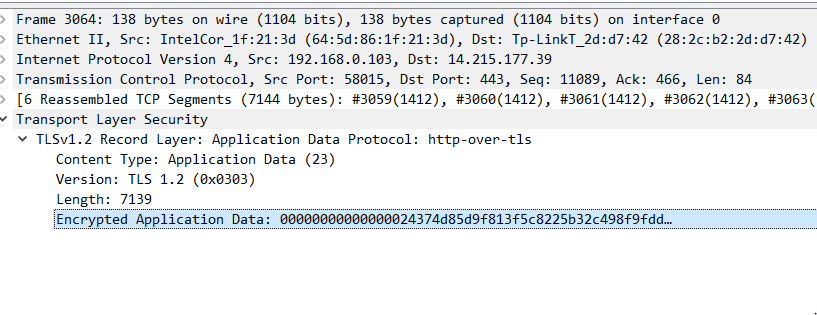
1.内容加密：采用混合加密技术，中间者无法直接查看明文内容

2.验证身份：通过证书认证客户端访问的是自己的服务器

3.保护数据完整性：防止传输的内容被中间人冒充或者篡改

**SSL/TLS概念：**

SSL(Secure Socket Layer 安全套接层)是基于HTTPS下的一个协议加密层，最初是由网景公司（Netscape）研发，后被IETF（The Internet Engineering Task Force - 互联网工程任务组）标准化后写入RFC （Request For Comments 请求注释）



SSL由从前的网景公司开发，有1/2/3三个版本，但现在只使用版本3

TLS是SSL的标准化后的产物，有1.0/1.1/1.2三个版本

TLS1.0和SSL3.0几乎没有区别，事实上我们现在用的都是TLS，但因为历史上习惯了SSL这个称呼

SSL/TLS解决的其实是明文传输的几个安全风险：

1）窃听风险

2）篡改风险

3）冒充风险

相应的，SSL/TLS是采用这么几个方式进行应对的：

1）加密：非对称加密+对称加密，主要解决的是窃听风险

2）校验：数字签名，主要解决的是篡改风险

3）证书：数字证书，主要解决的是冒充风险

**基本的运行过程：**

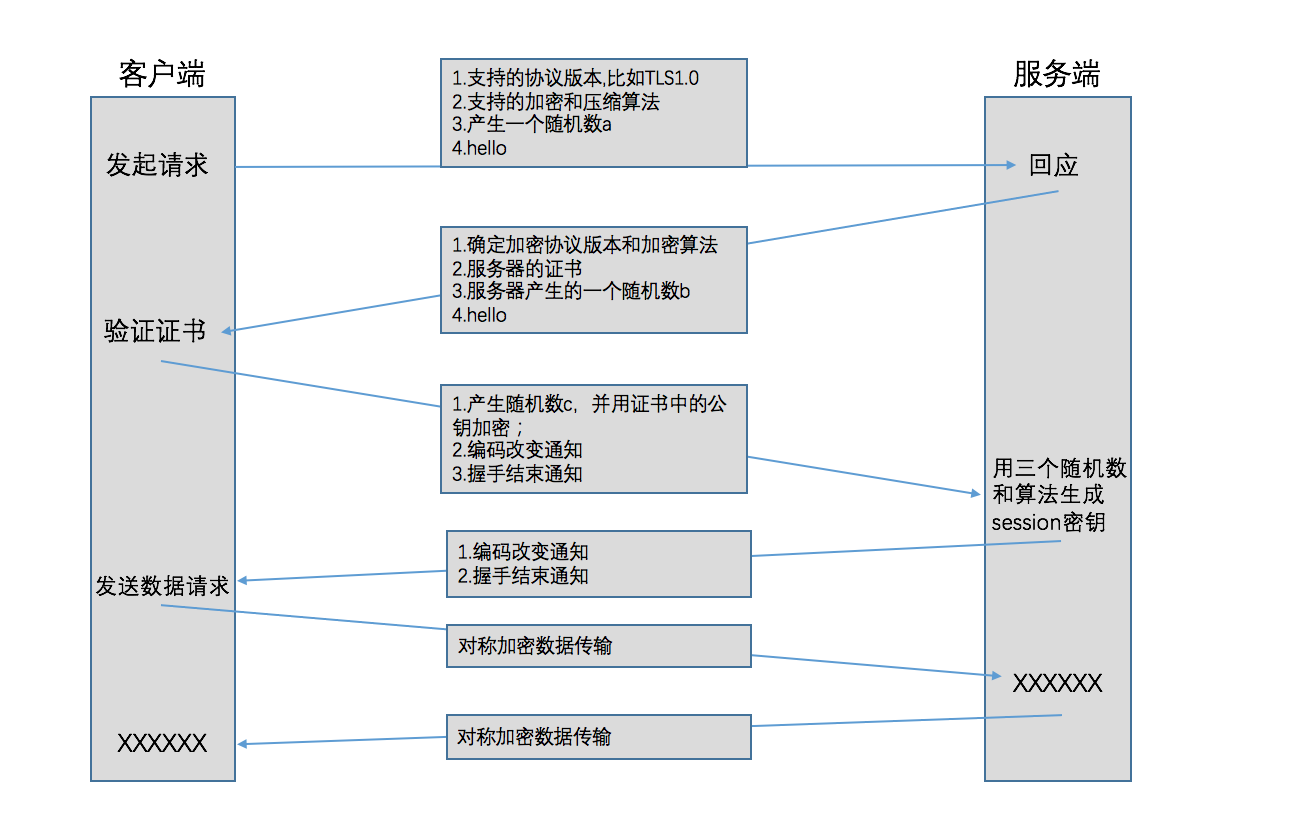
（1） 客户端向服务器端索要并验证公钥。

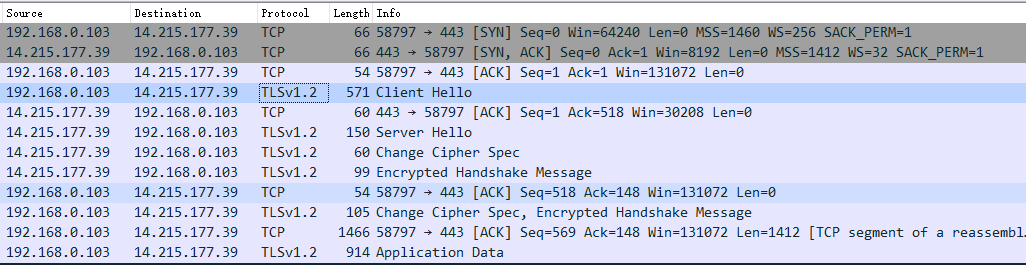
（2） 双方协商生成"对话密钥"。

（3） 双方采用"对话密钥"进行加密通信。

前两步，又称为"握手阶段"（handshake）。

SSL协议在握手阶段使用的是非对称加密，在传输阶段使用的是对称加密，也就是说在SSL上传送的数据是使用对称密钥加密的





**非对称加密和对称加密：**

对称加密：加密和解密的密钥一样，比如用123加密就是用123解密

非对称加密：把密钥分为公钥和私钥，公钥是公开的所有人都可以认领，私钥是保密的只有一个人知道。公钥和私钥是配对关系，公钥加密就用私钥解密，私钥加密就用公钥解密

**数字签名：**

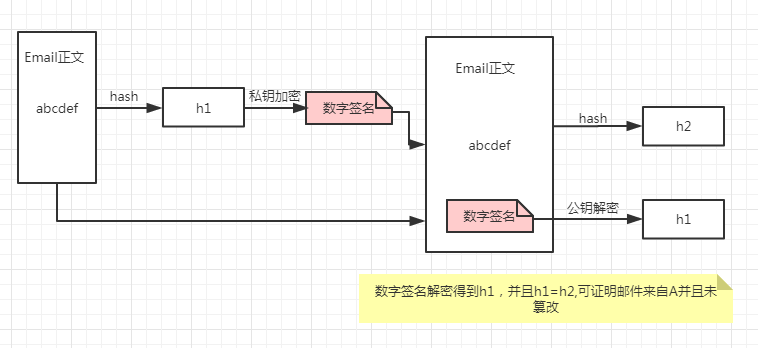
对文件本身加密可能是个耗时过程，比如加密的文件足够大，那么私钥加密整个文件以及拿到文件后的解密无疑是巨大的开销。

数字签名可以解决这个问题：

A先对这个Email执行哈希运算得到hash值简称“摘要”，取名h1

然后用自己的私钥对摘要加密，生成的东西叫“数字签名”

把数字签名加在Email正文后面，一起发送给B

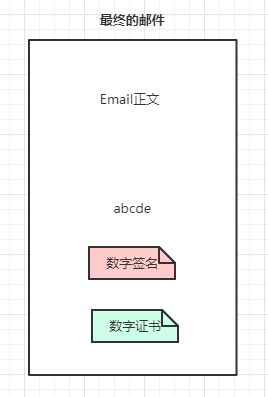


**数字证书：**

作用是验证公钥的真实性

1.数字证书的生成：首先A去找"证书中心"（certificate authority，简称CA），为公钥做认证。证书中心用自己的私钥，对A的公钥和一些相关信息一起加密，生成"数字证书"（Digital Certificate）：

2. A在邮件正文下方除了数字签名，另外加上这张数字证书



3. B收到Email后用CA的公钥解密这份数字证书，拿到A的公钥，然后验证数字签名，证明公钥的真实性

HTTPS实例：

1. 首先，客户端向服务器发出加密请求。
2. 服务器用自己的私钥加密网页以后，连同本身的数字证书，一起发送给客户端。
3. 客户端（浏览器）的“证书管理器”，有“受信任的根证书颁发机构”列表。客户端会根据这张列表，查看解开数字证书的公钥是否在列表之内。
4. 如果数字证书记载的网址，与你正在浏览的网址不一致，就说明这张证书可能被冒用，浏览器会发出警告。
5. 如果这张数字证书不是由受信任的机构颁发的，浏览器会发出另一种警告。
6. 如果数字证书是可靠的，客户端就可以使用证书中的服务器公钥，对信息进行加密，然后与服务器交换加密信息。