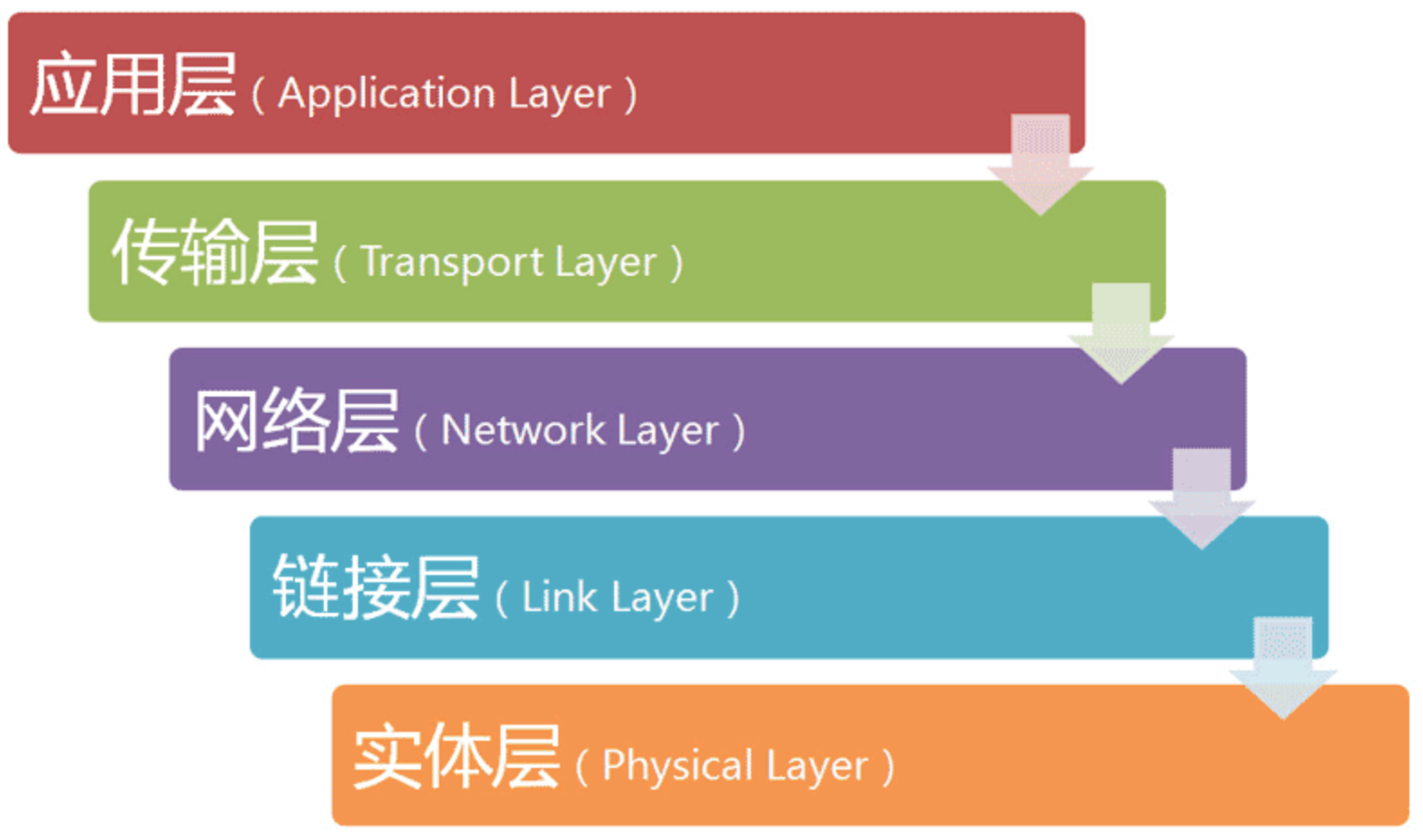
**互联网协议**

**一、概念**

互联网的实现，分成好几层。每一层都有自己的功能，就像建筑物一样，每一层都靠下一层支持。

用户接触到的，只是最上面的一层，根本没有感觉到下面的层。要理解互联网，必须从最下层开始，自下而上理解每一层的功能。



越下面的层，越靠近硬件；越上面的层，越靠近用户。

每一层都是为了完成一种功能。为了实现这些功能，就需要大家都遵守共同的规则。大家都遵守的规则，就叫做"协议"（protocol）。

**二、实体层**

要组成互联网必须得把电脑连接起来，可以通过光缆、电缆、双绞线、无线电波等多种方式。这就叫做“实体层”。它就是把电脑连接起来的物理手段。它主要规定了网络的一些电气特性，作用是负责传送0和1的电信号。

**三、链接层**

3.1 定义

单纯的0和1没有任何意义，必须规定解读方式：多少个电信号算一组？每个信号位有何意义？

这就是"链接层"的功能，它在"实体层"的上方，确定了0和1的分组方式。

3.2 以太网协议

早期的时候，每家公司都有自己的电信号分组方式。逐渐地，一种叫做["以太网"](http://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E7%BD%91)（Ethernet）的协议，占据了主导地位。

以太网规定，一组电信号构成一个数据包，叫做"帧"（Frame）。每一帧分成两个部分：标头（Head）和数据（Data）。



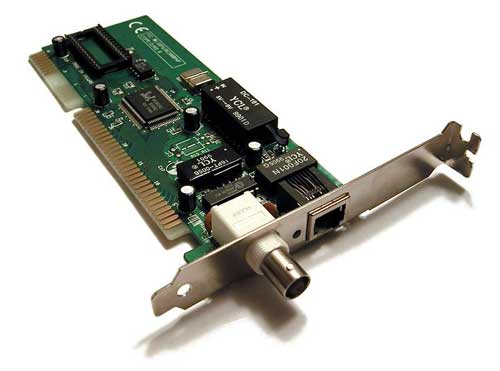
"标头"包含数据包的一些说明项，比如发送者、接收者、数据类型等等；"数据"则是数据包的具体内容。

"标头"的长度，固定为18字节。"数据"的长度，最短为46字节，最长为1500字节。因此，整个"帧"最短为64字节，最长为1518字节。如果数据很长，就必须分割成多个帧进行发送。

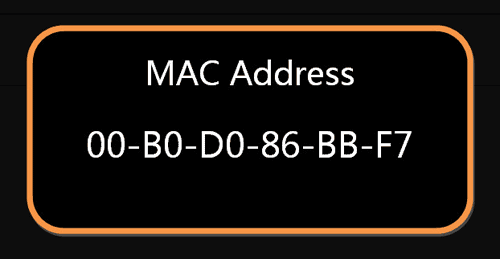
3.3 MAC地址

上面提到，以太网数据包的"标头"，包含了发送者和接受者的信息。那么，发送者和接受者是如何标识呢？

以太网规定，连入网络的所有设备，都必须具有"网卡"接口。数据包必须是从一块网卡，传送到另一块网卡。网卡的地址，就是数据包的发送地址和接收地址，这叫做MAC地址。



每块网卡出厂的时候，都有一个全世界独一无二的MAC地址，长度是48个二进制位，通常用12个十六进制数表示。



前6个十六进制数是厂商编号，后6个是该厂商的网卡流水号。有了MAC地址，就可以定位网卡和数据包的路径了。

3.4 广播

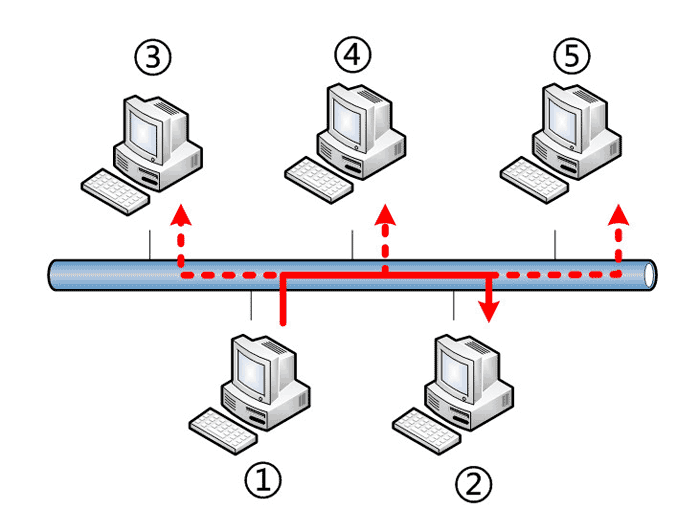
定义地址只是第一步，后面还有更多的步骤。

首先，一块网卡怎么会知道另一块网卡的MAC地址？

回答是有一种ARP协议，可以解决这个问题。这个留到后面介绍，这里只需要知道，以太网数据包必须知道接收方的MAC地址，然后才能发送。

其次，就算有了MAC地址，系统怎样才能把数据包准确送到接收方？

回答是以太网采用了一种很"原始"的方式，它不是把数据包准确送到接收方，而是向本网络内所有计算机发送，让每台计算机自己判断，是否为接收方。



上图中，1号计算机向2号计算机发送一个数据包，同一个子网络的3号、4号、5号计算机都会收到这个包。它们读取这个包的"标头"，找到接收方的MAC地址，然后与自身的MAC地址相比较，如果两者相同，就接受这个包，做进一步处理，否则就丢弃这个包。这种发送方式就叫做"广播"（broadcasting）。

有了数据包的定义、网卡的MAC地址、广播的发送方式，"链接层"就可以在多台计算机之间传送数据了。

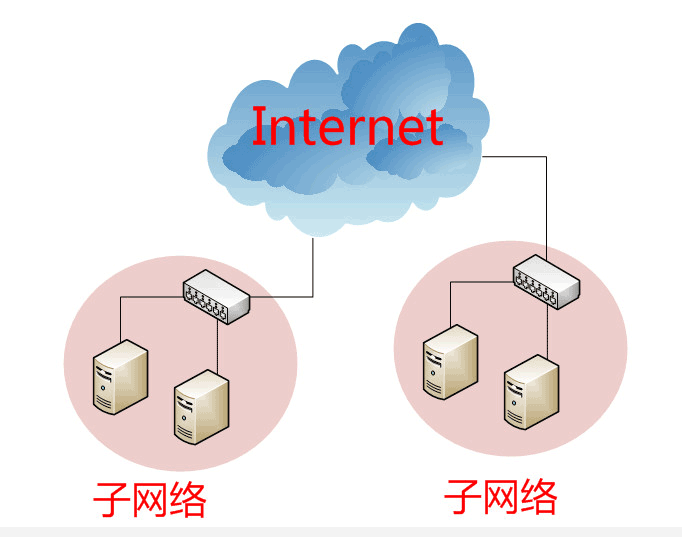
**四、网络层**

4.1 网络层的由来

以太网协议，依靠MAC地址发送数据。理论上，单单依靠MAC地址，上海的网卡就可以找到洛杉矶的网卡了，技术上是可以实现的。

但是，这样做有一个重大的缺点。以太网采用广播方式发送数据包，所有成员人手一"包"，不仅效率低，而且局限在发送者所在的子网络。也就是说，如果两台计算机不在同一个子网络，广播是传不过去的。这种设计是合理的，否则互联网上每一台计算机都会收到所有包，那会引起灾难。

互联网是无数子网络共同组成的一个巨型网络，很难想象上海和洛杉矶的电脑会在同一个子网络，这几乎是不可能的。



因此，必须找到一种方法，能够区分哪些MAC地址属于同一个子网络，哪些不是。如果是同一个子网络，就采用广播方式发送，否则就采用"路由"方式发送。（"路由"的意思，就是指如何向不同的子网络分发数据包，这是一个很大的主题，本文不涉及。）遗憾的是，MAC地址本身无法做到这一点。它只与厂商有关，与所处网络无关。

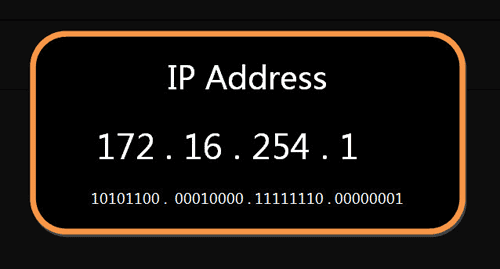
这就导致了"网络层"的诞生。它的作用是引进一套新的地址，使得我们能够区分不同的计算机是否属于同一个子网络。这套地址就叫做"网络地址"，简称"网址"。

于是，"网络层"出现以后，每台计算机有了两种地址，一种是MAC地址，另一种是网络地址。两种地址之间没有任何联系，MAC地址是绑定在网卡上的，网络地址则是管理员分配的，它们只是随机组合在一起。

网络地址帮助我们确定计算机所在的子网络，MAC地址则将数据包送到该子网络中的目标网卡。因此，从逻辑上可以推断，必定是先处理网络地址，然后再处理MAC地址。

4.2 IP协议

规定网络地址的协议，叫做IP协议。它所定义的地址，就被称为IP地址。目前，广泛采用的是IP协议第四版，简称IPv4。这个版本规定，网络地址由32个二进制位组成。



习惯上，我们用分成四段的十进制数表示IP地址，从0.0.0.0一直到255.255.255.255。

互联网上的每一台计算机，都会分配到一个IP地址。这个地址分成两个部分，前一部分代表网络，后一部分代表主机。比如，IP地址172.16.254.1，这是一个32位的地址，假定它的网络部分是前24位（172.16.254），那么主机部分就是后8位（最后的那个1）。处于同一个子网络的电脑，它们IP地址的网络部分必定是相同的，也就是说172.16.254.2应该与172.16.254.1处在同一个子网络。

但是，问题在于单单从IP地址，我们无法判断网络部分。还是以172.16.254.1为例，它的网络部分，到底是前24位，还是前16位，甚至前28位，从IP地址上是看不出来的。

那么，怎样才能从IP地址，判断两台计算机是否属于同一个子网络呢？这就要用到另一个参数"子网掩码"（subnet mask）。

所谓"子网掩码"，就是表示子网络特征的一个参数。它在形式上等同于IP地址，也是一个32位二进制数字，它的网络部分全部为1，主机部分全部为0。比如，IP地址172.16.254.1，如果已知网络部分是前24位，主机部分是后8位，那么子网络掩码就是11111111.11111111.11111111.00000000，写成十进制就是255.255.255.0。

知道"子网掩码"，我们就能判断，任意两个IP地址是否处在同一个子网络。方法是将两个IP地址与子网掩码分别进行AND运算（两个数位都为1，运算结果为1，否则为0），然后比较结果是否相同，如果是的话，就表明它们在同一个子网络中，否则就不是。

比如，已知IP地址172.16.254.1和172.16.254.233的子网掩码都是255.255.255.0，请问它们是否在同一个子网络？两者与子网掩码分别进行AND运算，结果都是172.16.254.0，因此它们在同一个子网络。

**总结一下，IP协议的作用主要有两个，一个是为每一台计算机分配IP地址，另一个是确定哪些地址在同一个子网络。**

4.3 IP数据包

根据IP协议发送的数据，就叫做IP数据包。不难想象，其中必定包括IP地址信息。但是前面说过，以太网数据包只包含MAC地址，并没有IP地址的栏位。那么是否需要修改数据定义，再添加一个栏位呢？

回答是不需要，我们可以把IP数据包直接放进以太网数据包的"数据"部分，因此完全不用修改以太网的规格。这就是互联网分层结构的好处：上层的变动完全不涉及下层的结构。具体来说，IP数据包也分为"标头"和"数据"两个部分。



"标头"部分主要包括版本、长度、IP地址等信息，"数据"部分则是IP数据包的具体内容。它放进以太网数据包后，以太网数据包就变成了下面这样。



IP数据包的"标头"部分的长度为20到60字节，整个数据包的总长度最大为65,535字节。因此，理论上，一个IP数据包的"数据"部分，最长为65,515字节。前面说过，以太网数据包的"数据"部分，最长只有1500字节。因此，如果IP数据包超过了1500字节，它就需要分割成几个以太网数据包，分开发送了。

**4.4 ARP协议**

关于"网络层"，还有最后一点需要说明。因为IP数据包是放在以太网数据包里发送的，所以我们必须同时知道两个地址，一个是对方的MAC地址，另一个是对方的IP地址。通常情况下，对方的IP地址是已知的（后文会解释），但是我们不知道它的MAC地址。所以，我们需要一种机制，能够从IP地址得到MAC地址。这里又可以分成两种情况。

第一种情况，如果两台主机不在同一个子网络，那么事实上没有办法得到对方的MAC地址，只能把数据包传送到两个子网络连接处的"网关"（gateway），让网关去处理。

第二种情况，如果两台主机在同一个子网络，那么我们可以用ARP协议，得到对方的MAC地址。ARP协议也是发出一个数据包（包含在以太网数据包中），其中包含它所要查询主机的IP地址，在对方的MAC地址这一栏，填的是FF:FF:FF:FF:FF:FF，表示这是一个"广播"地址。它所在子网络的每一台主机，都会收到这个数据包，从中取出IP地址，与自身的IP地址进行比较。如果两者相同，都做出回复，向对方报告自己的MAC地址，否则就丢弃这个包。

总之，有了ARP协议之后，我们就可以得到同一个子网络内的主机MAC地址，可以把数据包发送到任意一台主机之上了。

**五、传输层**

5.1 传输层的由来

有了MAC地址和IP地址，我们已经可以在互联网上任意两台主机上建立通信。

接下来的问题是，同一台主机上有许多程序都需要用到网络，比如，你一边浏览网页，一边与朋友在线聊天。当一个数据包从互联网上发来的时候，你怎么知道，它是表示网页的内容，还是表示在线聊天的内容？

也就是说，我们还需要一个参数，表示这个数据包到底供哪个程序（进程）使用。这个参数就叫做"端口"（port），它其实是每一个使用网卡的程序的编号。每个数据包都发到主机的特定端口，所以不同的程序就能取到自己所需要的数据。

"端口"是0到65535之间的一个整数，正好16个二进制位。0到1023的端口被系统占用，用户只能选用大于1023的端口。不管是浏览网页还是在线聊天，应用程序会随机选用一个端口，然后与服务器的相应端口联系。

"传输层"的功能，就是建立"端口到端口"的通信。相比之下，"网络层"的功能是建立"主机到主机"的通信。只要确定主机和端口，我们就能实现程序之间的交流。因此，Unix系统就把主机+端口，叫做"套接字"（socket）。有了它，就可以进行网络应用程序开发了。

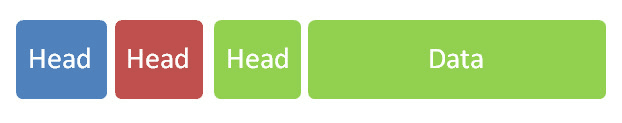
5.2 UDP协议 ->传输协议

现在，我们必须在数据包中加入端口信息，这就需要新的协议。最简单的实现叫做UDP协议，它的格式几乎就是在数据前面，加上端口号。

UDP数据包，也是由"标头"和"数据"两部分组成。



"标头"部分主要定义了发出端口和接收端口，"数据"部分就是具体的内容。然后，把整个UDP数据包放入IP数据包的"数据"部分，而前面说过，IP数据包又是放在以太网数据包之中的，所以整个以太网数据包现在变成了下面这样：



UDP数据包非常简单，"标头"部分一共只有8个字节，总长度不超过65,535字节，正好放进一个IP数据包。

5.3 TCP协议 –> 传输协议

UDP协议的优点是比较简单，容易实现，但是缺点是可靠性较差，一旦数据包发出，无法知道对方是否收到。

为了解决这个问题，提高网络可靠性，TCP协议就诞生了。这个协议非常复杂，但可以近似认为，它就是有确认机制的UDP协议，每发出一个数据包都要求确认。如果有一个数据包遗失，就收不到确认，发出方就知道有必要重发这个数据包了。

因此，TCP协议能够确保数据不会遗失。它的缺点是过程复杂、实现困难、消耗较多的资源。

TCP数据包和UDP数据包一样，都是内嵌在IP数据包的"数据"部分。TCP数据包没有长度限制，理论上可以无限长，但是为了保证网络的效率，通常TCP数据包的长度不会超过IP数据包的长度，以确保单个TCP数据包不必再分割。

**六、应用层**

应用程序收到"传输层"的数据，接下来就要进行解读。由于互联网是开放架构，数据来源五花八门，必须事先规定好格式，否则根本无法解读。

"应用层"的作用，就是规定应用程序的数据格式。

举例来说，TCP协议可以为各种各样的程序传递数据，比如Email、WWW、FTP等等。那么，必须有不同协议规定电子邮件、网页、FTP数据的格式，这些应用程序协议就构成了"应用层"。

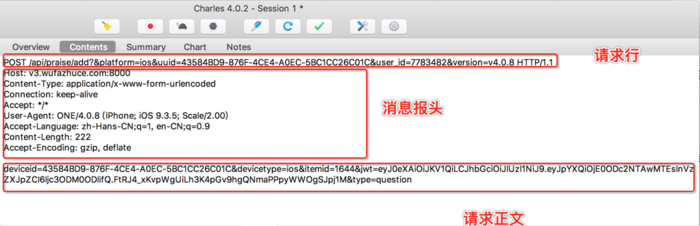
这是最高的一层，直接面对用户。它的数据就放在TCP数据包的"数据"部分。因此，现在的以太网的数据包就变成下面这样。

6.1 HTTP协议->超文本传输协议

HTTP (全称 Hyper Text Transfer Protocol )，一般称为超文本传输协议，也是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计 HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。一种详细规定了浏览器和万维网服务器之间互相通信的规则。

6.1.1 网络请求

由三个部分组成 ，请求行(包含:请求方式、请求的URL、协议的版本、末尾换行符)、消息报头(包含消息的长度、消息的类型、语言等)、请求正文（包含请求的参数）。



6.1.2 数据响应

由三部分组成：状态行（包含状态码、服务器http协议的版本、状态码的文本描述、末尾换行符），消息报头、响应正文(返回的内容)

常见状态代码 状态描述 说明：

200 OK //客户端请求成功

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

6.2 HTTPS协议

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，是一个安全通信通道，用于在客户计算机和服务器之间交换信息。它使用安全套接字层(SSL)进行信息交换，简单来说它是HTTP的安全版。

a. **Http与Https的区别**

简单的讲,Http协议的客户端和服务端通信采用的是明文(post请求也只是将数据放在header中,但依然是明文传输。而Hppts协议的客户端和服务端间通信采用的秘文传输（对称加密和非对称加密相结合）。Https=http+SSL/TLS(传输层安全协议)

b. **HTTPS是如何保障安全的**

HTTPS其实就是**secure http**的意思,也就是HTTP的安全升级版。稍微了解网络基础的同学都知道，HTTP是应用层协议，位于HTTP协议之下是传输协议TCP。TCP负责传输，HTTP则定义了数据如何进行包装。

**传输加密的流程**:原先是应用层将数据直接给到TCP进行传输，现在改成应用层将数据给到TLS/SSL(传输层安全协议)，将数据加密后，再给到TCP进行传输。这个是非对称加密。（对称加密是：加解密都是一样的密码;非对称加密：加解密使用不同的秘钥）

c. **HTTPS是如何加密数据的**

1. 握手认证:

客户端发起一次请求后,服务端返回数字证书,客户端接受到后,生成一个对称密钥,再用服务器下发证书中的**公钥对其加密**并发送,服务端接受到客户端发送的加密后的对称密钥后**用自己的私钥解密**,获得**对称密钥**。到此服务端和客户端都拥有了可以用来加密数据的对称密钥，并且可以保证该密钥没有泄漏。

2. 传输阶段

客户端和服务端用上面得到的对称密钥将数据加密后正常传输。

d. **HTTPS加解密全过程**

**1. 客户端发起HTTPS请求**

这个没什么好说的，就是用户在浏览器里输入一个https网址，然后连接到server的443端口。

**2. 服务端的配置**

采用HTTPS协议的服务器必须要有一套数字证书，可以自己制作，也可以向组织申请。区别就是自己颁发的证书需要客户端验证通过，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面(startssl就是个不错的选择，有1年的免费服务)。这套证书其实就是一对公钥和私钥。如果对公钥和私钥不太理解，可以想象成一把钥匙和一个锁头，只是全世界只有你一个人有这把钥匙，你可以把锁头给别人，别人可以用这个锁把重要的东西锁起来，然后发给你，因为只有你一个人有这把钥匙，所以只有你才能看到被这把锁锁起来的东西。

**3. 传送证书**

这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构，过期时间等等。

**4. 客户端解析证书**

这部分工作是有客户端的TLS来完成的，首先会验证公钥是否有效，比如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告框，提示证书存在问题。如果证书没有问题，那么就生成一个随机值。然后用证书对该随机值进行加密。就好像上面说的，把随机值用锁头锁起来，这样除非有钥匙，不然看不到被锁住的内容。

**5. 传送加密信息**

这部分传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，以后客户端和服务端的通信就可以通过这个随机值来进行加密解密了。

**6. 服务段解密信息**

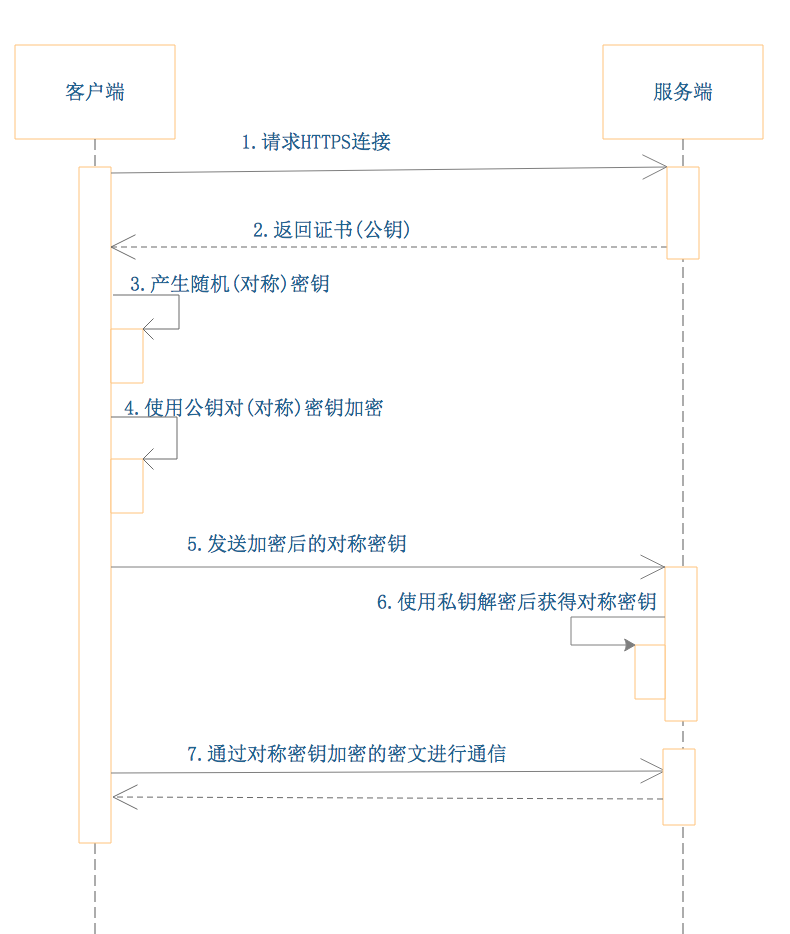
服务端用私钥解密后，得到了客户端传过来的随机值(私钥)，然后把内容通过该值进行对称加密。所谓对称加密就是，将信息和私钥通过某种算法混合在一起，这样除非知道私钥，不然无法获取内容，而正好客户端和服务端都知道这个私钥，所以只要加密算法够彪悍，私钥够复杂，数据就够安全。

**7. 传输加密后的信息**

这部分信息是服务段用私钥加密后的信息，可以在客户端被还原。

**8. 客户端解密信息**

客户端用之前生成的私钥解密服务段传过来的信息，于是获取了解密后的内容。整个过程第三方即使监听到了数据，也束手无策。



# HTTPS的劣势

https的主要缺点就是性能问题。造成https性能低于http的原因有两个：

1. 对数据进行加解密决定了它比http慢。
2. 另外一个重要原因是https禁用了缓存。相关测试数据表明使用HTTPS协议传输数据的工作效率只有使用HTTP协议传输的十分之一。因此对于一个网站来说，只有对安全要求极高的的数据才会选择使用https进行传输。