计算机网络方面总结

网络协议模型

应用层、传输层、网络层、数据链路层、物理层

应用层：负责向用户提供应用程序，比如HTTP、FTP、Telnet、DNS、SMTP等

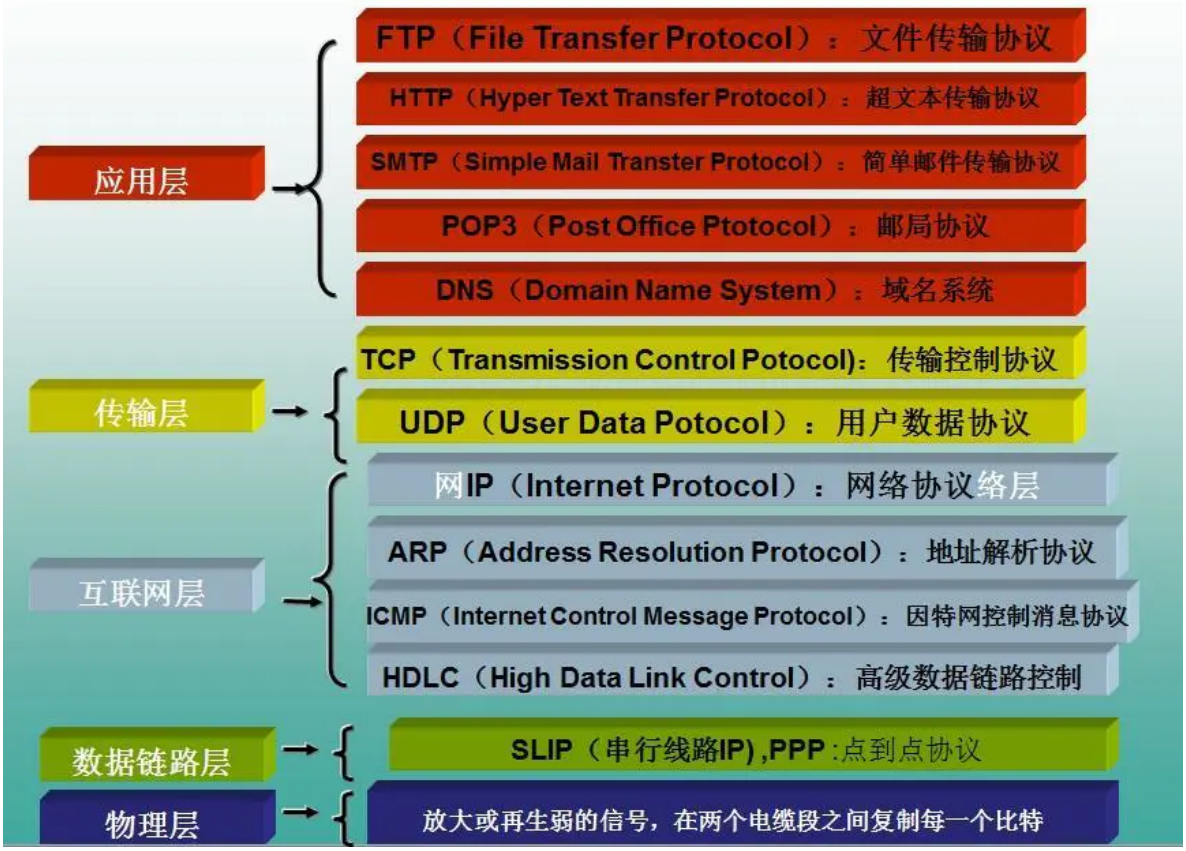
传输层：负责对报文进行分组和重组，并且以TCP和UDP协议格式封装报文

网络层：负责路由以及把分组报文发送给目标网络或者主机

数据链路层：传输有地址的帧以及错误检测功能。负责封装和解封装IP报文，发送和接受ARP/RARP报文等

物理层：以二进制数据形式在物理媒体上传输数据

OSI七层模型	TCP/IP概念层模型	功能	TCP/IP协议族
应用层	应用层	文件传输，电子邮件，文件服务，虚拟终端	TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
表示层		数据格式化，代码转换，数据加密	没有协议
会话层		解除或建立与别的接点的联系	没有协议
传输层	传输层	提供端对端的接口	TCP，UDP
网络层	网络层	为数据包选择路由	IP，ICMP，RIP，OSPF，BGP，IGMP
数据链路层	链路层	传输有地址的帧以及错误检测功能	SLIP，CSLIP，PPP，ARP，RARP，MTU
物理层		以二进制数据形式在物理媒体上传输数据	ISO2110，IEEE802，IEEE802.2



TCP与UDP

[一文搞懂TCP与UDP的区别]

[TCP和UDP的区别]

UDP特点：

1. 面向无连接

首先 UDP 是不需要和 TCP一样在发送数据前进行三次握手建立连接的，想发数据就可以开始发送了。并且也只是数据报文的搬运工，不会对数据报文进行任何拆分和拼接操作。

具体来说就是：

- 在发送端，应用层将数据传递给传输层的 UDP 协议，UDP 只会给数据增加一个 UDP 头标识下是 UDP 协议，然后就传递给网络层了
- 在接收端，网络层将数据传递给传输层，UDP 只去除 IP 报文头就传递给应用层，不会任何拼接操作

2. 有单播，多播，广播的功能

UDP 不止支持一对一的传输方式，同样支持一对多，多对多，多对一的方式，也就是说 UDP 提供了单播，多播，广播的功能。

3. UDP是面向报文的

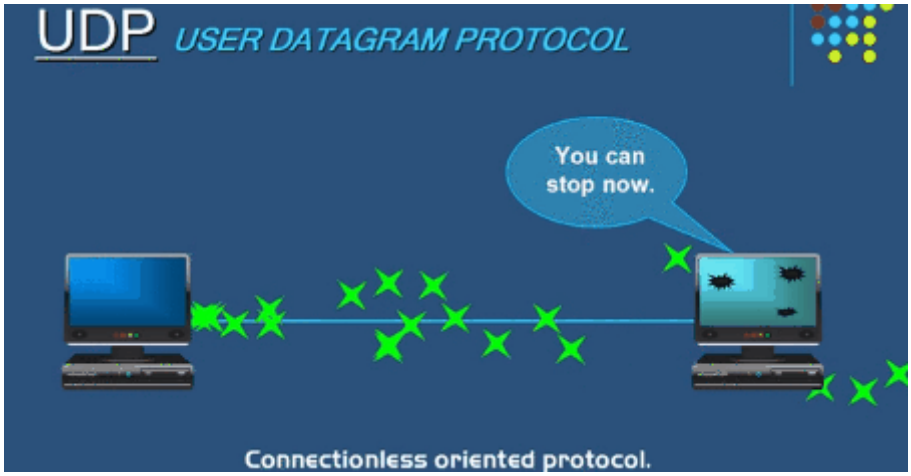
发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付IP层。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。因此，应用程序必须选择合适大小的报文

4. 不可靠性

首先不可靠性体现在无连接上，通信都不需要建立连接，想发就发，这样的情况肯定不可靠。

并且收到什么数据就传递什么数据，并且也不会备份数据，发送数据也不会关心对方是否已经正确接收到数据了。

再者网络环境时好时坏，但是 UDP 因为没有拥塞控制，一直会以恒定的速度发送数据。即使网络条件不好，也不会对发送速率进行调整。这样实现的弊端就是在网络条件不好的情况下可能会导致丢包，但是优点也很明显，在某些实时性要求高的场景（比如电话会议）就需要使用 UDP 而不是 TCP。



从上面的动态图可以得知，UDP只会把想发的数据报文一股脑的丢给对方，并不在意数据有无安全完整到达。

5. 头部开销小，传输数据报文时是很高效的。

UDP Header

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source port																Destination port															
Length																Checksum															

UDP 头部包含了以下几个数据：

- 两个十六位的端口号，分别为源端口（可选字段）和目标端口
- 整个数据报文的长度
- 整个数据报文的检验和（IPv4 可选 字段），该字段用于发现头部信息和数据中的错误

因此 UDP 的头部开销小，只有八字节，相比 TCP 的至少二十字节要少得多，在传输数据报文时是很高效的

TCP特点：

- 面向连接

面向连接，是指发送数据之前必须在两端建立连接。建立连接的方法是“三次握手”，这样能建立可靠的连接。建立连接，是为数据的可靠传输打下了基础。

- 仅支持单播传输

每条TCP传输连接只能有两个端点，只能进行点对点的数据传输，不支持多播和广播传输方式。

- 面向字节流

TCP不像UDP一样那样一个个报文独立地传输，而是在不保留报文边界的情况下以字节流方式进行传输。

- 可靠传输

对于可靠传输，判断丢包，误码靠的是TCP的段编号以及确认号。TCP为了保证报文传输的可靠，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的字节发回一个相应的确认(ACK)；如果发送端实体在合理的往返时延(RTT)内未收到确认，那么对应的数据（假设丢失了）将会被重传。

- 提供拥塞控制

当网络出现拥塞的时候，TCP能够减小向网络注入数据的速率和数量，缓解拥塞

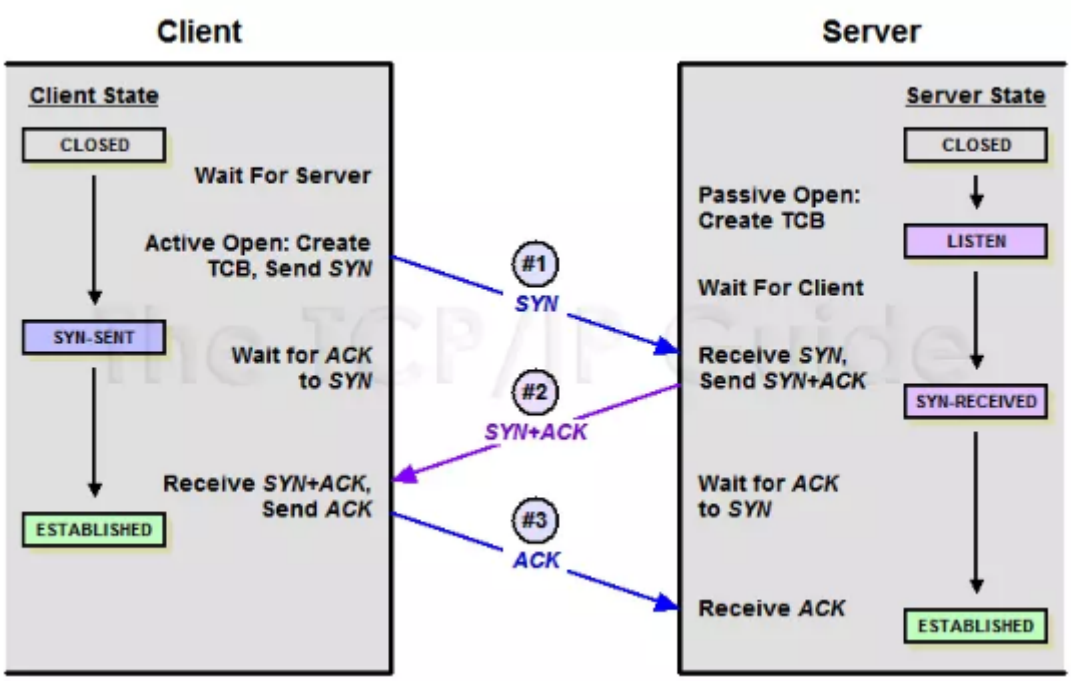
- TCP提供全双工通信

TCP允许通信双方的应用程序在任何时候都能发送数据，因为TCP连接的两端都设有缓存，用来临时存放双向通信的数据。当然，TCP可以立即发送一个数据段，也可以缓存一段时间以便一次发送更多的数据段（最大的数据段大小取决于MSS）

TCP与UDP区别：

	UDP	TCP
是否连接	无连接	面向连接
是否可靠	不可靠传输，不使用流量控制和拥塞控制	可靠传输，使用流量控制和拥塞控制
连接对象个数	支持一对一，一对多，多对一和多对多交互通信	只能是一对一通信
传输方式	面向报文	面向字节流
首部开销	首部开销小，仅8字节	首部最小20字节，最大60字节
适用场景	适用于实时应用（IP电话、视频会议、直播等）	适用于要求可靠传输的应用，例如文件传输

TCP连接过程-三次握手：



第一次握手

客户端向服务端发送连接请求报文段。该报文段中包含自身的数据通讯初始序号。请求发送后，客户端便进入 SYN-SENT 状态。

第二次握手

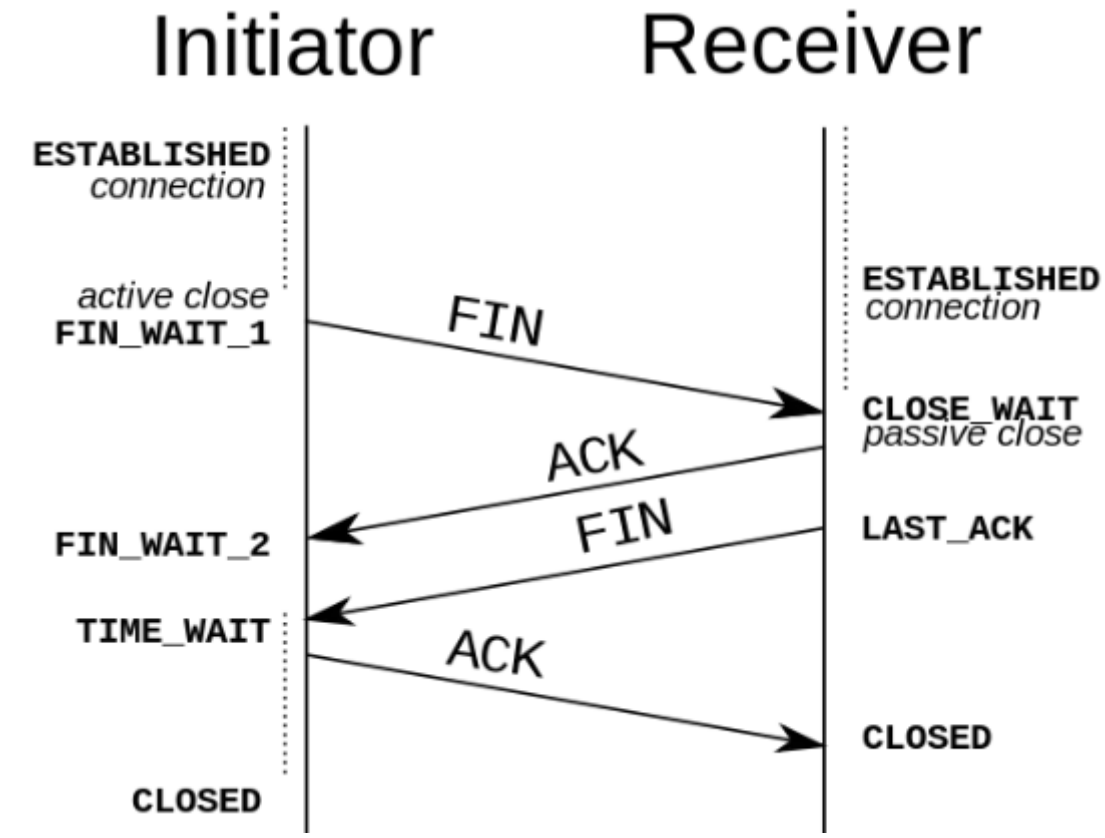
服务端收到连接请求报文段后，如果同意连接，则会发送一个应答，该应答中也会包含自身的数据通讯初始序号，发送完成后便进入 SYN-RECEIVED 状态。

第三次握手

当客户端收到连接同意的应答后，还要向服务端发送一个确认报文。客户端发完这个报文段后便进入 ESTABLISHED 状态，服务端收到这个应答后也进入 ESTABLISHED 状态，此时连接建立成功。

这里可能大家会有个疑惑：为什么 TCP 建立连接需要三次握手，而不是两次？这是因为这是为了防止出现失效的连接请求报文段被服务端接收的情况，从而产生错误。

TCP断开连接-四次握手



TCP 是全双工的，在断开连接时两端都需要发送 FIN 和 ACK。

第一次握手

若客户端 A 认为数据发送完成，则它需要向服务端 B 发送连接释放请求。

第二次握手

B 收到连接释放请求后，会告诉应用层要释放 TCP 链接。然后会发送 ACK 包，并进入 CLOSE_WAIT 状态，此时表明 A 到 B 的连接已经释放，不再接收 A 发的数据了。但是因为 TCP 连接是双向的，所以 B 仍旧可以发送数据给 A。

第三次握手

B 如果此时还有没发完的数据会继续发送，完毕后会向 A 发送连接释放请求，然后 B 便进入 LAST-ACK 状态。

第四次握手

A 收到释放请求后，向 B 发送确认应答，此时 A 进入 TIME-WAIT 状态。该状态会持续 2MSL（最大段生存期，指报文段在网络中生存的时间，超时会被抛弃） 时间，若该时间段内没有 B 的重发请求的话，就进入 CLOSED 状态。当 B 收到确认应答后，也便进入 CLOSED 状态。

TCP报头

- 源端口号、目标端口号

  TCP协议通过使用"端口"来标识源端和目标端的应用进程。端口号可以使用0到65535之间的任何数字，但是这些端口号已经被分为公认端口、注册端口和动态/私有端口。在收到服务请求时，操作系统动态地为客户端的应用程序分配端口号。

- 顺序号
- 确认号
- 头部长度

  占4比特。给出头部占32比特的数目。由于TCP报头的长度随TCP选项字段内容的不同而变化，因此报头中包含一个指定报头字段的字段。该字段以32比特为单位，所以报头长度一定是32比特的整数倍，有时需要在报头末尾补0。如果报头没有TCP选项字段，则报头长度值为5，表示报头一个有160比特，即20字节。

- 标志位 （U、A、P、R、S、F)

  占6比特。各比特的含义如下：
  >>> URG：报文段紧急。
  >>> ACK：确认序号有效。
  >>> PSH：接收方应该尽快将这个报文段交给应用层。
  >>> RST：重建连接。
  >>> SYN：发起一个连接。在握手完成后SYN为1，表示TCP建立已连接。此后的所有报文段中，SYN都被置0。
  >>> FIN：释放一个连接。如果源主机数据发送完毕，将把该连接下要发送的最后一个报文段的报头中的FIN位置1，或将该报文段后面发送的报头中该位置1。

- 窗口大小字段

  占16比特。此字段用来进行流量控制。单位为字节数，这个值是本机期望一次接收的字节数。接收计算机可接收的新数据字节的数量，根据接收缓冲区可用资源的大小，其值随计算机所发送的每个报文段而变化。源主机可以利用接收到的窗口值决定下一个报文段的大小。

- TCP校验和字段

  占16比特。对整个TCP报文段，即TCP头部和TCP数据进行校验和计算，并由目标端进行验证。

- 紧急指针

UDP报头

UDP Header

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source port																Destination port															
Length																Checksum															

UDP 头部包含了以下几个数据：

- 两个十六位的端口号，分别为源端口（可选字段）和目标端口
- 整个数据报文的长度
- 整个数据报文的检验和（IPv4 可选 字段），该字段用于发现头部信息和数据中的错误

因此 UDP 的头部开销小，只有八字节，相比 TCP 的至少二十字节要少得多，在传输数据报文时是很高效的

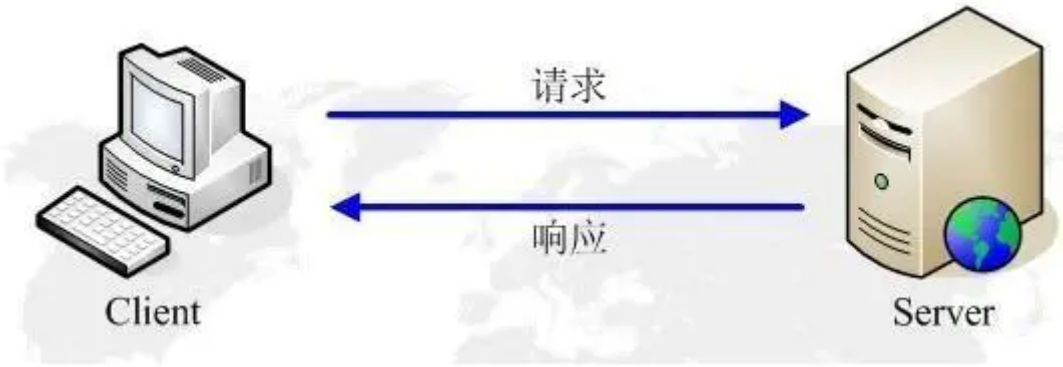
HTTP

HTTP概念

> 引自维基百科HTTP:超文本传输协议（英文：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。 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。通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

HTTP请求响应模型

HTTP由请求和响应构成，是一个标准的客户端服务器模型（B/S）。HTTP协议永远都是客户端发起请求，服务器回送响应。见下图:



HTTP是一个无状态的协议。无状态是指客户机（Web浏览器）和服务器之间不需要建立持久的连接，这意味着当一个客户端向服务器端发出请求，然后服务器返回响应(response)，连接就被关闭了，在服务器端不保留连接的有关信息.HTTP遵循请求(Request)/应答(Response)模型。客户机（浏览器）向服务器发送请求，服务器处理请求并返回适当的应答。所有HTTP连接都被构造成一套请求和应答。

HTTP工作过程

一次HTTP操作称为一个事务，其整个工作流程如下所示：

地址解析 → 封装HTTP请求数据包 → 封装成TCP包，建立TCP连接 → 客户机发送请求命令 → 服务器响应 → 服务器关闭TCP连接

1. **地址解析**

   例如用浏览器请求这个页面：localhost.com:8080/index.htm

   从中分解出协议、主机名、端口、对象路径等部分，对于我们的这个地址，解析得到的结果如下所示：

   ```
   协议名：http
   主机名：localhost.com
   端口：8080
   对象路径：/index.htm
   ```

   之后，需要通过域名系统DNS解析域名localhost.com,得到主机的IP地址

2. **封装HTTP请求数据包**

   把以上部分结合本机自身的信息，封装成一个HTTP请求数据包

3. **封装成TCP包，建立TCP连接（TCP的三次握手）**

   在HTTP工作开始之前，客户机（Web浏览器）首先要通过网络与服务器建立连接，该连接是通过TCP来完成的，该协议与IP协议共同构建Internet，即著名的TCP/IP协议族，因此Internet又被称作是TCP/IP网络。HTTP是比TCP更高层次的应用层协议，根据规则，只有低层协议建立之后才能，才能进行更层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。这里是8080端口。

4. **客户机发送请求命令**

   建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可内容。

5. **服务器响应**

   服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME信息包括服务器信息、实体信息和可能的内容。

   实体消息是服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据

6. **服务器关闭TCP连接**

   一般情况下，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接。然而如果浏览器或者服务器在其头部信息加入了这行代码：

   `Connection:keep-alive`

   TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽。

HTTP缺点

- 通信使用明文，可能被窃听
- 不验证通信方的身份，可能遭遇伪装
- 无法证明报文的完整性，有可能遭到篡改

这些缺点在网络通信中对企业安全是很致命的问题。

HTTPS

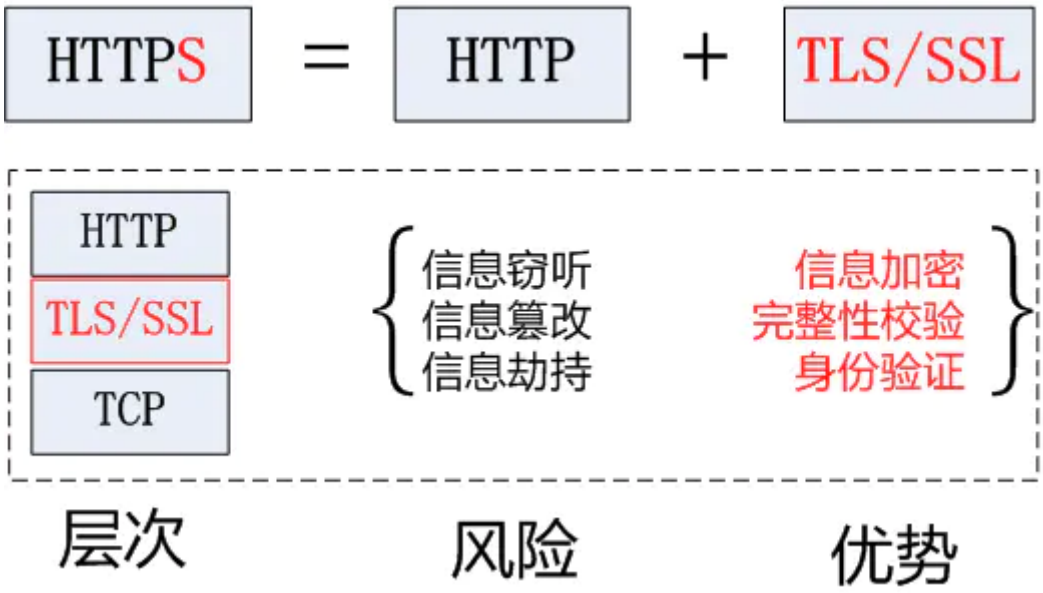
HTTP + 加密 + 认证 + 完整性保护 = HTTPS

HTTPS概念

> 引自维基百科HTTPS：超文本传输安全协议（英语：Hypertext Transfer Protocol Secure，缩写：HTTPS，常称为HTTP over TLS，HTTP over SSL或HTTP Secure）是一种通过计算机网络进行安全通信的传输协议。HTTPS经由HTTP进行通信，但利用SSL/TLS来加密数据包。HTTPS开发的主要目的，是提供对网站服务器的身份认证，保护交换数据的隐私与完整性。这个协议由网景公司（Netscape）在1994年首次提出，随后扩展到互联网上。历史上，HTTPS连接经常用于万维网上的交易支付和企业信息系统中敏感信息的传输。在2000年代晚期和2010年代早期，HTTPS开始广泛使用于保护所有类型网站上的网页真实性，保护账户和保持用户通信，身份和网络浏览的私密性。

HTTP协议采用明文传输，存在信息窃听、信息篡改和信息劫持的风险，而协议TLS/SSL具有身份验证、信息加密和完整性校验的功能，可以避免此类问题的发生。

TLS/SSL全称安全传输层协议Transport Layer Security，是介于TCP和HTTP之间的一层安全协议，不影响原有的TCP协议和HTTP协议，所以使用HTTPS基本上不需要对HTTP页面进行太多的改造。



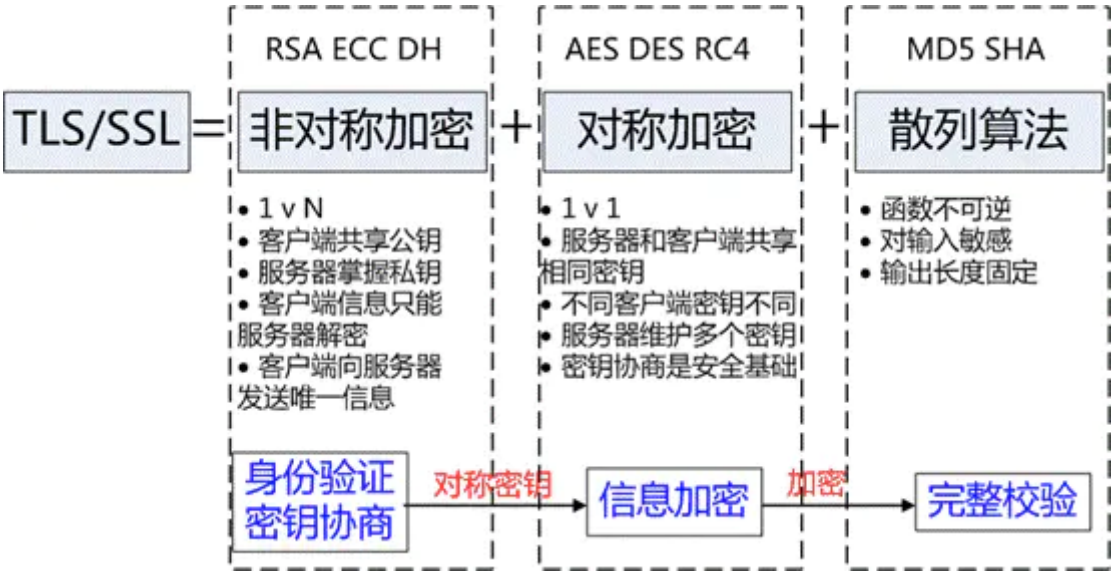
HTTPS是在HTTP上建立SSL加密层，并对传输数据进行加密，是HTTP协议的安全版。HTTPS主要作用是：

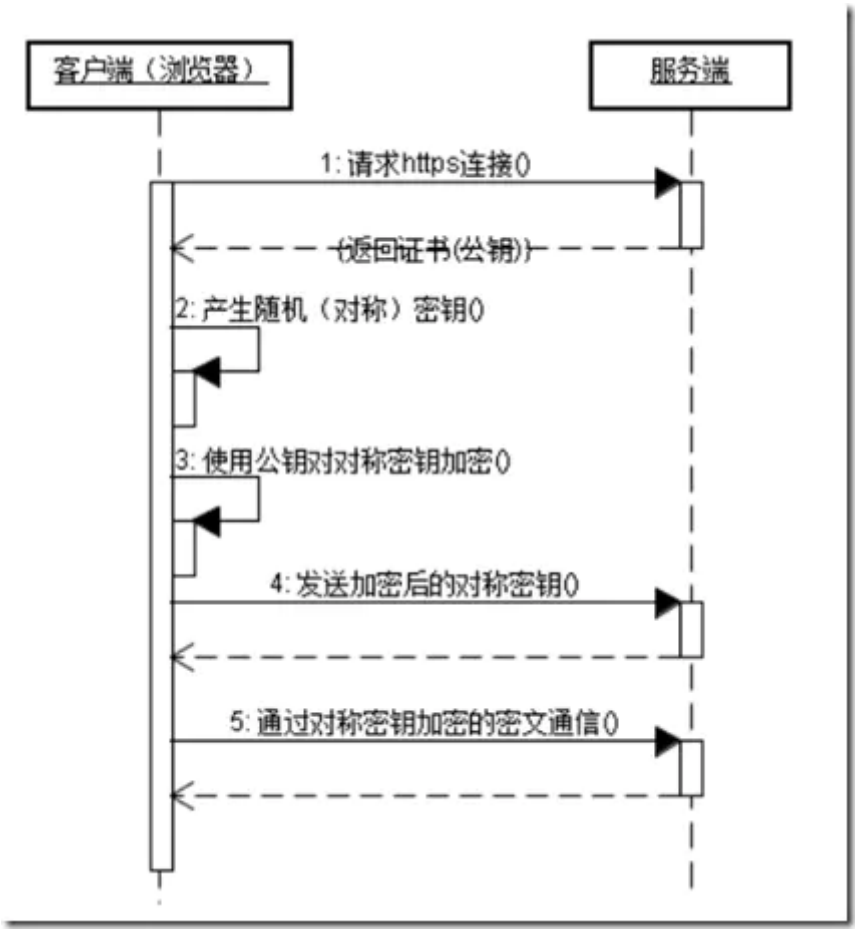
- 对数据进行加密，并建立一个信息安全通道，来保证传输过程中的数据安全
- 对网站服务器进行真实身份认证

HTTP和HTTPS的区别就在于HTTPS比HTTP多了一层TLS/SSL协议。

TLS/SSL工作原理

HTTPS协议的主要功能基本都依赖于TLS/SSL协议，而TLS/SSL的功能实现主要依赖于三类基本算法：散列函数Hash、对称加密和非对称加密。利用非对称加密实现身份认证和密钥协商，对称加密算法采用协商的密钥对数据加密，基于散列函数验证信息的完整性。





结合三类算法的特点，TLS的基本工作方式是，客户端使用非对称加密与服务器进行通信，实现身份验证并协商对称加密使用的密钥， 然后对称加密算法采用协商密钥对信息以及信息摘要进行加密通信，不同的节点之间采用的对称密钥不同，从而可以保证信息只能通信双方获取。