理解协议：TCP/UDP/HTTP/HTTPS

作者：胡健

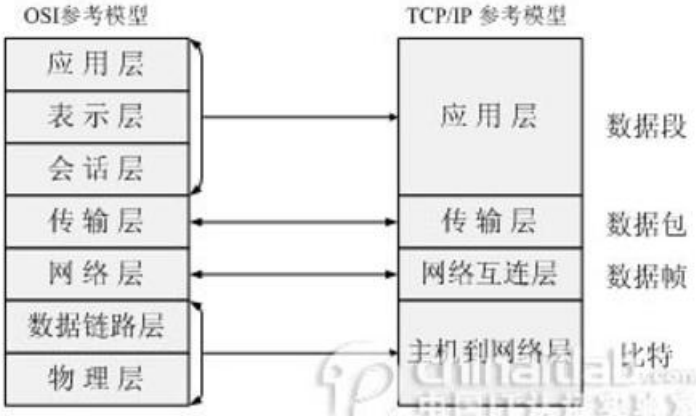
理解协议：TCP/UDP/HTTP/HTTPS.....- 1 -
1、 什么叫协议.....- 1 -
2、 TCP/IP 协议族.....- 2 -
3、 传输层协议 TCP/UDP.....- 3 -
3.1 UDP 协议的一些注意点.....- 3 -
3.2 TCP 协议的一些注意点.....- 4 -
3.2.1 TCP 建立连接和释放连接.....- 5 -
3.2.2 TCP 数据传输与拥塞控制.....- 6 -
4、 应用层协议 HTTP/HTTPS.....- 7 -
5、 结语.....- 13 -

1、什么叫协议

协议一般和通信一起出现，需求时通信，协议则为解决方案。无论是人与人之间，还是机器与机器之间，当有通信需求的时候，就要制定出相应的协议，也可以叫规则。应该指定好双方通信的方式方法，这样通信才能高效。再具体一点，协议就是一套让通信双方不得不遵守的规则，并且如果双方都严格按照这些规则通信的话，除了不可抗力之外均可成功，并且协议讲究简洁高效，负载越小，通信越快。

对于计算机来说，他们之间的通信就全靠协议，没有协议，就没有办法将多台计算机联合起来，也就没有所谓的互联网。例如，TCP/UDP 作为传输层协议，将上层协议的数据包传递到下层协议，并且在对方机器上将下层协议的数据包传递到上层协议，完成两台计算机通信的一部分。

2、TCP/IP 协议族



TCP/IP参考模型

TCP/IP 是计算机之间通信的协议族，是一个分层模型，其实最为规范的是 OSI 模型，OSI 模型更加完备，但是太过于复杂，实现起来难度较大，TCP/IP 协议已经可以很好的对两台或多台计算机进行通信，也是可实现的方案，所以现代操作系统都将 TCP/IP 作为网络通信协议模块。

TCP/IP 模型分为 4 层，从上到下分别是应用层，传输层，网络层，物理层。下面分别说明每一层的功能和实例。

应用层：这是程序员可以实际操作的一层，程序员可以不关心下层协议，只需要将数据包丢到应用层协议，然后再从应用层协议获取数据就可以了。比如 HTTP 协议，FTP 协议，等。当然，这些是著名的协议，著名的协议都指定占用一个著名的端口号，比如 HTTP 协议指定了 80 端口作为通信端口。其实，程序员可以实现自己的协议，基于下层（传输层）即可，然后选择一个端口作为通信端口，端口号是一个 16 位的整数，所以范围是 0~65535，0-1000 作为著名端口一般别去这里面选择自己的端口，可以从比较大的数字开始选择，比如 10000，当然也可以选择 80 端口，但是你会发现某些时候会发生“端口被占用”的错误，为了避免不必要的麻烦，选择一个非著名端口是一个明智的选择。

传输层：传输层从应用层获取数据，然后丢给下层协议，或者从下层协议获取数据包，然后解析出数据，传递给上层协议（应用层）。包括 UDP 协议和 TCP 协议。UDP 不提供可靠性保证，也不需要提前建立连接，所以一方在发送数据包的时候，甚至都不知道对方是否在接收。而 TCP 则是面向连接的，发送数据包前必须要经过建立连接的过程，确保双方都在线，然后在发送，而且通信保证可靠性。UDP 的有点是占用系统资源少，协议简单，数据通信快；缺点是没有可靠性保证。TCP 的有点是可靠，缺点就是占用系统资源多，没有 UDP 快速。所以在选择传输层协议的时候，要看自己的实际场景。这里稍微再说一下，话说这个占用的资源到底是什么呢？首先，什么叫资源？资源在计算机里面又是什么？TCP 和 UDP 会占用一些什么资源？对于 TCP 连接，一个连接需要一个文件描述符，一个读缓冲和一个写缓冲，一个文件描述符的大小大约为 1K，Linux 下读写缓冲的大小为 4096 字节，也就是 4k，也就是一个 UDP 连接需要大约 1k+8k=9k 内存，这是一个 TCP 连接占用的最小内存。总体来讲，所谓资源占用就是内存占用和 CPU 占用，TCP 的资源占用大于 UDP 的原因在于，TCP 的实现机制远比 UDP 复杂得多，需要增加额外的内存和 CPU 时间来做协议通信，所以资源占用较 UDP 高。

网络层：在 TCP/IP 网络协议中，网络层是整个协议族的核心，网络层实现的目的是

将本机的数据包发送到目标主机上，所以，实现两台计算机之间的通信依靠的是网络层。网络层的主要协议为 IP，ARP 等。IP 协议提供尽力而为的服务，丢包与否要看网络的负载情况，ARP 是一种局域网内部主机发现的协议。

物理层：物理层解决实际的字节传输问题，通过各种介质提供数据包进行端到端的传输服务。

3、传输层协议 TCP/UDP

现在我想重点来学习一下传输层的协议，TCP/UDP，两种协议都可以完成传输工作，但是 TCP 具备可靠性保证，基于连接，流式传输，而 UDP 则更为任意，并不面向连接，传输更加迅速。总结来说，各有各的有点，也各具缺点，当面向的应用是不允许出现差错的话，选择 TCP 是必须的；但是如果丢掉一写包并不会影响整体业务的话，比如视频播放业务，丢掉一辆帧不太可能对整个视频产生影响，此时选择 UDP 可以加快传输速度，对了，TCP 对网络的贡献真的很大，甚至为了整个网络的未来，甚至具备拥塞控制，但是 UDP 就没有，可以任意发送，但是也可能发了都对方都接不到。还需要注意的是，传输层仅仅是在该层进行设计，下层还是依靠网络层的服务质量，而 IP 协议提供的也是不可靠服务，所以，整体来说，没有绝对的可靠性，TCP 只是在传输层上实现了可靠性，而 TCP 依然需要依靠 IP 的不可靠服务来将数据包路由出去。

3.1 UDP 协议的一些注意点

UDP 不仅不面向连接，而且不保证数据包按序到达。

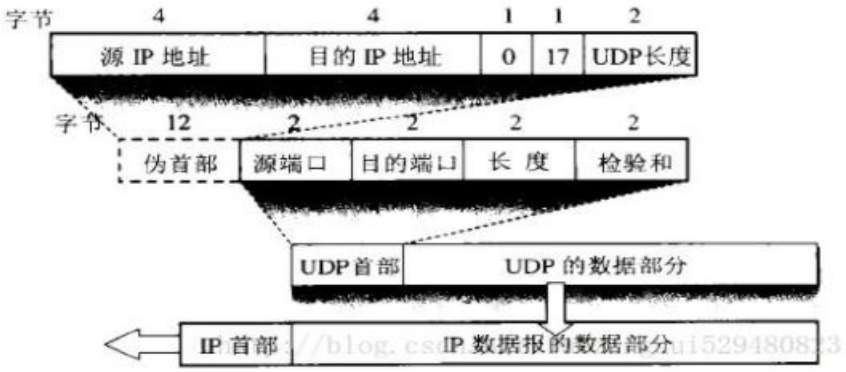


图 3.1 UDP 数据包格式

图 3.1 展示了 UDP 数据包的格式，注意到有一个字段为“伪首部”，该部分是为了计算校验和而引入的，在进行检验和计算时，会添加一个伪首部一起进行运算。伪首部（占用 12 个字节）为：4 个字节的源 IP 地址、4 个字节的目的 IP 地址、1 个字节的 0、一个字节的数字 17、以及占用 2 个字节 UDP 长度。这个伪首部不是报文的真正首部，只是引入为了计算校验和。 可笑的是，UDP 头部的校验和是可选的，应用层忽略或者接管数据校验都是可行的。UDP 的头部增加了源端口和目的端口，这样的话，数据包被发送到目的主机后，拆开 UDP 数据包之后根据端口号来交付数据了。

3.2 TCP 协议的一些注意点

TCP 协议提供的是面向连接的，具备拥塞控制机制的，按序交付的传输层协议。是一种提供可靠性保证的协议，图 3.2 是 TCP 协议的数据包格式。

图 3.2 TCP 格式

●源、目标端口号字段：占 16 比特。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。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。在服务器端，每种服务在"众所周知的端口"（Well-Know Port）为用户提供服务。

●顺序号字段：占 32 比特。用来标识从 TCP 源端向 TCP 目标端发送的数据字节流，它表示在这个报文段中的第一个数据字节。

●确认号字段：占 32 比特。只有 ACK 标志为 1 时，确认号字段才有效。它包含目标端所期望收到源端的下一个数据字节。

●头部长度字段：占 4 比特。给出头部占 32 比特的数目。没有任何选项字段的 TCP 头部长度为 20 字节；最多可以有 60 字节的 TCP 头部。

●标志位字段（U、A、P、R、S、F）：占 6 比特。各比特的含义如下：

◆URG：紧急指针（urgent pointer）有效。

◆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。

◆FIN：释放一个连接。

●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。

●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。

●紧急指针字段：占 16 比特。它是一个偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。

●选项字段：占 32 比特。可能包括"窗口扩大因子"、"时间戳"等选项。

3.2.1 TCP 建立连接和释放连接

TCP 连接建立连接需要经历三次握手过程，下面图 3.3 为建立连接时的三次握手示意图。

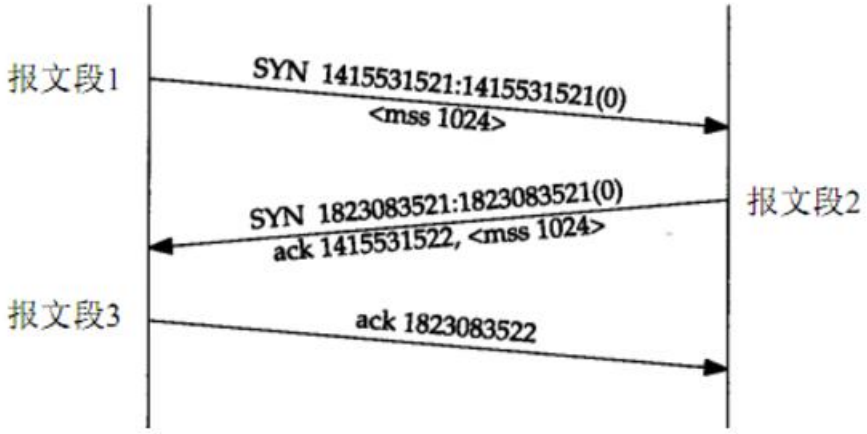


图 3.3 TCP 建立连接过程

（1）客户端首先向服务端发送一个 SYN 数据包，并且将客户端的初始化序列号告诉服务端

（2）服务端接收到之后同样向客户端发送一个 SYN 数据包，包含服务端使用的起始序列号，以及对客户端序列号的 ACK 作为服务端期望接收到的下一个字节的序列号。

（3）客户端就收到来自服务端的 ACK 之后，同样发送一个 ACK 数据包来回应，代表客户端期望接下来从服务端收到的字节的序列号。

经过上面三个步骤，也就是三次握手之后，一个 TCP 连接就建立起来了，接下来就可以根据通信双方协商好的变量（端口号，序列号，窗口大小）来进行数据传输了。

TCP 断开连接则需要经历四次挥手过程，图 3.4 就是整个断开连接的过程。

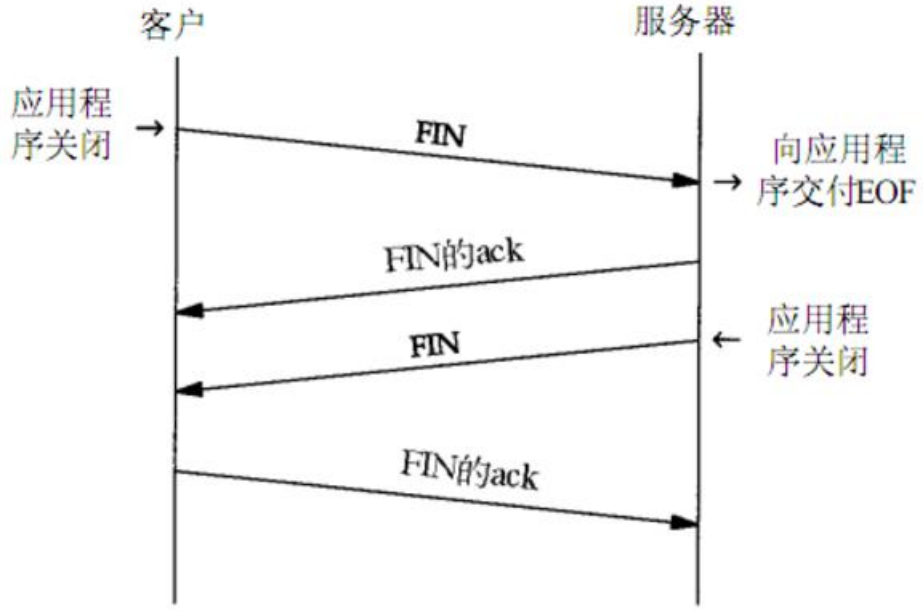


图 3.4 TCP 断开连接的过程

（1）客户端首先发送一个 FIN 数据包，告诉服务端数据传输已经结束，现在要开始断开连接了

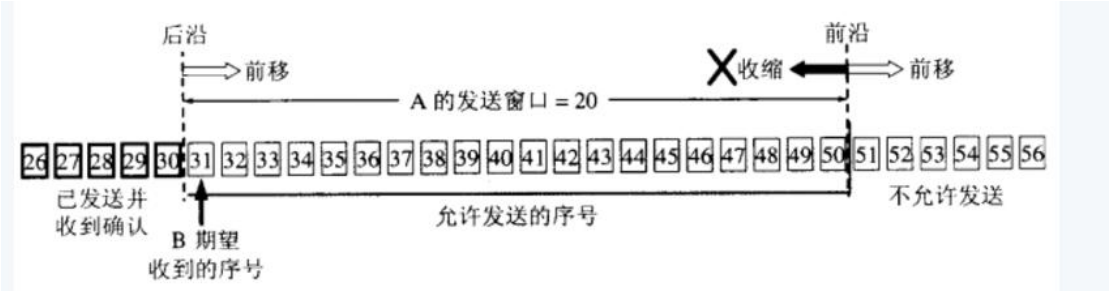
（2）服务端接收到 FIN 数据包之后，会立刻回复客户端一个 ACK 数据包，告诉客户端服务端已经知道了，而此时客户端依然在等待

（3）服务端结束了工作之后，也会想客户端发送一个 FIN 数据包，告诉客户端服务端已经准备关闭连接了，然后数据传输就结束了

（4）客户端接收到服务端发来的 FIN 之后，会立刻发送 ACK 数据包，表示客户端已经知道了，到这个时候，客户端和服务端都已经知道了双方已经关闭连接的事情了

3.2.2 TCP 数据传输与拥塞控制

（1）滑动窗口协议



应用层需要发送的数据都会被编号，然后 TCP 用一个窗口来发送数据，这个窗口的大小根据建立连接时确认的窗口大小来设定。应该说，应用层的数据会放在 TCP 连接的发送缓冲区中，根据滑动窗口协议，数据可以分为这么几部分，已经发送给对方，而且已经得到 ACK 的数据，已经发送但是还没有得到 ACK 的数据，可以发送但是还没有发送的数据，以及不允许发送的数据。发送端将根据接收到的 ACK 来移动这个窗口。接收端可能得到乱序的数据，但是为了避免网络重传，接收端会暂时接收下来。只要窗口的第一个序号没有得到 ACK，这个窗口都不会移动。

（2）拥塞控制

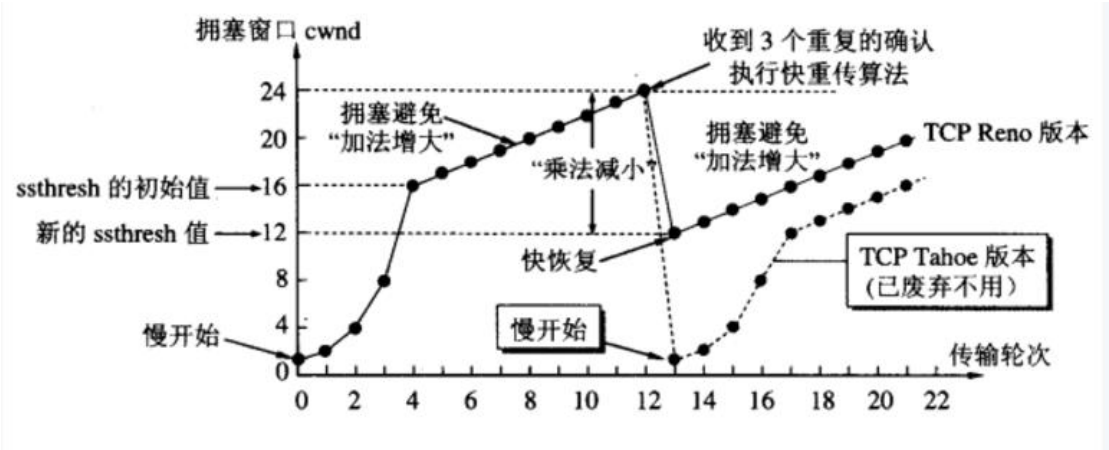


图 3.5 TCP 拥塞控制算法

所谓拥塞控制，就是别让网络太拥挤。有 Tahoe 算法和 Reno 算法。

Reno 算法：慢开始，指数增长，慢增长，快速重传。减半慢增长。

Tahoe 算法：慢开始，指数增长，慢增长，快速重传，慢开始。

（3）流量控制

TCP 流量控制通过滑动窗口来实现，发送发发送的窗口大小不得大于接收方回馈的窗口大小。所以当接收发的接收缓冲区没有空闲的时候，就会想发送发发发送一个窗口大小为 0 的数据包，发送发就会暂停发送数据给接收端，这样就可以控制流量。

但是有一种情况是，当接收方缓存区没有空闲了时，接收方向发送方发送了一个窗口大小为 0 的数据包，发送方接到后暂停发送数据；当接收发有空闲接收缓冲时，想要发送一个窗口大小来请求数据，但是这个数据包丢失了，这个时候，出现了一种情况，发送发已经暂停发送数据，并且在等待接收发来通知自己，而接收发现在已经发送了一个通知数据包，但是这个数据包丢了，所以他在等待的数据不可能达到。这就造成了死锁，为了解决这个问题，TCP 规定当一端接收到了一个窗口为 0 的数据包时，就启动一个计时器，当时间到时发送数据包探测一下是否可以再次发送数据，如果可以就继续发送，如果不可以，就再次计时，一直到可以发送数据包为止。

4、应用层协议 HTTP/HTTPS

HTTP（Hyper Text Transfer Protocol）是一种用于从服务端传输数据到客户端的明文文本协议，数据包可读，可理解其意义，所以，可能安全性没有保障，所以有了 HTTPS 协议，HTTPS 协议应该是是 HTTP 的安全版本。HTTP 协议的具有简单，灵活，无连接，无状态等特点，它允许传输任意内容，任意类型的文件，每个连接只处理一个请求，所以是无连接的，服务器处理完一个请求之后会断开连接。但是 HTTP 可以设置为长连接，处理完一个请求之后不会立刻断开连接，更适合与多个请求的连接。

（1）HTTP 的 URL

URL（uniform resource locator）用来指向网络上的一个资源，HTTP 的 URL 就是在浏览器地址栏输入的字符串，类似：http://host[:port]/[path]，host 是必须的，port 和 path 是可选的，如果没有 port，将默认为 80 端口，如果没有 path，则服务器会返回类似 http://host/index.html 这样的网页，可以在服务端设置具体的返回内容。

URL 中还可以包含请求参数，当 HTTP 的请求类型为 GET 时，请求的参数会包含在 URL 中一并发送到服务端，服务端可以从类似 key=value 的字符串中解析出请求参数。

（2）HTTP 请求

HTTP 请求分为三部分，分别为请求行、消息报头，请求体。

请求行：请求行以请求方法开头，请求方法将在下文提到，一般有 GET，POST，接着空一格，后面是请求的 URI，然后空一格，最后加上协议版本。比如下面就是一个合法的请求行：

GET / HTTP/1.1

消息报头：请求报头运行客户端向服务端发送一些附加信息，当然也需要将客户端自己的一些信息附加进去。常见的附加信息如下：

字段	说明
Accept	表示浏览器支持的 MIME 类型
Accept-Encoding	浏览器支持的压缩类型
Accept-Language	浏览器支持的语言类型，并且优先支持靠前的语言类型
Cache-Control	指定请求和响应遵循的缓存机制
Connection	当浏览器与服务器通信时对于长连接如何进行处理：close/keep-alive
Cookie	向服务器返回cookie，这些cookie是之前服务器发给浏览器的
Host	请求的服务器URL
Referer	该页面的来源URL
User-Agent	用户客户端的一些必要信息

图 4.1 HTTP 请求头

请求体：包含请求的数据

（3）HTTP 回复

HTTP 回复也包含三部分，分别是状态行，消息报头，响应体。

状态行：用于高速客户端请求结果，状态码下文会提到，一般回复 200 代表请求成功，下面是状态行的格式：

首先是 HTTP 版本，接着是状态码，然后是状态码所代表的含义。

消息报头：服务器返回时附加的一些信息，下面是一些信息

字段	说明
Cache-Control	告诉浏览器或者其他客户，什么环境可以安全地缓存文档
Connection	当client和server通信时对于长链接如何进行处理
Content-Encoding	数据在传输过程中所使用的压缩编码方式
Content-Type	数据的类型
Date	数据从服务器发送的时间
Expires	应该在什么时候认为文档已经过期，从而不再缓存它？
Server	服务器名字。Servlet一般不设置这个值，而是由Web服务器自己设置
Set-Cookie	设置和页面关联的cookie
Transfer-Encoding	数据传输的方式

图 4.2 HTTP 返回消息头

响应体：服务器处理请求得到的结果将放在响应体里面。

（4）HTTP 请求方法

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

图 4.3 HTTP 请求方法

（5）HTTP 状态码

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

图 4.4 HTTP 状态码

状态码	状态码英文名称	中文描述
100	Continue	继续。客户端应继续其请求
101	Switching Protocols	切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议
200	OK	请求成功。一般用于GET与POST请求
201	Created	已创建。成功请求并创建了新的资源
202	Accepted	已接受。已经接受请求，但未处理完成
203	Non-Authoritative Information	非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本
204	No Content	无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档
205	Reset Content	重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域
206	Partial Content	部分内容。服务器成功处理了部分GET请求

图 4.5 HTTP 状态码之 2**

300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择
301	Moved Permanently	永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替
302	Found	临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI
303	See Other	查看其它地址。与301类似。使用GET和POST请求查看
304	Not Modified	未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源
305	Use Proxy	使用代理。所请求的资源必须通过代理访问
306	Unused	已经被废弃的HTTP状态码
307	Temporary Redirect	临时重定向。与302类似。使用GET请求重定向

图 4.6 HTTP 状态码之 3**

400	Bad Request	客户端请求的语法错误，服务器无法理解
401	Unauthorized	请求要求用户的身份认证
402	Payment Required	保留，将来使用
403	Forbidden	服务器理解请求客户端的请求，但是拒绝执行此请求
404	Not Found	服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面
405	Method Not Allowed	客户端请求中的方法被禁止
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权
408	Request Time-out	服务器等待客户端发送的请求时间过长，超时

图 4.7 HTTP 状态码之 4**（1）

409	Conflict	服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突
410	Gone	客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息
412	Precondition Failed	客户端请求信息的先决条件错误
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式
416	Requested range not satisfiable	客户端请求的范围无效
417	Expectation Failed	服务器无法满足Expect的请求头信息

图 4.8 HTTP 状态码之 4**（2）

500	Internal Server Error	服务器内部错误，无法完成请求
501	Not Implemented	服务器不支持请求的功能，无法完成请求
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求
503	Service Unavailable	由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中
504	Gateway Time-out	充当网关或代理的服务器，未及时从远端服务器获取请求
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理

图 4.9 HTTP 状态码之 5**

（6）HTTPS

HTTPS 较 HTTP 加入了 SSL 层，做安全，所以可以算是 HTTP 的安全版本。好吧，HTTPS 就是为了敏感信息被第三方获取。HTTP 默认 80 端口，HTTPS 默认端口为 443 端口。

1、客户端发起 HTTPS 请求

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、服务端解密信息

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。

5、结语

从协议说起，然后说道 OSI，TCP/IP 协议模型，然后讨论了各层协议的功能，然后重新认识了 TCP/UDP，重点在 TCP，然后在应用层重新学习了 HTTP 和稍微提了 HTTPS。就这样吧，需要学习的太多，总是觉得静不下来好好看看基础的东西，看着看着觉得需要去衍生的内容太多，边做边学吧。