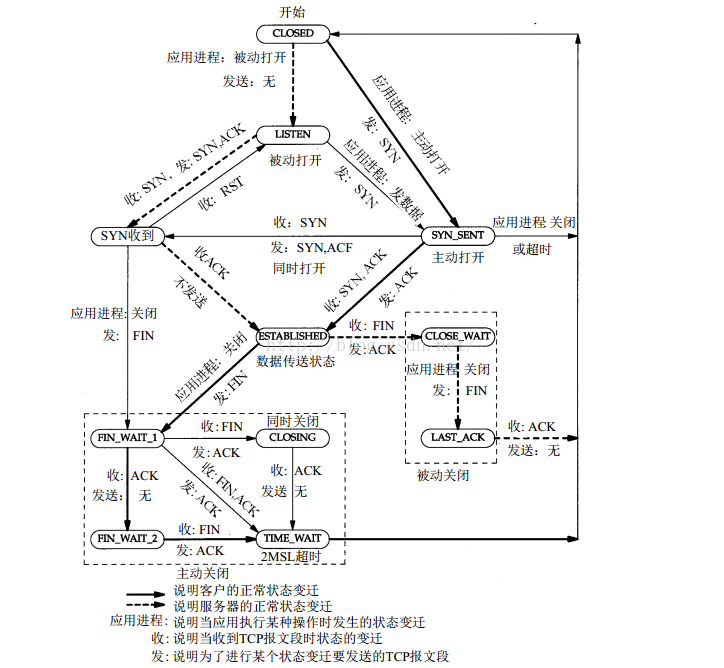
**1.计算机网络**

**1.1 TCP状态转换图**



**1.2 IP头部**

  
通常20字节，最长不超过60字节

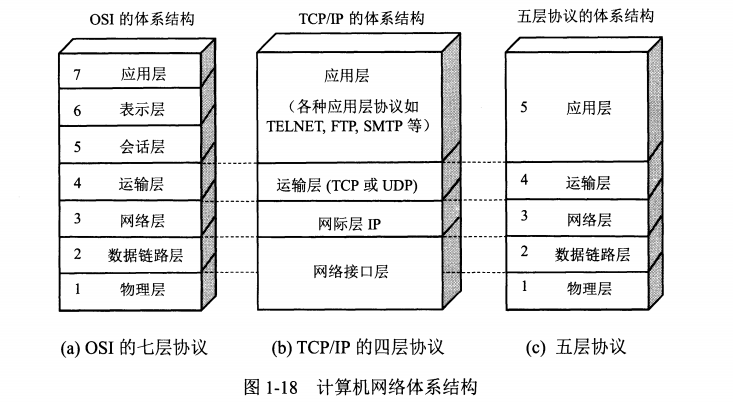
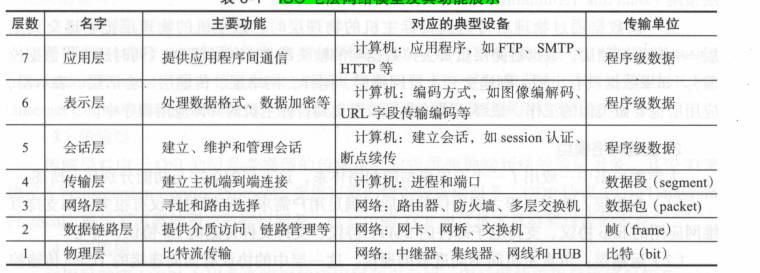
**1.3 TCP头部**

  
通常20字节，最长不超过60字节

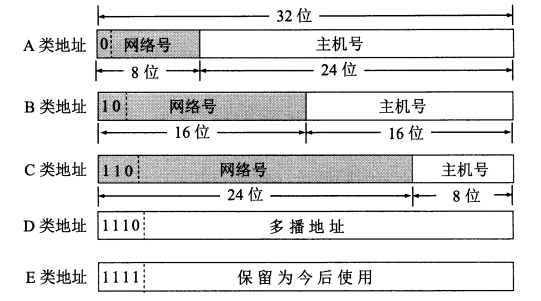
**1.4 UDP头部**



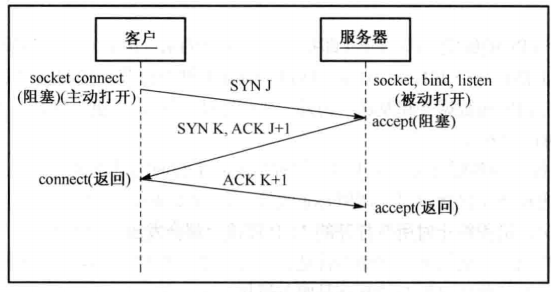
**1.5网络7层、4层 、5层模型(体系结构)**

**1.6 IP地址分类**



**1.7 3次握手(三次握手)**

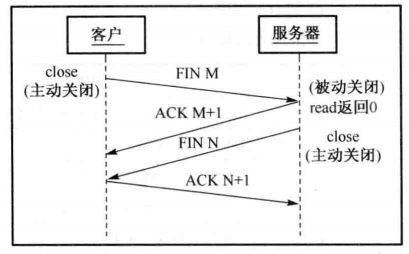


服务器通过调用socket、bind、listen函数准备好接受外来连接，成为被动打开，而客户端调用socket函数主动打开：

1. 客户端向服务器发送一个同步分节SYN J，connect进入阻塞状态；
2. 服务器向客户端响应一个同步分节SYN K，并对SYN J进行确认ACK J+1；
3. 客户端再向服务器发送一个确认位ACK K+1；

客户端的connect在三次握手的第二次返回，服务器的accept在第三次返回。

**1.8 4次挥手(四次挥手)**



1. 某个应用进程首先调用close函数执行主动关闭，这一端的TCP于是发送一个终止位FIN M分节表示数据发送完毕；
2. 另一端接收到终止位FIN分节后执行被动关闭，对这个分节进行确认，发送确认分节ACK M+1；
3. 一段时间后，调用close函数关闭套接口执行主动关闭，发送一个终止分节FIN N；
4. 接收到这个FIN的元发送端TCP对它进行确认，返回一个确认分节ACK N+1；  
   1. **交换机与路由器的区别**

（1）交换机工作于数据链路层，能识别 MAC 地址，根据 MAC 地址转发链路层数据帧。具有自学机制来维护 IP 地址与 MAC 地址的映射。

（2）路由器位于网络层，能识别 IP 地址并根据 IP 地址转发分组。维护着路由表，根据路由表选择最佳路线。

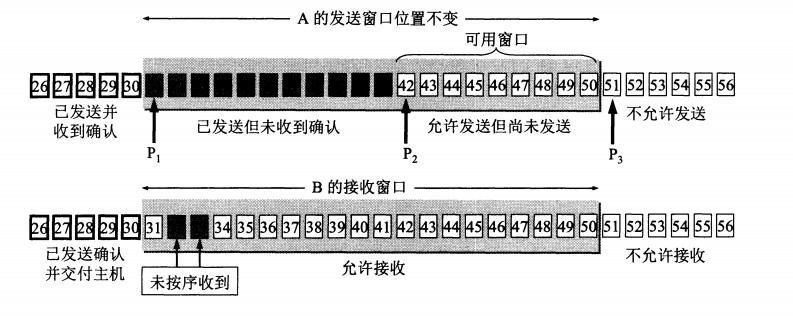
* 1. **TIME\_WAIT(2MSL)**

执行主动关闭的那一端进入TIME\_WAIT状态，存在理由如下：

（1）实现终止TCP全双工连接的可靠性，确保最后一个确认报文段能够到达。如果被动关闭的一端没收到主动关闭一端最终发送来的确认报文段，那么就会重新发送连接释放请求报文段，主动关闭的一端等待一段时间就是为了处理这种情况的发生。

（2）允许老的重复分节在网络中消失，可能存在“已失效的连接请求报文段”，为了防止这种报文段出现在本次连接之外，需要等待一段时间，防止串话。

**1.11 TCP滑动窗口**



“窗口”对应的是一段可以被发送者发送的字节序列，其连续的范围称之为“窗口”； “滑动”则是指这段“允许发送的范围”是可以随着发送的过程而变化的，方式就是按顺序“滑动”

**1.12 TCP 可靠传输**



（1）确认和重传：接收方收到报文就会确认，发送方发送一段时间后没有收到确认就重传。

（2）数据校验

（3）数据合理分片和排序：

　　UDP：IP数据报大于1500字节,大于MTU.这个时候发送方IP层就需要分片(fragmentation).把数据报分成若干片,使每一片都小于MTU.而接收方IP层则需要进行数据报的重组.这样就会多做许多事情,而更严重的是,由于UDP的特性,当某一片数据传送中丢失时,接收方便无法重组数据报.将导致丢弃整个UDP数据报.

　　tcp会按MTU合理分片，接收方会缓存未按序到达的数据，重新排序后再交给应用层。

（4）流量控制：当接收方来不及处理发送方的数据，能提示发送方降低发送的速率，防止包丢失。

（5）拥塞控制：当网络拥塞时，减少数据的发送。

**1.13 TCP 流量控制**

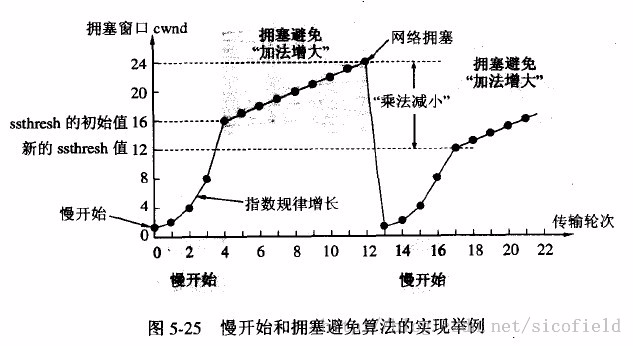
所谓流量控制，主要是接收方传递信息给发送方，使其不要发送数据太快，是一种端到端的控制。主要的方式就是返回的ACK中会包含自己的接收窗口的大小，并且利用大小来控制发送方的数据发送

**1.14 TCP 拥塞控制**

网络中的链路容量和交换结点中的缓存和处理机都有着工作的极限，当网络的需求超过它们的工作极限时，就出现了拥塞。拥塞控制就是防止过多的数据注入到网络中，这样可以使网络中的路由器或链路不致过载。常用的方法就是：

1. 慢开始、拥塞控制

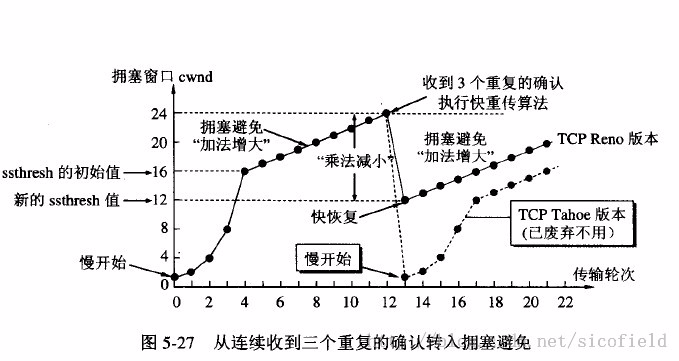
2. 快重传、快恢复



ssthresh：处理拥塞时参照的一个参数。例子中初始值为16，后来变为12。

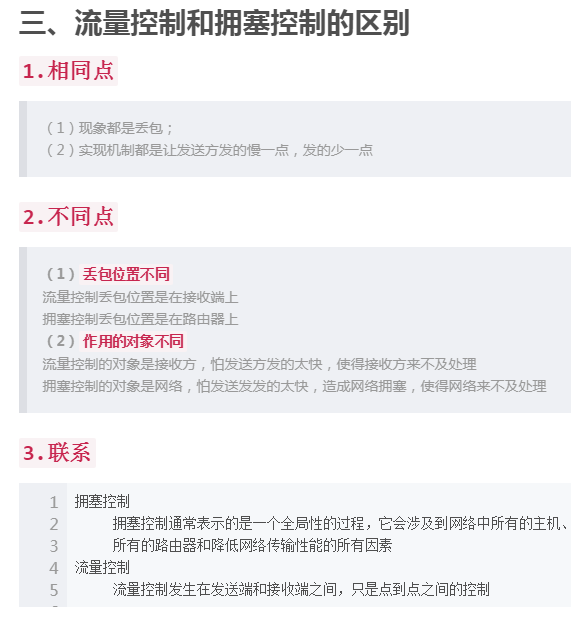
当cwnd> ssthresh，cwnd以慢开始的方法指数增长；

当cwnd< ssthresh，cwnd以拥塞避免的方法线性增长。



快重传：收到3个同样的确认就立刻重传，不等到超时；

快恢复：cwnd不是从1重新开始。



**1.15 网络编程一般步骤**

* TCP：
* 服务端：socket -> bind -> listen -> accept -> recv/send -> close。
* 客户端：socket -> connect -> send/recv -> close。
* UDP：
* 服务端：socket -> bind -> recvfrom/sendto -> close。
* 客户端：socket -> sendto/recvfrom -> close。
  1. **TCP和UDP区别**

（1）TCP面向连接的（三次握手），提供客户端与服务器的连接；而UDP是一种无连接的协议，因为UDP的客户端与服务器不必存在长期的关系，例如一个UDP客户创建一个套接字并发送一个数据报给一个服务器，然后立即用同一个套接字发送另一个数据报给另一个服务器。同样，一个UDP服务器可以用同一个套接字从若干不同的客户一连串接收多个数据报；

（2）TCP通过序号、重传、流量控制、拥塞控制实现可靠传输；而UDP不保障可靠传输，尽最大努力交付。

（3）TCP面向字节流传输，因此可以被分割并在接收端重组；UDP面向数据报传输。

**1.17 TCP为什么不是两次握手而是三次**

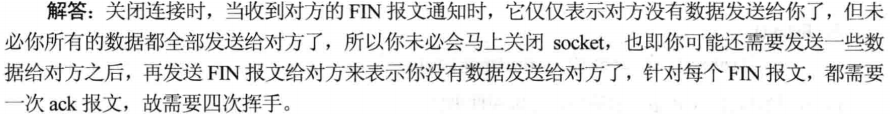
如果仅两次连接可能出现一种情况：客户端发送完连接报文（第一次握手）后由于网络不好，延时很久后报文到达服务端，服务端接收到报文后向客户端发起连接（第二次握手）。此时客户端会认定此报文为失效报文，但在两次握手情况下服务端会认为已经建立起了连接，服务端会一直等待客户端发送数据，但因为客户端会认为服务端第二次握手的回复是对失效请求的回复，不会去处理。这就造成了服务端一直等待客户端数据的情况，浪费资源。

**1.18 TCP为什么挥手是四次而不是三次**

TCP是全双工的，它允许两个方向的数据传输被独立关闭。当主动发起关闭的一方关闭连接之后，TCP进入半关闭状态，此时主动方可以只关闭输出流。

之所以不是三次而是四次主要是因为被动关闭方将"对主动关闭报文的确认"和"关闭连接"两个操作分两次进行。

对主动关闭报文的确认是为了快速告知主动关闭方，此关闭连接报文已经收到。此时被动方不立即关闭连接是为了将缓冲中剩下的数据从输出流发回主动关闭方（主动方接收到数据后同样要进行确认），因此要把"确认关闭"和"关闭连接"分两次进行。



* 1. **阻塞和非阻塞I/O区别**

（1）如果内核缓冲没有数据可读时，read()系统调用会一直等待有数据到来后才从阻塞态中返回，这就是阻塞I/O。

（2）非阻塞I/O在遇到上述情况时会立即返回给用户态进程一个返回值，并设置errno为EAGAIN。

（3）对于往缓冲区写的操作同理。

* 1. **同步和异步区别**

（1）同步I/O指处理I/O操作的进程和处理I/O操作的进程是同一个。

（2）异步I/O中I/O操作由操作系统完成，并不由产生I/O的用户进程执行。

* 1. **Reactor和Proactor区别**

（1）Reactor模式是同步I/O，处理I/O操作的依旧是产生I/O的程序；Proactor是异步I/O，产生I/O调用的用户进程不会等待I/O发生，具体I/O操作由操作系统完成。

（2）异步I/O需要操作系统支持，Linux异步I/O为AIO，Windows为IOCP。

**1.22 域名系统 DNS (使用UDP)**

1. 层次结构

域名服务器可以分为以下四类：

(1) 根域名服务器：解析顶级域名；

(2) 顶级域名服务器：解析二级域名；

(3) 权限域名服务器：解析区内的域名；

区和域的概念不同，可以在一个域中划分多个区。图 b 在域 [abc.com](http://abc.com/) 中划分了两个区：[abc.com](http://abc.com/) 和 [y.abc.com](http://y.abc.com/)

因此就需要两个权限域名服务器：

(4) 本地域名服务器：也称为默认域名服务器。可以在其中配置高速缓存。

2. 解析过程

主机向本地域名服务器解析的过程采用递归，而本地域名服务器向其它域名服务器解析可以使用递归和迭代两种方式。

迭代的方式下，本地域名服务器向一个域名服务器解析请求解析之后，结果返回到本地域名服务器，然后本地域名服务器继续向其它域名服务器请求解析；而递归地方式下，结果不是直接返回的，而是继续向前请求解析，最后的结果才会返回。

**1.23 Web 页面请求过程(URL请求过程)**

浏览器中输入URL，首先浏览器要将URL解析为IP地址，解析域名就要用到DNS协议，首先主机会查询DNS的缓存，如果没有就给本地DNS发送查询请求。DNS查询分为两种方式，一种是递归查询，一种是迭代查询。如果是迭代查询，本地的DNS服务器，向根域名服务器发送查询请求，根域名服务器告知该域名的一级域名服务器，然后本地服务器给该一级域名服务器发送查询请求，然后依次类推直到查询到该域名的IP地址。DNS服务器是基于UDP的，因此会用到UDP协议。

得到IP地址后，浏览器就要与服务器建立一个http连接。因此要用到http协议，http协议报文格式上面已经提到。http生成一个get请求报文，将该报文传给TCP层处理。如果采用https还会先对http数据进行加密。TCP层如果有需要先将HTTP数据包分片，分片依据路径MTU和MSS。TCP的数据包然后会发送给IP层，用到IP协议。IP层通过路由选路，一跳一跳发送到目的地址。当然在一个网段内的寻址是通过以太网协议实现(也可以是其他物理层协议，比如PPP，SLIP)，以太网协议需要直到目的IP地址的物理地址，有需要ARP协议。

**1.24 ARP协议**

应用接受用户提交的数据，触发TCP建立连接，TCP的第一个SYN报文通过connect函数到达IP层，IP层通过查询路由表：

　　如果目的IP和自己在同一个网段：

　　当IP层的ARP高速缓存表中存在目的IP对应的MAC地址时，则调用网络接口send函数（参数为IP Packet和目的MAC））将数据提交给网络接口，网络接口完成Ethernet Header + IP + CRC的封装，并发送出去；

　　当IP层的ARP高速缓存表中不存在目的IP对应的MAC地址时，则IP层将TCP的SYN缓存下来，发送ARP广播请求目的IP的MAC，收到ARP应答之后，将应答之中的<IP地址，对应的MAC>对缓存在本地ARP高速缓存表中，然后完成TCP SYN的IP封装，调用网络接口send函数（参数为IP Packet和目的MAC））将数据提交给网络接口，网络接口完成Ethernet Header + IP + CRC的封装，并发送出去；。

　　如果目的IP地址和自己不在同一个网段，就需要将包发送给默认网关(网关是默认的数据出口)，这需要知道默认网关的MAC地址：

　　当IP层的ARP高速缓存表中存在默认网关对应的MAC地址时，则调用网络接口send函数（参数为IP Packet和默认网关的MAC）将数据提交给网络接口，网络接口完成Ethernet Header + IP + CRC

　　当IP层的ARP高速缓存表中不存在默认网关对应的MAC地址时，则IP层将TCP的SYN缓存下来，发送ARP广播请求默认网关的MAC，收到ARP应答之后，将应答之中的<默认网关地址，对应的MAC>对缓存在本地ARP高速缓存表中，然后完成TCP SYN的IP封装，调用网络接口send函数（参数为IP Packet和默认网关的MAC）将数据提交给网络接口，网络接口完成Ethernet Header + IP + CRC的封装，并发送出去。

**ARP的位置**

　　OSI模型有七层，TCP在第4层传输层，IP在第3层网络层，而ARP在第2层数据链路层。高层对低层是有强依赖的，所以TCP的建立前要进行ARP的请求和应答。

　　ARP高速缓存表在IP层使用。如果每次建立TCP连接都发送ARP请求，会降低效率，因此在主机、交换机、路由器上都会有ARP缓存表。建立TCP连接时先查询ARP缓存表，如果有效，直接读取ARP表项的内容进行第二层数据包的发送；只有表失效时才进行ARP请求和应答进行MAC地址的获取，以建立TCP连接。

**ARP的作用**

　　要了解ARP的作用，首先要分清两个“地址”：

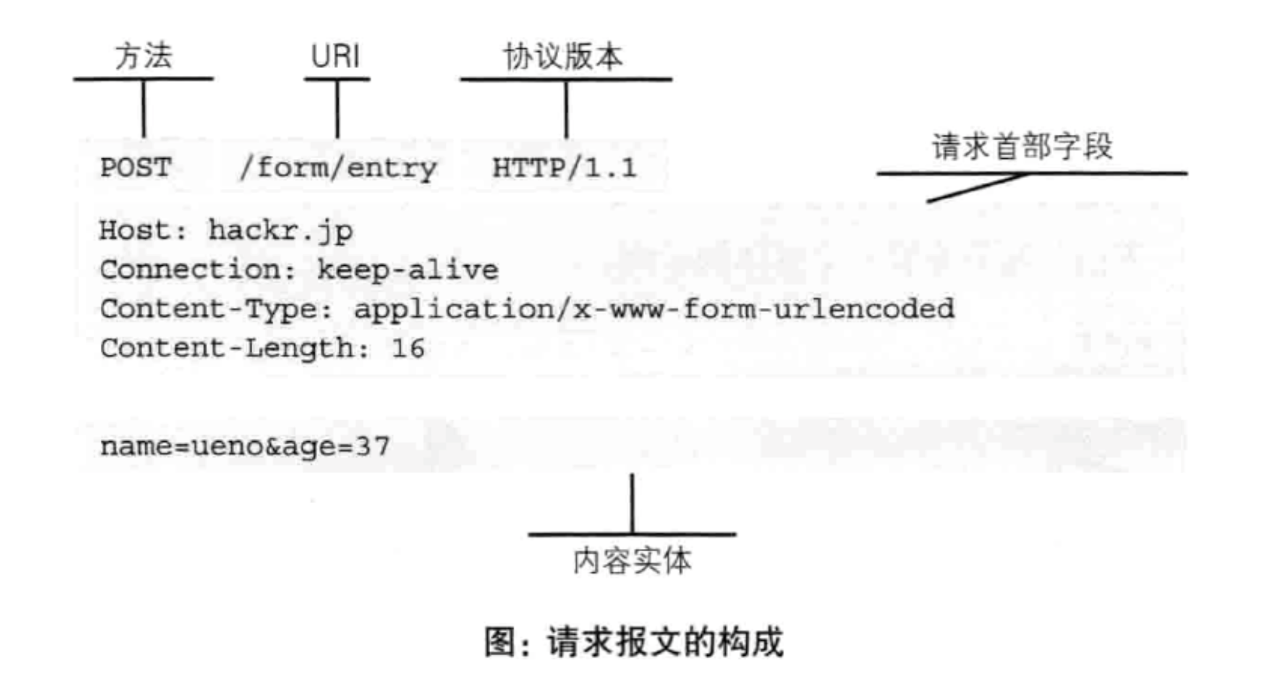
　　（1）TCP/IP的32bit IP地址。仅知道主机的IP地址不能让内核发送数据帧给主机。

　　（2）网络接口的硬件地址，它是一个48bit的值，用来标识不同的以太网或令牌环网络接口。在硬件层次上，进行数据交换必须有正确的接口地址，内核必须知道目的端的硬件地址才能发送数据。

　　简言之，就是在以太网中，一台主机要把数据帧发送到同一局域网上的另一台主机时，设备驱动程序必须知道以太网地址才能发送数据。而我们只知道IP地址，这时就需要采用ARP协议将IP地址映射为以太网地址。

　　要注意一点，一般认为ARP协议只使适用于局域网。

**1.25 HTTP请求报文**

  
如图，由四部分组成，分别是：

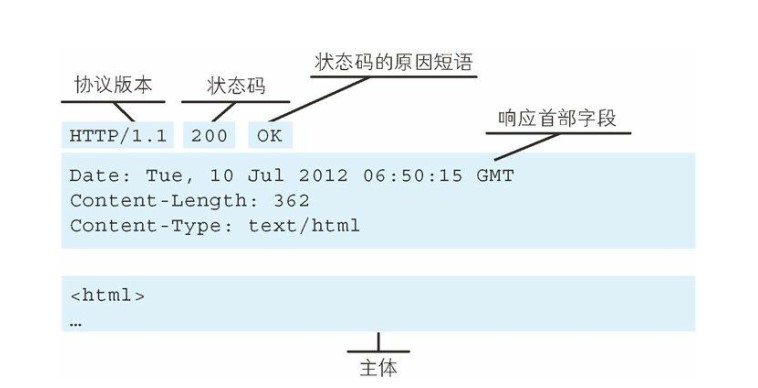
请求行 请求方法 URI 协议版本 \r\n

请求头部 name：val \r\n

空行

请求数据

**1.26 HTTP响应报文**



如图，由三部分组成，分别是：

状态行 协议版本 状态码 状态码文本描述

消息报头

响应正文

**1.27 HTTP请求方法**

GET：向特定资源发送请求

POST：向指定资源提交数据进行处理请求，如提交表单或者上传文件

HEAD：获取报文首部。和 GET 方法一样，但是不返回报文实体主体部分。主要用于确认 URL 的有效性以及资源更新的日期时间等。

PUT：向指定资源上传其最新内容。由于自身不带验证机制，任何人都可以上传文件，因此存在安全性问题，一般 WEB 网站不使用该方法。

DELETE：请求服务器删除所标识的资源文件。与 PUT 功能相反，并且同样不带验证机制。

OPTIONS：返回服务器对特定资源所支持的HTTP方法。会返回 Allow: GET, POST, HEAD, OPTIONS 这样的内容。

TRACE：追踪路径，回显服务器收到的资源，主要用于测试或者诊断，服务器会将通信路径返回给客户端。

Get和post的区别：

POST 主要目的不是获取资源，而是传输实体主体数据。

GET 和 POST 的请求都能使用额外的参数，但是 GET 的参数是以查询字符串出现在 URL中，而 POST 的参数存储在实体主体部分。

GET 的传参方式相比于 POST 安全性较差，因为 GET 传的参数在 URL 是可见的，可能会泄露私密信息。并且 GET 只支持 ASCII 字符，如果参数为中文则可能会出现乱码，而 POST 支持标准字符集

**1.28 HTTP 状态码**

服务器返回的响应报文中第一行为状态行，包含了状态码以及原因短语，来告知客户端请求的结果。

|  |  |  |
| --- | --- | --- |
| 状态码 | 类别 | 原因短语 |
| 1XX | Informational（指示信息状态码） | 接收的请求正在处理 |
| 2XX | Success（成功状态码） | 请求正常处理完毕 |
| 3XX | Redirection（重定向状态码） | 需要进行附加操作以完成请求 |
| 4XX | Client Error（客户端错误状态码） | 服务器无法处理请求 |
| 5XX | Server Error（服务器错误状态码） | 服务器处理请求出错 |

200 OK

204 No Content：请求已经成功处理，但是返回的响应报文不包含实体的主体部分。一般在只需要从客户端往服务器发送信息，而不需要返回数据时使用。

206 Partial Content

301 Moved Permanently：永久性重定向

302 Found：临时性重定向

303 See Other

304 Not Modified：如果请求报文首部包含一些条件，例如：If-Match，If-ModifiedSince，If-None-Match，If-Range，If-Unmodified-Since，但是不满足条件，则服务器会返回 304 状态码。

307 Temporary Redirect：临时重定向，与 302 的含义类似，但是 307 要求浏览器不会把重定向请求的 POST 方法改成 GET 方法。

400 Bad Request：请求报文中存在语法错误

401 Unauthorized：该状态码表示发送的请求需要有通过 HTTP 认证（BASIC 认证、DIGEST 认证）的认证信息。如果之前已进行过一次请求，则表示用户认证失败。

403 Forbidden：请求被拒绝，服务器端没有必要给出拒绝的详细理由。

404 Not Found：请求资源不存在

500 Internal Server Error：服务器正在执行请求时发生错误

503 Service Unavilable：该状态码表明服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。

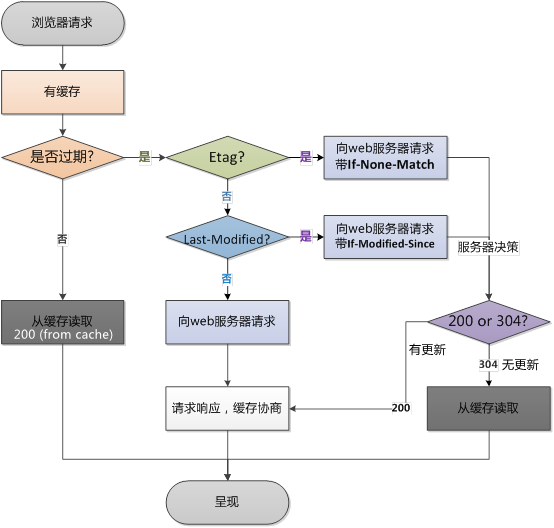
**1.29 HTTP常用字段**

Cache-Control、Connection、Accept-Charset、Accept-Encoding、Accept-Language、Host、If-Match、If-Modified-Since、Allow

**1.30 Session 和 Cookie 区别**

Session 是服务器用来跟踪用户的一种手段，每个 Session 都有一个唯一标识：Session ID。当服务器创建了一个 Session 时，给客户端发送的响应报文就包含了 Set-Cookie 字段，其中有一个名为 sid 的键值对，这个键值对就是 Session ID。客户端收到后就把 Cookie 保存在浏览器中，并且之后发送的请求报文都包含 Session ID。HTTP 就是 Session 和 Cookie 这两种方式一起合作来实现跟踪用户状态的，而 Session 用于服务器端，Cookie 用于客户端。

**1.31 HTTP缓存**



1、Exprires: Expires是Web服务器响应消息头字段，在响应http请求时告诉浏览器在过期时间前浏览器可以直接从浏览器缓存取数据，而无需再次请求。

2、Cache-Control: Cache-Control与Expires的作用一致，都是指明当前资源的有效期，控制浏览器是否直接从浏览器缓存取数据还是重新发请求到服务器取数据。 只不过Cache-Control的选择更多，设置更细致，如果同时设置的话，其优先级高于Expires。

3、Last-Modified/If-Modified-Since：Last-Modified/If-Modified-Since要配合Cache-Control使用。 Last-Modified：标示这个响应资源的最后修改时间。 If-Modified-Since：当资源过期时（使用Cache-Control标识的max-age），发现资源具有Last-Modified声明，则再次向web服务器请求时带上头 If-Modified-Since，表示请求时间。

4、Etag/If-None-Match：Etag/If-None-Match也要配合Cache-Control使用。 Etag：web服务器响应请求时，告诉浏览器当前资源在服务器的唯一标识（生成规则由服务器决定）。 Apache中，ETag的值，默认是对文件的索引节（INode），大小（Size）和最后修改时间（MTime）进行Hash后得到的。 If-None-Match：当资源过期时（使用Cache-Control标识的max-age），发现资源具有Etage声明，则再次向web服务器请求时带上头If-None-Match （Etag的值）。

**1.32 HTTP分块传输**

分块传输（Chunked Transfer Coding）可以把数据分割成多块，让浏览器逐步显示页面。

**1.33 HTTP范围请求**

如果网络出现中断，服务器只发送了一部分数据，范围请求使得客户端能够只请求未发送的那部分数据，从而避免服务器端重新发送所有数据。

在请求报文首部中添加 Range 字段，然后指定请求的范围，例如 Range : bytes = 5001-10000。请求成功的话服务器发送 206 Partial Content 状态。

**1.34 HTTP内容协商**

通过内容协商返回最合适的内容，例如根据浏览器的默认语言选择返回中文界面还是英文界面。

涉及以下首部字段：Accept、Accept-Charset、Accept-Encoding、Accept-Language、Content-Language。

**1.35 HTTP虚拟主机**

使用虚拟主机技术，使得一台服务器拥有多个域名，并且在逻辑上可以看成多个服务器。

**1.36 HTTP通信数据转发**

代理

代理服务器接受客户端的请求，并且转发给其它服务器。代理服务器一般是透明的，不会改变 URL。

使用代理的主要目的是：缓存、网络访问控制以及记录访问日志。

网关

与代理服务器不同的是，网关服务器会将 HTTP 转化为其它协议进行通信，从而其它非 HTTP 服务器的服务。

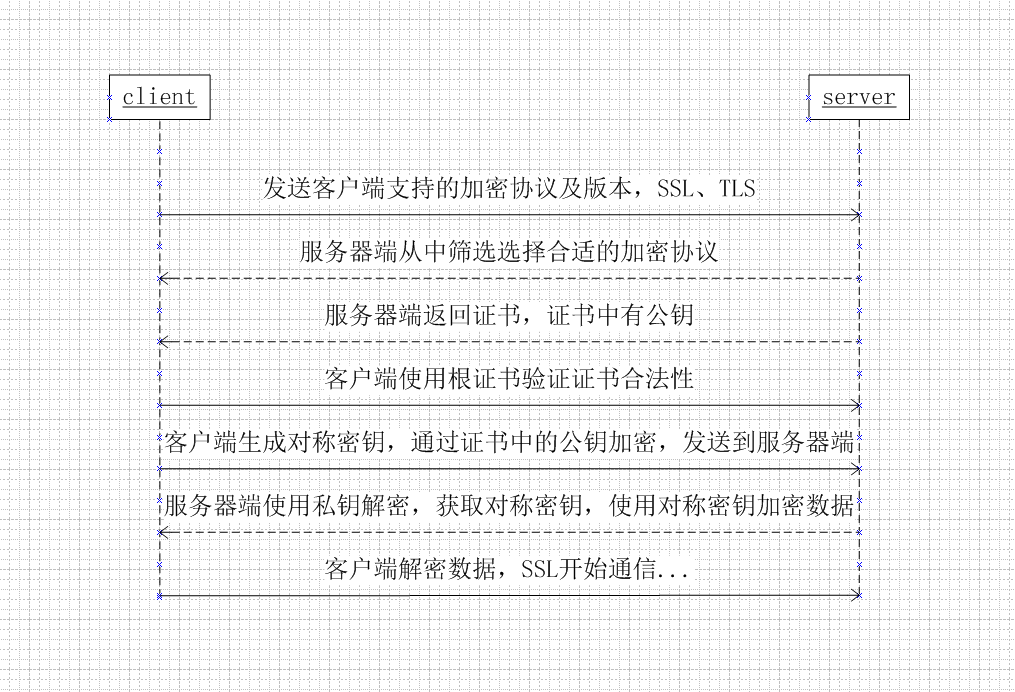
隧道

使用 SSL 等加密手段，为客户端和服务器之间建立一条安全的通信线路。

**1.37 HTTPs**

HTTPs是同处于应用层而基于TLS、SSL协议层之上的协议，也就是说，HTTPs是以安全为目标的HTTP通道，是HTTP的安全版本，安全基础是TLS协议

1. HTTPs协议需要向CA申请证书，一般免费的整数很少，需要缴费
2. HTTP是超文本传输协议，信息是明文传输，HTTPs则是具有安全性的SSL加密传输协议
3. 端口不一样，HTTPs端口是443，HTTP端口是80
4. HTTP连接很简单，无状态，而HTTPs是可进行加密传输、身份认证的网络协议，比HTTP安全
5. HTTPs应用得少，比较耗性能，对安全性要求没这么高要求的应用来说，HTTP就够了



HTTP 有以下安全性问题：

1. 通信使用明文，内容可能会被窃听；
2. 不验证通信方的身份，因此有可能遭遇伪装；
3. 无法证明报文的完整性，所以有可能已遭篡改。

HTTPs 并不是新协议，而是 HTTP 先和 SSL（Secure Socket Layer）通信，再由 SSL 和 TCP 通信。通过使用 SSL，HTTPs 提供了加密、认证和完整性保护。

认证

通过使用 证书 来对通信方进行认证。证书中有公开密钥数据，如果可以验证公开密钥的确属于通信方的，那么就可以确定通信方是可靠的。

数字证书认证机构（CA，Certificate Authority）颁发的公开密钥证书，可以通过 CA 对其进行验证。

进行 HTTPs 通信时，服务器会把证书发送给客户端，客户端取得其中的公开密钥之后，就可以开始加密过程。

使用 OpenSSL 这套开源程序，每个人都可以构建一套属于自己的认证机构，从而自己给自己颁发服务器证书。浏览器在访问该服务器时，会显示“无法确认连接安全性”或“该网站的安全证书存在问题”等警告消息。

**1.38 HTTP1.1 和 1.0的区别**

    HTTP1.1默认持久连接节省通信量，只要客户端服务端任意一端没有明确提出断开TCP连接，就一直保持连接，可以发送多次HTTP请求

    管线化，客户端可以同时发出多个HTTP请求，而不用一个个等待响应

    断点续传

    HTTP 1.1增加host字段

    100(Continue) Status(节约带宽)  客户端事先发送一个只带头域的请求，如果服务器因为权限拒绝了请求，就回送响应码            401（Unauthorized）

    HTTP/1.1在1.0的基础上加入了一些cache的新特性，当缓存对象的Age超过Expire时变为stale对象，cache不需要直接抛弃stale对象，而是与源服务器进行重新激活（revalidation）。

**1.39 HTTP2.0 的特点**

a、HTTP/2采用二进制格式而非文本格式

b、HTTP/2是完全多路复用的，而非有序并阻塞的——只需一个HTTP连接就可以实现多个请求响应

c、使用报头压缩，HTTP/2降低了开销

d、HTTP/2让服务器可以将响应主动“推送”到客户端缓存中

**1.40 CSRF( 跨站请求伪造)、XSS(跨站脚本攻击)**

xss：用户过分信任网站，放任来自浏览器地址栏代表的那个网站代码在自己本地任意执行。如果没有浏览器的安全机制限制，xss代码可以在用户浏览器为所欲为；

csrf：网站过分信任用户，放任来自所谓通过访问控制机制的代表合法用户的请求执行网站的某个特定功能。

xss(盗用用户的身份)原理上利用的是浏览器**可以拼接成任意的javascript**，然后**黑客拼接好javascript**让浏览器自动地给服务器端发出多个请求（get、post请求）。

csrf(服务端程序员背锅，没验证好)原理上利用的是网站服务器端所有参数都是**可预先构造**的原理，然后**黑客拼接好具体请求url**，可以引诱你提交他构造好的请求。