操作系统

进程和线程的区别

进程是资源拥有的基本单位，线程是独立调度与独立运行的基本单位。

进程是程序运行时的实例，每个进程都有各自独立的地址空间，一个进程不能访问另一个进程的内存。

线程存在于进程中，同一个进程中的多个线程可以共享进程中的堆空间，但是每个线程都有各自的寄存器和栈。

进程的状态转换

就绪状态：CPU时间片用完了由 运行状态 转换为 就绪状态

运行状态：获得CPU调度时由 就绪状态 转换为 运行状态

阻塞状态：运行状态 因等待某个事件发生而进入 阻塞状态，事件发生后由 阻塞状态 转换为 就绪状态

进程的互斥和同步

互斥：是指某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的。

同步：是指**在互斥的基础上（大多数情况），通过其它机制实现访问者对资源的有序访问**。在大多数情况下，同步已经实现了互斥，特别是所有写入资源的情况必定是互斥的。少数情况是指可以允许多个访问者同时访问资源。

简单地说：**同步体现的是一种协作性，互斥体现的是一种排他性**。

进程间的通信方式

管道+命名管道+信号+信号量+消息队列+共享内存+套接字

（1） 管道(pipe)：管道是一种半双工的通信方式，数据只能单向流动，而且**只能在具有亲缘关系的进程间使用**。进程的亲缘关系通常是指父子进程关系。

（2）命名管道 (named pipe)： 命名管道也是半双工的通信方式，但是**它允许无亲缘关系进程间的通信**。

（3）信号量(semophore)： 信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，**主要作为进程间以及同一进程内不同线程之间的同步手段**。

（4） 消息队列(message queue)： 消息队列是由消息的链表，存放在内核中并由消息队列标识符标识。消息队列克服了信号传递信息少、管道只能承载无格式字节流以及缓冲区大小受限等缺点。

（5）信号 (signal)： 信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。

（6）共享内存(shared memory)： 共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。**共享内存是最快的 IPC 方式，它是针对其他进程间通信方式运行效率低而专门设计的**。它往往与其他通信机制，如信号量，配合使用，来实现进程间的同步和通信。

（7）套接字(socket)： 套接字也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同终端及其间的进程通信。

进程(作业)调度算法

（1）**先来先服务（FCFS，First-Come-First-Served）**：此算法的原则是按照作业到达后备作业队列（或进程进入就绪队列）的先后次序来选择作业（或进程）。

（2）**短作业优先（SJF，Shortest Process Next）**：这种调度算法主要用于作业调度，它从作业后备队列中挑选所需运行时间（估计值）最短的作业进入主存运行。

（3）**时间片轮转调度算法（RR，Round-Robin）**：当某个进程执行的时间片用完时，调度程序便停

止该进程的执行，并将它送就绪队列的末尾，等待分配下一时间片再执行。然后把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片。这样就可以保证就绪队列中的所有进程，在一给定的时间内，均能获得一时间片处理机执行时间。

（4）**高响应比优先（HRRN，Highest Response Ratio Next）**：按照高响应比（（已等待时间＋要求运行时间）/ 要求运行时间）优先的原则，在每次选择作业投入运行时，先计算此时后备作业队列中每个作业的响应比RP然后选择其值最大的作业投入运行。

（5）**优先权(Priority)调度算法**: 按照进程的优先权大小来调度，使高优先权进程得到优先处理的调度策略称为优先权调度算法。注意：优先数越多，优先权越小。

（6）**多级队列调度算法**：多队列调度是根据作业的性质和类型的不同，将就绪队列再分为若干个子队列，所有的作业（或进程）按其性质排入相应的队列中，而不同的就绪队列采用不同的调度算法。

死锁产生的原因，死锁产生的必要条件是什么，如何预防死锁，如何避免死锁，死锁定理?

死锁产生的**原因**：(1)竞争资源；（2)进程推进顺序不当。

死锁产生的**必要条件**：

（1）**互斥条件**：一个资源一次只能被一个进程所使用，即是排它性使用。

（2）**不剥夺条件**：一个资源仅能被占有它的进程所释放，而不能被别的进程强占。

（3）**请求与保持条件**：进程已经保持了至少一个资源，但又提出了新的资源要求，而该资源又已被其它进程占有，此时请求进程阻塞，但又对已经获得的其它资源保持不放。

（4）**环路等待条件**：当每类资源只有一个时，在发生死锁时，必然存在一个进程—资源的环形链。

预防死锁：破坏四个必要条件之一。

死锁的避免：**银行家算法**，该方法允许进程动态地申请资源，系统在进行资源分配之前，先计算资源分配的安全性。若此次分配不会导致系统从安全状态向不安全状态转换，便可将资源分配给进程；否则不分配资源，进程必须阻塞等待，从而避免发生死锁。

银行家算法 http://www.cnblogs.com/diylab/archive/2008/03/25/1121770.html

死锁定理：S为死锁状态的充分条件是：当且仅当S状态的资源分配图是不可完全简化的，该充分条件称为死锁定理。

死锁的解除：

（1）方法1：强制性地从系统中撤消一个或多个死锁的进程以断开循环等待链，并收回分配给终止进程的全部资源供剩下的进程使用。

（2）方法2：使用一个有效的挂起和解除机构来挂起一些死锁的进程，其实质是从被挂起的进程那里抢占资源以解除死锁。

分段式存储管理 和 分页式存储管理的区别?

分页式存储管理：分页存储管理是将一个进程的地址（逻辑地址空间）空间划分成若干个大小相等的区域，称为页，相应地，将内存空间划分成与页相同大小（为了保证页内偏移一致）的若干个物理块，称为块或页框（页架）。在为进程分配内存时，将进程中的若干页分别装入多个不相邻接的块中。

分段式存储管理：在分段存储管理方式中，作业的地址空间被划分为若干个段，每个段是一组完整的逻辑信息，如有主程序段、子程序段、数据段及堆栈段等，每个段都有自己的名字，都是从零开始编址的一段连续的地址空间，各段长度是不等的。

两者的区别：

1.页是信息的物理单位，分页是为了实现非连续的分配，以便解决内存的碎片问题，或者说分页是为了系统管理的需要。

2.页的大小固定是由系统确定的，将逻辑地址划分为页号和页内地址是由机器硬件实现的。而段的长度是不固定的，决定与用户的程序长度，通常由编译程序进行编译时根据信息的性质来划分。

3.分页式存储管理的作业地址空间是一维的，分段式的存储管理的作业管理地址空间是二维的。

页面置换算法有哪些?

（1）**最佳置换算法（Optimal）**：即选择那些永不使用的，或者是在最长时间内不再被访问的页面置换出去。（它是一种理想化的算法，性能最好，但在实际上难于实现）。

（2）**先进先出置换算法FIFO**：该算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰。

（3）**最近最久未使用置换算法LRU**（Least Recently Used）：该算法是选择最近最久未使用的页面予以淘汰，系统在每个页面设置一个访问字段，用以记录这个页面自上次被访问以来所经历的时间T，当要淘汰一个页面时，选择T最大的页面。(**注：操作系统不一定是这么实现的，可能是维护一个优先队列来实现的**)

（4）**Clock置换算法**：也叫最近未用算法NRU（Not Recently Used）。该算法为每个页面设置一位访问位，将内存中的所有页面都通过链接指针链成一个循环队列。当某页被访问时，其访问位置“1”。在选择一页淘汰时，就检查其访问位，如果是“0”，就选择该页换出；若为“1”，则重新置为“0”，暂不换出该页，在循环队列中检查下一个页面，直到访问位为“0”的页面为止。由于该算法只有一位访问位，只能用它表示该页是否已经使用过，而置换时是将未使用过的页面换出去，所以把该算法称为最近未用算法。

（5）**最少使用置换算法LFU**：该算法选择最近时期使用最少的页面作为淘汰页。

生产者消费者问题

代码实现

哲学家就餐的问题

程序员面试金典 P298 代码实现

========

数据库

SQL的表连接方式

SQL中连接按结果集分为：内连接，外连接，全连接，交叉连接

内连接：inner join on，两表都满足的组合。内连接分为等值连接，不等连接，自然连接。

等值连接：两表中相同的列都会出现在结果集中。

自然连接：两表中具体相同列表的列会合并为同一列出现在结果集中。

外连接：分为左（外）连接，右（外）连接，全连接

左（外）连接：A left (outer) join B，以A表为基础，A表的全部数据，B表有的组合，没有的为null。

右（外）连接：A right (outer) join B，以B表为基础，B表的全部数据，A表有的组合，没有的为null。

全连接：A full (outer) join 两表相同的组合在一起，A表有，B表没有的数据（显示为null），同样B表有，A表没有的显示为null。

交叉连接：cross join，就是笛卡尔乘积，没有用where子句的交叉连接将产生连接所涉及的笛卡尔积第一个表的行数乘以第二个表的行数等于笛卡尔积和结果集的大小。

三范式

1NF：表中的字段都是单一属性，不再可分。

2NF：在1NF的基础上，表中所有的非主属性都必须完全依赖于任意一组候选键，不能仅依赖于候选键中的某个属性。

3NF：在2NF的基础上，表中所有的属性都不依赖其他非主属性。

简单的说就是：**1NF表示每个属性不可分割，2NF表示非主属性不存在对主键的部分依赖，3NF表示不存在非主属性对主键的依赖传递。**

知乎上关于三范式的理解 https://www.zhihu.com/question/24696366

表的操作

表的创建：create table 表名 (列名1 类型 约束，列名2 类型 约束...)
表的删除：drop table 表名
表的更改：alter table 表名 add|drop 列名|约束名
插入记录：insert into 表名...values...
更新记录：update 表名 set 列名=值 where 条件
删除记录：delete from 表名 where 条件

数据的完整性

数据完整性指的是存储在数据库中的数据的一致性和准确性。
完整性分类：
（1）实体完整性：主键值必须唯一且非空。（主键约束）
（2）引用完整性（也叫参照完整性）：外键要么为空，要么引用主表中存在的记录。（外键约束）。
（3）用户自定义完整性：针对某一具体关系数据库中的约束条件。

SQL的查询优化

（1）从表连接的角度优化：尽量使用内连接，因为内连接是两表都满足的行的组合，而外连接是以其中一个表的全部为基准。
（2）尽量使用存储过程代替临时写SQL语句：因为存储过程是预先编译好的SQL语句的集合，这样可以减少编译时间。
（3）从索引的角度优化：对那些常用的查询字段建立索引，这样查询时直接进行索引扫描，不读取数据块。
（4）还有一些常用的select优化技巧：
A 只查询那些需要访问的字段，来代替select * 这种查询方式
B 将过滤记录越多的where语句向前移：在一个SQL语句中，如果一个where条件过滤的数据库记录越多，定位越准确，则该where条件越应该前移。

索引的作用，聚集索引与非聚集索引的区别

索引是一个数据库对象，使用索引，可以使数据库程序无须对整个数据进行扫描，就可以在其中找到目标数据，从而提高查找效率。索引的底层采用的是B树。

聚集索引：根据记录的key在表中排序数据行。
非聚集索引：独立于记录的结构，非聚集索引包含的key，且每个键值项都有指向该键值的数据行的指针。
字典的例子：字典前面的目录，可以按照拼音和部首去查询，我们想查询一个字，只需要根据拼音或者部首去查询，就可以快速的定位到这个汉字了，这个就是索引的好处，拼音查询法就是聚集索引，部首查询就是一个非聚集索引。

聚集索引与非聚集索引的区别：
（1）聚集索引的物理存储按索引排序，非聚集所以的物理存储不按索引排序。
（2）聚集索引插入，更新数据的速度比非聚集索引慢，单查询速度更快。
（3）聚集索引的叶级结点保存的是时间的数据项，而非聚集结点的叶级结点保存的是指向数据项的指针。
（4）一个表只能有一个聚集索引（因为只有一种排序方式），但可以有多个非聚集索引。

存储过程与函数的区别
（1）函数有返回值，存储过程没有返回值。
（2）因为存储过程没有返回值，所以不能将存储过程的执行结果赋值给变量；函数有返回值类型，调用函数时，可以将函数的执行结果赋值给变量。也就是说，函数可以在select语句中使用，而存储过程则不能。
这里的函数是指SQL中的function，例如sum，average等，并不是指程序中的函数，它可以在存储过程中使用

========

计算机网络

OSI，TCP/IP，五层协议的体系结构
OSI分层（7层）：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。
TCP/IP分层（4层）：网络接口层、网络层、运输层、应用层。
五层协议（5层）：物理层、数据链路层、网络层、运输层、应用层。

[OSI参考模型是7层，TCP/IP模型并不包括物理层，其网络接口层下面就是物理网络，所以分成了4层，讲解时一般折中，采用五层协议的方式进行讲解]

TCP/IP模型和OSI参考模型的对应关系
网络接口层--对应OSI参考模型的物理层和数据链路层；[主要是物理地址寻址、数据成帧等处理]
网络层--对应OSI参考模型的网络层；[主要对数据包进行路由选择等]
运输层--对应OSI参考模型的运输层；[主要对数据包进行分段并处理拥塞控制和流量控制等]
应用层--对应OSI参考模型的5、6、7层。[主要对数据包进行校验、加密解密、格式转换和解析等]

每一层的作用如下：
物理层：激活、维持、关闭通信端点之间的机械特性、电气特性、功能特性以及过程特性。该层为上层协议提供了一个传输数据的物理媒体。
数据链路层：数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。
网络层：网络层负责对子网间的数据包进行**路由选择**。此外，网络层还可以实现拥塞控制、网际互连等功能。
传输层：第一个端到端，即主机到主机的层次。传输层负责将上层数据分段并提供端到端的、可靠的或不可靠的传输。此外，传输层还要处理端到端的**差错控制和流量控制**问题。
会话层：会话层管理主机之间的会话进程，即负责建立、管理、终止进程之间的会话。会话层还利用在数据中插入校验点来实现数据的同步。
表示层：表示层对上层数据或信息进行变换以保证一个主机应用层信息可以被另一个主机的应用程序理解。表示层的数据转换包括数据的加密、压缩、格式转换等。
应用层：为操作系统或网络应用程序提供访问网络服务的接口。

TCP/IP体系的四个层次
TCP/IP体系共有四个层次：应用层（Application Layer）、传输层（Transport Layer）、网络互连层（Internet Layer）和网络接口层（Host-to-Network Layer）

（1）网络接口层（Host-to-Network Layer）=> 接收和发送数据报
网络接口层主要负责将数据报发送到网络传输介质上以及从网络上接收TCP/IP数据报，相当于OSI参考模型中的物理层和数据链路层。
在实际中，先后流行的以太网、令牌环网、ATM、帧中继等都可视为其底层协议。它将发送的信息组帧，并通过物理层向选定网络发送，或者从网络上接收物理帧，将去除数据控制信息的IP数据报交给

网络互连层。

（2）网络互连层（Internet Layer）=> 数据报封装和路由寻址功能

网络互连层的主要功能是寻址和对数据报的封装以及重要的路由选择功能。

这些功能大部分都是由IP协议来完成的，再加上地址解析协议（Address Resolution Protocol，ARP）、因特网控制报文协议（Internet Control Message Protocol，ICMP）等协议从旁协助，所以IP协议是本层众多实体中的核心。下面简单介绍这几个协议。

网际协议（Internet Protocol，IP）。该协议是一个无连接的协议，主要负责将数据报从源结点转发到目的结点。也就是说，IP协议通过对每个数据报中都有的源地址和目的地址进行分析，然后进行路由选择（即选择一条到达目标的最佳路径），最后再转发到目的地。需要注意的是：IP协议只是负责对数据进行转发，并不对数据进行检查。也就是说，它不负责数据的可靠性，这样设计的主要目的是提高IP协议传送和转发数据的效率。

地址解析协议（Address Resolution Protocol，ARP）。该协议主要负责将TCP/IP网络中的IP地址解析和转换成计算机的物理地址，以便于物理设备（如网卡）按该地址来接收数据。

反向地址解析协议（Reverse Address Resolution Protocol，RARP）。该协议的作用与ARP的作用相反，它主要负责将设备的物理地址解析和转换成IP地址。

因特网控制报文协议（Internet Control Message Protocol，ICMP）。该协议主要负责发送和传递包含控制信息的数据报，这些控制信息包括哪台计算机出了什么错误、网络路由出现了什么错误等内容。

（3）传输层（Transport Layer）=> 应用进程间的端到端通信

传输层主要负责在应用进程之间的端到端通信，即从某个应用进程传输到另外一个应用进程。它与OSI参考模型的传输层功能类似，也对高层屏蔽了底层通信网络的实现细节。

传输层在某一时刻可能要同时为多个不同的应用进程服务，因此为了识别不同的应用进程，传输层在每一个分组中必须增加用于识别信源和信宿的应用程序的标识，同时，每一个分组还要附带校验和，以保证接收端能校验分组的正确性，这样可以将数据报发送到合适的应用进程。这个增加的标识称为端口（port）或者端口号（port ID）。

TCP/IP体系结构的传输层包含两个主要协议，即传输控制协议（Transport Control Protocol，TCP）和用户数据报协议（User Datagram Protocol，UDP）。这两个协议分别应用于有不同要求的应用进程。

（4）应用层（Application Layer）=> 不同协议

应用层是TCP/IP的最高层，它包括了多种高层协议，并且总有新的协议加入。与OSI的应用层类似，它是直接为应用进程服务的一层。即**当不同的应用进程数据通信或者数据交换时，就去调用应用层的不同协议实体，让这些实体去调用TCP或者UDP层服务来进行网络传输。**

与OSI不同，TCP/IP中包含了许多具体的应用层协议。

简单邮件传输协议（Simple Mail Transportation Protocol，SMTP）：该协议主要用于在电子邮件服务器之间传输电子邮件。

域名系统（Domain Name System，DNS）：该协议用于域名与IP地址之间的转换。

超文本传输协议（Hypertext Transportation Protocol，HTTP）：该协议是为因特网上传输和处理超文本或者WWW（World Wide Web）页面而服务的应用层协议。

文件传输协议（File Transportation Protocol，FTP）：该协议是为网络中传输文件进程服务的，所谓传输文件，是指将文件从一台计算机通过网络复制到另一台计算机中。

远程终端协议（Telnet）：该协议是用于远程登录网络主机的一个应用层协议。

简单网络管理协议（Simple Network Management Protocol，SNMP）：该协议用于在控制台与网络设备（如路由器、交换机等）之间交换网络管理信息。

TCP与UDP的区别

TCP协议是一种可靠的、面向连接的协议，保证通信主机之间有可靠的字节流传输，完成流量控制功能，协调收发双方的发送与接收速度，达到正确传输的目的。

UDP是一种不可靠、无连接的协议，其特点是协议简单、额外开销小、效率较高，但是不能保证传输是否正确。

UDP是面向无连接的、不可靠的数据报服务；TCP是面向连接的、可靠的字节流服务。

TCP对应的协议和UDP对应的协议

TCP对应的协议：

（1） FTP：文件传输协议，默认使用21端口。

（2） Telnet：一种用于远程登陆的端口，用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于DOS模式下的通信服务，默认使用23端口。

（3） SMTP：简单邮件传送协议，用于发送邮件，默认使用25号端口。

（4） POP3：和SMTP对应，POP3用于接收邮件，默认使用110端口。

（5）HTTP协议：从Web服务器传输超文本到本地浏览器的传送协议，默认使用80端口。

UDP对应的协议：

（1） DNS：域名解析服务，将域名地址转换为IP地址，默认使用53号端口。

（2） SNMP：简单网络管理协议，默认使用161号端口，是用来管理网络设备的。

（3） TFTP(Trival File Transfer Protocal)：简单文件传输协议，默认使用69号端口。

TCP/IP协议的特点

1.高可靠性

TCP/IP采用重新确认的方法保证数据的可靠传输，并采用窗口流量控制机制使可靠性得到进一步保证。

TCP的可靠性如何保证?

TCP的可靠性是通过顺序编号（SEQ）和确认（ACK）来实现的。

2.安全性

为建立TCP连接，在连接的每一端都必须与该连接的安全性控制达成一致。**IP在它的控制分组头中有若干字段允许有选择地对传输的信息实施保护。**

3.灵活性

TCP/IP要求下层支持该协议，而对上层应用协议不作特殊要求。因此，TCP/IP的使用不受传输介质和网络应用软件的限制。

TCP流量控制和拥塞控制

流量控制：滑动窗口协议

所谓流量控制就是让发送方的发送速率不要太快，要让接收方来得及接收。

TCP滑动窗口：TCP滑动窗口用来暂存两台主机间要传送的数据，有点类似CACHE。

每个TCP/IP主机有两个滑动窗口：一个用于接收数据，另一个用于发送数据。

通过每个TCP传输的字段指定顺序号，以获得可靠性。如果一个分段被分解成几个小段，接收主机会知道是否所有小段都已收到。通过发送应答，用以确认别的主机收到了数据。**对于发送的每一个小段，接收主机必须在一个指定的时间返回一个确认。如果发送者未收到确认，数据会被重新发送；如果收到的数据包损坏，接收主机会舍弃它，因为确认未被发送，发送者会重新发送分段。**

设A向B发送数据。在连接建立时，B告诉了A：“我的接收窗口是 rwnd = 400 ”(这里的 rwnd 表示 receiver window) 。因此，发送方的发送窗口不能超过接收方给出的接收窗口的数值。请注意，TCP的窗口单位是字节，不是报文段。

拥塞控制

拥塞控制：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。拥塞控制所要做的都有一个前提：网络能够承受现有的网络负荷。拥塞控制是一个全局性的过程，涉及到所有的主机、路由器，以及与降低网络传输性能有关的所有因素。

拥塞控制方法

慢开始(slow-start)和拥塞避免(congestion avoidance)

发送方维持一个拥塞窗口 cwnd (congestion window)的状态变量。拥塞窗口的大小取决于网络的拥塞程度，并且动态地在变化。发送方让自己的发送窗口等于拥塞窗口的大小。

发送方控制拥塞窗口的原则是：只要网络没有出现拥塞，拥塞窗口就再增大一些，以便把更多的分组发送出去。但只要网络出现拥塞，拥塞窗口就减小一些，以减少注入到网络中的分组数。

慢开始算法：每经过一个传输轮次，拥塞窗口 cwnd 就加倍。慢开始的"慢"并不是指cwnd的增长速率慢，而是指在TCP开始发送报文段时先设置cwnd=1，使得发送方在开始时只发送一个报文段（目的是试探一下网络的拥塞情况），然后再逐渐增大cwnd。

为了防止拥塞窗口cwnd增长过大引起网络拥塞，还需要设置一个慢开始门限ssthresh状态变量（如何设置ssthresh）。慢开始门限ssthresh的用法如下：

当 cwnd < ssthresh 时，使用上述的慢开始算法。

当 cwnd > ssthresh 时，停止使用慢开始算法而改用拥塞避免算法。

当 cwnd = ssthresh 时，既可使用慢开始算法，也可使用拥塞避免算法。

拥塞避免算法：让拥塞窗口cwnd缓慢地增大，即每经过一个往返时间RTT就把发送方的拥塞窗口cwnd加1，而不是加倍。这样拥塞窗口cwnd按线性规律缓慢增长，比慢开始算法的拥塞窗口增长速率缓慢得多。

无论在慢开始阶段还是在拥塞避免阶段，只要发送方判断网络出现拥塞（其根据就是没有收到确认），就要把慢开始门限ssthresh设置为出现拥塞时的发送方窗口值的一半（但不能小于2）。然后把拥塞窗口cwnd重新设置为1，执行慢开始算法。这样做的目的就是要迅速减少主机发送到网络中的分组数，使得发生拥塞的路由器有足够时间把队列中积压的分组处理完毕。

快重传(fast retransmit)和快恢复(fast recovery)

快重传算法：要求接收方每收到一个失序的报文段后就立即发出重复确认（为的是使发送方及早知道有报文段没有到达对方）而不要等到自己发送数据时才进行捎带确认。

快重传算法还规定，发送方只要一连收到三个重复确认就应当立即重传对方尚未收到的报文段，而不必继续等待设置的重传计时器到期。

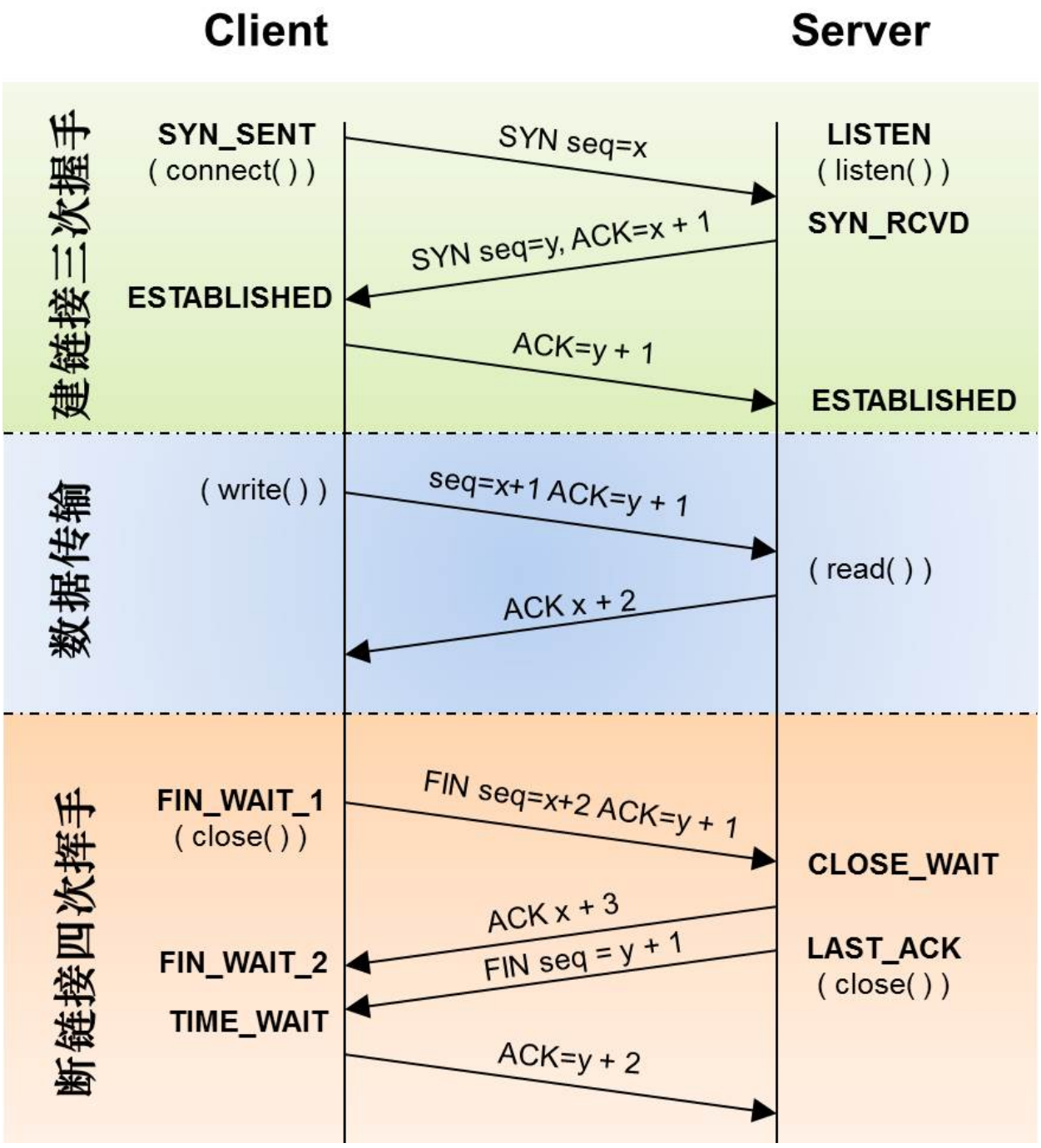
快恢复过程

1.当发送方连续收到三个重复确认，就执行"乘法减小"算法，把慢开始门限ssthresh减半。这是为了预防网络发生拥塞。请注意：接下去不执行慢开始算法。

2.由于发送方现在认为网络很可能没有发生拥塞，因此与慢开始不同之处是现在不执行慢开始算法（即拥塞窗口cwnd现在不设置为1），而是把cwnd值设置为慢开始门限ssthresh减半后的数值，然后开始执行拥塞避免算法（"加法增大"），使拥塞窗口缓慢地线性增大。

TCP三次握手和四次挥手的全过程

连接建立：三次握手



首先Client端发送连接请求报文，Server端接收连接请求后回复ACK报文，并为这次连接分配资源。Client端接收到ACK报文后也向Server端发生ACK报文，并分配资源，这样TCP连接就建立了。

连接断开：四次握手

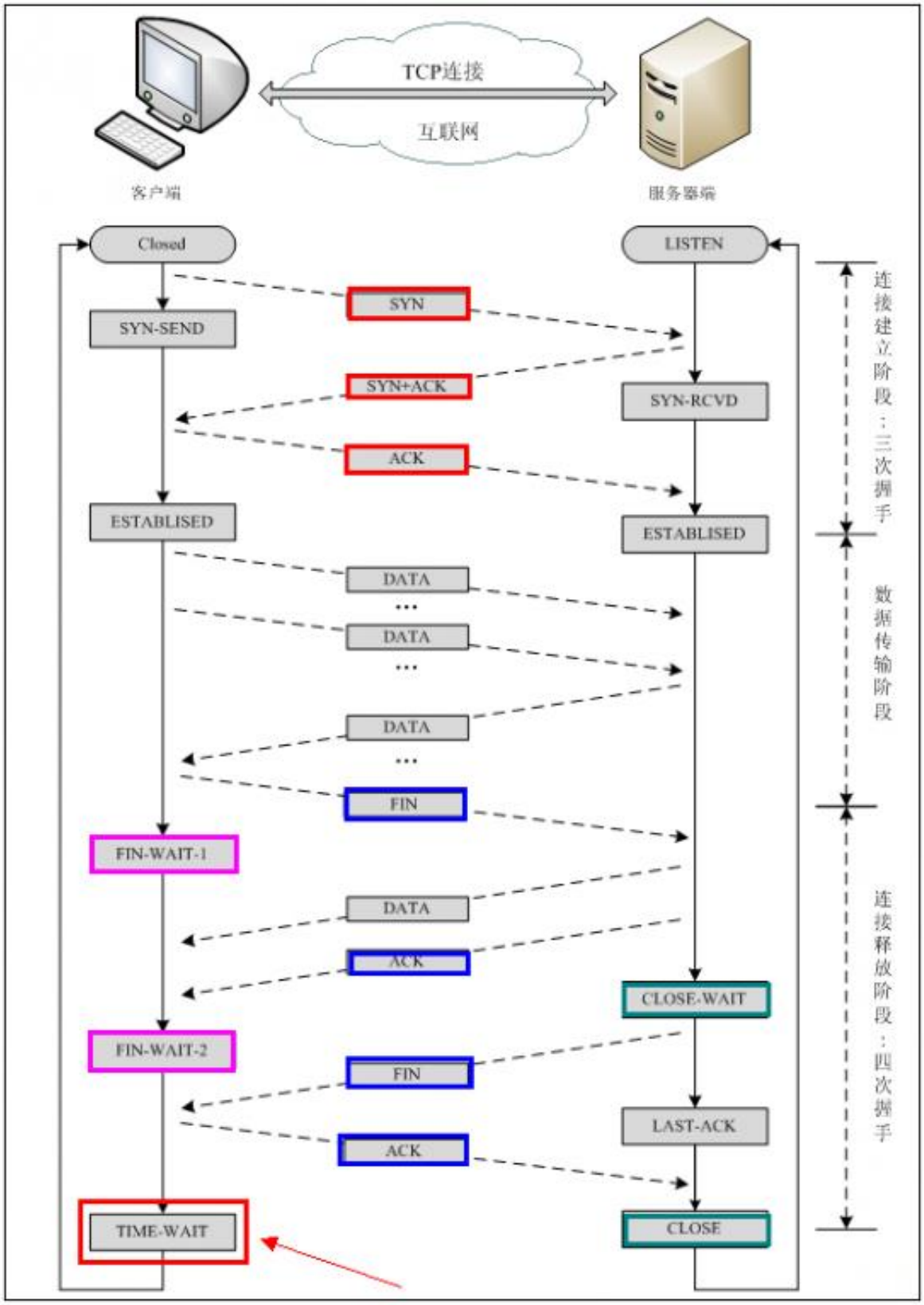
假设Client端发起中断连接请求，也就是发送FIN报文。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据，所以你先发送ACK，"告诉Client端，你的请求我收到了，但是我还没准备好，请你继续等我的消息"。这个时候Client端就进入**FIN_WAIT**状态，继续等待Server端的FIN报文。当Server端确定数据已发送完成，则向Client端发送FIN报文，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。Client端收到FIN报文后就知道可以关闭连接了，但是它还是不相信网络，怕Server端不知道要关闭，所以发送ACK后进入**TIME_WAIT**状态，如果Server端没有收到ACK则可以重传。Server端收到ACK后，就知道可以断开连接了。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，Client端也可以关闭连接了。TCP连接就这样关闭了！

简单来说，两端都要发送一次FIN，并得到对方的ACK，这样才可以放心地断开连接。

由于TCP连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务后就能发送一个FIN来终止这个方向的连接。**收到一个 FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据**。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

为什么连接的时候是三次握手，关闭的时候却是四次握手?
因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Clien端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步撒手。

为什么TIME_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态?
这是因为虽然双方都同意关闭连接了，而且握手的4个报文也都协调和发送完毕，按理可以直接回到CLOSED状态（就好比从SYN_SEND状态到ESTABLISH状态那样）。但是因为我们必须要假想网络是不可靠的，你无法保证你最后发送的ACK报文会一定被对方收到，因此**对方处于LAST_ACK状态下的SOCKET可能会因为超时未收到ACK报文，而重发FIN报文，所以这个TIME_WAIT状态的作用就是用来重发可能丢失的ACK报文。**
如果client端最后发送的ACK报文丢失了，server端会重发FIN报文，client端收到后继续发ACK报文。



在浏览器中输入www.baidu.com 后执行的全部过程

现在假设如果我们在客户端浏览器中输入http://www.baidu.com ，而baidu.com为要访问的服务器，下面详细分析客户端为了访问服务器而执行的一系列关于协议的操作：

1、客户端浏览器请求DNS服务器解析域名www.baidu.com 对应的IP地址，然后通过这个IP地址和默认端口80，和服务器建立TCP连接，连接建立之后通过TCP将HTTP会话封装成数据包。

2、在客户端的传输层，把HTTP会话请求分成报文段，添加源和目的端口（如服务器使用80端口监听客户端的请求，客户端由系统随机选择一个端口如5000，与服务器进行交换，服务器把相应的请求返回给客户端的5000端口）然后使用IP层的IP地址查找目的端。

3、在客户端的网络层，通过查找路由表确定如何到达服务器，期间可能经过多个路由器，这些都是由路由器来完成的工作，主要是通过查找路由表来决定通过哪个路径到达服务器。
4、在客户端的链路层，数据包通过链路层发送到路由器，通过邻居协议查找给定IP地址的MAC地址，然后发送ARP请求查找目的地址，如果得到回应后就可以使用ARP的请求应答交换的IP数据包现在就可以传输了，然后发送IP数据包到达服务器的地址。

HTTP协议包括哪些请求?
HTTP六种请求方法：get，post，update，delete，head，options
GET：请求读取由URL所标志的数据
POST：给服务器添加或者更新数据
PUT：在给定的URL下存储一个文档
DELETE：删除给定的URL所标志的资源
OPTIONS：服务器针对特定资源所支持的HTTP请求方法
HEAD：向服务器索要与GET请求相一致的响应，只不过响应体将不会被返回。这一方法可以在不必传输整个响应内容的情况下，就可以获取包含在响应消息头中的元信息

HTTP中POST与GET的区别
GET是从服务器上获取数据，POST是向服务器传送数据。
GET是将请求参数加到URL中，POST是将请求数据放在请求体中。
GET传送的数据量较小，不能超过2KB(1024字节?)，POST传送的数据量较大，默认为不受限制。

HTTP协议的格式
HTTP请求包含的内容：1.请求行 2.HTTP头 3.内容
第一部分请求行写法是固定的，由三部分组成，第一部分是请求方法，第二部分是请求网址，第三部分是HTTP版本。
第二部分包含的头包括：1.请求头(request header) 2.普通头(general header) 3.实体头(entity header)。通常来说，由于GET请求往往不包含内容实体，因此也不会有实体头。
第三部分内容只在POST请求中存在，因为GET请求并不包含任何实体。

HTTP响应包含的内容：1.状态行 2.HTTP头 3.内容
第一部分是HTTP版本，第二部分是响应状态码，第三部分是状态码的描述，因此也可以把第二和第三部分看成一个部分。
信息类 (100-199) 响应成功 (200-299) 重定向类 (300-399) 客户端错误类 (400-499) 服务端错误类 (500-599)
第二部分包含的头包括：1.响应头(response header) 2.普通头(general header) 3.实体头(entity header)。
第三部分HTTP响应内容就是HTTP请求所请求的信息。这个信息可以是一个HTML，也可以是一个图片。

HTTP头的分类
HTTP头并不是严格要求的，仅仅是一个标签，如果浏览器可以解析就会按照某些标准（比如浏览器自身标准，W3C的标准）去解释这个头，否则不识别的头就会被浏览器无视。对服务器也是同理。
通用头(General header)
通用头即可以包含在HTTP请求中，也可以包含在HTTP响应中。通用头的作用是描述HTTP协议本身。比如描述HTTP是否持久连接的Connection头，HTTP发送日期的Date头，描述HTTP所在TCP连接时间的Keep-Alive头，用于缓存控制的Cache-Control头等。
实体头(Entity header)
实体头是那些描述HTTP信息的头。既可以出现在HTTP POST方法的请求中，也可以出现在HTTP响应中。比如Content-Type和Content-length都是描述实体的类型和大小的头都属于实体头。其它还有用

于描述实体的Content-Language、Content-MD5、Content-Encoding以及控制实体缓存的Expires和Last-Modifies头等。

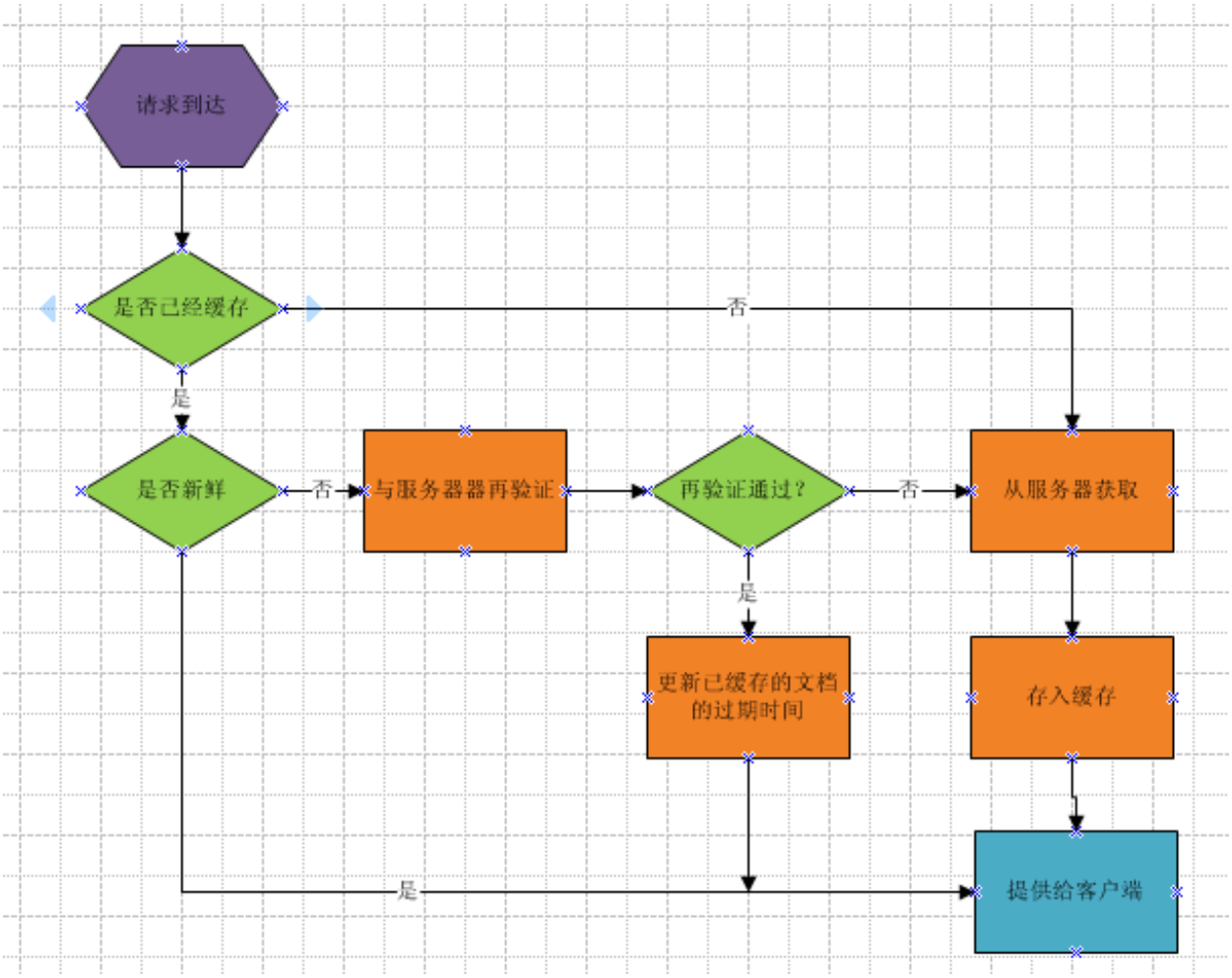
请求头(HTTP Request Header)

请求头是那些由客户端发往服务端以便帮助服务端更好的满足客户端请求的头。请求头只能出现在HTTP请求中。比如告诉服务器只接收某种响应内容的Accept头，发送Cookies的Cookie头，显示请求主机域的HOST头，用于缓存的If-Match、If-Match-Since、If-None-Match头，用于只取HTTP响应信息中部分信息的Range头，用于附属HTML相关请求引用的Referer头等。

响应头(HTTP Response Header)

HTTP响应头是那些描述HTTP响应本身的头，这里面并不包含描述HTTP响应中第三部分也就是HTTP信息的头（这部分由实体头负责）。比如说定时刷新的Refresh头，当遇到503错误时自动重试的Retry-After头，显示服务器信息的Server头，设置COOKIE的Set-Cookie头，告诉客户端可以部分请求的Accept-Ranges头等。

HTTP缓存



HTTP缓存的处理流程：

1）请求处理

用户发起一个http请求，缓存获取到URL，根据URL查找是否有匹配的副本，这个副本可能在内存中，也可能在本地磁盘。

2） 新鲜度检测

如果缓存中存在所请求资源的副本，则进行新鲜度检测。新鲜度检测举个简单的例子，我们在商店买了一瓶汽水，汽水瓶上肯定会标有过期时间，我们会根据这个过期时间和现在的时间做对比，看看饮料过期了没，如果没过期，我们正常喝就行了，如果已经过期，我们肯定要找商家。其实这就是一个新鲜度检测的过程，HTTP请求的新鲜度检测流程也是这样的，HTTP发起一个请求时，发现缓存中有

相应的副本，接着就会检查这个副本有没有过期，如果没有过期，直接使用。如果已经过期，则进行再验证。

3）服务器再验证

缓存中的文档过期了并不代表它和服务器上的不一样，所以这个时候就需要问问服务器，过期的这段时间里这个文档到底有没有改变。如果改变了，缓存就会获取一份新的文档副本，然后发送给客户端。如果没有改变，缓存只需要获取新的首部，包括一个新的过期时间，并对缓存中的首部更新。

4）创建响应并返回

我们希望缓存看起来就像是来自原始服务器一样，缓存将已缓存的服务器响应首部作为响应首部，发送给客户端。

保质期的实现

HTTP中，通过**Cache-Control**首部和**Expires**首部为文档指定了过期时间，通过对过期时间的判断，缓存就可以知道文档是不是在保质期内。Expires首部和Cache-Control:max-age 首部都是来告诉缓存文档有没有过期，为什么需要两个响应首部来做这件简单的事情了？其实这一切都是历史原因，Expires首部是HTTP 1.0中提出来的，因为他使用的是绝对日期，如果服务端和客户端时钟不同步的话（实际上这种情况非常常见），缓存可能就会认为文档已经过了保质期。

HTTP 1.1为了修正这个问题，引入了Cache-Control:max-age 首部，这个首部使用相对时间来控制保质期，让一切变得更加合理。举个例子，我们买了一瓶汽水，如果使用Expires首部来标注保质期，就会这么写：饮料过期时间：2012年12月21日，如果某个2货不知道今天多少号，他还真不知道这饮料过期没，我小时候饮料都这么写。后来，有个挺有名的卖牛奶的，大概就叫蒙牛，他发明了一种标注保质期的方法，他怎么搞了？他这么写：保质期：12个月，行，牛逼了，我牛奶一年前就生产出来的牛奶，今天要发给厂家，发之前，先往包装上印上生产日期（当然是印发货那天），然后告诉你，明年才过期，这多聪明，搞成相对的，毒死你。也许HTTP 1.1借鉴了这个伟大的发明，于是就有了Cache-Control:max-age 首部。

服务器再验证的实现

缓存要问问服务器，牛奶已经过期了，到底还能不能喝。我说错了，是文档，不是牛奶。HTTP中，使用两个请求首部来完成这个功能：**If-Modified-Sice**和**If-None-Match**。为啥又要两个首部来完成这个功能了？答案还是因为历史的原因。一开始使用 If-Modified-Sice:首部，date是上一次缓存牛奶时响应中**Last-Modified**首部的值。

客户端拿着这个值，问服务器，这段时间内这个牛奶你有没有修改过？服务器看了看这个牛奶的修改时间，如果没有修改过，会返回一个**304 Not Modified**的响应；如果修改过，把最新的牛奶返回给客户端。后来，人们发现这样有问题，因为就算修改时间变化了，文档也不一定发生改变！于是乎，就有了 If-None-Match:首部，tag是上一次缓存文档时响应中Etag的值，**Etag是一种唯一标识资源的方式**，就像java中的hashcode，如果hashcode不一样，那么两个对象肯定不一样！

试探性过期

如果响应中既没有Cache-Control:max-age 首部又没有Expires首部，缓存可以计算出一个试探性最大使用期。这东西打个比方就是缓存会根据响应的Last-Modified来决定文档靠不靠谱，需不需要再验证，如果Last-Modified中的日期是很早之前，那缓存就认为这文档挺靠谱，近期之内应该不会变化；如果Last-Modified中的日期是最近几天，那缓存可能就认为这文档可能经常改变，不靠谱。当然这么粗略的判断想想就知道不严谨，所以我们一定要设置Expires首部和Cache-Control首部。

HTTP和HTTPS的区别

HTTPS(Secure Hypertext Transfer Protocol)安全超文本传输协议

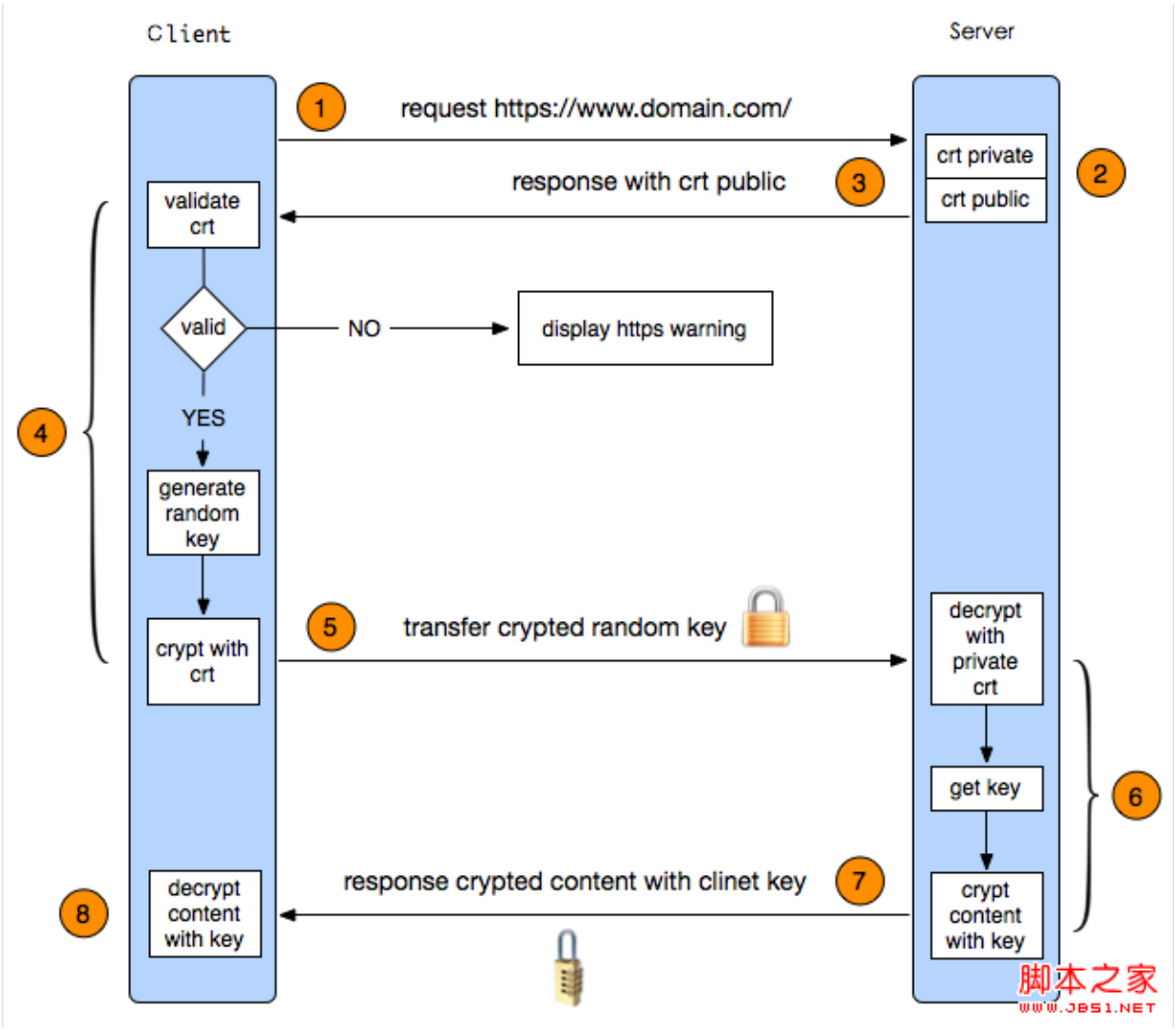
它是一个安全通信通道，基于HTTP开发，用于在客户计算机和服务器之间交换信息。它使用安全套接字层(Secure Socket Layer)进行信息交换，简单来说它是HTTP的安全版。

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。

HTTP和HTTPS使用的是完全不同的连接方式用的端口也不一样：前者是80，后者是443。

HTTP的连接很简单，是无状态的。HTTPS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，要比HTTP协议安全，一般需要到CA申请证书，免费证书很少，需要交费。

HTTPS连接建立



中间人攻击（Man-in-the-Middle Attack = MTMA）

中间人攻击：中间人获取server发给client的公钥，自己伪造一对公私钥，然后伪造自己让client以为它是server，然后将伪造的公钥发给client，并拦截client发给server的密文，用伪造的私钥即可得到client发出去的内容，最后用真实的公钥对内容进行加密发给server。

解决办法：数字证书，可信任的中间人 证书链

IP地址的分类 (点分四段十进制)

A类地址以00开头，第一个字节作为网络号，地址范围是：0.0.0.0~127.255.255.255 / 8；

B类地址以10开头，前两个字节作为网络号，地址范围是：128.0.0.0~191.255.255.255 / 16；

C类地址以110开头，前三个字节作为网络号，地址范围是：192.0.0.0~223.255.255.255 / 24；

D类地址以1110开头，地址范围是224.0.0.0~239.255.255.255，D类地址作为组播地址；

E类地址以1111开头，地址范围是240.0.0.0~255.255.255.255，E类地址为保留地址，供以后使用。

注：只有A，B，C有网络号和主机号之分，D类地址和E类地址没有划分网络号和主机号。

=> 0 (A) +128=128 (B) +64=192 (C) +32=224 (D) +16=240 (E)

=> 增量从 128 到 64 到 32 到 16，每次是上次的一半

特殊的IP地址

（1）网络地址

IP地址由网络号（包括子网号）和主机号组成，网络地址的主机号全为0，网络地址代表着整个网络。[IP地址=网络号+主机号，网络地址是主机号全为0的特殊情况]

（2）广播地址

广播地址通常称为直接广播地址，是为了区分受限广播地址。

广播地址与网络地址的主机号正好相反，广播地址中，主机号为全1。当向某个网络的广播地址发送消息时，该网络内的所有主机都能收到该广播消息。[广播地址是主机号全为1的特殊情况]

（3）组播地址

D类地址就是组播地址，提供一对多的通信方式。

（4）255.255.255.255

该IP地址指的是受限的广播地址。受限广播地址与一般广播地址（直接广播地址）的区别在于，受限广播地址只能用于本地网络，路由器不会转发以受限广播地址为目的地址的分组；一般广播地址既可在本地广播，也可跨网段广播。例如：主机192.168.1.1/30上的直接广播数据包后，另外一个网段192.168.1.5/30也能收到该数据报；若发送受限广播数据报，则不能收到。

注：一般的广播地址（直接广播地址）能够通过某些路由器（当然不是所有的路由器），而受限的广播地址不能通过路由器。

（5）0.0.0.0

常用于寻找自己的IP地址，例如在RARP，BOOTP和DHCP协议中，若某个未知IP地址的主机想要知道自己的IP地址，它就以255.255.255.255为目的地址，向本地范围（具体而言是被各个路由器屏蔽的范围内）的服务器发送IP请求分组。

（6）回环地址

127.0.0.0/8被用作回环地址，回环地址表示本机的地址，常用于对本机的测试，用的最多的是127.0.0.1。

（7）A、B、C类私有地址

私有地址(private address)也叫专用地址，它们不会在全球使用，只具有本地意义。

A类私有地址：10.0.0.0/8，范围是：10.0.0.0~10.255.255.255

B类私有地址：172.16.0.0/12，范围是：172.16.0.0~172.31.255.255

C类私有地址：192.168.0.0/16，范围是：192.168.0.0~192.168.255.255

分配IP地址时需要注意什么?

在分配网络号和主机号时应遵守以下几条准则：

1.网络号不能是127，它被用作回环地址

2.不能将主机号都设置为1，这是广播地址

3.不能将主机号都设置为0，这是网络地址

4.对于本网络来说，主机号应该唯一

NAT协议、DHCP协议、DNS协议的作用

NAT协议：网络地址转换(NAT，Network Address Translation)属接入广域网(WAN)技术，是一种将私有（保留）地址转化为合法IP地址的转换技术，它被广泛应用于各种类型Internet接入方式和各种类型的网络中。原因很简单，NAT不仅完美地解决了IP地址不足的问题，而且还能够有效地避免来自网络外部的攻击，隐藏并保护网络内部的计算机。

DHCP协议：动态主机设置协议（Dynamic Host Configuration Protocol， DHCP）是一个局域网的网络协议，使用UDP协议工作，主要有两个用途：给内部网络或网络服务供应商自动分配IP地址，给用户或者内部网络管理员作为对所有计算机作中央管理的手段。

DNS协议：DNS 是域名系统 (Domain Name System) 的缩写，是因特网的一项核心服务，它作为可以将域名和IP地址相互映射的一个分布式数据库，能够使人更方便的访问互联网，而不用去记住能够被机器直接读取的IP地址。

ARP协议的工作原理
首先，每台主机都会在自己的ARP缓冲区中建立一个 ARP列表，以表示IP地址和MAC地址的对应关系。当源主机需要将一个数据包要发送到目的主机时，会首先检查自己 ARP列表中是否存在该 IP地址对应的MAC地址，如果有，就直接将数据包发送到这个MAC地址；如果没有，就**向本地网段发起一个ARP请求的广播包，查询此目的主机对应的MAC地址**。此ARP请求数据包里包括源主机的IP地址、MAC地址、以及目的主机的IP地址。网络中所有的主机收到这个ARP请求后，会检查数据包中的目的IP是否和自己的IP地址一致。如果不相同就忽略此数据包；如果相同，该主机首先将发送端的MAC地址和IP地址添加到自己的ARP列表中，如果ARP表中已经存在该IP的信息，则将其覆盖，然后给源主机发送一个 ARP响应数据包，告诉对方自己是它需要查找的MAC地址；源主机收到这个ARP响应数据包后，将得到的目的主机的IP地址和MAC地址添加到自己的ARP列表中，并利用此信息开始数据的传输。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

常见的路由选择协议，以及它们的区别
常见的路由选择协议有：RIP协议、OSPF协议。
RIP协议：底层是贝尔曼福特算法，它选择路由的度量标准（metric)是跳数，最大跳数是15跳，如果大于15跳，它就会丢弃数据包。
=> 距离向量路由算法
OSPF协议：底层是迪杰斯特拉算法，是链路状态路由选择协议，它选择路由的度量标准是带宽，延迟。
=> 最短路径路由算法

路由设备与相关层
物理层：中继器（Repeater，也叫放大器），集线器。
数据链路层：网桥，交换机。
网络层：路由器。
网关：网络层以上的设备。

参考资料
1.操作系统常见面试题总结
2.数据库常见面试题总结
3.计算机网络常见面试题总结
4.SQL索引一步到位
5.TCP的流量控制和拥塞控制
6.HTTP协议之缓存