1. OSI 7层模型（TCP 4层）

每层协议

OSI七层模型是一种框架性的设计方法，设计的主要目的是为了解决**异种网络互联**时遇到的兼容问题，主要功能就是帮助不同类型的主机实现数据传输。最大优点是将***服务，协议，接口***三者明确的区分开来，通过七个层次化的结构模型使得不同的主机不同的网络之间实现可靠的通讯。

服务说明下一层为上一层提供什么功能，**接口**说明上一层如何实现下一层提供的服务，**协议**涉及本层如何实现自己的服务

- 应用层 (Application)

  ```
  网络服务与最终用户的一个接口，
  协议有：HTTP FTP TFTP SMTP SNMP DNS TELNET HTTPS POP3 DHCP
  ```

- 表示层（Presentation Layer）

  ```
  数据的表示、安全、压缩。（在五层模型里面已经合并到了应用层）
  格式有，JPEG、ASCll、DECOIC、加密格式等
  ```

- 会话层（Session Layer）

  ```
  建立、管理、终止会话。（在五层模型里面已经合并到了应用层）
  对应主机进程，指本地主机与远程主机正在进行的会话
  ```

- 传输层 (Transport)

  ```
  定义传输数据的协议端口号，以及流控和差错校验。
  协议有：TCP UDP，数据包一旦离开网卡即进入网络传输层
  ```

  - **传输控制协议TCP：**提供面向连接的、可靠地数据传输服务，其数据传输的单位是报文段。
  - **用户数据报协议UDP：**提供无连接的、尽最大努力交付的数据传输服务（不保证数据传输的可靠性），其数据传输的单位是用户数据报。

- 网络层 (Network)

  ```
  进行逻辑地址寻址，实现不同网络之间的路径选择。
  协议有：ICMP IGMP IP（IPV4 IPV6） ARP RARP
  ```

  在发送数据的时候，网络层把运输层产生的报文段或用户数据报封装成分组或包进行传送。分组也叫**IP**

数据包或者**数据报**。所以，网络层也是把数据封装成数据报。

- 数据链路层 (Link)

  > 建立逻辑连接、进行硬件地址寻址、差错校验[2] 等功能。（由底层网络定义协议）
  > 将比特组合成字节进而组合成帧，用MAC地址访问介质，错误发现但不能纠正。

  当两个相邻节点之间传送数据时，数据链路层将网络层上交下来的IP数据报**组装成帧**。

- 物理层（Physical Layer）

  > 建立、维护、断开物理连接。（由底层网络定义协议）

TCP/IP 层级模型结构（五层模型），应用层之间的协议通过逐级调用传输层（Transport layer）、网络层（Network Layer）和物理数据链路层（Physical Data Link）而可以实现应用层的应用程序通信互联。

url到页面的过程，以及遇到的问题

> 参考资料：http://www.cnblogs.com/engeng/articles/5943382.html

总体来说分为以下几个阶段：
- DNS解析
- TCP连接
- 发送HTTP请求
- 服务器处理请求并返回HTTP报文
- 浏览器解析渲染页面
- 连接结束

DNS解析

在浏览器输入网址，例如www.baidu.com，这是方便记忆的网址，而我们真正需要的是百度服务器的IP地址，就需要做一个IP地址和网址的转换，这个过程就是**DNS解析**。

解析过程

DNS解析是一个**递归查询**的过程。
首先在本地域名服务器中查询IP地址，如果没有找到的情况下，本地域名服务器会向根域名服务器发送一个请求，如果根域名服务器也不存在该域名时，本地域名会向com顶级域名服务器发送一个请求，依次类推下去。直到最后本地域名服务器得到google的IP地址并把它缓存到本地，供下次查询使用。

> 网址真正的解析过程为：. -> .com -> google.com. -> www.google.com.。

DNS优化

递归查询请求太多，耗时间。利用**DNS缓存**进行优化。

```
DNS存在着多级缓存，从离浏览器的距离排序的话，有以下几种：
    浏览器缓存，系统缓存（host），路由器缓存，ISP服务器缓存，
    根域名服务器缓存，顶级域名服务器缓存，主域名服务器缓存。
    修改hosts文件翻墙原理：https://www.zhihu.com/question/19782572
```

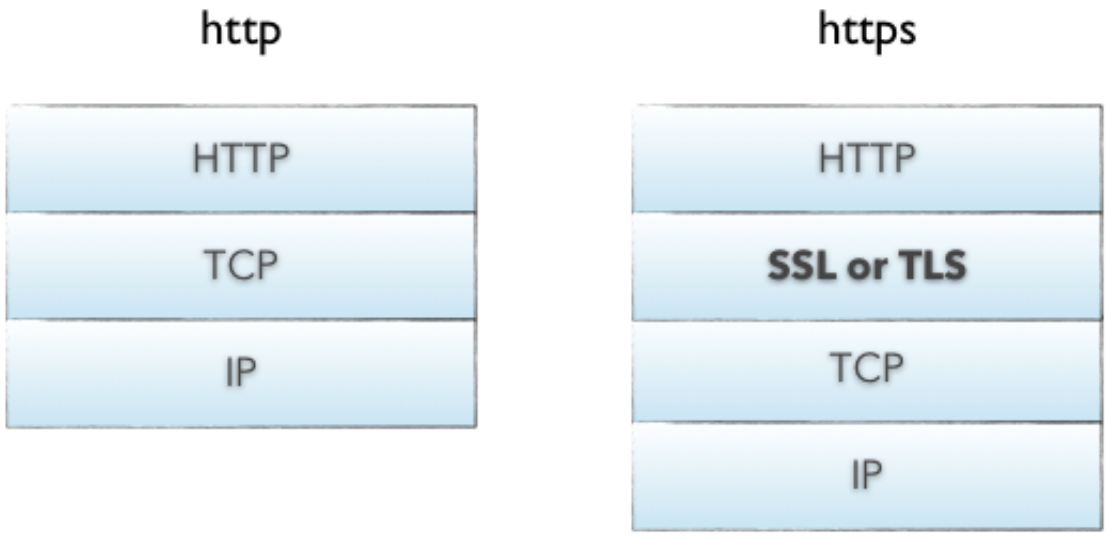
DNS每次返回的IP地址并不是都一样，**DNS负载均衡**。DNS可以返回一个合适的机器的IP给用户，例如可以根据每台机器的负载量，该机器离用户地理位置的距离等等，这种过程就是DNS负载均衡，又叫做DNS重定向。
CDN（内容分发网络）就是DNS重定向技术。

TCP链接

HTTP协议是使用TCP作为其传输层协议的，所以需要三次握手创建连接。
【TCP优化：流量控制、拥塞避免、慢启动 http://blog.jobbole.com/105205/ 】

HTTPS协议：本质上是HTTP + SSL（or TLS），在HTTP报文进入TCP报文之前，先使用SSL对HTTP报文进行加密。避免传输过程中被截取，由于HTTP报文是明文，会存在信息泄露的风险。

HTTPS过程：HTTPS在传输数据之前需要客户端与服务器进行一个握手(TLS/SSL握手)，在握手过程中将确立双方加密传输数据的密码信息。加密、握手产生时延，是否使用HTTPS需要根据具体情况在安全和性能方面做出权衡。



发出HTTP请求

前端工程师眼中的HTTP，它主要发生在客户端。
发送HTTP请求的过程就是构建HTTP请求报文并通过TCP协议中发送到服务器指定端口(HTTP协议80/8080, HTTPS协议443)。HTTP请求报文是由三部分组成: **请求行**, **请求报头**和**请求正文**。

请求行

```
格式如下：
Method Request-URL HTTP-Version CRLF

eg：GET index.html HTTP/1.1
常用的方法有：GET, POST, PUT, DELETE, OPTIONS, HEAD。

## GET和POST有什么区别？ ##
```

请求报头

```
请求报头允许客户端向服务器传递请求的附加信息和客户端自身的信息。
PS：客户端不一定特指浏览器，有时候也可使用Linux下的CURL命令以及HTTP客户端测试工具等。
```

请求报头中使用了Accept, Accept-Encoding, Accept-Language, Cache-Control, Connection, Cookie等字段。Accept用于指定客户端用于接受哪些类型的信息，Accept-Encoding与Accept类似，它用于指定接受的编码方式。Connection设置为Keep-alive用于告诉客户端本次HTTP请求结束之后并不需要关闭TCP连接，这样可以使下次HTTP请求使用相同的TCP通道，节省TCP连接建立的时间。

请求正文

当使用POST, PUT等方法时，通常需要客户端向服务器传递数据。这些数据就储存在请求正文中。在请求包头中有一些与请求正文相关的信息，例如: 现在的Web应用通常采用Rest架构，请求的数据格式一般为json。这时就需要设置Content-Type: application/json。

服务器处理请求并返回HTTP报文

后端工程师眼中的HTTP。
后端从在固定的端口接收到TCP报文开始，这一部分对应于编程语言中的socket。
它会对TCP连接进行处理，对HTTP协议进行解析，并按照报文格式进一步封装成HTTP Request对象，供上层使用。这一部分工作一般是由Web服务器去进行，我使用过的Web服务器有Tomcat, Jetty和Netty等等。

HTTP响应报文也是由三部分组成: **状态码**, **响应报头**和**响应报文**。

状态码

状态码是由3位数组成，第一个数字定义了响应的类别，且有五种可能取值:

```
1xx：指示信息–表示请求已接收，继续处理。
2xx：成功–表示请求已被成功接收、理解、接受。
3xx：重定向–要完成请求必须进行更进一步的操作。
4xx：客户端错误–请求有语法错误或请求无法实现。
5xx：服务器端错误–服务器未能实现合法的请求。
```

平时遇到比较常见的状态码有:200, 204, 301, 302, 304, 400, 401, 403, 404, 422, 500(分别表示什么请自行查找)。

TODO：**HTTP缓存**

响应报头

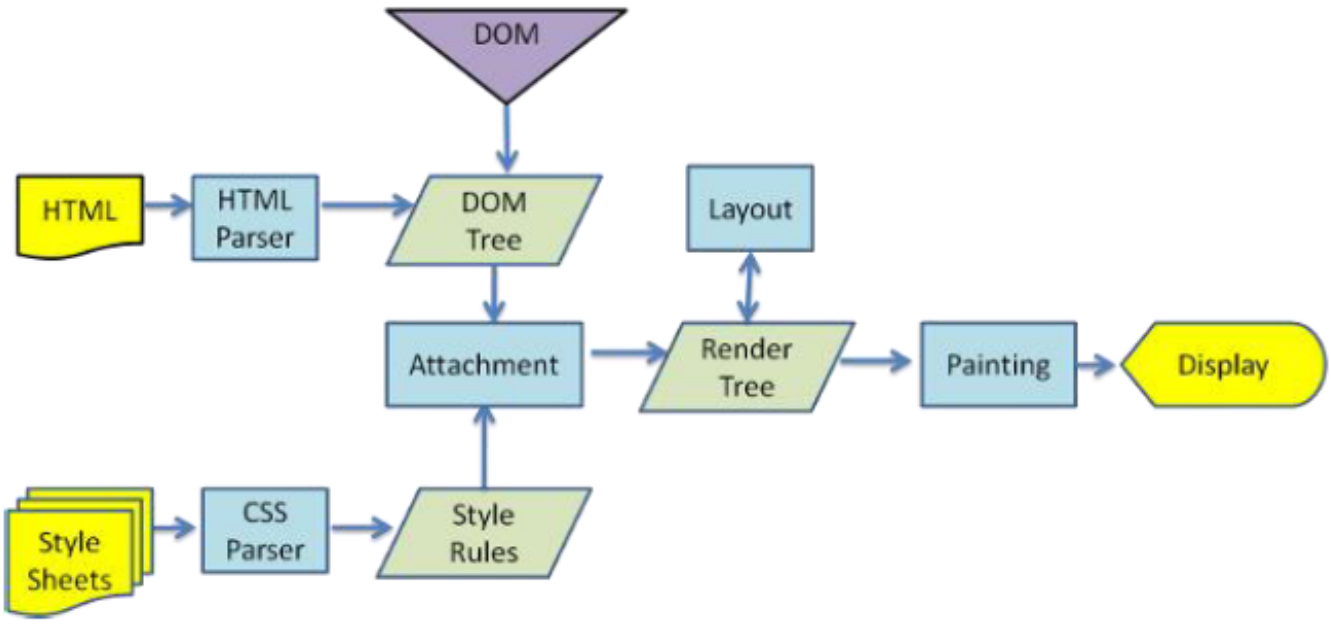
常见的响应报头字段有: Server, Connection…。

响应报文

服务器返回给浏览器的文本信息，通常HTML, CSS, JS, 图片等文件就放在这一部分。

浏览器解析渲染页面

浏览器在收到HTML,CSS,JS文件后，它是如何把页面呈现到屏幕上的？下图对应的就是WebKit渲染的过程。



首先浏览器解析HTML文件构建DOM树，然后解析CSS文件构建渲染树,等到渲染树构建完成后，浏览器开始布局渲染树并将其绘制到屏幕上。

涉及到两个概念：reflow（回流）和repain（重绘）。
- DOM节点中的各个元素都是以盒模型的形式存在，这些都需要浏览器去计算其位置和大小等，这个过程称为reflow
- 当盒模型的位置,大小以及其他属性，如颜色,字体,等确定下来之后，浏览器便开始绘制内容，这个过程称为repain

Web优化

就是要快！如何尽快的加载资源?
答案就是能不从网络中加载的资源就不从网络中加载，当我们合理使用缓存，将资源放在浏览器端，这是最快的方式。如果资源必须从网络中加载，则要考虑缩短连接时间，即DNS优化部分;减少响应内容大小，即对内容进行压缩。另一方面，如果加载的资源数比较少的话，也可以快速的响应用户。当资源到达浏览器之后，浏览器开始进行解析渲染，浏览器中最耗时的部分就是reflow，所以围绕这一部分就是考虑如何减少reflow的次数。

2. HTTP

HTTP/HTTPS 1.0/1.½.0

HTTP 1.0

在 HTTP/1.0 中,大多实现为每个请求/响应交换使用新的连接。

HTTP 1.0规定浏览器与服务器**只保持短暂的连接**，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。

But 一个包含有许多图像的网页文件中并没有包含真正的图像数据内容，而只是指明了这些图像的URL地址。则需要再次连接，再次请求和相应，比较费时。

HTTP 1.1

在 HTTP/1.1 中,一个连接可用于一次或多次请求/响应交换,保持持久连接。减少连接次数，由于服务器保持链接需要占用一定的资源，所以需要权衡。

1、
为了克服HTTP 1.0的这个缺陷，HTTP 1.1支持**持久连接**，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟。

一个包含有许多图像的网页文件的多个请求和应答可以在一个连接中传输，但每个单独的网页文件的请求和应答仍然需要使用各自的连接。

2、
HTTP 1.1还允许客户端**不用等待**上一次请求结果返回，就可以发出下一次请求，但服务器端必须按照接收到客户端请求的先后顺序依次回送响应结果，以保证客户端能够区分出每次请求的响应内容，这样也显著地减少了整个下载过程所需要的时间。
通过Connection请求头：Keep Alive和Close来判定。

3、
HTTP 1.1还通过增加更多的请求头和响应头来改进和扩充HTTP 1.0的功能。**支持Host请求头字段**，保证WEB浏览器可以使用主机头名来明确表示要访问服务器上的哪个WEB站点，这才实现了在一台WEB服务器

上可以在同一个IP地址和端口号上使用不同的主机名来创建多个虚拟WEB站点。

4、
HTTP 1.1还提供了与身份认证、状态管理和Cache缓存等机制相关的请求头和响应头。

5、
支持断点续传，目前的Web服务器绝大多数都采用了HTTP/1.1。

HTTP 2.0

与HTTP1.0的区别在于：
- 2.0采用的是二进制格式而非文本格式
解析起来更高效，错误更少。
- 2.0是完全的多路复用，非有序并阻塞的，只需要一个连接即可实现并行。
1.1存在线端阻塞问题，一次提交一个请求效率高，多了会变慢。
采用多路复用，很好地解决上面的问题，能同时处理多个消息的请求和相应，一个连接就能加载一个页面。
- 使用报头压缩，2.0降低了开销。
一个网页往往有很多资源需要加载，每个请求都有1400字节的消息头，耗时。
- 2.0让服务请可以将响应主动“推送”到客户端缓存中。
通过主动推送将会需要的内容，避免往返的延迟。

HTTP 与 HTTPS

http是HTTP协议运行在TCP之上。所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份。
https是HTTP运行在**SSL/TLS之上，SSL/TLS**运行在TCP之上。所有传输的内容都经过加密，加密采用对称加密，但对称加密的密钥用服务器方的证书进行了非对称加密。此外客户端可以验证服务器端的身份，如果配置了客户端验证，服务器方也可以验证客户端的身份。

SSL(Secure Sockets Layer 安全套接层),及其继任者传输层安全（Transport Layer Security，**TLS**）是为网络通信提供安全及数据完整性的一种安全协议。

GET/POST 以及幂等性

GET和POST的区别

GET - 从指定的资源请求数据。

```
请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：
/test/demo_form.asp?name1=value1&name2=value2
```

POST - 向指定的资源提交要被处理的数据

```
请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

区别：

```
GET 请求可被缓存
GET 请求保留在浏览器历史记录中
GET 请求可被收藏为书签
GET 请求不应在处理敏感数据时使用，数据在URL中对所有人都是可见的。
GET 请求有长度限制

POST 请求不会被缓存
POST 请求不会保留在浏览器历史记录中
POST 不能被收藏为书签
POST 请求对数据长度没有要求
POST 数据不会显示在 URL 中。
```

Get是安全的、幂等的。

所谓**安全的**意味着该操作用于获取信息而非修改信息。
所谓**幂等的**意味着对同一URL的多个请求应该返回同样的结果。

PS：

对于单目运算，如果一个运算对于在范围内的所有的一个数多次进行该运算所得的结果和进行一次该运算所得的结果是一样的，那么我们就称该运算是幂等的。比如绝对值运算就是一个例子，在实数集中，有abs(a)=abs(abs(a))。

对于双目运算，则要求当参与运算的两个值是等值的情况下，如果满足运算结果与参与运算的两个值相等，则称该运算幂等，如求两个数的最大值的函数，有在在实数集中幂等，即max(x,x) = x。

POST表示可能改变服务器上的资源的请求。

ost是向服务器提交数据的一种请求，在FORM（表单）中，Method默认为"GET"。

HTTP协议头相关

通常HTTP消息包括*客户机向服务器的请求消息*和*服务器向客户机的响应消息*。这两种类型的消息由一个**起始行**，一个或者多个**头域**，一个只是**头域结束的空行**和**可选的消息体**组成。

每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。

通用头域

通用头域包含请求和响应消息都支持的头域，通用头域包含Cache-Control、Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。

请求消息（请求头域）

Method Request-URI HTTP-Version CRLF
另外，请求头域允许客户端向服务器传递关于请求或者关于客户机的附加信息。

响应信息

HTTP-Version Status-Code Reason-Phrase CRLF

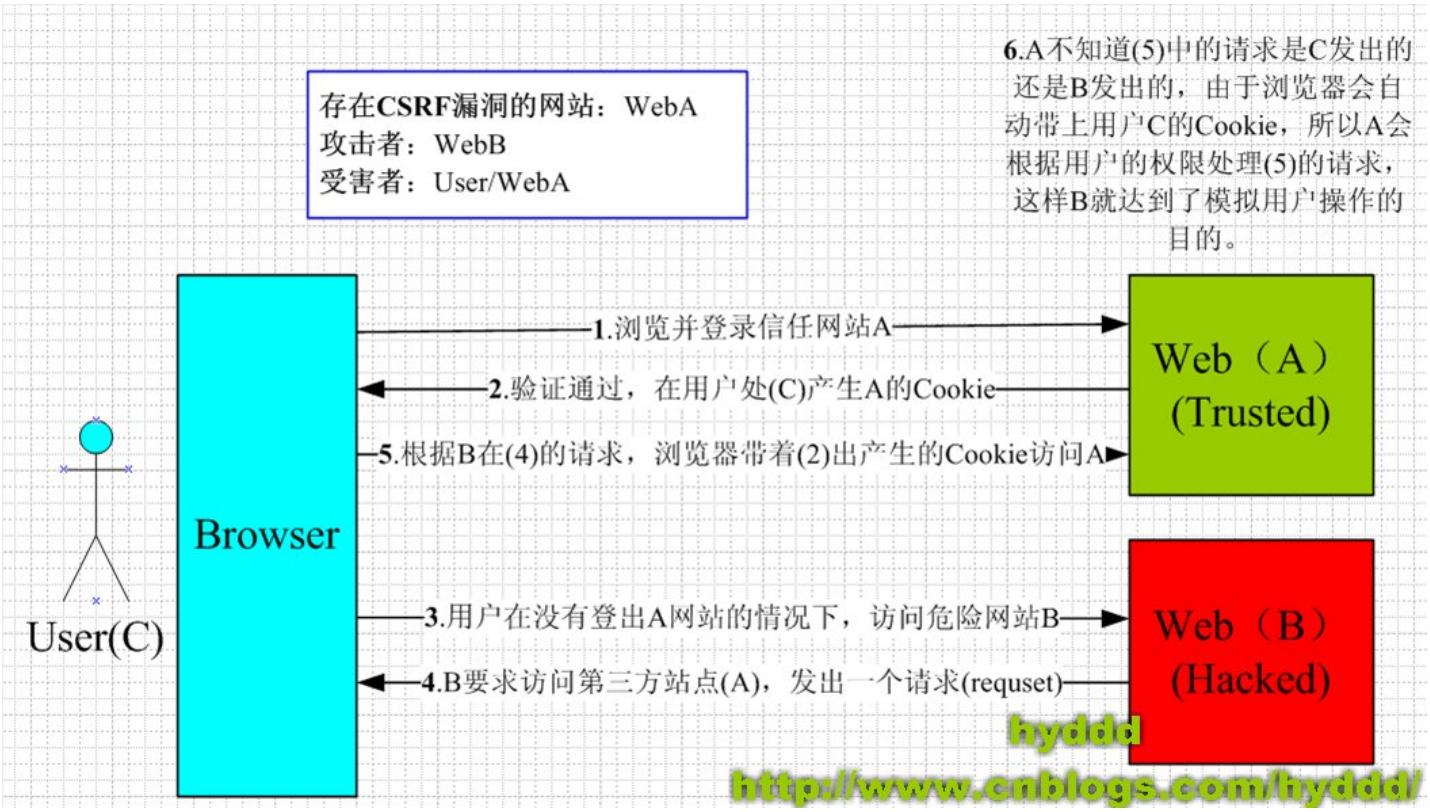
响应头域允许服务器传递不能放在状态行的附加信息，这些域主要描述服务器的信息和Request-URI进一步的信息。

网络攻击

CSRF

CSRF（Cross-site request forgery），中文名称：跨站请求伪造。**攻击者盗用了你的身份，以你的名义发送恶意请求。** Web的隐式身份验证机制！Web的身份验证机制虽然可以保证一个请求是来自于某个用户的浏览器，但却无法保证该请求是用户批准发送的。CSRF攻击的一般是由服务端解决。

原理



服务器防御 - 增加伪随机数

- **Cookie Hashing**：在表单里面增加Hash值，以认证这确实是用户发送的请求。然后在服务器端进行Hash值验证。
- **验证码**：常用的随机字符串验证。
- **Anti Token**：当客户端请求页面时，服务器会生成一个随机数Token，并且将Token放置到session当中，然后将Token发给客户端（一般通过构造hidden表单）。下次客户端提交请求时，Token会随着表单一起提交到服务器端。

XSS

XSS（cross-site scripting跨域脚本攻击）攻击是最常见的Web攻击，其重点是“跨域”和“客户端执行”。

反射性xss

发出请求时，XSS代码出现在URL中，作为输入提交到服务器端，服务器端解析后响应，XSS代码随响应内容一起传回给浏览器，最后浏览器解析执行XSS代码。这个过程像一次反射，故叫反射型XSS。

存储型xss

存储型XSS和反射型XSS的差别仅在于，提交的代码会存储在服务器端（数据库，内存，文件系统等），下次请求目标页面时不用再提交XSS代码。

> 后端尽可能对提交数据做过滤，在场景需求而不过滤的情况下，前端就需要做些处理了。

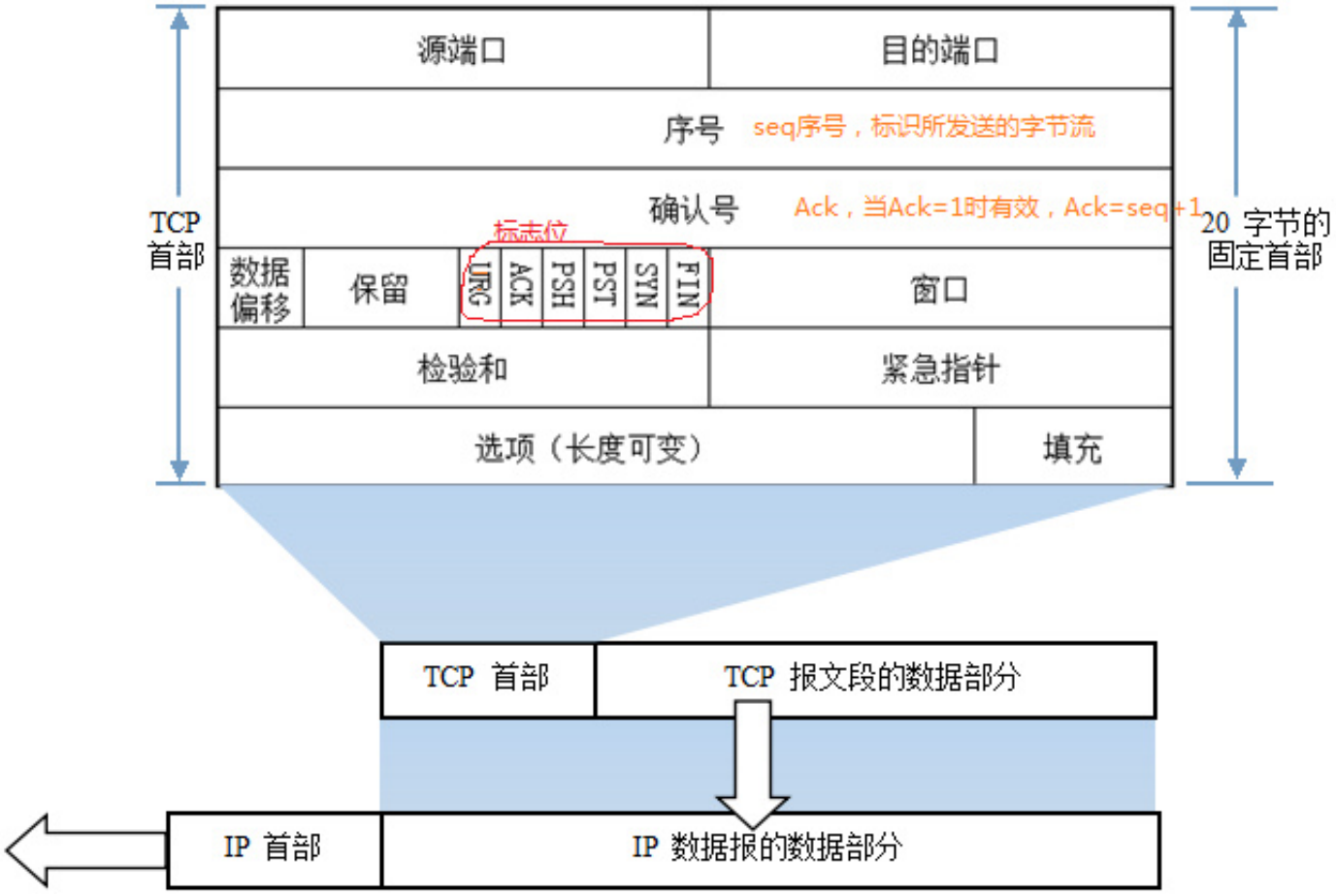
DOM xss

DOM XSS和反射型XSS、存储型XSS的差别在于DOM XSS的代码并不需要服务器参与，触发XSS靠的是浏览器端的DOM解析，完全是客户端的事情。Wifi劫持。

```
    使用HTTPS，用密钥来保护。
    参考资料：http://www.cnblogs.com/lovesong/p/5199623.html
```

3. TCP/IP

三次握手、四次挥手



（1）**序号**：Seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。

（2）**确认序号**：Ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。

（3）标志位：共6个，即URG、ACK、PSH、RST、SYN、FIN等，具体含义如下：

（A）URG：紧急指针（urgent pointer）有效。

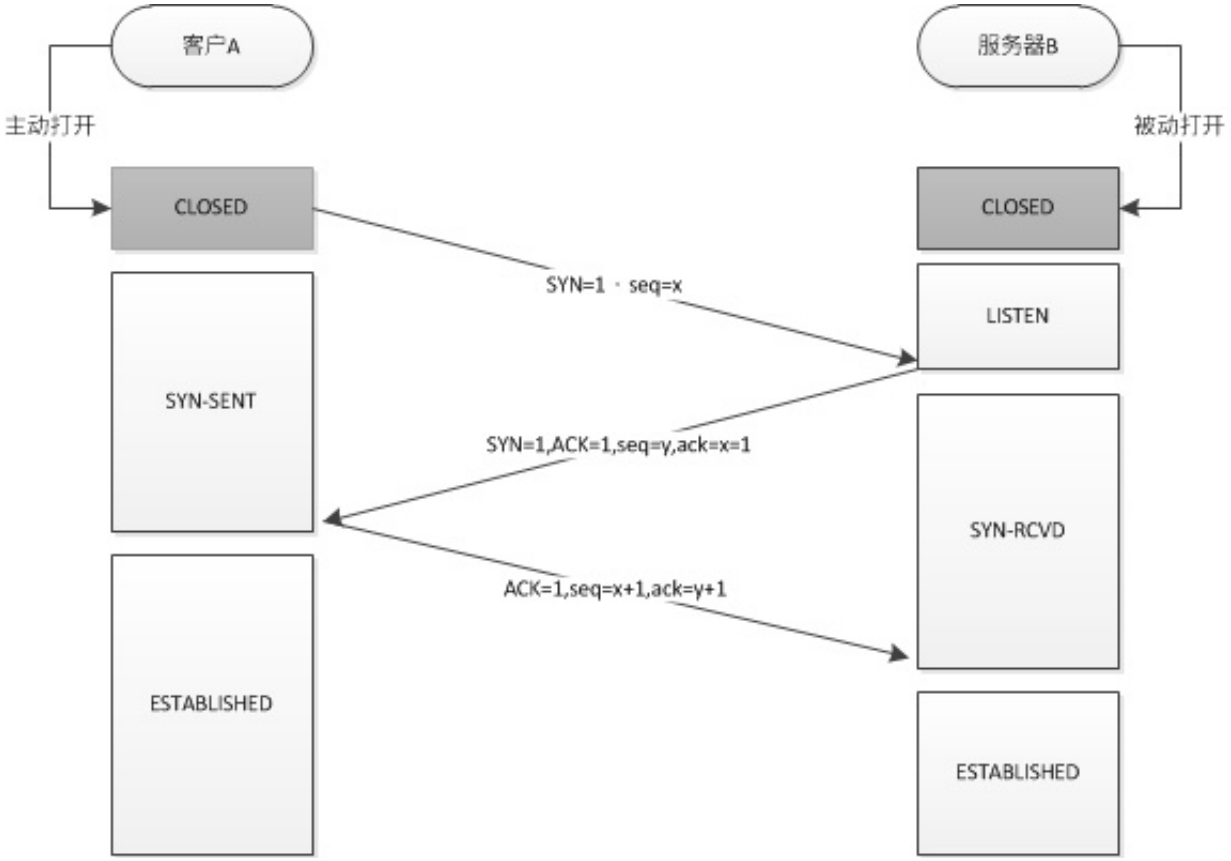
（B）**ACK：确认序号有效。**

（C）PSH：接收方应该尽快将这个报文交给应用层。

（D）RST：重置连接。

（E）**SYN：发起一个新连接。**
（F）**FIN：释放一个连接。**

三次握手：

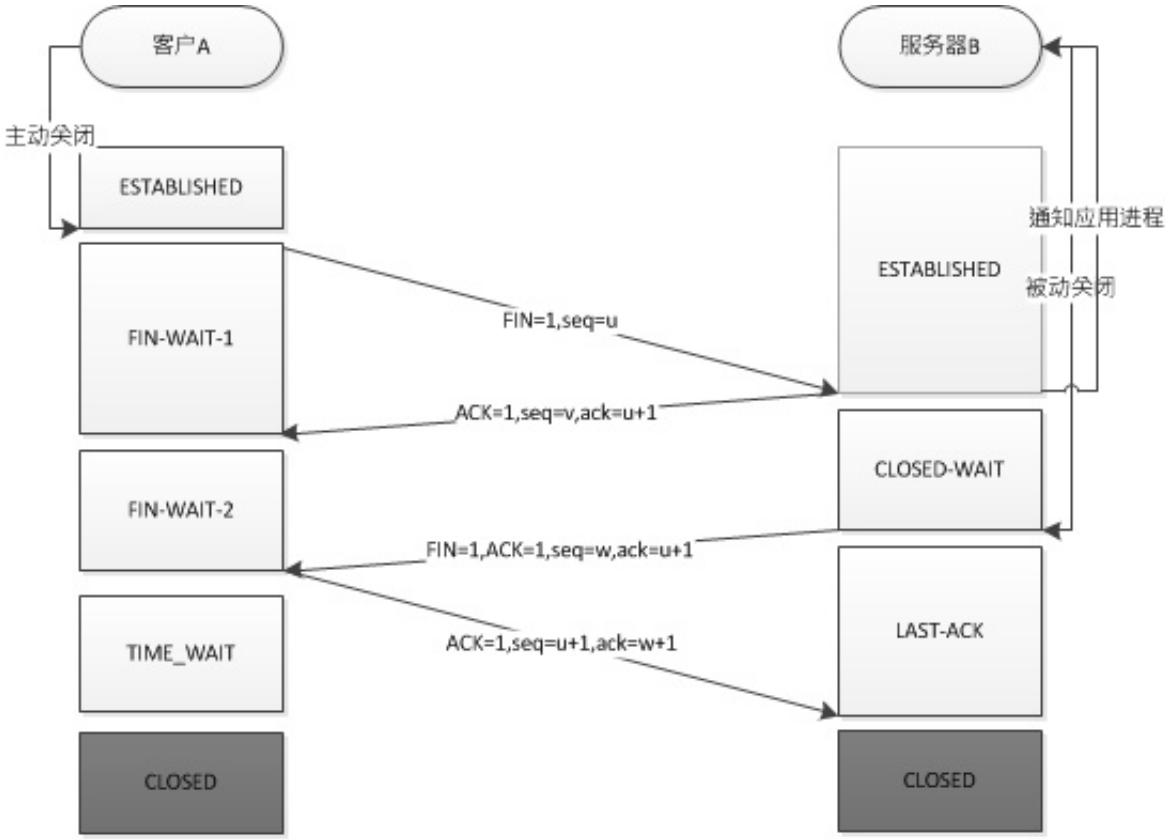


Syn攻击

Syn攻击就是在短时间内伪造大量不存在的IP地址，向服务器不断地发送syn包，服务器回复确认包，并等待客户的确认，由于源地址是不存在的，服务器需要不断的重发直 至超时，这些伪造的SYN包将长时间占用未连接队列，正常的SYN请求被丢弃，目标系统运行缓慢，严重者引起网络堵塞甚至系统瘫痪。

> Syn攻击是典型的DDos攻击。

四次挥手



为什么建立连接是三次握手，而关闭连接却是四次挥手呢?

这是因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送。

拥塞控制（过程、阈值）

拥塞控制就是防止过多的数据注入网络中，这样可以使网络中的路由器或链路不致过载。拥塞控制是一个全局性的过程，和流量控制不同，流量控制指点对点通信量的控制。

慢开始与拥塞避免【Tahoe】

发送方维持一个叫做**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。

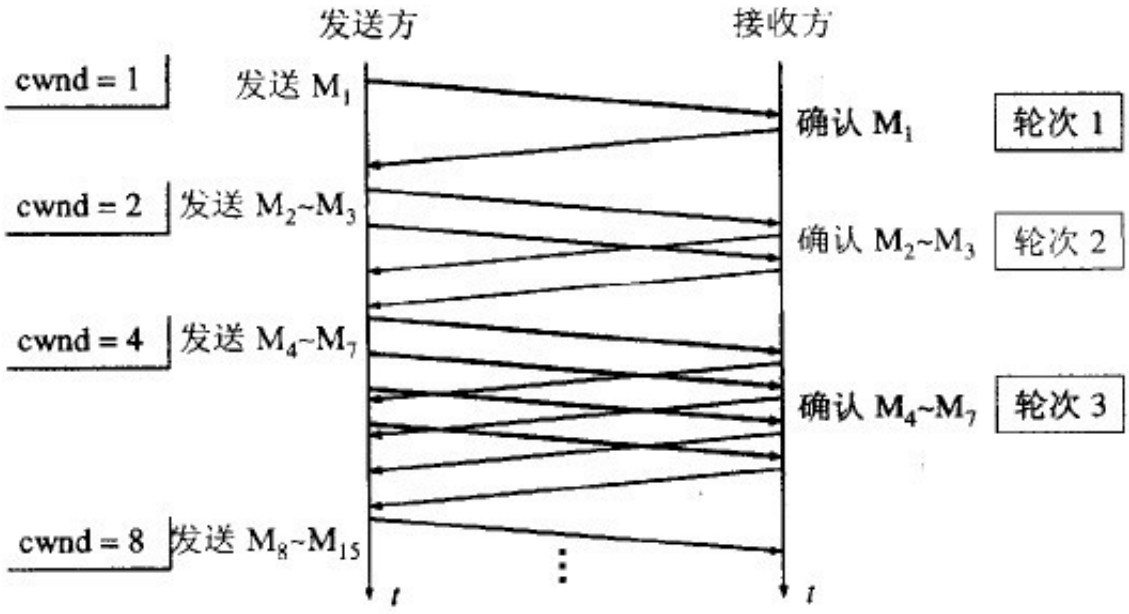


图 5-24　发送方每收到一个确认就把窗口 cwnd 加 1

慢开始算法的思路就是，不要一开始就发送大量的数据，先探测一下网络的拥塞程度，也就是说由小到大逐渐增加拥塞窗口的大小。

当然收到单个确认但此确认多个数据报的时候就加相应的数值。所以一次传输轮次之后拥塞窗口就加倍。这就是乘法增长，和后面的拥塞避免算法的加法增长比较。

为了防止cwnd增长过大引起网络拥塞，还需设置一个**慢开始门限**ssthresh状态变量。ssthresh的用法如下：

- 当cwnd<ssthresh时，使用**慢开始算法**。 `乘法增长 ×2`
- 当cwnd>ssthresh时，改用**拥塞避免算法**。 `加法增长 +1`
- 当cwnd=ssthresh时，慢开始与拥塞避免算法任意。

无论是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，虽然没有收到确认可能是其他原因的分组丢失，但是因为无法判定，所以都当做拥塞来处理*），就把慢开始门限设置为出现拥塞时的发送窗口大小的一半。然后把拥塞窗口设置为1，执行慢开始算法。如下图：

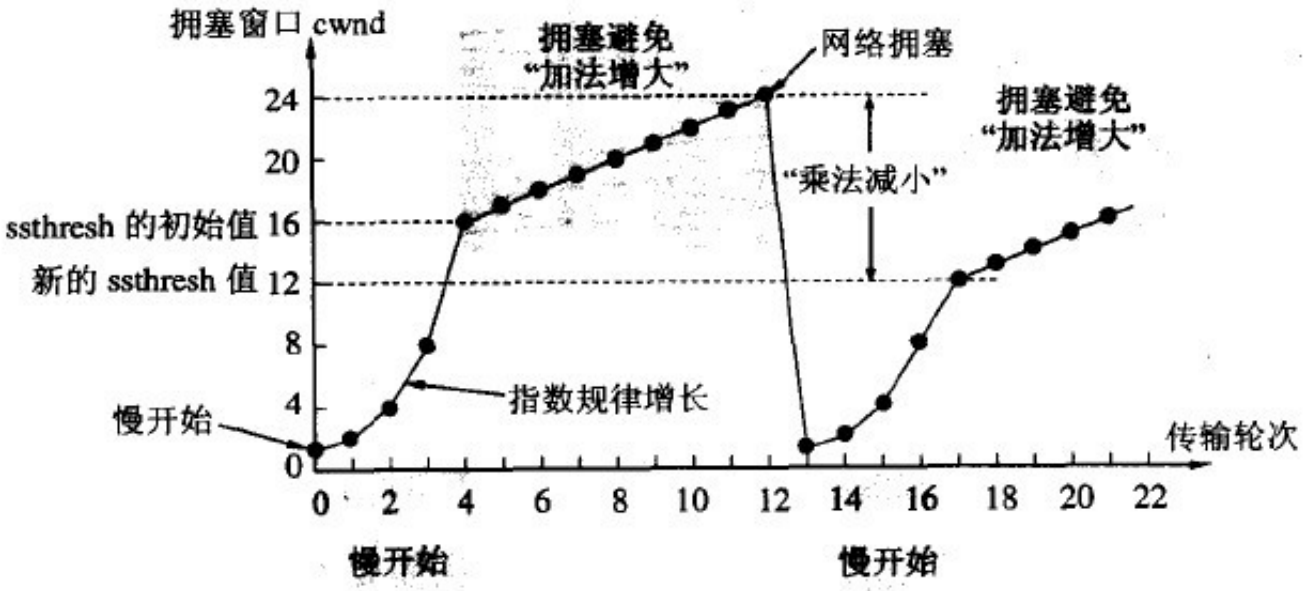


图 5-25　慢开始和拥塞避免算法的实现举例

快重传与快恢复【Reno】

快重传要求接收方在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捎带确认。快重传算法规定，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时间到期。如下图:

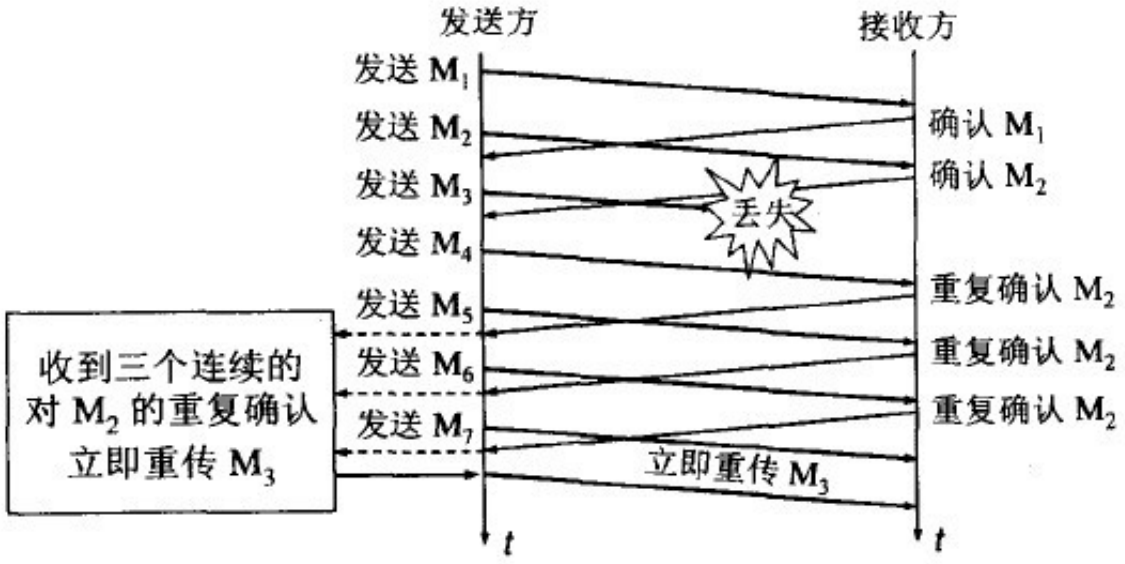


图 5-26　快重传的示意图

快重传配合使用的还有**快恢复**算法，有以下两个要点:
①当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把ssthresh门限减半。但是接下去并不执行慢开始算法。

②考虑到如果网络出现拥塞的话就不会收到好几个重复的确认，所以发送方现在认为网络可能没有出现拥塞。所以此时不执行慢开始算法，而是将cwnd设置为ssthresh的大小，然后执行拥塞避免算法。如下图：

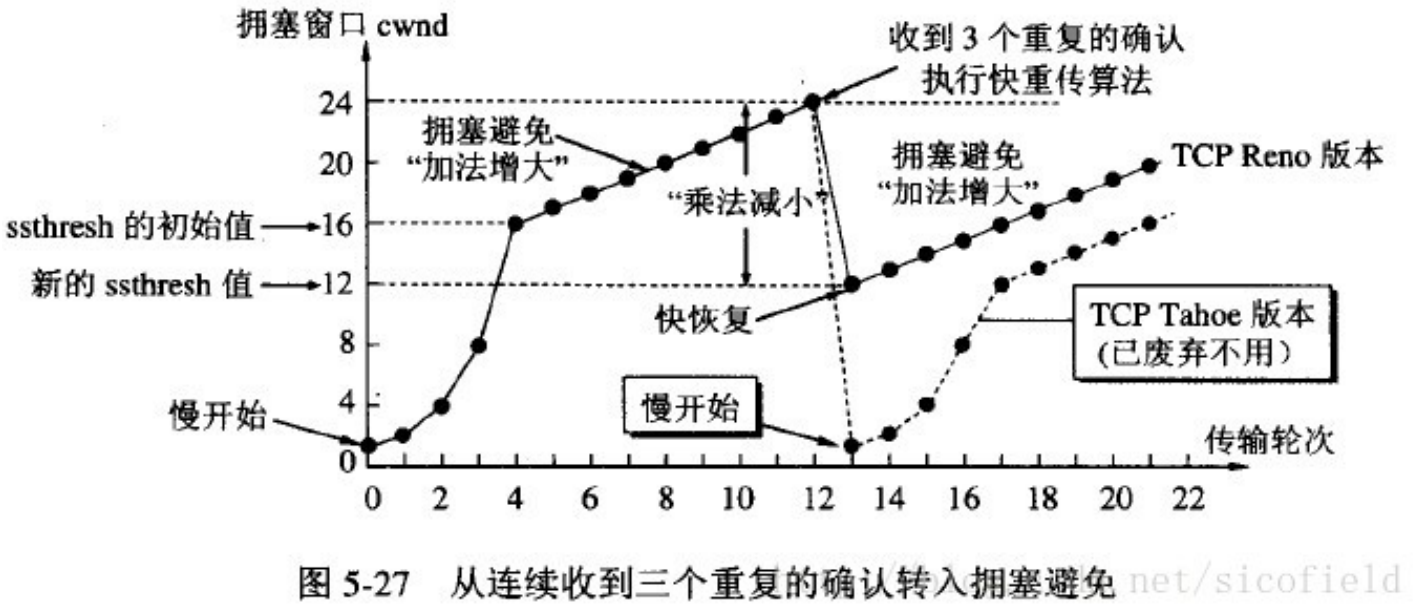


图 5-27　从连续收到三个重复的确认转入拥塞避免

不至于引起：引起网络的激烈震荡，以及网络利用率的降低。

流量控制与滑动窗口

TCP传输通过调整发送方窗口和接收方窗口的大小可以实现**流量控制**。TCP滑动窗口技术通过动态改变窗口大小来调节两台主机间数据传输。

每个TCP/IP主机支持全双工数据传输，因此TCP有两个滑动窗口：一个用于接收数据，另一个用于发送数据。TCP使用肯定确认技术，其确认号指的是 `下一个` 所期待的字节。

假定发送方设备以每一次三个数据包的方式发送数据，也就是说，窗口大小为3。发送方发送序列号为1、2、3的三个数据包，接收方设备成功接收数据包，用序列号4确认。发送方设备收到确认，继续以窗口大小3发送数据。当接收方设备要求降低或者增大网络流量时，可以对窗口大小进行减小或者增加，本例降低窗口大小为2，每一次发送两个数据包。当接收方设备要求窗口大小为0，表明接收方已经接收了全部数据，或者接收方应用程序没有时间读取数据，要求暂停发送。发送方接收到携带窗口号为0的确认，停止这一方向的数据传输。

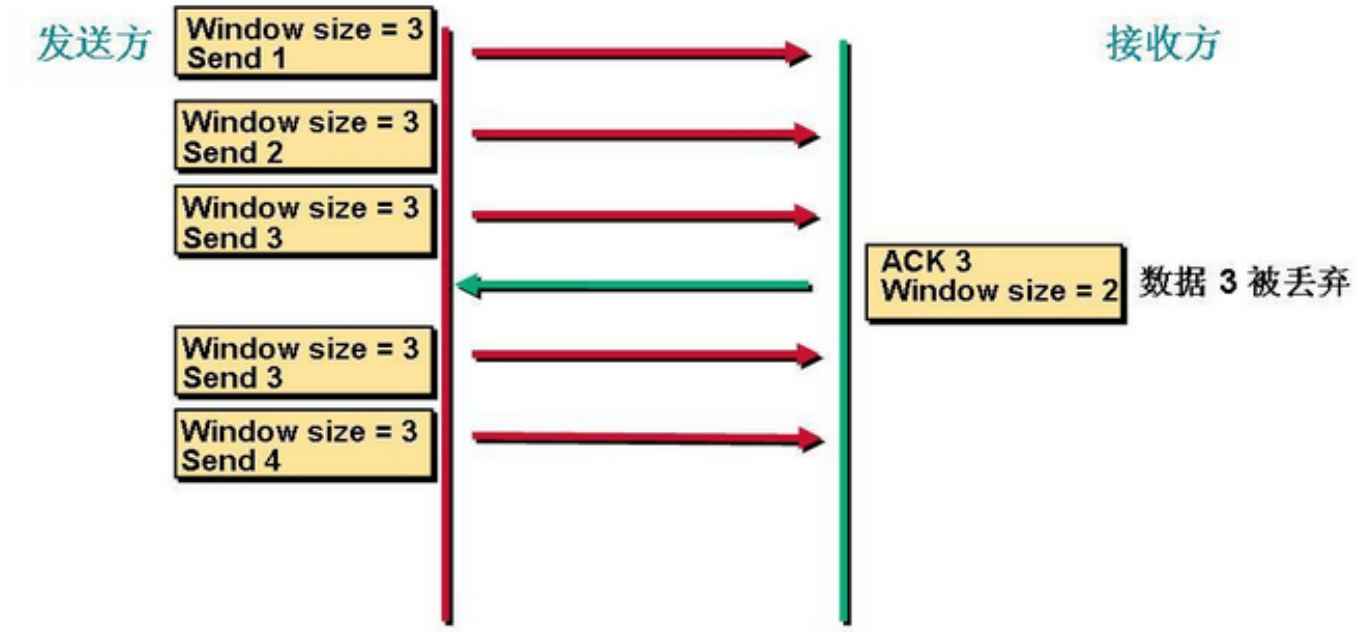
固定窗口有什么问题

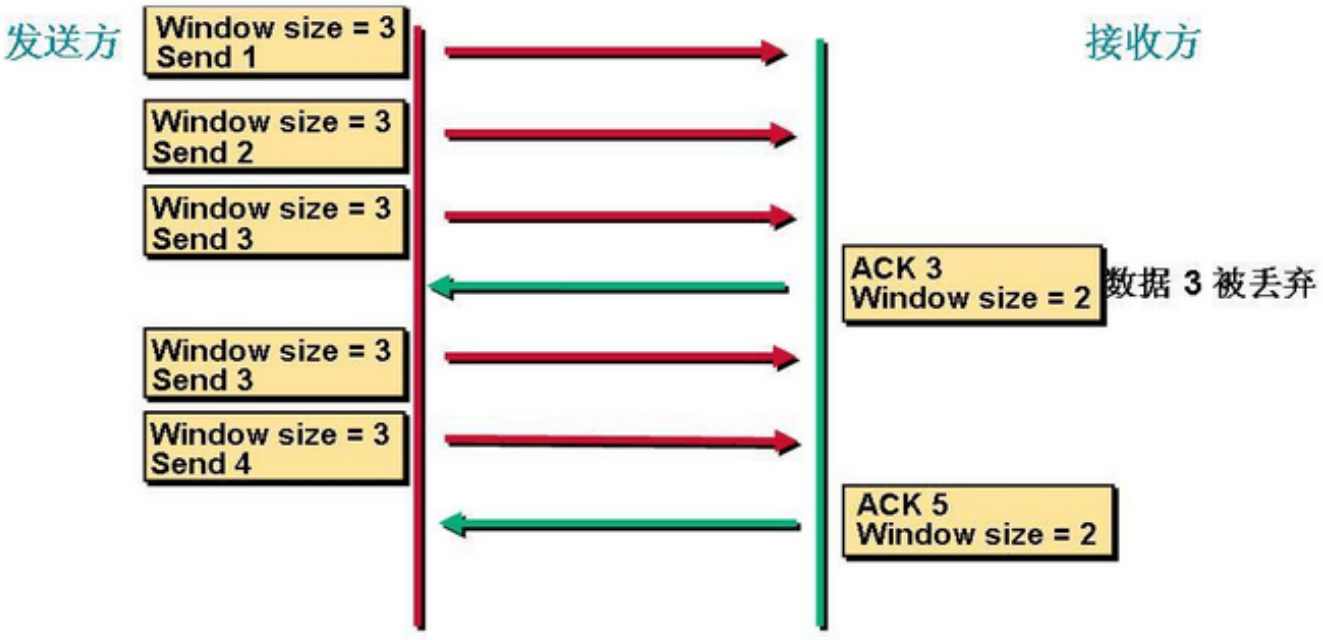
如果说**窗口过小**，那么当传输比较大的数据的时候需要不停的对数据进行确认，这个时候就会造成很大的延迟。

如果说**窗口过大**，发送方下一次还是发送100个数据，但是接受方还是只能处理50个数据。这样就避免了不必要的数据来拥塞我们的链路。

滑动窗口协议

所以我们就引入了滑动窗口机制，窗口的大小并不是固定的而是根据我们之间的链路的带宽的大小，这个时候链路是否拥塞。接受方是否能处理这么多数据了。





滑动窗口协议，是TCP使用的一种流量控制方法。该协议允许发送方在停止并等待确认前可以连续发送多个分组。由于发送方不必每发一个分组就停下来等待确认，因此该协议可以加速数据的传输。

当发送窗口和接收窗口的大小都等于1时，就是停止等待协议。

发端窗口的大小取决于收端的窗口大小rwnd（TCP报文的窗口大小字段）和拥塞窗口大小cwnd（见拥塞控制），即发端窗口大小 = min{ rwnd , cwnd };

参考资料：http://blog.jobbole.com/105500/

TCP与UDP比较

TCP与UDP区别总结：
1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接
2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付。
3、TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的，保留了应用程序传输下来的报文的边界，UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）
4、每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信
5、TCP首部开销20字节;UDP的首部开销小，只有8个字节
6、TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道。

子网划分

这个网址上有两个例子可以参考：http://yuanbin.blog.51cto.com/363003/112029/

子网划分的两个例子

例1：本例通过子网数来划分子网，未考虑主机数。
一家集团公司有12家子公司，每家子公司又有4个部门。上级给出一个172.16.0.0/16的网段，让给每家子公司以及子公司的部门分配网段。
思路：既然有12家子公司，那么就要划分12个子网段，但是每家子公司又有4个部门，因此又要在每家子公司所属的网段中划分4个子网分配给各部门。
步骤：
A. 先划分各子公司的所属网段。
有12家子公司，那么就有2的n次方≥12，n的最小值=4。因此，网络位需要向主机位借4位。那么就可以从172.16.0.0/16这个大网段中划出2的4次方=16个子网。
详细过程：
先将172.16.0.0/16用二进制表示
10101100.00010000.00000000.00000000/16
借4位后（可划分出16个子网）：
1) **10101100.00010000.0000**0000.00000000/20【172.16.0.0/20】

2) 10101100.00010000.00010000.00000000/20【172.16.16.0/20】
3) 10101100.00010000.00100000.00000000/20【172.16.32.0/20】
4) 10101100.00010000.00110000.00000000/20【172.16.48.0/20】
5) 10101100.00010000.01000000.00000000/20【172.16.64.0/20】
6) 10101100.00010000.01010000.00000000/20【172.16.80.0/20】
7) 10101100.00010000.01100000.00000000/20【172.16.96.0/20】
8) 10101100.00010000.01110000.00000000/20【172.16.112.0/20】
9) 10101100.00010000.10000000.00000000/20【172.16.128.0/20】
10) 10101100.00010000.10010000.00000000/20【172.16.144.0/20】
11) 10101100.00010000.10100000.00000000/20【172.16.160.0/20】
12) 10101100.00010000.10110000.00000000/20【172.16.176.0/20】
13) 10101100.00010000.11000000.00000000/20【172.16.192.0/20】
14) 10101100.00010000.11010000.00000000/20【172.16.208.0/20】
15) 10101100.00010000.11100000.00000000/20【172.16.224.0/20】
16) 10101100.00010000.11110000.00000000/20【172.16.240.0/20】
我们从这16个子网中选择12个即可，就将前12个分给下面的各子公司。每个子公司最多容纳主机数目为2的12次方-2=4094。

B. 再划分子公司各部门的所属网段
以甲公司获得172.16.0.0/20为例，其他子公司的部门网段划分同甲公司。
有4个部门，那么就有2的n次方≥4，n的最小值=2。因此，网络位需要向主机位借2位。那么就可以从172.16.0.0/20这个网段中再划出2的2次方=4个子网，正符合要求。
详细过程：
先将172.16.0.0/20用二进制表示
10101100.00010000.00000000.00000000/20
借2位后（可划分出4个子网）：
① 10101100.00010000.00000000.00000000/22【172.16.0.0/22】
② 10101100.00010000.00000100.00000000/22【172.16.4.0/22】
③ 10101100.00010000.00001000.00000000/22【172.16.8.0/22】
④ 10101100.00010000.00001100.00000000/22【172.16.12.0/22】
将这4个网段分给甲公司的4个部门即可。每个部门最多容纳主机数目为2的10次方-2=1024。

例2：本例通过计算主机数来划分子网。
某集团公司给下属子公司甲分配了一段IP地址192.168.5.0/24，现在甲公司有两层办公楼（1楼和2楼），统一从1楼的路由器上公网。1楼有100台电脑联网，2楼有53台电脑联网。如果你是该公司的网管，你该怎么去规划这个IP?
根据需求，画出下面这个简单的拓扑。将192.168.5.0/24划成3个网段，1楼一个网段，至少拥有101个可用IP地址；2楼一个网段，至少拥有54个可用IP地址；1楼和2楼的路由器互联用一个网段，需要2个IP地址。

思路:我们在划分子网时优先考虑最大主机数来划分。在本例中，我们就先使用最大主机数来划分子网。101个可用IP地址，那就要保证至少7位的主机位可用（2的m次方-2≥101，m的最小值=7）。如果保留7位主机位，那就只能划出两个网段，剩下的一个网段就划不出来了。但是我们剩下的一个网段只需要2个IP地址并且2楼的网段只需要54个可用IP，因此，我们可以从第一次划出的两个网段中选择一个网段来继续划分2楼的网

段和路由器互联使用的网段。

步骤：

A. 先根据大的主机数需求，划分子网

因为要保证1楼网段至少有101个可用IP地址，所以，主机位要保留至少7位。

先将192.168.5.0/24用二进制表示：

11000000.10101000.00000101.00000000/24

主机位保留7位，即在现有基础上网络位向主机位借1位（可划分出2个子网）：

① 11000000.10101000.00000101.00000000/25【192.168.5.0/25】

② 11000000.10101000.00000101.10000000/25【192.168.5.128/25】

1楼网段从这两个子网段中选择一个即可，我们选择192.168.5.0/25。

2楼网段和路由器互联使用的网段从192.168.5.128/25中再次划分得到。

B. 再划分2楼使用的网段

2楼使用的网段从192.168.5.128/25这个子网段中再次划分子网获得。因为2楼至少要有54个可用IP地址，所以，主机位至少要保留6位（2的m次方-2≥54，m的最小值=6）。

先将192.168.5.128/25用二进制表示：

11000000.10101000.00000101.10000000/25

主机位保留6位，即在现有基础上网络位向主机位借1位（可划分出2个子网）：

① 11000000.10101000.00000101.10000000/26【192.168.5.128/26】

② 11000000.10101000.00000101.11000000/26【192.168.5.192/26】

2楼网段从这两个子网段中选择一个即可，我们选择192.168.5.128/26。

路由器互联使用的网段从192.168.5.192/26中再次划分得到。

C. 最后划分路由器互联使用的网段

路由器互联使用的网段从192.168.5.192/26这个子网段中再次划分子网获得。因为只需要2个可用IP地址，所以，主机位只要保留2位即可（2的m次方-2≥2，m的最小值=2）。

先将192.168.5.192/26用二进制表示：

11000000.10101000.00000101.11000000/26

主机位保留2位，即在现有基础上网络位向主机位借4位（可划分出16个子网）：

① 11000000.10101000.00000101.11000000/30【192.168.5.192/30】

② 11000000.10101000.00000101.11000100/30【192.168.5.196/30】

③ 11000000.10101000.00000101.11001000/30【192.168.5.200/30】

……

④ 11000000.10101000.00000101.11110100/30【192.168.5.244/30】

⑤ 11000000.10101000.00000101.11111000/30【192.168.5.248/30】

⑥ 11000000.10101000.00000101.11111100/30【192.168.5.252/30】

路由器互联网段我们从这16个子网中选择一个即可，我们就选择192.168.5.252/30。

D. 整理本例的规划地址

1楼：

网络地址：【192.168.5.0/25】

主机IP地址：【192.168.5.1/25—192.168.5.126/25】

广播地址：【192.168.5.127/25】

2楼：
网络地址：【192.168.5.128/26】
主机IP地址：【192.168.5.129/26—192.168.5.190/26】
广播地址：【192.168.5.191/26】
路由器互联：
网络地址：【192.168.5.252/30】
两个IP地址：【192.168.5.253/30、192.168.5.254/30】
广播地址：【192.168.5.255/30】

DDos攻击

- 某饭店可以容纳100人同时就餐，某日有个商家恶意竞争，雇佣了200人来这个饭店坐着不吃不喝，导致饭店满满当当无法正常营业。**（DDOS攻击成功）【SYN攻击，3次握手的时候，伪造IP，发送SYN请求，但服务器ACK的时候没有得到回复，就不断地重发。】**
- 老板当即大怒，派人把不吃不喝影响正常营业的人全都轰了出去，且不再让他们进来捣乱，饭店恢复了正常营业。**（添加规则和黑名单进行DDOS防御，防御成功）**
- 主动攻击的商家心存不满，这次请了五千人逐批次来捣乱，导致该饭店再次无法正常营业。**（增加DDOS流量，改变攻击方式）**
- 饭店把那些捣乱的人轰出去只后，另一批接踵而来。此时老板将饭店营业规模扩大，该饭店可同时容纳1万人就餐，5000人同时来捣乱饭店营业也不会受到影响。**（增加硬防与其抗衡）**

同时也可以：分析行为判断异常结点，动态指纹识别，带宽控制。

```
漫画DDos攻击：https://www.leiphone.com/news/201509/9zGlIDvLhwguqOtg.html
```

(B)IO / NIO / AIO

一个系统的优化空间，往往都在低效率的I/O环节上，很少看到一个系统CPU、内存的性能是其整个系统的瓶颈。也正因为如此，Java在I/O上也一直在做持续的优化，从JDK 1.4开始便引入了NIO模型，大大的提高了以往BIO模型下的操作效率。

- 同步阻塞IO（BIO)
- 同步非阻塞IO (NIO)
- 异步阻塞IO（AIO）

1 **同步** 指的是用户进程触发IO操作并等待或者轮询的去查看IO操作是否就绪 自己上街买衣服，自己亲自干这件事，别的事干不了。
2 **异步** 异步是指用户进程触发IO操作以后便开始做自己的事情，而当IO操作已经完成的时候会得到IO完成的通知（异步的特点就是通知） 告诉朋友自己合适衣服的尺寸，大小，颜色，让朋友委托去卖，然后自己可以去干别的事。（使用异步IO时，Java将IO读写委托给OS处理，需要将数据缓冲区地址和大小传给OS）

3 **阻塞** 所谓阻塞方式的意思是指, 当试图对该文件描述符进行读写时, 如果当时没有东西可读,或者暂时不可写, 程序就进入等待 状态, 直到有东西可读或者可写为止 去公交站充值，发现这个时候，充值员不在（可能上厕所去了），然后我们就在这里等待，一直等到充值员回来为止。（当然现实社会，可不是这样，但是在计算机里确实如此。）
4 **非阻塞** 非阻塞状态下, 如果没有东西可读, 或者不可写, 读写函数马上返回, 而不会等待， 银行里取款办业务时，领取一张小票，领取完后我们自己可以玩玩手机，或者与别人聊聊天，当轮我们时，银行的喇叭会通知，这时候我们就可以去了。

三者原理，各个语言是怎么实现的

BIO （Blocking I/O）： 同步阻塞I/O模式，数据的读取写入必须阻塞在一个线程内等待其完成。这里使用那个经典的烧开水例子，这里假设一个烧开水的场景，有一排水壶在烧开水，BIO的工作模式就是， 叫一个线程停留在一个水壶那，直到这个水壶烧开，才去处理下一个水壶。但是实际上线程在等待水壶烧开的时间段什么都没有做。

BIO**主要的问题**在于每当有一个新的客户端请求接入时，服务端必须创建一个新的线程来处理这条链路，在需要满足高性能、高并发的场景是没法应用的（大量创建新的线程会严重影响服务器性能，甚至罢工）。

为了改进这种一连接一线程的模型，我们可以使用线程池来管理这些线程（需要了解更多请参考前面提供的文章），实现1个或多个线程处理N个客户端的模型（但是底层还是使用的同步阻塞I/O），通常被称为“伪异步I/O模型“。

NIO （New I/O）： 同时支持阻塞与非阻塞模式，但这里我们以其同步非阻塞I/O模式来说明，那么什么叫做同步非阻塞？如果还拿烧开水来说，NIO的做法是叫一个线程不断的轮询每个水壶的状态，看看是否有水壶的状态发生了改变，从而进行下一步的操作。

JDK 1.4中的java.nio.*包中引入新的Java I/O库，其目的是提高速度。速度的提高在文件I/O和网络I/O中都可能会发生。对于低负载、低并发的应用程序，可以使用同步阻塞I/O来提升开发速率和更好的维护性；对于高负载、高并发的（网络）应用，应使用NIO的非阻塞模式来开发。

- 通道 Channel： 我们对数据的读取和写入要通过Channel，它就像水管一样，是一个通道。通道不同于流的地方就是通道是双向的，可以用于读、写和同时读写操作。Channel主要分两大类：1、SelectableChannel：用户网络读写。2、FileChannel：用于文件操作。后面代码会涉及的ServerSocketChannel和SocketChannel都是SelectableChannel的子类。
- 多路复用器 Selector： Selector提供选择已经就绪的任务的能力：Selector会不断轮询注册在其上的Channel，如果某个Channel上面发生读或者写事件，这个Channel就处于就绪状态，会被Selector轮询出来，然后通过SelectionKey可以获取就绪Channel的集合，进行后续的I/O操作。

AIO （ Asynchronous I/O）： 异步非阻塞I/O模型。异步非阻塞与同步非阻塞的区别在哪里？异步非阻塞无需一个线程去轮询所有IO操作的状态改变，在相应的状态改变后，系统会通知对应的线程来处理。对应到烧开水中就是，为每个水壶上面装了一个开关，水烧开之后，水壶会自动通知我水烧开了。

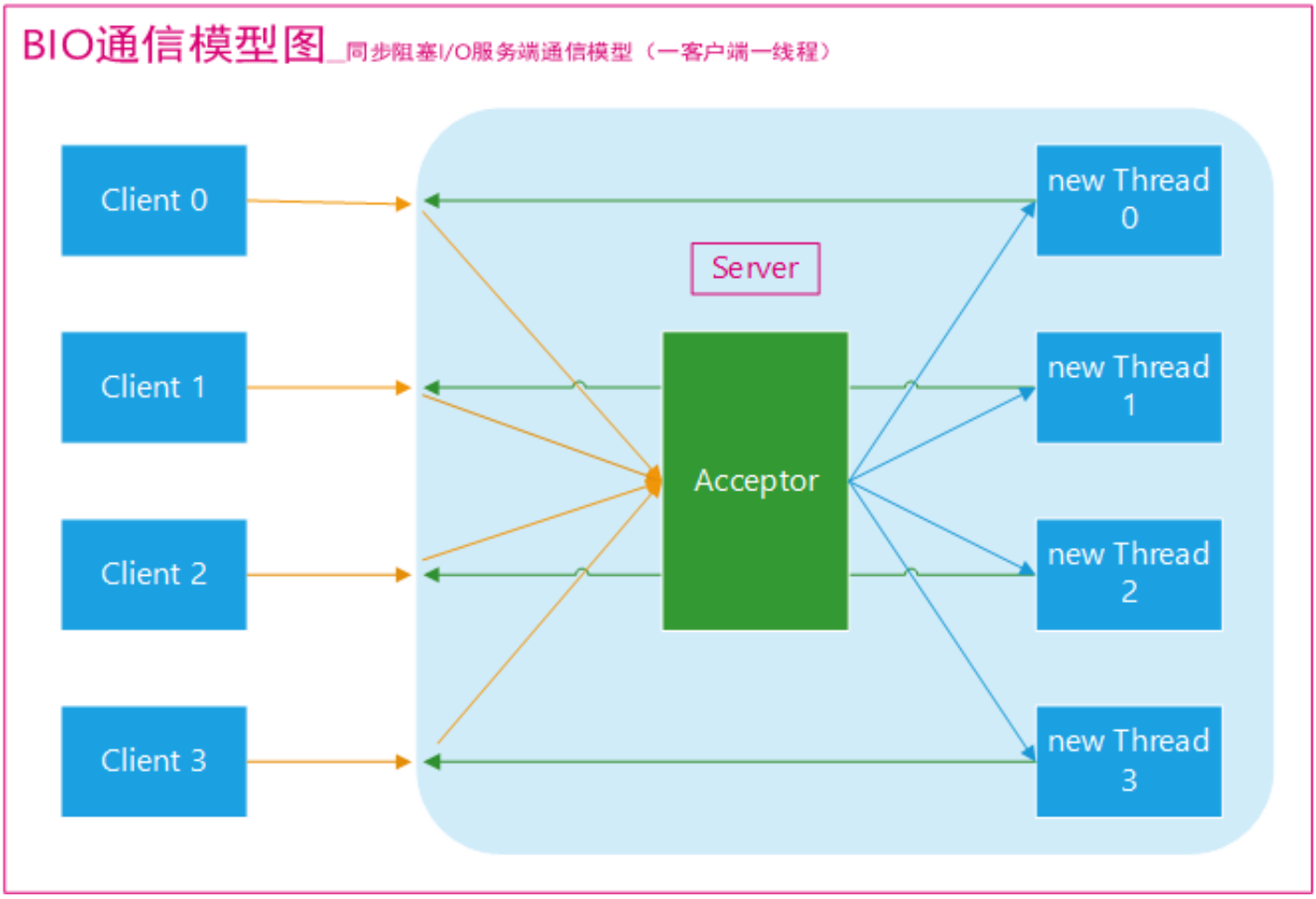
NIO 2.0引入了新的异步通道的概念，并提供了异步文件通道和异步套接字通道的实现。

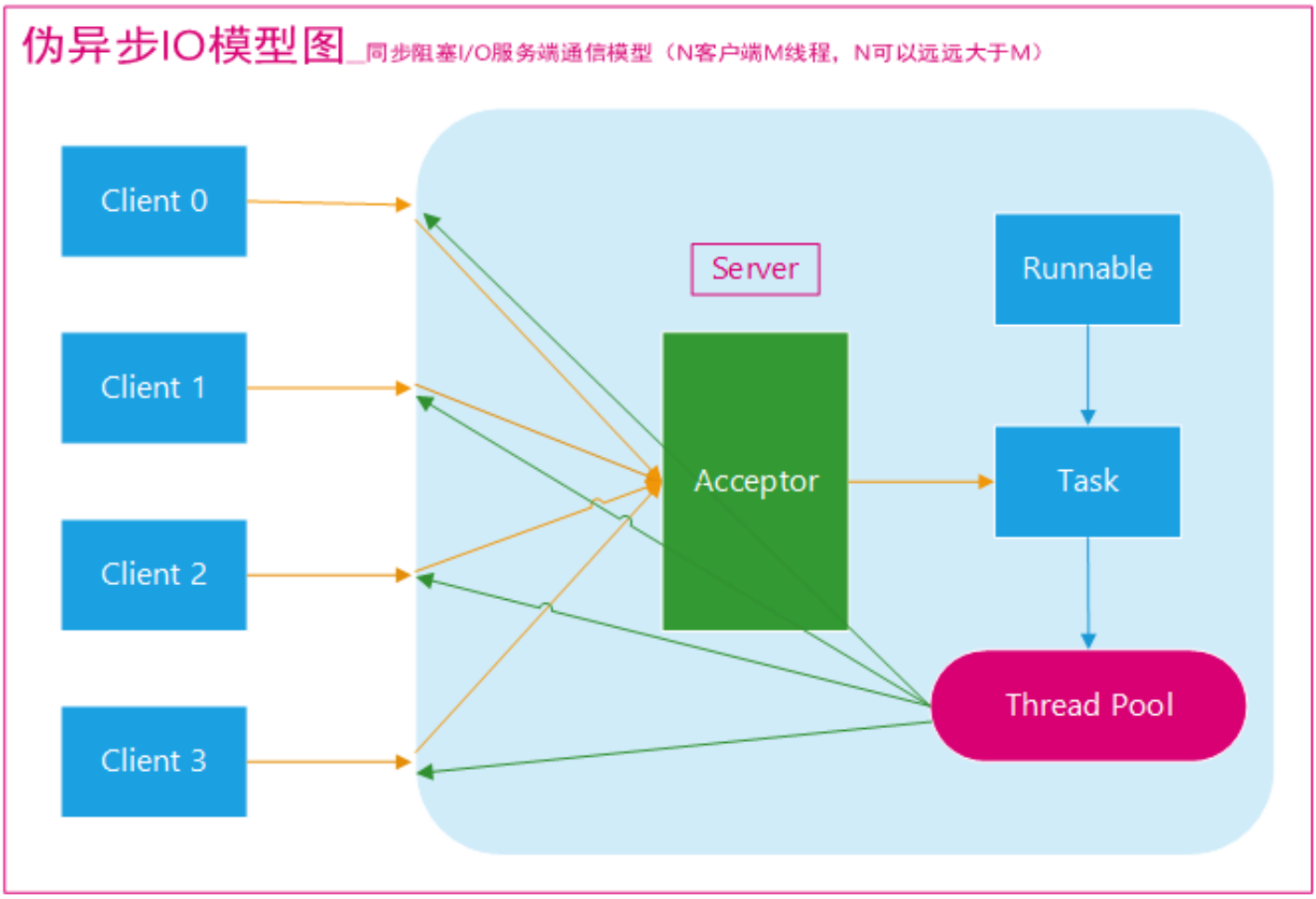
进程中的IO调用步骤大致可以分为以下四步：

- 进程向操作系统请求数据；
- 操作系统把外部数据加载到内核的缓冲区中;
- 操作系统把内核的缓冲区拷贝到进程的缓冲区；
- 进程获得数据完成自己的功能；

当操作系统在把外部数据放到进程缓冲区的这段时间（即上述的第二，三步），如果应用进程是挂起等待的，那么就是同步IO，反之，就是异步IO，也就是AIO 。

参考资料：http://blog.csdn.net/anxpp/article/details/51512200





属性 ↓ I/O 模型 →	同步阻塞 I/O（BIO）	伪异步 I/O	非阻塞 I/O（NIO）	异步 I/O（AIO）
Client 数:I/O 线程	1:1	M:N(M>=N)	M:1	M:0
I/O（阻塞）类型	阻塞 I/O	阻塞 I/O	非阻塞 I/O	非阻塞 I/O
I/O（同步）类型	同步 I/O	同步 I/O	同步 I/O（多路复用）	异步 I/O
API 使用难度	简单	简单	复杂	一般
调试难度	简单	简单	复杂	复杂
可靠性	非常差	差	高	高
吞吐量	低	中	高	高

Netty

Netty是业界最流行的NIO框架之一，具有良好的健壮性、功能、性能、可定制性和可扩展性。同时，它提供的十分简单的API，大大简化了我们的网络编程。

netty 满足了过去用 socket 进行网络编程时的几乎所有美好愿景。

参考资料：`http://blog.csdn.net/anxpp/article/details/52108238`

Linux内核select/poll/epoll

select，poll，epoll都是多路复用IO的函数。简单说就是在一个线程里，可以同时处理多个文件描述符的读写。

select/poll的实现很类似，epoll是从select/poll扩展而来，主要是为了解决select/poll天生的缺陷。

使用Select可以完成非阻塞（所谓非阻塞方式non-block，就是进程或 线程执行此函数时不必非要等待事件的发生，一旦执行肯定返回，以返回值的不同来反映函数的执行情况，如果事件发生则与阻塞方式相同，若事件没有发生则返回 一个代码来告知事件未发生，而进程或线程继续执行，所以效率较高）方式工作的程序，它能够监视我们需要监视的文件描述符的变化情况——读写或是异常。

服务端会有很多链接进来， epoll会把他们都监视起来，然后像拨开关一样，谁有数据就拨向谁，然后调用相应的代码处理。

select, poll, epoll 都是I/O多路复用的具体的实现，之所以有这三个鬼存在，其实是他们出现是有先后顺序的。

Select Poll Epoll
支持最大连接数 1024（x86） or 2048（x64）无上限 无上限
IO效率 每次调用进行线性遍历，时间复杂度为O（N） 每次调用进行线性遍历，时间复杂度为O（N） 使用“事件”通知方式，每当fd就绪，系统注册的回调函数就会被调用，将就绪fd放到rdllist里面，这 epoll*wait返回的时候我们就拿到了就绪的fd。时间发复杂度O（1）*
*fd拷贝 每次select都拷贝 每次poll都拷贝 调用epoll*ctl时拷贝进内核并由内核保存，之后每次epoll_wait不拷贝

4. 其他

交换机与路由器有什么区别?

①工作所处的OSI层次不一样，交换机工作在OSI第二层数据链路层，路由器工作在OSI第三层网络层
②寻址方式不同：交换机根据MAC地址寻址，路由器根据IP地址寻址
③转发速不同：交换机的转发速度快，路由器转发速度相对较慢。

ARP是地址解析协议，简单语言解释一下工作原理。

地址解析协议，即ARP（Address Resolution Protocol），是根据IP地址获取物理地址的一个TCP/IP协议。

(1) 首先，每个主机都会在自己的ARP缓冲区中建立一个ARP列表，以表示IP地址和MAC地址之间的对应关系。
(2) 当源主机要发送数据时，首先检查ARP列表中是否有对应IP地址的目的主机的MAC地址，如果有，则直接发送数据，如果没有，就向本网段的所有主机发送ARP数据包，该数据包包括的内容有：源主机IP地址，源主机MAC地址，目的主机的IP地址。
(3) 当本网络的所有主机收到该ARP数据包时，首先检查数据包中的IP地址是否是自己的IP地址，如果不是，则忽略该数据包，如果是，则首先从数据包中取出源主机的IP和MAC地址写入到ARP列表中，如果已经存在，则覆盖，然后将自己的MAC地址写入ARP响应包中，告诉源主机自己是它想要找的MAC地址。
(4) 源主机收到ARP响应包后。将目的主机的IP和MAC地址写入ARP列表，并利用此信息发送数据。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

> 而RARP，就是ARP的逆向，也就是将MAC物理地址转换为IP地址。

ICMP协议?

PING命令就是使用ICMP协议。通过IP协议发送的，IP协议是一种无连接的，不可靠的数据包协议。

ICMP是InternetControl Message Protocol，因特网控制报文协议。它是TCP/IP协议族的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由器是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。ICMP报文有两种：**差错报告报文**和**询问报文**。

DHCP协议?

动态主机配置协议，是一种让系统得以连接到网络上，并获取所需要的配置参数手段。通常被应用在大型的局域网络环境中，主要作用是集中的管理、分配IP地址，使网络环境中的主机动态的获得IP地址、Gateway地址、DNS服务器地址等信息，并能够提升地址的使用率。

网桥的作用?

网桥是一个局域网与另一个局域网之间建立连接的桥梁

私有（保留）地址?

A类：10.0.0.0——10.255.255.255
B类：172.16.0.0——172.31.255.255
C类：192.168.0.0——192.168.255.255

数据链路层协议可能提供的服务?

最重要的是帧定界（成帧）、透明传输以及差错检测。

网络接口卡（网卡）的功能?

（1）进行串行/并行转换。
（2）对数据进行缓存。
（3）在计算机的操作系统安装设备驱动程序。
（4）实现以太网协议。

TTL是什么？作用是什么？哪些工具会用到它（ping traceroute ifconfig netstat）？

TTL是指生存时间，简单来说，它表示了数据包在网络中的时间，经过一个路由器后TTL就减一，这样TTL最终会减为0，当TTL为0时，则将数据包丢弃，这样也就是因为两个路由器之间可能形成环，如果没有TTL的限制，则数据包将会在这个环上一直死转，由于有了TTL，最终TTL为0后，则将数据包丢弃。

- ping发送数据包里面有TTL，但是并非是必须的，即是没有TTL也是能正常工作的，
- traceroute正是因为有了TTL才能正常工作，

  通过traceroute我们可以知道信息从你的计算机到互联网另一端的主机是走的什么路径。

- ifconfig是用来配置网卡信息的，不需要TTL，

  许多windows非常熟悉ipconfig命令行工具，它被用来获取网络接口配置信息并对此进行修改。

- netstat是用来显示路由表的，也是不需要TTL的。

  Netstat 命令用于显示各种网络相关信息，如网络连接，路由表，接口状态 (Interface Statistics)，masquerade 连接，多播成员 (Multicast Memberships) 等等。

运输层协议与网络层协议的区别?

网络层协议负责的是提供主机间的逻辑通信
运输层协议负责的是提供进程间的逻辑通信

说说静态路由和动态路由有什么区别。

静态路由是由管理员手工配置的，适合比较简单的网络或需要做路由特殊控制。而动态路由则是由动态路由协议自动维护的，不需人工干预，适合比较复杂大型的网络。
路由器能够自动地建立自己的路由表，并且能够根据实际实际情况的变化适时地进行调整。动态路由机制的运作依赖路由器的两个基本功能：对路由表的维护；路由器之间适时的路由信息交换。

HTTP的长连接和短连接?

HTTP的长连接和短连接本质上是TCP长连接和短连接。HTTP属于应用层协议.
短连接:浏览器和服务器每进行一次HTTP操作，就建立一次连接，但任务结束就中断连接。

长连接:当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的 TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现长连接要客户端和服务端都支持长连接。

TCP短连接: client向server发起连接请求，server接到请求，然后双方建立连接。client向server发送消息，server回应client，然后一次读写就完成了，这时候双方任何一个都可以发起close操作，不过一般都是client先发起 close操作.短连接一般只会在 client/server间传递一次读写操作
TCP长连接: client向server发起连接，server接受client连接，双方建立连接。Client与server完成一次读写之后，它们之间的连接并不会主动关闭，后续的读写操作会继续使用这个连接。

IO中同步与异步，阻塞与非阻塞区别

同步和异步关注的是消息通信机制 (synchronous communication/asynchronous communication)
所谓**同步**，就是在发出一个*调用*时，在没有得到结果之前，该*调用*就不返回。但是一旦调用返回，就得到返回值了。
换句话说，就是由*调用者*主动等待这个*调用*的结果。
而**异步**则是相反，*调用*在发出之后，这个调用就直接返回了，所以没有返回结果。换句话说，当一个异步过程调用发出后，调用者不会立刻得到结果。而是在*调用*发出后，*被调用者*通过状态、通知来通知调用者，或通过回调函数处理这个调用。

阻塞和非阻塞关注的是程序在等待调用结果（消息，返回值）时的状态.
阻塞调用是指调用结果返回之前，当前线程会被挂起。函数只有在得到结果之后才会返回。
非阻塞：不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。

Ip地址分段

A类网络的IP地址范围为1.0.0.1－127.255.255.254；
B类网络的IP地址范围为：128.1.0.1－191.255.255.254；
C类网络的IP地址范围为：192.0.1.1－223.255.255.254、

类别	最大网络数	IP地址范围	最大主机数	私有IP地址范围
A	126（2^7-2)	0.0.0.0-127.255.255.255	16777214	10.0.0.0-10.255.255.255
B	16384(2^14)	128.0.0.0-191.255.255.255	65534	172.16.0.0-172.31.255.255
C	2097152(2^21)	192.0.0.0-223.255.255.255	254	192.168.0.0-192.168.255.255

为什么TIME_WAIT状态还需要等2*MSL（Max SegmentLifetime，最大分段生存期）秒之后才能返回到CLOSED状态呢?

因为虽然双方都同意关闭连接了，而且**挥手**的4个报文也都发送完毕，按理可以直接回到CLOSED状态（就好比从SYN*SENT状态到ESTABLISH状态那样），但是我们必须假想网络是不可靠的，你无法保证你最后发*

*送的ACK报文一定会被对方收到，就是说对方处于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。

TCP/IP接受发送缓冲区相关概念

Tcp收发需要缓冲区，udp不需要缓冲区。

每个TCP套接口有一个发送缓冲区，可以用SO*SNDBUF套接口选项来改变这一缓冲区的大小。当应用进程调用write往套接口写数据时，内核从应用进程缓冲区中拷贝所有数据到套接口的发送缓冲区，如果套接口发送缓冲区容不下应用程序的所有数据，或者是应用进程的缓冲区大于套接口的发送缓冲区，或者是套接口的发送缓冲区中有别的数据，应用进程将被挂起。内核将不从write返回。直到应用进程缓冲区中的所有数据都拷贝到套接口发送缓冲区。所以，从写一个TCP套接口的write调用成功返回仅仅表示我们可以重新使用应用进程缓冲区，它并不是告诉我们对方收到数据。TCP发给对方的数据，对方在收到数据时必须给予确认，只有在收到对方的确认时，本方TCP才会把TCP发送缓冲区中的数据删除。由于应用程序调用send的速度跟网络介质发送数据的速度存在差异，所以，一部分应用数据被组织成tcp数据报之后，会缓存在tcp socket的发送缓存队列中，等待网络空闲时再发送出去。同时，tcp协议要求对端在收到tcp数据报后，要对其序号进行ACK，只有当收到一个tcp数据报的ACK之后，才可以把这个tcp数据报(以一个struct sk*buff的形式存在)从socket的发送缓冲队列中清除。

UDP因为是不可靠连接，不必保存应用进程的数据拷贝，应用进程中的数据在沿协议栈向下传递时，以某种形式拷贝到内核缓冲区，当数据链路层把数据传出后就把内核缓冲区中数据拷贝删除。因此它不需要一个发送缓冲区。写UDP套接口的write返回表示应用程序的数据或数据分片已经进入链路层的输出队列，如果输出队列没有足够的空间存放数据，将返回错误ENOBUFS.

IO多路复用相关问题?

概念：IO多路复用是指内核一旦发现进程指定的一个或者多个IO条件准备读取，它就通知该进程。
轮询（polling）用一个进程，但是使用非阻塞的I/O读取数据，当一个I/O不可读的时候立刻返回，检查下一个是否可读，这种方法比较浪费CPU时间，因为大多数时间是不可读，但是仍花费时间不断反复执行read系统调用。
异步I/O（asynchronousI/O），当一个描述符准备好的时候用一个信号告诉进程，但是由于信号个数有限，多个描述符时不适用
I/O多路转接（I/O multiplexing）（貌似也翻译多路复用），先构造一张有关描述符的列表（epoll中为队列），然后调用一个函数，直到这些描述符中的一个准备好时才返回，返回时告诉进程哪些I/O就绪。select和epoll这两个机制都是多路I/O机制的解决方案，select为POSIX标准中的，而epoll为Linux所特有的。
（1）Select、Epoll机制相关概念(Epoll与Select机制区别)
epoll模型及优缺点？（很重要！）
主要有3点，对应于select的3个缺点：1 连接数受限 2 查找配对速度慢 3数据由内核拷贝到用户态。
1.select的句柄数目受限，在linux/posix*types.h头文件有这样的声明：#defineFD*SETSIZE 1024 表示select最多同时监听1024个fd。而epoll监视的描述符数量不受限制，它的限制是最大的打开文件句柄数目。

2.epoll的最大好处是不会随着FD（文件描述符，File Descriptor）的数目增长而降低效率，在selec中采用轮询处理，其中的数据结构类似一个数组的数据结构， epoll是维护一个队列，直接看队列是不是空就可以了。

epoll只会对"活跃"的socket进行操作—这是因为在内核实现中epoll是根据每个fd上面的callback函数实现的。那么，只有"活跃"的socket才会主动的去调用 callback函数（把这个句柄加入队列），其他idle状态句柄则不会，在这点上，epoll实现了一个"伪"AIO。但是如果绝大部分的I/O都是“活跃的”，每个I/O端口使用率很高的话，epoll效率不一定比select高（可能是要维护队列复杂）。
epoll是通过每个fd定义的回调函数来实现的。只有就绪的fd才会执行回调函数

3.使用mmap加速内核与用户空间的消息传递。无论是select,poll还是epoll都需要内核把FD（文件描述符）消息通知给用户空间，如何避免不必要的内存拷贝就很重要，在这点上，epoll是通过内核于用户空间mmap同一块内存实现的。

mmap并不分配空间, 只是将文件映射到调用进程的地址空间里, 然后你就可以用memcpy等操作写文件, 而不用write()了.写完后用msync()同步一下, 你所写的内容就保存到文件里了.

4、支持电平触发和边沿触发（只告诉进程哪些文件描述符刚刚变为就绪状态，它只说一遍，如果我们没有采取行动，那么它将不会再次告知，这种方式称为边缘触发）两种方式，理论上边缘触发的性能要更高一些，但是代码实现相当复杂。
poll的实现和select非常相似，只是描述fd集合的方式不同，poll使用pollfd结构而不是select的fd_set结构，其他的都差不多。

参考资料：https://segmentfault.com/a/1190000003063859

Linux下du，df的区别?

du查看目录大小，df查看磁盘使用情况。

du会把指定目录下所有文件、目录、目录下的文件都统计。是建立在文件系统能看到的的确确是有这样一些文件的基础上的。也就是说我们能在文件系统里面看到的文件才会被du统计。

df命令的功能是用来检查linux服务器的文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

参考资料：
计算机网络相关问题：http://blog.csdn.net/u012286517/article/details/50373759