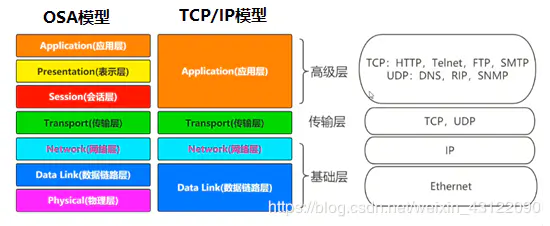
# 计算机网络编程



**OSI7层**：

应用层：产生网络流量和用户交互的软件

表示层：加密，压缩，开发人员

会话层：服务端和客户端建立的会话；查木马netstate

传输层 可靠传输建立会话 不可靠传输 流浪控制

网络层 IP地址编址 选择最佳路径

数据链路层：输入如何封装 添加物理层地址

物理层 电压接口比标准

**应用层** http https ftp DNS SMTP POP3 RDP

**传输层两个应用协议场景**

TCP 分段编号流量控制建立会话netstaet -n(查看TCP/IP连接状态)

UDP 一个数据包就能完成数据通信不用建立会话

网络层

IP（RIP OSPF BGP） ICMP IGMP ARP

**传输层和应用层之间的关系**

http=TCP+80

https=TCP+443

ftp=TCP+21

SMTP=TCP+25

POP3=TCP+110

RDP=TCP+3389

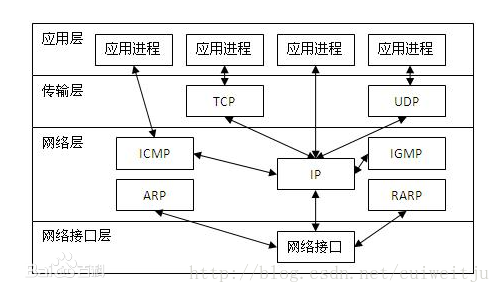
共享文件夹=TCP+445

SQL=TCP+1433

DNS=UDP+53 or TCP+53

**应用层和服务之间关系**

**应用层安全**



源IP地址和目的IP地址以及源端口号和目的端口号的组合称为套接字

## √路由器和交换机的区别

交换机：工作在数据链路层

路由器: 路由工作在第三层（网络层）；负责host-host的路径。又国际协议ICMP

网关：现在指的网关就是路由器IP

**工作层次不同：**  
交换机主要工作在数据链路层  
路由器工作在网络层  
**转发依据不同：**  
交换机转发所依据的对象时：MAC地址。（物理地址）  
路由转发所依据的对象是：IP地址。（网络地址）  
**主要功能不同：**  
交换机主要用于组建局域网，  
而路由主要功能是将由交换机组好的局域网相互连接起来，或者接入Internet。  
交换机能做的，路由都能做。  
交换机不能分割广播域，路由可以。  
路由还可以提供防火墙的功能。  
路由配置比交换机复杂。

## √地址位数

IP：32位 4段

Mac：48位（00-16-EA-AE-3C-40）6位 16进制数（1个4位），每段（1byte）

## √如何判断所在子网和广播地址

127.0.0.1回环地址



A类网络号：1-127 子网数量2

上面是网络号+主机号

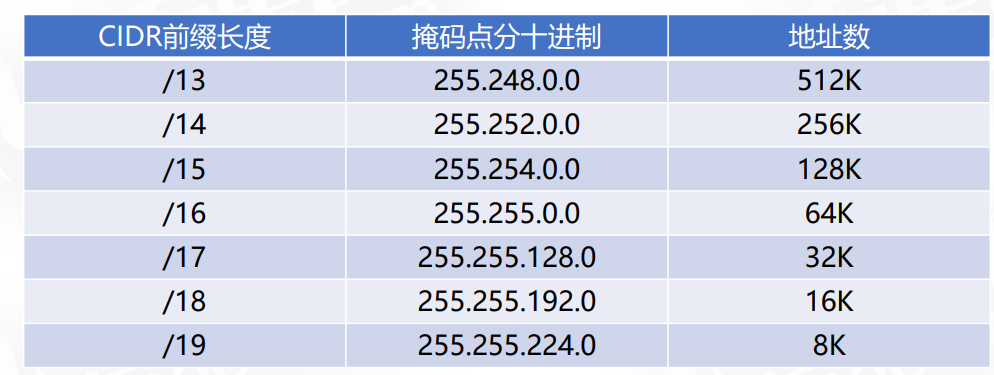
有时候一个公司只有256个主机，申请一个网段太浪费了

于是有了网络号+子网号+主机号

子网掩码是为了判断网络号



无分类编址



193.10.10.129/13：表示前面13位是网络地址后面19位是主机地址，2^19=512K

广播地址就是最后一个主机号。网关设备对于有NAPT表记录出入的地址和端口

## √IPv4不够用是如何解决的🡪NAT技术中的NAPT

转NAT技术：内网映射成网关端口，对外就只有一个IP。

内网地址内部使用，不同内网，IP可重复，但要避免和外网一样

外网是全球范围使用，全球公网唯一。

三类内网网址支持不同量级

10开头支持千万

172开头支持百万

192.168开头支持万

## NAT(Network Address Translation网络地址转换)

用于内网中主机要和因特网上的主机通信。由NAT路由器将主机的本地IP地址转换为全球IP地址，分为静态转换和动态NAT转换。NAT优点，解决了IPV4数量受限问题；隐藏了内网IP，因此也增加了内网的网络安全；此外，NAT对来自外部的数据查看其NAT映射记录，对没有映射的数据包进行拒绝，提高了网络安全性。

缺点：NAT设备会对数据包进行修改，降低了发送数据的效率；

有些协议无法通过NAT，就需要通过穿透技术解决

静态转换：一个公网IP对应一个私有IP，是1对1的转换；只是进行理论IP转换，端口没有转换

动态转换：NAPT端口多路复用技术，转换IP地址同时进行传输层的端口转换。对外只有一个公网IP，通过端口来区分不同私有IP主机数据（session id值）。NAPT又分为完全锥形FC（对访问的IP端口都不限制）受限雏形（IP受限，端口不受限）；端口受限型NAT（IP端口都受限）对称型NAT（对每个外部主机或端口的会话都会映射为不同的端口）。为建立P2P通信连接就需要判断NAT的类型，才能设计合理的通信方案。

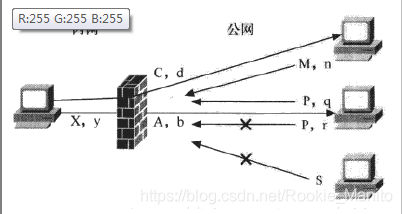
穿透技术

完全锥形nat：内网主机IP和端口{x:y}🡪{A:b},公网IP都可以访问A:b

受限锥形nat：比如公网M（*{x:y}🡪{A:b}）*，只有M可以访问

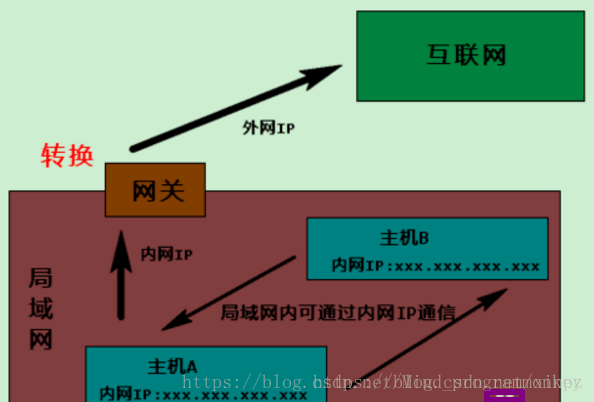
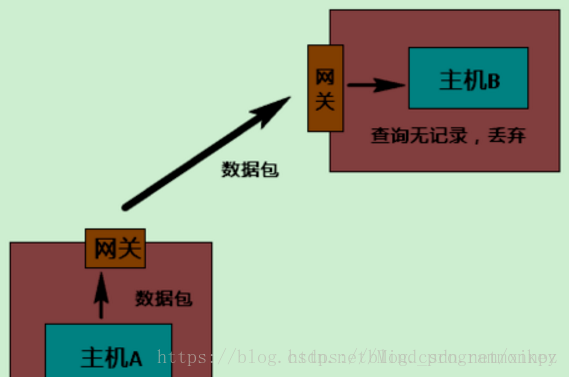
端口受限圆锥nat：内网主机IP和端口{x:y}🡪{A:b}只有M的p端口可以访问

对称nat：每个公网对应一个映射



当内网中的主机没有静态IP地址要被外网稳定访问时可以使用内网穿透

在网络穿透技术中就需要判别NAPT类型，

及时知道主机B在外网对于的IP和端口，A给B发送数据会被网关拦截丢弃。

要想解决以上两大障碍，我们需要借助一台具有公网 IP 的服务器进行桥接

内网穿透是通过服务器中继转发数据来实现的将内网端口映射到公网，速度上没有P2P直连的快

还有P2P内网穿透，就需要知道NAPT类型了

## √IPv6

128位

取消了arp，用多点传播代替。

## √IP层相关协议

### ARP（address resolution protocol）地址解析协议

询问IP对于的MAC

☆主机和主机通讯的数据包依据OSI从上到下进行数据封装，数据封装完毕再向外发出。

所以在局域网的通信中，不仅需要源目的IP地址和需要源MAC封装

* 上层应用更关心IP地址，不关心MAC。需要通过ARP协议获知目标主机的MAC地址完成数据封装

过程描述: 在ping的时候只是输入了IP地址封装到链路层的时候发现没有在ARP缓存表中找到对于MAC，于是向发出一个请求包到局域网，局域网如果有该IP就应答原路返回；没有就由网关对外网发起arp请求，找到就返回。

缓存表具有时效性，电脑设备重启就会清空。arp -a可以查看缓存表

广播请求单播回应。为了提高效率，请求主机会广播arp包。单播返回。

报文有请求方去IP和MAC

ARP协议完成了**IP地址与物理地址的映射**。每个主机都设有一个**ARP高速缓存**，里面有所在的局域网上的每个主机和**路由器的IP地址**到**硬件地址**的映射表。当源主机要发送数据包到目的主机时，会检查自己的**arp高速缓存**中有没有目的主机的MAC地址，如果有，则直接将数据包发到这个MAC地址，如果没有，就向所在的局域网发起一个ARP请求的广播（带上自己的IP地址和MAC映射）。收到请求的主机检查自己的IP地址和目的主机的IP地址是否一样，如果一样，则先**保存源主机的映射**到自己的ARP缓存，然后给源主机发送一个ARP响应数据包。源主机收到响应数据包后，先添加目的主机的IP地址与MAC地址映射，再进行数据传输。如果源主机一直没有收到响应，表示ARP查询失败。

如果所要找的主机和源主机不在一个局域网，那么就要通过arp找到一个位于本局域网上的某个**路由器的硬件地址**，然后把分组发送给这个路由器，让这个路由器把分组转发给下一个网络。剩下的工作就由下一个网络来做。

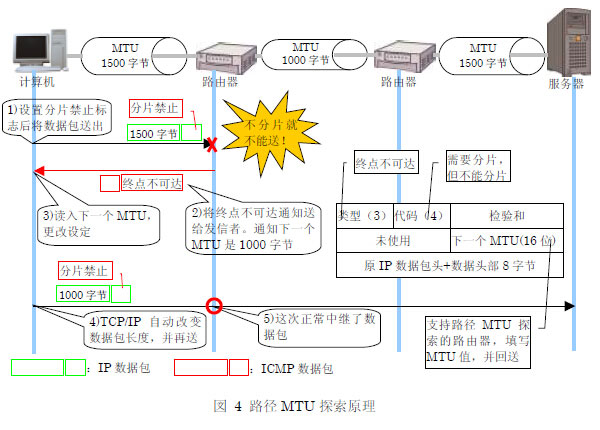
### ICMP协议（国际控制报文协议）

差错报告报文（错误通知）和询问报文（信息查询）（TTL表示记录了IP数据报文在网络中的寿命，TTL=0网络设备要丢弃该报文）

在IP报文传输过程中，可能某个路由器不能处理所有数据包，将数据包丢弃；搞错端口号，被服务器拒绝。ICMP协议是为了辅助IP协议，交换各种各样的控制信息而被制造出来了。

应用：ping，Traceroute应用

ICMP可以探测网路最大MTU

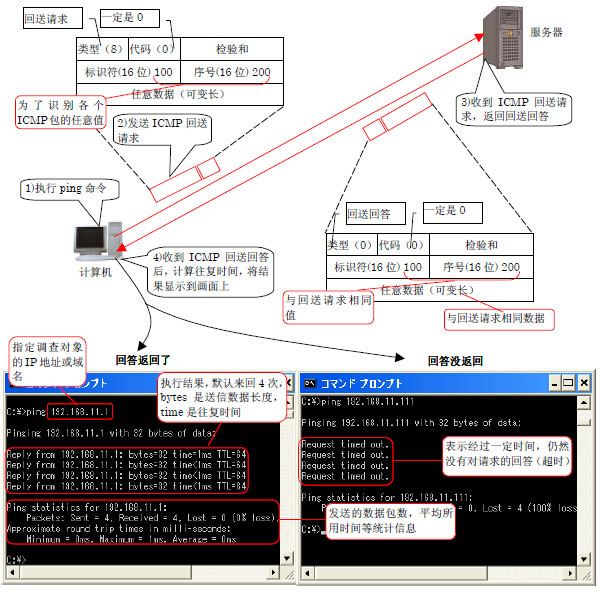


MTU是最大传输单元：1500byte（以太网）这是是由网路的种类决定。

如何探测网路MTU：探索不用分片的IP数据包。

ICMP可以改变路由，在计算机不知知道发给哪个路由，就发给默认路由，默认路由通过ICMP报文通知发送方，从而改变路由。

ICMP实现ping

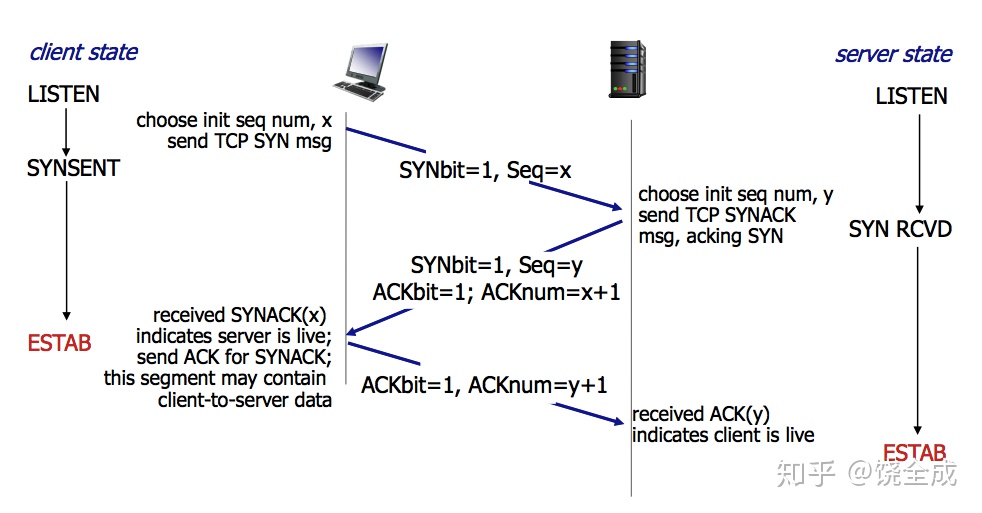


向目标服务器发出会送请求（类型是8，代码是0）。在这个会送请求报文里，还追加了标识符和序列号字段（分别是16位的字段来确定是自己发出去的，送信数据长度）。Ping在接收到回报之后，根据时间差就能知道往返时间

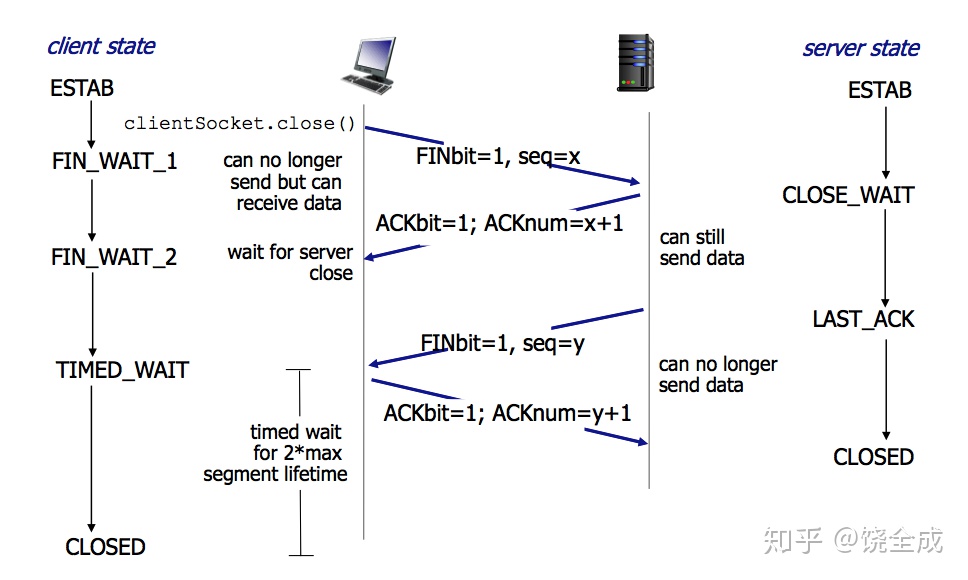
### IGMP是IP组播管理协议

组播

## TCP三次握手和四次挥手过程



1. 客户端发送syn(synchronization)同步位给服务端，自己状态编程synsend状态
2. 服务端收到syn后向客户端发送syn同步位和ack确认位，自己状态变为sys rcvd
3. 客户端收到同步位和确认位后，确定了服务器端具有接收和发送能力，状态改为建立连接状态，同时给服务器端发送ack确认
4. 服务器端收到ack后知道客户端也有发送和响应能力，于是建立连接



TCP连接是双向传输对等模式（双方可以同时发送和接收数据）。

1. 服务器要关闭连接，向服务器发送fin段
2. 服务器收到fin后回复ack确认位。
3. 客户端收到ack后，关闭了客户端的发送连接。但此时服务端可以发送数据给客户端

等服务端发送完所有数据后，会发送一个Fin段关闭自己方向上的连接。

1. 客户端收到fin段后向服务器发送ack确认
2. 服务器关闭了连接，客户端等待两个报表生命周期后页关闭连接

## 为什么要三次握手，能不能两次握手或者四次握手呢？

三次握手是为了确认双方都有接收能力。

客户端发送syn同步位到客户端收到服务端回复syn和ack确认位，此时客户端知道服务端有接收和发送能力，于是建立了连接；接着客户端发送ack到服务器，服务器知道客户端有发送和接收的能力，于是建立连接。

如果是两次握手，服务器端是无法知道客户端是否有接收能力的。

同时失效连接可能造成服务器建立错误连接。浪费了服务器端大量资源。

四次握手也是可以的，只是消耗的资源比三次握手更多

## 释放的时候为什么要四次？

TCP是双向传输对等连接的。也就是说服务器和客户端双方都能同时发送和传输。

客户端发出断开请求，服务器回复ack，只是客户端不再发送信息；服务器端会把上次请求的信息发完后，发出断开请求fin，客户端收到会回复ack，此时服务器端的连接彻底断开；在等待两个报表生命周期后服务器端处于关闭状态

## Time\_wait等待多长时间？为什么最后客户端要等待两个MSL报表周期才断开呢？

因为客户端必须在服务器关闭之后关闭，如果在两个报表周期都没有再次受到服务器端的fin位，说明服务器已经受到之前发送的ack并且关掉了。

如果不等待直接关掉连接，则服务器端可能因为ack丢失等原因无法关闭造成资源浪费。

另一方面，可以使得失效的报文可以报废，从而不影响新连接。

在连接处于2MSL等待阶段，任何迟到的报文段都会被丢弃。

不过实际应用中，可以通过设置so\_reuseaddr选项，不必等待2MSL结束即可使用被占用的端口。

## 如果已经建立了连接，但客户端出现了故障怎么办？

服务器每次受到客户端的请求后都会重新复位一个计时器，时间通常位2h，若两小时内都没有受到客户端任何数据，则服务器会发送一个探测文段，以后每隔75s发送一次，连续10个探测文段都没翻译，服务器就认为客户端发生故障，接着就关闭连接。

## √第三次握手中，如果客户端的ACK未送达服务器，会怎样？

服务器端没有接收到ack，就会每隔3s重新发送之前的ack和syn，默认重发5次后就会自动关闭连接进入closed状态。

客户端会出现两种情况：

服务器重发过程中，客户端收到并发送ack，进入establish状态

在服务器关闭之后向服务器发送数据，服务器会以RST包（Reset？？）应答

Rst包用于强制关闭TCP连接。

## √MSL、TIME\_WAIT、CLOSE\_WAIT

MSL：maxium segment Lifetime, 报文最大生存时间。TCP报文是IP报文的数据部分；超过报文时间这个IP报文就会被丢弃

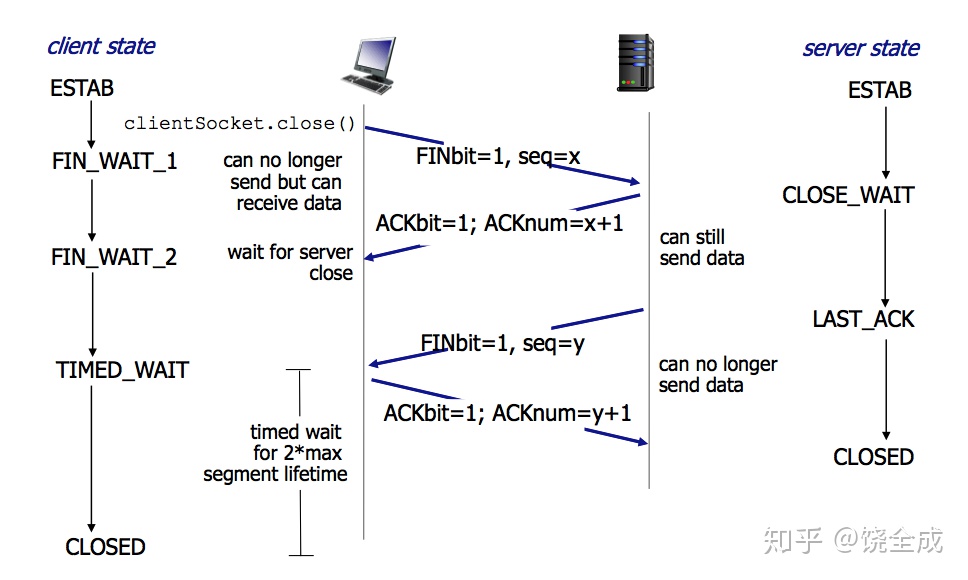
MSL规定是2分钟，实际应用常用30s、1分钟

TIME\_WAIT状态：TCP释放过程，服务器返回ack后，客户端就停止发送数据，状态变为等待服务器关闭状态，一直到服务器关闭，客户端才变成关闭状态。

Close\_wait: 服务器向客户端发送ack后，还会向客户端发送数据，这段时间服务器处于close\_wait状态。一直到结束发送数据，向客户端发送fin段，状态变为等待最后确认状态，last\_ack

## 大量TIME\_WAIT状态存在原因

Time\_wait发生在服务器端发送fin端请求关闭时，客户端等待的2MSL过程。

1. TCP连接多用于短连接。尤其实在http请求中，如果connection头部取值被设置位close时，基本都由服务器发起主动关闭。
2. 在每个tcp连接中，在等待对方传输完数据，等待对方关闭过程中一直处于TIME\_wait状态，在收到对方FIN请求后还要等待2MSL。
3. 

如何解决大量TME\_WAIT存在导致新连接创建失败的问题？

1. 客户端http请求头，connect设置位keep alive，即保持存活一段时间
2. 服务器端

允许timewait状态的socket被重用，通过设置好resuseaddr选项即可

缩减time\_wait时间为1MSL。

http请求中connect参数可以是close和keep\_alive

默认是close也就是说，浏览器发起请求后，告知服务器在完成请求后关闭连接，那么主动关闭方就是服务器。Time\_wait就处于服务器端。

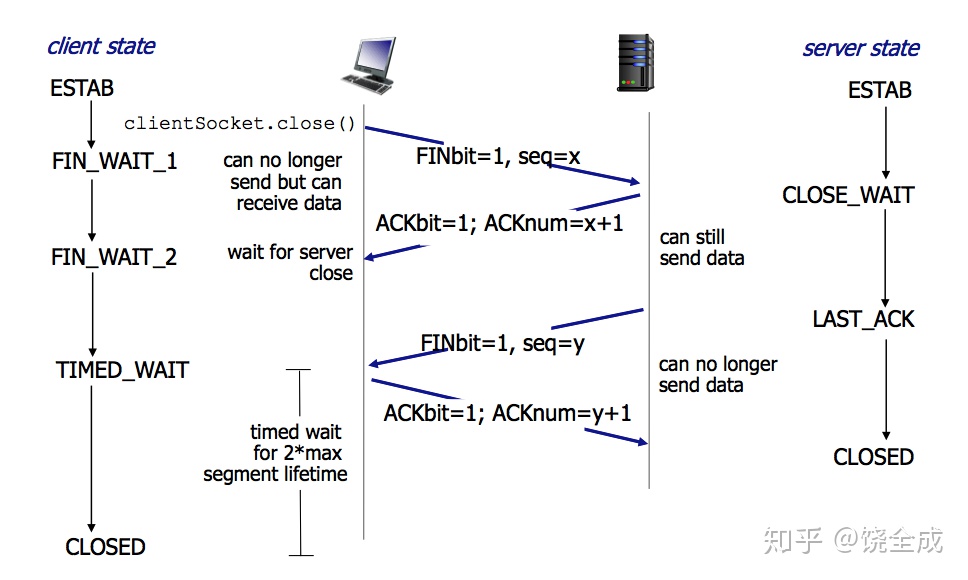
## √Time\_wait状态存在的必要性？起始就是问2MSL的必要性

可靠地实现TCP全双工连接的终止。

1.主动发最后发出的ack是用来关闭被动方的连接的，ack在传输过程中会迟到甚至丢失，如果是迟到，如果没有有timewait阶段，会让新连接误以为是新TCP建立的数据。因此需要timewait取等待所有报文被被抛弃。以免影响新连接。

2. 同时确认ack成功被被动发接收，成功关闭被动发的连接。

具体体现在被动端没有收到最后的ack，多次发送fin，对新连接产生影响



## √大量close\_wait状态

Close\_wait发生在服务器端接收到客户端发来的Fin段

服务器迟迟没有发出FIN导致的

用户服务器的程序实现有问题导致的大量close\_wait

我们要讨论的场景是什么？我们先介绍两个系统调用，前面也提到并且用到的close(socket)和shutdown(socket,HOW)接着往下分析。

我们知道一个进程打开一个socket，然后此进程fork出子进程的时候，父进程已打开的socket是会被继承的，即子进程能够继续访问这个socket。其结果就是，一个socket被两个进程打开，一个父进程和一个子进程，此时socket的引用计数会变成2。

调用close(socket)时，内核先检查socket上的引用计数器：如果引用计数大于1，那么将这个引用计数减1，然后直接返回。如果引用计数等于1，那么内核才会真正关闭此socket。(通过发送FIN到对端来关闭TCP连接)。因此这个socket并没有被关闭（也就是没有向对端发送fin）。被动端因为代码自身原有，没有回调去主动关闭发，一直没有发出Fin

调用shutdown(socket，HOW)时，内核不会检查此socket对应的引用计数器，直接向对端发送FIN来关闭TCP连接。需要同时关闭父端的socket

例子2 Server 程序处于CLOSE\_WAIT状态，而不是LAST\_ACK状态，说明还没有发FIN给Client，那么可能是在关闭连接之前还有许多数据要发送或者其他事要做，  
导致没有发这个FIN packet。

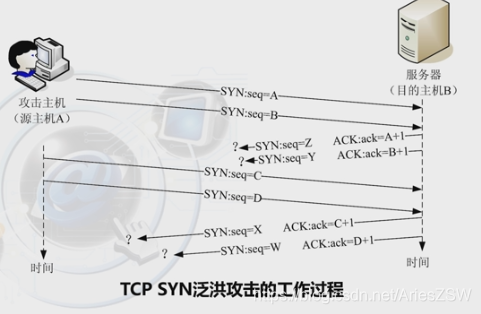
流氓软件到了服务器之后就会根据这个让被动方连接处于一直写状态，消耗资源。

解决方法：

当主动关闭的一方发送FIN到被动关闭这边后，被动关闭这边的 TCP马上回应一个ACK过去，同时向上面应用程序提交一个ERROR，因此可以在被动方的上层应用增加加closesocket（）指令

int nRet = recv(s,....);  
if (nRet == SOCKET\_ERROR)  
{  
// closesocket(s);  
 return FALSE;  
}

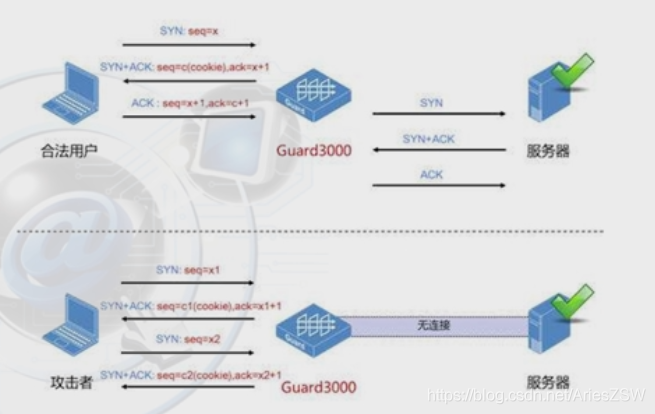
## SYN泛洪攻击



目的主机B的SYN+ ACK确认报文段全部得不到回复。  
在目的主机B的队列中存在大量的”半开放状态”的连接，最终将队的存储空间填满，并因资源耗尽而瘫瘓.

方法：

* 1. 缩短syn timeout等待超时时间。能提升几倍
  2. Syn cookie是linux内核一部分



SYN Cookie是对TCP服务器端的三次握手做一些修改，专门用来防范SYN Flood攻击的一种手段。它的原理是，在TCP服务器接收到TCP SYN包并返回TCP SYN + ACK包时，不分配一个专门的数据区，而是根据这个SYN包计算出一个cookie值。这个cookie作为将要返回的SYN ACK包的初始序列号。当客户端返回一个ACK包时，根据包头信息计算cookie，与返回的确认序列号(初始序列号 + 1)进行对比，如果相同，则是一个正常连接，然后，分配资源，建立连接。实现的关键在于cookie的计算，cookie的计算应该包含本次连接的状态信息，使攻击者不能伪造。

* 1. 防火墙的带宽控制技术
  2. 基于IP的防守：DNS域名解析到新IP。攻击者攻击的是个空IP。CDN加速，隐藏了真实的IP，将流量分散到不同节点

## 客户端不断进行请求链接会怎样？DDos(Distributed Denial of Service)攻击

DOS就是拒绝服务

DDOS就是攻击者有多台主机对服务器进行攻击

SYN洪攻击是ddos的一种

攻击端发起TCPsyn建立请求，服务器端返回ack后，供给端不给予回复，使得服务器端进入等待区。等待区结束后又发起虚假请求，使得服务器资源耗尽，直到瘫痪

预防没法做到，只能解决DDos

1. 减少time\_out时间
2. 对于syn攻击可以采用syn\_cookie, 确定了cookie值再分配资源
3. 基于IP攻击的，可以DNS将域名解析到新IP
4. 防火墙的带宽控制技术
5. CDN加速，受到攻击的时候，将流量分散到各个节点

## 流量控制具体实现？用来干嘛？受谁控制，由谁确定

TCP有发送缓冲区和接收缓冲区

由接收端请确定控制，接收端有个缓冲区放置没有确认应答的数据，。。。

TCP的发送方和接收方都有各自的缓冲区。利用滑动窗口机制是实现数据传输端的流量控制。不在报文段，数据传输端的发送窗口不能超过接收端给出的接收窗口的数值。灵活控制传输速度。如果没有窗口，就需要接收端确认数据后才能发送，效率大大降低。

有了窗口，在发送端等待确认的时候也可以继续发送数据。

**为了降低丢包率同时增加传输速率**，保证数据的可靠传输

关于滑窗的相关知识点：

窗口大小16位放在TCP首位“窗口大小中”；

接收端通过ack来通知窗口大小；

窗口越大，网络吞吐量越大

窗口大小是指无需的等待确认应答可以一次连续发送的最大值。

操作系统内核为了维护滑窗，会开辟发送缓存区，记录哪些数据没有应答；只有确认过的数据才从缓冲区删除；

数据接收端一旦发现自己缓冲区快满了，就会将窗口大小设置成更小的数；传输端就会减慢传输。

接收端发现自己的数据满了，会将窗口设置位0，发送端停止发送。

一个socket的两端，都会有send和recv两个方法，如client发送数据到server，那么就是客户端进程调用send发送数据，而send的作用是将数据拷贝进入socket的内核发送缓冲区之中，然后send便会在上层返回。

## TCP发送报文时机选择？

①TCP会维持一个变量，它等于最大报文长度MSS，只要缓冲区存放的数据达到MSS自己才打包成以一个TCP报文发出去

②发送方的计时器期限到了，就把缓冲区数据发出。

③应用软件指明发送

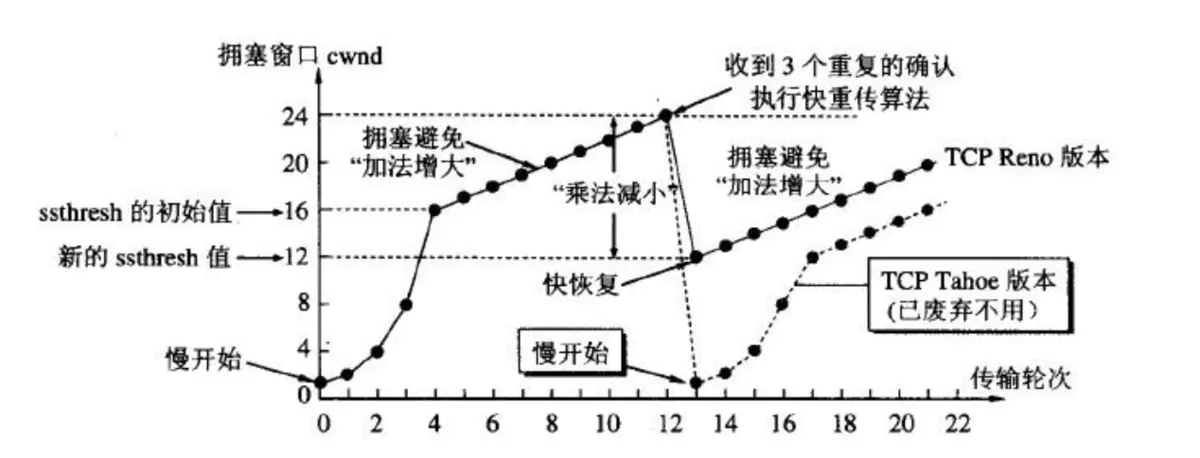
## 拥塞控制具体实现？用来干嘛

慢启动，快恢复，拥塞避免，快重传

拥塞控制是为了解决过多的数据注入到网络,导致网**络奔溃**,超过负荷,拥塞控制包含四个策略

* + 1. 慢开始。窗口先设置成1，每次传输量指数增长直到达到阈值。慢开始结束
    2. 拥塞避免。每次传输线性增长，直到触发了网络拥塞，窗口大小和阈值都变成拥塞时最大值的一半。然后重新开始慢重传
    3. 快重传：当接收方收到顺序错误的数据时，不再接收数据，同时重复发起对于之前数据的确认，发送方得知自己数据在发送中丢失，立刻发起重传，不需等待下一次发送信息时一起发送过去。且重传和拥塞一样，进入快恢复阶段。
    4. 快恢复。

在发生重传和拥塞的时候，结束拥塞避免阶段，直接进入线性传输阶段，提高恢复速度。（窗口大小直接为阈值，线性增长）



## 流量控制和拥塞控制的区别

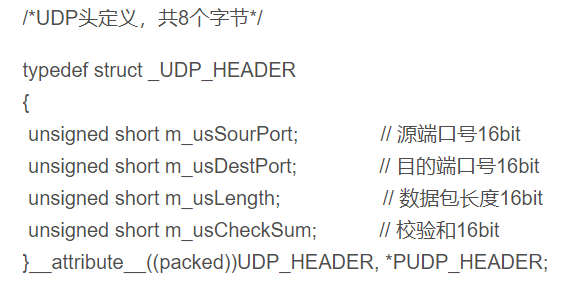
流量控制是会话两端接受能力之间的协调，保证数据的正确传输，同时尽可能提高传输速率

拥塞控制是针对网络情况，避免过多数据进入网络，是由带宽等因素限制的

**Udp头部信息**

64位，8字节

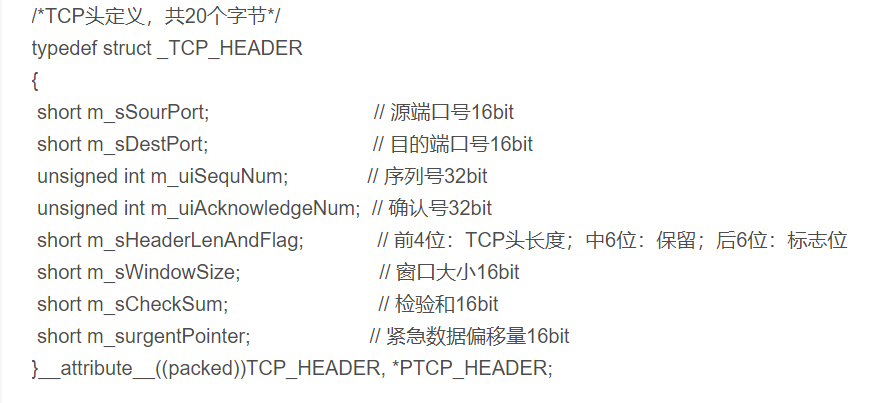




**Tcp头部信息**

20字节





## √ 三次握手最后一个 ACK 可以携带数据么？

可以携带数据，因为这个第三次就是确认建立的恢复。

可以携带也可以不携带。

不携带的话，就不需要消耗seq

## √TCP可靠性如何实现的？

校验、排序、去重、应答、超时重发、流量控制、拥塞控制

数据包校验：目的是检测数据在传输过程中的任何变化，若校验出包出错，则求其报文段并且不给出响应，这时TCP发送数据端超时后会重发数据；

对失序数据包重排列：既然TCP报文段作为IP数据报传来传输，而IP数据报的到达可能会失序，因此TCP报文段的到达也可能会失序列。TCP将对失序的数据进行重新排序，然后才交给应用层。

丢弃重复数据：对于重复数据，能够丢弃重复数据

应答机制：当TCP收到TCP连接另一端的数据，它将发送一个确认。这个确认不是立刻发送，通常将推迟几分之一秒；

超时重发：当TCP发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能收到一个确认，将重发这个报文段

**流量控制**：TCP连接的每一方都有固定的缓冲空间。TCP的接收端只允许另一端发宋接收端缓冲区所能容纳的数据，这可以放置较快主句只是较慢主机的缓冲区溢出，这就是流量控制。TCP使用的流量控制协议是可**变大小的滑动窗口协议**。

**拥堵控制**：慢启动机制，先发少量的数据，就像探路一样，先摸清当前网络的拥堵状况后，再决定按照多大的速度传送数据。设置一个慢启动阈值，当以指数增长达到阈值时就停止质数增长，按照线性增长方式增长；线性增长达到网络拥堵时立即乘法减小，拥堵窗口置回1，进项新一轮慢启动。每次收到一个ack应答，拥堵窗口就+1

**慢启动，快恢复**：先发送少量数据，确定拥堵情况后，再决定按照多大速度传输数据。设置一个慢启动阈值，当以指数增长到阈值就按照线性增长方式，遇到网络拥堵立刻乘法减小。拥堵窗口重置1，开启新一轮慢启动

聊软项目怎么做 UDP 可靠传输的？

发送数据包的时候调用的什么 API

## 运行再TCP或UDP上的应用层协议

运行再TCP协议上的协议：

HTTP 超文本传输协议

HTTPS 安全超文本传输协议

FTP 文件传输协议

POP3 邮局协议，收邮件用

SMTP 简单邮件传输协议，用来发送电子邮件

TELNET 网络电传，通过终端登入到网络

SSH （secure shell,用于替换安全性差的TELNET），用于加密安全登入

**运行在UDP协议上的协议**

DHCP 动态主机配置协议，动态配置IP地址

NTP network time protocal网络事件协议，用于网络同步

BOOTP boot protocol 启动协议，应用于无盘设备

**运行在TCP和UDP协议上**：

DNS，用于完成地址查找，邮件转发等工作

ECHO（回绕协议）用于差错及测量应答时间（运行TCP和UDP协议上）

SNMP（简单网络管理协议），用于网络信息的收集和网络管理

DHCP（动态主机配置协议）动态IP地址

ARP地址解析协议，用于动态解析以太网硬件的地址

域名解析（DNS）的过程

1. 在浏览器输入域名，操作系统会检查机子本地的host文件是否有这个网址映射关系，有就完成域名解析
2. 如果hosts里面没有域名映射，则**递归**查找本地DNS解析缓存，是否有这个网址映射关系，如果有，直接返回完成域名解析。
3. 如果hosts文件和本地缓存都没有相关的映射，首先会找TCP/IP参数中设置的首选DNS服务器（本地DNS服务器），此服务器收到查询时，如果要查询域名，包含在本地配置区域资源中，则返回解析结果个客户及，完成域名解析。
4. 如果要查询的域名不在本地DNS服务器区域解析，但改DNS服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析。
5. 如果本地DNS服务器本地区域文件和缓存解析都失效，发给转发器，没有转发器就发给根DNS，返回一个负责该顶级域名服务器的IP。本地DNS发送请求到该IP

, 重复45步骤得到域名解析

## 请你说明在浏览器输入网址到看到整个内容的过程

在应用层、传输层，链路层，物理层发什么什么

* 1. DHCP(动态主机配置协议dynamic hosts configure protocal)配置主机信息

主机一开始没有IP地址以及其他信息，那么需要使用DHCP来获取

主机生成DHCP请求报文，将这个报文放在具有目标端口67和源端口68的**UDP**报文段中。

该报文被放入一个具有广播IP目标地址255.255.255.255和源IP地址0.0.0.0的IP数据报中。

该数据报被放置在MAC帧中，该帧具有目标地址ff.ff.ff.ff.ff.ff(6byte) 6字节，将广播到与交换机连接的所有设备。

连接在交换机的DHCP服务器收到广播帧后，不断地向上分解得到IP数据报、UDP报文，之后生成DHCP、Ack报文。该报文包含一下信息（IP地址、DNS服务的器的IP地址，默认网关路由器的IP地址和子网掩码）；该报文又会被放入**UDP报文**段中，UDP报文段又被放入IP数据报中，最后放入Mac帧中。

该帧的目的地址是请求主机的MAC地址，因为交换机之前发送广播帧之后就记录了MAC地址到其转发接口的交换表项，因此交换机可以原路发返回。

主机收到该帧后不断得到DHCP报文。之后就配置它的IP地址、子网掩码和DNS服务器的IP地址，并在其IP转发表中安装默认网关。

* 1. ARP解析mac地址

DHCP只知道网关路由器的IP地址，不知道MAC地址。主机发送ARP报文，目的地址设置为广播地址，网关路由器接收到该帧后发现主机IP和端口IP匹配，就返回ARP回答报文，包括了网关MAC地址，发回给主机。

* 1. DNS解析域名

获得网关的MAC地址后，可以继续DNS解析。网关（路由器具有内部网关协议RIP，OSPF）收到DNS报文后根据路由表中以及配置好的网关到达DNS服务器。DNS在数据库中查找带解析的域名。

找到DNS记录后发送DNS回答报文，将报文放在UDP报文段，UDP在放在IP数据报中，通过**路由**反向转发回网关，并经过以太网交换机到到主机

* 1. 获得HTTP服务器的IP地址后，主机就能够后生成TCP套接字，该套接字将用于向Web服务器发送HTTP Get报文。与HTTP服务器进行三次握手来建立TCP连接，生成一个具有目标端口80的TCP SYN报文段（同步标志置1），并向HTTP服务器发送该报文段。HTTP服务器收到该报文段之后生成TCP SYN ACK（确认标志）报文段，发回给主机。主机收到后状态编程建立通信，发回ack给HTTP服务器，TCP通信建立，HTTP生成 GET报文并发给HTTP服务器；HTTP生成一个响应报文，将web页面内容放入报文主体中，发回给主机。浏览器收到HTTP响应报文后抽取web页面内推，之后进行渲染，显示web页面。

在DHCP之后为什么还要ARP获得网关MAC？？？同一个网段内的寻址是通过以太网协议实现的，以太网协议需要知道MAC地址；其他无PPP，SLIP。

## 请说明VPN如何工作？

虚拟专用网络

1. 让世界上任意两台电脑进入一个虚拟的局域网，用起来和一个家庭的局域网无区别
2. 访问外网

实现过程：工作在链路层即IP层

具体说，vpn是通过编写一套网卡驱动并注册到操作系统实现的虚拟网卡卡，这样数据经网卡收发就能将进行拦截处理。

翻墙用途：需要在vps部署vpn server，客户端数据经过虚拟网卡加密封装后转发给server，它再转发给目标服务器。这个给和ss server类似。由vpn协议加密从而绕过GFW（great fire wall）实现访问墙外网站。

IP地址编码分为那两部分？

IP的地址编码分为哪两个部分？

主机号+网络号

要和子网掩码按位与后可以知道哪些是主机号和网络号

## UDP最大可以发多少字节？1472

受限于底层的限制

1. 链路层数据帧内容最大1500,MTU最大传输单位位1500
2. 网络层，因为IP包的首部要占用20字节，MTU=1500-20（IP头）=1480
3. 传输层，对UDP包的首部要占用8字节（TCP头部开销20字节），所以这里MTU为1480-8=1472.当我们的UDP包中的数据多余MTU（1472）时，发送方IP层需要分片进行传输，而接收方IP层则需要进行数据报重组，由于UDP是不可靠的传输协议，如果分片导致重组失败，将导致UDP数据包被丢弃。

从上面分析来看，在普通局域网络环境下，UDP数据最大为1500-20-8=1472字节

在网络编程中，internet中的路由器可能有设置成不同的值（小于默认值），internet上的标准MTU576，所以UDP编程数据长度最好在576-20-8=548字节以内。

Socket编程时，UDP协议要求包小于64K。TCP没有限定，TCP包头中没有“包长度”字段，完全依靠IP层区处理分帧。这就是为什么TCP常常被称左一种“流协议”的原因。开发者在使用TCP服务的时候，不必关心数据包的大小，只需将socket看作一条数据流入库口，往里卖弄放数据就是了，TCP协议本身会将进行拥塞/流量控制

## UDP数据报报头解析

头占8字节64位

16bits:: 源端口，即发送端的端口

16bit: 目标端口，即接收端端口

16 ：udp数据包总大小 包头+数据，单位字节

16 错误检查吗，基于算法检查udp数据包是否损坏

## TCP和UDP的优点和缺点及区别

TCP优点：tcp的可靠体现在TCP在数据传输之前会有三次握手建立连接。在数据传递时，有数据校验，数据重排，响应机制，超时重发，流量控制和拥堵控制等控制机制。数据传输完毕后会断开来节约系统资源。对于故障连接，TCP有报文探测机制，及时断开连接，放置资源浪费。

TCP缺点：慢，效率低，占用系统系统资源，上述的连接过程和传输过程涉及到的数据的校验，重排，回应机制，重发机制，流量控制，阻塞控制等都会占用系统CPU。同时TCP的三次握手中的syn同位号发送会造成DDos攻击

UDP的优点比起TCP快一些。由于没有TCP那么多机制，所以UDP可攻击点就会少一些，但也存在无法避免的攻击如：UDP flood攻击。。。。

UDP的缺点：不可靠，不稳定。因为UDP没有TCP那些可靠机制，在数据传输时，如果网络质量不好，就会容易丢包。基于上门的优缺点。

应用场景：TCP应用于对通讯质量有要求的时候（比如HTTP，HTTPS，FTP等文件协议，POP、SMTP等邮件传输的协议）

UDP应用场景：QQ语音，QQ视频，TFTP。。。。有些应用场景下对可靠性要求不高会用得到。

区别：5点

（连接模式）**TCP是双向传输对等连接，UDP是无连接的**。它逻辑通信信道是全双工的可靠信道，UDP则是不可靠的信道。

（连接个数）TCP是点对点，UDP支持1对1和1对多，多对1以及多对多。

（资源消耗）**对系统资源要求不同（TCP多**）

**（数据内容）流模式和数据报模式**。UDP没有阻塞控制，因此网络出现阻塞不会使源主机发送速率降低（对实时应用很有用，如IP电话，实时音视频）但阻塞会丢失部分信息

**（传输特性正确性和实时性）TCP保证数据正确性**，无差错，不丢失，不重复，且按序到达；UDP不保证可靠交付，但是实时性较好。

## FTP协议



* FTP客户首先发起建立1个与FTP服务器端口号21之间的TCP控制连接, 指定TCP作为传输层协议
* 客户在建立的控制连接上获得身份认证
* 客户在建立的控制连接上发送命令来浏览远程主机的目录.
* 当服务器接收到1个文件传输命令时, 在服务器端口号20创建1个与客户 的TCP数据连接
* 1个文件传输后,服务器结束这个TCP数据连接.
* 之后 再次传输，服务器创建第2个TCP与客户的数据连接来传输下一个文件.

 控制连接: 带外发送控制信息（对比 HTTP 带内控制信息）

 FTP 服务器要维护用户状态信息: 当前目录, 先前的身份认证（对比HTTP的无状态连接）

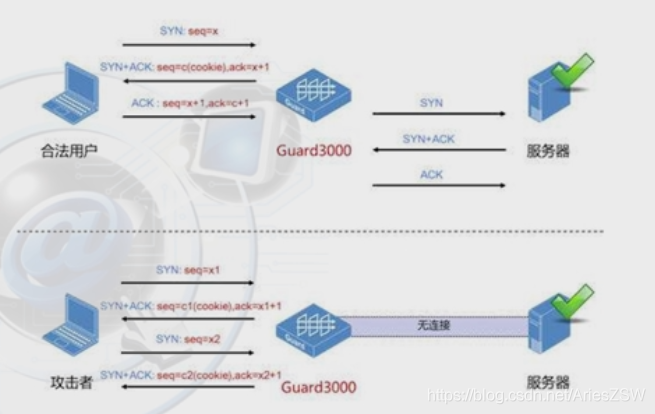
## 常见的网络攻击DOS，DDOS

传输层TCP连接中的syn攻击

形成原因：服务器队列中充满了半连接状态的连接，消耗服务器资源，直至服务器瘫痪

解决方案：

1. 降低syn的timout时间
2. 利用linux内核中syn\_cookie在收到syn请求时候生成cookie值（seq序列）放入syn报的头部，等到返回ack报中前面序列号与seq+1值对比成功后在分配数据
3. TCP可以得知数据来源，发现恶意攻击，将IP列入黑名单
4. 防护墙的带宽控制技术（限制恶意流量攻击）
5. Cdn隐藏真实地址，分散流量到不同节点



应用层的udp攻击

造成原因： 通过大量UDP小包，造成网络拥塞，使得网络瘫痪。

属于带宽攻击。

解决方法：

1. 这种攻击数据具有一定规律。毕竟是无效的数据。防火墙可以拦截此类数据。

30s后恢复正常

1. 一般都是针对某一个IP发起的，所以当对某个IP有超过数据的时候就把数据包丢弃。

## Http和https区别和作用

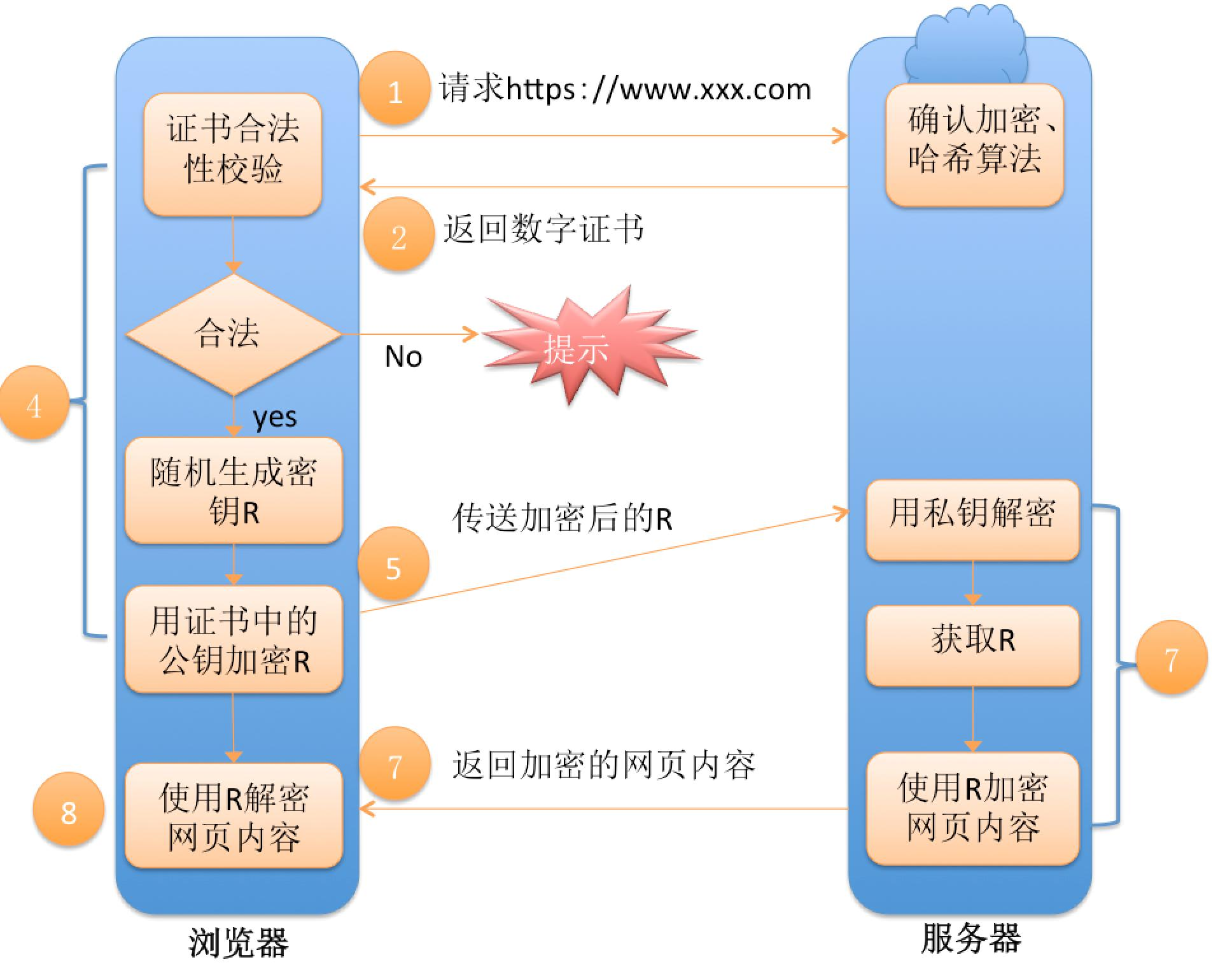
http是超文本传输协议，为了传输html文本内容。

http有md5加密（属于对称加密同一把钥匙进行加锁）到银行卡和密码等敏感信息，中途被抓包或者拦截可能出现安全问题

https=ssl+http

在建立传输之前会建立ssl连接

## SSL 加密过程



公钥，密钥，私钥

1. 浏览器发起https 请求建立SSL会话
2. http服务器确认加密，生成公钥和私钥，公钥在证书中返回给浏览器端
3. 浏览器会在证书列表确认证书合法性。如果合法就建立SSL会话反之拒绝
4. 接着浏览器随机生成密钥R，并用公钥加密发送给服务器，服务器用私钥解密获得密钥R
5. 浏览器用R加密网页内容

## http 与 https基础 内容（报文结构，报文内容）

### 请求报文



请求方式：post，get，put,delete

请求url:（连接方式（keep-alive,close），代理浏览器）；内容长度，cookie（Jsessionid=。。。。）

http协议版本

报文头

报文体：用户名，密码等信息

长短连接：close短连接，发送完内容，服务器端发起关闭连接

Keep-alive在长时间内保持连接状态

### 响应报文

HTTP/1.1 200 OK

Date: Sat, 31 Dec 2005 23:59:59 GMT

Content-Type: text/html;charset=ISO-8859-1

Content-Length: 122

＜html＞

＜head＞

＜title＞Wrox Homepage＜/title＞

＜/head＞

＜body＞

＜!-- body goes here --＞

＜/body＞

＜/html＞

* 200 - 请求成功
* 301 - 资源（网页等）被永久转移到其它URL
* 404 - 请求的资源（网页等）不存在（客户端的错误）
* 500 - 内部服务器错误（服务器端的错误）

https 优缺点与解决方法

优点：

相比较http更加安全，虽然中间人，得知加密算法，证书机构可能以中间人盗取信息。但是缩小了消息泄漏的可能，增大了泄漏成本。

缺点：

SSL证书体系并不安全，某些国际控制了CA根证书的情况下，可作为中间人窃取信息。

需要握手连接，消耗资源多，速度比http慢

## Session 和 Cookie 区别？

Session是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中

Cookie是用来保存用户信息的一种机制，同时也包含了seesionid。

有了sessionid还要使用用户和密码登入

Session是服务器端记录用户信息；

Cookie是客服端记录用户信息；

第一发起http会话后，服务器端会创建session，同时给客户端发回sessionid，客户端把sessionid记录在cookie中。通过读取cookie中的cookieid就能让服务器知道用户身份。

禁用cookie怎么办？会使用一种叫做URL重写的技术来进行会话跟踪，即每次HTTP交互，URL后面都会被附加上一个诸如 sid=xxxxx 这样的参数，服务端据此来识别用户。

## 集群怎么存 SessionId？又问禁用 Cookie 怎么存？

Session是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中

服务器端保存session的方法很多，内存，数据库，文件都有。

集群的时候也要考虑Session的转移，在大型的网站，一般会有专门的Session服务器集群，用来保存用户会话，这个时候 Session 信息都是放在内存的，使用一些缓存服务比如Memcached之类的来放 Session。

# http请求指令

PUT请求：如果两个请求相同，后一个请求会把第一个请求覆盖掉。（所以PUT用来改资源）

Post请求：后一个请求不会把第一个请求覆盖掉。（所以Post用来增资源）

**GET方法：**对这个资源的查操作。

# Post和get区别

1、GET参数通过URL传递，POST放在Request body中

2、GET请求会被浏览器主动cache，而POST不会，除非手动设置。

3、GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

4、Get 请求中有非 ASCII 字符，会在请求之前进行转码，POST不用，因为POST在Request body中，通过 MIME，也就可以传输非 ASCII 字符。

5、 一般我们在浏览器输入一个网址访问网站都是GET请求

6、HTTP的底层是TCP/IP。HTTP只是个行为准则，而TCP才是GET和POST怎么实现的基本。GET/POST都是TCP链接。GET和POST能做的事情是一样一样的。但是请求的数据量太大对浏览器和服务器都是很大负担。所以业界有了不成文规定，（大多数）浏览器通常都会限制url长度在2K个字节，而（大多数）服务器最多处理64K大小的url。

7、GET产生一个TCP数据包；POST产生两个TCP数据包。对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

8、在网络环境好的情况下，发一次包的时间和发两次包的时间差别基本可以无视。而在网络环境差的情况下，两次包的TCP在验证数据包完整性上，有非常大的优点。但并不是所有浏览器都会在POST中发送两次包，Firefox就只发送一次。

如何验证数据包的完整性？

所有接口接收jason格式数据，将字段根据ascii码从低到高排序，非ascii进行urlencode编码，字段之间&&连接，在字符串末尾加上key。全部转为大写然后MD5加密。

最终这个字符串就是签名

Md5是不可逆的加密，但是黑客可以利用**彩虹表**进行破解

# 大文件断点续传实现过程

将大文件文件分成小包，每个包大概1M左右；

客户端和服务器端都记录已完成传输的range

当网络在此连接的时候，回到刚才传输的位置继续。

如何解决丢包和包错误问题

# MD5

无论是网站或者软件有MD5是为了保证文件的正确性，防止一些人盗用程序，加些木马或者篡改版权，设计的一套验证系统。**每个文件**都可以用**MD5验证程序**算出一个固定的MD5码来。

**具体例子**: 在UNIX下有很多软件在下载的时候都有一个文件名相同，文件扩展名为.md5的文件，在这个文件中通常只有一行文本，大致结构如：  
MD5 (tanajiya.tar.gz) = 0ca175b9c0f726a831d895e269332461

我组产品包含大量音频和图片资源，MD5主要就用来检测这些**资源文件的完整性**。

**主要思路**：先计算出**所有资源文件的MD5值**，存到一个**xml**文件中，作为标准的MD5值。然后把这个xml文件放到我们的产品中，每次更新程序以后，算出资源文件的MD5值，同样存到xml文件中。对比两个xml文件，并打印出发生变化的文件名。

## Md5加密的过程

1. 去掉appid，去吃data中空和无用的数据
2. 将数据转换成hex（16进制）对数据进行排序
3. 增加appid对数据进行md5加密生成签名sgn（追加在数据的最后一行）

## Md5加密签名的应用场景

Md5是不可逆加密算法；通过同样算法比较签名-🡪确认数据的一致

1. 银行卡密码通过md5加密后保存在密码段
2. 版权保护
3. 文件完整性
4. 是否由木马