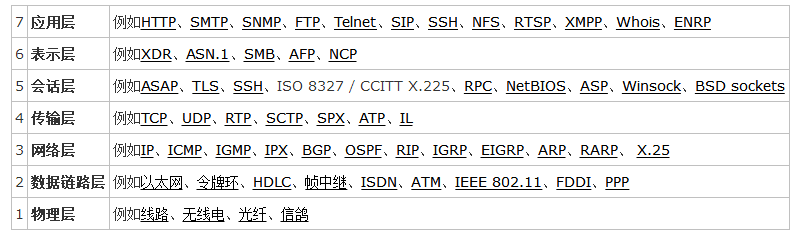
Linux基础

OSI



## 局域网类型

以太网

**总线型网络**

以太网技术美国施乐公司无偿捐献

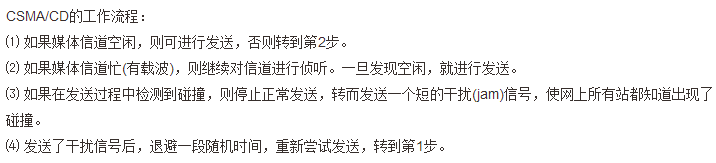
**mac** **标识一台物理设备在网络中的身份** 地址唯一

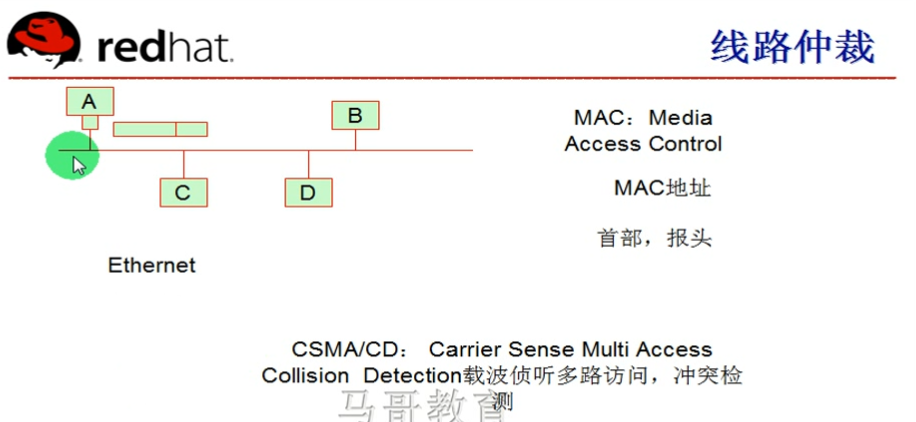
为什么要线路仲裁?

**总线型网络中线路仲裁机制CSMA/CD**

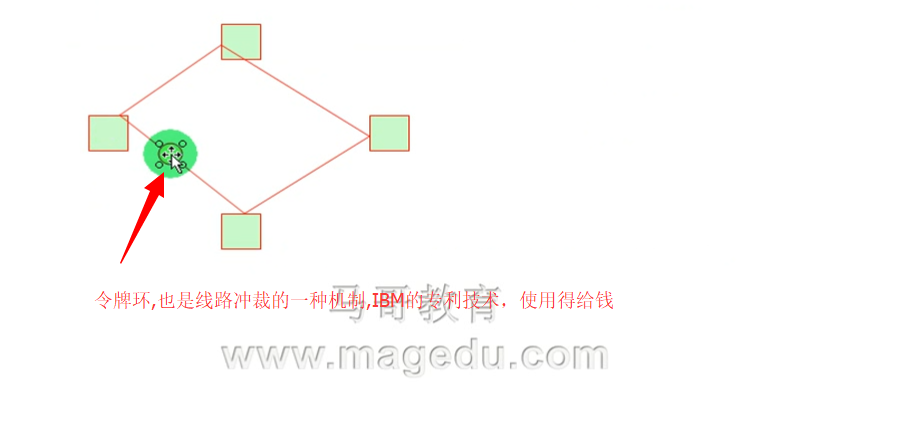
**在同一时刻只能有一个设备发送数据,一个设备接收数据**

**CSMA/CD 载波侦听多路访问冲突检测机制**

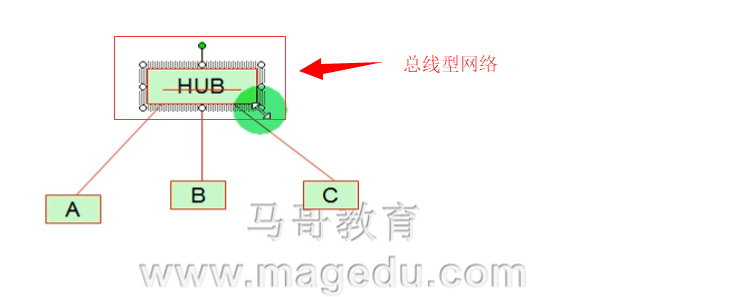




**环形网络**

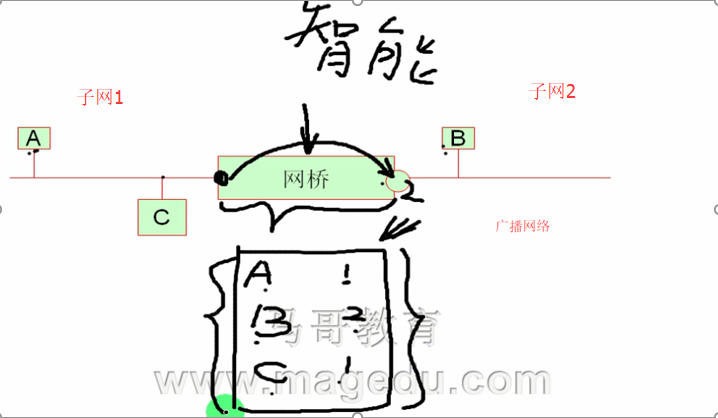


**使用hub集线器是总线型网络**



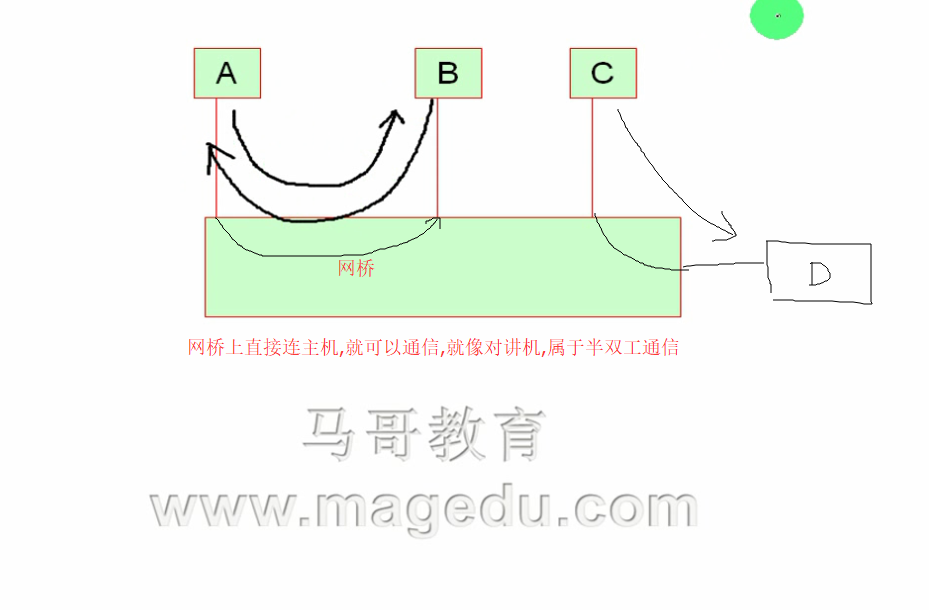
## 网桥

网桥中维护的表的数据是设备广播自己学习而得到的,也可以手动填进去

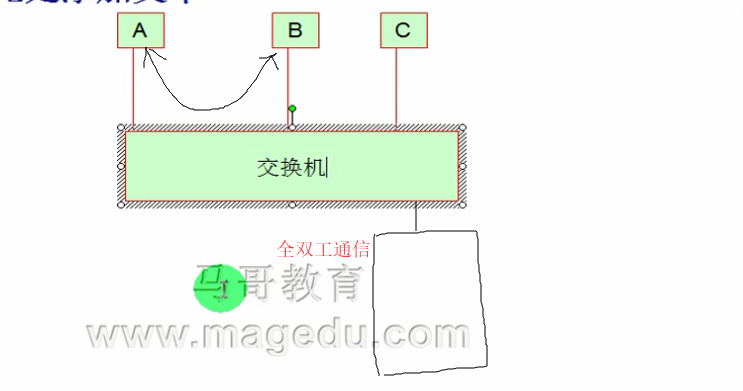


把子网1和子网2隔离,子网1和子网2各自管理自己的冲突,降低了全局冲突,传输效率提高

网桥上只接主机

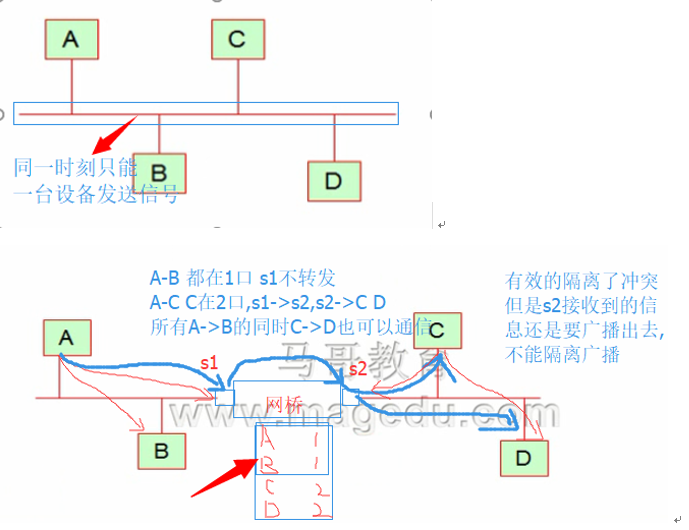


双绞线: 搅在一起抗干扰,只有2根有用,全双工通信的线缆,就可以让主机工作在全双工模式下, A发B收 B发A收



A和B C 通信,首先要广播喊一嗓子,如果主机很多,就容易形成**广播风暴**,

交换机隔离的是冲突,不能隔离广播



A-B

1. 广播, 发送方MAC地址为FFFFFF ,这是一个广播物理地址,任何设备都可以收到,网桥也要转发,这个网络中的所有设备都知道了A的MAC地址

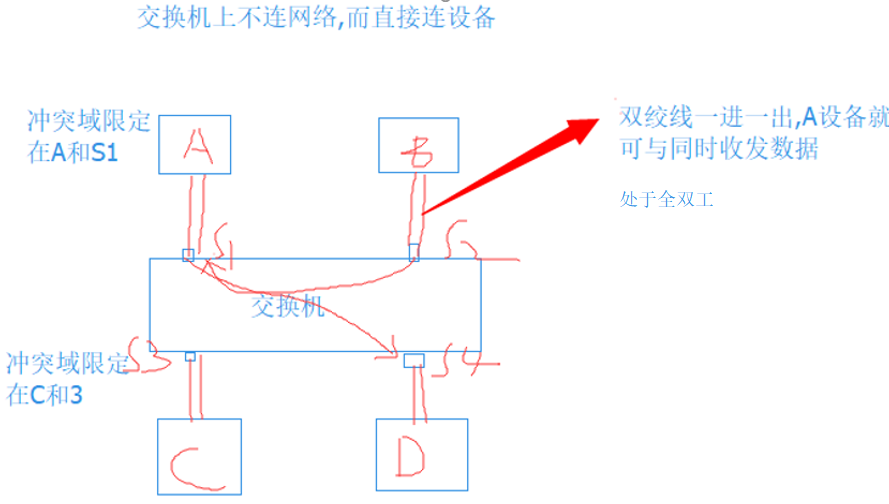
2. A通过广播,网络中设备回应一声,得知B的MAC地址,A-B 就可以了

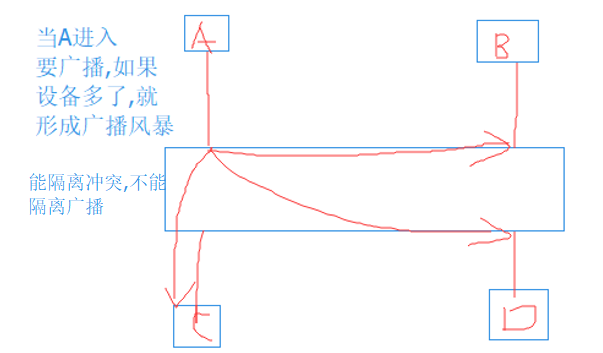
3. A-B,网桥得知他们都在1,就不转发,

4.A-C 那么 C-D 就不能通信,这时S2和C D 是冲突的

## 交换机

**多个口的网桥就是交换机**



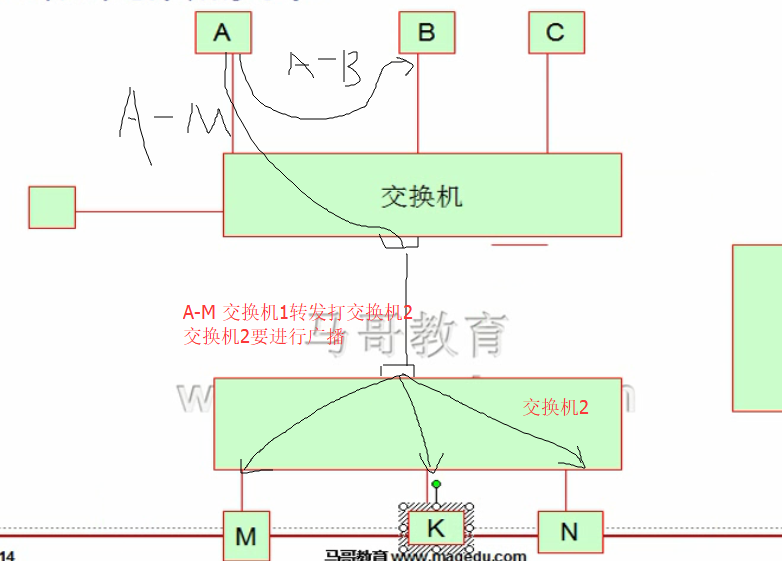


必须要隔离广播,不然广播信号把带宽占了,通信就会受到影响

子网掩码识别是不是本地网络

就像上面控制冲突一样,进行子网切割

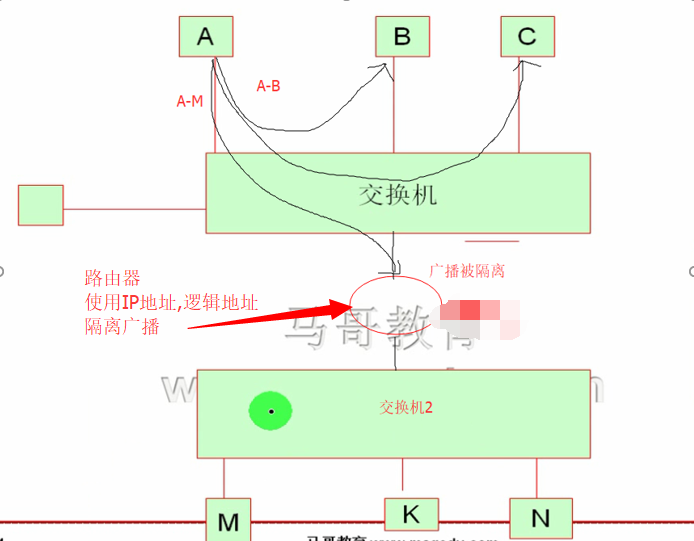
平面地址 : 物理地址就是MAC地址,广播网络中的MAC地址,交换机内两个地址通信,就依赖于MAC地址

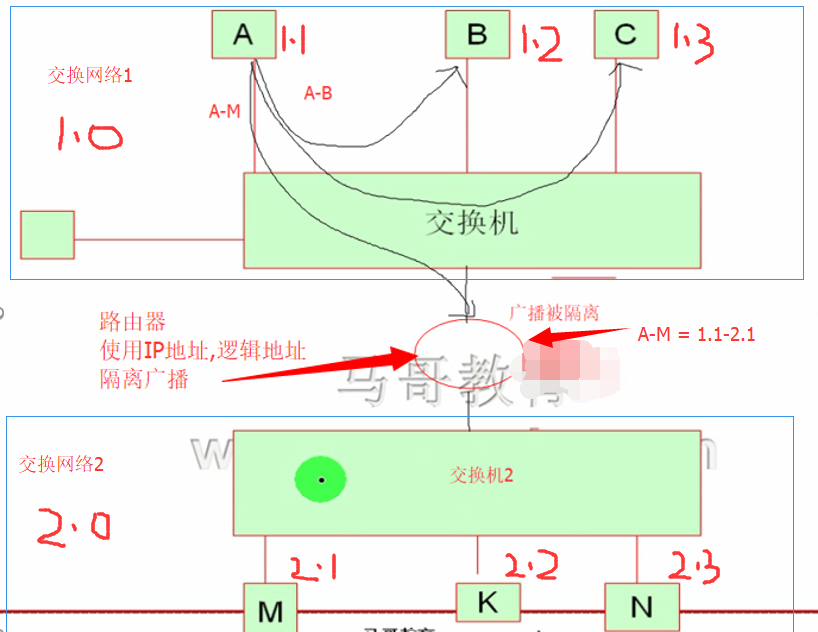


## 路由器

不让交换机2进行广播

引入逻辑地址: IP地址,路由器就是标记不同的交换网络.交换网络1,交换网络2





一个交换机可以连接多个不同的冲突域

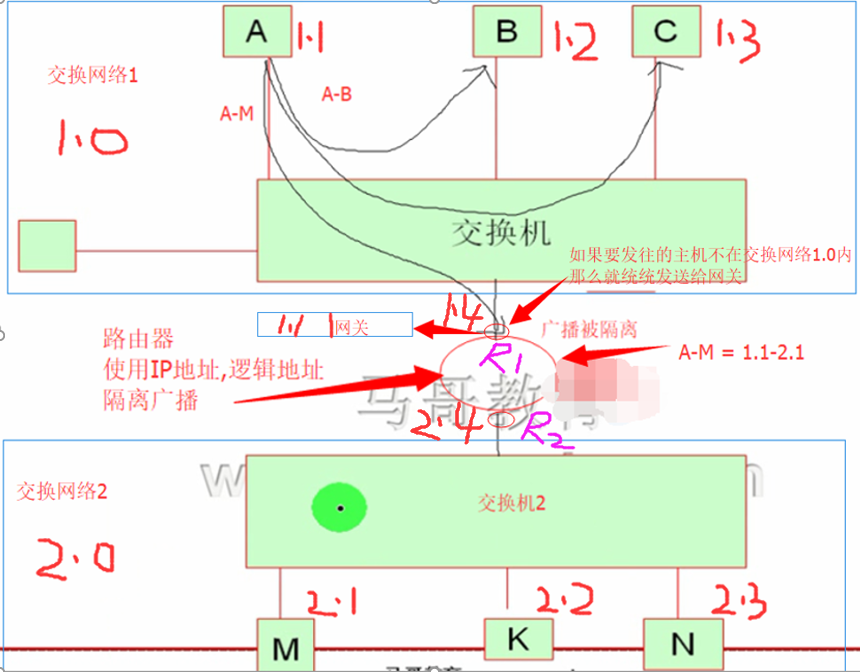
一个路由器可以连接多个不同的广播域

网关 默认网关 网关也有MAC地址

1.0 和 2.0 两个网络 就靠逻辑地址(IP地址)来区分的

**1.4 和 2.4 就是网关**

R路由器中也维护了一张路由表,标记了网络和它映射的子网,数据报文从哪个口进行转发



1.1 - 2.1 报文到达R1 , 2.1在2.0于是转给R2. R1 R2 也有自己的MAC地址

**报文就分为源IP,目标IP,源MAC,目标MAC**

**MAC的基础上在封装一层IP逻辑地址**

如果是在本地网络中通信

A主机--B主机(1.1--1.2),首先是A主机要广播(目标MAC是FFFFFF)**,**于是1.0网络内部的主机都会收到这个广播,问有没有1.2,于是1.2就回复报文,这个报文把自己的MAC地址告1.1主机,就可以通信了

如果是在不同子网中通信

* 1. -- 2.1 ,首先是把发往的IP地址与子网掩码像技术,得到不是本网络,于是就把报文发给网

关,网关接收到数据以后根据路由表查找端口对应到2.0网络是什么端口,这里是R2端

口,于是将报文转给R2端口,R2广播得到2.1的MAC地址,于是将报文转给2.1,

广播的目的就是通过逻辑地址找MAC地址,这个过程就是ARP地址解析,反过来就是RARP

路由器有**自动学习**的能力,并且向外广播 我能到达xxx网络,当与之相连的路由器就把这条记录保存在自己的路由表中

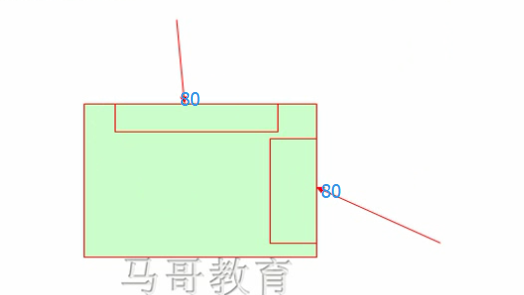
## IP地址

ip 标识了主机到主机的通信,网络到网络的通信

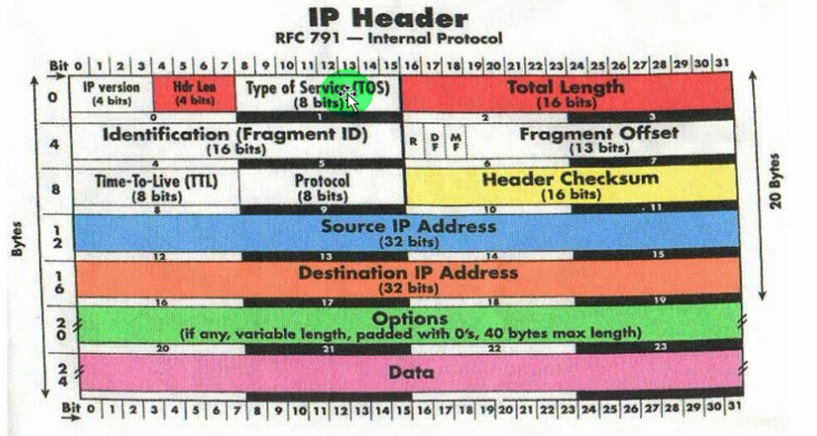
端口标识了进程到进程的通信 : 0 - 65535

国际上规定使用80端口来进行web服务

**一个端口只能有一个进程监听和使用**

如果一个主机有有两块网卡,那么就有两个IP地址,每个IP地址都有65535个端口可用

**socket = IP + port**



IP地址: 点分十进制

0000 0000 - 1111 1111 0-255

221.32.221.2.2

IP地址: 网络地址 + 本网络的主机地址

**算网络地址使用子网掩码与IP地址进行与运算**

**A类**

用头8位标识网络地址 子网掩码 255.0.0.0

网络地址: **0** 000 0001 - **0** 111 1111

127个A类,127用于回环,

共2^7-1 = 126

1-126

主机地址全0 标识网络地址

主机地址全1 标识广播地址

容纳的主机: 2^24 - 2个

**B类**

用头16位标识网络地址 子网掩码 255.255.0.0

网络地址: **10** 00 0000 - **10** 11 1111

共 2^14个B类网络

128-191

主机地址全0 标识网络地址

主机地址全1 标识广播地址

容纳的主机: 2^16 - 2个

**C类**

用24位标识网络地址 子网掩码 255.255.255.0

网络地址: **110** 0 000 - **110** 1 1111

共2^21个C类网络

192-223

主机地址全0 标识网络地址

主机地址全1 标识广播地址

容纳的主机: 2^8 - 2 = 254个

**D类**

网络地址: **1110** 0000 - **1110** 1111

224-239

**E类**

**ICANN 国际IP地址分配机构来统一分配**

私有地址,公司内部可以使用的,不能拿到互联网上使用

A类: 10.0.0.0/ 8个

B类: 172.16.0.0/16-172.31.0.0/16

C类:192.168.0.0/24-192.168.255.255/24 24标识掩码长度

**这些私有地址不会在网络上路由,只能内部使用**

**目标地址是一个主机的叫做主机路由**

**目标地址是一个网络的叫做网络路由**

如果路由表中没有条目,就走默认路由 0.0.0.0

子网划分,就是像主机地址借位来用着网络地址,但是划分子网要浪费一些主机地址,被用作网络地址和广播地址



## TCP和UDP

**TCP**

有链接的协议, **可靠**的传输协议先3次握手建立链接(是一个双向通道),效率上没有UDP高

**UDP**

无链接的协议, **不可靠**的传输协议 不用建立链接,扔出去就不管了,有的报文丢了也不会管,

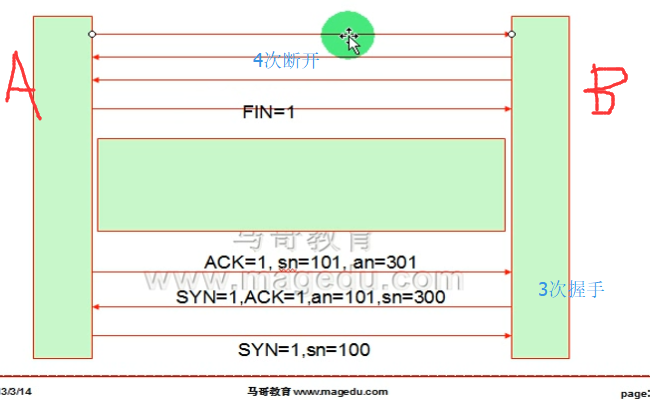
因此不可靠,但是效率高,

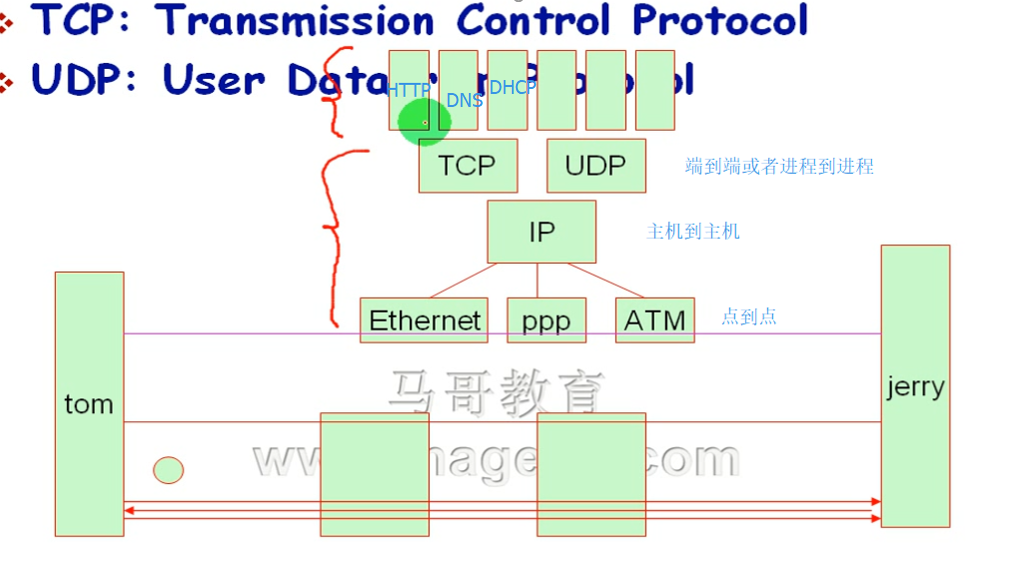
即时通讯就是用UDP,速度快,对于要求服务质量高的需要使用TCP

http,https,ftp,smtp…这些应用层协议都是用的TCP

DNS查询, DHCP ,TFTP应用层协议用的UDP







两块网卡绑定在一起使用,内核就当作一个网卡来用,一块坏了,不影响,两块性能好一倍,有点像RAID

## 网络配置

IP

NETMASK

GATWAY

HOSTNAME

DNS1 响应最快的些在最前面

DNS2

DNS3

手动指定

DHCP Dynamic Host configuration protocol 动态主机配置协议, DHCP Server 提供服务,如果没有服务停止了,咋办

169.254就起作用了,这是一个免费使用的,没有网关.

路由

网络是属于内核的功能,IP地址无论配置在主机上的哪块网卡上,只要通过IP地址有访问进入,主机都会响应,因为网络是内核的功能.

网络设备

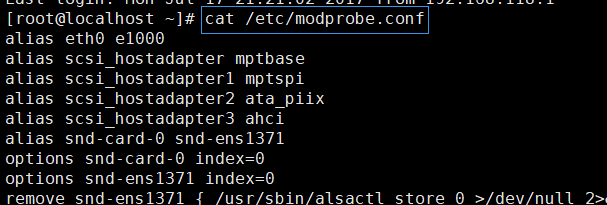
**lo** (loopback ):本地回环

**eth[0-9]** :以太网卡

**pppx** : 点对点

redhat5 对网卡命名

/etc/modprobe.conf 中命名



Redhat6

在/etc/udev/rules.d/70-persistent-net.rules 中对网卡命名

以后在引用网卡设备就使用命名就可以找到

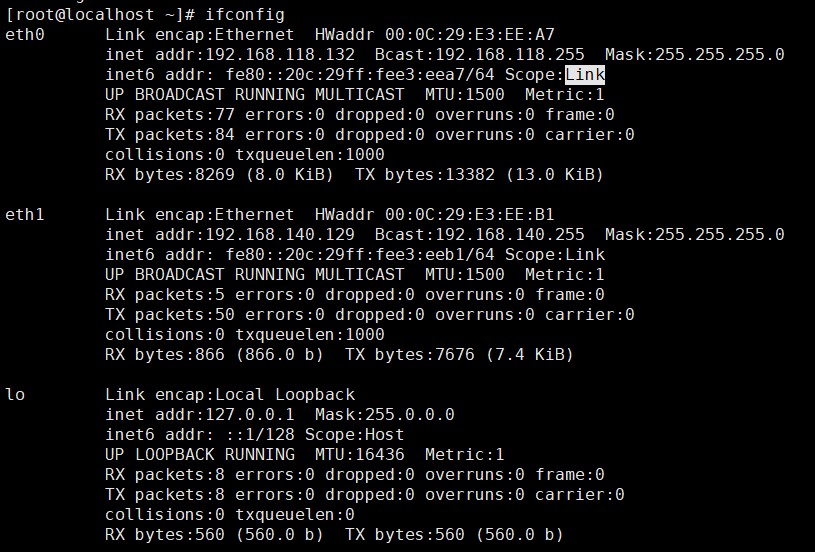
## ifconfig

ifconfig

-a 显示所有网络接口的配置信息

ethx 显示某一个网络接口的配置信息

ifconfig

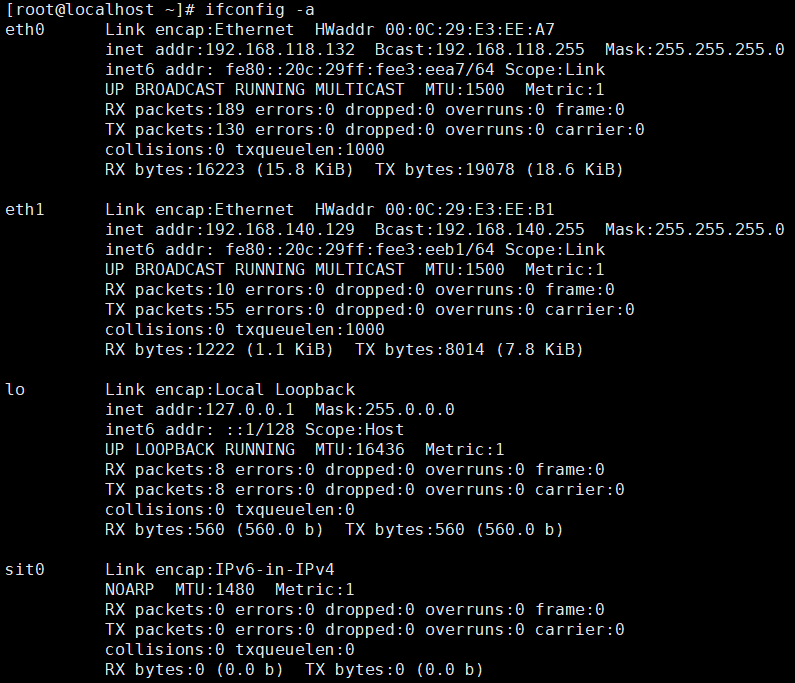


* 启用  多播  正在运行 多播

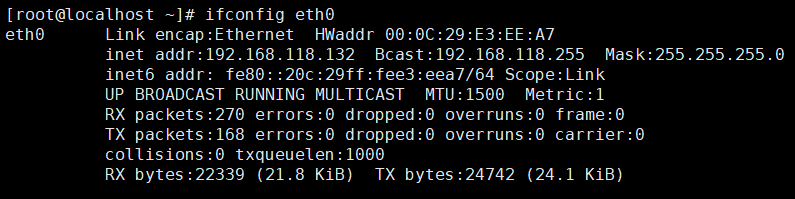
 最大传输单元



Ifconfig –a

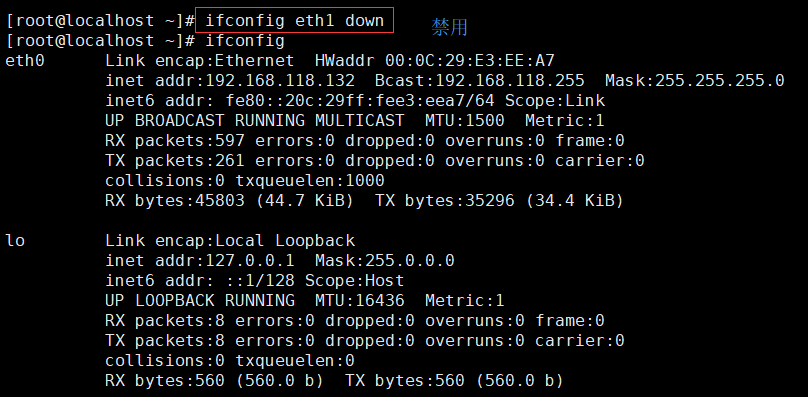


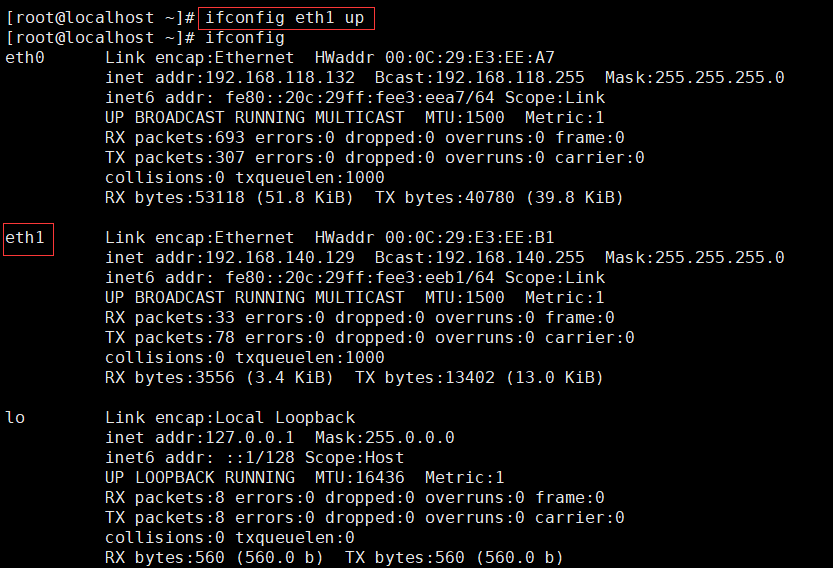
ifconfig eth0



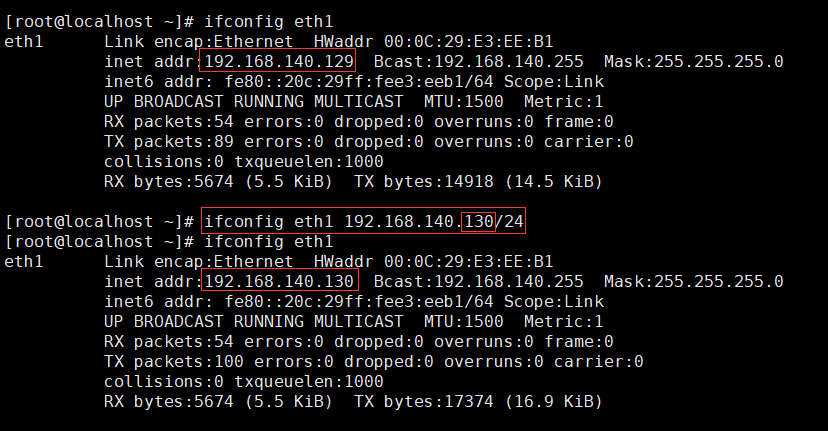
ifconfig ethx down/up

停用/启用 ethx4

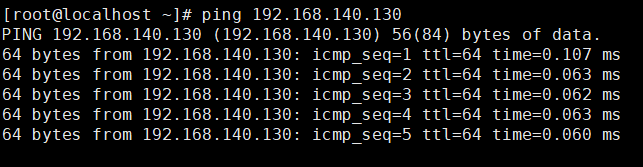




**ifconfig ethx ip/mask [up|down]** 手动可以配置IP地址



手动配置的会立即生效,但是**重启网络服务或者主机**后就不起作用了



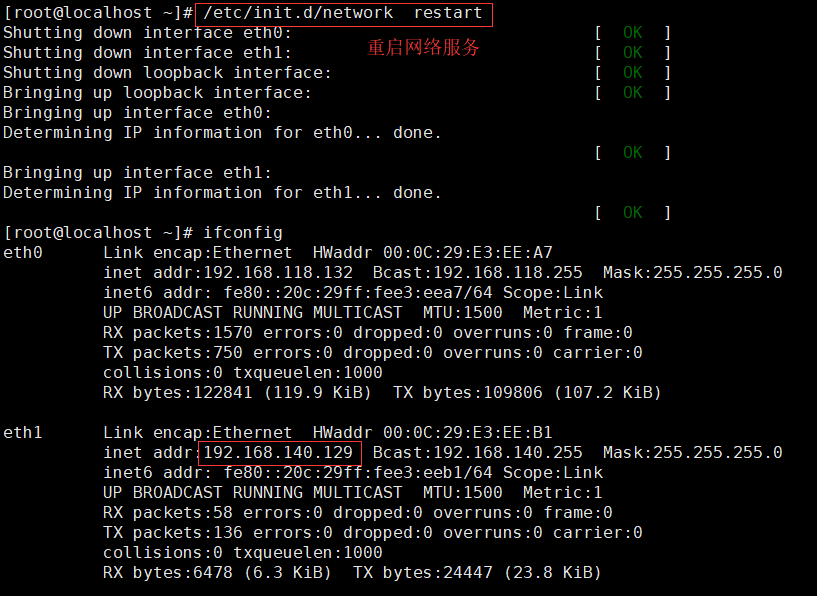
**网络服务**

Redhat5 网络服务脚本

/etc/init.d/network {start|stop|restart|status}

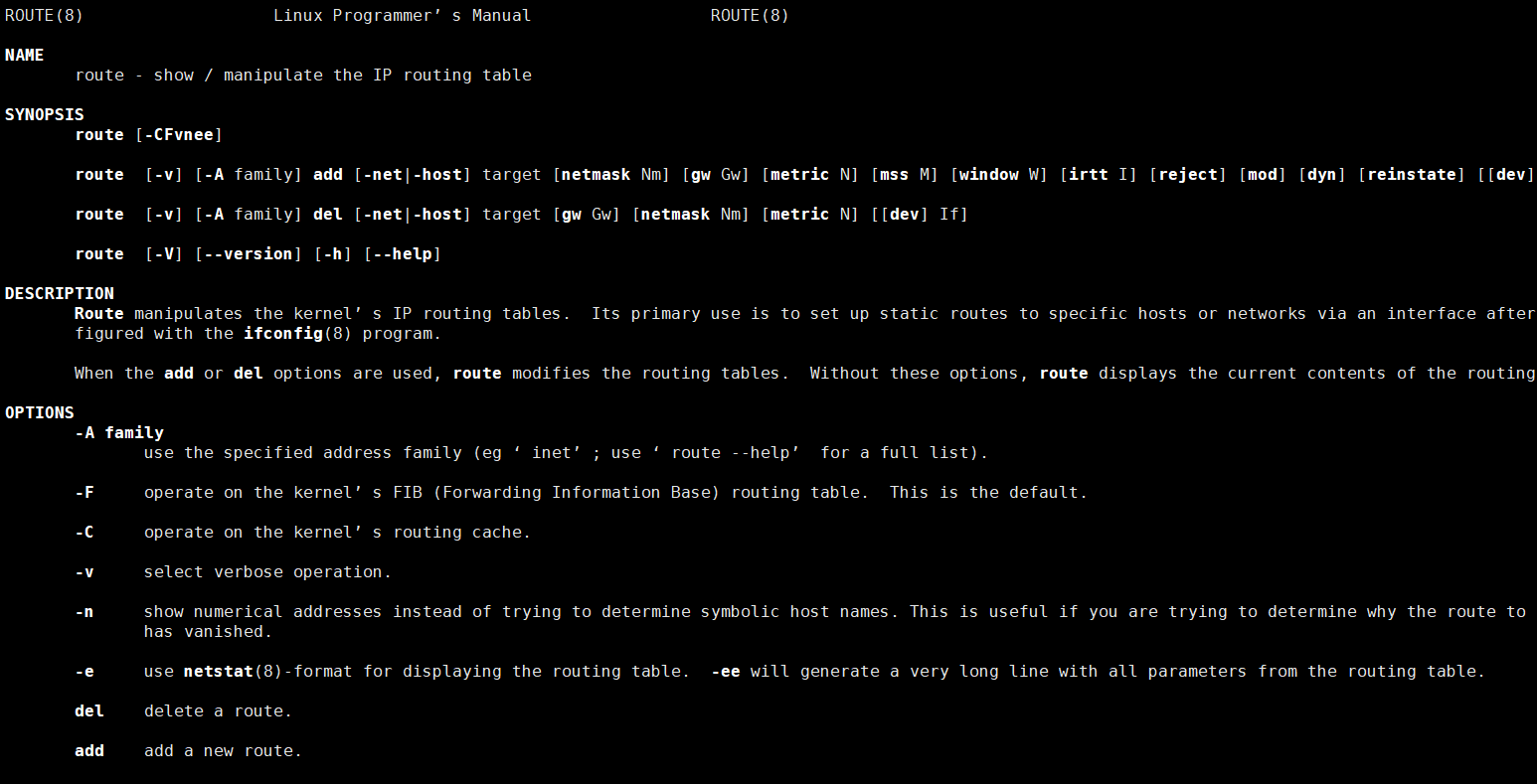
Redhat6 网络服务脚本

/etc/init.d/NetworkManager { start|stop|restart|status } 不好使,建议禁止掉,启用redhat5的网络服务脚本功能

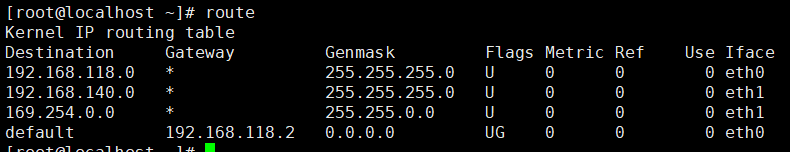


## 网关

route



查看本机路由表



route

add 添加

-host : 主机路由

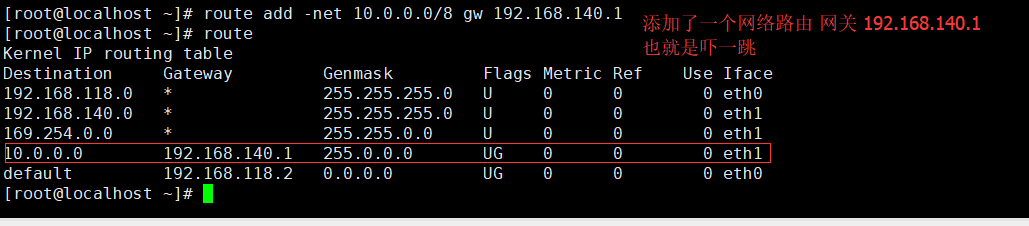
-net : 网络路由

del 删除

-host : 主机路由

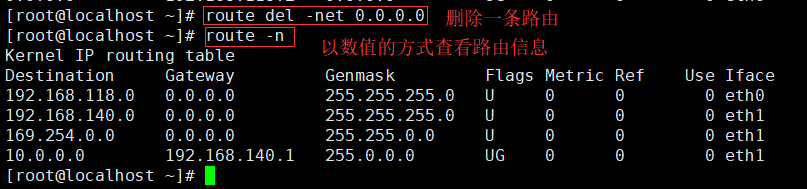
-net : 网络路由

就是通过192.168.140.1这个网关可以达到下一跳10.0.0.0这个网络

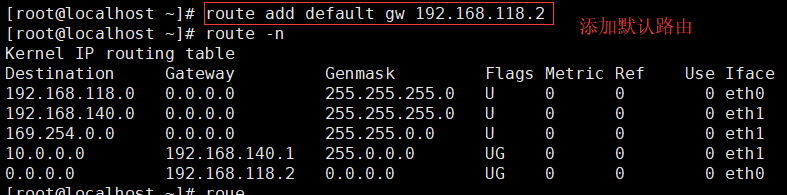




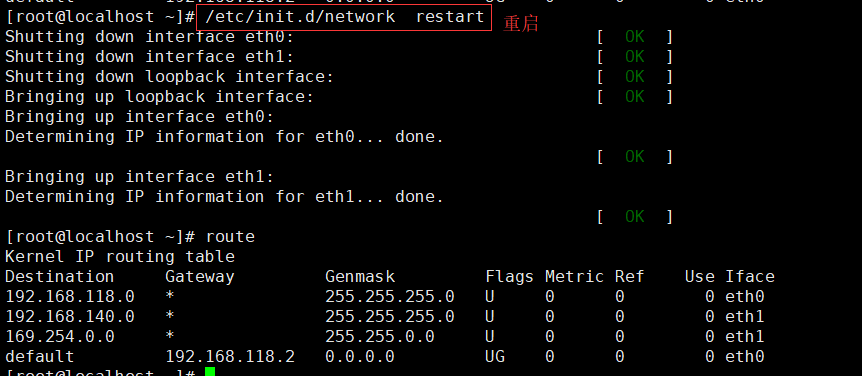
网关路由有下一跳路由,本地网络就没有网关不需要网关就可以到达的







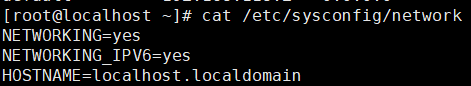
**以上改动重启网络服务或者主机后就失效了**



## 配置文件

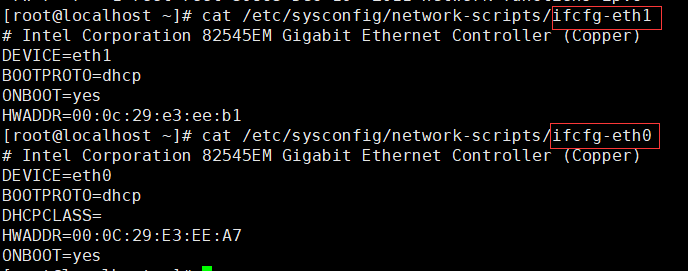
**网络配置文件**

/etc/sysconfig/network



**网络接口配置文件**

/etc/sysconfig/network-scripts/ifcfg-eth#



DEVICE=eth1 关联的设备名称要与文件名的后半部ifcfg-eth1 保存一致

BOOTPROTO={static|none|dhcp|bootp} 引导协议,如果使用静态地址配置static或者none,

dhcp表示使用DHCP服务器获取地址

IPADDR=: IP地址

NETMASK=: 子网掩码

GATEWAY=: 设定默认网关

ONBOOT=: 开机时是否自动激活此网络接口

HWADDR=: 硬件地址,要与硬件中的地址保存一致,可省;

USERCTL={yes|no} 是否允许普通用户控制此接口

PEERDNS={yes|no} 是否在BOOTPROTO=dhcp时,接受由DHCP服务器指定的NDS地址

这样配置的地址不会立即生效,但是重启网络设备或者主机的时候会生效.

**路由配置文件**

在/etc/sysconfig/network-scripts/中为**eth#** 添加route-eth# 文件名

添加格式1

/etc/sysconfig/network-scripts/route-eth#

dest via nexthop

添加格式2

ADDRESS0=

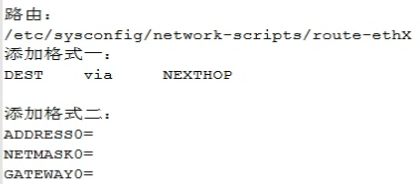
NETMASK0=

GATEWAY0=

ADDRESS1=

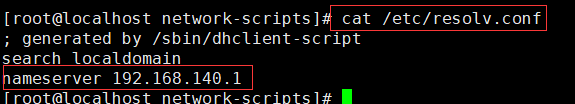
NETMASK1=

GATEWAY1=



**DNS服务配置文件**

DNS服务器指定的方法只有一种



nameserver DNSIP1

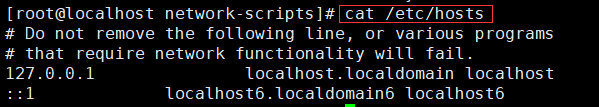
nameserver DNSIP2

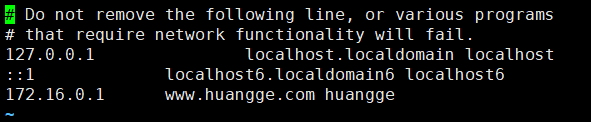
nameserver DNSIP3

最多可以有3个

**指定本地解析配置文件**

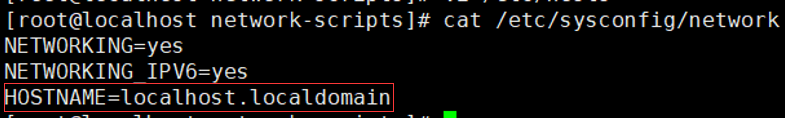
/etc/hosts

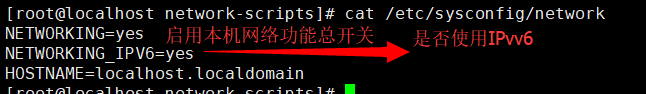




**主机名配置文件**

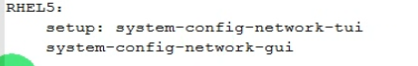
hostname xxx 立即生效,重启网络或者主机失效





也可一定义网关,这是全局的,如果在网络接口定了网关,以网络接口中配置的网关.全局服从局部

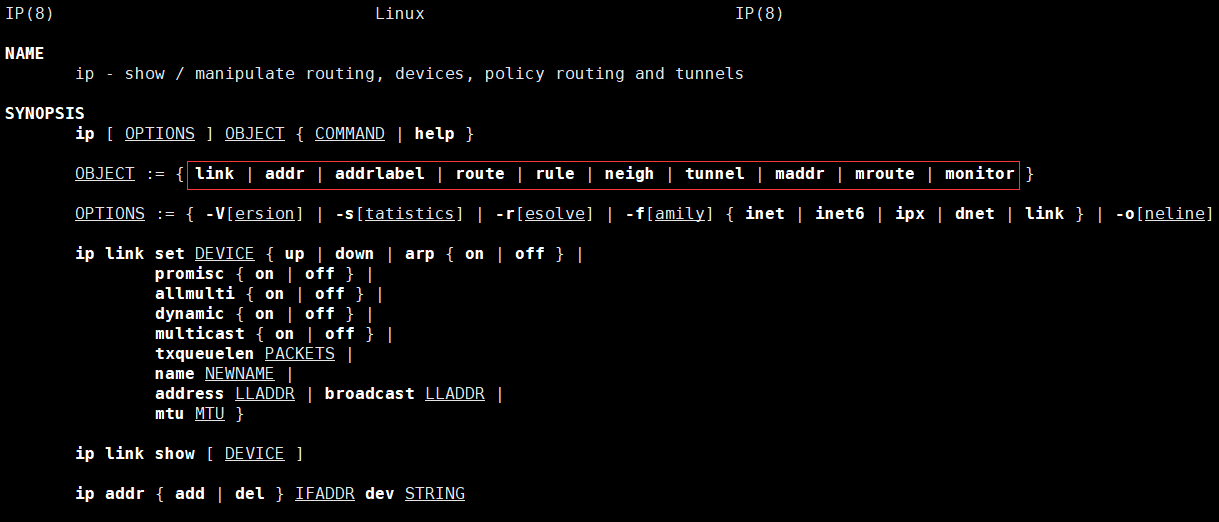
图形化界面配置



ifconifg 老旧

iproute2包 功能非常强大

**ip**



ip

link: 网络接口属性

addr: 协议地址

route: 路由

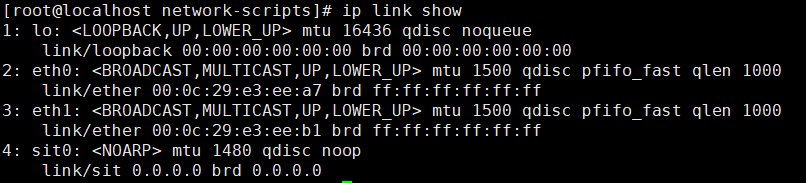
**link**

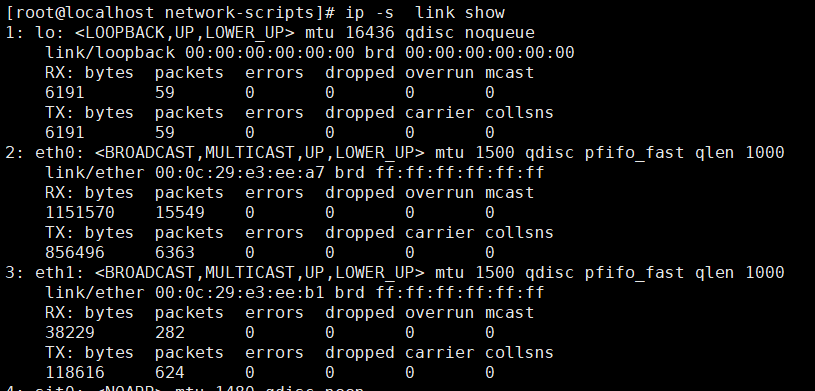
**show**

ip –s link show

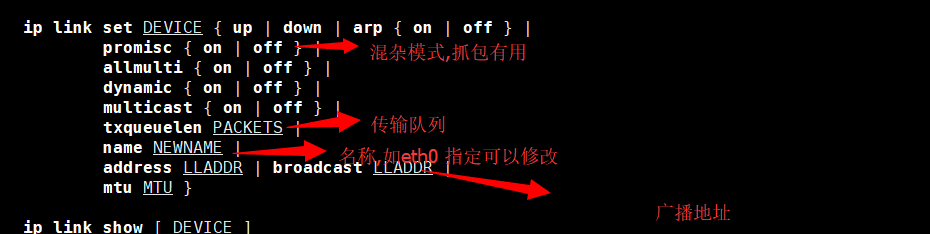
**set**

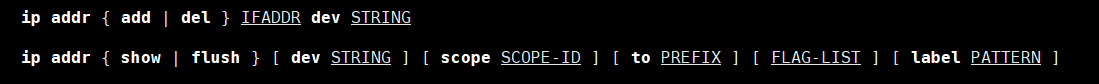
ip link set DEV{up|down}





ip link set





一块网卡可以使用多个地址:

网络设备可以有别名

eth#:#: eth0:0 eth1:1

配置方法:

1. ifconfig eth#:# IP/MASK 立即生效,重启网络或者主机失效
2. /etc/sysconfig/network-script/ifcfg-eth#:#

DEVICE=eth#:#

非主要地址不能使用DHCP动态获取

## addr

**add**

ip addr add 10.2.2.2/8 dev eth1

ip addr add 10.3.3.3/8 dev eth1 label eth1:1

**del**

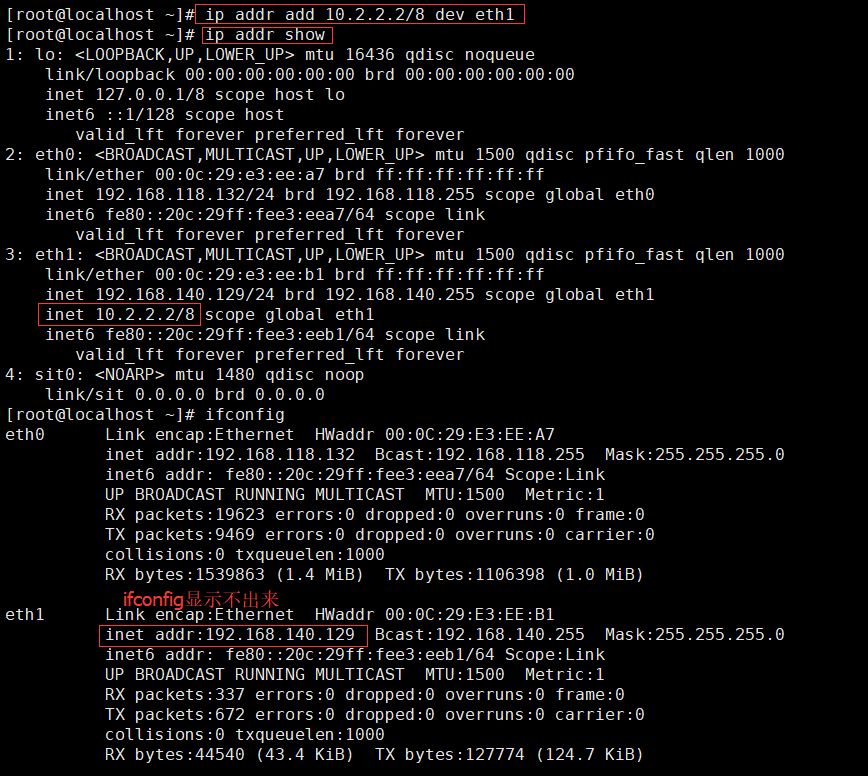
ip addr del 10.3.3.3/8 dev eth1

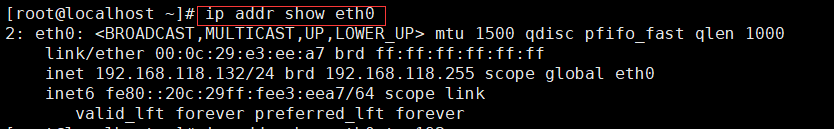
**show**

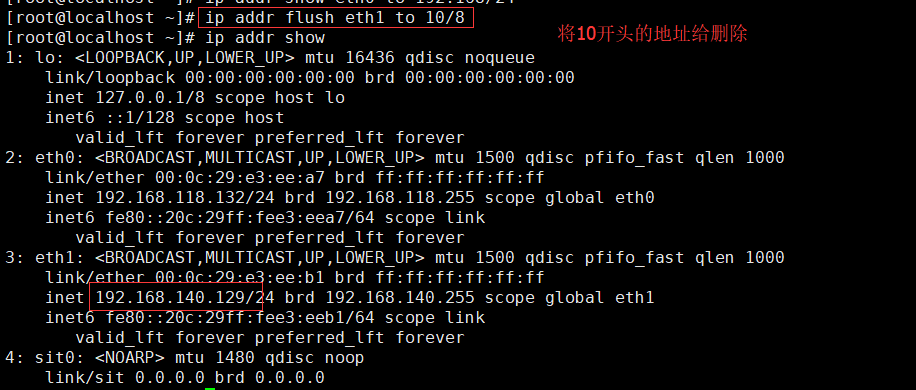
ip addr show

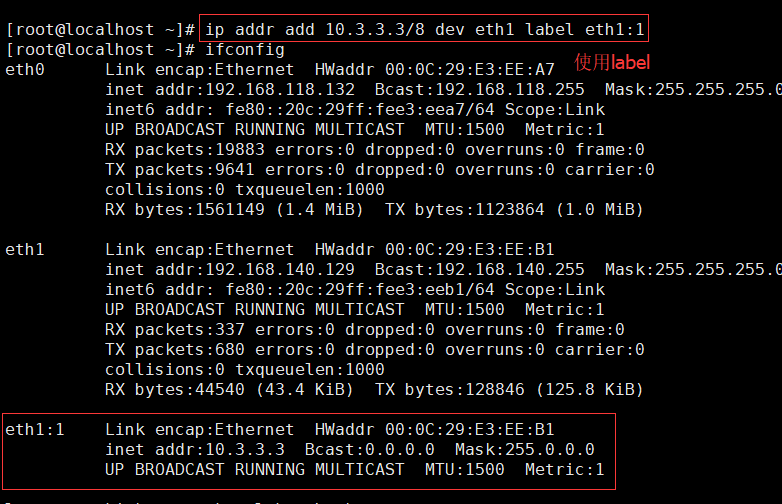
**flush**

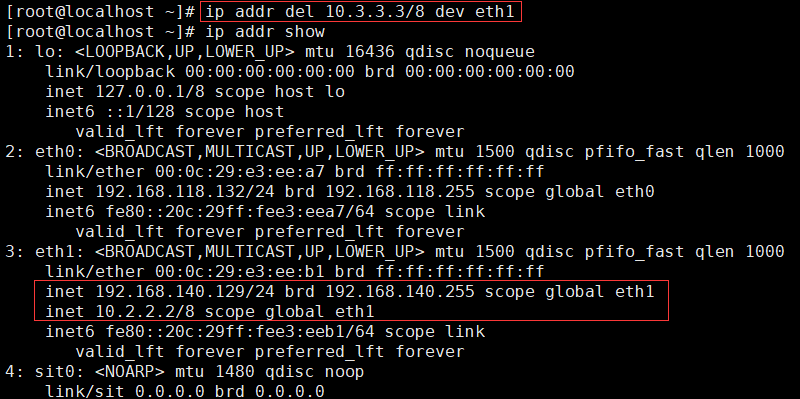
ip addr flush eth1 to 10/8

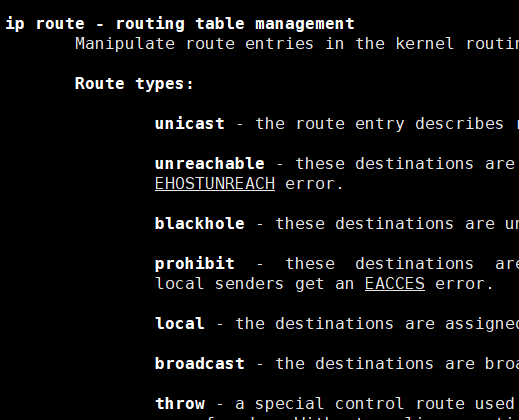


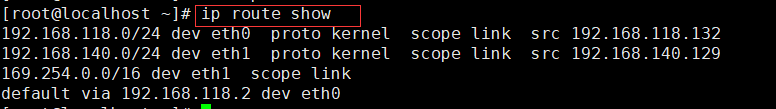














## netstat

-r 显示路由表信息

-n 以数字的方式显示路由表信息

-t 显示建立的tcp链接

-u 显示UDP链接,

-l 显示监听状态的连接

-p 显示监听指定的套接字的进程的进程以及进程号

