

GNU/Linux-Route



GNU/Linux-Route

一：构架 Linux 下的 Router



GNU/Linux-Route

多少年来，路由器的发展有起有伏。90年代中期，传统路由器成为制约因特网发展的瓶颈。ATM交换机取而代之，成为IP骨干网的核心，路由器变成了配角。进入90年代末期，Internet规模进一步扩大，流量每半年翻一番，ATM网又成为瓶颈，路由器东山再起，Gbps路由交换机在1997年面世后，人们又开始以Gbps路由交换机取代ATM交换机，架构以路由器为核心的骨干网。

GNU/Linux-Route

路由器优点：

1. 比较适用于较大和大规模的网络
2. 为复杂的网络拓扑结构提供负载共用和提供在数据传输时最佳路径
3. 能更好的处理各种数据
4. 在路由器上可以设置相关的安全访问控制，提高网络安全性



GNU/Linux-Route

路由器优点：

- 5. 隔离不需要的网络通信量。（比如广播等）
- 6. 提高网络的可用率（也就是说将超网化为子网）
- 7. 减少主机承载响应的负担



GNU/Linux-Route

路由器缺点：

1. 按照不同等级（不同用途），路由器的价格也会提升很高，对于 TCO 预算不是很好（除软路由）
2. 安装比较复杂
3. 不支持非路由协议（也就是不可路由的协议）



GNU/Linux-Route

1. 路由：路由实际上指的就是有关数据在传输时，数据到达目的所需要的经过的路径，这就是我们所谓路由。

2. 路由器：

路由器是一种连接多个网络或网段的网络设备，它能将不同网络或网段之间的数据信息进行“翻译”，以使它们能够相互“读”懂对方的数据，从而构成一个更大的网络。

路由器就是一种可以把超网划分成为子网的一种设备，同时还可以让任何分割的子网中的计算机都可以与另一个子网络相连接。

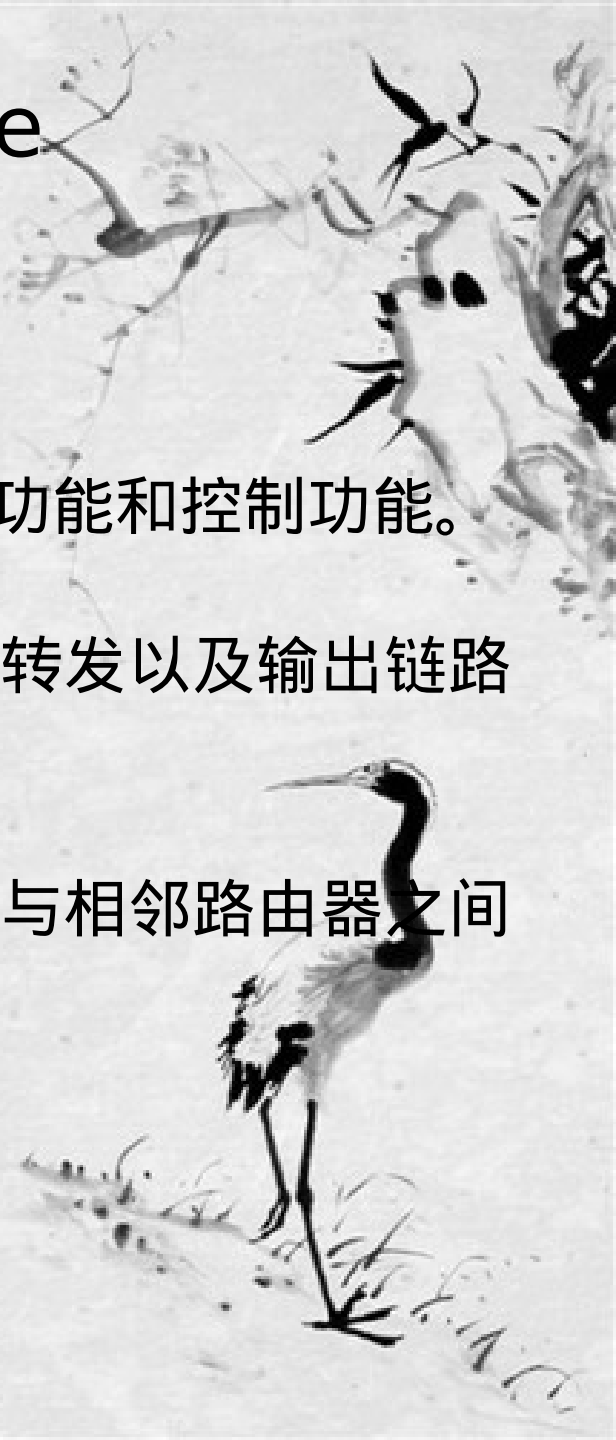
GNU/Linux-Route

3. 路由器的功能：

路由器有两大典型功能：即数据通道功能和控制功能。

1. 数据通道功能包括转发决定、背板转发以及输出链路调度等，一般由特定的硬件来完成；

2. 控制功能一般用软件来实现，包括与相邻路由器之间的信息交换、系统配置、系统管理等。

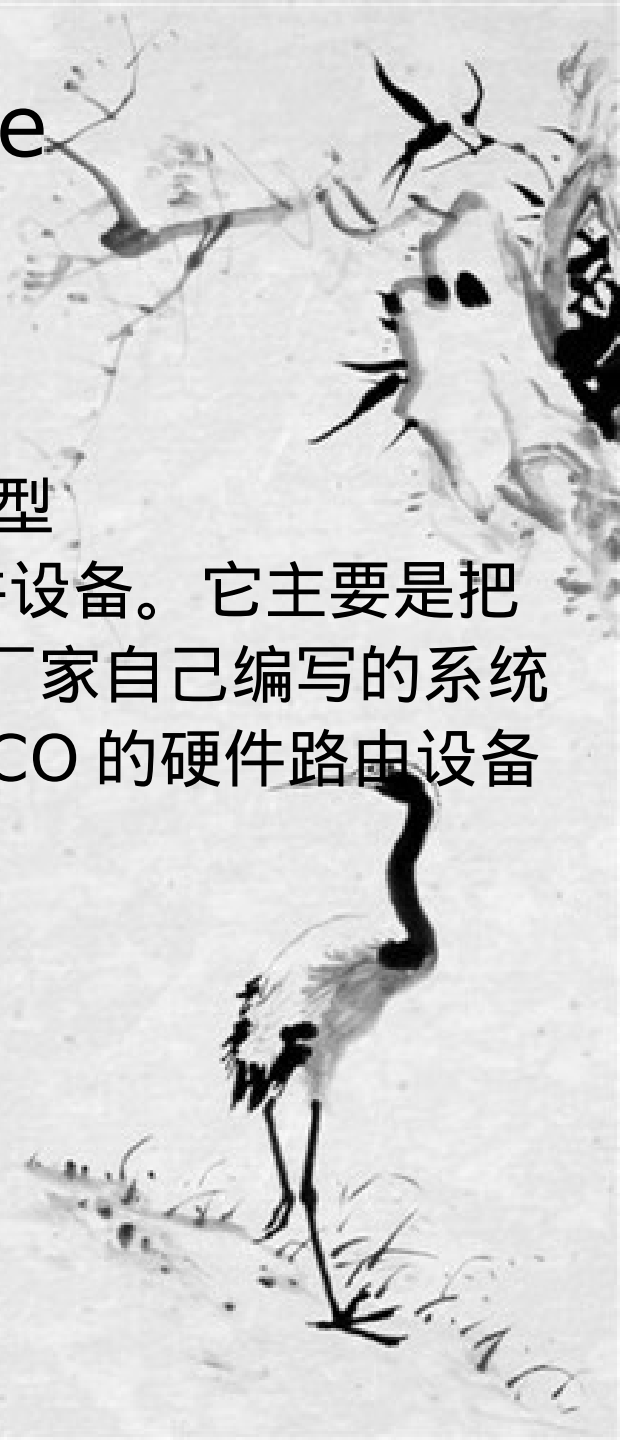


GNU/Linux-Route

4. 路由器的类型：

按照路由器的实体，路由器实际上有两种类型

1) 硬件路由器：这是一种专门的硬件设备。它主要是把——“路由”这一功能作为主要的功能通过由厂家自己编写的系统来对这种功能精心设计和优化的。比如 CISCO 的硬件路由设备及其 CISCO 的 IOS



GNU/Linux-Route

2) 软件路由器：

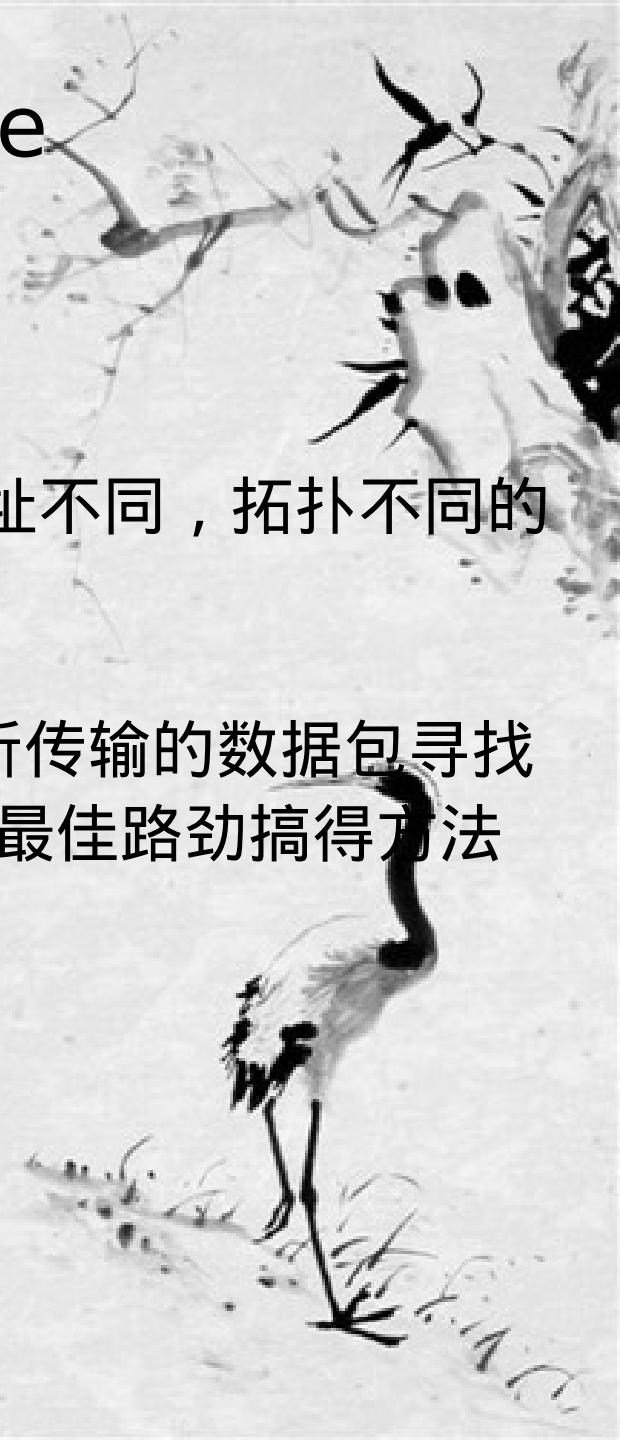
这种路由器主要是基于在某中 OS 中，这种路由只能是一种服务，但是在许多的地方不可能与硬件路由器相比。软件路由器一般都是用在小型网络中。如果是比较大型的网络就应该使用硬件路由了。软件路由中有：基于 Linux 的，有基于 UNIX 的，有基于 Windows 的。但是如果你打算使用动态路由的话，也可以对于 Windows 来说本身就可以支持。但是效果不是很好。而对于 Linux 和 UNIX 来说 Quagga 是一个不错的选择。而且他本身就可以与硬件路由器进行相互合作。功能十分强大。

GNU/Linux-Route

ROUTER 原理：

1. 一般 router 是用于将很多逻辑地址不同，拓扑不同的网络连接在一起

2.Router 的主要工作就是能够为要所传输的数据包寻找出最佳路径，保证数据包的快速到达。找到最佳路劲搞得方法可以是某种算法或管理员手工指定



GNU/Linux-Route



路由表实际上可以分为两类

1. 静态路由表：静态路由表是由系统管理员事先设置好的，然后在将设置好的路径保存在路由表中。静态路由表在网络拓扑发生变化的时候是不会被改变的。

2. 动态路由表：动态路由表是依靠相关的路由选择协议和现有的网络拓扑结构来自动学习和记忆网络的拓扑从而计算出最佳等各种网络传输路径。并且把这些自行的保存在路由表中。随着网络拓扑或路由选择协议发生改变，那么路由表也会发生相应的变化，按照相应的改变，从新计算出最佳等网络传输路径而后在保存到路由表中。

GNU/Linux-Route

静态路由与动态路由等适用范围

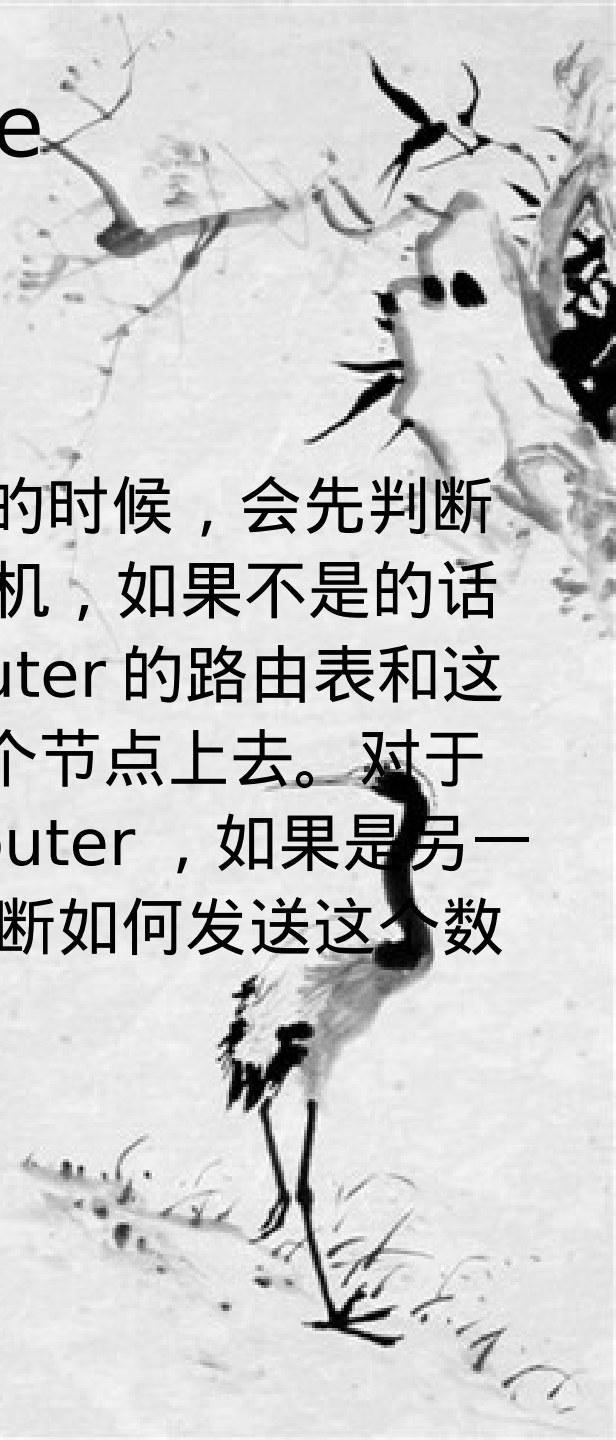
静态路由和动态路由有各自的特点和适用范围，因此在网络中动态路由通常作为静态路由的补充。当一个分组在路由器中进行寻径时，路由器首先查找静态路由，如果查到则根据相应的静态路由转发分组；否则再查找动态路由



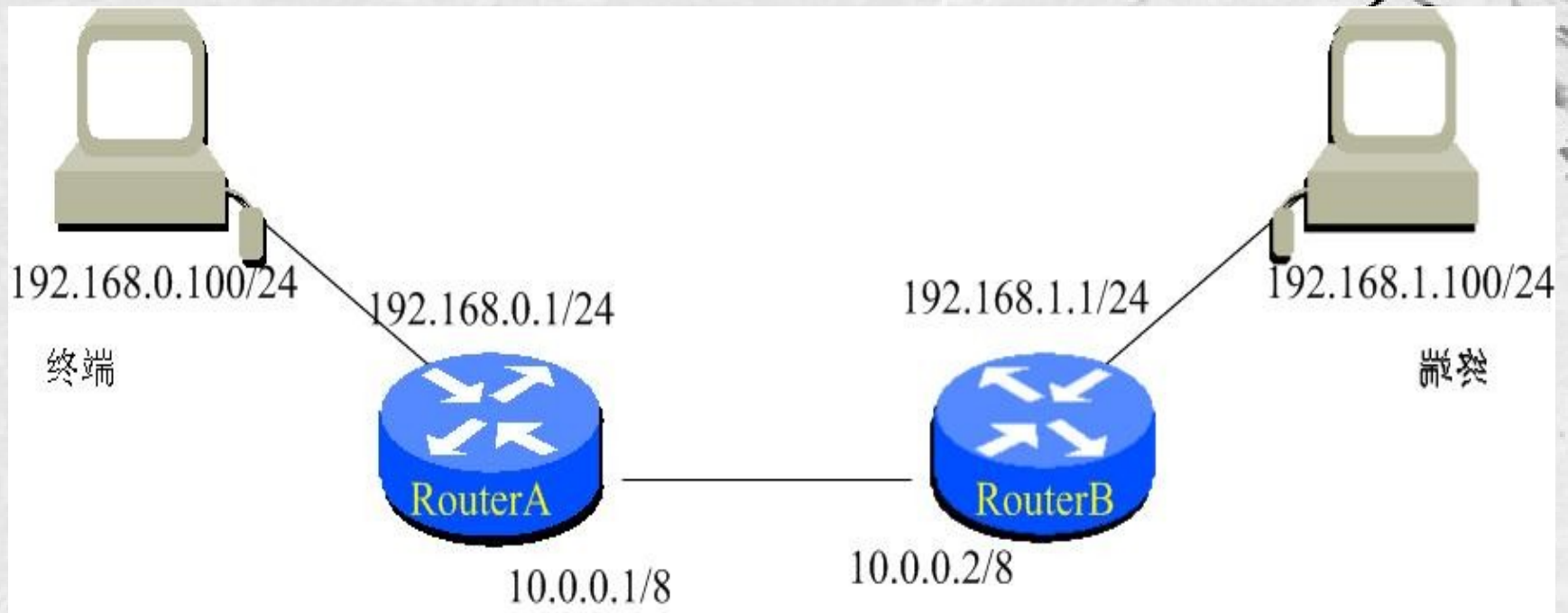
GNU/Linux-Route

IP 数据包的传输

在某一台计算机进行 IP 数据包传输的时候，会先判断这个数据包是不是发送给本地网络中的计算机，如果不是的话，则通过 router（也就是网关），根据 router 的路由表和这个 IP 的目的地将这个 IP 数据包发送到下一个节点上去。对于下一个节点也许是目的地，也许是另一个 router，如果是另一个 router 的话，那么就由那个 router 来判断如何发送这个数据包了。如下图



GNU/Linux-Route



GNU/Linux-Route

计算机 192.168.0.100/24 打算给 192.168.1.100/24 发送一个数据，那么就必须通过 0 段的网关 192.168.0.1/24，而后 router 将这个数据经过路由表通过这个路由另一个端口 10.0.0.1/8 发送到另一个 router 的 10.0.0.2/8 上，这样数据就穿过了一个路由器，我们一般都称为“一跳”，而后 routerB 在根据自己的路由表在把这个数据给 1 段的网关 192.168.1.1/24。实际上这也是路由 B 的另一个 IP。然后在把这个数据包通过 192.168.1.1/24 传送给 192.168.1.100/24. 就可以了。我们可以看到。在这里完全是 router table 在发挥重要的作用。

GNU/Linux-Route



路由器的功能

1. 转发数据包

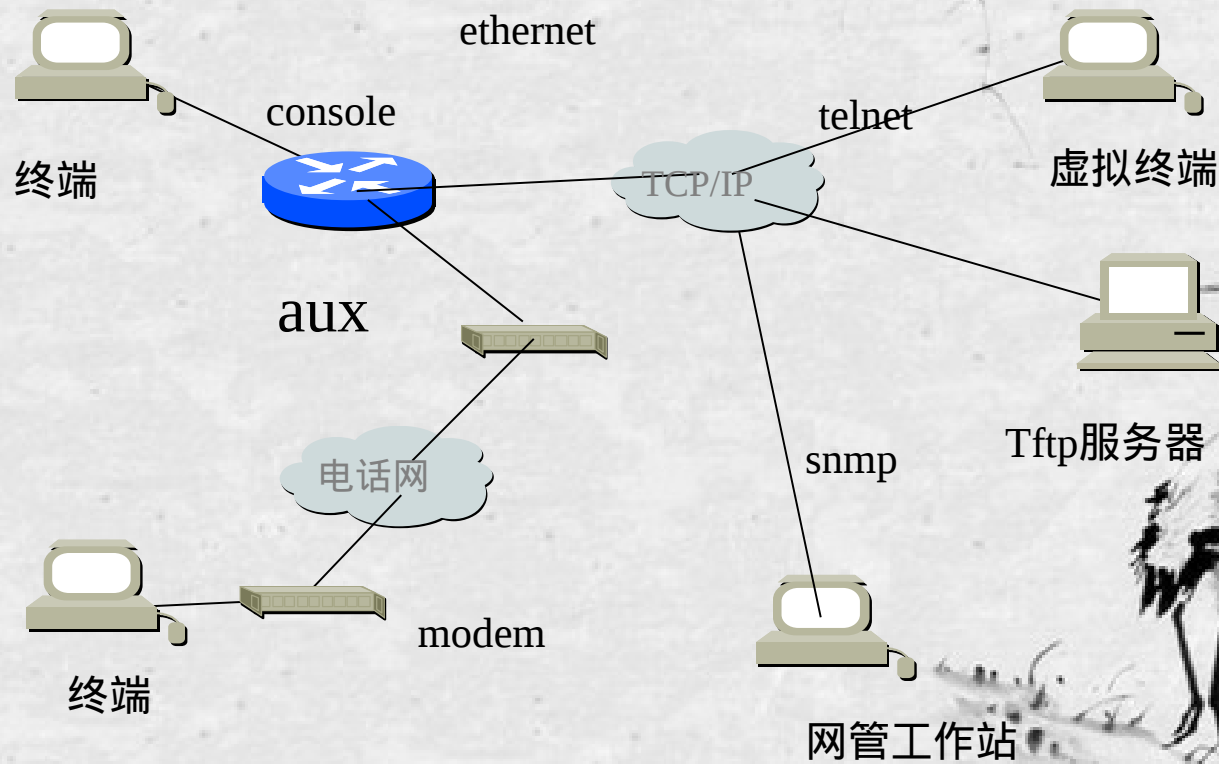
2. 选择最佳路径发送数据包

3. 路由器在转发数据的时候可以把比较大的数据包分解成比较小的数据包，这样可以保证数据包能够很好的穿越路由，我们把这个步骤叫做分解数据包。到达目的地后，在将由路由分解的数据包组合在一起。我们把这个步骤叫做组装

4. 路由器允许让使用不同协议的网络能够连接在一起形成一个网络。实际上在这个时候 router 就成为不同网络协议通信的平台了。

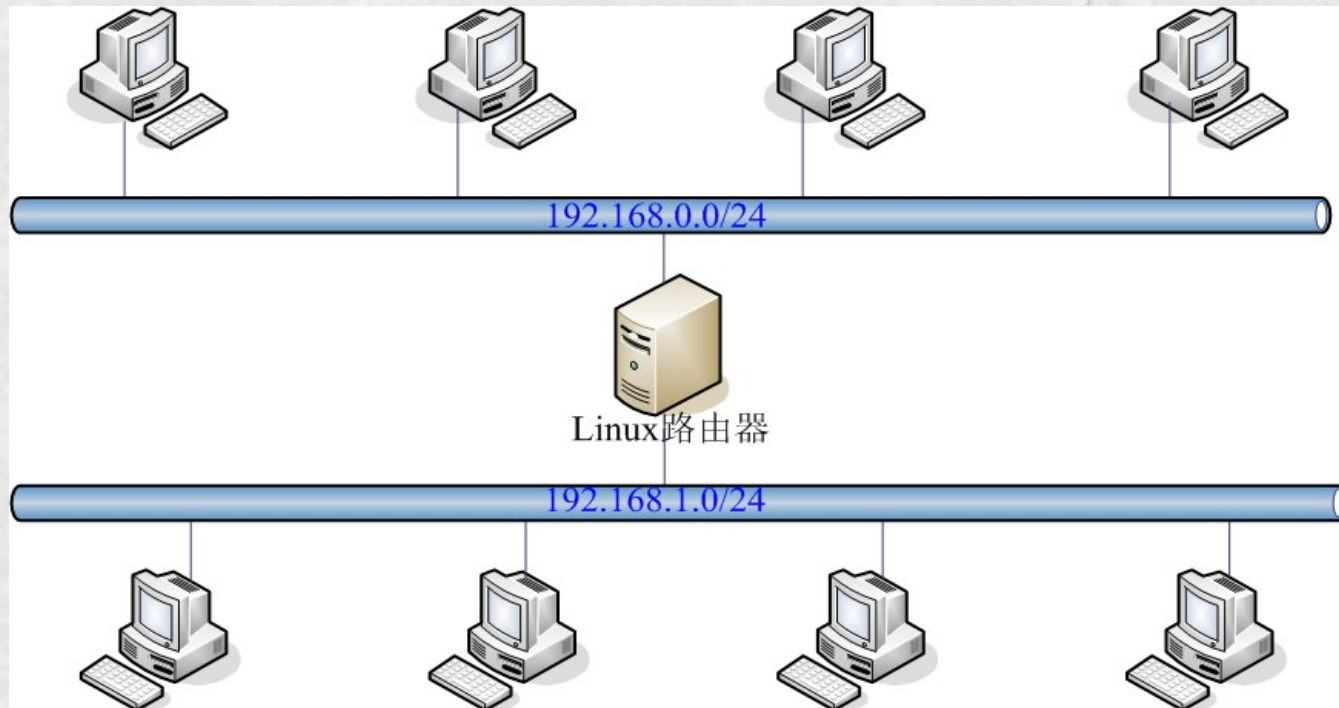
GNU/Linux-Route

路由器的用途



GNU/Linux-Route

建立基于 Linux 的静态路由



GNU/Linux-Route

在指定 Linux Route 上建立静态路由

1. 将多个网卡分别配置好 IP/ 或将 1 块网卡配置多个 IP

2. 开启路由转发

```
#vim /usr/lib/sysctl.d/00-system.conf
```

加入

```
net.ipv4.ip_forward = 1
```



GNU/Linux-Route

3. 将指定的目的网络条目写入至配置文件中

```
#cd /etc/sysconfig/network-scripts/route-  
<interface>
```

如

```
#echo "192.168.1.0/24 via 192.168.1.1 dev  
enp0s3" > route-enp0s3  
#nmcli dev con enp0s3
```

注意：

1. 有时当使用 `ip route` 查看路由表时，可能不存在所添加的条目。但条目已经存在

2. `/etc/sysconfig/network` 中也可以设定网关。但此网关为全局网关，将直接覆盖网卡配置及 `route-<interface>` 中所定义的网关

GNU/Linux-Route

在指定 Linux Route 上建立静态路由

4. 客户端分别配置好 IP 地址及网关

5. 确认客户端 / 服务器的路由表

6. 测试



GNU/Linux-Route

Destination 为目标的网络地址

Gateway 为通项这个网络段应该使用哪个网关

Flags 为标记。 U 为使用中。 G 为网关

Metric 为计量。可以说有的时候代表跳数

Ref 为参考

Use 为使用情况

Iface 为网关基于在哪个设备上。



GNU/Linux-Route

注：

如果您打算在设置客户端设置网关，您可以利用 route 执行下面的命令：

```
# route -add default gw 网关的 IP
```

如果打算删除网关，执行：

```
# route -del default gw 网关的 IP
```

如果打算添加一个静态路由，可执行：

```
# route add -net 192.168.2.0 netmask 255.255.255.0
```

如果打算删除一个静态路由，可执行：

```
# route del -net 192.168.2.0 netmask 255.255.255.0
```


GNU/Linux-Route

路由器是通过相应的计算得出最佳的网络路径，从而能够快速让数据包到达目的地。对于静态路由来说当网络出现问题时路由器不可能自己来重新计算网络最佳路径的。但是如果你使用动态的路由就可以。

注意：路由器如何能得出最佳的路径是看管理人员适用了什么样等动态路由协议

由此可见，如果对于打算使用动态协议的路由，就必须要知道一些动态路由协议。在下面我们列举了几个：

GNU/Linux-Route

1. 路由算法

如果打算适用动态路由协议，那么需要有个前提条件，那就是你需要知道什么是路由算法：

路由算法实际上就是寻找数据包传输中所能使用的最佳路径的计算方法。一个好的路由算法，在设计的时候最起码有以下几个特征：

- 1) 最优化：指路由算法选择最佳路径的能力。
- 2) 简洁性：算法设计简洁，利用最少的软件和开销，提供最有效的功能。

GNU/Linux-Route

1. 路由算法

3) 坚固性：路由算法处于非正常或不可预料的环境时，如硬件故障、负载过高或操作失误时，都能正确运行。由于路由器分布在网络联接点上，所以在它们出故障时会产生严重后果。最好的路由器算法通常能经受时间的考验，并在各种网络环境下被证实是可靠的。

4) 快速收敛：收敛是在最佳路径的判断上所有路由器达到一致的过程。当某个网络事件引起路由可用或不可用时，路由器就发出更新信息。路由更新信息遍及整个网络，引发重新计算最佳路径，最终达到所有路由器一致公认的最佳路径。收敛慢的路由算法会造成路径循环或网络中断。

GNU/Linux-Route

1. 路由算法

5) 灵活性：路由算法可以快速、准确地适应各种网络环境。例如，某个网段发生故障，路由算法要能很快发现故障，并为使用该网段的所有路由选择另一条最佳路径。



GNU/Linux-Route

2 . 路由算法的实施过程

1) 源计算机发送一个数据给目标段 , kernel 会对比这个数据是否发送给本地网络段。如果是则发送。此步骤是在本地网络中查找目的计算机

2) 如果不是 , 则通过 router 的路由表来判定这个数据包发送给哪个网络段的计算机 , 然后发送。

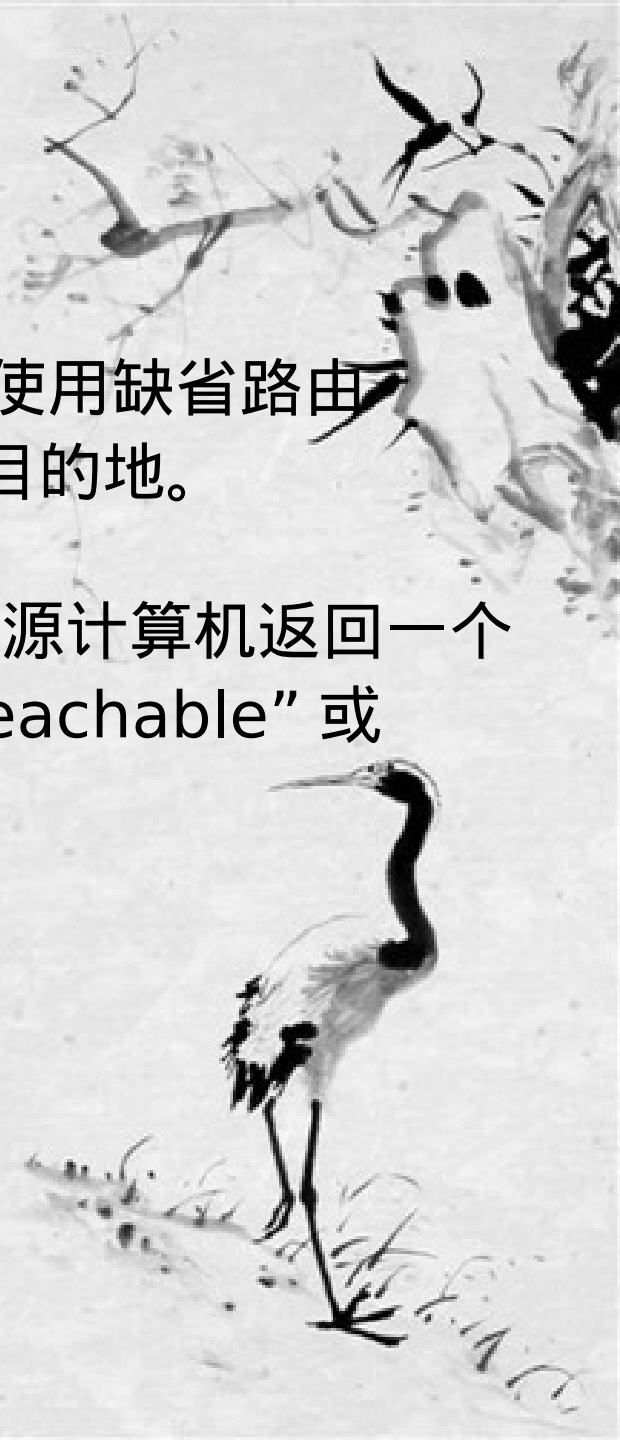
3) 如果没有找到匹配的 IP 地址 , 就会在 router 表中查找匹配的网络 ID , 然后就把这个数据包给另一个 router , 由其他 router 在寻找 , 来发送这个数据包给目的地计算机。

GNU/Linux-Route

2 . 路由算法的实施过程

4) 如果没有找到匹配的网络段，则使用缺省路由 0.0.0.0 来进行泛洪。如果找到，则发送到目的地。

5) 如果没有找到，则利用 ICMP 给源计算机返回一个目的不可达报告。比如：“network is unreachable” 或者 “No route to host”



GNU/Linux-Route

我们知道了路由算法之后，我们在来看看动态路由选择协议：

路由选择协议



GNU/Linux-Route

距离向量路由选择协议

1 . 距离向量路由选择协议也称为 B e l l m a n - F o r d 协议。
。距离向量协议路由器定期向相邻路由器发送两条消息：

- 到达目的网络所经过的跳距离，或者网络的数量。
- 下一个跳是什么，或者达到目的网络要使用的方向（ 向量 ）。



GNU/Linux-Route

工作原理

距离向量路由器定期向相邻的路由器发送它们的整个路由选择表。

距离相邻路由器在从相邻路由器接收到的信息的基础上建立自己的路由选择信息表。

而后，将信息传递到它的相邻路由器。



GNU/Linux-Route

工作原理

当在互连网络上无法使用某个路由时，距离向量路由器将通过路由变化或者网络链路寿命而了解这种变化。

路由器与故障链路相邻的路由器将在整个网络上发送“路由改变传输”（或者“路由无效”）消息。寿命将在所有的路由选择信息中设置。

当无法使用某个路由，并且并没有用新信息向网络发出这个信息时，距离向量路由选择算法在那个路由上设置一个寿命计时器。

当路由达到寿命计时器的终点时，它将从路由选择表中删除。寿命计时器根据所使用的路由选择协议不同而不同。

GNU/Linux-Route

无论使用何种类型的路由选择算法，互连网络上的所有路由器都需要时间以更新它们的路由选择表中的改动，这个过程称为聚合。因而，在距离向量路由选择中，聚合包括：

- 每个路由器接收到更新的路由选择信息。
- 每个路由器更新它自己的路由选择信息表。
- 每个路由器用它自己的信息（例如加入一个跳）更新其度。
- 每个路由器向它的邻居广播新信息。

距离向量路由选择是最古老的一种路由选择协议算法。正如前面说明的，算法的本质就是，每个路由器根据它从其他路由器接收到的信息而建立它自己的路由选择表。

GNU/Linux-Route

典型的距离向量路由选择协议 ----RIP（路由信息协议）



GNU/Linux-Route

RIP（路由信息协议）

RIP 协议最初是为 Xerox 网络系统的 Xerox parc 通用协议而设计的，是 Internet 中常用的路由协议。RIP 采用距离向量算法，即路由器根据距离选择路由，所以也称为距离向量协议。路由器收集所有可到达目的地的不同路径，并且保存有关到达每个目的地的最少站点数的路径信息，除到达目的地的最佳路径外，任何其它信息均予以丢弃。同时路由器也把所收集的路由信息用 RIP 协议通知相邻的其它路由器。这样，正确的路由信息逐渐扩散到了全网。RIP 使用非常广泛，它简单、可靠，便于配置。但是 RIP 只适用于小型的同构网络，因为它允许的最大站点数为 15，任何超过 15 个站点的目的地均被标记为不可达。而且 RIP 每隔 30 秒一次的路由信息广播也是造成网络的广播风暴的重要原因之一。

GNU/Linux-Route

RIP 的 V1 和 V2

RIPv1 是族类路由 (Classful Routing) 协议，因路由上不包括掩码信息，所以网络上的所有设备必须使用相同的子网掩码不支持 VLSM。RIPv2 可发送子网掩码信息，是非族类路由 (Classless Routing) 协议，支持 VLSM。

RIP 使用 UDP 数据更新路由信息。路由器每隔 30s 更新一次路由信息，如果在 180s 内没有收到相邻路由器的回应，则认为去往该路由器的路由不可用，该路由器不可到达。如果在 240s 后仍未收到该路由器的应答，则把有关该路由器的路由信息从路由表中删除。

GNU/Linux-Route

RIP 具有以下特点：

不同厂商的路由器可以通过 RIP 互联；

配置简单，适用于小型网络（小于 15 跳）；

RIPv1 不支持 VLSM；

需消耗广域网带宽；

需消耗 CPU、内存资源。



GNU/Linux-Route

RIP 的算法简单，但在路径较多时收敛速度慢，广播路由信息时占用的带宽资源较多，它适用于网络拓扑结构相对简单且数据链路故障率极低的小型网络中，在大型网络中，一般不使用 RIP



GNU/Linux-Route

链路状态路由选择协议

链路状态路由选择协议的目的是映射互连网络的拓扑结构。每个链路状态路由器提供关于它邻居的拓扑结构的信息。这包括：

- 路由器所连接的网段（链路）。
- 那些链路的情况（状态）。



GNU/Linux-Route

链路状态路由选择协议

整个信息在网络上泛洪，目的是所有的路由器可以接收到网络的拓扑结构。

链路状态路由器并不会广播包含在它们的路由表内的所有信息。相反，链路状态路由器将发送关于已经改动的路由的信息。

链路状态路由器将向它们的邻居发送呼叫消息，这称为链路状态数据包 (L S P) 或者链路状态通告 (L S A)。然后，邻居将 L S P 复制到它们的路由选择表中，并传递那个信息到网络的剩余部分。这个过程称为泛洪 (flooding)。它的结果是向网络发送第 1 手信息，为网络建立更新路由的准确映射。

GNU/Linux-Route

链路状态路由选择协议

链路状态路由选择协议使用称为代价的方法，而不是使用跳。代价是自动或人工赋值的。根据链路状态协议的算法，代价可以计算数据包必须穿越的跳数目、链路带宽、链路上的当前负载，或者甚至其他由管理员加入的权重来评价。



GNU/Linux-Route

链路状态路由选择协议

其工作原理：

- 1) 当一个链路状态路由器进入链路状态互连网络时，它发送一个呼叫数据包，以了解其邻居。
- 2) 邻居用关于它们所连接的链路以及相关的代价度的信息进行应答。
- 3) 起始的路由器用这个信息来建立它的路由选择表。



GNU/Linux-Route

4) 然后，作为定期更新的一部分。路由器向它的邻居发送链路状态数据包。这个 L S P 包括了那个路由器的链路及相关代价。

5) 每个邻居赋值数据包，并且将 L S P 传递到下一个邻居。这个过程称为泛洪。

6) 因为路由器并没有在向前泛洪 L S P 之前重新计算路由选择数据库，聚合时间减少了。



GNU/Linux-Route

链路状态路由选择协议的优点：

一：即路由选择循环不可能形成，原因是链路状态协议建立它们自己的路由选择信息表的方式。



GNU/Linux-Route

链路状态路由选择协议的优点：

二：在链路状态互连网络中聚合是非常快的，原因是一旦路由选择拓扑出现变动，则更新在互连网络上迅速泛洪。这些优点又释放了路由器的资源，因为对不好的路由信息所花费的处理能力和带宽消耗都很少。维护路由器区域的链路状态数据库将在路由器上加入 R A M 负担。类似的是，D i j k s t r a 算法不得不在每次路由改变的时候运行；这在所有的路由器上加重了 C P U 的负担。D i j k s t r a 算法首先是最短的路径，在这里对路径长度的迭代确定了最短的路径生成树。

GNU/Linux-Route

典型的链路状态路由协议 : OSPF

开放式最短路径优先 (Open Shortest Path First)



GNU/Linux-Route

OSPF：开放式最短路径优先（Open Shortest Path First）是一种链路状态路由协议。每一个 OSPF 路由器都维护一个相同的网络拓扑数据库，从这个数据库中，可以构造一个最短路径树来计算路由表。OSPF 的收敛速度比 RIP 要快，而且在更新路由信息时，产生的流量也较少。为了管理大规模的网络，OSPF 采用分层的连接结构。将自治系统分为不同的区域，以减少路由重计算的时间。



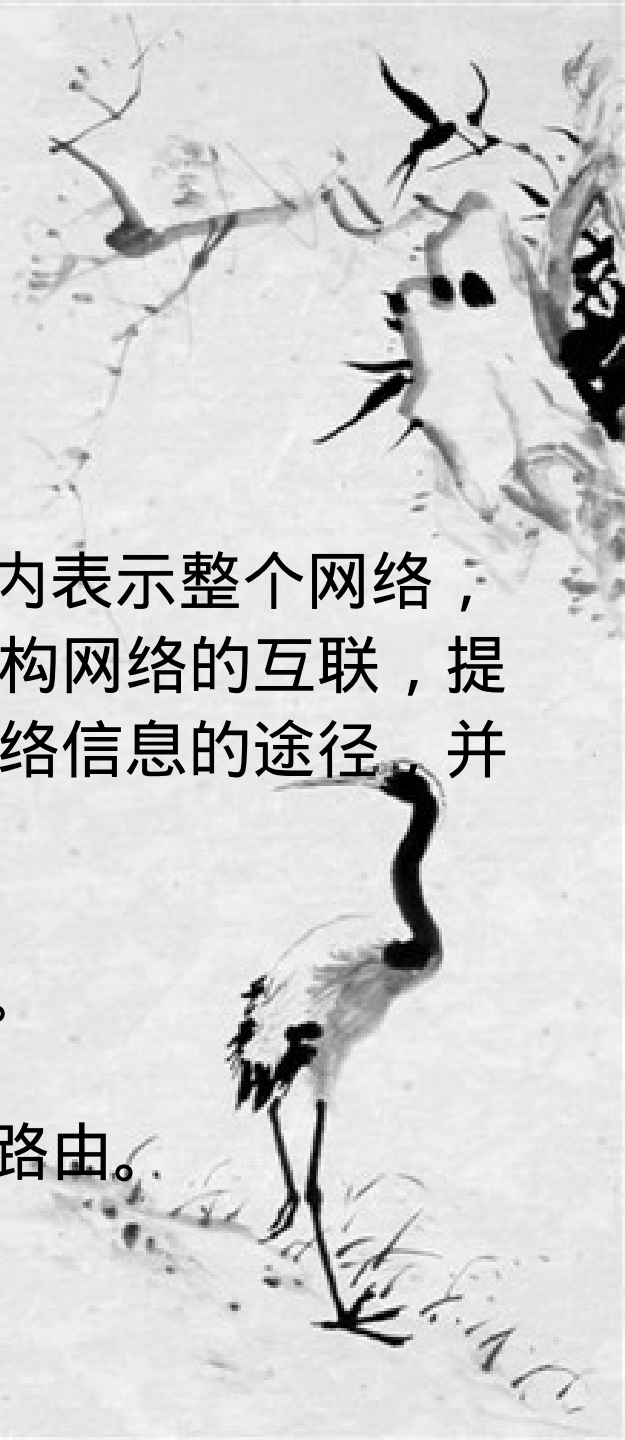
GNU/Linux-Route

OSPF 协议的优点

OSPF 能够在自己的链路状态数据库内表示整个网络，这极大地减少了收敛时间，并且支持大型异构网络的互联，提供了一个异构网络间通过同一种协议交换网络信息的途径，并且不容易出现错误的路由信息。

OSPF 支持通往相同目的的多重路径。

OSPF 使用路由标签区分不同的外部路由。



GNU/Linux-Route

OSPF 协议的优点

OSPF 支持路由验证，只有互相通过路由验证的路由器之间才能交换路由信息；并且可以对不同的区域定义不同的验证方式，从而提高了网络的安全性。



GNU/Linux-Route

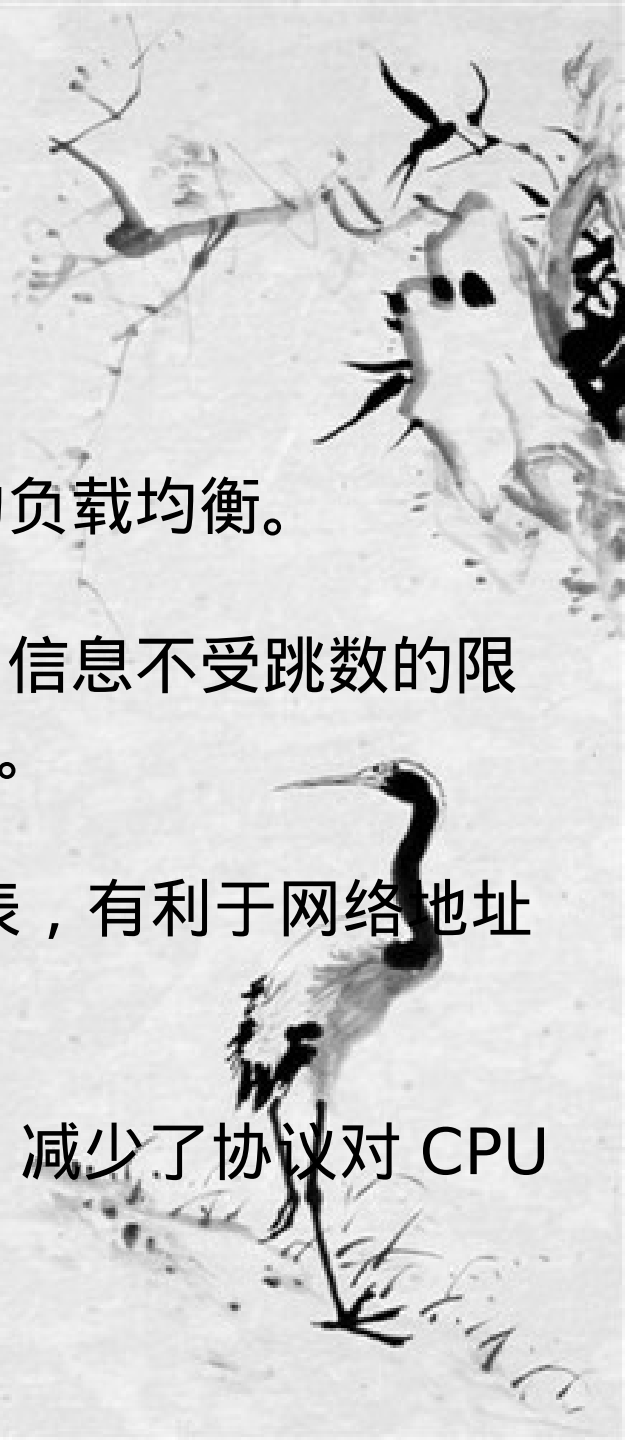
OSPF 协议的优点

OSPF 支持费用相同的多条链路上的负载均衡。

OSPF 是一个非族类路由协议，路由信息不受跳数的限制，减少了因分级路由带来的子网分离问题。

OSPF 支持 VLSM 和非族类路由查表，有利于网络地址的有效管理。

OSPF 使用 AREA 对网络进行分层，减少了协议对 CPU 处理时间和内存的需求。



GNU/Linux-Route

建立基于 Linux 的动态路由



GNU/Linux-Route

动态路由结构：

RouterA<--192.168.10.0/24-->ClientA

10.0.0.0/30

RouterB<--192.168.20.0/24-->ClientB



GNU/Linux-Route

为了我们能够让 192.168.10.0/24 与 192.168.20.0/24 两个网络段相互通信，需要路由器实现要求。此时可利用 RIP 来实现动态路由的功能。



GNU/Linux-Route

1. 安装 quagga

```
#yum install quagga -y
```

2. 准备 RIP 配置

```
#cp /usr/share/doc/quagga-*/ripd.conf.sample  
/etc/quagga/ripd.conf  
#cd /etc/quagga
```

3. 启动 RIP 及 quagga 服务

```
#systemctl start zebra  
#systemctl start ripd
```



GNU/Linux-Route

4. 查看服务端口

```
#netstat -lant | grep 2601 ←zebra  
#netstat -lant | grep 2602 ←ripd
```

5. 配置 quagga


```
#vtysh
```



GNU/Linux-Route

6. 配置信息 (RouterA)

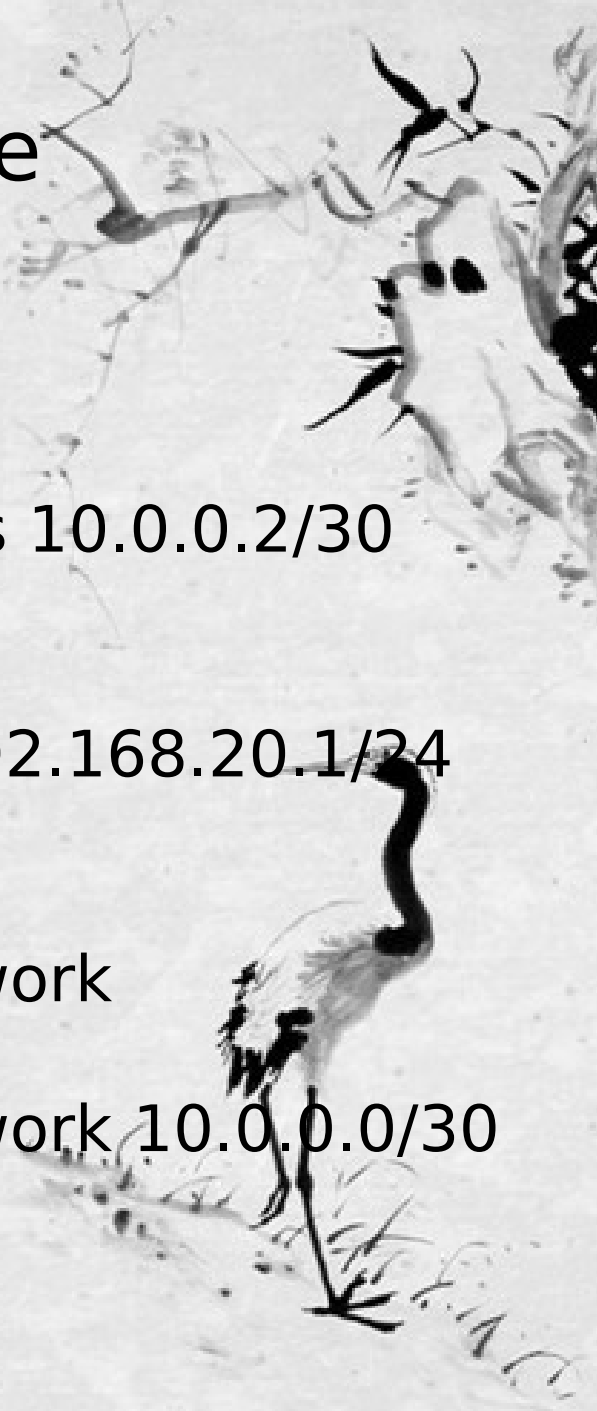
```
host_name#conf t
host_name(config)#int enp0s3
host_name(config-if)#ip address
192.168.10.1/24
host_name(config-if)#exit
host_name(config)#int enp0s8
host_name(config-if)#ip addr 10.0.0.1/30
host_name(config-if)#exit
host_name(config)#router rip
host_name(config-router)# network
192.168.10.0/24
host_name(config-router)# network 10.0.0.0/30
host_name(config-router)# end
host_name#copy run start
host_name#exit
```



GNU/Linux-Route

6. 配置信息 (RouterB)

```
host_name#conf t
host_name(config)#int enp0s3
host_name(config-if)#ip address 10.0.0.2/30
host_name(config-if)#exit
host_name(config)#int enp0s8
host_name(config-if)#ip addr 192.168.20.1/24
host_name(config-if)#exit
host_name(config)#router rip
host_name(config-router)# network
192.168.20.0/24
host_name(config-router)# network 10.0.0.0/30
host_name(config-router)# end
host_name#copy run start
host_name#exit
```



GNU/Linux-Route

7. 开启 2 台路由器的 ip 转发功能

```
#echo 1 > /proc/sys/net/ipv4/ip_forward
```

8. 查看路由器上的路由表

```
#vtysh
```

```
#sh ip route
```

9. 客户端测试



GNU/Linux-Route

利用 OSPF 搭建动态路由协议

OSPF 首先是在一个 AS 中进行工作的。所以在你配置 OSPF 的时候配置一个 AS 区域号

OSPF 的配置如下：



GNU/Linux-Route

1. 安装 quagga

```
#yum install quagga -y
```

2. 准备 ospfd 配置

```
#cp /usr/share/doc/quagga-  
*/ospfd.conf.sample /etc/quagga/ospfd.conf  
#cd /etc/quagga
```

3. 启动 ospfd 及 quagga 服务

```
#systemctl start ospfd  
#systemctl start zebra
```



GNU/Linux-Route

4. 查看服务端口

```
#netstat -lant | grep 2601 ←zebra  
#netstat -lant | grep 2604 ←ospfd
```

5. 配置 quagga

```
#vtysh
```



GNU/Linux-Route

6. 实现 OSPF 路由 (RouterA)

```
host_name#conf t
host_name(config)#int enp0s3
host_name(config-if)#ip address
192.168.10.1/24
host_name(config-if)#exit
host_name(config)#int enp0s8
host_name(config-if)#ip addr 10.0.0.1/30
host_name(config-if)#exit
```



GNU/Linux-Route

6. 实现 OSPF 路由 (RouterB)

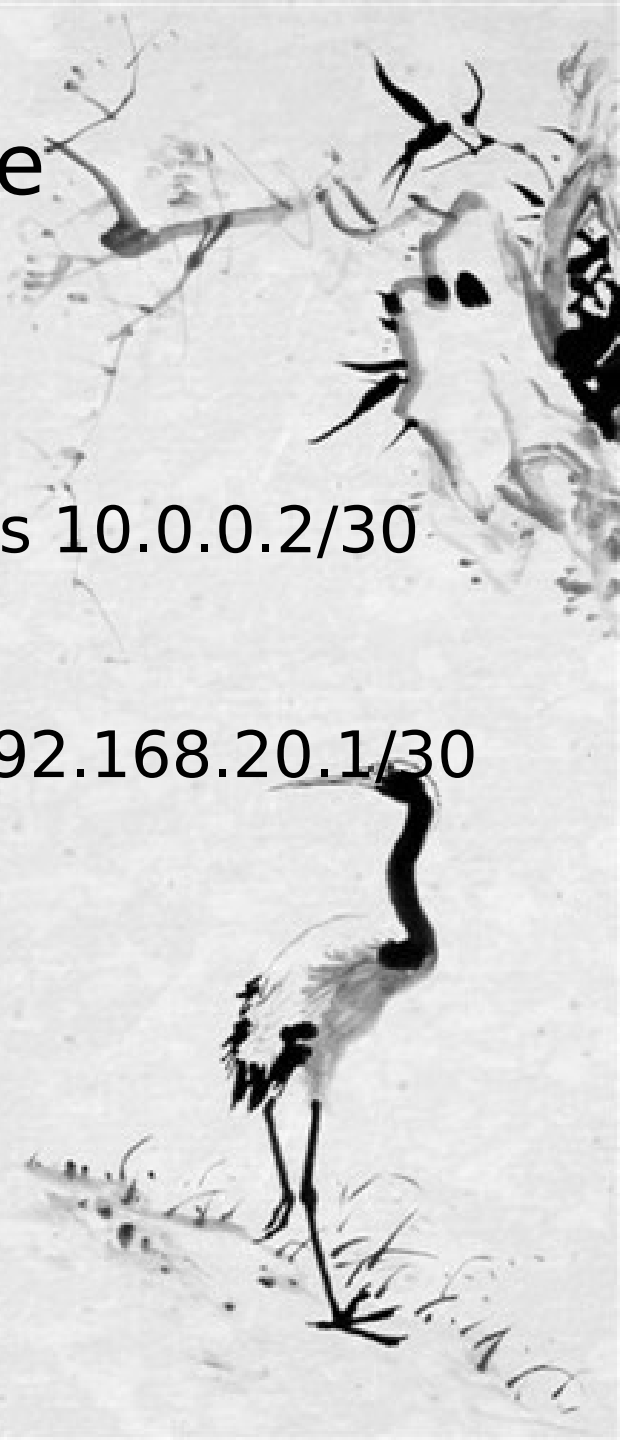
```
host_name(config)#router ospf
host_name(config-router)# network
192.168.10.0/24 area 10
host_name(config-router)# network 10.0.0.0/30
area 10
host_name(config-router)# ^z
host_name#copy run start
```



GNU/Linux-Route

6. 实现 OSPF 路由 (RouterB)

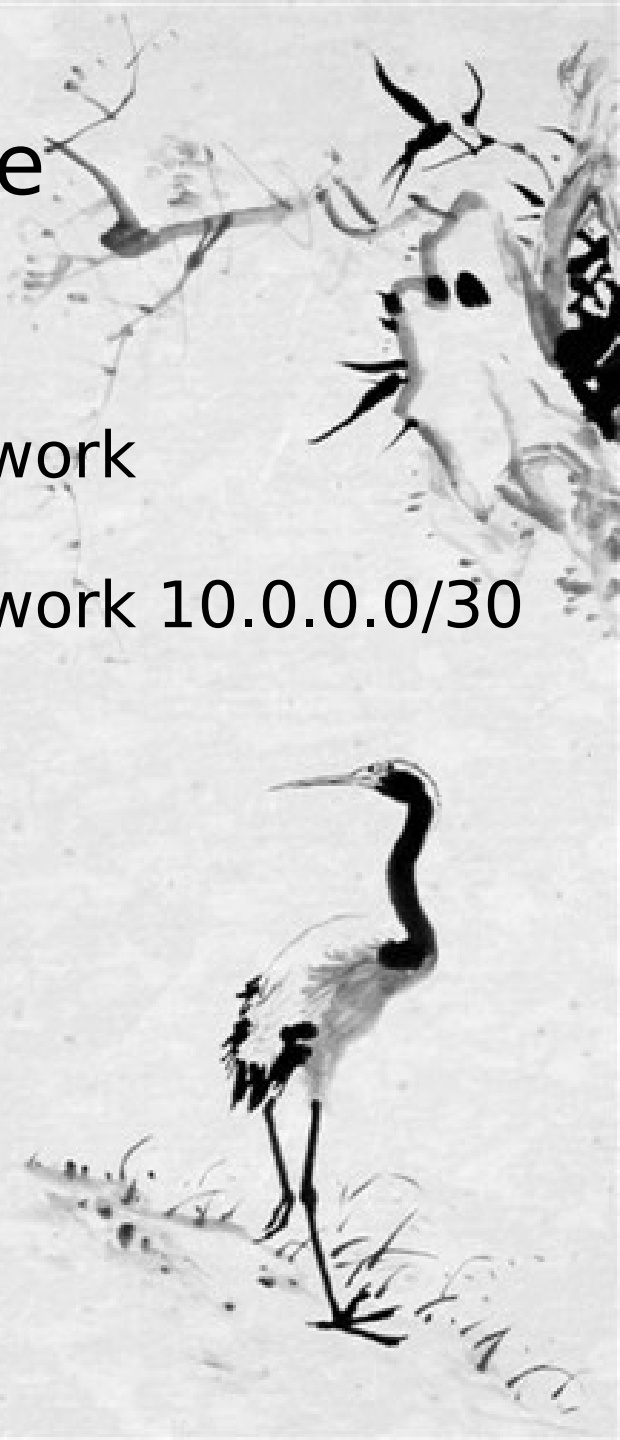
```
host_name#conf t
host_name(config)#int enp0s3
host_name(config-if)#ip address 10.0.0.2/30
host_name(config-if)#exit
host_name(config)#int enp0s8
host_name(config-if)#ip addr 192.168.20.1/30
host_name(config-if)#exit
```



GNU/Linux-Route

6. 实现 OSPF 路由 (RouterB)

```
host_name(config)#router ospf
host_name(config-router)# network
192.168.20.0/24 area 10
host_name(config-router)# network 10.0.0.0/30
area 10
host_name(config-router)# ^z
host_name#copy run start
```



GNU/Linux-Route

7. 开启 2 台路由器的 ip 转发功能

```
#echo 1 > /proc/sys/net/ipv4/ip_forward
```

8. 查看路由器上的路由表

```
#vtysh
```

```
#sh ip route
```

9. 客户端测试



GNU/Linux-Route

静态路由优先于动态路由：

静态路由一直优先于动态路由，主要原因在于系统认为由管理人员建立的路由是最可靠也是最快捷的。因此静态路由优先级最高。

如不使用静态路由，当动态路由按照自己的不通动态路由协议发现出各种不同的最佳路径的时候，它还可以判断 AD(管理距离) 值来决定哪个是最优的。

AD 值不仅在动态路由上拥有，而且静态路由也有 AD 值。而且静态路由的 AD 值比动态路由小。这也就是为什么静态路由比动态路由优先。

GNU/Linux-Route

AD(默认的管理距离) 值说明

路由源	默认 AD
-----	-------

连接接口	0
------	---

静态路由	1
------	---

EIGRP	90
-------	----

IGRP	100
------	-----



GNU/Linux-Route

AD(默认的管理距离) 值说明

路由源

默认 AD

OSPF

110

RIP

120

External

170

未知

255(这个路由将不会使用)

