第 3 章

运维必备的网络管理技能

Linux 系统在服务器市场占有很大的份额，尤其是在互联网时代，要使用计算机就离不开网络。本章将讲解 Linux 系统的网络配置。在开始配置网络之前，需要了解一些基本的网络原理。

对于提供互联网应用的服务器，网络防火墙是其抵御攻击破坏的安全屏障，如何在攻击时及时做出有效的措施是网络应用时时刻刻要面对的问题。高昂的硬件防火墙是一般开发者难以接受的。Linux 系统的出现，为开发者低成本解决安全问题提供了一种可行的方案。要熟练应用 Linux 防火墙，首先需要了解 TCP/IP 网络的基本原理，理解 Linux 防火墙的工作原理，并熟练掌握 Linux 系统下提供的各种工具。

如果管理的计算机有几十台，初始化服务器配置 IP 地址、网关和子网掩码等参数是一个烦琐耗时的过程。如果网络结构要更改，需要重新初始化网络参数，使用动态主机配置协议（Dynamic Host Configuration Protocol，DHCP）则可以避免此问题，客户端可以从 DHCP 服务端检索相关信息并完成相关网络配置，在系统重启后依然可以工作。尤其是在移动办公领域，只要区域内有一台 DHCP 服务器，用户就可以在办公室之间自由活动而不必担心网络参数配置的问题。DHCP 提供一种动态指定 IP 地址和相关网络配置参数的机制。

如今互联网应用越来越丰富，如仅仅用 IP 地址标识网络上的计算机是不可能完成任务的，也没有必要，于是产生了域名系统。域名系统通过一系列有意义的名称标识网络上的计算机，用户按域名请求某个网络服务时，域名系统负责将其解析为对应的 IP 地址，这便是 DNS。

本章主要涉及的知识点有：

- 网络管理协议
- 常用的网络管理命令
- Linux 的网络配置方法

3.1　网络管理协议介绍

要了解 Linux 的配置，首先需要了解相关的网络管理，本节主要介绍和网络配置密切相关的 TCP/IP 协议、UDP 协议和 ICMP 协议。

3.1.1　TCP/IP 概述

计算机网络是由地理上分散的、具有独立功能的多台计算机，通过通信设备和线路互相连接起来，在配有相应的网络软件的情况下，实现计算机之间通信和资源共享的系统。计算机网络按其所跨越的地理范围可分为局域网（Local Area Network，LAN）和广域网（Wide Area Network，WAN）。在整个计算机网络通信中，使用最为广泛的通信协议便是 TCP/IP 协议，为网络互联事实上的标准协议，每个接入互联网的计算机如果进行信息传输必然使用该协议。TCP/IP 协议主要包含传输控制协议（Transmission Control Protocol，TCP）和网际协议（Internet Protocol，IP）。

1. OSI 参考模型

计算机网络是为了实现计算机之间的通信，任何双方要成功地进行通信，必须遵守一定的信息交换规则和约定，在所有的网络中，每一层的目的都是向上一层提供一定的服务，同时利用下一层所提供的功能。TCP/IP 协议体系在和 OSI 协议体系的竞争中取得了决定性的胜利，得到了广泛的认可，成为事实上的网络协议体系标准。Linux 系统也是采用 TCP/IP 体系结构进行网络通信。TCP/IP 协议体系和 OSI 参考模型一样，也是一种分层结构。由基于硬件层次上的 4 个概念性层次构成，即网络接口层、网际互联层、传输层和应用层。OSI 参考模型与 TCP/IP 对比如图 3.1 所示。

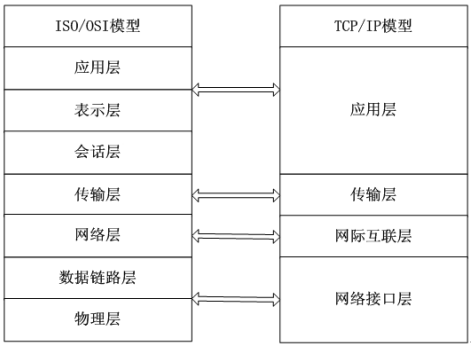

图 3.1　OSI 参考模型与 TCP/IP 协议对比

网络接口层主要为上层提供服务，完成链路控制等功能，网际互联层主要解决主机到主机之间的通信问题。其主要协议有：网际协议（IP）、地址解析协议（ARP）、反向地址解析协议

（RARP）和互联网控制报文协议（ICMP）。传输层为应用层提供端到端的通信功能，同时提供流量控制，确保数据完整和正确。TCP 协议位于该层，提供一种可靠的、面向连接的数据传输服务；与此对应的是 UDP 协议，提供不可靠的、无连接的数据报传输服务。应用层对应于 OSI 参考模型中的上面 3 层，为用户提供所需要的各种应用服务，如 FTP、Telnet、DNS、SMTP 等。

TCP/IP 协议体系及其实现中有很多概念和术语，为方便理解，本节集中介绍一些常用的概念与术语。

2. 包（packet）

包（packet）是网络上传输的数据片段，也称分组，同时称作 IP 数据报。用户数据按照规定划分为大小适中的若干组，每个组加上包头构成一个包，这个过程称为封装。网络上使用包为单位传输。包是一种统称，在不同的层次，包有不同的名字，如 TCP/IP 称作帧、IP 层称为 IP 数据报、TCP 层称为 TCP 报文等。图 3.2 为 IP 数据报格式。

<table>
<tr><td>0</td><td>4</td><td>8</td><td>16</td><td>20</td><td>31</td></tr>
<tr><td>版本</td><td>长度</td><td>服务类型</td><td colspan="3">总长度</td></tr>
<tr><td colspan="3">标识</td><td>标志</td><td colspan="2">分片位移</td></tr>
<tr><td colspan="2">时间</td><td>协议</td><td colspan="3">包头校验和</td></tr>
<tr><td colspan="6">源IP地址</td></tr>
<tr><td colspan="6">目的IP地址</td></tr>
<tr><td colspan="5">选项</td><td>填充</td></tr>
<tr><td colspan="6">数据</td></tr>
<tr><td colspan="6">其他</td></tr>
</table>

图 3.2　IP 数据报格式

3. 网络字节顺序

不同体系结构的计算机存储数据的格式和顺序不一样，要使用互联网互联就必须定义一个数据的表示标准。如一台计算机发送 1 个 32 位的整数至另外一台计算机，由于机器上存储整数的字节顺序可能不一样，按照源计算机的格式发送到目的主机可能会改变数字的值。TCP/IP 协议定义了一种所有机器在互联网分组的二进制字段中必须使用的网络标准字节顺序（network standard byte order），与此对应的是主机字节顺序，主机字节顺序是和各个主机密切相关的。传输时需要遵循以下转换规则：“主机字节顺序→网络字节顺序→主机字节顺序”，即发送方将主机字节顺序的整数转换为网络字节顺序然后发送出去，接收方收到数据后将网络字节顺序的整数转换为自己的主机字节顺序然后处理。

4. 地址解析协议（ARP）

TCP/IP 网络使用 IP 地址寻址，IP 包在 IP 层实现路由选择。但是 IP 包在数据链路层的传输却需要知道设备的物理地址，因此需要一种 IP 地址到物理地址的转换协议。TCP/IP 协议栈使用一种动态绑定技术来实现一种维护起来既高效又容易的机制，这就是地址解析协议（ARP）。

ARP 协议是在以太网这种有广播能力的网络中解决地址转换问题的方法。这种办法允许在不重新编译代码、不需维护一个集中式数据库的情况下，在网络中动态增加新机器。其原理简单描述为：当主机 A 想转换某一 IP 地址时，通过向网络中广播一个专门的报文分组，要求具有该 IP 地址机以其物理地址做出应答。所有主机都能收到这个请求，但是只有符合条件的主机才辨认该 IP 地址，同时发回一个应答，应答中包含其物理地址。主机 A 收到应答时便知道了该 IP 地址对应的物理硬件地址，并使用这个地址直接把数据分组发送出去。

3.1.2　UDP 与 ICMP 协议简介

UDP（User Datagram Protocol）是一种无连接的传输层协议，主要用于不要求分组顺序到达的传输中，分组传输顺序的检查与排序由应用层完成，提供面向事务的简单不可靠信息传送服务。UDP 具有不提供数据包分组、组装和不能对数据包进行排序的缺点，当报文发送之后，是无法得知其是否安全完整到达的，同时流量不易控制，若网络质量较差，则 UDP 协议数据包丢失会比较严重。UDP 协议具有资源消耗小、处理速度快的优点。

ICMP 是 Internet Control Message Protocol（Internet 控制报文协议）的缩写，属于 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、路由是否可用等网络本身的消息。如经常使用的用于检查网络通不通的 ping 命令，ping 的过程实际上就是 ICMP 协议工作的过程。ICMP 唯一的功能是报告问题而不是纠正错误，纠正错误的任务由发送方完成。

3.2　网络管理命令

在进行网络配置之前首先需要了解网络管理命令的使用，本节主要介绍网络管理中常用的命令。

3.2.1　检查网络是否通畅或网络连接速度 ping

ping 常常用来测试与目标主机或域名是否可达，通过发送 ICMP 数据包到网络主机、显示响应情况以及输出信息来确定目标主机或域名是否可达。ping 的结果通常情况下是可信的，由于有些服务器可以设置禁止 ping，从而使 ping 的结果并不是完全可信的。ping 命令常用的参数说明如表 3.1 所示。

Linux 下 ping 不会自动终止，需要按“Ctrl+c”组合键终止或用参数“-c”指定要求完成的回应次数。

表 3.1　ping命令常用参数说明

参数	说明
-d	使用 Socket 的 SO_DEBUG 功能
-f	极限检测，大量且快速地送网络封包给一台机器，看其回应
-n	只输出数值
-q	不显示任何传送封包的信息，只显示最后的结果
-r	忽略普通的 Routing Table，直接将数据包送到远端主机上
-R	记录路由过程
-v	详细显示指令的执行过程
-c	在发送指定数目的包后停止
-i	设定间隔几秒送一个网络封包给一台机器，预设值是一秒送一次
-I	使用指定的网络界面送出数据包
-l	设置在送出要求信息之前先行发出的数据包
-p	设置填满数据包的范本样式
-s	指定发送的数据字节数
-t	设置存活数值 TTL 的大小

ping 常见的用法如【示例 3-1】所示。

【示例 3-1】

```
#目的地址可以 ping 通
[root@CentOS ~]# ping 192.168.3.100
PING 192.168.3.100 (192.168.3.100) 56(84) bytes of data.
64 bytes from 192.168.3.100: icmp_seq=1 ttl=64 time=0.742 ms
64 bytes from 192.168.3.100: icmp_seq=2 ttl=64 time=0.046 ms

--- 192.168.3.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1993ms
rtt min/avg/max/mdev = 0.046/0.394/0.742/0.348 ms
#目的地址 ping 不通的情况
[root@CentOS ~]# ping 192.168.3.102
PING 192.168.3.102 (192.168.3.102) 56(84) bytes of data.
From 192.168.3.100 icmp_seq=1 Destination Host Unreachable
From 192.168.3.100 icmp_seq=2 Destination Host Unreachable
From 192.168.3.100 icmp_seq=3 Destination Host Unreachable
^C
--- 192.168.3.102 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3373ms
#ping 指定次数
[root@CentOS ~]# ping -c 1 192.168.3.100
PING 192.168.3.100 (192.168.3.100) 56(84) bytes of data.
64 bytes from 192.168.3.100: icmp_seq=1 ttl=64 time=0.235 ms
```

```
--- 192.168.3.100 ping statistics ---
1 packets transmitted,  1 received,  0% packet loss,  time 0ms
rtt min/avg/max/mdev = 0.235/0.235/0.235/0.000 ms
#指定时间间隔和次数限制的 ping
[root@CentOS ~]#  ping -c 3 -i 0.01 192.168.3.100
PING 192.168.3.100 (192.168.3.100) 56(84) bytes of data.
64 bytes from 192.168.3.100: icmp_seq=1 ttl=64 time=0.247 ms
64 bytes from 192.168.3.100: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 192.168.3.100: icmp_seq=3 ttl=64 time=0.026 ms

--- 192.168.3.100 ping statistics ---
3 packets transmitted,  3 received,  0% packet loss,  time 20ms
rtt min/avg/max/mdev = 0.026/0.101/0.247/0.103 ms
#ping 外网域名
[root@CentOS ~]# ping   -c 2 www.baidu.com
PING www.a.shifen.com (180.97.33.107) 56(84) bytes of data.
64 bytes from 180.97.33.107: icmp_seq=1 ttl=128 time=36.6 ms
64 bytes from 180.97.33.107: icmp_seq=2 ttl=128 time=36.1 ms

--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 36.125/36.373/36.622/0.313 ms
```

除了以上示例，ping 的各个参数还可以结合使用，读者可上机加以练习。

3.2.2　配置网络或显示当前网络接口状态 ifconfig

ifconfig 命令可以用于查看、配置、启用或禁用指定网络接口，如配置网卡的 IP 地址、掩码、广播地址、网关等，Windows 类似的命令为 ifconfig。语法如下：

```
#ifconfig interface [[-net -host] address [parameters]]
```

其中，interface 是网络接口名，address 是分配给指定接口的主机名或 IP 地址。-net 和-host 参数分别告诉 ifconfig 将这个地址作为网络号或是主机地址。与之前版本的网卡命名规则不同，CentOS 7.5 采用固件信息、网络拓扑等信息来命名网卡，这种方法更易于固定网卡的名称。Linux 系统中的网卡 lo 为本地环回接口，IP 地址固定为 127.0.0.1，子网掩码为 8 位，表示本机。ifconfig 常见的使用方法如【示例 3-2】所示。

【示例 3-2】

```
#查看网卡基本信息
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 192.168.228.129  netmask 255.255.255.0  broadcast 192.168.228.255
       inet6 fe80::ba7c:5b41:cee2:890  prefixlen 64  scopeid 0x20<link>
       ether 00:0c:29:8c:1e:52  txqueuelen 1000  (Ethernet)
       RX packets 111913  bytes 110290347 (105.1 MiB)
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34709  bytes 2150095 (2.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 420  bytes 36416 (35.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 420  bytes 36416 (35.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

#命令后面可接网络接口用于查看指定网络接口的信息
[root@localhost ~]# ifconfig ens33
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.228.129  netmask 255.255.255.0  broadcast 192.168.228.255
        inet6 fe80::ba7c:5b41:cee2:890  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8c:1e:52  txqueuelen 1000  (Ethernet)
        RX packets 112001  bytes 110296577 (105.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34735  bytes 2153674 (2.0 MiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

说明：

- 第 1 行：UP 表示此网络接口为启用状态，RUNNING 表示网卡设备已连接，MULTICAST 表示支持组播，mtu 为数据包最大传输单元。
- 第 2 行：依次为网卡 IP、子网掩码、广播地址。
- 第 3 行：IPv6 地址。
- 第 4 行：Ethernet（以太网）表示连接类型，ether 为网卡的 MAC 地址。
- 第 5 行：接收数据包个数、大小统计信息。
- 第 6 行：异常接收包的数量，如丢包量、错误等。
- 第 7 行：发送数据包个数、大小统计信息。
- 第 8 行：发送包的数量，如丢包量、错误等。

如果第 6 行和第 8 行中的丢包量、错误包量较高，通常表示物理链路存在问题，例如网线干扰过大、距离太长等。

设置 IP 地址使用以下命令：

```
#设置网卡 IP 地址
[root@CentOS ~]# ifconfig ens33:1 192.168.100.100 netmask 255.255.255.0 up
```

设置完后使用 ifconfig 命令查看，可以看到两个网卡信息，分别为 ens33 和 ens33:1。若继续设置其他 IP，可以使用类似的方法，如【示例 3-3】所示。

【示例 3-3】

```
#更改网卡的 MAC 地址
[root@CentOS ~]# ifconfig ens33:1 hw ether 00:0c:29:0b:07:77
[root@CentOS ~]# ifconfig ens33:1 | grep ether
       ether 00:0c:29:0b:07:77  txqueuelen 1000  (Ethernet)
#将某个网络接口禁用
#使用另一种形式表示子网掩码
[root@CentOS ~]# ifconfig ens33:1 192.168.100.170/24 up
[root@CentOS ~]# ifconfig ens33:1 down
[root@CentOS ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 192.168.128.129  netmask 255.255.255.0  broadcast 192.168.128.255
       inet6 fe80::20c:29ff:fe0b:776  prefixlen 64  scopeid 0x20<link>
       ether 00:0c:29:0b:07:77  txqueuelen 1000  (Ethernet)
       RX packets 1350  bytes 126861 (123.8 KiB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 878  bytes 158623 (154.9 KiB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
       inet 127.0.0.1  netmask 255.0.0.0
       inet6 ::1  prefixlen 128  scopeid 0x10<host>
       loop  txqueuelen 0  (Local Loopback)
       RX packets 8  bytes 764 (764.0 B)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 8  bytes 764 (764.0 B)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

除以上功能外，ifconfig 还可以设置网卡的 MTU。以上的设置会在重启后丢失，如需重启后依然生效，可以通过设置网络接口文件永久生效。更多使用方法可以参考系统帮助 man ifconfig。

在 CentOS 和 RHEL 中使用命令 ifup 和 ifdown 加网络接口名，可以启用、禁用对应的网络接口。

3.2.3　显示添加或修改路由表 route

route 命令用于查看或编辑计算机的 IP 路由表。route 命令的语法如下：

```
route [-f] [-p] [command] [destination] [mask netmask] [gateway] [metric][ [dev] If ]
```

参数说明：

- command：指定想要进行的操作，如 add、change、delete、print。

- destination：指定该路由的网络目标。
- mask netmask：指定与网络目标相关的子网掩码。
- gateway：网关。
- metric：为路由指定一个整数成本指标，当路由表的多个路由进行选择时可以使用。
- dev if：为可以访问目标的网络接口指定接口索引。

route 使用方法如【示例 3-4】所示。

【示例 3-4】

```
#显示所有路由表
[root@CentOS ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.3.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0
#添加一条路由:发往192.168.60.0网段的全部要经过网关192.168.19.1
route add -net 192.168.60.0 netmask 255.255.255.0 gw 192.168.19.1
#删除一条路由，删除的时候不需要网关
route del -net 192.168.60.0 netmask 255.255.255.0
```

3.2.4 复制文件至其他系统 scp

本地主机需要和远程主机进行数据迁移或文件传送时，可以使用 ftp，或搭建 Web 服务，另外可选的方法有 scp 或 rsync。scp 可以将本地文件传送到远程主机或从远程主机拉取文件到本地。其一般语法如下所示，注意由于各个发行版不同，scp 语法不尽相同，具体使用方法可查看系统帮助。

```
   scp [-1245BCpqrv] [-c cipher] [F SSH_config] [-I identity_file] [-l limit] [-o
SSH_option] [-P port] [-S program] [[user@]host1:] file1 […] [[suer@]host2:]file2
```

scp 命令执行成功返回 0，失败或有异常时返回大于 0 的值，常用参数说明如表 3.2 所示。

表 3.2　scp 命令常用参数说明

参数	说明
-P	指定远程连接端口
-q	把进度参数关掉
-r	递归地复制整个文件夹
-V	冗余模式。打印排错信息方便问题定位

scp 使用方法如【示例 3-5】所示。

【示例 3-5】

```
#将本地文件传送至远程主机192.168.3.100的/usr 路径下
[root@CentOS ~]# scp -P 12345  cgi_mon   root@192.168.3.100:/usr
root@192.168.3.100's password:
```

```
cgi_mon                                              100% 6922     6.8KB/s   00:00
#拉取远程主机文件至本地路径
[root@CentOS ~]# scp -P 12345 root@192.168.3.100:/etc/hosts ./
root@192.168.3.100's password:
hosts                                                100%  284     0.3KB/s   00:00
#如需传送目录，可以使用参数“r”
[root@CentOS soft]# scp -r  -P 12345 root@192.168.3.100:/usr/local/apache2.
root@192.168.3.100's password:
logresolve.8           100% 1407     1.4KB/s   00:00
rotatelogs.8           100% 5334     5.2KB/s   00:00
……
#将本地目录传送至远程主机指定目录
[root@CentOS soft]# scp -r apache2 root@192.168.3.100:/data
root@192.168.3.100's password:
logresolve.8      100% 1407     1.4KB/s   00:00
rotatelogs.8      100% 5334     5.2KB/s   00:00
……
```

3.2.5　复制文件至其他系统 rsync

rsync 是 Linux 系统下常用的数据镜像备份工具，用于在不同的主机之间同步文件。除了单个文件，rsync 可以镜像保存整个目录树和文件系统，并可以增量同步，并保持文件原来的属性，如权限、时间戳等。rsync 数据传输过程中是加密的，保证数据的安全性。

rsync 命令语法如下：

```
Usage: rsync [OPTION]... SRC [SRC]... DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
  or   rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
  or   rsync [OPTION]... [USER@]HOST:SRC [DEST]
  or   rsync [OPTION]... [USER@]HOST::SRC [DEST]
  or   rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

OPTION 可以指定某些选项，如压缩传输、是否递归传输等，SRC 为本地目录或文件，USER 和 HOST 表示可以登录远程服务的用户名和主机，DEST 表示远程路径。rsync 常用参数如表 3.3 所示。由于参数众多，这里只列出某些有代表性的参数。

表 3.3　rsync 命令常用参数说明

参数	说明
-v	详细模式输出
-q	精简输出模式
-c	打开校验开关，强制对文件传输进行校验
-a	归档模式，表示以递归方式传输文件，并保持所有文件属性，等于-rlptgoD
-r	对子目录以递归模式处理
-R	使用相对路径信息

（续表）

参数	说明
-p	保持文件权限
-o	保持文件属主信息
-g	保持文件属组信息
-t	保持文件时间信息
-n	显示哪些文件将被传输
-W	复制文件，不进行增量检测
-e	指定使用 rsh、SSH 方式进行数据同步
--delete	删除那些 DST 中 SRC 没有的文件
--timeout=TIME	IP 超时时间，单位为秒
-z	对备份的文件在传输时进行压缩处理
--exclude=PATTERN	指定排除不需要传输的文件模式
--include=PATTERN	指定不排除而需要传输的文件模式
--exclude-from=FILE	排除 FILE 中指定模式的文件
--include-from=FILE	不排除 FILE 指定模式匹配的文件
--version	打印版本信息
-address	绑定到特定的地址
--config=FILE	指定其他的配置文件，不使用默认的 rsyncd.conf 文件
--port=PORT	指定其他的 rsync 服务端口
--progress	在传输时显示传输过程
--log-format=format	指定日志文件格式
--password-file=FILE	从 FILE 中得到密码

rsync 使用方法如【示例 3-6】所示。

【示例 3-6】

```
   #传送本地文件到远程主机
   [root@CentOS local]# rsync  -v  --port 56789  b.txt root@192.168.3.100::BACKUP
   b.txt
   sent 67 bytes  received 27 bytes  188.00 bytes/sec
   total size is 2  speedup is 0.02
   #传送目录至远程主机
   [root@CentOS local]# rsync  -avz  --port 56789  apache2
root@192.168.3.100::BACKUP
   #部分结果省略
   apache2/modules/mod_vhost_alias.so

   sent 27983476 bytes  received 187606 bytes  5122014.91 bytes/sec
   total size is 48113101  speedup is 1.71
   #拉取远程文件至本地
   [root@CentOS local]# rsync   --port 56789 -avz
```

```
root@192.168.3.100::BACKUP/apache2/test.txt .
    receiving incremental file list
    test.txt
    sent 47 bytes  received 102 bytes  298.00 bytes/sec
    total size is 2  speedup is 0.01
    #拉取远程目录至本地
    [root@CentOS local]# rsync   --port 56789 -avz
root@192.168.3.100::BACKUP/apache2 .
    #部分结果省略
    apache2/modules/mod_version.so
    apache2/modules/mod_vhost_alias.so
    sent 16140 bytes  received 13866892 bytes  590767.32 bytes/sec
    total size is 48113103  speedup is 3.47
```

rsync 具有增量传输的功能，利用此特性可以用来做文件的增量备份。通过 rsync 可以解决对实时性要求不高的数据备份需求。随着文件增多，rsync 做数据同步时，需要扫描所有文件后进行对比，然后进行差量传输。如果文件很多，扫描文件是非常耗时的，使用 rsync 反而比较低效。

使用 rsync 之前需要做一些简单的配置，读者可自行参考相关文档了解。

3.2.6　显示网络连接、路由表或接口状态 netstat

netstat 命令用于监控系统网络配置和工作状况，可以显示内核路由表、活动的网络状态以及每个网络接口有用的统计数字。常用的参数如表 3.4 所示。

表 3.4　netstat 命令常用参数说明

参数	说明
-a	显示所有连接中的 Socket
-c	持续列出网络状态
-h	在线帮助
-i	显示网络界面
-l	显示监控中的服务器的 Socket
-n	直接使用 IP 地址
-p	显示正在使用 Socket 的程序名称
-r	显示路由表
-s	显示网络工作信息统计表
-t	显示 TCP 端口情况
-u	显示 UDP 端口情况
-v	显示命令执行过程
-V	显示版本信息

netstat 常见使用方法如【示例 3-7】所示。

【示例 3-7】

```
#显示所有端口，包含 UDP 和 TCP 端口
[root@CentOS local]# netstat -a|head -4
getnameinfo failed
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address             Foreign Address           State
tcp       0      0 *:rquotad                *:*                     LISTEN
tcp       0      0 *:55631                  *:*                     LISTEN
……
#显示所有 TCP 端口
[root@CentOS local]# netstat -at
#部分结果省略
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address             Foreign Address           State
tcp       0      0 192.168.3.100:56789       *:*                     LISTEN
tcp       0      0 *:nfs                    *:*                      LISTEN
#
#显示所有 UDP 端口
[root@CentOS local]# netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address             Foreign Address           State
udp       0      0 *:nfs                    *:*
udp       0      0 *:43801                  *:*
#显示所有处于监听状态的端口并以数字方式显示而非服务名
[root@CentOS local]# netstat -ln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address             Foreign Address           State
tcp       0      0 0.0.0.0:111              0.0.0.0:*                LISTEN
tcp       0      0 192.168.3.100:56789       0.0.0.0:*                LISTEN
#显示所有 TCP 端口并显示对应的进程名称或进程号
[root@CentOS local]# netstat -plnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address State    PID/Program name
tcp    0     0 0.0.0.0:111      0.0.0.0:*         LISTEN     5734/rpcbind
tcp    0     0 0.0.0.0:58864    0.0.0.0:*         LISTEN     5818/rpc.mountd
#显示核心路由信息
[root@CentOS local]# netstat -r
Kernel IP routing table
Destination    Gateway        Genmask        Flags   MSS Window  irtt Iface
192.168.3.0    *              255.255.255.0  U         0 0          0 eth0
#显示网络接口列表
[root@CentOS local]# netstat -i
Kernel Interface table
Iface     MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
```

```
eth0      1500   0    26233      0      0      0    27142      0      0      0 BMRU
eth0:5    1500   0      - no statistics available -                              BMRU
lo       16436   0    45402      0      0      0    45402      0      0      0 LRU
#综合示例，统计各个 TCP 连接的各个状态对应的数量
[root@CentOS local]# netstat -plnta|sed '1,2d'|awk '{print $6}'|sort|uniq -c
     1 ESTABLISHED
    21 LISTEN
```

netstat 工具是运维工程师最常用的工具之一，经常被用来查看主机网络状态、监听列表等，因此需要掌握好此工具的使用方法。

3.2.7　探测至目的地址的路由信息 traceroute

traceroute 跟踪数据包到达网络主机所经过的路由，原理是试图以最小的 TTL 发出探测包来跟踪数据包到达目标主机所经过的网关，然后监听一个来自网关 ICMP 的应答。使用语法如下：

```
traceroute [-m Max_ttl] [-n ] [-p Port] [-q Nqueries] [-r] [-s SRC_Addr]
[-t TypeOfService] [-v] [-w WaitTime] Host [PacketSize]
```

常用参数如表 3.5 所示。

表 3.5　traceroute 命令常用参数说明

参数	说明
-f	设置第一个检测数据包的存活数值TTL的大小
-g	设置来源路由网关，最多可设置8个
-i	使用指定的网络界面送出数据包
-I	使用ICMP回应取代UDP资料信息
-m	设置检测数据包的最大存活数值TTL的大小，默认值为30次
-n	直接使用IP地址而非主机名称。当DNS不起作用时常用到这个参数
-p	设置UDP传输协议的通信端口。默认值是33434
-r	忽略普通的路由表Routing Table，直接将数据包送到远端主机上
-s	设置本地主机送出数据包的IP地址
-t	设置检测数据包的TOS数值
-v	详细显示指令的执行过程
-w	设置等待远端主机回报的时间，默认值为3秒
-x	开启或关闭数据包的正确性检验
-q n	在每次设置生存期时，把探测包的个数设置为n，默认值为3

traceroute 常用操作如【示例 3-8】所示。

【示例 3-8】

```
[root@CentOS local]#  ping www.php.net
PING www.php.net (69.147.83.199) 56(84) bytes of data.
```

```
   64 bytes from www.php.net (69.147.83.199): icmp_seq=1 ttl=50 time=213 ms
   #显示本地主机到 www.php.net 所经过的路由信息
   [root@CentOS local]# traceroute -n www.php.net
   traceroute to www.php.net (69.147.83.199), 30 hops max, 40 byte packets
   #第3跳到达深圳联通
    3  120.80.198.245 (120.80.198.245)  4.722 ms   4.273 ms   1.925 ms
   #第9跳到达美国
    9  208.178.58.173 (208.178.58.173)  185.117 ms 64.23.107.149 (64.23.107.149)
184.838 ms 208.178.58.173 (208.178.58.173)  185.422 ms
   #美国
   13  98.136.16.61 (98.136.16.61)  216.602 ms 209.131.32.53 (209.131.32.53)
216.779 ms 209.131.32.55 (209.131.32.55)  214.934 ms
   #第14跳到达 php.net 对应的主机信息
   14  69.147.83.199 (69.147.83.199)  213.893 ms   213.536 ms   213.476 ms
   #域名不可达，最大30跳
   [root@CentOS local]#  traceroute -n  www.mysql.com
   traceroute to www.mysql.com (137.254.60.6), 30 hops max, 40 byte packets
   16  141.146.0.137 (141.146.0.137)  201.945 ms   201.372 ms   201.241 ms
   17  * * *
   #部分结果省略
   29  * * *
   30  * * *
```

以上示例每行记录对应一跳，每跳表示一个网关，每行有 3 个时间，单位是 ms，如域名不通或主机不通可根据显示的网关信息定位。星号表示 ICMP 信息没有返回，以上示例访问 www.mysql.com 时不通，数据包到达某一节点时没有返回，可以将此结果提交 IDC 运营商，以便于解决问题。

traceroute 实际上是通过给目标机的一个非法 UDP 端口号发送一系列 UDP 数据包来工作的。使用默认设置时，本地机给每个路由器发送 3 个数据包，最多可经过 30 个路由器。如果已经经过了 30 个路由器，但还未到达目标机，那么 traceroute 将终止。每个数据包都对应一个 Max_ttl 值，同一跳步的数据包该值一样，不同跳步的数据包的值从 1 开始，每经过一个跳步值加 1。当本地机发出的数据包到达路由器时，路由器就响应一个 ICMPTimeExceed 消息，于是 traceroute 就显示出当前跳步数、路由器的 IP 地址或名字、3 个数据包分别对应的周转时间（以 ms 为单位）。如果本地机在指定的时间内未收到响应包，那么在数据包的周转时间栏就显示出一个星号。当一个跳步结束时，本地机根据当前路由器的路由信息，给下一个路由器发出 3 个数据包，周而复始，直到收到一个 ICMPPORT_UNREACHABLE 的消息，意味着已到达目标机，或已达到指定的最大跳步数。

3.2.8 测试、登录或控制远程主机 telnet

telnet 命令通常用来远程登录。telnet 程序是基于 TELNET 协议的远程登录客户端程序。TELNET 协议是 TCP/IP 协议族中的一员，是 Internet 远程登录服务的标准协议和主要方式，

为用户提供了在本地计算机上完成远程主机工作的能力。在客户端可以使用 telnet 在程序中输入命令，可以在本地控制服务器。由于 telnet 采用明文传送报文，因此安全性较差。telnet 可以确定远程服务端口的状态，以便确认服务是否正常。telnet 常用方法如【示例 3-9】所示。

【示例 3-9】

```
#检查对应服务是否正常
[root@CentOS Packages]# telnet 192.168.3.100 56789
Trying 192.168.3.100...
Connected to 192.168.3.100.
Escape character is '^]'.
@RSYNCD: 30.0
as
@ERROR: protocol startup error
Connection closed by foreign host.
[root@CentOS local]#  telnet www.php.net 80
Trying 69.147.83.199...
Connected to www.php.net.
Escape character is '^]'.
test
#部分结果省略
</html>Connection closed by foreign host.
```

如果发现端口可以正常 telnet 登录，就表示远程服务正常。除确认远程服务是否正常外，对于提供开放 telnet 功能的服务，使用 telnet 可以登录远程端口，输入合法的用户名和口令后，就可以进行其他工作了。更多的使用帮助可以查看系统帮助。

3.2.9　下载网络文件 wget

wget 类似 Windows 中的下载工具，大多数 Linux 发行版本都默认包含此工具。用法比较简单，如要下载某个文件，可以使用以下命令：

```
#使用语法为 wget [参数列表] [目标软件、网页的网址]
[root@CentOS data]# wget  http://ftp.gnu.org/gnu/wget/wget-1.14.tar.gz
```

wget 常用参数说明如表 3.6 所示。

表 3.6　wget命令常用参数说明

参数	说明
-b	后台执行
-d	显示调试信息
-nc	不覆盖已有的文件
-c	断点下传
-N	该参数指定 wget 只下载更新的文件
-S	显示服务器响应

（续表）

参数	说明
-T timeout	超时时间设置（单位秒）
-w time	重试延时（单位秒）
-Q quota=number	重试次数
-nd	不下载目录结构，把从服务器所有指定目录下载的文件都堆到当前目录里
-nH	不创建以目标主机域名为目录名的目录，将目标主机的目录结构直接下到当前目录下
-l [depth]	下载远程服务器目录结构的深度
-np	只下载目标站点指定目录及其子目录的内容

wget 具有强大的功能，比如断点续传，可同时支持 FTP 或 HTTP 协议下载，并可以设置代理服务器。常用方法如【示例 3-10】所示。

【示例 3-10】

```
   #下载某个文件
   [root@CentOS data]# wget  http://ftp.gnu.org/gnu/wget/wget-1.14.tar.gz
   --15:47:51--  http://ftp.gnu.org/gnu/wget/wget-1.14.tar.gz
             => `wget-1.14.tar.gz'
   Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
   Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
   HTTP request sent, awaiting response... 200 OK
   Length: 3,118,130 (3.0M) [application/x-gzip]

   100%[======================================================================>]
3,118,130    333.55K/s    ETA 00:00

   15:48:03 (273.52 KB/s) - `wget-1.14.tar.gz' saved [3118130/3118130]
   #断点续传
   [root@CentOS data]#  wget  -c  http://ftp.gnu.org/gnu/wget/wget-1.14.tar.gz
   --15:49:55--  http://ftp.gnu.org/gnu/wget/wget-1.14.tar.gz
             => `wget-1.14.tar.gz'
   Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
   Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
   HTTP request sent, awaiting response... 206 Partial Content
   Length: 3,118,130 (3.0M), 1,404,650 (1.3M) remaining [application/x-gzip]

   100%[++++++++++++++++++++++++++++++++++++++==============================>]
3,118,130    230.83K/s    ETA 00:00

   15:50:04 (230.52 KB/s) - `wget-1.14.tar.gz' saved [3118130/3118130]
   #批量下载，其中 download.txt 文件中是一系列网址
   [root@CentOS data]# wget -i download.txt
```

wget 其他用法可参考系统帮助，其功能等待慢慢探索。

3.3 Linux 网络配置

Linux 系统在服务器中占用较大份额，使用计算机首先要了解网络配置，本节主要介绍 Linux 系统的网络配置。

3.3.1 Linux 网络相关配置文件

Linux 网络配置相关的文件根据不同的发行版目录名称有所不同，但大同小异，主要有以下目录或文件。

（1）/etc/hostname：主要功能在于修改主机名称。

（2）/etc/sysconfig/network-scrips/ifcfg-enoN：设置网卡参数的文件，比如 IP 地址、子网掩码、广播地址、网关等，N 为一串数字。

（3）/etc/resolv.conf：此文件设置了 DNS 相关的信息，用于将域名解析到 IP。

（4）/etc/hosts：计算机的 IP 对应的主机名称或域名对应的 IP 地址，通过设置 /etc/nsswitch.conf 中的选项可以选择是 DNS 解析优先还是本地设置优先。

（5）/etc/nsswitch.conf（name service switch configuration，名字服务切换配置）：规定通过哪些途径，以及按照什么顺序通过这些途径来查找特定类型的信息。

3.3.2 配置 Linux 系统的 IP 地址

要设置主机的 IP 地址，可以直接通过终端命令设置，如想设置在系统重启后依然生效，可以设置对应的网络接口文件，如【示例 3-11】所示。

【示例 3-11】

```
[root@CentOS network-scripts]# cat ifcfg-eno33554984
TYPE=Ethernet
BOOTPROTO=none
IPADDR0=192.168.146.150
PREFIX0=24
GATEWAY0=192.168.146.2
DNS1=61.139.2.69
DNS2=192.168.146.2
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
```

```
NAME=eno33554984
UUID=3af72fa2-8186-4d54-83d7-8074fe8c057c
ONBOOT=yes
```

主要字段的含义如表 3.7 所示。

表 3.7　网卡设置参数说明

参数	说明
TYPE	设备连接类型，此处为以太网
BOOTPROTO	使用动态 IP 还是静态 IP
IPADDR0	第一 IP 地址
PREFIX0	第一 IP 地址对应的子网掩码长度
GATEWAY0	第一 IP 地址对应的网关
DNS1 和 DNS2	DNS 服务器地址
DEFROUTE	是否为默认路由
ONBOOT	系统启动时是否设置此网络接口
NAME	设备名，此处对应的网络接口为 eno33554984

设置完 ifcfg-eth0 文件后，需要重启网络服务才能生效，重启后使用 ifconfig 查看设置是否生效：

```
[root@CentOS network-scripts]# service network restart
```

同一个网络接口可以使用子接口的方式设置多个 IP 地址，如【示例 3-12】所示。

【示例 3-12】

```
[root@CentOS ~]# ifconfig eno33554984:2 192.168.146.152 netmask 255.255.255.0
up
[root@CentOS network-scripts]# ifconfig
eno33554984: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.146.150  netmask 255.255.255.0  broadcast 192.168.146.255
        inet6 fe80::20c:29ff:fe0b:780  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:0b:07:80  txqueuelen 1000  (Ethernet)
        RX packets 6453  bytes 6525511 (6.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2023  bytes 167541 (163.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eno33554984:2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.146.152  netmask 255.255.255.0  broadcast 192.168.146.255
        ether 00:0c:29:0b:07:80  txqueuelen 1000  (Ethernet)
```

当服务器重启或网络服务重启后，子接口配置将消失，如需重启后依然生效，可以将配置子接口命令加入/etc/rc.local 文件中。

3.3.3　设置主机名

主机名是识别某个计算机在网络中的标识。设置主机名，使用 hostname 命令即可。在单机情况下主机名可任意设置，如以下命令在重新登录后发现主机名已经改变。

```
[root@CentOS network-scripts]# hostname www.example.com
```

若要修改重启后依然生效，可以将主机名写入文件/etc/hostname 中，如【示例 3-13】所示。

【示例 3-13】

```
[root@www ~]# hostname
www.example.com
```

3.3.4　设置默认网关

设置好 IP 地址以后，如果要访问其他的子网或 Internet，用户还需要设置路由，在此不做介绍，这里采用设置默认网关的方法。在 Linux 中，设置默认网关有两种方法：

（1）第 1 种方法就是直接使用 route 命令。在设置默认网关之前，先用 route －n 命令查看路由表。可执行如下命令设置网关：

```
[root@CenOS /]# route add default gw 192.168.1.254
```

如果不想每次开机都执行 route 命令，可把要执行的命令写入/etc/rc.d/rc.local 文件中。

（2）第 2 种方法是在/etc/sysconfig/network-scripts/ifcfg-接口文件中添加如下字段：

```
GATEWAY=192.168.10.254
```

同样，只要是更改了脚本文件，就必须重启网络服务来使设置生效，可执行下面的命令：

```
[root@CentOS /]# service network restart
```

使用 service 命令时需要注意，由于 CentOS 7.5 中使用的是 systemd，因此开启和停止服务通常使用 systemctl 代替，但也可以使用 service。

使用命令方式配置默认路由通常适用于临时测试。

3.3.5　设置 DNS 服务器

设置 DNS 服务器，修改/etc/resolv.conf 文件即可。下面是一个 resolv.conf 文件的示例。

【示例 3-14】

```
[root@CentOS ~]#  cat /etc/resolv.conf
nameserver  192.168.3.1
nameserver  192.168.3.2
options rotate
```

```
options timeout:1 attempts:2
```

其中，192.168.3.1 为第一名字服务器，192.168.3.2 为第二名字服务器，options rotate 选项指在这 2 个 dns server 之间轮询，option timeout:1 表示解析超时时间 1s（默认为 5s），attempts 表示解析域名尝试的次数。如需添加 DNS 服务器，可直接修改此文件。需要注意的是，使用 nameserver 指定的 DNS 服务器只有前三条生效。

3.4 Linux 默认防火墙 firewalld

在 CentOS 7.5 中，系统安装了两个防火墙：firewalld 和 iptables，默认使用 firewalld。本节我们先介绍 firewalld 的特性和使用方法，下一节再介绍 iptables。

3.4.1 firewalld 特性

1. firewalld 和 iptables 的关系

firewalld 仅仅是替代了 iptables service 部分，底层还是使用 iptables 作为防火墙规则管理入口。firewalld 使用 Python 语言编写，提供了一个 daemon 和 service，还提供命令行和图形界面配置工具。

2. 静态防火墙和动态防火墙

iptables 一般称为静态防火墙，也就是即使只修改一条规则也要把所有规则都重新载入，这样做会在修改规则的时候对整个系统的网络造成影响。firewalld 一般称为动态防火墙，它的出现就是为了解决 iptables 的这一问题，任何规则的变更都只需要将变更部分保存并更新到运行中的 iptables 就可以了，这样也不会影响到整个系统的网络，只会影响到使用规则的应用的网络。

3. 采用 xml 作为配置文件

firewalld 的配置文件被放置在不同的 xml 文件当中，这使得对规则的维护变得更加容易和可读、有条理。相比于 iptables 的规则配置文件而言，这显然可以算作是一个进步。

4. 区域模型定义

有点类似 Windows 的防火墙，firewalld 通过对 iptables 自定义链的使用，抽象出一个区域模型的概念。针对各种规则统一成一套默认的标准使用规范和流程，使得防火墙在易用性和通用性上得到提升。

3.4.2　firewalld 基本术语

1. 网络区域 zone

网络区域 zone 定义了网络连接的可信等级。这是一个一对多的关系，这意味着一次连接可以仅仅是一个区域的一部分，而一个区域可以用于很多连接。firewalld 将网卡对应到不同的区域（zone）。zone 默认共有 9 个：block、dmz、drop、external、home、internal、public、trusted、work。不同区域之间的差异，是其对待数据包的默认行为不同。

根据区域名字，我们可以很直观地知道该区域的特征。在 CentOS 7.5 系统中，默认区域被设置为 public。

所有可用 zone 的 xml 配置文件被保存在/usr/lib/firewalld/zones/目录，该目录中的配置为默认配置，不允许管理员手动修改。

自定义 zone 配置需保存到/etc/firewalld/zones/目录。

2. 服务 service

服务 service 是端口和/或协议入口的组合。服务所使用的 TCP/UDP 端口的配置文件存放在 /usr/lib/firewalld/services/目录中，如 SSH 服务等。最新版本的 firewalld 中默认已经定义了 70 多种服务供我们使用。

当默认提供的服务不够用或者需要自定义某项服务的端口时，需要将 service 配置文件放置在/etc/firewalld/services/目录中。

service 配置的优势：

- 通过服务名字来管理规则更加人性化。
- 通过服务来组织端口分组的模式更加高效，如果一个服务使用了若干个网络端口，则服务的配置文件就相当于提供了到这些端口的规则管理的批量操作快捷方式。每加载一项 service 配置就意味着开放了对应的端口访问。

3.4.3　firewalld 使用

1. 管理 firewalld 服务

```
#设置开机启动服务
[root@CentOS ~]# systemctl enable firewalld.service
#设置开机禁用服务
[root@CentOS ~]# systemctl disable firewalld.service
#开启服务
[root@CentOS ~]# systemctl start firewalld.service
#停止服务
[root@CentOS ~]# systemctl stop firewalld.service
#查看 firewalld 状态
[root@CentOS ~]# systemctl status firewalld
# 查看版本
```

```
[root@CentOS ~]# firewall-cmd --version
# 查看帮助
[root@CentOS ~]# firewall-cmd --help
# 显示状态
[root@CentOS ~]# firewall-cmd --state
```

2. zone 管理

```
#列出所有支持的 zone
[root@CentOS ~]# firewall-cmd --get-zones
#查看当前的默认 zone
[root@CentOS ~]# firewall-cmd --get-default-zone
```

3. service 管理

```
# 分别列出所有支持的 service
[root@CentOS ~]# firewall-cmd --get-services

#查看当前 zone 中加载的 service
[root@CentOS ~]# firewall-cmd --list-services
```

4. 基本操作

```
# 查看所有打开的端口
[root@CentOS ~]# firewall-cmd --zone=public --list-ports
# 更新防火墙规则
[root@CentOS ~]# firewall-cmd --reload
# 查看区域信息
[root@CentOS ~]# firewall-cmd --get-active-zones
# 查看指定接口所属区域
[root@CentOS ~]# firewall-cmd --get-zone-of-interface=eth0
# 拒绝所有包
[root@CentOS ~]# firewall-cmd --panic-on
# 取消拒绝状态
[root@CentOS ~]# firewall-cmd --panic-off
# 查看是否拒绝
[root@CentOS ~]# firewall-cmd --query-panic
```

5. 添加端口

```
# 添加80端口
[root@CentOS ~]# firewall-cmd --zone=public --add-port=80/tcp --permanent
（--permanent 永久生效，没有此参数，重启后失效）
# 重新载入
```

```
[root@CentOS ~]# firewall-cmd --reload
# 查看80端口
[root@CentOS ~]# firewall-cmd --zone= public --query-port=80/tcp
# 删除端口
[root@CentOS ~]# firewall-cmd --zone= public --remove-port=80/tcp --permanent

# 为 public 区添加持久性的 HTTP 和 HTTPS 规则
[root@CentOS ~]# firewall-cmd --zone=public --add-service=http --permanent
[root@CentOS ~]# firewall-cmd --zone=public --add-service=https --permanent

# 列出所有的规则
[root@CentOS ~]# firewall-cmd -list-all
```

3.5 Linux 防火墙 iptables

在 CentOS 7 之前的版本均使用 iptables，因此本书也介绍 iptables 的使用。要使用 iptables 防火墙，必须要先了解 TCP/IP 网络的基本原理，理解 Linux 防火墙的工作原理。本节主要介绍 Linux 防火墙方面的知识。

3.5.1 Linux 内核防火墙的工作原理

Linux 内核提供的防火墙功能通过 netfiter 框架实现，并提供了 iptables 工具配置和修改防火墙的规则。

netfilter 的通用框架不依赖于具体的协议，而是为每种网络协议定义一套钩子函数。这些钩子函数在数据包经过协议栈的几个关键点时被调用，在这几个点中，协议栈将数据包及钩子函数作为参数，传递给 netfilter 框架。

对于每种网络协议定义的钩子函数，任何内核模块都可以对每种协议的一个或多个钩子函数进行注册，实现挂接。这样当某个数据包被传递给 netfilter 框架时，内核能检测到是否有有关模块对该协议和钩子函数进行了注册。若发现注册信息则调用该模块在注册时使用的回调函数，然后对应模块去检查、修改、丢弃该数据包及指示 netfilter 将该数据包传入用户空间的队列。

从以上描述可以得知钩子提供了一种方便的机制，以便在数据包通过 Linux 内核的不同位置上截获和操作处理数据包。

1. netfilter 的体系结构

网络数据包的通信主要经过以下相关步骤对应 netfilter 定义的钩子函数，更多信息可以参考源代码。

- NF_IP_PRE_ROUTING：网络数据包进入系统，经过了简单的检测后，数据包转交给该函数进行处理，然后根据系统设置的规则对数据包进行处理，如果数据包不被丢弃

就交给路由函数进行处理。在该函数中可以替换 IP 包的目的地址，即 DNAT。

- NF_IP_LOCAL_IN：所有发送给本机的数据包都要通过该函数的处理，该函数根据系统设置的规则对数据包进行处理，如果数据包不被丢弃就交给本地的应用程序。
- NF_IP_FORWARD：所有不是发送给本机的数据包都要通过该函数进行处理，该函数会根据系统设置的规则对数据包进行处理，若数据包不被丢弃则转到 NF_IP_POST_ROUTING 进行处理。
- NF_IP_LOCAL_OUT：所有从本地应用程序出来的数据包必须通过该函数的处理，该函数根据系统设置的规则对数据包进行处理，如果数据包不被丢弃就交给路由函数进行处理。
- NF_IP_POST_ROUTING：所有数据包在发给其他主机之前需要通过该函数的处理，该函数根据系统设置的规则对数据包进行处理，如果数据包不被丢弃，就将数据包发给数据链路层。在该函数中可以替换 IP 包的源地址，即 SNAT。

图 3.3 显示了数据包在通过 Linux 防火墙时的处理过程。

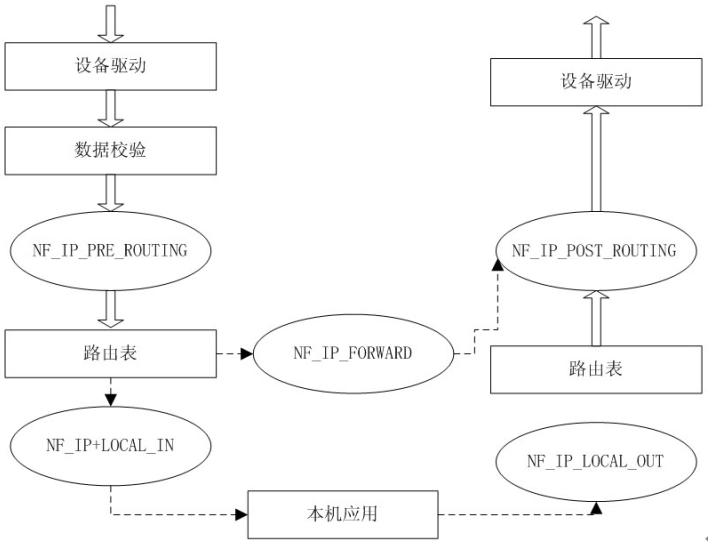

图 3.3　数据包在通过 Linux 防火墙时的处理过程

2. 包过滤

每个函数都可以对数据包进行处理，最基本的操作为对数据包进行过滤。系统管理员可以通过 iptables 工具来向内核模块注册多个过滤规则，并且指明过滤规则的优先权。设置完以后每个钩子按照规则进行匹配，如果与规则匹配，函数就会进行一些过滤操作，这些操作主要是以下几个：

- NF_ACCEPT：继续正常地传递包。
- NF_DROP：丢弃包，停止传送。
- NF_STOLEN：已经接管了包，不要继续传送。
- NF_QUEUE：排列包。
- NF_REPEAT：再次使用该钩子。

3. 包选择

在 netfilter 框架上已经创建了一个包选择系统，这个包选择工具默认已经注册了 3 个表，分别是过滤（filter）表、网络地址转换（NAT）表和 mangle 表。

在调用钩子函数时是按照表的顺序来调用的。例如，在执行 NF_IP_PRE_ROUTING 时，首先检查 filter 表，然后检查 Mangle 表，最后检查 NAT 表。

过滤（filter）表过滤包而不会改变包，仅仅是过滤的作用，实际上由网络过滤框架来提供 NF_IP_FORWARD 钩子的输出和输入接口使得很多过滤工作变得非常简单。从图 3.3 中可以看出，NF_IP_LOCAL_IN 和 NF_IP_LOCAL_OUT 也可以做过滤，但是只是针对本机。

网络地址转换（NAT）表分别服务于两套不同的网络过滤挂钩的包，对于非本地包，NF_IP_PRE_ROUTING 和 NF_IP_POST_ROUTING 挂钩可以完美地解决源地址和目的地址的变更。

NAT 表与 filter 表的区别在于只有新建连接的第 1 个包会在表中传送，结果将被用于以后所有来自这一连接的包。例如，某一个连接的第 1 个数据包在这个表中被替换了源地址，那么以后这条连接的所有包都将被替换源地址。

mangle 表被用于真正的改变包的信息，mangle 表和所有的 5 个网络过滤的钩子函数都有关。

4. 切换至 iptables

在正式学习 iptables 之前，还需要将默认使用的 firewalld 停止，并让系统将 iptables 作为默认防火墙。其命令如下：

```
#关闭并禁用 firewalld
[root@CentOS ~]# systemctl stop firewalld
[root@CentOS ~]# systemctl disable firewalld
#启动并启用 iptables
[root@CentOS ~]# systemctl start iptables
[root@CentOS ~]# systemctl enable iptables
#如果使用了 IPv6，还需要开启 ip6tables
#启动并启用 ip6tables
[root@CentOS ~]# systemctl start ip6tables
[root@CentOS ~]# systemctl enable ip6tables
```

使用上述命令之后就可以在 CentOS 7.5 中使用 iptables 了。

3.5.2　Linux 软件防火墙 iptables

iptables 工具用来设置、维护和检查 Linux 内核的 IP 包过滤规则。filter、NAT 和 mangle 表可以包含多个链（chain），每个链可以包含多条规则（rule）。iptables 主要对表（table）、链（chain）和规则（rule）进行管理。

iptables 预定义了 5 个链，分别对应 netfilter 的 5 个钩子函数。这 5 个链分别是 INPUT 链、FORWARD 链、OUTPUT 链、PREROUTING 链、POSTROUTING 链。

iptables 指令语法如下：

```
iptables [-t table] command [match] [-j target/jump]
```

“-t table”参数用来指定规则表，内建的规则表分别为 nat、mangle 和 filter，当未指定规则表时，默认为 filter。各个规则表的功能如下：

- nat：此规则表主要针对 PREROUTING 和 POSTROUTING 两个规则链，主要功能为进行源地址或目的地址的网址转换工作（SNAT、DNAT）。
- mangle：此规则表主要针对 PREROUTING、FORWARD 和 POSTROUTING 3 个规则链，某些特殊应用可以在此规则表里设定，比如为数据包做标记。
- filter：这个规则表是默认规则表，针对 INPUT、FORWARD 和 OUTPUT 3 个规则链，这个规则表主要用来进行封包过滤的处理动作，如 DROP、LOG、ACCEPT 或 REJECT。

iptables 的简单用法如【示例 3-15】所示。

【示例 3-15】

```
[root@CentOS ~]# iptables -t filter -A FORWARD -s 192.168.19.0/24 -j DROP
[root@CentOS ~]# iptables  -nL
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.19.0/24      0.0.0.0/0
```

其中，“-t filter”表示该规则作用于 filter 表，“-A”表示新增规则，“-s”表示 IP 段选项，“-j”表示指定动作。该规则表示在 filter 表 FORWARD 链上新增一条规则，发往 192.168.19.0/24 网段的包采取丢弃操作。要查看某个表下各个链的信息，可以使用“iptables -nL”。

要使 Linux 系统成为网络防火墙，当然除了内核支持之外，还需要启用 Linux 的网络转发功能。

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

若需要在系统启动时就具有该功能，则可将上面的命令写入/etc/rc.d/rc.local 文件中。

数据包通过表和链时需要遵循一定的顺序，当数据包到达防火墙时，如果 MAC 地址符合，就会由内核里相应的驱动程序接收，然后经过一系列操作，从而决定是发送给本地的程序还是转发给其他机器。数据包通过防火墙时分以下 3 种情况。

1. 以本地为目标的包

当一个数据包进入防火墙后，如果目的地址是本机，被防火墙进行检查的顺序如表 3.8 所示。如果在某一个步骤数据包被丢弃，当然就不会执行后面的检查了。

表 3.8　以本地为目的的包检查顺序

步骤	表	链	说明
1			数据包在链路上进行传输
2			数据包进入网络接口
3	mangle	PREROUTING	这个链用来 mangle 数据包，如对包进行改写或做标记
4	nat	PREROUTING	这个链主要用来做 DNAT
5			路由判断，如包是发往本地的还是要转发的
6	mangle	INPUT	在路由之后，被送往本地程序之前，对包进行改写或做标记
7	filter	INPUT	所有以本地为目的的包都需经过这个链，包的过滤规则设置在此
8			数据包到达本地程序，如服务程序或客户程序

2. 以本地为源的包

本地应用程序发出的数据包，被防火墙进行检查的顺序如表 3.9 所示。

表 3.9　以本地为源的包检查顺序

步骤	表	链	说明
1			本地程序，如服务程序或客户程序
2			路由判断
3	mangle	OUTPUT	用来 mangle 数据包，如对包进行改写或做标记
4	nat	OUTPUT	对发出的包进行 DNAT 操作
5	filter	OUTPUT	对本地发出的包进行过滤，包的过滤规则设置在此
6	mangle	POSTROUTING	进行数据包的修改
7	filter	POSTROUTING	在这里做 SNAT
8			离开网络接口
9			数据包在链路上传输

3. 被转发的包

需要通过防火墙转发的数据包，被防火墙进行检查的顺序如表 3.10 所示。

表 3.10　被转发的包检查顺序

步骤	表	链	说明
1			数据包在链路上传输
2			进入网络接口
3	mangle	PREROUTING	mangle 数据包，如对包进行改写或做标记
4	nat	PREROUTING	这个链主要用来做 DNAT
5			路由判断，如包是发往本地的，还是要转发
6	mangle	FORWARD	包继续被发送至 mangle 表的 FORWARD 链，在非常特殊的情况下才会用到。在这里，包被 mangle。这次 mangle 发生在最初的路由判断之后，在最后一次更改包的目的之前

（续表）

步骤	表	链	说明
7	filter	FORWARD	FORWARD 包继续被发送至这条 FORWARD 链。只有需要转发的包才会走到这里，并且针对这些包的所有过滤也在这里进行。注意，所有要转发的包都要经过这里
8	mangle	POSTROUTING	这个链也是针对一些特殊类型的包。这一步 mangle 是在所有更改包的目的地址的操作完成之后做的，但这时包还在本地上
9	nat	POSTROUTING	这个链就是用来做 SNAT 的，不推荐在此处过滤，因为某些包即使不满足条件也会通过
10			离开网络接口
11			数据包在链路上传输

在对包进行过滤时，常用的有以下 3 个动作。

（1）ACCEPT：一旦数据包满足了指定的匹配条件，数据包就会被 ACCEPT，并且不会再去匹配当前链中的其他规则或同一个表内的其他规则，但数据仍然需要通过其他表中的链。

（2）DROP：如果包符合条件，数据包就会被丢掉，并且不会向发送者返回任何信息，也不会向路由器返回信息。

（3）REJECT：和 DROP 基本一样，区别在于除了将包丢弃以外，还会向发送者返回错误信息。

要进一步了解各个链中规则的匹配顺序，就来学习一下 filter 表中 FORWARD 链的输出，如【示例 3-16】所示。

【示例 3-16】

```
[root@CentOS ~]# iptables -nvL
Chain FORWARD (policy DROP)
target prot opt source destination
ACCEPT all -- 192.168.100.0/24 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 69.147.0.0/24
ACCEPT all -- 172.16.0.0/16 0.0.0.0/0
mychain tcp -- 10.0.0.0/24 0.0.0.0 tcp dpt:80
```

“policy DROP”表示该链的默认规则为 DROP 操作。如现有一数据包，源地址为 192.168.1.58，目的地址为 137.254.60.6，协议为 TCP，目的端口为 80，当该数据包通过 FORWARD 链时，从上往下开始匹配：

（1）与第 1 条规则：源为 192.168.100.0/24，源不匹配。

（2）与第 2 条规则：目的为 69.147.0.0/24，目标不匹配。

（3）与第 3 条规则：源为 172.16.0.0/16，源不匹配。

（4）与第 4 条规则：源为 10.0.0.0/24，源不匹配。

经过匹配，以上所有规则都不符合，数据包将转交给默认规则处理。由于本示例中默认的

规则为 DROP，因此该数据包被丢弃。

再看另一个数据包，源地址为 192.168.1.58，目的地址为 69.147.83.199，协议为 TCP，目的端口为 80，当该数据包通过 FORWARD 链时，从上往下开始匹配：

（1）与第 1 条规则：源为 192.168.100.0/24，源不匹配。

（2）与第 2 条规则：源地址为任意（0.0.0.0/0），匹配；目的地址为 69.147.0.0/24，69.147.83.199 在范围内，匹配；源端口和目的端口为任意，匹配；协议为任意（all），匹配；规则链对该数据包的动作为 ACCEPT，因此该数据包通过。

如果数据包的源地址为 10.0.0.35，目的地址为 69.147.83.199，协议为 TCP，目的端口为 80，那么该数据包通过 FORWARD 链时，从上往下开始匹配，第 4 条规则匹配，动作为 mychain，此时数据包会被转到用户自己定义的规则链 mychain 进行处理。

3.5.3　iptables 配置实例

iptables 工具支持丰富的参数，可以对 IP 和端口、网络接口、TCP 标志位或 MAC 地址进行过滤，参数指定方式除传统方法外，可以支持“!”“ALL”或“NONE”等进行参数匹配。iptables 常用参数说明如表 3.11 所示。

表 3.11　iptables 命令参数含义说明

参数	含义
-A	新增规则到某个规则链中，该规则将会成为规则链中的最后一条规则
-D	从某个规则链中删除一条规则，可以输入完整规则，或直接指定规则编号加以删除
-R	替换某行规则，规则被替换后并不会改变顺序
-I	插入一条规则，原本该位置上的规则将会往后移动一个顺位
-L	列出某规则链中的所有规则
-F	删除规则链的所有规则
-Z	将数据包计数器归零
-N	定义新的规则链
-X	删除某个规则链
-P	定义不符合规则的数据包的默认处理方式
-E	修改某自定义规则链的名称
-p	匹配通信协议类型是否相符，可以使用!运算符进行反向匹配
-s	匹配数据包的来源 IP，可以匹配单个 IP 或某个网段
-d	匹配数据包的目的地 IP，设定方式同上
-i	匹配数据包是从哪个网络接口进入，可以使用通配字符+指定匹配范围
-o	匹配数据包要从哪个网络接口发出，设定方式同上
--sport	匹配数据包的源端口，可以匹配单一端口或一个范围
--dport	匹配数据包的目的端口号，设定方式同上
--tcp-flags	匹配 TCP 数据包的状态标志，如 SYN、ACK、FIN 等，另外可使用 ALL 和 NONE 进行匹配

（续表）

参数	含义
-m	匹配不连续的多个源端口或目的端口
-m	匹配数据包来源网络接口的 MAC 地址，不能用于 OUTPUT 和 POSTROUTING 规则链
ACCEPT	将数据包放行，进行完此处理动作后，将不再匹配其他规则，直接跳往下一个规则链
REJECT	阻塞数据包，并传送数据包通知对方
DROP	丢弃数据包不予处理，进行完此处理动作后，将不再匹配其他规则，直接中断过滤程序
REDIRECT	将数据包定向另一个端口，进行完此处理动作后，将会继续匹配其他规则
LOG	将数据包相关信息记录在/var/log 中
SNAT	改写数据包来源 IP 为某特定 IP 或 IP 范围，可以指定 port 对应的范围，进行完此处理动作后，将直接跳往下一个规则
DNAT	改写数据包目的地 IP 为某特定 IP 或 IP 范围，可以指定 port 对应的范围，进行完此处理动作后，将会直接跳往下一个规则链
RETURN	结束在目前规则链中的过滤程序，返回主规则链继续过滤
MARK	数据包做标记，以便提供作为后续过滤的条件判断依据，进行完此处理动作后，将会继续匹配其他规则

1. 简单应用示例

iptables 使用方法首先指定规则表，然后指定要执行的命令，接着指定参数匹配数据包的内容，最后是要采取的动作。下面通过一些示例来说明 iptables 的使用方法，如【示例 3-17】所示。

【示例 3-17】

```
    #清除所有规则
    [root@CentOS ~]# iptables -F
    #清除 nat 表中的所有规则
    [root@CentOS ~]# iptables  -t nat -F
    #允许来自192.168.3.0/24 连接 sshd 服务
    [root@CentOS ~]# iptables -A INPUT -p tcp -s  192.168.3.0/24  --dport 22 -j
ACCEPT
    #其他任何网段访问不能访问 sshd 服务
    [root@CentOS ~]# iptables -A INPUT -p tcp    --dport 22 -j  DROP
```

在上述示例中，“-F”表示清除已存在的所有规则，“-A”表示添加一条规则，“-p”指定协议为 TCP，“-s”指定源地址段，如果该参数忽略或为 0.0.0.0/0，则源地址表示任何地址，“-dport”指定目的端口。包的判断顺序为首先判断第 1 条规则，由于允许 192.168.3.0/24 网段的服务器访问 sshd 服务，因此包可以通过；如果是其他来源的主机，由于第 1 条规则不符合接着判断第 2 条规则，策略为禁止，因此包将被丢弃。

除以上示例外，iptables 可以为每个链指定默认规则，如果包不符合现存的所有规则，就按默认规则处理，方法如【示例 3-18】所示。

【示例 3-18】

```
    #清除所有规则
    [root@CentOS ~]# iptables -F
    #设置默认规则
    [root@CentOS ~]# iptables -t filter -P INPUT ACCEPT
    #允许来自192.168.3.0/24 连接 sshd 服务
    [root@CentOS ~]# iptables -A INPUT -p tcp -s  192.168.3.0/24  --dport 22 -j
ACCEPT
```

由于设置了默认规则，该示例的功能同示例 3-3。

上面所列举的示例仅仅是为了说明语法和原理，在实际使用中，还需要注意规则添加的顺序，使数据包通过的规则最少。

基于网络接口的过滤如【示例 3-19】所示。

【示例 3-19】

```
    [root@CentOS ~]#iptables -t filter -F
    [root@CentOS ~]#iptables -t filter -P FORWARD DROP
    [root@CentOS ~]#iptables -t filter -A FORWARD -p tcp -i eth2 -o eth1 --dport
80 -j ACCEPT
```

使用-i 参数来指定数据包的来源网络接口，使用-o 来指定数据包将从哪个网络接口出去。

在 INPUT 链中不能使用-o 选项，OUTPUT 链中不能使用-i 选项。

2. NAT 设置

通常网络中的数据包从包的源地址发出直到包要发送的目的地址，整个路径经过很多不同的连接，一般情况下这些连接不会更改数据包的内容，只是原样转发。如果发出数据包的主机使用的源地址是私用网络地址，该数据包将不能在互联网上传输。要能够使用私有网络访问互联网，就必须做 NAT（Network Address Translation，网络地址转换）。

NAT 分为两种不同的类型：源 NAT（SNAT）和目标 NAT（DNAT）。SNAT 通常用于使用了私网地址的局域网络，能够访问互联网。DNAT 是指修改包的目标地址，端口转发、负载均衡和透明代理都属于 DNAT。

在 SNAT 中改变源地址，如【示例 3-20】所示。

【示例 3-20】

```
    # SNAT 改变源地址为1.2.3.4
    [root@CentOS ~]# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
    #DNAT
    [root@CentOS ~]# iptables -t nat -A PREROUTING -p tcp -i eth2 -d 1.2.3.4 --dport
80 -j DNAT --to 192.168.3.88:80
```

如果做 SNAT 只能在 POSTROUTING 上进行，而做 DNAT 只能在 PREROUTING 内进行。

3.6 Linux 高级网络配置工具

目前很多 Linux 在使用之前的 arp、ifconfig 和 route 命令。虽然这些工具能够工作，但它们在 Linux 2.2 和更高版本的内核上显得有一些落伍。无论对于 Linux 开发者还是 Linux 系统管理员，网络程序调试时数据包的采集和分析是不可少的。tcpdump 是 Linux 中强大的数据包采集分析工具之一。本节主要介绍 iproute2 和 tcpdump 的相关知识。

3.6.1 高级网络管理工具 iproute2

相对于系统提供的 arp、ifconfig 和 route 等旧版本的命令，iproute2 工具包提供了更丰富的功能，除了提供网络参数设置、路由设置、带宽控制等功能，最新的 GRE 隧道也可以通过此工具进行配置。

现在大多数 Linux 发行版本都安装了 iproute2 软件包，如没有安装可以使用 yum 工具进行安装。应该注意的是，yum 工具需要联网才能使用。iproute2 工具包中的主要管理工具为 ip 命令。下面将介绍 iproute2 工具包的安装与使用。安装过程如【示例 3-21】所示。

【示例 3-21】

```
[root@CentOS Packages]# yum install -y iproute
#安装过程省略
[root@CentOS Packages]# rpm -qa|grep iproute
iproute-3.10.0-13.el7.x86_64
#检查安装情况
[root@CentOS Packages]# ip -V
ip utility, iproute2-ss130716
```

ip 命令的语法如【示例 3-22】所示。

【示例 3-22】

```
[root@CentOS ~]# ip help
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | addr | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddr | mroute | mrule | monitor | xfrm |
                   netns | l2tp | tcp_metrics | token }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -f[amily] { inet | inet6 | ipx | dnet | bridge | link } |
                    -4 | -6 | -I | -D | -B | -0 |
```

```
                    -l[oops] { maximum-addr-flush-attempts } |
                    -o[neline] | -t[imestamp] | -b[atch] [filename] |
                    -rc[vbuf] [size]}
```

1. 使用 ip 命令来查看网络配置

ip 命令是 iproute2 软件的命令工具，可以替代 ifconfig、route 等命令，查看网络配置的用法如【示例 3-23】所示。

【示例 3-23】

```
   #显示当前网卡参数，同 ifconfig
   [root@CentOS ~]# ip addr list
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
      link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
      inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
   2: eno16777736: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
      link/ether 00:0c:29:0b:07:76 brd ff:ff:ff:ff:ff:ff
      inet 192.168.128.133/24 brd 192.168.128.255 scope global dynamic eno16777736
        valid_lft 1149sec preferred_lft 1149sec
      inet6 fe80::20c:29ff:fe0b:776/64 scope link
        valid_lft forever preferred_lft forever
   3: eno33554984: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
      link/ether 00:0c:29:0b:07:80 brd ff:ff:ff:ff:ff:ff
      inet 192.168.146.150/24 brd 192.168.146.255 scope global eno33554984
        valid_lft forever preferred_lft forever
      inet6 fe80::20c:29ff:fe0b:780/64 scope link
        valid_lft forever preferred_lft forever
   #添加新的网络地址
   [root@CentOS ~]# ip addr add 192.168.128.140/24 dev eno16777736
   [root@CentOS ~]# ip addr list
   #部分结果省略
   4: eno16777736: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
      link/ether 00:0c:29:0b:07:76 brd ff:ff:ff:ff:ff:ff
      inet 192.168.128.133/24 brd 192.168.128.255 scope global dynamic eno16777736
        valid_lft 1776sec preferred_lft 1776sec
      inet 192.168.128.140/24 scope global secondary eno16777736
        valid_lft forever preferred_lft forever
      inet6 fe80::20c:29ff:fe0b:776/64 scope link
        valid_lft forever preferred_lft forever
   #删除网络地址
   [root@CentOS ~]# ip addr del 192.168.3.123/24 dev eth0
```

上面的命令显示了机器上所有的地址，以及这些地址属于哪些网络接口。“inet”表示

Internet (IPv4)。eth0 的 IP 地址与 192.168.3.88/24 相关联，“/24”指 IP 地址表示网络地址的位数，“lo”则为本地回路信息。

2. 显示路由信息

如需查看路由信息，可以使用“ip route list”命令，如【示例 3-24】所示。

【示例 3-24】

```
#查看路由情况
[root@CentOS ~]# ip route list
default via 192.168.146.2 dev eno33554984  proto static  metric 1024
192.168.128.0/24 dev eno16777736  proto kernel  scope link  src 192.168.128.133
192.168.146.0/24 dev eno33554984  proto kernel  scope link  src 192.168.146.150
[root@CentOS ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.146.2   0.0.0.0         UG    1024   0        0 eno33554984
192.168.128.0   0.0.0.0         255.255.255.0   U     0      0        0 eno16777736
192.168.146.0   0.0.0.0         255.255.255.0   U     0      0        0 eno33554984
#添加路由
[root@CentOS ~]# ip route add  192.168.3.1 dev eno33554984
```

上述示例首先查看系统中当前的路由情况，其功能和 route 命令类似。

以上只是初步介绍了 iproute2 的用法，更多信息请查看系统帮助。

3.6.2 网络数据采集与分析工具 tcpdump

tcpdump 即 dump traffic on a network，是根据使用者的定义对网络上的数据包进行截获的包分析工具。无论对于网络开发者还是系统管理员，数据包的获取与分析都是最重要的技术之一。对于系统管理员来说，在网络性能急剧下降的时候，可以通过 tcpdump 工具分析原因，找出造成网络阻塞的来源。对于程序开发者来说，可以通过 tcpdump 工具来调试程序。tcpdump 支持针对网络层、协议、主机、网络或端口的过滤，并提供 and、or、not 等逻辑语句过滤不必要的信息。

Linux 系统下 tcpdump 普通用户是不能正常执行的，一般通过 root 用户执行。

tcpdump 采用命令行方式，命令格式如下，参数说明如表 3.12 所示。

```
tcpdump [ -adeflnNOpqStvx ] [ -c 数量 ] [ -F 文件名 ]
      [ -i 网络接口 ] [ -r 文件名] [ -s snaplen ]
       [ -T 类型 ] [ -w 文件名 ] [表达式 ]
```

表 3.12 tcpdump 命令参数含义说明

参数	含义
-A	以 ASCII 码方式显示每一个数据包，在程序调试时可方便查看数据
-a	将网络地址和广播地址转变成名字

（续表）

参数	含义
-c	tcpdump 将在接收到指定数目的数据包后退出
-d	将匹配信息包的代码以人们能够理解的汇编格式给出
-dd	将匹配信息包的代码以 C 语言程序段的格式给出
-ddd	将匹配信息包的代码以十进制的形式给出
-e	在输出行打印出数据链路层的头部信息
-f	将外部的 Internet 地址以数字的形式打印出来
-F	使用文件作为过滤条件表达式的输入，此时命令行上的输入将被忽略
-i	指定监听的网络接口
-l	使标准输出变为缓冲行形式
-n	不把网络地址转换成名字
-N	不打印出 host 的域名部分
-q	打印很少的协议相关信息，输出行都比较简短
-r	从文件 file 中读取包数据
-s	设置 tcpdump 的数据包抓取长度，如果不设置默认为 68 字节
-t	在输出的每一行不打印时间戳
-tt	不对每行输出的时间进行格式处理
-ttt	tcpdump 输出时，每两行打印之间会延迟一个时间段，以 ms 为单位
-tttt	在每行打印的时间戳之前添加日期的打印
-v	输出一个稍微详细的信息，例如在 ip 包中可以包括 ttl 和服务类型的信息
-vv	输出详细的报文信息
-vvv	产生比-vv 更详细的输出
-x	当分析和打印时， tcpdump 会打印每个包的头部数据，同时会以十六进制打印出每个包的数据，但不包括连接层的头部
-xx	tcpdump 会打印每个包的头部数据，同时会以十六进制打印出每个包的数据，其中包括数据链路层的头部
-X	tcpdump 会打印每个包的头部数据，同时会以十六进制和 ASCII 码形式打印出每个包的数据，但不包括连接层的头部
-XX	tcpdump 会打印每个包的头部数据，同时会以十六进制和 ASCII 码形式打印出每个包的数据，其中包括数据链路层的头部

首先确认本机 tcpdump 是否安装，如果没有安装，可以使用【示例 3-25】中的方法安装。

【示例 3-25】

```
#安装 tcpdump
[root@CentOS Packages]# yum install -y tcpdump
#安装过程省略
```

tcpdump 最简单的使用方法如【示例 3-26】所示。

【示例 3-26】

```
[root@CentOS Packages]# tcpdump  -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes
15:47:05.143823 IP 192.168.146.150.SSH > 192.168.146.1.52161: Flags [P.], seq
1017381117:1017381313, ack 1398930582, win 140, length 196
15:47:05.144050 IP 192.168.146.1.52161 > 192.168.146.150.SSH: Flags [.], ack
196, win 16169, length 0
15:47:06.148824 IP 192.168.146.150.56971 > ns.sc.cninfo.net.domain: 29605+PTR?
1.146.168.192.in-addr.arpa. (44)
15:47:06.158878 IP ns.sc.cninfo.net.domain > 192.168.146.150.56971: 29605
NXDomain 0/0/0 (44)
#部分结果省略，按下 Ctrl+C 中止输出
```

以上示例演示了 tcpdump 最简单的使用方式，如不跟任何参数，tcpdump 会从系统接口列表中搜寻编号最小的已配置好的接口，不包括 loopback 接口，一旦找到第 1 个符合条件的接口，搜寻马上结束，并将获取的数据包打印出来。

tcpdump 利用表达式作为过滤数据包的条件，表达式可以是正则表达式。如果数据包符合表达式，则数据包被截获；如果没有给出任何条件，则接口上所有的信息包将会被截获。

表达式中一般有如下几种关键字：

（1）第 1 种是关于类型的关键字，如 host、net 和 port。例如 host 192.168.16.150 指明 192.168.16.150 为一台主机，而 net 192.168.16.150 则表示 192.168.16.150 为一个网络地址。如果没有指定类型，默认的类型是 host。

（2）第 2 种是确定数据包传输方向的关键字，包含 src、dst、dst or src 和 dst and src，这些关键字指明了数据包的传输方向。例如 src 192.168.16.150 指明数据包中的源地址是 192.168.16.150，而 dst 192.168.16.150 则指明数据包中的目的地址是 192.168.16.150。如果没有指明方向关键字，则默认是 src or dst 关键字。

（3）第 3 种是协议的关键字，如指明是 TCP 还是 UDP 协议。

除了这 3 种类型的关键字之外，还有 3 种逻辑运算，非运算是“not”或“!”，与运算是“and”或“&&”，或运算是“or”或“||”。通过这些关键字的组合可以实现复杂强大的条件。接下来看一个组合，如【示例 3-27】所示。

【示例 3-27】

```
[root@CentOS ~]# tcpdump -i any  tcp and  dst host  192.168.19.101  and  dst
port 3306  -s100  -XX  -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 100 bytes
16:08:05.539893 IP 192.168.19.101.49702 > 192.168.19.101.mysql: Flags [P.], seq
79:108, ack 158, win 1024, options [nop,nop,TS val 17107592 ecr 17107591], length
29
        0x0000:  0000 0304 0006 0000 0000 0000 0000 0800  ................
```

```
0x0010:  4508 0051 ffe8 4000 4006 929b c0a8 1365  E..Q..@.@......e
0x0020:  c0a8 1365 c226 0cea 32aa f5e0 c46e c925  ...e.&..2....n.%
0x0030:  8018 0400 a85e 0000 0101 080a 0105 0a88  .....^..........
0x0040:  0105 0a87 1900 0000 0373 656c 6563 7420  .........select.
0x0050:  2a20 6672 6f6d 206d 7973 716c            *.from.mysql
```

以上 tcpdump 表示抓取发往本机 3306 端口的请求。“-i any”表示截获本机所有网络接口的数据报，“tcp”表示 TCP 协议，“dst　host”表示数据包地址为 192.168.19.101，“dst port”表示目的地址为 3306，“-XX”表示同时会以十六进制和 ASCII 码形式打印出每个包的数据，“-s100”表示设置 tcpdump 的数据包抓取长度为 100 个字节，如果不设置默认为 68 字节，“-n”表示不对地址（如主机地址或端口号）进行数字表示到名字表示的转换。输出部分“16:08:05”表示时间，然后是发起请求的源 IP 端口、目的 IP 和端口，“Flags[P.]”是 TCP 包中的标志信息（S 是 SYN 标志，F 表示 FIN，P 表示 PUSH，R 表示 RST，“.”表示没有标记，详细说明可进一步参考 TCP 各种状态之间的转换规则）。

3.7 动态主机配置协议（DHCP）

使用动态主机配置协议（Dynamic Host Configuration Protocol，DHCP）可以避免网络参数变化后一些烦琐的配置，客户端可以从 DHCP 服务端检索相关信息并完成相关网络配置，在系统重启后依然可以工作。DHCP 基于 C/S 模式，主要用于大型网络。DHCP 提供一种动态指定 IP 地址和相关网络配置参数的机制。本节主要介绍 DHCP 的工作原理及 DHCP 服务端与 DHCP 客户端的部署过程。

3.7.1　DHCP 的工作原理

动态主机配置协议（DHCP）是用来自动给客户端分配 TCP/IP 信息的网络协议，如 IP 地址、网关、子网掩码等信息。每个 DHCP 客户端通过广播连接到区域内的 DHCP 服务器，该服务器会响应请求，返回 IP 地址、网关和其他网络配置信息。DHCP 的请求过程如图 3.4 所示。

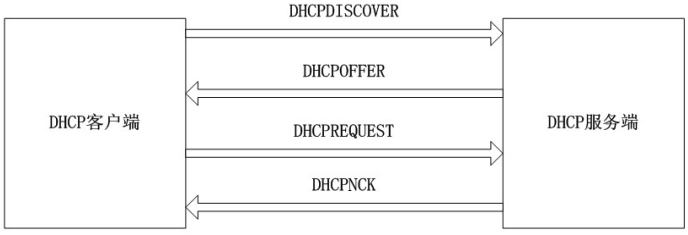

图 3.4　DHCP 请求过程

客户端请求 IP 地址和配置参数的过程有以下几个步骤：

步骤 01 客户端需要寻求网络 IP 地址和其他网络参数，然后向网络中广播，客户端发出的请求名称叫 DHCPDISCOVER。如广播网络中有可以分配 IP 地址的服务器，服务器会返回响应应答，告诉客户端可以分配，服务器返回包的名称叫 DHCPOFFER，包内包含可用的 IP 地址和参数。

步骤 02 如果客户在发出 DHCPOFFER 包后一段时间内没有接收到响应，会重新发送请求，如广播区域内有多于一台的 DHCP 服务器，由客户端决定使用哪个。

步骤 03 当客户端选定了某个目标服务器后，会广播 DHCPREQUEST 包，用以通知选定的 DHCP 服务器和未选定的 DHCP 服务器。

步骤 04 服务端收到 DHCPREQUEST 后会检查收到的包，如果包内的地址和所提供的地址一致，证明现在客户端接收的是自己提供的地址；如果不是，就说明自己提供的地址未被采纳。如被选定的服务器在接收到 DHCPREQUEST 包以后，因为某些原因可能不能向客户端提供这个 IP 地址或参数，可以向客户端发送 DHCPNAK 包。

步骤 05 客户端在收到包后，检查内部的 IP 地址和租用时间，如果发现有问题，就发包拒绝这个地址，然后重新发送 DHCPDISCOVER 包。如果没有问题，就接受这个配置参数。

3.7.2 配置 DHCP 服务器

本节主要介绍 DHCP 服务器的配置过程，包含安装、配置文件设置、服务器启动等步骤。

1. 软件安装

DHCP 服务依赖的软件可以从 rpm 包安装或从源码进行安装，本节以 yum 工具为例说明 DHCP 服务的安装过程，如【示例 3-28】所示。

【示例 3-28】

```
#确认当前系统是否安装相应软件包
[root@CentOS ~]# rpm -qa|grep dhcp
#如以上命令无输出，就说明没有安装 dhcp
#如使用 rpm 安装，就使用如下命令
[root@CentOS Packages]# yum install -y dhcp
Loaded plugins: fastestmirror, langpacks
base                                                    | 3.6 kB     00:00
extras                                                  | 3.4 kB     00:00
updates                                                 | 3.4 kB     00:00
Loading mirror speeds from cached hostfile
 * base: mirrors.yun-idc.com
 * extras: mirrors.pubyun.com
 * updates: mirrors.yun-idc.com
……
```

经过上面的设置，DHCP 服务已经安装完毕，主要的文件如下：

- /etc/dhcp/dhcpd.conf：DHCP 主配置文件。

- /usr/lib/systemd/system/dhcpd.service：DHCP 服务单元。

2. 编辑配置文件/etc/dhcp/dhcpd.conf

要配置 DHCP 服务器，需修改配置文件/etc/dhcp/dhcpd.conf。如果不存在就创建该文件。本示例实现的功能为当前网络内的服务器分配指定 IP 段的 IP 地址，并设置过期时间为 2 天。配置文件如【示例 3-29】所示。

【示例 3-29】

```
[root@CentOS Packages]# cat  -n /etc/dhcp/dhcpd.conf
      #指定接收 DHCP 请求的网卡的子网地址，注意不是本机的 IP 地址。netmask 为子网掩码
    1  subnet  192.168.19.0  netmask 255.255.255.0{
       #指定默认网关
    2  option routers 192.168.19.1;
       #指定默认子网掩码
    3  option subnet-mask 255.255.255.0;
       #指定最大租用周期
    4  max-lease-time 172800 ;
       #此 DHCP 服务分配的 IP 地址范围
    5  range 192.168.19.230 192.168.19.240;
    6  }
```

以上示例文件列出了一个子网的声明，包括 routers 默认网关、subnet-mask 默认子网掩码和 max-lease-time 最大租用周期，单位是秒。需要特别说明的是，在本地需有一个网络接口的 IP 地址为 192.168.19.0 网络，DHCP 服务才能启动。

配置文件的更多选项可以使用命令“man dhcpd.conf”获取更多帮助信息。

【示例 3-30】

```
[root@CentOS Packages]# systemctl start dhcpd.service
```

如启动失败可以参考屏幕输出定位错误内容，或查看/var/log/messages 的内容，然后参考 dhcpd.conf 的帮助文档。

3.7.3　配置 DHCP 客户端

当服务端启动成功后，客户端需要与服务端网络联通，然后做以下配置以便自动获取 IP 地址。客户端网卡配置如【示例 3-31】所示。

【示例 3-31】

```
[root@CentOS ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
HWADDR=00:0c:29:be:db:d5
TYPE=Ethernet
UUID=363f47a9-dfb8-4c5a-bedf-3f060cf99eab
```

```
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp
```

如需使用 DHCP 服务， BOOTPROTO=dhcp 表示将当前主机的网络 IP 地址设置为自动获取方式。需要说明的是 DHCP 客户端无须使用 CentOS 7.5，使用其他版本的 Linux 或 Windows 操作系统均可，在本例中使用的是 CentOS 6 作为客户端。测试过程如【示例 3-32】所示。

【示例 3-32】

```
[root@CentOS ~]# service network restart
#启动成功后确认成功获取到指定 IP 段的 IP 地址。
[root@CentOS ~]# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:0C:29:BE:DB:D5
          inet addr:192.168.19.230  Bcast:192.168.19.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febe:dbd5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:573 errors:0 dropped:0 overruns:0 frame:0
          TX packets:482 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59482 (58.0 KiB)  TX bytes:67044 (65.4 KiB)
```

客户端配置为自动获取 IP 地址，然后重启网络接口，启动成功后使用 ifconfig 查看成功获取到 IP 地址。

本节介绍了 DHCP 的基本功能，如需了解 DHCP 其他更多的功能，可参考 DHCP 的帮助文档或其他资料。

3.8 Linux 域名服务 DNS

如今互联网应用越来越丰富，仅仅用 IP 地址标识网络上的计算机是不可能完成任务的，而且也没有必要，于是产生了域名系统。域名系统通过一系列有意义的名称标识网络上的计算机，用户按域名请求某个网络服务时，域名系统负责将其解析为对应的 IP 地址，这便是 DNS。本节将详细介绍有关 DNS 的一些知识。

3.8.1 DNS 简介

目前提供网络服务的应用使用唯一的 32 位的 IP 地址来标识，但由于数字比较复杂、难以记忆，因此产生了域名系统。通过域名系统，可以使用易于理解和形象的字符串名称来标识网络应用。访问互联网应用可以使用域名，也可以通过 IP 地址直接访问该应用。在使用域名访问网络应用时，DNS 负责将其解析为 IP 地址。

DNS 是一个分布式数据库系统，扩充性好，由于是分布式的存储，数据量的增长并不会影响其性能。新加入的网络应用可以由 DNS 负责将新主机的信息传播到网络中的其他部分。

域名查询有两种常用的方式：递归查询和迭代查询。

递归查询由最初的域名服务器代替客户端进行域名查询。若该域名服务器不能直接回答，则会在域中各分支的上下进行递归查询，最终将返回查询结果给客户端，在域名服务器查询期间，客户端将完全处于等待状态。

迭代查询每次由客户端发起请求，若请求的域名服务器能提供需要查询的信息则返回主机地址信息。若不能提供，则引导客户端到其他域名服务器查询。

以上两种方式类似需要寻找东西的过程，一种是找个人替自己寻找，另外一种是自己完成，首先到一个地方寻找，若没有则向另外一个地方寻找。

DNS 域名服务器的分类有高速缓存服务器、主 DNS 服务器和辅助 DNS 服务器。高速缓存服务器将每次域名查询的结果缓存到本机，主 DNS 服务器则提供特定域的权威信息，是可信赖的，辅助 DNS 服务器信息则来源于主 DNS 服务器。

3.8.2　DNS 服务器配置

目前网络上的域名服务系统使用最多的为 BIND（Berkeley Internet Name Domain）软件，该软件实现了 DNS 协议。本节主要介绍 DNS 服务器的配置过程，包含安装、配置文件设置、服务器启动等步骤。

1. 软件安装

DNS 服务依赖的软件可以从 rpm 包安装或从源码进行安装，本节以 rpm 包为例说明 DNS 服务的安装过程，如【示例 3-33】所示。

【示例 3-33】

```
   #确认系统中相关的软件是否已经安装
   [root@CentOS Packages]# yum install -y bind bind-utils
   Loaded plugins: fastestmirror, langpacks
   base                                                | 3.6 kB     00:00
   extras                                              | 3.4 kB     00:00
   updates                                             | 3.4 kB     00:00
   Loading mirror speeds from cached hostfile
    * base: mirrors.yun-idc.com
    * extras: mirrors.sina.cn
    * updates: mirrors.sina.cn
   Package 32:bind-utils-9.9.4-14.el7_0.1.x86_64 already installed and latest
version
   ……
```

经过上面的设置，DNS 服务已经安装完毕，主要的文件如下：

- /etc/named.conf：DNS 主配置文件。
- /usr/lib/systemd/system/named.service：DNS 服务控制单元。

2. 编辑配置文件/etc/named.conf

要配置 DNS 服务器，需修改配置文件/etc/named.conf。如果不存在就创建该文件。

本示例实现的功能是搭建一个域名服务器 ns.oa.com，位于 192.168.19.101，其他主机可以通过该域名服务器解析已经注册的以“oa.com”结尾的域名。配置文件如【示例 3-34】所示，如需添加注释，可以使用以“#”“//”“;”开头的行或使用“/* */”包含。

【示例 3-34】

```
[root@CentOS named]# cat -n  /etc/named.conf
#此处列出的配置文件已将注释等内容略去
options {
       listen-on port 53 { any; };
       listen-on-v6 port 53 { ::1; };
       directory      "/var/named";
       dump-file      "/var/named/data/cache_dump.db";
       statistics-file "/var/named/data/named_stats.txt";
       memstatistics-file "/var/named/data/named_mem_stats.txt";
       allow-query    { any; };

       recursion yes;

       dnssec-enable yes;
       dnssec-validation yes;
       dnssec-lookaside auto;

       /* Path to ISC DLV key */
       bindkeys-file "/etc/named.iscdlv.key";

       managed-keys-directory "/var/named/dynamic";

       pid-file "/run/named/named.pid";
       session-keyfile "/run/named/session.key";
};

logging {
       channel default_debug {
              file "data/named.run";
              severity dynamic;
       };
};

zone "." IN {
```

```
        type hint;
        file "named.ca";
#以下为添加的配置项
zone "oa.com" IN {
        type master;
        file "oa.com.zone";
        allow-update { none;};
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

name.conf 配置文件中的配置项非常多，以下为主要的配置项说明：

- options：全局服务器的配置选项，即在 options 中指定的参数，对配置中的任何域都有效，若在服务器上要配置多个域，如 test1.com 和 test2.com，则在 option 中指定的选项对这些域都生效。
- listen-on port：DNS 服务实际是一个监听在本机 53 端口的 TCP 服务程序。该选项用于指定域名服务监听的网络接口，如监听在本机 IP 或 127.0.0.1 上。此处“any”表示接收所有主机的连接。
- directory：指定 named 从/var/named 目录下读取 DNS 数据文件，这个目录用户可自行指定并创建，指定后所有的 DNS 数据文件都存放在此目录下，注意此目录下的文件所属的组应为 named，否则域名服务无法读取数据文件。
- dump-file：当执行导出命令时将 DNS 服务器的缓存数据存储到指定的文件中。
- statistics-file：指定 named 服务的统计文件。当执行统计命令时，会将内存中的统计信息追加到该文件中。
- allow-query：允许哪些客户端可以访问 DNS 服务，此处“any”表示任意主机。
- zone：每一个 zone 就是定义一个域的相关信息及指定了 named 服务从哪些文件中获得 DNS 各个域名的数据文件。

3. 编辑 DNS 数据文件/var/named/oa.com.zone

该文件为 DNS 数据文件，可以配置每个域名指向的实际 IP，此文件可通过复制目录/var/named 中的 named.localhost 获得模板。文件配置内容如【示例 3-35】所示。

【示例 3-35】

```
[root@CentOS named]# cat -n  oa.com.zone
    1  $TTL  3600
    2  @      IN SOA  ns.oa.com. root (
    3                                  2015    ; serial
    4                                  1D      ; refresh
    5                                  1H      ; retry
    6                                  1W      ; expire
    7                                  3H )    ; minimum
```

```
 8         NS      ns
 9  ns      A 192.168.19.1
10  test      A  192.168.19.101
11  bbs      A  192.168.19.102
```

下面说明各个参数的含义：

- TTL：表示域名缓存周期字段，指定该资源文件中的信息存放在 DNS 缓存服务器的时间，此处设置为 3600 秒，表示超过 3600 秒则 DNS 缓存服务器重新获取该域名的信息。
- @：表示本域，SOA 描述了一个授权区域，如有 oa.com 的域名请求将到 ns.oa.com 域查找。root 表示接收信息的邮箱，此处为本地的 root 用户。
- serial：表示该区域文件的版本号。当区域文件中的数据改变时，这个数值将要改变。从服务器在一定时间以后请求主服务器的 SOA 记录，并将该序列号值与缓存中的 SOA 记录的序列号相比较，如果数值改变了，从服务器将重新拉取主服务器的数据信息。
- refresh：指定了从域名服务器将要检查主域名服务器的 SOA 记录的时间间隔，单位为秒。
- retry：指定了从域名服务器的一个请求或一个区域刷新失败后，从服务器重新与主服务器联系的时间间隔，单位是秒。
- expire：在指定的时间内，如果从服务器还不能联系到主服务器，从服务器将丢去所有的区域数据。
- Minimum：如果没有明确指定 TTL 的值，则 minimum 表示域名默认的缓存周期。
- A：表示主机记录，用于将一个主机名与一个或一组 IP 地址相对应。
- NS：一条 NS 记录指向一个给定区域的主域名服务器，以及包含该服务器主机名的资源记录。
- CNAME：用来将一个域名和该域名的别名相关联，访问域名的别名和访问域名的原始名字将解析到同样的主机地址。

第 9~11 行分别定义了相关域名指向的 IP 地址。

默认权限可能会阻止 bind 访问 oa.com.zone 文件，因由 root 用户使用命令 chgrp named oa.com.zone 修改文件所属的用户组。

4. 启动域名服务

启动域名服务可以使用 BIND 软件提供的/etc/init.d/named 脚本，如【示例 3-36】所示。

【示例 3-36】

```
[root@CentOS Packages]# systemctl start named.service
```

如启动失败，可以参考屏幕输出定位错误内容，或查看/var/log/messages 的内容，更多信息参考系统帮助“man named.conf”。

3.8.3　DNS 服务测试

经过上一节的步骤，DNS 服务端已经部署完毕，客户端需要做一定设置才能访问域名服务器，操作步骤如下。

步骤 01 配置/etc/resolv.conf。

如需正确地解析域名，客户端需要设置 DNS 服务器地址。DNS 服务器地址修改如【示例 3-37】所示。

【示例 3-37】

```
[root@CentOS ~]# cat   /etc/resolv.conf
nameserver 192.168.19.1
```

步骤 02 域名测试。

域名测试可以使用 ping、nslookup 或 dig 命令。

【示例 3-38】

```
[root@CentOS ~]# nslookup   bbs.oa.com
#先使用 server 命令确认是否使用本机作为解析 DNS
> server
Default server: 192.168.19.1
Address: 192.168.19.1#53
> bbs.oa.com
Server:        192.168.19.1
Address:       192.168.19.1#53

Name:   bbs.oa.com
Address: 192.168.19.102
```

上述示例说明了 bbs.oa.com 成功解析到 192.168.19.102。

经过以上的部署和测试演示了 DNS 域名系统的初步功能，要了解更进一步的信息可参考系统帮助或其他资料。

3.9　小结

目前 Linux 系统主要用作服务器，在互联网时代，要使用计算机就离不开网络。本章主要讲解的是 Linux 系统的网络配置。在开始配置网络之前，介绍了一些网络协议和概念。之后介绍 Linux 系统中的网络配置、Linux 内核防火墙的工作原理和使用方法，并通过一些实例的介绍使读者可以掌握 firewalld 和 iptables 的使用方法。网络数据采集与分析工具 tcpdump 在网络程序的调试过程中具有非常重要的作用，需上机多加练习。

第 4 章
路由管理

对于 Linux 系统而言网络至关重要，这是因为大多数 Linux 服务器都依赖于网络。而数据包在网络中从主机出发，经过传输、转发最终到达目标主机所依赖的是网络中的路由器。路由器是根据路由条目转发数据包，因此路由管理的实质是路由条目管理。路由管理是运维人员的重要工作之一，本章将介绍 Linux 系统中的路由管理。

本章涉及的主要知识点有：

- 路由的基本知识
- 路由的分类
- 配置 Linux 路由
- 策略路由

4.1 认识路由

路由不仅存在于路由器中，也存在于操作系统中，不仅 Linux 系统有，Windows 中也存在。本节将简单介绍路由的概念。

4.1.1 路由的基本概念

路由器传递数据包的方法与现代邮政系统工作机制相似，先按行政区域划分设立邮局。如果信件传递仅发生在邮局内部，直接分拣投递即可，例如北京市东城区某小区发往东城区另一小区的信件，只需在东城区邮政局分拣投递即可。如果信件发往外埠则需要借助邮政局间的运输网络，例如由北京市东城区发往四川省成都市成华区的邮件，则需要东城区邮政局将邮件交由北京市邮政局，再由北京市邮政局通过运输网络发往四川省，然后层层下发直到邮件到达收件人手中。

数据包的传递过程与邮政系统类似，也是先将计算机分组划分成不同子网，然后通过子网间的路由器传递数据包，如图 4.1 所示。

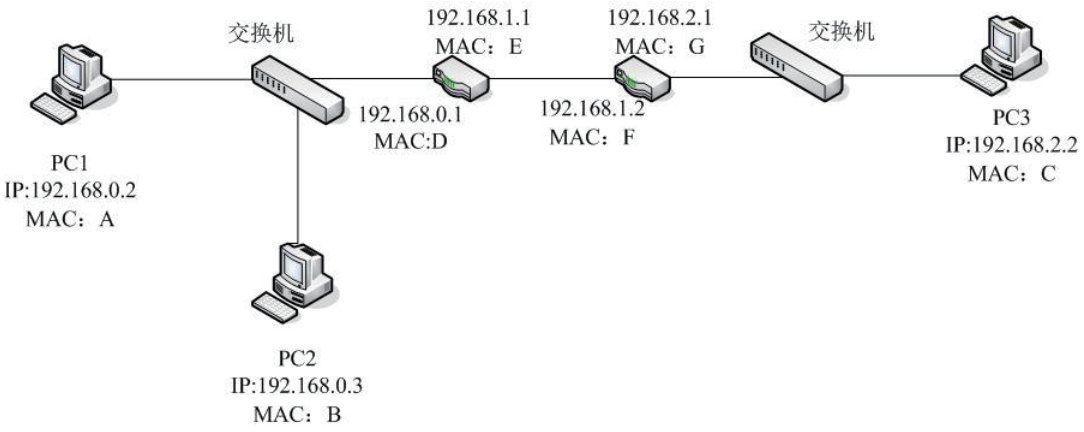

图 4.1　子网拓扑

在图 4.1 中共有 2 个交换机、路由器及 3 台计算机，共包含 192.168.0.0/24、192.168.1.0/24、192.168.2.0/24 3 个子网，本例中各设置的 IP 地址等信息均已在图中。需要特别说明的是图 4.1 中所示的 MAC 地址正确格式应为“00-51-5B-C0-00-08”或“00:51:5B:C0:00:08”，此处为方便说明简写为 A、B、C 等格式。

与邮政系统类似，当数据包发送的目标地址为同一子网时，数据包由交换机直接传送给目标主机。例如，当 PC1 发送数据包给 PC2 时，PC1 会首先发送 ARP 广播寻找 PC2 的 MAC 地址，ARP 广播会发送给子网 192.168.0.0/24 中的所有计算机。图 4.1 中所示的 PC2 和 IP 地址为 192.168.0.1 的路由器接口均会收到广播，但只有 PC2 会回应 PC1 的广播并将其 MAC 地址反馈给 PC1。当 PC1 收到 PC2 的 MAC 地址后，将目的 IP、MAC 地址正确写入数据包，然后交由交换机发送给 PC2，数据包传输过程完成了。

另一种情况是 PC1 发送的数据包目的地址与 PC1 不在同一子网时，这时就与邮政系统类似，PC1 会将数据包交给自己的“上级”默认网关。例如，在图 4.1 中 PC1 发送数据包给 PC3，数据包发送过程如下：

（1）由于 PC1 与 PC3 的地址不在同一子网，因此 PC1 会将数据包的 MAC 地址修改为 D 发送出去，交换机会将数据包交给 192.168.0.1，即路由器。

（2）与路由器相连的网络分别为 192.168.0.0/24 和 192.168.1.0/24，因此路由器并不知道网络 192.168.2.0/24 位于何处，此时就需要借助路由条目的帮助。为了能将数据包成功发往 PC3，路由器中就需要一条指向 192.168.2.0/24 的路由条目，这个条目指明数据包的下一跳地址为 192.168.1.2。

（3）根据路由条目指示，路由器会将数据包目的 MAC 地址修改为 F，将数据包发送给路由器 2。

（4）路由器 2 与 192.168.2.0/24 直接相联，因此收到数据包后，路由器 2 会将目的 MAC 修改为 C 发送给交换机，交换机再将数据包交给 PC3，数据包传输完成。

纵观上述过程，无论是数据包发送给默认网关，还是由一个路由器发送给另一个路由器，都离不开路由的参与。

4.1.2 路由的原理

在 4.1.1 节中介绍了路由的基本作用，但路由运作的机制比较复杂，本小节将简要介绍路由的基本原理。

一个路由条目至少包含 3 个要素：子网，子网掩码和下一跳地址（在有些设备中使用的是下一跳设备），其主要含义如下：

- 子网：目标子网的网络号，默认路由的子网号为 0.0.0.0。
- 子网掩码：目标子网的子网掩码，默认路由的子网掩码为 0.0.0.0。
- 下一跳地址：目标子网数据包的转发地址。在有些路由器中可以使用下一跳设备，设备通常是本地接口。

通常计算机中会有多条路由条目，计算机发出数据包时会进行计算，将目标 IP 地址与路由条目中的子网掩码按位与，即二进制按位做乘法。如果按位与的结果与路由条目的子网相同，就采用此路由条目的下一跳地址作为转发目的地。例如在图 4.1 中，路由器发往 PC3 的数据包地址为 192.168.2.2，与 192.168.2.0/24 网络的子网掩码按位与的结果为 192.168.2.0，与子网相同，因此就会将数据包转发给 192.168.1.2。

无论是计算机还是路由器，在计算路由时都遵循精确匹配原则，即如果多条路由条目都匹配目标地址，则使用最精确的条目作为转发路径。例如 IP 地址 192.168.2.2 能同时匹配 192.168.0.0/16 和 192.168.2.0/24，但基于精确匹配原则，最终将使用 192.168.2.0/24 这条路由条目。

4.1.3 Linux 系统中的路由表

在计算机中通常不止一个路由条目，能正常通信的计算机至少有两个路由条目，而路由器中的路由条目可能会更多。这些路由条目存储于路由表中，如果要在 Linux 系统中查看路由表，可以使用 route 命令，如【示例 4-1】所示。

【示例 4-1】

```
[root@localhost ~]# route
Kernel IP routing table
Destination  Gateway      Genmask       Flags  Metric Ref  Use Iface
default      172.16.45.1  0.0.0.0       UG     0      0    0   eno16777736
172.16.45.0  0.0.0.0      255.255.255.0 U      0      0    0   eno16777736
```

在示例 4-1 中，命令输出了两个路由条目，第一条是指向默认网关的默认路由，第二条是与计算机直接相连的子网路由。命令输出中 Flags 字段中的 U 表示路由条目可用，G 表示正在使用的网关。

4.1.4　静态路由和动态路由

在 4.1.1 小节中介绍了数据包如何从 PC1 发送到 PC3 的整个过程，当数据包到达 192.168.0.1 后，路由器将会计算下一步的路径，其依据就是路由器中保存的路由条目。本小节将介绍这些路由条目的来源及分类。

路由的来源有三种，第一种是路由器和计算机根据自身的网络连接自动生成的直联路由，即与自身所在同一子网的路由，只要网络持续连接直联路由就会一直存在并生效；第二种是由管理员手动添加的静态路由，静态路由仅适合于网络运作简单的环境，Linux 系统中添加的路由多为静态路由；最后一种是由动态路由协议生成的动态路由。

静态路由的缺点很明显，当路由器数量增加时，子网数量也增多，这时就需要在每个路由器上为每个子网添加路由，否则就会出现无法访问的问题。如果其中一台路由器出现问题，路由条目就会失效，也会造成无法访问的问题。

为了解决静态路由的这些问题，动态路由协议应运而生。动态路由协议会根据网络状况调整各路由器的路由条目，最大程度上保持网络通畅。常见的动态路由协议有 RIP、OSPF、BGP、IGRP 等。

（1）RIP（Routing Information Protocol，路由信息协议）是最简单的路由协议，RIP 协议要求路由器以 30 秒为周期，向相邻的路由器交换信息，从而让每个路由器都建立路由表。RIP 建立的路由表以距离为单位，通过一个路由器称为一跳，RIP 总是希望数据包通过最少的跳数到达目的地。RIP 最大的优点是配置简单，但仅适用于小型网络，如果跳数超过 15，数据包将不可达。由于路由器每 30 秒向相邻路由器交换信息，因此 RIP 协议的收敛时间相对较长（收敛时间是指路由协议让每个路由器建立精确并稳定的路由表的时间长度，时间越长，网络发生变化后，路由表生成得越慢，网络稳定需要的时间也越长）。

（2）OSPF（Open Shortest Path First，开放最短路径优先）是一个相对比较复杂的动态路由协议。OSPF 一般用于一个路由域内，称为自治系统（Autonomous System）。在这个自治系统内部，所有加入到 OSPF 的路由器都会通过路由协议相互交换信息，以维护自治系统的结构数据库，最后路由器会通过数据库计算出 OSPF 路由表。与 RIP 相比，OSPF 协议根据链路状态计算路由表，更适合于大型网络，其收敛速度也更快。

（3）BGP（Border Gateway Protocol，边界网关协议）是一个用来处理自治系统之间的路由关系的路由协议，最适合处理像 Internet 这样十分巨大的网络。BGP 既不完全是距离矢量协议，也不完全像 OSPF 那样使用链路状态，BGP 使用的是通路向量路由协议。BGP 使用 TCP 协议进行可靠的传输，同时还使用了路由汇聚、增量更新等功能，极大地增加了网络可靠性和稳定性。

（4）IGRP（Interior Gateway Routing Protocol，内部网关路由协议）是由 Cisco 公司设计的专用于 Cisco 设备上的一种路由协议。IGRP 是一种距离向量路由协议，其要求路由器以 90 秒为周期向相邻的路由器发送路由表的全部或部分，由此区域内的所有路由器都可以计算出所有网络的距离。由于使用网络延迟、带宽、可靠性及负载等都被用作路由选择，因此 IGRP 的稳定性相当不错。

动态路由协议除了上面介绍的 4 种之外，还有许多例如 IS-IS 等，此处不再赘述，感兴趣的读者可自行参考相关资料了解。

4.2 配置 Linux 静态路由

与其他操作系统不同，Linux 系统作为常见的服务器操作系统，其可能会遇到更多样的网络环境。除了常见服务器使用的 Internet 网络连接外，通常还会有公司内部网络、远程访问相关的网络等，此时就需要正确设置路由，否则无法正确访问。本节将简要介绍如何在 Linux 系统中设置静态路由。

4.2.1 配置网络接口地址

设置静态路由的前提是网络接口上配置有 IP 地址等信息，否则路由条目无法生效。在网络接口上配置单个 IP 地址的相关知识已在第 3 章中介绍过，此处介绍如何在同一接口上配置多个 IP 地址的方法。

（1）使用子接口

使用子接口在网络接口上配置多个 IP 地址是一个比较常见的做法，子接口名字形如 eno16777736:1，其中 eno16777736 是网络接口的名称，“:1”表示这是一个子接口。配置过程如【示例 4-2】所示。

【示例 4-2】

```
[root@localhost ~]# ifconfig eno16777736:1 172.16.45.134/24 up
[root@localhost ~]# ifconfig
……
eno16777736:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.45.134  netmask 255.255.255.0  broadcast 172.16.45.255
        ether 00:0c:29:23:7c:d2  txqueuelen 1000  (Ethernet)
……
```

使用以上命令配置的子接口将在重启后消失，若要重启后继续生效则需要将上述命令写入文件/etc/ec.local 中。

（2）使用多配置

CentOS 7.5 允许在一个网络接口上配置多个不同 IP 地址、子网掩码、网关和 DNS 服务器地址等，但同时只能激活一个配置。多配置在图形界面中可以单击“Applications”，然后在弹出的菜中依次单击“System tools”→“Settings”，打开设置界面，如图 4.2 所示。

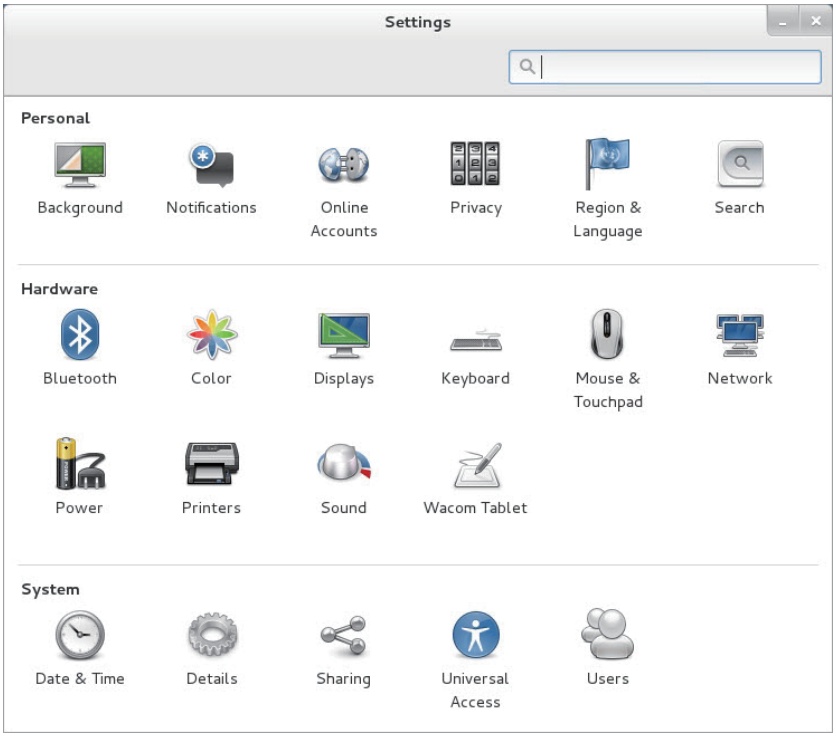

图 4.2　设置界面

在设置界面中可以找到 CentOS 7.5 中几乎所有的常规设置，此时单击“Network”弹出网络设置界面，如图 4.3 所示。

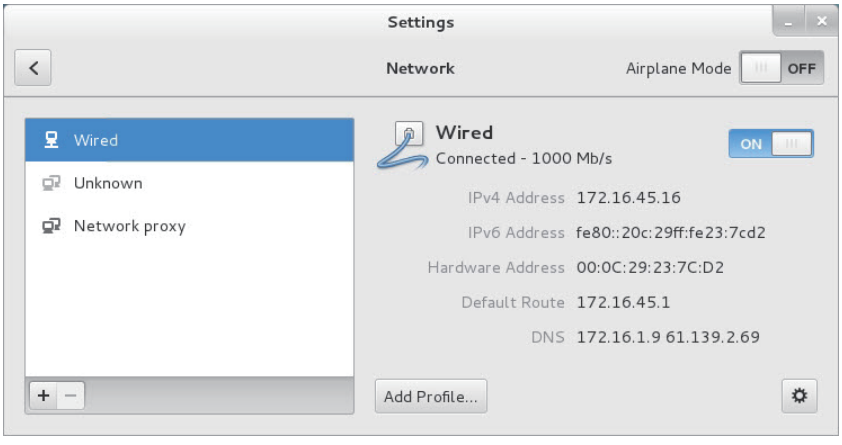

图 4.3　网络设置界面

在网络设置界面中可以看到网络接口相关设置，此时可以单击“Add Profile”按钮为已连接网络连接添加配置文件。添加配置文件的界面如图 4.4 所示。

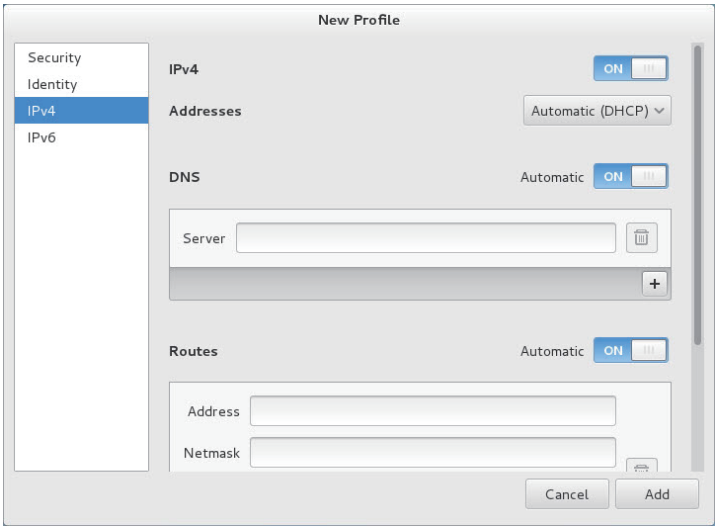

图 4.4　添加配置文件

在新配置中可以添加诸如 802.1x、IPv4 等类型网络，以常见的 IPv4 网络为例，可以在左侧选择 IPv4，然后在右侧的“Addresses”中选择“Manual”。之后就可以在下面填入 IP 地址、子网掩码、网关、DNS、静态路由等信息了。

多次添加即可在同一个网络连接上添加多个配置文件，这些配置文件可以在网络设置界面的右侧看到，如图 4.5 所示。

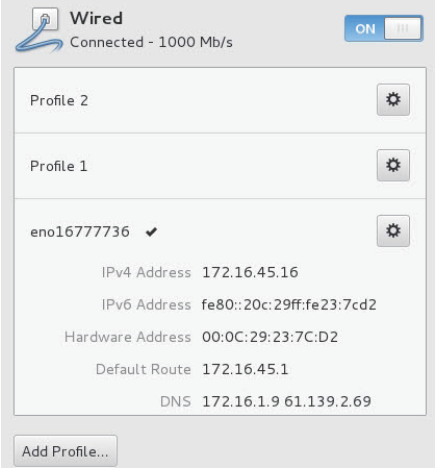

图 4.5　网络连接的多配置文件

添加了多配置文件后，接下来的任务就是切换配置文件让不同的配置文件在不同的网络环境中生效。切换配置文件需要单击桌面右上角的联网图标菜单，将弹出所有的配置文件列表，如图 4.6 所示。

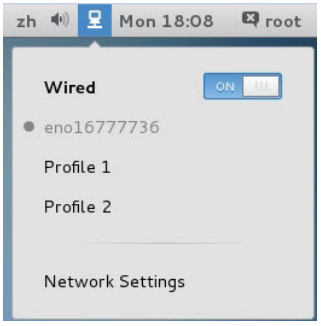

图 4.6　网络配置文件列表

此时只需要单击对应的配置文件名称，就可以让相应的配置文件生效，如果系统重启就采用上一次生效的配置文件。

4.2.2　接口 IP 地址与直联路由

无论使用哪种方式为网络接口配置 IP 地址等信息，只要网络接口接入某个子网，路由表就会立即为子网添加相应的直联路由。可以使用 route 命令查看路由表验证，如【示例 4-3】所示。

【示例 4-3】

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.16.45.1     0.0.0.0         UG    100    0        0 eno16777736
172.16.45.0     0.0.0.0         255.255.255.0   U     100    0        0 eno16777736
```

在【示例 4-3】的命令输出中，第二条就是与 172.16.45.0 子网的直联路由，这是由接口 eno16777736 的 IP 配置决定的。如果此接口的 IP 地址发生变化或有新的接口拥有了 IP 地址，路由表中的直联路由也会发生变化，如【示例 4-4】所示。

【示例 4-4】

```
[root@localhost ~]# ifconfig eno16777736:1 192.168.1.100/24 up
[root@localhost ~]# route -n
Kernel IP routing table
Destination  Gateway      Genmask         Flags  Metric Ref  Use  Iface
default      172.16.45.1  0.0.0.0         UG     100      0    0    eno16777736
172.16.45.0  0.0.0.0      255.255.255.0   U      100      0    0    eno16777736
192.168.1.0  0.0.0.0      255.255.255.0   U      0        0    0    eno16777736
```

由以上示例可以看到当接口的 IP 地址发生改变后，路由表中的直联路由也发生了改变。

4.2.3 route 命令

在 Linux 系统中，查看、添加、删除路由的是 route 命令。其添加删除路由时的基本格式如【示例 4-5】所示。

【示例 4-5】

```
route add|dell [-net|-host] ipaddress1 netmask netmask gw ipaddress2|dev
```

各项参数含义如下：

- add|del：表示添加或删除一个路由条目。
- -net|-host：路由条目的目的地是一个子网或一台主机。
- ipaddress1：目标子网的子网号或目标主机的 IP 地址。
- netmask：目标子网或主机的子网掩码，当目标为主机时，子网掩码长度应为 32 位。
- gw：用于指定下一跳地址或下一跳设备。通常将 Linux 作为一台路由器使用时才会使用下一跳设备。

除以上列举的参数之外，还有一个用于显示路由表时使用的选项 n，此选项表示使用 IP 地址显示而不尝试使用域名。IP 地址转换为域名需要解析，因此使用选项 n 可以快速显示路由表，如【示例 4-6】所示。

【示例 4-6】

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination  Gateway      Genmask        Flags  Metric  Ref  Use  Iface
default      172.16.45.1  0.0.0.0        UG     100     0    0    eno16777736
172.16.45.0  0.0.0.0      255.255.255.0  U      100     0    0    eno16777736
```

【示例 4-6】所示的命令输出了系统内核的路由表，路由表中的几个字段含义如下：

- Destination：目标网络号或目标主机 IP 地址，default 表示这是一条默认路由。
- Gateway：网关地址即下一跳地址，其中 0.0.0.0 或“*”表示主机与该子网直接相联，无须下一跳地址（直联路由）。
- Genmask：子网对应的子网掩码。
- Flags：路由标记。
- Metric：路由条目的代价值。Metric 数值越高，代价越大，此值一般在有多条到目标网络的路由时才起作用。
- Ref：路由条目被引用的次数。
- Use：路由条目被路由软件查找的次数。
- Iface：到达目标网络使用的本地接口。

在上面的字段中，Flags 路由标记用于指示路由条目的状态，常见的状态标记及含义如下：

- U：当前路由处于活动状态（可用状态）。

- H：路由条目的目标是主机而不是子网。
- G：指向默认网关的路由。
- R：恢复动态路由产生的路由。
- D：由后台程序动态产生的。
- M：此条目经过了后台程序修改。
- C：缓存的路由条目。
- !：拒绝路由。

route 命令还可以用于添加默认路由（通常称为默认网关），但更多的是用于添加静态路由，使用方法如【示例 4-7】所示。

【示例 4-7】

```
   #添加、删除默认路由
   [root@localhost ~]# route add default gw 172.16.45.1
   [root@localhost ~]# route del default gw 172.16.45.1
   #添加、删除到网络的路由
   #子网掩码采用不同的形式，因此以下两条语句的功能相同
   [root@localhost ~]# route add -net 192.168.19.0/24 gw 172.16.45.100
   [root@localhost ~]# route add -net 192.168.19.0 netmask 255.255.255.0 gw
172.16.45.100
   [root@localhost ~]# route -n
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   0.0.0.0         172.16.45.1     0.0.0.0         UG    100    0        0 eno16777736
   172.16.45.0     0.0.0.0         255.255.255.0   U     0      0        0 eno16777736
   192.168.19.0    172.16.45.100   255.255.255.0   UG    0   0        0 eno16777736
   [root@localhost ~]# route del -net 192.168.19.0/24
   #添加、删除到主机的路由
   [root@localhost ~]# route add -host 192.168.100.80 gw 172.16.45.100
   [root@localhost ~]# route -n
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   0.0.0.0         172.16.45.1     0.0.0.0         UG    100    0        0 eno16777736
   172.16.45.0     0.0.0.0         255.255.255.0   U     100    0        0 eno16777736
   192.168.100.80  172.16.45.100   255.255.255.255 UGH   0   0        0 eno16777736
   [root@localhost ~]# route del -host 192.168.100.80
```

4.2.4　Linux 路由器配置实例

学习完之前的基础知识之后，可以利用 Linux 来制作一个路由器。本小节将简单介绍路由器包括的功能及如何将 Linux 配置成一个实用的路由器。

一个实用的路由器最起码应该包括 DHCP、数据包转发、NAT 等，DHCP 用来为子网中的计算机分配 IP 地址、网关、DNS 等信息；数据包转发是路由器的核心功能，用来将数据包

准确地转发到相应的子网；NAT 功能用来做地址转换，即将子网发往外部网络的数据包地址做转换。在 Linux 系统上配置路由器可以采取两个方案，其一是使用 Linux 自身的内核转发功能、配置 DHCP 服务并使用防火墙的地址伪装做 NAT 功能；其二是使用其他路由软件，例如著名的 Zebra 等。本小节将采用 Linux 内核的数据包转发功能作为示例讲解。

在本节的路由器配置中，采用的拓扑图如图 4.7 所示，路由器的一端连接子网 192.168.0.0/24。

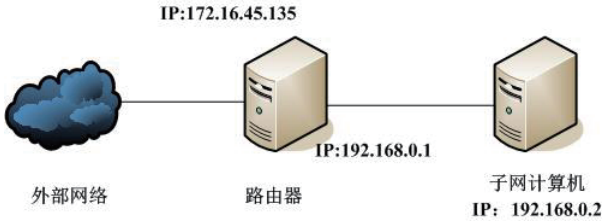

图 4.7　路由拓扑

在开始配置路由器之前，需要先为路由器和子网计算机上的网络连接正确配置 IP 地址，确保路由器能正常访问外部网络。配置子网计算机时，默认网关应该为 192.168.0.1。配置完 IP 地址后，接下来需要配置内核转发，让内核具有转发数据包的功能，如【示例 4-8】所示。

【示例 4-8】

```
#添加内核转发参数
[root@localhost ~]# cat /etc/sysctl.conf
……
net.ipv4.ip_forward = 1
#让内核参数生效
[root@localhost ~]# sysctl -p
net.ipv4.ip_forward = 1
```

这样 Linux 就具备了数据包转发功能，接下来需要让 Linux 防火墙具备 NAT 功能，这个功能通常由防火墙 iptables 来完成，如【示例 4-9】所示。

【示例 4-9】

```
#禁用并停止 firewalld
[root@localhost ~]# systemctl disable firewalld
[root@localhost ~]# systemctl stop firewalld
#安装 iptables 防火墙
[root@localhost ~]# yum install -y iptables-services
#启用并开启 iptables
[root@localhost ~]# systemctl start iptables
[root@localhost ~]# systemctl enable iptables
#在 eno16777736接口上开启地址伪装
[root@localhost ~]# iptables -t nat -I POSTROUTING -o eno16777736 -j MASQUERADE
```

接下来就可以在子网计算机上访问外部网络了，可以通过 ping 命令、curl 访问网址等方

式验证。

由于本书的第 3 章已介绍过 DNS、DHCP 等知识，因此此处不再赘述，读者可自行参考相关资料配置。

4.3 Linux 的策略路由

传统的路由是一个指向目标子网的“指路牌”，任何人来“问路”，路由都会明确指向目标。传统路由这种“不问来人情况”的处理策略越来越不适合现代计算机网络，举例来说“行人与汽车”走的“路”应该是不同的。这样策略路由就兴起了，策略路由是近些年来兴起的一个比较新的路由概念。策略路由可以根据多种不同的策略决定数据包通过的路径。本节将简要介绍 Linux 系统中的路由及策略路由的使用。

4.3.1 策略路由的概念

并不是所有环境都适合策略路由。策略路由与传统路由相比最大的不同是，策略路由通常有不止一条到达目的网络的路径，例如在一个网络中有两个出口联通和电信等。因此策略路由在企业的实施环境首要条件是网络有多个出口，例如一电信一联通，又如一出口速度较快、另一出口速度相对更慢一些等。

策略路由按实现的方式大体可以分为三类，第一种是按目的地址进行路由，即根据目的地不同选择不同的出口。由于这种按目的地址进行路由的方法特别适合做双线服务器，因此在国内使用的较多。第二种是按源地址进行路由，即根据发出数据包的计算机地址决定选择哪个出口，这种方法适用于多种环境，例如多线机房、客户定制的更快速的网络等。第三种是智能均衡策略路由，这是一种出现时间较晚的方式，在这种方式下会自动识别网络带宽及负载，根据带宽和负载动态地决定数据包从哪个出口发出。

无论使用何种策略路由，都必须注意保护连接的持续性，特别是在出口上使用了 NAT 的网络中，即需要保证内部网络主机与外部通信时，数据包的往返都使用的是同一出口，否则可能会造成资源浪费，甚至无法连接的情况出现。

4.3.2 路由表管理

在 Linux 系统中，策略路由可以通过路由表来实现，但 Linux 系统中的路由表并不像普通路由器那样简单，本小节将介绍 Linux 系统中的路由表。

默认情况下，Linux 并非只有一个路由表，因为如果系统中只有一个路由表，策略路由的许多功能将无法实现。数据包转发时，并不需要将所有路由表都搜索计算一次，数据包应该使用哪个路由表路由，取决于系统设定的规则。查看系统默认的规则，使用命令“ip rule list”

或“ip rule show”，如【示例 4-10】所示。

【示例 4-10】

```
[root@localhost ~]# ip rule list
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

【示例 4-10】展示的是没有经过修改的 Linux 系统规则列表，其中输出了 3 个路由表 local、main 及 default。每条规则前面的数字表示规则的优先级，数值越小，表明优先级越高，而“from all”表明所有的数据包都需要经过路由表的匹配。

由此可以看出【示例 4-10】所示的处理过程应该是，内核转发的数据包先使用表 local 转发，如果没有匹配的路由条目再依次使用表 main 和 default。为了搞清 Linux 系统数据包的转发流程，有必要搞清这 3 张路由表的内容，如【示例 4-11】所示。

【示例 4-11】

```
[root@localhost ~]# ip route list table local
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
broadcast 172.16.45.0 dev eno16777736  proto kernel  scope link  src 172.16.45.13
local 172.16.45.13 dev eno16777736  proto kernel  scope host  src 172.16.45.13
broadcast 172.16.45.255 dev eno16777736  proto kernel  scope link  src
172.16.45.13
[root@localhost ~]# ip route list table main
default via 172.16.45.1 dev eno16777736  proto static  metric 100
172.16.45.0/24 dev eno16777736  proto kernel  scope link  src 172.16.45.13
172.16.45.0/24 dev eno16777736  proto kernel  scope link  src 172.16.45.13
metric 100
[root@localhost ~]# ip route list table default
[root@localhost ~]#
```

【示例 4-11】分别输出了 3 张路由表中的路由条目，其中表 default 中的路由条目为空，此处不做讨论。由于计算机的 IP 地址为 172.16.45.13，因此表 local 中的路由条目为来自广播和目的地为本地接口 IP 的数据包路由。而表 main 中的路由条目，很明显是指向本地子网（子网 172.16.45.0/24）和指向默认网关的默认路由。

由于环境并不复杂，因此系统默认的规则中并没有决定哪些数据包应该使用具体的某个路由表。了解 Linux 系统的路由表机制后，接下来我们就可以利用这些机制建立自己需要的路由表和规则了。建立一个路由表和相应的规则如【示例 4-12】所示。

【示例 4-12】

```
#建立一个名为 test1的路由表
```

```
[root@localhost ~]# echo 100 test1 >> /etc/iproute2/rt_tables
#建立一个规则，规定所有来自192.168.19.0/24的数据包都使用路由表 test1中的条目路由
[root@localhost ~]# ip rule add from 192.168.19.0/24 table test1
#列出规则
[root@localhost ~]# ip rule list
0:      from all lookup local
32765:  from 192.168.19.0/24 lookup test1
32766:  from all lookup main
32767:  from all lookup default
```

在【示例 4-12】中，先使用编辑 rt_tables 的方式添加一个名为 test1 的路由表，然后添加一条规则，规定所有源地址为 192.168.19.0/24 的包通过 test1 路由表路由。由于我们并没有向路由表 test1 中添加任何路由条目，因此此时路由表 test1 还为空，如何添加将在下一小节中介绍。

添加 test1 路由表时，使用了数字 100 作为保留值（保留值为 table ID，test1 相当于 table ID 的别名，此值与优先级无关，优先级将自动分配），通常建议这个值小于 253 且不重复，具体可以查看 rt_tables 文件中的说明。除了以上这种编辑文件的方法外，还可使用指定 table ID 的方法添加路由表，如【示例 4-13】所示。

【示例 4-13】

```
[root@localhost ~]# ip rule add from 192.168.18.0/24 table 2 pref 1500 prohibit
```

以上示例将添加一个 table ID 为 2 的路由表，并指定其优先级为 1500。

删除路由表与以上过程相反，首先需要删除相关的规则，然后编辑文件 rt_tables，删除其中的相关配置，如【示例 4-14】所示。

【示例 4-14】

```
#删除规则和 rt_tables 中的相关内容
[root@localhost ~]# ip rule del table test1
[root@localhost ~]# cat /etc/iproute2/rt_tables
#
# reserved values
……
#
#1      inr.ruhep
#验证结果
[root@localhost ~]# ip rule list
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

系统重启后，规则将失效，如需继续生效可以将设置规则的相关语句写入/etc/rc.local 中。

4.3.3 规则与路由管理

从前面几小节的内容中不难看出，Linux 策略路由管理的两个核心分别是规则与路由表中路由的管理。虽然之前已经介绍过路由管理的相关概念，但与之前的路由管理相比，此处将要麻烦一些，因为在策略路由中还需要细化一些参数。

1. 规则

在策略路由中，规则如同一个筛选器，将数据包按预先的设置“送给”路由表，完成路由过程。添加一条规则使用命令 ip，格式如下：

```
Ip rule [add|del] SELECTOR ACTION
```

在以上格式中，“[add|del]”表示添加或删除一条规则，“SELECTOR”表示数据包选择部分，“ACTION”表示执行的操作。其中“SELECTOR”可以选择数据包的多种选项，常见选项如下所示：

- from：源地址。
- to：目的地址。
- tos：数据包的 TOS（Type of Service）域，用于标明数据包的用途。
- fwmark：防火墙参数。
- dev：参与设备，具体包括两个选项 iif 和 oif，分别表示接收和发送设置匹配。
- pref：指定优先级。

在以上选项中，无疑 from 和 to 是常用的选项。除以上选项外，还有一些其他选项，读者可阅读相关文档了解或参考 ip-rule 的手册页。

与“SELECTOR”一样，“ACTION”执行的动作也有多种：

- table：指明使用的 table ID 或表名。
- nat：透明网关，同 NAT 相似。
- prohibit：丢弃包并返回“Communication is administratively prohibited”的错误消息。
- unreachable：丢弃包并返回“Network is unreachable”的错误消息。
- realms：指定数据包分类，此选项主要用于配合 tc 做流量整形。

在“ACTION”执行的动作中，table 和 nat 是较常用的，prohibit 和 unreachable 主要用来禁止通信，因此使用较少。

在【示例 4-15】中列举了一些常见的示例。

【示例 4-15】

```
#以源地址作为路由依据
ip rule add from 192.168.19.0/24 table test1
ip rule add from 192.168.17.100/32 table test1
#以源地址和 tos 作为路由依据
ip rule add from 192.168.0.0/16 tos 0x10 table test1
```

```
#以目标地址作为路由依据
ip route add to 192.168.100.100/32 table test1
ip route add to 192.168.101.0/24 table test1
#以防火墙标记为路由依据，需要防火墙使用选项 set-mark 标记
ip rule add fwmark 1 table 1
```

2. 路由管理

与之前介绍的使用 route 命令添加路由相比，策略路由的路由管理稍稍复杂一些，其格式如下所示：

```
ip route add ipaddress via ipaddress1 table table_name
```

其中，ipaddress 参数表示网络号，via 选项指定的参数 ipaddress1 表示网关 ip 地址，即下一跳地址，table_name 表示路由表名。

一些比较常见的路由条目如【示例 4-16】所示。

【示例 4-16】

```
#添加到 test1的默认路由
ip route add default via 192.168.11.1 table test1
#发往192.168.15.0/24网络的包下一跳地址是192.168.11.1
ip route add 192.168.15.0/24 via 192.168.11.1 table test1
```

4.3.4　策略路由应用实例

在前面几小节中，介绍了 Linux 系统的策略路由的运作机制，本小节将通过几个实例介绍策略路由的应用。

在本小节的例子中，Linux 主机连接了两个子网（192.168.1.0/24 及 192.168.2.0/24），拥有两个出口（第一个是 172.16.33.2，对端网关为 172.16.33.1；另一个为 172.16.34.2，对端网关为 172.16.34.1），如图 4.8 所示。

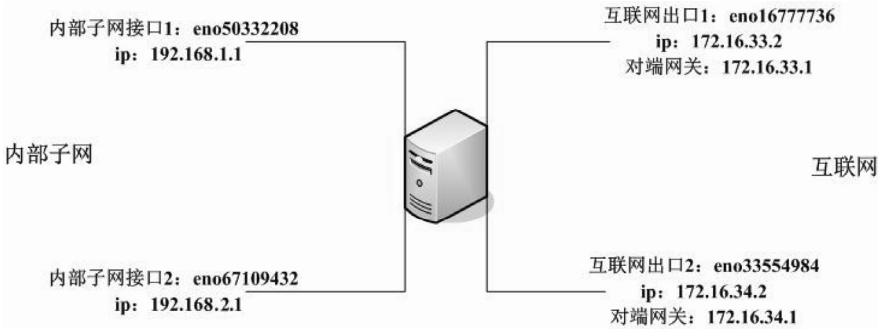

图 4.8　策略路由拓扑

需要说明的是出口及对端网关地址应为网络供应商提供的公网 IP 地址，此处出于安全考虑以私网地址代替，其配置过程仅需将 IP 地址等信息替换即可，其他并无不同之处。

1. 选择出口

如图 4.8 所示的网络结构，现有两个出口，其中互联网出口 1 为所有内部子网的默认出口，出口 2 比出口 1 速度更快，但仅供内部网络中的 VIP 用户使用。假定有 VIP 用户的 IP 地址为 192.168.1.52 和 192.168.2.54，现需要配置这两个地址的流量，使用出口 2 以获得更快的速度。

（1）配置默认路由

根据以上信息先配置接口 eno16777736，将 IP 地址、默认网关等信息一并设置，而出口 2 仅正确设置 IP 地址及子网掩码即可，无须设置默认网关。关于这些设置，读者可自行参考第 3 章中的相关内容，此处不再赘述。

（2）配置策略路由

在上一步配置中，已将所有子网的数据包的转发出口设置为出口 1，现在需要配置 VIP 用户的数据包从出口 2 进行转发，其配置方法如【示例 4-17】所示。

【示例 4-17】

```
#建立路由表 T1
[root@localhost ~]# echo 200 T1 >> /etc/iproute2/rt_tables
#设置 VIP 用户的数据包，使用路由表 T1路由
[root@localhost ~]# ip rule add from 192.168.1.52/32 table T1
[root@localhost ~]# ip rule add from 192.168.2.54/32 table T1
[root@localhost ~]# ip rule ls
0:      from all lookup local
32764:  from 192.168.2.54 lookup T1
32765:  from 192.168.1.52 lookup T1
32766:  from all lookup main
32767:  from all lookup default
#为路由表 T1添加出口2的默认路由
[root@localhost ~]# ip route add default via 172.16.34.1 table T1
[root@localhost ~]# ip route list table T1
default via 172.16.34.1 dev eno33554984
```

以上是整个配置过程，但以上配置在系统重启后会消失，因此需要添加规则，将添加路由表 T1 默认路由的语句写入/etc/rc.local 中。

2. 负载均衡

负载均衡的配置方法与选择出口的配置方法略有不同，因为负载均衡时需要考虑一个新的问题，需要保证连接的持续性，即从互联网出口 1 进来的数据包返回时也从出口 1 返回；出口 2 亦相同。负载均衡的配置方法如【示例 4-18】所示。

【示例 4-18】

```
#添加两个路由表 table1和 table2
[root@localhost ~]# echo 100 table1 >> /etc/iproute2/rt_tables
```

```
    [root@localhost ~]# echo 200 table2 >> /etc/iproute2/rt_tables
    #分别添加返回路由
    [root@localhost ~]# ip route add default via 172.16.33.1 dev eno16777736 src
172.16.33.2 table1
    [root@localhost ~]# ip rule add from 172.16.33.2 table table1
    [root@localhost ~]# ip route add default via 172.16.34.1 dev eno33554984 src
172.16.34.2 table2
    [root@localhost ~]# ip rule add from 172.16.34.2 table table2
    #设置负载均衡策略
    [root@localhost ~]# ip route add default scope global nexthop via 172.16.33.2
dev eno16777736 weight 1 netxthop via 172.16.34.2 dev eno33554984 weight 1
```

在上面的示例中，weight 用于指定出口的权重，此处都设置为 1，表示平等对待，如果需要区别对待可以修改此值。与之前的设置方法相同，如果需要设置在重启后仍然生效，可以将规则和路由添加到文件/etc/rc.local 中。

本小节仅讨论了策略路由如何实施，并没有包含诸如 NAT 等问题，关于路由器转发相关内容可参考 4.2 节中的相关内容。

4.4 小结

路由是 Linux 系统中相当重要的内容，本章从实际应用出发主要介绍了 Linux 系统的路由相关内容，通过实例介绍了传统路由的设置、数据包转发等内容。对于 Linux 上的策略路由问题，剖析了 Linux 策略路由的运作机制，并通过实例介绍了策略路由的应用。