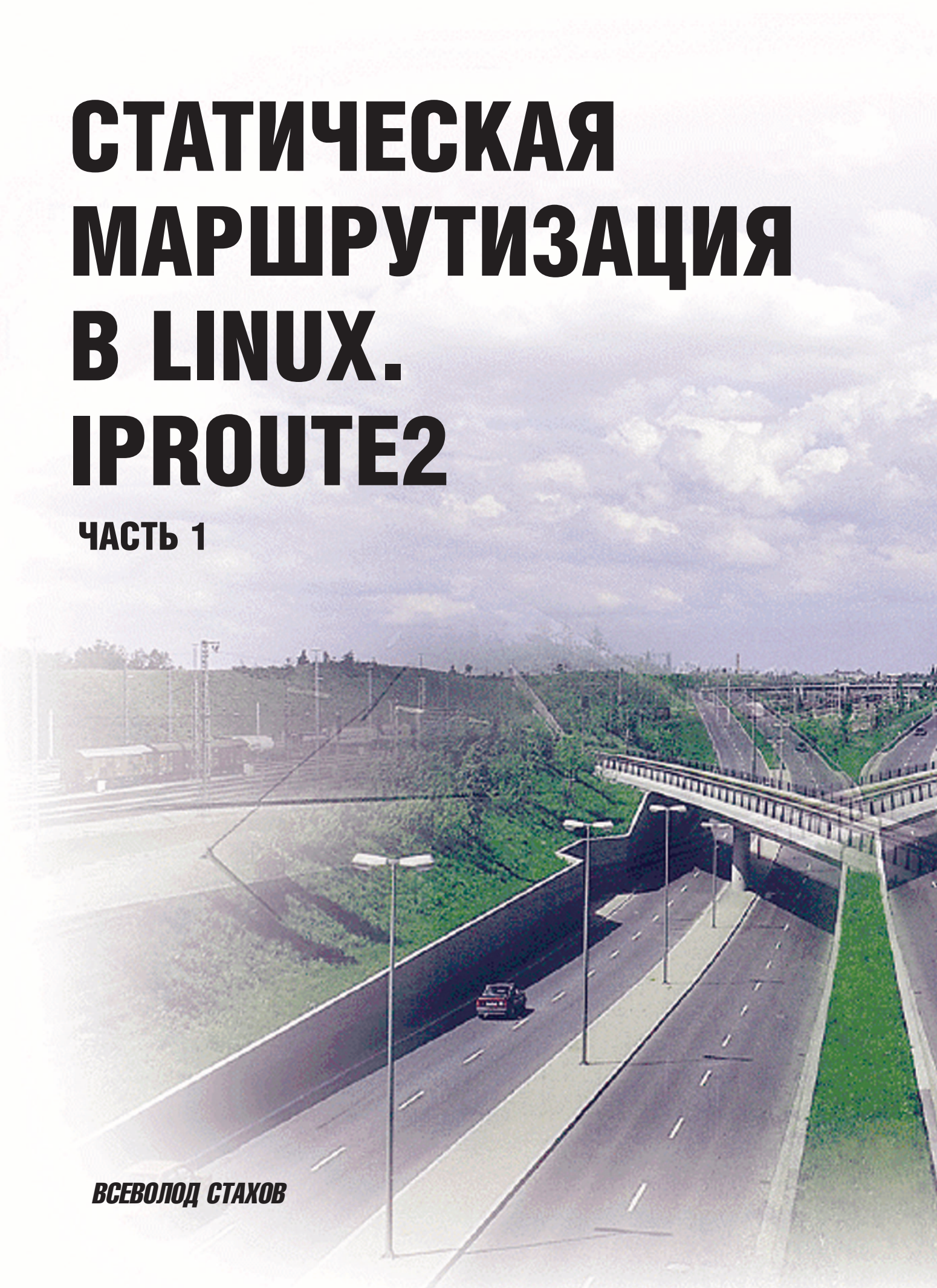
СТАТИЧЕСКАЯ МАРШРУТИЗАЦИЯ В LINUX. IPROUTE2

ЧАСТЬ 1



ВСЕВОЛОД СТАХОВ

Наверное, любой из вас хотя бы отдаленно знает, что такое маршрутизация. Итак, маршрутизация – это, как бы это банально не звучало, есть выбор маршрута. В данной статье под этим термином я буду понимать выбор маршрута следования сетевого IP-пакета. Дело в том, что современные программные маршрутизаторы (а рассказывать я буду как раз об одном из представителей данного класса устройств) умеют полноценно работать только с протоколом IP.

Почему же я решил описать построение маршрутизатора именно на основе ОС GNU/Linux(*)? Тут две основные причины:

- ядро GNU/Linux способно уместиться на дискете, что может позволить создать весьма функциональный маршрутизатор вне зависимости от конкретной машины, кроме этого, можно «оживить» старые машины и заставить их работать на пользу людям;
- ядро Linux(2.4, 2.2) поддерживает очень полезные функции маршрутизации, и может быть специально «заточено» под использование в качестве маршрутизатора, кроме этого, стандартный брандмауэр Linux 2.4 – iptables может «метить» (не подумайте ничего плохого) определённые пакеты, а ядро может выполнять выбор маршрута согласно этим меткам.

Это открывает широкие возможности при создании сетей со сложной структурой. Ещё очень важной особенностью является универсальность GNU/Linux, по-моему, эта ОС поддерживает в той или иной степени все распространённые сетевые протоколы. Ещё одной немаловажной особенностью является бесплатность всей системы маршрутизации. С точки зрения многих администраторов, маршрутизатор – это просто черный ящик, принимающий и передающий пакеты, однако грамотная настройка маршрутизации – залог эффективности и зачастую безопасности всей сети. Очень интересно использовать маршрутизацию для распределения нагрузки, передачи определённого трафика на определённый хост (для анализа) и уменьшения опасности DoS-атак. Маршрутизация способна ограничивать сетевые «штормы» и существенно увеличить пропускную способность сети. Я решил построить эту статью в виде конкретных примеров настройки маршрутизации (в дальнейшем я иногда буду употреблять слово роутинг).

Маршрутизация бывает статической и динамической. Отличие в том, что при статической маршрутизации все правила передачи пакетов прописываются статически и могут быть изменены только вручную, динамическая маршрутизация применяется, когда в сети существует не-

сколько маршрутизаторов, и нахождение пути до удаленного хоста становится нетривиальной задачей. Динамическая маршрутизация больше подходит для часто меняющихся сетей со сложной структурой. Хотя GNU/Linux поддерживает оба типа маршрутизации, но в рамках данной статьи я буду рассказывать о статической маршрутизации при помощи пакета iproute2, написанного нашим программистом Алексеем Кузнецовым. Для начала работы необходимо настроить соответствующим образом ядро и установить пакет iproute. Остановлюсь на настройке ядра. В ядре необходимо включить ряд опций маршрутизации (думаю, нет нужды объяснять, как настраивать и компилировать ядро).Я предполагаю, что вы настраиваете ядро командой:

```
make menuconfig
```

На странице Networking Options необходимо включить следующие элементы:

- IP: advanced router – включение расширенных возможностей маршрутизации;
- IP: policy routing – маршрутизация по некоторым внутренним полям пакетов (обычно применяется совместно с брандмауэром), а также для расширенных возможностей маршрутизации, например, маршрутизация согласно адресу-источнику пакета (source-based routing);
- IP: use netfilter MARK value as routing key – включение возможности маршрутизации согласно маркировке пакета брандмауэром;
- IP: use TOS value as routing key – маршрутизация пакетов на основе заголовка тип сервиса (TOS), помогает увеличить пропускную способность сети при наличии нескольких путей прохождения пакетов;
- IP: large routing tables – включение больших (>64 правил) таблиц маршрутизации ядра.

Можно также включить поддержку туннелей, но я не буду на этом задерживаться. После настройки ядра необходимо установить iproute2 в большинстве дистрибутивов GNU/Linux эта программа входит в дистрибутив, например, для Debian GNU/Linux команда будет выглядеть так:

```
apt-get install iproute
```

исходные коды могут быть получены по адресу: ftp://ftp.inr.ac.ru/IProuting/IProute2-xxx.tar.gz, компиляция стандартная, но цели install в Makefile нет – необходимо скопировать бинарные файлы из каталога ip (cp ifcfg ip routef routel rtacct rtmon rtpr /sbin) и из каталога tc (cp tc /sbin) в /sbin, а ./etc/iproute2/ – в /etc/iproute2/.

Не поленитесь также скачать Linux Advanced Routing and Traffic Control HOWTO, которое может быть найдено на узле www.lartc.org. Это руководство необходимо для настройки сложной статической маршрутизации на основе Linux. Я сам настраивал маршрутизацию в сети на основе этого руководства, поэтому если эта статья не решила вашей проблемы, лучше обратиться к данному документу.

Пакет iproute состоит фактически из двух утилит управления трафиком:

- ip – управление собственно маршрутизацией;
- tc – управление очередями маршрутизации.

Для начала расскажу об общих принципах команды ip, синтаксис команды таков:

```
ip [опции] {объект маршрутизации} {команда или HELP}
```

Из опций наиболее полезным является выбор семейства IP

- 4 – IPv4;
- 6 – IPv6.

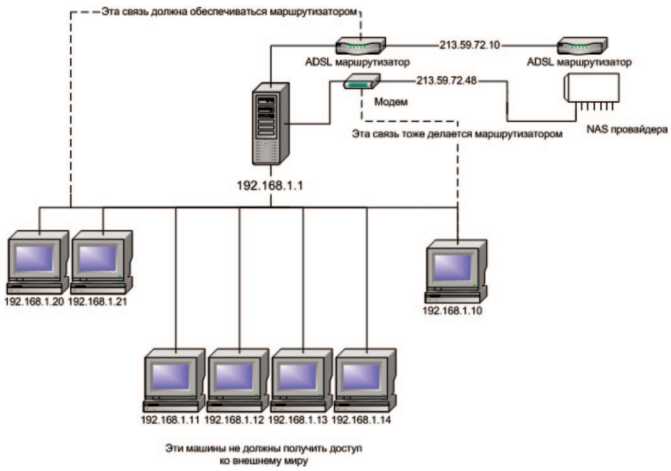
Список объектов маршрутизации:

- link – сетевое устройство (реальное физическое или виртуальное, например vlan или туннель);
- address – IP-адрес устройства;
- neighbour – кеш ARP;
- route – таблицы маршрутизации;
- rule – правила маршрутизации;
- maddress – широковещательный адрес;
- mroute – широковещательные таблицы маршрутизации;
- tunnel – IP-туннель.

Команды для разных объектов разные, но для всех объектов существует стандартный набор команд add (добавить), delete (удалить) и show (показать можно также применять list или ls). Синтаксис различных команд для разных объектов может быть совершенно разным, поэтому я не буду описывать здесь все команды каждого объекта. Я буду придерживаться стиля Linux Adv. Routing HOWTO и приведу полезные примеры употребления команды ip. Для начала просмотрим сетевые устройства, присутствующие на нашей тестовой машине (пусть у неё будут IP-адреса 192.168.1.1 и 192.168.2.1):

```
# ip link list
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop
     link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST, MULTICAST, UP> mtu 1500 qdisc ↵
pfifo_fast qlen 10
    link/ether 48:54:e8:01:ef:56 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST, MULTICAST, UP> mtu 1500 qdisc ↵
pfifo_fast qlen 10
     link/ether 00:e0:4c:39:ef:56 brd ff:ff:ff:ff:ff:ff
```

Теперь настало время перейти к рассмотрению простейшего случая организации маршрутизации. Допустим, в локальной сети крупных размеров есть три компьютера, которым положено иметь доступ к глобальной сети. При этом имеется два соединения с провайдером – быстрое ADSL и медленное модемное. Желательно один компьютер (с адресом 192.168.1.10) направить в глобальную сеть через модем, а два других (с адресами 192.168.1.20 и 192.168.21) через ADSL. Трафик, направленный во внешний мир с других компьютеров, желательно перенаправлять на сниффер, расположенный по адресу 192.168.1.100, причем сниффер может располагаться и на данном компьютере (tcpdump -i ethX). Схема подключения примерно такова:



Просмотрим сетевые карты на сервере, список будет примерно таким:

```
# ip link list
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc ↵
pfifo_fast qlen 10
    link/ether 48:54:e8:01:ef:56 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
3764: ppp0: <POINTTOPOINT,MULTICAST,NOARP,UP> mtu 1492 ↵
qdisc pfifo_fast qlen
    link/ppp
    inet 213.59.72.1 peer 213.59.72.48/24 scope global ppp0
3765: ppp1: <POINTTOPOINT,MULTICAST,NOARP,UP> mtu 1492 ↵
qdisc pfifo_fast qlen
    link/ppp
    inet 213.59.72.2 peer 213.59.72.10/24 scope global ppp1
```

Очевидно, что ppp0 соотвествует модемному соединению, а ppp1 – ADSL-соединению. Просмотрим таблицы маршрутизации (здесь идет отображение всех таблиц маршрутизации ядра, ядро принимает решение о применении той или иной таблицы на основании адреса источника пакета, утилита route способна оперировать только с таблицой main и local, iproute2 дает возможность создавать собственные таблицы, что будет описано несколько позже):

```
# ip rule list

0: from all lookup local
32766: from all lookup main
32767: from all lookup default
```

Как видно, пока наши таблицы применимы ко всем пакетам. Добавим новую таблицу для машин, связанных с Интернетом через ADSL:

```
# echo 200 inet_adsl >> /etc/iproute2/rt_tables
```

Эта команда требует некоторого пояснения: номер 200 выбран произвольно, главное, чтобы он не совпадал с другими номерами таблиц маршрутизации, имя inet_adsl также дается произвольно, но потом этой таблицей можно управлять по имени, так что в ваших же интересах дать понятное имя таблице, дабы облегчить себе процесс дальнейшей настройки.

Добавим в таблицу правила приема пакетов:

```
# ip rule add from 192.168.1.20 table inet_adsl
# ip rule add from 192.168.1.21 table inet_adsl
```

Эти команды, думаю, являются понятными, поэтому сразу просмотрим наши таблицы маршрутизации:

```
# ip rule list
0: from all lookup local
32764: from 192.168.1.20 lookup inet_adsl
32765: from 192.168.1.21 lookup inet_adsl
32766: from all lookup main
32767: from all lookup default
```

Теперь необходимо добавить маршрутизатор по умолчанию для таблицы inet_adsl, тогда все пакеты от необходимых машин будут направляться к заданному шлюзу:

```
# ip route add default via 213.79.52.10 dev ppp1 table ↵
inet_adsl
```

После этого необходимо сбросить кеш маршрутизатора:

```
# ip route flush cache
```

Теперь очередь настроить модемное соединение. Думаю, следующие команды не должны вызвать сложности:

```
# echo 201 inet_modem >> /etc/iproute2/rt_tables
# ip rule add from 192.168.1.10 table inet_modem
# ip route add default via 213.79.52.48 dev ppp0 table ↵
inet_modem
# ip route flush cache
```

Для просмотра таблиц маршрутизации можно использовать команду:

```
# ip route list [table table_name]
```

Теперь необходимо включить сниффер для отслеживания пакетов, которые пришли из локальной сети. Добавим виртуальную сетевую карту:

```
# ifconfig eth0:1 192.168.1.100 up
```

И настроим правила маршрутизации так, чтобы пакеты с локальной сети, направленные во внешнюю сеть, направлялись на адрес 192.168.1.100, т.е. чтобы администратор мог наблюдать за попытками выхода во внешнюю сеть. Эта проблема не так тривиальна, как предыдущая, но решение все-таки существует. Задача решается интеграцией возможностей netfilter (iptables) и iproute2. Внутри ядра существует возможность установки на пакетах меток (метки устанавливает iptables, но учтите, что эти метки существуют только в пределах ядра, и не выходят за границы данного компьютера). Подробное описание системы netfilter выходит за рамки данной статьи, поэтому я ограничусь описанием процесса установки метки на конкретном примере:

```
# iptables -A PREROUTING -i eth0 -s 192.168.1.0/24 -d ! ↵
192.168.1.0/24
           -t mangle -j MARK --set-mark 2
```

Некоторые комментарии: обратите внимание на флаг -j MARK и --set-mark: последний флаг может устанавливать метку от 1 до 255.

После установки правила iptables неоходимо вновь вернуться к настройке iproute2. Учтите, что сейчас все необходимые нам пакеты помечены меткой 2, осталось

только направить все такие пакеты на сниффер, расположенный по адресу 192.168.1.100:

```
# echo 202 sniffing >> /etc/iproute2/rt_tables
# ip rule add fwmark 2 table sniffing
```

Заметьте, эта строка выполняет выборку пакетов согласно их метке.

```
# ip route add default via 19.168.1.100 dev eth0:1 table ↵
sniffing
# ip route flush cache
```

Запускаем собственно сниффер (в фоновом режиме):

```
# tcpdump -i eth0:1 > /var/log/tcpdump.log &
```

При этом необходимо позаботиться о правильной установке прав доступа к файлу дампа, установите правильную umask или установите атрибуты вручную:

```
# touch /var/log/tcpdump.log
# chattr 600 /var/log/tcpdump.log
```

Для повышения безопасности можно также запустить сниффер через chroot:

```
(chroot /var/log tcpdump -i eth0:1)
```

но обычно это делается в инициализационном скрипте.

Есть ещё несколько нюансов в данном примере, а именно установки сетевых опций ядра. Опции ядра обычно устанавливаются посредством файловой системы /proc занесением необходимых значений в определенные файлы. Для нас необходимо отключить icmp redirect ответы, чтобы наш маршрутизатор не сообщал клиентам о выборе необходимого маршрута непосредственно (это лишит нас возможности установки меток на пакеты, а кроме того, понимается далеко не всеми клиентами по умолчанию). Для этого делаем следующее:

```
# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
# echo 0 > /proc/sys/net/ipv4/conf/default/send_redirects
# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

Не забывайте также о правильной настройке брандмауэра, а также, если имеется несколько подсетей, желательно убедиться, что выключена прямая передача пакетов из подсети в подсеть (т.е. если пакет направлен в другую подсеть, он не должен передаваться на другой сетевой интерфейс без обработки):

```
# echo 0 > /proc/sys/net/ipv4/ip_forward
```

Единственный серьезный минус приведенной схемы – возможность подмены IP-адреса. К сожалению, этот недостаток исправить невозможно, но можно дополнительно отслеживать обращения ко внешней сети. На этом я завершу описание этой «простенькой» задачки для администратора и перейду к описанию установки IP-туннелей.

Вообще любой сетевой туннель выполняет инкапсуляцию пакетов (фактически к каждому пакету добавляется необходимый заголовок). Туннели позволяют организовать связь нескольких подсетей одним соединением. В ядро Linux интегрирована поддержка нескольких типов IP-туннелей. Управление туннелями осуществляется посредством команды ip tunnel. Но для начала необходимо включить поддержку туннелей в ядре. На странице Networking options отмечаем следующие опции:

- IP: tunneling – поддержка туннелей ядром;
- IP: GRE tunnels over ip – поддержка GRE-туннелей, которые обладают возможностью инкапсулировать IPv6-трафик, кроме этого, GRE является стандартом де-факто в маршрутизаторах Cisco, поэтому для организации туннеля между Linux-машиной и маршрутизатором Cisco применяйте GRE-туннели.

Представим себе организацию туннеля между двумя компьютерами и соединяющем две подсети:

Для добавления GRE-туннеля можно воспользоваться следующими: командами на сервере 192.168.1.1:

```
# ip tunnel add tuna mode gre remote 192.168.2.1 local ↵
192.168.1.1 ttl 255
```

Эта команда задает GRE-туннель от машины 192.168.1.1 до машины 192.168.2.1, для создания IPV6-туннеля используется тип sit (mode sit), при этом необходимо вручную добавлять IPV6-адрес туннелю (ip --6 addr add IPv6_addr dev tunsit). Учтите, что вы можете добавлять туннель с любым именем, состоящим из букв и цифр. Поле ttl является необязательным, но каждому пакету, проходящему через туннель будет присваиваться заданный ttl.

Вторым этапом настройки туннеля является настройка маршрутизации через этот туннель: включаем виртуальный сетевой интерфейс, созданный предыдущей командой:

```
# ip link set tuna up
```

теперь необходимо назначить созданному туннелю IP-адрес:

```
# ip addr add 192.168.1.101 dev tuna
```

добавляем маршрут к сети 192.168.2.0/24 через созданный туннель:

```
# ip route add 192.168.2.0/24 dev tuna
```

Последнее действие можно выполнить и с помощью старой утилиты route:

```
route add -net 192.168.2.0 netmask 255.255.255.0 dev tuna
```

но синтаксис iproute, на мой взгляд, несколько проще. На другом конце туннеля (192.168.2.1) проделываем подобные действия:

```
# ip tunnel add tunb mode gre remote 192.168.1.1 local ↵
192.168.2.1 ttl 255
# ip link set tunb up
# ip addr add 192.168.2.101 dev tunb
# ip route add 192.168.1.0/24 dev tunb
```

После этого туннель начинает функционировать. Учтите также, что к данным, проходящим по туннелю, дописывается дополнительный заголовок 20 байт длиной, таким образом, MTU для туннеля составляет не 1500, а 1480 байт. Для решения этой проблемы несколько модифицируем команду добавления маршрута, указав mtu:

```
# ip route add 192.168.2.0/24 dev tuna mtu 1480
```

Явное указание mtu – очень полезная вещь во многих случаях, например, при организации VLAN (IEE802.1q) также необходимо уменьшать значение MTU интерфейса.

Если планируется организовать туннель с маршрутизатором CISCO, то его конфигурация может выглядеть следующим образом:

```
interface Tunnel1
description IP tunnel
no ip address
no ip directed-broadcast
ip address 192.168.2.101/24
tunnel source Serial0
tunnel destination 192.168.1.101
tunnel mode ipip
ip route 192.168.1.0/24 Tunnel1
```

Итак, подведём итог. Для выполнения статической маршрутизации лучше всего подходит iproute2 для GNU/Linux. Маршрутизация позволяет выполнять достаточно сложные операции по передаче пакетов, при этом возможно грамотно установить политику доступа к определённым подсетям и узлам сети. Одним из наиболее полезных в практическом плане инструментов оптимизации сетевых операций является управление очередями устройств, но это предмет моей следующей статьи, которую вы сможете, скорее всего, найти в следующем номере журнала. Для установки маршрутизатора не требуется мощного компьютера, в некоторых случаях достаточно floppy-дистрибутива Linux. Одним из таких дистрибутивов является ориентированный на маршрутизацию дистрибутив linuxrouter (www.linuxrouter.org). Он построен на базе ядра 2.2 и 2.0, что является приемлемым вариантом для построения маршрутизатора (включает iproute2, но, к сожалению, я не нашел в составе дистрибутива утилиты tc).

Если же в вашей сети несколько маршрутизаторов или структура сети является непостоянной, то лучшим выбором является установка динамического маршрутизатора, имеющего возможность автоматического обновления маршрутных таблиц. Для любителей маршрутизаторов Cisco могу посоветовать роутер Zebra, эмулирующий синтаксис Cisco IOS. Ну вот и все, разговор о пакете IProute будет продолжен в следующем номере. Приведу список полезных ссылок:

http://www.lartc.org – Linux Advanced Routing and Traffic Control HOWTO – обязательный документ, помогающий грамотно настроить маршрутизатор.

http://www.linuxrouter.org – floppy-дисрибутив GNU/Linux, ориентированный на маршрутизацию.

http://www.opennet.ru – большая подборка документации о маршрутизации.

Брокмайер, Лебланк, Маккарти. «Маршрутизация в Linux». Издательский дом «Вильямс», 2002. В книге рассмотрены общие вопросы маршрутизации и настройки демона динамической маршрутизации gated.

PS: Сейчас ведутся активные споры по наименованию Linux. Так как эта операционная система базируется на ПО GNU, то было предложено называть её GNU/Linux, поэтому далее я буду называть операционную систему GNU/Linux, а ядро – Linux, что соответствует истине.