Настройка сети

printable page

6.1 - Прежде, чем мы пойдем дальше

Для понимания основ этого документа, вам поможет, если бы вы прочитали бы и по крайней мере частично понимали бы секции FAQ Конфигурация Ядра и Установка, и ifconfig(8) и netstat(1) страницы руководства.

Если Вы - сетевой администратор, и вы настраиваете протоколы маршрутизации, если вы используете ваш блок OpenBSD в качестве маршрутизатора, если вам нужно тщательно изучить организацию сетей IP, вам действительно нужно прочитать Осмысление Адресации IP (Understanding IP Addressing). Это - отличный документ. "Understanding IP Addressing" содержит фундаментальное знание для расчетов при работе с сетями IP, особенно когда вы имеете дело или отвечаете за более, чем одну сеть.

Если вы работаете с приложениями такими так веб серверы, ftp серверы, и почтовыми серверами, вы можете извлечь существенную пользу, читая RFC. Наиболее вероятно, что вы не можете прочитать все из них. Выберите некоторые темы, которые вас заинтересовали, или те, которые вы используете в вашей сетевой среде. Поищите их, поймите, как они намереваются работать. RFC определяет много (тысячи)стандартов для протоколов Internet и как они должны работать.

6.2 - Настройка сети

Как правило, OpenBSD инициализирует процесс конфигурации при установке. Тем не менее, хорошо бы разобраться в этом процессе и как оно вообще работает. Все сетевые настройки хранятся в плоских текстовых файлах в директории /etc.

6.2.1 - Обнаружение и настройка ваших сетевых интерфейсов

В OpenBSD, интерфейсы называются по имени типа карты, а не типа соединения. Вы можете увидеть, как ваша сетевая карта инициализируется в течение процесса загрузки или после процесса загрузки, используя команду dmesg(8). У вас также есть шанс увидеть имя вашего сетевого интерфейса, используя команду ifconfig(8). Например, здесь вывод dmesg для сетевой карты Intel Fast Ethernet, который использует устройство именуемое fxp.

```
fxp0 at pci0 dev 10 function 0 "Intel 82557" rev 0x0c: irq 5, address 00:02:b3:2b:10:f7
inphy0 at fxp0 phy 1: i82555 10/100 media interface, rev. 4
```

Если вы не знаете имени вашего устройства, пожалуйста посмотрите на список поддерживаемой аппаратуры для вашей платформы. Вы найдете список множества распространенных имен карт и их OpenBSD имена устройств. Объедините короткое символьное имя устройства (например fxp) с числом назначенным ядром и вы получите имя интерфейса (например fxp0). The number is assigned based on various criteria, depending upon the card and other details of the system. Some cards are assigned by the order they are found during bus probing. Others may be by hardware resource settings or MAC address.

Вы можете обнаружить, что сетевые интерфейсы определяются, используя утилиту ifconfig(8). Следующая команда покажет все

сетевые интерфейсы в системе. Этот пример показывает что имеется только один физический Ethernet интерфейс fxp(4).

```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
lo1: flags=8008<LOOPBACK,MULTICAST> mtu 33224
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
       address: 00:04:ac:dd:39:6a
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::204:acff:fedd:396a%fxp0 prefixlen 64 scopeid 0x1
pflog0: flags=0<> mtu 33224
pfsync0: flags=0<> mtu 2020
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
sl1: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
ppp1: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
tun0: flags=10<POINTOPOINT> mtu 3000
tun1: flags=10<POINTOPOINT> mtu 3000
enc0: flags=0<> mtu 1536
bridge0: flags=0<> mtu 1500
bridge1: flags=0<> mtu 1500
vlan0: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
vlan1: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
gre0: flags=9010<POINTOPOINT,LINK0,MULTICAST> mtu 1450
carp0: flags=0<> mtu 1500
carp1: flags=0<> mtu 1500
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif1: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif2: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif3: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
```

Как вы можете увидеть здесь, ifconfig(8) дает нам намного больше информации чем, нам нужно в этом пункте. Но, это все еще позволяет нам увидеть наш интерфейс. В вышеуказанном примере, карта интерфейса уже сконфигурирована. Это очевидно, поскольку сеть IP уже сконфигурирована на fxp0, это следует из значения "inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255". Также, **UP** и **RUNNING** флаги установлены.

Наконец, вы обратите внимание, что несколько других интерфейсов включены по умолчанию. Эти - виртуальные интерфейсы, которые обслуживают различные функции. Они описаны на следующих страницах руководства:

- lo – Локальный интерфейс
- pflog - Регистрирующий интерфейс пакетного Фильтра
- sl - Сетевой интерфейс SLIP
- ppp – Протокол Точка - Точка
- tun - Туннельный сетевой интерфейс
- enc - Инкапсулированный интерфейс
- bridge - Интерфейс Моста Ethernet
- vlan - Интерфейс Инкапсуляции IEEE 802.1Q
- gre - GRE/MobileIP Интерфейса Инкапсуляции
- gif - Туннельный Интерфейс Generic IPv4/IPv6
- carp – Интерфейс Избыточного Протокола Общего Адреса (Common Address Redundancy Protocol)

Если ваш интерфейс не сконфигурирован на этапе загрузки с помощью файла /etc/hostname.if(5), первый шаг - это создать файл /etc/hostname.xxx, где имя вашего интерфейса заменит "xxx". Из информации из вышеприведенного примера, имя должно быть */etc/hostname.fxp0*.

Формат этого файла прост:

```
address_family address netmask broadcast [other options]
```

Более детально о формате этого файла можете найти в страницах руководства hostname.if(5). Вам рекомендуеятс ознакомиться и с более сложными конфигурациями.

Типичный интерфейс конфигурационного файла, сконфигурированного для адресов IPv4, должен выглядеть похожим на это:

```
$ cat /etc/hostname.fxp0
inet 10.0.0.38 255.255.255.0 NONE
```

в данном случае мы задаем адрес IPv4 (inet), IP адрес 10.0.0.38, маска подсети 255.255.255.0, широковещательный адрес не указан (будет установлен для данного случая по умолчанию в 10.0.0.255).

Вы должны бы также определить типы носителя для Ethernet, например, если бы вы захотели бы установить принудительно полный дуплексный режим 100baseTX.

```
inet 10.0.0.38 255.255.255.0 NONE media 100baseTX mediaopt full-duplex
```

(Разумеется, работа в полно-дуплексном режиме невозможна, если этот режим не установлен на обеих сторонах соединения! При отсутствии специальных требований, установка параметров носителя может быть исключена.A more likely case might be to force 10base-T or half duplex when your infrastructure requires it.)

Или, вы можете захотеть использовать специальные флаги специфические на определенных интерфейсах. Формат файла hostname сильно не изменится!

```
$ cat /etc/hostname.vlan0
inet 172.21.0.31 255.255.255.0 NONE vlan 2 vlandev fxp1
```

6.2.2 - Шлюз по умолчанию

Поместите IP-адрес вашего шлюза в /etc/mygate. Это позволит устанавливать его при загрузке. Этот файл содержит всего одну строку, указывающую адрес шлюза:

```
10.0.0.1
```

Возможно указать и символические имя, однако следует иметь ввиду: может случиться, что ваша служба разрешения доменных имен, резолвер, будет сконфигурирован или доступен только ПОСЛЕ установки шлюза по умолчанию. Поэтому лучше указывать IP адрес или что-нибудь указанное в файле */etc/hosts*

6.2.3 - Разрешение DNS

Служба разрешения доменных имен управляется конфигурационным файлом /etc/resolv.conf. Ниже пример файла:

```
search example.com
nameserver 125.2.3.4
nameserver 125.2.3.5
lookup file bind
```

В этом примере доменное имя по умолчанию example.com, имеются два сервера DNS с адресами 125.2.3.4 и 125.2.3.5, файл /etc/hosts будет использован до получения информации от DNS серверов.

Если вы пользуетесь DHCP, почитайте пункт 6.4 - DHCP использование resolv.conf.tail(5).

6.2.4 - Имя хоста

каждая UNIX машина имеет имя. В OpenBSD имя задается как "Fully Qualified Domain Name" (FQDN) одной строкой в файле /etc/myname. если машина имеет имя "puffy" и находится в домене "example.com" файл будет содержать следующее:

```
puffy.example.com
```

6.2.5 - Активация изменений

Для этого вы можете или перезагрузиться или запустить скрипт **/etc/netstart**. Можно это сделать просто напечатав (имея права root):

```
# sh /etc/netstart
writing to routing socket: File exists
add net 127: gateway 127.0.0.1: File exists
writing to routing socket: File exists
add net 224.0.0.0: gateway 127.0.0.1: File exists
```

Обратите внимание на то, что появилось несколько ошибок. Выполняя этот сценарий, вы изменяете существующие настройки. Например, некоторые маршруты уже присутствуют в таблице маршрутизации ядра. Начиная с этого момента система должна работать. Кроме того, вы можете убедиться, что ваш интерфейс был установлен правильно с помощью ifconfig(8).

Хотя в OpenBSD возможна полная переконфигурация сетевых настроек системы без перезагрузки, рестарт системы ОЧЕНЬ рекомендуется после внесения значительного объема изменений. Причина этого сотоит в том, что окружающая нашу систему среда может оказаться при загрузке несколько иной, чем когда система полностью запущена и выполняется. К примеру, вы указали DNS-сервер символическим именем во всех файлах, и у вас все работает после переконфигурации, однако после перезагрузки внешний DNS-сервер может оказаться недоступным и конфигурирование сети окажется невыполненным.

6.2.6 - Проверка маршрутизации

Вы можете также проверить ваши маршруты через netstat(1) или route(8). Если у вас есть проблемы маршрутизации, вы можете захотеть использовать флаг -n для команды route(8), который печатает адреса IP, не делая DNS преобразование и имен хостов. Вот пример просмотра ваших таблиц маршрутизации, используя обе программы.

```
$ netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags     Refs      Use    Mtu  Interface
default            10.0.0.1           UGS          0       86      -  fxp0
127/8              127.0.0.1          UGRS         0        0      -  lo0
127.0.0.1          127.0.0.1          UH           0        0      -  lo0
10.0.0/24          link#1             UC           0        0      -  fxp0
10.0.0.1           aa:0:4:0:81:d      UHL          1        0      -  fxp0
10.0.0.38          127.0.0.1          UGHS         0        0      -  lo0
224/4              127.0.0.1          URS          0        0      -  lo0

Encap:
Source             Port  Destination        Port  Proto SA(Address/SPI/Proto)

$ route show
Routing tables

Internet:
Destination      Gateway            Flags
default          10.0.0.1           UG
127.0.0.0        LOCALHOST          UG
localhost        LOCALHOST          UH
10.0.0.0         link#1             U
10.0.0.1         aa:0:4:0:81:d      UH
10.0.0.38        LOCALHOST          UGH
BASE-ADDRESS.MCA LOCALHOST          U
```

6.2.7 - Настройка OpenBSD для работы в качестве шлюза

Это - основная информация, которая вам нужна для настройки вашего OpenBSD в качестве шлюза (также называемым маршрутизатором). Если вы используете OpenBSD в качестве маршрутизатора Internet, мы напоминаем, чтобы вы также прочитали инструкции установки пакетного фильтра ниже, чтобы блокировать потенциально злонамеренный трафик. Также, из-за низкой доступности адресов IPv4 у провайдеров услуг и местных регистраторов, вы можете захотеть посмотреть на Сетевую Трансляцию Адресов (NAT) для информации о сохранении вашего адресного пространства IP.

GENERIC ядро уже имеет собрано с поддержкой IP Forwarding, но нужно включить ее. Вы можете сделать это, используя утилиту sysctl(8). Чтобы изменения стали постоянными вы должны отредактировать файл /etc/sysctl.conf для разрешения IP Forwarding. Чтобы сделать это добавьте эту строку в этот файл конфигурации.

```
net.inet.ip.forwarding=1
```

Чтобы сделать эти изменения, не перезагружаясь, вы должны использовать утилиту sysctl(8) непосредственно. Запомните - эти изменения не будет действовать после перезагрузки, и должны быть запущены вновь администратором root.

```
# sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
```

Теперь модифицируйте маршруты на других хостах на обеих сторонах. Есть много возможностей использования OpenBSD в качестве маршрутизатора, используя программное обеспечение такое как OpenBSD'ный собственный OpenBGPD, routed(8), mrtd, zebra, и quagga. OpenBSD имеет поддержку в коллекции портов для zebra, quagga, and mrtd. OpenBGPD и routed установлены в качестве части базовой системы. OpenBSD поддерживает некоторые T1, HSSI, ATM, FDDI, Ethernet, и последовательные (PPP/SLIP) интерфейсы.

6.2.8 - Настройка алиасов для интерфейса

OpenBSD имеет простой механизм для установки IP псевдонимов (алиасов) на интерфейсе. Чтобы сделать это просто отредактируйте файл /etc/hostname.<if>. Этот файл будет считан в процессе загрузки сценарием /etc/netstart(8), который является частью иерархии запуска rc. Для примера, мы предполагаем, что пользователь имеет интерфейс dc0 и находится в сети 192.168.0.0. Другая важная информация:

IP for dc0 is 192.168.0.2
NETMASK is 255.255.255.0

Немного замечаний относительно алиасов. В OpenBSD вы используете только имя интерфейса. Нет различия между первым и вторым псевдонимами. В отличие от некоторых других операционных систем, OpenBSD не ссылается на них как dc0:0, dc0:1. Если вы ссылаетесь на специфический адрес IP псевдонима с помощью ifconfig или добавляете псевдоним, не забываете набирать "ifconfig int alias" вместо просто "ifconfig int" в командной строке. Вы можете удалить псевдонимы с помощью "ifconfig int delete".

Допустим, что Вы используете многочисленные адреса IP, которые - в той же подсети IP с псевдонимами, ваша установка netmask для каждого псевдонима становится 255.255.255.255. Им не нужно следовать за netmask первого связанного IP интерфейса. В этом примере, /etc/hostname.dc0, два псевдонима добавлены к устройству dc0, которое, между прочим, было сконфигурировано как 192.168.0.2 netmask 255.255.255.0.

```
# cat /etc/hostname.dc0
inet 192.168.0.2 255.255.255.0 media 100baseTX
inet alias 192.168.0.3 255.255.255.255
inet alias 192.168.0.4 255.255.255.255
```

Как только вы сделали этот файл, можно просто перезагрузиться, чтобы изменения вступили в силу. Вы можете, тем не менее, задать псевдонимы вручную, используя утилиту ifconfig(8). Чтобы образовать первый псевдоним, вы должны использовать команду:

```
# ifconfig dc0 inet alias 192.168.0.3 netmask 255.255.255.255
```

(опять-таки рекомендуется перезагрузить систему, чтобы убедиться в корректности ваших настроек!)

Чтобы посмотреть эти псевдонимы, Вы должны использовать команду:

```
$ ifconfig -A
dc0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
        media: Ethernet manual
        inet 192.168.0.2  netmask 0xffffff00 broadcast 192.168.0.255
        inet 192.168.0.3 netmask 0xffffffff broadcast 192.168.0.3
```

Как настроить фильтрацию и использовать OpenBSD в качестве файрволла?

Пакетный Фильтр (далее именуемый PF) является системным для OpenBSD для TCP/IP трафика и Сетевой Трансляции Адресов (NAT). PF также допускает нормализацию, и ограничение TCP/IP трафика, и обеспечение контроля пропускной способности, и присваивание приоритетов пакетам, и может быть использован, чтобы создавать мощные и и гибкие файрволы. Это описано в Руководстве Пользователя PF.

6.4 Динамический Протокол Конфигурации Хостов (DHCP)

Динамический Протокол Конфигурации Хостов - способ "автоматической" конфигурации сетевых интерфейсов. OpenBSD может быть сервером DHCP (конфигурируя другие машины), клиентом DHCP (сконфигурированной другой машиной), и, в некоторых случаях, может сочетать оба случая.

6.4.1 - Клиент DHCP

Для того, чтобы использовать клиент DHCP dhclient(8), включенного в OpenBSD, отредактируйте /etc/hostname.xl0 (в предположении, что ваш основной интерфейс ethernet является xl0. Ваш может быть ep0 или fxp0 или какой-нибудь другой.) Все что вам нужно - это положить файл hostname - 'dhcp':

```
# echo dhcp > /etc/hostname.xl0
```

Это заставит OpenBSD автоматически запускать клиента DHCP при загрузке. OpenBSD заберет свои адрес IP, шлюз по умолчанию, и серверы DNS от сервера DHCP.

Если вы хотите запустить клиент DHCP из командной строки, убедитесь, что существует файл /etc/dhclient.conf, а затем пытайтесь:

```
# dhclient fxp0
```

Где fxp0 – интерфейс, на котором вы хотите получить DHCP.

Независимо от того, как вы запускаете клиента DHCP, вы можете отредактировать файл /etc/dhclient.conf, чтобы не корректировать ваш DNS согласно планам сервера dhcp, , сначала раскомментируйте строки 'request' в нем (эти строки пример установок по умолчанию, но вам нужно раскомментировать их, чтобы перезаписать значения dhclient по умолчанию.)

```
request subnet-mask, broadcast-address, time-offset, routers,
      domain-name, domain-name-servers, host-name, lpr-servers, ntp-servers;
```

и затем удалите domain-name-servers. Конечно, вы можете захотеть удалить hostname, или другие установочные параметры тоже.

Изменяя опции в вашем файле dhclient.conf(5), вы сообщаете клиенту DHCP как формировать ваш файл resolv.conf(5). Клиент DHCP перезаписывает любую информацию, которую вы уже имеете в resolv.conf(5) на информацию, которая извлекается от сервера DHCP. Следовательно, вы потеряете любые изменения, которые вы делали вручную в resolv.conf.

Есть два доступных механизма избежать этого:

OPTION MODIFIERS (default, supersede, prepend, and append))(по умолчанию, замена, добавление в начало и в конец), позволят вам перезаписать любую из опций dhclient.conf(5).
resolv.conf.tail(5) позволяет вам добавлять все, что вы хотите в файл resolv.conf(5) созданный утилитой dhclient(8).

Пример должен быть, если вы используете DHCP, но вы хотите добавить lookup file bind в resolv.conf(5) созданный утилитой dhclient(8). Для этого нет опции в dhclient.conf, так что вы должны использовать resolv.conf.tail, чтобы сохранять это.

```
# echo "lookup file bind" > /etc/resolv.conf.tail
```

Теперь ваш resolv.conf(5) должен включить "lookup file bind" в конце.

```
nameserver 192.168.1.1
nameserver 192.168.1.2
lookup file bind
```

6.4.2 - Сервер DHCP

Если Вы хотите использовать OpenBSD в качестве сервера DHCP dhcpd(8), отредактируйте /etc/rc.conf.local, чтобы он содержал строку dhcpd_flags="". Поместите интерфейсы, которые вы хотите, чтобы dhcpd прослушивал, в /etc/dhcpd.interfaces.

```
# echo xl1 xl2 xl3 >/etc/dhcpd.interfaces
```

Затем, отредактируйте /etc/dhcpd.conf. Опции в достаточной степени очевидны.

```
option  domain-name "example.com";
option  domain-name-servers 192.168.1.3, 192.168.1.5;

subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers 192.168.1.1;

        range 192.168.1.32 192.168.1.127;
}
```

Это сообщит вашим клиентам DHCP, чтобы domain = example.com добавлялся к запросам DNS (так, если пользователь набирает 'telnet joe', тогда он пошлется на joe.example.com) Это укажет им на серверы DNS 192.168.1.3 и 192.168.1.5. Для хостов, которые находятся в той же сети, что и ethernet интерфейс машины OpenBSD, который лежит в диапазоне 192.168.1.0/24, он назначит им адрес IP между 192.168.1.32 и 192.168.1.127. Он установит их шлюз по умолчанию в качестве 192.168.1.1.

Если Вы хотите запустить dhcpd(8) из командной строки, после редактирования /etc/dhcpd.conf, попытайтесь:

```
# touch /var/db/dhcpd.leases
# dhcpd fxp0
```

Строка с touch нужна для создания пустого файла dhcpd.leases перед тем как dhcpd(8) сможет запуститься. Сценарии запуска OpenBSD создает этот файл, если нужно в процессе загрузки, но если вы запускаете dhcpd(8) вручную, вы должны создать его сначала. fxp0 - интерфейс на котором вы хотите начать обслуживать DHCP.

Если вы предоставляете сервис DHCP для Windows, вы можете захотеть, чтобы dhcpd(8) давал клиенту адрес 'WINS' сервера. Для этого, просто добавьте следующую строку к вашему /etc/dhcpd.conf:

```
option    netbios-name-servers    192.168.92.55;
```

(где 192.168.92.55 - IP вашего Windows или Samba сервера.) Смотри dhcp-options(5) для подробностей об опциях, которые могут понадобиться вашим клиентам DHCP.

6.5 - PPP

The Point to Point Protocol (PPP) (Протокол точка - точка) - обычно используется, для создания соединения с вашим провайдером через модем. В OpenBSD существует 2 пути, чтобы сделать это:

pppd(8) - PPP демон ядра ppp(8) - PPP демон уровня пользователя

Как ppp так и pppd выполняют аналогичные функции различными путями. pppd работает с драйвером ядра ppp(4), тогда как ppp работает на уровне пользователя с tun(4). Этот документ рассказывает только о PPP демоне пользовательского уровня, так как его легче отлаживать и взаимодействовать с ним. Для начала вам нужна некоторая простая информация о вашем провайдере. Вот список полезной информации, которая вам понадобится.

Ваш наборный номер провайдера
Ваш DNS сервер
Ваше имя пользователя и пароль
Ваш шлюз

Без некоторой из них вы можете обойтись, но все же она будет полезна в установке ppp. В качестве конфигурационного файла PPP демон пользовательского уровня использует файл /etc/ppp/ppp.conf. . Много полезных файлов вы найдете просматривая каталог /etc/ppp, которые содержат различные установки для многих различных ситуаций. Вы должны изучить содержимое этого каталога.

Начальная настройка PPP(8)

Начальная установка для PPP демона пользовательского уровня состоит из редактирования вашего файла */etc/ppp/ppp.conf*. Этот файл не существует по умолчанию, но есть файл */etc/ppp/ppp.conf.sample*, который вы можете просто отредактировать, чтобы создать ваш собственный файл *ppp.conf*. Здесь я начну с самой простой и вероятно наиболее используемой установки. Вот простенький файл *ppp.conf*, который устанавливает некоторые значения по умолчанию:

```
default:
set log Phase Chat LCP IPCP CCP tun command
set device /dev/cua01
set speed 115200
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
```

Секция после метки default: выполняться всякий раз. Здесь мы установили всю нашу критическую информацию. С помощью " set log" мы задали наш уровень логирования. Он может быть изменен: сошлемся на на ppp(8) для более подробной информации по установке уровней логирования. Наше устройство задается строкой " set device". Это устройство, на котором включен модем. В этом примере, модем находится на 2-ом com порту. Следовательно 1-ый com порт должен быть на */dev/cua00*. С помощью "set speed" мы установили скорость нашего диалап соединения и с помощью "set dial" мы установили наши наборные диалап

параметры. С их помощью мы можем изменить наше время тайм-аута, и т.п.. Эта строка должна оставаться почти такой же не смотря ни на что.

Теперь мы можем идти далее и устанавить информацию специфическую по нашему провайдеру. Мы сделаем это с помощью дополнения другими метками под нашей секцией default:. Эта метка может быть названа как угодно - можете просто использовать имя вашего провайдера. Здесь я использую myisp: в качестве нашей метки, отсылающейся на нашего провайдера. Вот простые настройки, включающие все, что нам нужно для получения нашего соединения:

```
myisp:
set phone 1234567
set login "ABORT NO\\sCARRIER TIMEOUT 5 ogin:--ogin: ppp word: ppp"
set timeout 120
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
add default HISADDR
enable dns
```

Здесь мы установили основную информацию, характеризующую провайдера. Первая опция " set phone " устанавливает ваш наборный номер провайдера. " set login" устанавливает наши опции входа. Здесь мы установили тайм-аут на 5; это означает, что мы прервем нашу попытку входа после 5 секунд, если несущий сигнал не обнаружен. В противном случае ждем отправленного " login:" и посылаем ваше имя пользователя и пароль.

В этом примере наше Имя Username = ppp and Password = ppp. Эти значения должны быть изменены. Строка " set timeout" устанавливает тайм-аут бездействия для соединения в 120 секунд. Строка " set ifaddr " немного мудрена. Вот более исчерпывающее объяснение.

```
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
```

В вышеуказанной строке, мы имеем следующий формат "**set ifaddr [myaddr[/nn] [hisaddr[/nn] [netmask [triggeraddr]]**". Так первый заданный IP – это тот, который мы хотим в качестве нашего IP. Если у вас есть статический адрес IP, установите его здесь. В нашем примере мы используем /0, который сообщает, что никакие биты этого IP адреса не нуждаются в сочетании и целая часть может быть заменена. Второй IP определен тем, что мы ожидаем в качестве их IP. Если вы знаете его вы можете установить его. Кроме того, в нашей строке мы не знаем, что будет назначено, так что мы позволили, чтобы они сообщали нам. Третья опция - нашfa netmask, здесь установлена на 255.255.255.0. Тем не менее, только адрес в дипазоне myaddr будет принят. Это полезно, когда происходит согласование с некоторыми реализациями PPP, которые не назначат IP, если их peer запрашивают ``0.0.0.0.

Следующая опция использовала " "add default HISADDR" для устанавливания нашего маршрута по умолчанию для их IP. Это 'создает проблемы', в смысле, что если их IP должен изменяться, наш маршрут автоматически будет скорректирован. С помощью "enable dns" мы сообщаем нашему провайдеру аутентификации наших nameserver адресов. аНЕ делайте это, если вы запускаете локальный DNS, поскольку ppp просто обойдет его использование, вводя некоторые nameserver строки в */etc/resolv.conf*.

Вместо традиционных методов входа (login), многие провайдеры теперь используют аутентификацию CHAP или PAP. Если это ваш случай, наша конфигурация будет выглядеть немного по-другому:

```
myisp:
set phone 1234567
set authname ppp
set authkey ppp
set login
set timeout 120
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
add default HISADDR
```

```
enable dns
```

В вышеуказанном примере, мы определяем наше имя пользователя (ppp) и пароль (ppp) используя authname и authkey, соответственно. Нет необходимости определять которая из двух аутентификации CHAP или PAP использована - это согласовывается автоматически. "set login" просто определяется для попытки залогиниться, с именем пользователя и паролем определенным прежде.

Использование PPP(8)

Теперь, когда мы настроили наш *ppp.conf* файл, мы можем начать попытки соединия с нашим провайдером. Я опишу подробно некоторые обычно используемые аргументы с ppp:

ppp -auto myisp - Это запустит ppp, сконфигурирует ваши интерфейсы и подключит к вашему провайдеру, затем перейдет в фоновый режим.
ppp -ddial myisp - Это аналогично -auto, но если ваше соединение обрывается, она опытается снова и пересоединится.

Если вышеуказанное не приводит к успеху, попробуйте выполнить */usr/sbin/ppp* без опций - это запустит ppp в диалоговом режиме. Опции могут определяться поодиночке, чтобы проверить на наличие ошибки или других проблем. Используя установку определенную выше, ppp будет вести лог в /var/log/ppp.log. Этот лог, также как и страницы руководства, целиком содержит полезную информацию.

Дополнительные возможности ppp(8)

В некоторых ситуациях вы можете захотеть выполнять команды в момент соединенения или разъединения. Для этих ситуаций вы можете создать два файла: */etc/ppp/ppp.linkup* и */etc/ppp/ppp.linkdown*. конфигурации можно посмотреть здесь:

ppp.linkup
ppp.linkdown

Разновидности ppp(8)

Много провайдеров теперь предлагают xDSL сервисы, которые быстрее, чем традиционные диалап методы. Это включает такие варианты как например, ADSL и SDSL. Хотя никакого физического набора номера не происходит, соединение все еще основана Протоколе Точка - Точка. Примеры:

PPPoE
PPPoA
PPTP

PPPoE/PPPoA

The Point to Point Protocol over Ethernet (PPPoE) (Протокол точка – точка поверх Ethernet), - метод для посылки пакетов PPP в фреймах Ethernet. Point to Point Protocol over ATM (PPPoA), - обычно работает в сетях ATM, как например те, которые

используются в Великобритании и Бельгии.

Обычно это означает, что Вы можете установить соединение с вашим провайдером, используя только стандартную карту Ethernet и основанный на Ethernet DSL модем (в противоположность к только USB модему).

Если у вас есть модем, который общается по протоколу PPPoE/PPPoA, то возможно сконфигурировать модем, чтобы он создавал соединения. Кроме того, если модем имеет режим `bridge', то возможно включить его и имеем модем "pass through" (пропускающий) пакеты в PPPoE программное обеспечение запущенное на машине (смотри ниже).

Основной программный интерфейс на PPPoE/PPPoA в OpenBSD - это pppoe(8), который является реализацией уровня пользователя (почти такой же как и описанный нами ppp(8), выше). PPPoE реализация уровня ядра, pppoe(4), является частью OpenBSD.

PPTP

The Point to Point Tunneling Protocol (PPTP) (Туннельный протокол точка-точка) является собственным протоколом Микрософт. Клиент pptp доступен на интерфейсах с pppd(8) и способен соединяться с основанной на PPTP виртуальными частными сетями (VPN), используемыми некоторыми кабельными и xDSL провайдерами. pptp сам должен быть установлен из пакетов или портов. Дальнейшие инструкции по установке и использованию pptp доступны на страницах руководства, которые устанавливаются вместе с пакетом pptp.

6.6 - Тюнинг параметров сети

Главной целью OpenBSD является создание *РАБОЧЕЙ СИСТЕМЫ*, устраивающей большинство пользователей. Крутить настройки без их понимания - куда скорее путь к поломке системы, нежели к повышению производительности. Всегда начинайте со стандартных настроек, меняйте их только в том случае, если действительно есть какие-либо проблемы.

ОЧЕНЬ МАЛОМУ кому это необходмо!

6.6.1 - Каким образом настроить ядро для того, чтобы увеличить количество повторных попыток и таймауты для сессий TCP?

Вы обычно используете это для разрешения проблем маршрутизации или соединения. Конечно, для этого для наибольшего эффекта, обе стороны соединения должны использовать аналогичные значения.

Для того, чтобы наладить это, используйте sysctl и увеличивайте значения параметров:

```
net.inet.tcp.keepinittime
net.inet.tcp.keepidle
net.inet.tcp.keepintvl
```

Используя sysctl -a, вы можете увидеть текущие значения этих параметров (и множества других). Чтобы изменить один параметр, сделайте так

```
sysctl net.inet.tcp.keepidle = 28800.
```

6.6.2 - Каким образом можно включить направленные широковещательные запросы?

Обычно, вам не надо делать этого. Это позволяет кому-либо посылать трафик на широковещательный адрес(а) вашей присоединенной сети(или сетях), если вы используете ваш OpenBSD в качестве маршрутизатора.

Рассмотрим некоторые случаи, в закрытых сетях, где это может быть полезным, особенно при использовании более старых реализаций протокола NetBIOS. Для этого используется другой параметр sysctl. sysctl net.inet.ip.directed-broadcast=1 включает это. Прочитайте о smurf атаках, если вы хотите узнать почему это выключено по умолчанию.

6.6.3 - Я не хочу, чтобы ядро динамически распределяло определенные порты

Для этого также есть параметр в sysctl. Из sysctl(8):

Установите список резервных портов TCP, которые не должны быть распределены ядром динамически. Это может быть использовано для удержания демонов от захвата специфических портов, которые нужны для функционирования других программ. Элементы списка могут быть разделены запятыми и/или интервалом.

```
#  sysctl net.inet.tcp.baddynamic=749,750,751,760,761,871
```

Также возможно добавить или удалить порты из текущего списка.

```
#  sysctl net.inet.tcp.baddynamic=+748
#  sysctl net.inet.tcp.baddynamic=-871
```

6.6.4 - Как мне увеличить производительность на реально высокоскоростных соединениях с большим трафиком?

Если вы видите ограничение производительности, когда используете высокоскоростное WAN соединение передающее большое количество информации, вы можете наблюдать увеличение производительности изменяя значение следующих параметров sysctls:

```
net.inet.tcp.recvspace
net.inet.tcp.sendspace
```

Попробуйте 65536 вместо значения по умолчанию 16384. Обратите внимание, что от этого будет немного пользы. Не делайте этого до тех пор пока ваша наблюдаемая производительность не станет действительно ниже, чем вы предполагаете.

6.7 - Базовое использование NFS

NFS, или сетевая файловая система, используется для совместного использования файловой системы по сети. Перед установкой сервера NFS прочитайте следующие страницы руководства:

- nfsd(8)
- mountd(8)
- exports(5)

В этом разделе приводятся шаги по простой установке NFS. В этом примере описывается сервер NFS в LAN с клиентами из LAN. Здесь не обсуждаются вопросы безопасности NFS. Мы считаем, что вы уже установили фильтрацию пакетов или другую защиту с помощью сетевого экрана для предотвращения доступа извне.Если вы разрешаете внешний доступ к серверу NFS, вам нужно быть особенно осторожным к сохранности данных, мы рекомендуем использовать IPSec. Иначе ваш трафик NFS будет потенциально доступен для посторонних. Существуют атаки с подменой IP- адреса на на адрес, разрешенный сервером NFS. При правильной настройке IPSec защитит вас от этих видов атак.

Настройка сервера NFS

Данные сервисы должны быть включены и запущены для работы сервера:

- portmap(8)
- mountd(8)
- nfsd(8)

По умолчанию они выключены в OpenBSD. Добавьте следующее в rc.conf.local(8) для включения:

```
portmap=YES
nfs_server=YES
```

Следующим шагом мы отконфигурируем список файловых систем дабы сделать доступным для клиентов монтирование.

В этой установке сервер имеет адрес 10.0.0.1 . Этот сервер NFS обслуживает только клиентов из своей сети. Первый шаг в установке NFS - настройка в файле /etc/exports. Этот файл содержит список файловых систем, доступных через NFS, и определяет, кто имеет к ним доступ. В /etc/exports можно использовать различные опции, читайте о них в exports(5) странице руководства. Пример файла /etc/exports для рассматриваемого нами сервера :

```
#
# NFS exports Database
# See exports(5) for more information.  Be very careful, misconfiguration
# of this file can result in your filesystems being readable by the world.
/work -alldirs -ro -network=10.0.0 -mask=255.255.255.0
```

Это означает, что локальная файловая система /work будет доступна через NFS. Опция -alldirs означает, что клиенты могут монтировать любую точку внутри /work как /work. Допустим, существует каталог /work/monday, клиенты могут монтировать /work (и иметь доступ внутри него ко всем файлам и каталогам) или же могут смонтировать /work/monday, получив доступ к файлам и директориям внутри только оного. Опция -ro задает доступ только на чтение. Последние два аргумента означают, что только клиенты из сети 10.0.0.0 с маской 255.255.255.0 могут монтировать эту файловую систему. Это важно для серверов, которые доступны из различных сетей.

Еще одно замечание относительно безопасности. Не добавляйте файловую систему в /etc/exports без списка разрешенных хостов. Иначе любой, кто может видеть Вашу машину по сети, сможет смонтировать экспортируемые NFS.

Запускаем сервис сервера. Это можно сделать, включив сарвис используя инструкцию ниже и перезагрузку или запустив сервис вручную.

```
# /usr/sbin/portmap
# echo -n >/var/db/mountdtab
# /sbin/mountd
```

```
# /sbin/nfsd -tun 4
```

Параметрами, переданными nfsd, мы можем включить TCP (-t) или UDP (-u) соединения и включить 4 подключения (-n) к запущенному nfsd. Вы должны установить требуемое значение количества подключений к NFS серверу, чтобы обеспечить возможностью обработки максимально требуемого количества запросов клиентов.

Теперь вы готовы к монтированию экспортируемой файловой системы клиентами.

Запомните: Если вы внесли изменения в /etc/exports в то время как NFS уже работает, вы должны сообщить об этом mountd! Просто пошлите ему сигнал HUP:

```
# kill -HUP `cat /var/run/mountd.pid`
```

Монтирование файловой системы

NFS может быть смонтирована клиентом без запуска дополнительных сервисов или демонов. Все это монтируется как любая другая файловая система.

NFS монтируется с помощью mount(8), или, точнее, mount_nfs(8). Для монтирования директории /work хоста 10.0.0.1 в точку монтирования /mnt на локальной машине используйте (помните, не обязательно использовать IP адрес; mount может использовать доменное имя хоста):

```
# mount -t nfs 10.0.0.1:/work /mnt
```

Для монтирования при загрузке добавьте в /etc/fstab примерно следующее:

```
10.0.0.1:/work /mnt nfs rw 0 0
```

важно использовать 0 0 в конце данной строки файловой системы NFS, чтобы ваш компьютер не пытался запустить на ней fsck. При необходимости следует воспользоваться и другими безопасными опциями, как то: noexec, nodev, nosuid. К примеру:

```
10.0.0.1:/work /mnt nfs rw,nodev,nosuid 0 0
```

В этом случае устройства и программы setuid с сервера NFS не смогут нарушить меры безопасности на клиенте NFS. Если вы не планируете запускать программы на клиенте NFS, добавьте noexec в этот список.

When accessing an NFS mount as the root user, the server automatically maps root's access to username "nobody" and group "nobody". This is important to know when considering file permissions. For example, take a file with these permissions:

```
-rw-------    1 root     wheel           0 Dec 31 03:00 _daily.B20143
```

If this file was on an NFS share and the root user tried to access this file from the NFS client, access would be denied. This is because the server uses the credentials of the user "nobody" when root tries to access the file. Since the user nobody doesn't have permissions to access the file, access is denied.

The user and group that root are mapped to are configurable via the exports(5) file on the NFS server.

Получение статистики NFS

Вы можете проверить, насколько правильно работает NFS и все демоны работают и регистрирует RPC. Для этого используется rpcinfo(8).

```
$ rpcinfo -p 10.0.0.1
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    633  mountd
    100005    3   udp    633  mountd
    100005    1   tcp    916  mountd
    100005    3   tcp    916  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
```

В процессе работы есть несколько утилит для просмотра происходящего с NFS. Одна из них showmount(8), которая показывает, какие файловые системы смонтированы и кем они смонтированы. Есть также nfsstat(8), который более подробную статистику. Для использования showmount(8), выполните /usr/bin/showmount -a host. Например:

```
$ /usr/bin/showmount -a 10.0.0.1
All mount points on 10.0.0.1:
10.0.0.37:/work
```

вывод нам показывает что 10.0.0.37 имеет примонтированный /work экспортируемый с сервера 10.0.0.1.

6.9 - Настройка сетевого моста в OpenBSD

Мост - это соединение между двумя или более раздельными сетями. В отличие от маршрутизатора, пакеты предаются между сетями "невидимо/прозрачно" -- логически, две различных сети объединены в один сегмент с обоих сторон сетевого моста. Мост обеспечит транспорт перенаправлением пакетов из одной сети в другую, при этом обеспечив уменьшение трафика за счет перенаправления только тех пакетов, которые необходимы, при этом упростив схему сети.

Заметим, что в силу своей "невидимости/прозрачности", интерфейсы моста могут не иметь собственных IP адресов. В этом случае, интерфейсы могут быть эффективны в двух режимах работы, как часть моста и как обычный, автономый сетевой адаптер. Если никакой интерфейс не имеет IP, мост сможет передавать необходимые данные, но не будет удобен в удаленном управлении (что может использоваться как feature).

Пример применения моста

Мои старые компьютеры в стойке не имеют встроенного сетевого адаптора 10BASE-TX. В то же время имеются AUI или AAUI соединения, у меня нет возможности подключить их по коаксиалу. Одна из машин в стойке - OpenBSD терминал-сервер - имеет постоянное соединение с высокоскоростной сетью. Добавление дополнительной сетевой карты с возможностью подключения коаксиалом позволяет сделать мою машину мостом в коаксиальную сеть.

Теперь моя система имеет два сетевых адаптора, Intel EtherExpress/100 (fxp0) и 3c590-Combo card (ep0) с коаксиальным портом.

fxp0 обеспечивает связь с моей сетью и будет поэтому иметь IP адрес, ep0 используется только как мост и не будет иметь IP адрес. Машины, подключенные к коаксиальному сегменту будут работать как часть моей сети. Как мы это сделаем?

Файл hostname.fxp0 содержит конфигурационную информацию для карты fxp0. Эта машина использует DHCP, таким образом файл содержит:

```
$ cat /etc/hostname.fxp0
dhcp NONE NONE NONE NONE
```

Здесь ничего необычного.

Карта ep0 немного отличается, как мы предполагаем:

```
$ cat /etc/hostname.ep0
up media 10base2
```

Здесь мы указываем системе при активации интерфейса используя ifconfig(8) установить в 10BASE-2 (coax). IP адрес или дополнительные настройки для интерфейса не указываем. Опции карты ep card детально описываются в странице руководства (man) для ep.

Теперь мы должны создать мост. Мост инициализируется при наличии файла вида bridgename.bridge0. Здесь приводится пример для нашего случая:

```
$ cat /etc/bridgename.bridge0
add fxp0
add ep0
up
```

Это означает, что надо создать мост между двумя сетевыми картами, fxp0 и ep0, и активировать его. Имеет значение, в котором карты перечислены? Нет, помните, мост симметричен -- пакеты перенаправляются в обоих направлениях.

Вот так! Перегружаемся и получаем работающий мост.

Фильтрация на мосте

Конечно, такой простой мост тоже может найти применение, но, вероятно, нам захочется ОБРАБОТАТЬ пакеты, поскольку они проходят через ваш мост. Как вы можете ожидать, для фильтрации трафика на мосту может испльзоваться пакетный фильтр PF.

Имейте ввиду, в силу специфики моста, трафик проходит через оба интерфейса, таким образом надо настраивать фильтрацию только на одном интерфейсе. Ваши правила вида "разрешено все" соответствуют нижеприведенному:

```
pass in  on ep0  all
pass out on ep0  all
pass in  on fxp0 all
pass out on fxp0 all
```

Теперь, к примеру, мы желаем фильтровать трафик этих старых машин, пропуская только Web и SSH трафик. В этом случае мы собираемся пропускать весь входящий и исходящий трафик на ep0, фильтруя на fxp0, с использованием keep state для ответного

трафика:

```
# Pass all traffic through ep0
pass in quick on ep0 all
pass out quick on ep0 all

# Block fxp0 traffic
block in  on fxp0 all
block out on fxp0 all

pass in quick on fxp0 proto tcp from any to any port {22, 80} \
     flags S/SA keep state
```

Обратите внимание, что этот набор правил предотвращает что-либо, но поступающий HTTP и SSH трафик от любой достижимой машиной моста или любой из других узлов находится "за" ним. Другие результаты могут быть получены путем фильтрации другого интерфейса.

Для того, чтобы проверять и управлять мостом, который вы создали, используйте команду brconfig(8), которая может также быть использована, чтобы создавать мост после загрузки.

Рекомендации по настройке моста

НАСТОЯТЕЛЬНО рекомендуется настраивать фильтрацию только на одном интерфейсе. Хотя возможно фильтровать на обоих интерфейсах, вам действительно нужно понимать это очень хорошо, чтобы делать все правильно.

Используя опцию *blocknonip* из brconfig(8) или bridgename.bridge0, вы можете предотвращать не-IP трафик (как например, IPX или NETBEUI) от проскальзывания через ваши фильтры. Это может быть важным в некоторых ситуациях, но вы должны понять, что мосты работают для всех типов трафика, не только для IP.

Наведение мостов требует, чтобы сетевые адаптеры были в "Promiscuous mode" (беспорядочный режим) -- они слушают ВЕСЬ сетевой трафик, а не только тот, что направлялся на интерфейс. Это дает более высокую нагрузку на процессор и шину, чем могло бы ожидаться. Некоторые сетевые адаптеры не работают правильно в этом режиме, например чип TI ThunderLAN (tl(4))не может работать как часть моста.

6.10 - Каким образом загрузиться через PXE? (i386, amd64)

Дозагрузочная Среда Выполнения, или PXE (Preboot Execution Environment), - путь загружать компьютер из сети, а не с жесткого диска, флоппи или CD-ROM. Технология первоначально была разработана Intel, но поддерживается теперь большинством основных сетевых карт и компьютерными изготовителями. Обратите внимание, что есть несколько других сетевых протоколов загрузки, PXE сравнительно последний. По традиции, загрузка PXE сделана, используя ПЗУ на сетевых адаптерах или материнской плате системы, но доступны загрузочные флоппи-диски из различных источников, которые также разрешают PXE загрузку. Многие ПЗУ в более старых сетевых адаптерах поддерживают сетевую загрузку но НЕ поддерживают PXE; OpenBSD/i386 или amd64 не может быть к настоящему времени загружен через сеть при использовании этих карт.

Как работает загрузка через PXE?

Сначала разберемся как OpenBSD загружается на i386 и amd64 платформах. При запуске процесса загрузки, PXE-способный сетевой адаптер транслирует DHCP запрос через сеть. Сервер DHCP назначит адаптеру IP адрес, и дает ему(адаптеру) имя файла, который нужно извлечь из tftp(1) сервера и исполнить. Этот файл затем проводит остальную часть процесса загрузки. Для OpenBSD, файл - pxeboot, который заменяет стандартный загрузочный boot(8) файл. pxeboot(8) - затем способен загрузить и выполнить ядро (как например, bsd или bsd.rd) из того же tftp(1) сервера.

Как мне это настроить

Первый и очевидный шаг - Вы должны иметь компьютер способный для PXE загрузки -или сетевой адаптер. Порядочно документации укажет все современные сетевые адаптеры и компьютеры – способные на PXE загрузку, но это - очевидно не истина в последней инстанции -- многие дешевые системы не включают PXE ПЗУ или используют более старый сетевой протокол загрузки. Вам также нужно правильно сконфигурированные сервера DHCP и TFTP

Предположим, машина OpenBSD является источником загрузочных файлов (это НЕ обязательно), ваш dhcpd.conf файл сервера DHCP должен иметь следующую строку:

```
filename "pxeboot";
```

чтобы DHCP сервер попробовал использовать этот файл для загрузки рабочей станции. Например:

```
shared-network LOCAL-NET {
        option  domain-name "example.com";
        option  domain-name-servers 192.168.1.3, 192.168.1.5;

        subnet 192.168.1.0 netmask 255.255.255.0 {
                option routers 192.168.1.1;
                filename "pxeboot";
                range 192.168.1.32 192.168.1.127;
                default-lease-time 86400;
                max-lease-time 90000;
        }
}
```

Вы также должны запустить демон tftpd(8). Это обычно делается через inetd(8). Стандартная OpenBSD поставка имеет строку образца в inetd.conf, которая подходит для вас:

```
#tftp  dgram   udp  wait  root  /usr/libexec/tftpd   tftpd -s /tftpboot
```

в которой просто нужно удалить символ '#' и послать inetd(8) сигнал -HUP, чтобы перезагрузить /etc/inetd.conf. tftpd(8) обслуживает файлы из конкретной директории, в случае этой строки, эта директория - /tftpboot, которую мы используем для этого примера. Очевидно, эта директория должен быть создана и заполнена. Обычно, вам нужно иметь там только несколько файлов для PXE загрузки:

pxeboot, загрузчик PXE загрузки (подходят те же функции как и при загрузке в базовой дисковой системе).
bsd.rd, инсталяционное ядро или bsd, модифицированное пользователем ядро.
/etc/boot.conf, файл конфигурации загрузки.

Обратите внимание, что /etc/boot.conf нужен только в случае, если ядро, которое вы хотите загрузить не называется bsd, или другие установки pxeboot по умолчанию не такие, как вам нужно (например, вы хотите использовать последовательную [serial] консоль). Вы можете протестировать ваш tftpd(8) сервер, используя tftp(1) клиент, убедитесь, что можете скачивать необходимые файлы.

Когда ваши DHCP и TFTP сервера работают, и вы готовы. Вы должны активизировать загрузку PXE на вашей системной или сетевой карте; обратитесь к вашей документации по системе. Как только вы включите этот режим, вы должны увидеть что-то подобное:

```
Intel UNDI, PXE-2.0 (build 067)
Copyright (C) 1997,1998 Intel Corporation

For Realtek RTL 8139(X) PCI Fast Ethernet Controller v1.00 (990420)

DHCP MAC ADDR: 00 E0 C5 C8 CF E1
CLIENT IP: 192.168.1.76  MASK: 255.255.255.0  DHCP IP: 192.168.1.252
GATEWAY IP: 192.168.1.1
probing: pc0 com0 com1 apm pxe![2.1] mem[540k 28m a20=on]
disk: hd0*
net: mac 00:e0:c5:c8:cf:e1, ip 192.168.1.76, server 192.168.1.252
>> OpenBSD/i386 PXEBOOT 1.00
boot>
```

С этого момента, у вас есть стандартное приглашение загрузки OpenBSD. Если вы просто набираете "bsd.rd", просто скачивается файл bsd.rd от сервера TFTP.

```
 >> OpenBSD/i386 PXEBOOT 1.00
boot> bsd.rd
booting tftp:bsd.rd: 4375152+733120 [58+122112+105468]=0x516d04
entry point at 0x100120

Copyright (c) 1982, 1986, 1989, 1991, 1993
        The Regents of the University of California.  All rights reserved.

Copyright (c) 1995-2008 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 4.3 (RAMDISK_CD) #645: Wed Mar 12 11:31:03 MDT 2008
  ...
```

Инсталляционное ядро bsd.rd теперь загрузится.

Могу ли я загрузить другие ядра, кроме bsd.rd через PXE?

Да, хотя с инструментальными средствами к настоящему времени в OpenBSD, загрузка PXE первоначально предназначена для установки OS.

6.11 - Протокол Избыточности Общего Адреса Common Address Redundancy Protocol (CARP)

6.11.1 Что такое CARP и как он работает?

CARP является средством для достижения резервирования на уровне системы, путем объединения многочисленных компьютеров в единственный, виртуальным сетевым интерфейсом между ними, для того чтобы, если какая-либо машина выходит из строя, другая может ответить взамен, и/или допускать уровни распределения нагрузки между системами. CARP является улучшением над стандартом the Virtual Router Redundancy Protocol (VRRP) [Протоколом Избыточности Виртуального Роутера]. CARP был разработан после того как посчитали, что VRRP не достаточно свободен из-за возможно перекрывающего патента Cisco. Для более

подробного ознакомления о началах CARP'а и юридические вопросы, окружающие VRRP, пожалуйста посетите эту страницу.

Для того чтобы избежать юридических конфликтов, Ryan McBride (с помощью Michael Shalayeff, Marco Pfatschbacher и Markus Friedl) разработал CARP, чтобы он коренным образом был другим. Включение криптографии - наиболее выдающееся изменение, но все еще только одно из многих.

Как это работает: CARP является широковещательным протоколом. Он группирует несколько физических компьютеров вместе под одним или более виртуальными адресами. Из них, одна система - мастер и отвечает на все пакеты предназначенные для группы, другие системы действуют как горячее резервирование. Независимо от IP и MAC адреса локального физического интерфейса, пакеты посылаемые на CARP адрес возвращаются с CARP информацией.

AB конфигурируемых интервалах, мастер оповещает о своей работе по IP протоколу номер 112. Если мастер становится offline, другие системы в CARP группе начинают оповещение. Хост, способный оповещать наиболее часто, становится новым мастером. Когда основная система возвращается, она становится обратно хостом по умолчанию, хотя если это - более желательно для одного хоста, чтобы он был мастером там, где это возможно (например, один хост является быстрым Sun Fire V120 и другие - сравнительно медленные SPARCstation IPCs), вы можете так сконфигурировать их.

Пока высоко избыточные и отказоустойчивые аппаратные средства минимизирует потребность в CARP, они не снимают вопрос о потребности в CARP. Отсутствует аппаратура отказоустойчивости, которая способна помогать, если кто-нибудь выдергивает силовой шнур, или если ваш системный администратор наберет reboot не в том окошке. CARP также упрощает процесс наложения заплаток и перезагрузки, делая этот период прозрачным для пользователей, и упрощает процесс тестирования программного обеспечения или аппаратную модернизацию - если они не работают, вы можете отступить на ваши резервы пока не устраните неисправности (fixed).

Есть, тем не менее, ситуации в которых CARP не поможет. CARP спроектирован таким образом, что требует, чтобы участники группы находились в той же физической подсети со статическим адресом IP, хотя с введением директивы carpdev, нет более потребности в IP адресах на физических интерфейсах. Аналогично, сервисы, требующие постоянное соединение с сервером (например, SSH или IRC), не будут точно переброшены в другую систему – все же в этом случае, CARP может попробовать снизить время простоя. CARP самостоятельно не синхронизирует данные между приложениями, это должно быть сделано через "альтернативные каналы", как например, pfsync(4) (для избыточной фильтрации), вручную дублируя данные между блоками с rsync, или все, что соответствует вашему приложению.

6.11.2 - Конфигурация

Элементы управления CARP расположены в двух местах: sysctl(8) и ifconfig(8). Давайте посмотрим на sysctls сначала.

Первый sysctl-овский параметр, net.inet.carp.allow, определяет может ли хост управлять CARP пакетами полностью. Ясно, необходимо использовать CARP. sysctl включает tuj по умолчанию.

Второй, net.inet.carp.arpbalance, используется для загрузки балансировки. Если эта характеристика включена, CARP получает хеши исходящих IP запроса. Хеш - затем используется, чтобы выбирать виртуальный хост из доступного пула для управления запросом. Этот параметр выключен по умолчанию.

Третий, net.inet.carp.log, ведет лог ошибок CARP. Выключено по умолчанию.

Четвертый, net.inet.carp.preempt включает естественный отбор из числа CARP хостов. Наиболее годный для работы (то есть, способный оповещать наиболее часто), станет мастером. Выключено по умолчанию, с той целью, чтобы система, которая не является мастером, не будет пытаться восстановить(установить) статус мастера.

Все эти переменные sysctl документированы в sysctl(3).

В завершении конфигурации CARP, мы зависим от ifconfig(8). Специфические CARP команды advbase и advskew имеют дело с интервалом между CARP оповещениями. Формула (в секундах) - это advskew поделеная на 256, затем добавленая к advbase. advbase, может быть использована, чтобы уменьшить сетевой трафик или позволить более длинное время ожидания прежде, чем резервный хост примет эстафету; advskew позволяет вам управлять тем какой хост станет мастером без большой задержки в случае сбоев (если это потребуется).

Затем, pass устанавливает пароль, и vhid устанавливает виртуальный идентификационный номер хоста CARP группы. Вам нужно назначить уникальное число для каждой CARP группы, даже если бы (для целей балансировки нагрузки), они разделяют (share) тот же IP адрес. CARP ограничен 255 группами.

Наконец, carpdev определяет какой физический интерфейс принадлежит к конкретной группе CARP. По умолчанию, какой-либо интерфейс, имеющий IP адрес в той же подсети, назначенной CARP интерфейсу, будет использован.

Давайте поместим все эти настройки вместе в основной конфигурации. Скажем, вы развертываете два идентично сконфигурированных Web сервера, *rachael* (192.168.0.5) и pris (192.168.0.6), чтобы заменить более старую систему, которая была по адресу 192.168.0.7. Команды:

```
rachael# ifconfig carp0 create
rachael# ifconfig carp0 vhid 1 pass tyrell carpdev fxp0 \
    192.168.0.7 255.255.255.0
```

создадут carp0 интерфейс и дадут ему vhid равным 1, пароль *tyrell*, и IP адрес 192.168.0.7 с маской 255.255.255.0. Назначает fxp0 в качестве элемента интерфейса. Чтобы сделать это постоянным после перезагрузки, вы можете создать /etc/hostname.carp0 файл, который выглядит похожим на этот:

```
inet 192.168.0.7 255.255.255.0 192.168.0.255 vhid 1 pass tyrell carpdev fxp0
```

Обратите внимание, что широковещательный адрес определен в этой строке, в дополнительнении к vhid и паролю. Отсутствие этого - общая причина ошибок, так что это нужно заполнить.

Сделайте тот же на pris. Какая угодно система, поднимающая интерфейс CARP впервые, будет мастером (в предположении, что параметр выключен; в противном случае это тоже истина, когда preempt включен).

Но скажем, вы не развертываете с нуля. *Rachael* был уже на месте по адресу 192.168.0.7. Как вам работать с этим? К счастью, CARP может иметься дело с этой ситуацией. Вы просто назначаете адрес на CARP интерфейс и оставляете физический интерфейс определенный ключевым словом `carpdev' без IP адреса. Тем не менее, он стремится быть чистым, чтобы иметь IP для каждой системы - он делает индивидуальный мониторинг и доступ значительно проще.

Давайте добавим другой уровень сложности; мы хотим , чтобы *rachael* оставался мастером, когда это возможно. Есть несколько причин, почему мы могли захотеть этого: аппаратные различия, простое предубеждение, "если эта система - не мастер, есть проблема," или зная по умолчанию мастера не прибегать к помощи сценариев (scripting), чтобы выполнять анализ и отсылать по электронной почте выходные данные ifconfig.

На *rachael*, мы будем использовать sysctl который мы создали выше, затем отредактируйте /etc/sysctl.conf, чтобы изменения остались постоянно.

```
rachael# sysctl net.inet.carp.preempt=1
```

Мы сделаем конфигурацию на *pris*, тоже:

```
pris# ifconfig carp0 advskew 100
```

Это немного задерживает *pris-вские* оповещения, с целью, чтобы rachael был мастером, когда жив.

Обратите внимание, что если вы используете PF на CARP компьютере, вы должны разрешить пропускать (pass) "proto carp" на всех связанных интерфейсах, строкой аналогичной этой:

```
pass on fxp0 proto carp keep state
```

6.11.3 - Балансировка нагрузки

Пролетело несколько месяцев. Наша компания из предшествующего примера выросла кстати, где единственный внутренний Web сервер просто едва управляется с загрузкой. Что делать? CARP спешит на помощь. Пора пробовать балансировку загрузки. Создайте новый CARP интерфейс и группу на *rachael*:

```
rachael# ifconfig carp1 create
rachael# ifconfig carp1 vhid 2 advskew 100 pass bryant carpdev fxp0 \
    192.168.0.7 255.255.255.0
```

На *pris*, мы создадим новую группу и интерфейс также, затем устанавливаем параметр "preempt" sysctl:

```
pris# ifconfig create carp1
pris# ifconfig carp1 vhid 2 pass bryant carpdev fxp0 192.168.0.7 255.255.255.0
pris# sysctl net.inet.carp.preempt=1'
```

Теперь у нас есть две CARP группы с тем же IP адресом. Каждая группа ассиметрична по отношению к другому хосту, с той целью, чтобы *rachael* остался мастером оригинальной группы, но *pris* захватит власть в новой.

Все что мы должны сделать теперь - это включить выравнивание нагрузки в sysctl, что мы обсуждали прежде, в обеих машинах:

```
# sysctl net.inet.carp.arpbalance=1
```

Пока эти примеры - для двух машинного кластера, те же принципы относятся к многим другим системам. Пожалуйста, обратите внимание, тем не менее, что не стоит ожидать, что вы достигнете идеального 50/50 распределения между двумя машинами - CARP использует хэш исходящих IP адресов, чтобы определять какая система обработает запрос, скорее, чем по нагрузке.

Еще информация о CARP

- carp(4)
- ifconfig(8)
- sysctl(8)
- sysctl(3)
- Firewall Failover with pfsync and CARP, автор Ryan McBride

6.12 - Использование OpenNTPD

Точное время важно для многих прикладных программ. Однако, наверное, некоторые замечали, что их пятидолларовые часы показывают время точнее, чем их компьютер за $2000. В дополнение к тому, чтобы просто знать точное время, также часто бывает важным синхронизировать его на нескольких компьютерах. Некоторое время на ntp.org применяли программу Network Time Protocol (RFC1305, RFC2030), доступную через порты, которую можно использовать для синхронизации времени на компьютерах через Интернет. Однако она использует нетривиальную настройку, сложный, для проведения аудита, код и высокие требования к памяти. Она играет важную роль для некоторых людей, но далека от повсеместного использования.

OpenNTPD является попыткой решить некоторые из этих проблем, сделать программу синхронизации времени на вашем компьютере простой для администрирования, безопасной и совместимой с NTP. ntpd(8) использует легкий в понимании файл конфигурации /etc/ntpd.conf.

Просто активировав ntpd (8) с помощью rc.conf.local, вы включите синхронизацию времени на вашем компьютере с серверами pool.ntp.org (несколько публичных time-серверов). Как только ваши часы точно установлены, ntpd будет содержать их в высокой степени точности, тем не менее, если ваши часы различаются на несколько минут, то *очень* рекомендуется, чтобы вы подвели их близко к точному времени, так как может занять дни или недели, чтобы подвести те же, но сильно отстающие часы для синхронизации. Вы можете делать это вручную с помощью команды date(1) или полуавтоматически, используя команду rdate(8), либо устанавливая ваши аппаратные часы вручную.

6.12.1 - "Но OpenNTPD не так точен как демон от ntp.org!"

Возможно, вы правы. Это не было целью дизайна OpenNTPD, он был предназначен для того, чтобы быть свободным, простым, надежным и безопасным. Если для вас точность в микросекундах действительно важнее, чем выгоды OpenNTPD, не стесняйтесь использовать демон ntpd от ntp.org, поскольку он останется доступным через порты и пакеты. Нет плана или желания иметь раздутый OpenNTPD каждой мыслимой характеристикой.

6.12.2 - "Кто-то говорит, что OpenNTPD `вреден'!"

Некоторые люди не поняли целей OpenNTPD -- простота, безопасность и легкость в поддержании ваших компьютерных часов в точности. Если важно поддержание точности времени, некоторые пользователи сообщали о лучших результатах OpenNTPD по сравнению с ntpd от ntp.org. Если важна безопасность, то код OpenNTPD более удобочитаемый (и, таким образом, проще проводить его аудит) и был написан, используя стандартные функциональные вызовы OpenBSD, типа strlcpy, а не с помощью портируемых функций вроде strcpy. OpenNTPD написан, чтобы быть безопасным с самого начала, а не чтобы быть "сделанным безопасным позже". Если все больше людей используют синхронизацию времени это высоко ценимо, OpenNTPD делает этот процесс простым для его использования большим количеством людей. Если это `вредно', нам же хуже.

Бывают случаи, в которых целесообразнее применение демона ntpd от ntp.org. Тем менее, считается, что для остальных 95% пользователей OpenNTPD является более чем достаточным.

Более полный ответ на этот вопрос был дан одним из разработчиков OpenNTPD. Он может быть прочитан здесь.

6.12.3 - Почему другие мои машины не могут синхронизироваться по OpenNTPD?

Демон ntpd(8) не прослушает все [any] адреса по умолчанию. Так для того, чтобы использовать его как сервер, вы должны раскоментировать строку "#listen on *" в /etc/ntpd.conf и перезапустить демон ntpd(8). Конечно, если вы хотите, прослушивать на конкретном IP адресе, а не на всех доступных адресах и интерфейсах, замените "*" желаемым адресом.

Когда у вас ntpd(8) начал слушать, может случиться, что другие машины все еще не могут синхронизироваться с ним! Недавно запущенный ntpd(8) демон (например, если вы только что перезапустили его после модификации ntpd.conf) отказывается снабжать информацией о времени других клиентов, пока они сначала не отрегулирует свои собственные часы на разумный уровень точности. В то время как ntpd(8) рассматривает свою информацию о времени точной, он заявляет это сообщением "clock now synced" ["часы теперь синхронизированы"] в /var/log/daemon. Даже если бы системные часы безупречно точны в начале, то может потребоваться до 10 минут, чтобы быть синхронизированными, и часы или дни, если часы точно не установлены в начале.

6.13 - Какие есть возможности по использованию беспроводных сетей?

Поддерживаемые OpenBSD чипсеты беспроводных устройств:

- awi(4) AMD 802.11 PCnet Mobile
- an(4) Aironet Communications 4500/4800
- wi(4) Prism2/2.5/3
- atw(4) ADMtek ADM8211
- ath(4) драйвер для Atheros IEEE 802.11a/b/g (AP)
- iwi(4) Intel PRO/Wireless 2200BG/2225BG/2915ABG IEEE 802.11a/b/g (NFF)
- ipw(4) Intel PRO/Wireless 2100 IEEE 802.11b (NFF)
- atu(4) Atmel AT76C50x USB IEEE 802.11b.
- ral(4) и ural(4) [USB] Ralink Technology RT25x0 IEEE 802.11a/b/g (AP)
- rtw(4) Realtek 8180 IEEE 802.11b.
- acx(4) TI ACX100/ACX111. (NFF) (AP)
- pgt(4) Conexant/Intersil Prism GT Full-MAC. (NFF) (AP)
- rum(4) Ralink Technology RT2501USB (AP)
- wpi(4) Intel PRO/Wireless 3945ABG (NFF)
- uath(4) Atheros AR5005UG/AR5005UX USB2.0 (work-in-progress). (NFF)

(AP) означает что карта может быть использована как точка доступа.

(NFF) означает, что чип требует закрытого ПО (firmware) для своей работы, которое не может быть включено в OpenBSD.

Устройства, основанные на этих чипах могут быть использованы наравне с другими сетевыми устройствами для подключения машины с OpenBSD к уже существующей беспроводной сети (см. соответствующие руководства man для получения исчерпывающей информации по этому поводу) используя стандартный ifconfig(8). Некоторые из устройств, работая в режиме "точки доступа", могут быть использованы непосредственно для создания полноценной беспроводной ТД, интегрированной в фаерволл (в данном случае - в машину на базе OpenBSD [прим. пер.]).

Стоит отметить, что при использовании беспроводных сетевых карт Intel необходимо установить соответствующее аппаратно-программное обеспечение (firmware), которое не является свободно распространяемым и, таким образом, не может быть включено в состав OpenBSD. Вы можете связаться с Intel и высказать им всё, что вы думаете по этому поводу или сообщить какое устройство вы приобрели взамен.

Еще один способ включения OpenBSD в беспроводную сеть состоит в совместном использовании обычной сетевой карты и внешней точки доступа. Очевидное преимущество выражается в том, что подобная схема позволяет удобно расположить антенну в месте, где сигнал будет наиболее качественным, что зачастую отнюдь не является местом непосредственно рядом с машиной.

6.14 - Как мне настроить equal-cost multipath routing?

Equal-cost multipath routing означает, что имеется несколько маршрутов в таблице маршрутизации к данной сети, например маршрут по умолчанию default route, 0.0.0.0/0. Когда ядро определяет, каким путем отправить пакет для данной сети, оно может выбрать любой из equal-cost маршрутов. В большинстве случаев такая маршрутизация используется для создания

резервированного соединения с сетью логически более "корневого" уровня, например, подключения к сети интернет.

Команда route(8) используется для добавления/удаления/изменения маршрутов в таблице маршрутизации routing table. Аргумент -mpath используется при добавлении multipath routes.

```
# route add -mpath default 10.130.128.1
# route add -mpath default 10.132.0.1
```

Проверим маршрутизацию:

```
# netstat -rnf inet | grep default
default     10.130.128.1      UGS       2      134      -     fxp1
default     10.132.0.1        UGS       0      172      -     fxp2
```

В этом примере мы видим, что маршрут по умолчанию достижим используя 10.130.128.1 через интерфейс fxp1, а другой вариант - 10.132.0.1 и fxp2.

Используя файл mygate(5) мы не сможем настроить по умолчанию multipath default routes, нам потребуется еще отредактировать файлы hostname.if(5) для интерфейсов fxp1 и fxp2. Файл /etc/mygate нам надо будет удалить.

```
/etc/hostname.fxp1
    !route add -mpath default 10.130.128.1
/etc/hostname.fxp2
    !route add -mpath default 10.132.0.1
```

Не забудьте для использования multipath routes включить нужные опции sysctl(3).

```
# sysctl net.inet.ip.multipath=1
# sysctl net.inet6.ip6.multipath=1
```

Не забудьте отредактировать sysctl.conf(5) чтобы запомнить изменения.

А теперь попробуем выполнить трассировку разными маршрутами. Ядро будет распределять трафик в режиме балансировки (load balance) multipath route.

```
# traceroute -n 154.11.0.4
traceroute to 154.11.0.4 (154.11.0.4), 64 hops max, 60 byte packets
 1  10.130.128.1  19.337 ms  18.194 ms  18.849 ms
 2  154.11.95.170  17.642 ms  18.176 ms  17.731 ms
 3  154.11.5.33  110.486 ms  19.478 ms  100.949 ms
 4  154.11.0.4  32.772 ms  33.534 ms  32.835 ms

# traceroute -n 154.11.0.5
traceroute to 154.11.0.5 (154.11.0.5), 64 hops max, 60 byte packets
 1  10.132.0.1  14.175 ms  14.503 ms  14.58 ms
 2  154.11.95.38  13.664 ms  13.962 ms  13.445 ms
 3  208.38.16.151  13.964 ms  13.347 ms  13.788 ms
 4  154.11.0.5  30.177 ms  30.95 ms  30.593 ms
```

Для дополнительной информации о способе выбора маршрута читайте RFC2992, "Analysis of an Equal-Cost Multi-Path Algorithm".

Стоит заметить, если интерфейс, используемый при multipath route потеряет соединение (например, нет carrier), ядро тем не менее все равно будет пытаться отправить через него пакеты. Конечно, этот трафик никуда не попадет. Очень рекомендуется использовать ifstated(8) для проверки доступности интерфейса и коррекции таблицы маршрутизации.

Перевод соответствует $OpenBSD: faq6.html,v 1.268 2008/05/01 20:41:24 steven Exp $