Сети. Конспект лекций.

Первоисточник: Сергей Браун
Автор: Наталья Бабаева
Бета-ридеры: Сергей Браун, Юрий Фролов

Модель OSI:
7 — прикладной уровень (Application layer)
6 — уровень представления (Presentation)
5 — сеансовый уровень (Session)
4 — транспортный уровень (Transport)
3 — сетевой уровень (Network)
2 — канальный уровень (Datalink)
1 — физический уровень (Physical)

Модель OSI — это абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов. То есть, если вы хотите сделать свой стек протоколов для взаимодействия, построить свою сеть, то в идеале, оно должно быть похоже на эту модель.

На физическом уровне (1) — разъемы, провода, модуляции сигналов.
На канальном уровне (2) — обеспечение передачи точка-точка (в одной сети). На уровне работают хабы, свитчи. Адресация MAC-адресами. MAC (Media Access Control) и LLC (Logical Link Control) – два подуровня.
На сетевом уровне (3) — пакет путешествует между сетями (передача данных между локальными сетями). ARP-запрос — запрос по протоколу ARP (Address Resolution Protocol) чтобы узнать MAC по IP. На сетевом уровне для адресации используются IP-адреса.
Транспортный уровень (4) отвечает за гарантированную доставку.
Сеансовый уровень (5) отвечает за установку соединения. Могут устанавливаться контрольные точки.
На уровне представления (6) можно переставить порядок байт (**в сети прямой порядок байт!**). Тут же происходит шифрование.

IP-адрес
IP-адрес — это 32 бита (4 байта), 4 октета. А еще это адрес компьютера в сети.
Есть консорциум IANA, который раздает IP-адреса по 5 организациям (ARIN, RIPE, APNIC, AfriNIC, LACNIC). Им выдаются сети класса A. Далее – распределение по заявкам от организаций со статусом LIR (Local Internet Resource) подсетями /22 или крупнее, а в случае выделения провайдеро-независимого блока - /24 (класс C) и крупнее.
Маска подсети — указывает, какая часть IP-адреса приходится на адрес сети, а какая — на адрес хоста в ней. В принципе, это последовательность скольких-то единичек, потом нули, всего 32 бита, там где единицы – бит входящий в адрес сети. Может записываться как IP (255.255.255.0 — это 24 единички и 8 нулей, первые 24 символа будут адресом сети, оставшиеся 8 — адресом хоста), может указываться как число от 0 до 32 (192.168.1.100/24 — здесь «/24» — это маска, то есть 24 единички в начале, остальные — нули. Сеть здесь 192.168.1.0).
Широковещательный запрос — отправка пакета всем устройствам в сети. Для этого есть специальный адрес: в IP-адрес после адреса сети вместо адреса хоста ставятся все единички.
Соответственно, максимально возможное количество компов в сети вычисляется по формуле 2^(32-маска)-2. Так как, когда вместо адреса хоста все нули — это адрес сети, а когда все единички — это широковещательный запрос, соответственно, теряем два адреса.

Маска **/32 —** адрес ровно один (одного компа)
Маска **/31** — используется для маршрутизации для соединения точка-точка, или если два адреса на один комп — для экономии адресов. В настройки адреса на устройстве не ставится.
0.0.0.0/0 — весь интернет.
255.255.255.255 — широковещательный запрос всем (в локальной сети).
Адреса, которые запрещены в инете, эти адреса можем использовать для себя**:**
Для собственных локальных сетей:
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
и
127.0.0.0/8 – loopback – адреса которые доступны только внутри одного хоста
Классификация сетей
A: 0-127/8
B: 128-191/16
C: 192-223/24
D: 224-239
В сети класса А 2^24 компов в одной сетке.

Протоколы. Удаленный доступ.
В винде протокол RDP.
telnet — команда и протокол. Специфицирует передачу символов ANSI и всё. Не рекомендуется использовать не через свою сеть, так как нет шифрования. Но он простой и его знают все устройства и не тормозят.
Команда:
```
$ telnet 192.168.1.55
```
После IP можно указать порт.
Некоторые порты:
80 — http
25 — smtp
143 — imap
22 — ssh
Можно посмотреть, открыт ли порт. Если пустая строка и можно нажать Enter, значит открыт. Не рекомендуется настраивать telnet на компе.

SSH — Secure Socket Shell. Протокол использует шифрование SSL (Secure Socket Layer).
Перевесить ssh на другой порт не помешает, ибо про 22 все в курсе (и злобные хакеры найдут открытый 22 порт и сделают вам DdoS атаку и подвесят ваш сервер нафиг). Настройки в /etc/ssh/sshd_config (server) или в /etc/ssh/ssh_config (client). Там есть строчка port, вот её и надо менять.
Команда для установки:
```
# apt-get install openssh-server
```
Зайти куда-то по ssh с текущим логином:
```
$ ssh 1.2.5.8
```
или:
```
$ ssh myhost.foo.net
```
с другим логином (user):
```
$ ssh user@1.2.5.8
```
с указанием порта (на котором висит ssh там, куда заходите), у нас 1234:
```
$ ssh 1.2.5.8 -l user -p 1234
```

Как выполнится команда — попадем в консольку другого компа. Как закроем консольку или скажем exit, соединение умрёт.

Чтоб скопировать файлики с сервера на сервер есть команда **scp**. Нельзя с её помощью копировать с удалённого компа на удалённый. Только от себя кому-то или от кого-то себе. Работает через ssh (и ssh'ный порт).
Флаг -r позволяет копировать папки (рекурсивно). Порт указывается флагом -P
Скопировать папку ipc с удаленного компа с ip 192.168.1.37, где пользователь mdr и ssh на 221 порту себе в домашнюю папку /home/nata:

```
$ scp -P 221 -r mdr@192.168.1.37:/home/mdr/ipc /home/nata
```

При копировании нужно следить чтобы были права на чтение из папки источника и на запись в соответствующую целевую директорию у тех пользователей под которыми к ним обращаются. Это – одна из самых частых ошибок при которых не происходит копирования.

Настройка сетевых интерфейсов.

Интерфейсом с точки зрения ОС является устройство, через которое система получает и передает IP-пакеты. Роль интерфейса локальной сети может выполнять одно (или несколько) из следующих устройств: Ethernet-карта, ISDN-адаптер или модем, подключенный к последовательному порту. Каждое устройство (не весь компьютер!) имеет свой IP-адрес (а может, и не один).

Команда **ifconfig** показывает поднятые интерфейсы:

```
$ ifconfig
```

Результат выглядит примерно так:

```
nata@nata-U31Jg:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr bc:ae:c5:5f:78:45
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:44 Base address:0x2000

lo        Link encap:Локальная петля (Loopback)
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6821 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6821 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1471191 (1.4 MB)  TX bytes:1471191 (1.4 MB)

wlan0     Link encap:Ethernet  HWaddr 00:25:d3:95:b9:c2
          inet addr:192.168.1.38 Bcast:192.168.1.255
          Mask:255.255.255.0
          inet6 addr: fe80::225:d3ff:fe95:b9c2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13247 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11801 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6664433 (6.6 MB)  TX bytes:2250776 (2.2 MB)
          Interrupt:17 Память:ffffc900117c8000-ffffc900117c8100
```

1 — название интерфейса.
2 — поднят он или опущен (включен или выключен) и далее – другие флаги состояния.
3 — IP-адрес, широковещательный адрес, маска подсети.
4 — MAC-адрес сетевухи, на которой поднят интерфейс.

Команда

```
# ifconfig eth0 172.16.1.1 netmask 255.255.255.0
```

устанавливает IP 172.16.1.1 с маской 255.255.255.0 на интерфейс eth0.

Показать все сконфигурированные и несконфигурированные интерфейсы:

```
$ ifconfig -a
```

Команда **route** может показывать и задавать маршрут.

```
# route add -net  129.11.0.0/16 gw 1.1.1.2
```

сеть 129.11.0.0/16 доступна через шлюз 1.1.1.2

Команда ip.
Часть пакета iproute2. Может, похоже, всё. Управляет разными шнягами в ядре. (Если честно – то почти всем IP-стеком ядра linux)
```
$ ip help
```
(вызов справки, на help можно заменить любой параметр, тогда справка, по возможности, покажется только по этому параметру)
```
$ ip link
```
Это управление сетевыми картами. Расскажет об интерфейсах.
В том, что выводит:
флаг LOWER_UP — если это виртуальный интерфейс, то показывает, что основной интерфейс включен тоже.
mtu (max transfer unit) — максимальный кусок, который гоняем.
qdisc — очередь на отправку.
qlen — сколько пакетов можно поставить в очередь.
Сокращение команды:
```
$ ip l
```
Поднять интерфейс:
```
# ip link set eth0 up
```
Опустить интерфейс:
```
# ip link set eth0 down
# ip l help
```
promisc — переход в режим “promisc” - получать все пакеты, приходящие на сетевую карту, а не только предназначенные этому хосту
address — указать MAC.
Если перенастраиваем mtu (Maximum Transfer Unit), то на всех устройствах в сети – тогда можно передавать пакеты изменённого размера (больше или меньше).

Управление всем что связано с IP-адресами:
```
# ip addr
```

Добавить IP:
```
# ip addr add 192.168.1.10/24 dev eth0
```
То же самое:
```
# ip a a 192.168.1.10/24 dev eth0
```
Создать соединение точка-точка:
```
# ip a a 1.1.1.1 peer 2.2.2.2 dev eth0
```
Удалить IP:
```
# ip a d 192.168.1.10/24 dev eth0
```
Команда
```
# ip addr flush
```
снесёт все IP-адреса. Не надо так делать (если не знаете что именно это и надо сделать!!!).

Показать все маршруты (в табличке main):
```
$ ip route
$ ip r
```

Добавить маршрут:
```
# ip r a default via 192.168.1.100
```
Лучше не добавлять маршруты, а заменять. Если его нет, он добавится.

Все пакетики, пришедшие из сетки 1.1.1.0 отправлять на 2.2.2.2, подменяя адрес источника на 3.3.3.3:

```
# ip r r 1.1.1.0/24 via 2.2.2.2 dev eth2 src 3.3.3.3
```

Посмотреть маршрут до отпределенного IP:

```
$ ip route get 8.8.8.8
```

Посмотреть соседей:

```
$ ip neigh
```

Таблиц маршрутизации 256 штук минимум. В них хранятся маршруты.
В /etc/iproute2/rt_tables хранится список таблиц. Посмотреть:

```
$ cat /etc/iproute2/rt_tables
```

Посмотреть содержимое таблицы local

```
$ ip r l table local
```

Посмотреть содержимое таблицы main (оно выводится по умолчанию, если не указана другая таблица)

```
$ ip r l table main
```

Изначально есть 3 таблицы: local (255), main (254), default (253). В таблице local лежат все записи и добавляются при добавлении IP-адресов. В этой же таблице все записи про loopback. Руками лучше не трогать!
В таблице main все «обычные» маршруты.

Для работы с таблицами и определения порядка их просмотра используется

```
# ip rule
```

Пришел пакет, отфильтровался фильтрами и смотрит по таблице маршрутизации «а куда дальше?». Можно перенаправлять поиск маршрута в зависимости от пакета в разные таблицы.
Типа:

```
from x1 unlim
from x2 unlim
from y lim
from all default
```

Где x1,x2,y — пулы IP'шников(сетки).
Посмотреть правила просмотра таблиц маршрутизации:

```
# ip rule s
```

Добавить правило просмотра и маршрут в свою таблицу:

```
# ip rule add from ivan_ip table mytable
# ip route add default via ip_slow_inet dev eth2 table mytable
```

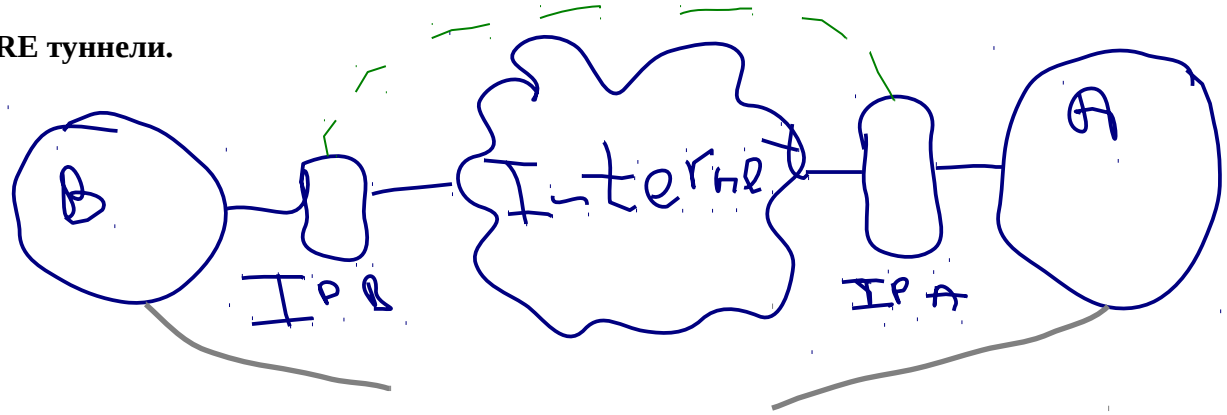
Если направление зависит от источника (from), то это называется source based routing.

Чтобы пакеты ходили через твой комп:

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

(Это называется «включить форвардинг пакетов», «включить маршрутизацию» и т.п.)

GRE туннели.



Тут серые IP

Хотим ходить от B к A напрямую. А там «серые» IP'шники. Делаем туннель (зелененький) – «виртуальный провод, протянутый через интернеты». Прямоугольнички — линуксовые (или cisco или ещё какие-то) маршрутизаторы. GRE — это такой протокол (см википедию ;)). По сути своей «приклеивает» ещё один IP заголовок к IP-пакету . Получается пакет в пакете. IPA, IPB — IP-адреса.

Создать туннель на A:

```
# ip tunnel add tun1 local IPA remote IPB mode gre
```

Создать туннель на B:

```
# ip tunnel add tun1 local IPB remote IPA mode gre
```

Поднять туннель на A:

```
# ip link set tun1 up
```

Поднять туннель на B:

```
# ip link set tun1 up
```

Добавить IP на A:

```
# ip a a 10.10.1.1/24 dev tun1
```

Добавить IP на B:

```
# ip a a 10.10.1.2/24 dev tun1
```

Потом стоит попинговать эти самые 10.10.x.y и на другой стороне. Посмотреть, что происходит можно с помощью **команды tcpdump.** То есть, вы кого-то пингуете и тот, кого вы пингуете говорит у себя tcpdump и втыкает в то, что там показывается.

```
# tcpdump -i tun1 -vv
# tcpdump -nq
```

(крутой вид, -n это с IP'шниками) (да и если у вас нет инета и рабочего DNS – хотя бы работает в отличие от первого варианта)

В туннелях нет ARP, ибо не нужен :-).

Если с туннелями что-то не работает (такое, что непонятно, почему), можно попробовать вылечить с помощью:

```
# modeprobe ip_gre
```

А ещё лучше – выполнять загрузку модуля до настройки туннелей...

O_O!!!!
Вернуть инет в лаборатории если вы сломали маршрутизацию:
```
# ip r d default
# ip r a default gw 192.168.1.100 dev eth0
```
(адрес роутера в лабе 192.168.1.100)

Нечто про группы адресов вообще и multicast в частности.
unicast — 1 к 1
broadcast — 1 всем в локальной сети
multicast — 1 отправляет кому-то (протоколы IGMP, PIM и пр.)
geocast

multicast-адреса: 224.0.00-239.255.255.255
Используются для IPTV, вещания видео, аудио.
Была передача точка-точка. Поток видео льется чуваку на комп. А если 5 чуваков смотрят видео в одной сети? Оно будет литься 5 раз. Скушается весь канал до них. Плохо же. Чтоб раздавать такое и придумана multicast-передача и маршрутизация multicast.
Multicast-трафик «раздаётся» по группам. Попадание в группу – подпиской.
224.0.0.1 — группа, которая поддерживает multucast (все)
224.0.0.2 — все роутеры, которые поддерживают multicast

multicast внутри одной сети:
Понадобится:
```
# apt-get install smcroute
```
Выбираем любой IP из подсети (пр. 224.4.4.4) и говорю, что я по конкретному интерфейсу вхожу в эту группу.
Добавить маршрут сюда:
224.0.0.0/4 dev eth0
```
$ ping 224.0.0.1
```
(откликнутся все, у кого multicast поднят)
```
# netstat -g
# apt-get install wdb
```
(Рисовалка общая, что ли...)

```
# smcroute -j eth1 224.5.5.5
```
где 224.5.5.5 - № группы
```
# wdb 224.5.5.5
```
(для группы)

еще про идеологию multicast:
Есть большие цепочки свичей и роутеров. Есть где-то в конце цепочки два чувака, которые смотрят телик. Хотим пригнать канал на этот свич. Если смотрят двое, то распараллелить канал уже на самом свиче.
На нашей стороне нужно выполнить операцию подписки на группу. На свиче нужно включить поддержку IGMP.
Igmp-snooping. Свич рассылает компам сообщения «я умею IGMP версии такой-то». Если пришел запрос, смотрит, есть ли кто-то подписанный на эту группу. Если есть — добавляет, если нет — ищет, кто это может раздать, т. е., выступает уже клиентом.
Если мы внутри локальной сети, то просто соответствие multicast портов и IP.

DHCP — Dynamic Host Configuration Protocol.
Работает в стеке протоколов TCP/IP.
Сисадмин настраивает DHCP-сервер, там задает пул адресов. Для каждой такой сети — маска, шлюз, DNS-адреса, политики и т. д.
Порты:
67 — сервер
68 — клиент
Работает, в основном, по UDP.
4 фазы:

1) discover (dhcp-discover) — клиент отправляет полностью широковещательный запрос со всеми «F»
2) сервер, увидев запрос, разгребает его, предлагает IP'шник. Lease-offer предложение, в нём указывается предлагаемый клиенту IP.
3) клиент может получить несколько разных ответов. Клиент выбирает, обычно, первый и отправляет запрос request «да-да, я хочу этот IP».
4) Сервер отправляет dhcp-acknowledge — подтверждение, что IP закреплён за клиентом.

Есть ещё dhcp_release — клиент говорит, что он освободил IP.

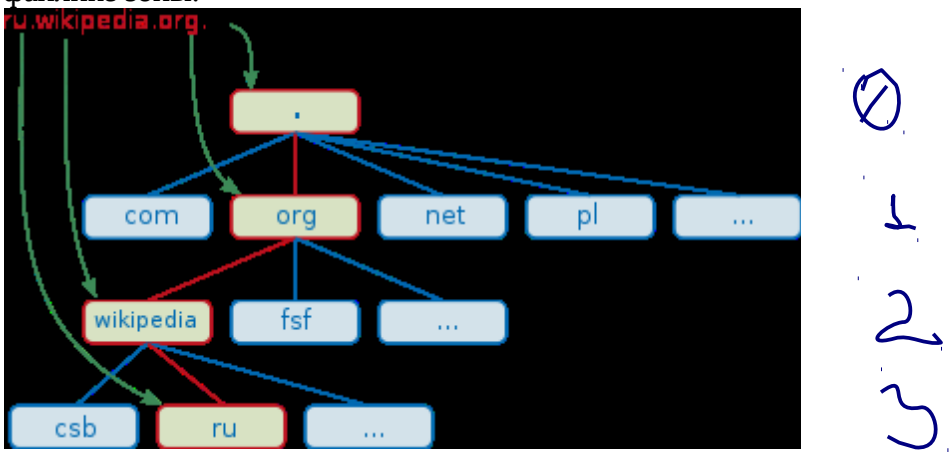
Получить IP у клиента на интерфейсе eth0:
```
# dhcpcd eth0
```
или
```
# dhclient eth0
```
В файле /etc/network/interfaces хранятся настройки интерфейсов. Если IP-адрес получается динамически (по DHCP), то там указано:
```
auto eth0
iface eth0 inet dhcp
```

IP по дефолту выдается на кокое-то время. По окончании времени должен быть запрос (DHCP_REQUEST) на выдачу (возможно, этого же IP). (это время называется default_lease_time)
Есть max время, на которое выдается IP. Через него нужно все равно вернуть IP. (это max_lease_time)
Надо учитывать, где настраиваем:
Гостевая точка — 20 и 30 минут
В лабе — сутки
Дома — тоже можно сутки, можно и неделю
Нельзя ставить default_lease_time и max_lease_time в одно и то же значение. max_lease_time должно быть больше хотя бы на несколько минут.

Чтоб было, что настраивать:
```
# apt-get install dhcp3-server
# apt-get install dhcp3-client
```
Настройки DHCP-сервера в файле /etc/dhcp3/dhcpd.conf
Там же можно посмотреть примеры настройки подсетей(subnet) и всего такого.
В настройках имеет влияние наиболее специфичная опция.
Не рекомендуется использовать:
192.168.0.0/24
192.168.1.0/24
10.10.10.0
10.0.0.0
Ибо они стоят у чертовой уймы устройств по дефолту, и у кучи народа без фантазии.
Для загрузки по сети (используя PXE):
next-server IP
filename path;
IP — это IP tftp-сервера, path — путь на самом сервере к файлу.
Перезапустить DHCP (чтоб изменения в конфиге заработали):
```
# /etc/init.d/isc-dhcp-server restart
```
или
```
# service dhcp restart
```

DNS — Domain Name Service
В мире чуть меньше 4 млрд. IP-адресов. ~15 байт на имя + 4 байта на IP ≈ 20 байт
4 млрд. * 20 = 80 млрд. байт ≈ 4 Гб. Это много, поиск по такой базе затруднён. Эти 4 гига должны быть на каждом компе и маршрутизаторе и везде. И обновлять изменения. Невозможно. Так и придумали DNS.
Основная задача — преобразование доменных имен в IP-адреса и обратно.
Есть домен точка — домен верхнего (нулевого) уровня, там поддомены, они первого уровня. Набор серверов (сейчас 13) отвечают за домен точка. Это root-серверы. Они по UDP (53 порт)

отвечают, где есть серверы, которые ответственны за домен первого уровня. Есть файлик hint — подсказка — там эти root-серверы прописаны.
Далее - в каждой зоне свои серверы (min 2 — один master, остальные slave).
Важно правильно писать номер изменения зоны (год-месяц-день-номер изменения в этот день). Он сравнивается между мастером и слейвом. Если у мастера номер больше, то слейв перезатягивает себе файл зоны.
Есть институт регистраторов, которые могут регистрировать доменные имена второго уровня. Всего может быть сколько угодно уровней.
Сервера есть рекурсивные и нерекурсивные, кэширующие и некэширующие.
Рекурсивные — сами идут по цепочке и возвращают тебе IP искомого сервера.
Нерекурсивные — возвращают адреса серверов, отвечающих за домен следующего уровня.
Кэширующие — запоминают запросы на какое-то время. Есть ttl в секундах.
Некэширующие — не запоминают.
Те, кто отвечают за домен точка — нерекурсивный и некэширующий.
В инете сервера рекурсивные (только у дурачков, если не считать локальных серверов интернет - провайдеров)
У любой записи есть TTL(Time To Live) — через столько будут переспрашивать запись.
Прописывается в файлике зоны.



DNS-ответы — записи с различными типами:
Наиболее важные типы DNS-записей:

- **Запись A** (*address record*) или запись адреса — связывает имя хоста с адресом IP. Например, запрос A-записи на имя `referrals.icann.org` вернет его IP адрес — `192.0.34.164`
- **Запись AAAA** (*IPv6 address record*) связывает имя хоста с адресом протокола IPv6. Например, запрос AAAA-записи на имя `K.ROOT-SERVERS.NET` вернет его IPv6 адрес — `2001:7fd::1`
- **Запись CNAME** (*canonical name record*) или каноническая запись имени (псевдоним) используется для перенаправления на другое имя
- **Запись MX** (*mail exchange*) или почтовый обменник указывает сервер(ы) обмена почтой для данного домена.
- **Запись NS** (*name server*) указывает на DNS-сервер для данного домена.
- **Запись PTR** (*pointer*) или запись указателя связывает IP хоста с его каноническим именем. (По IP получить имя).
- **Запись SOA** (*Start of Authority*) или начальная запись зоны указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, *тайминги* (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов.

- **SRV**-запись (*server selection*) указывает на серверы для сервисов, используется, в частности, для Jabber и Active Directory.

На запрос отвечают полным списком серверов. И при каждом запросе верхний уходит вниз, чтобы размазать нагрузку. Записями можно балансировать нагрузку на сервера.

Обратная зона DNS.
Основное применение обратной зоны — почтовые проверки.
in_addr.arpa В этой зоне все IP-адреса. Зона пример: 1.168.192.in_addr.arpa).

```
# apt-get install bind9
```

Настройки в /etc/bind/named.conf
Перезапуск:

```
# service bind9 restart
```

Squid

```
# apt-get install squid
```

Основан на access-list — списках доступа.

Name — имя списка

```
acl name src ip
http_port 3128
```

По браузеру. user_agent узнается путема набора whatismyuseragent в адресной строке браузера

```
acl name browser user_agent_name
```

Авторизация squid

```
# apt-get install apache2-utils
# htpasswd -c filename user
```

в squid.conf поставить
auth_param basic

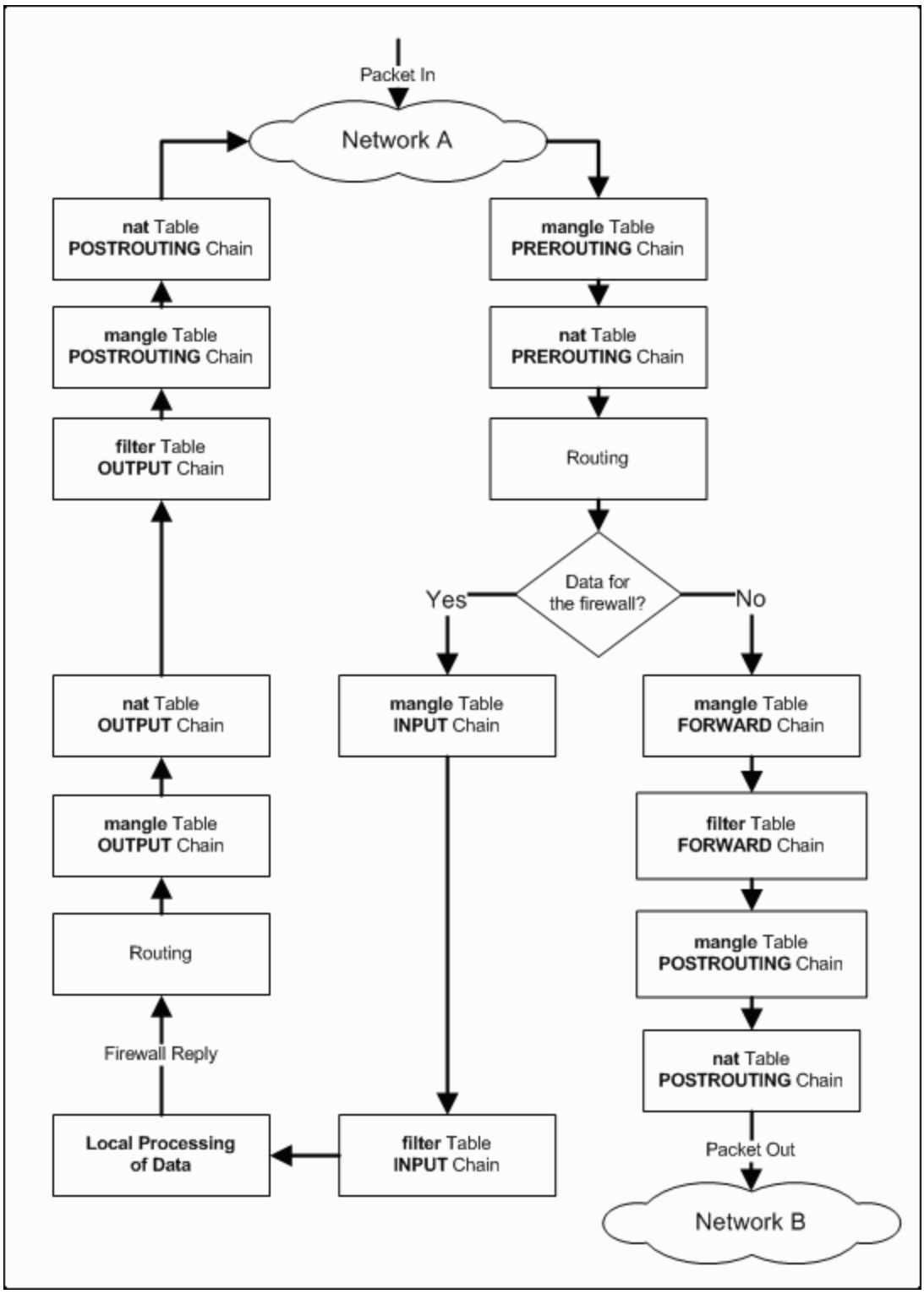
Перезагрузка:

```
# service squid restart
```

Если не работает, то:

```
# killall squid
# /etc/init.d/squid start
```

Фильтрация пакетов в Unix.
За фильтрацию пакетов отвечают iptables.
Таблицы: mangle, nat, filter.
Цепочки: FORWARD, PREROUTING, INPUT, POSTROUTING, OUTPUT.
Не все цепочки есть в каждой таблице.
На одной машине:



mangle — только в ней можно изменять заголовки пакета
nat — только в ней можно изменять адреса назначения и адрес отправителя

Критерии:
Общие:
`-i — интерфейс, на который приходит пакет`
`-o — интерфейс, с которого уходит пакет`
`-p — протокол`

```
-s, --src — адрес источника/отправителя
-d, --dst — адрес назначения/получателя
```
TCP:
```
--sport — порт источника
--dport — порт назначения
```
Ещё:
```
--tcp-flags
--icmp-type
-m mac — фильтрация по MAC-адресу
```

```
# iptables [-t table] -A chain rule
```
Вместо -A (add) можно -D (delete)
Есть стандартные действия: ACCEPT, DROP, REJECT.
Если пакет ничему не соответствует — политика по умолчанию (ACCEPT, DROP, REJECT — для стандартных цепочек, а для своих цепочек — возврат в то место, где ушли в эту цепочку). Первые правила нужно делать такие, чтоб они обрабатывали большую часть траффика.
Есть 2 счетчика: сколько пакетов соответствовало правилу и сколько байт прошло. Короче, можно считать траффик.

Примерны дополнительных полезных условий фильтрации:
conlimit — ограничить количество соединений с одного адреса на сервер
iprange
limit — ограничение скорости (в пакетах в час)
owner
recent — полезно от DdoS атак
u32 – страшно, но можно фильтровать по любым битам пакета IP!

Цели (полезное) или «что делать с пакетом»:
DNAT — куда перенаправлять
LOG
MASQUERADE — подставить свой IP
SNAT — замена адреса источника
TEE — чтоб спараллелить траффик
TTL — поменять TTL
и т. д.

Примерчики:
```
# iptables -t nat -A POSTROUTING -j SNAT --to-source ip_for_inet
# iptables -A INPUT -j ACCEPT
```
(в таблицу filter (она по дефолту, если не указана другая), цепочку INPUT — все пакеты принимать.
Есть 2 способа, чтобы увидеть, что происходит:

1) conntrack — трассировщик — входит в iptables. В файл /proc/net/ip_conntrack записывается содержимое пакетов.
2) tcpdump — видим, что пакеты идут не туда.

Зайти на D-Link

IP роутера 10.90.90.90 (десять три раза девяносто)

1) Чтобы зайти на роутер вам понадобится навесить себе на интерфейс IP из той же подсети. То есть, сказать у себя на компе

```
# ip a a 10.90.90.x/24 dev eth0
```

Где x — число от 1 до 254 и не равное 90.

2) Сказать

```
$ telnet 10.90.90.90
```

Вас спросят логин-пароль, нажмите 2 раза Enter.

3) Если зайти на роутер не удается — попробуйте его попинговать. Если не пингуется — или роутер лежит, или вы налажали.
4) Если коннект разрывается и вас выкидывает каждую минуту и чаще, то, возможно, ваш IP (тот самый X) уже кто-то занял. Спросите громко «никто не занял такой-то IP?». Потом попробуйте переназначить. То есть, удалите старый (вторую a в команде заменить на d). И повторить назначение IP, взяв другой X.

Посмотреть соответствие MAC и порта на D-Link:

> show fdb

Это вам понадобится для того, чтобы добавить свой порт в какие-нибудь VLAN'ы.

VLAN — Virtual Local Area Network

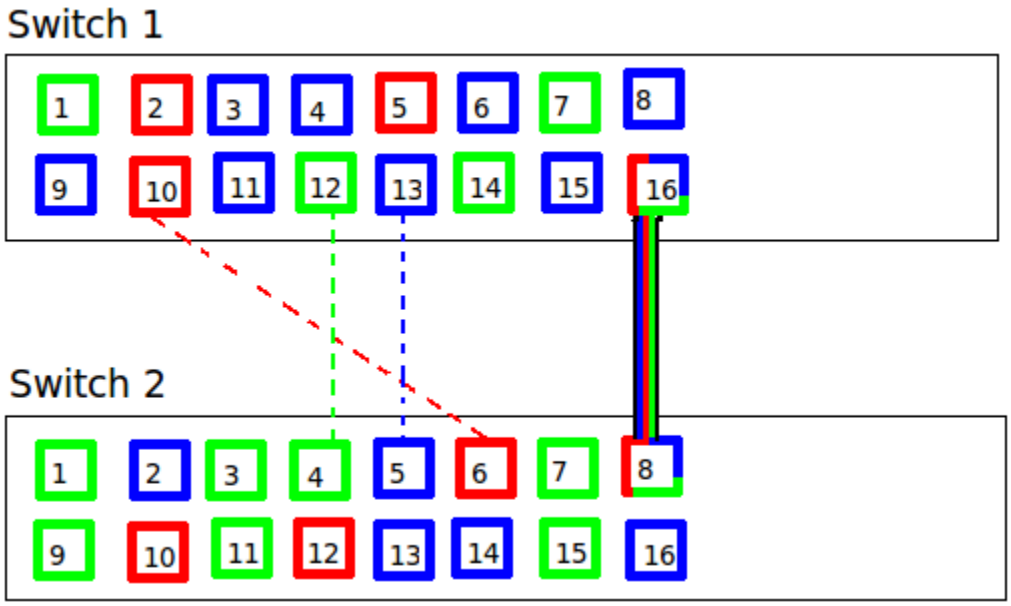
Общая идеология:

Виртуальные локальные сети. Эта штука работает на канальном уровне. Она позволяет делать локальную сеть, физически состоящую из нескольких железок (свичей), но которая видится как сеть, в которой одна многопортовая железка. И наоборот, на одной железке можно уместить несколько сетей так, что они не будут видеть друг-друга.

То есть, мы можем создавать логическую топологию сети не зависимо от физической топологии.

Создаётся VLAN, туда добавляются порты, которые в этот VLAN входят. Порты могут быть тегированными (tagged) и нетегированными (untagged), то есть, с меткой и без метки. Метка — это число, вроде как, идентификатор принадлежности к VLAN'у. Эта самая метка указывается при создании VLAN'а. Тегированные порты нужны для того, чтобы через них мог проходить траффик нескольких VLAN'ов.

То есть, такая картинка:



Здесь :

VLAN1 — зеленый, в него входят порты 1, 7, 12, 14, 16 на switch1 и порты 1, 3, 4, 7, 8, 9, 11, 15 на switch2.

VLAN2 — синий, в него входят порты 3, 4, 6, 8, 9, 11, 13, 15, 16 на switch1 и порты 2, 5, 8, 13, 14, 16 на switch2.

VLAN3 — красный, в него входят порты 2, 5, 10, 16 на switch1 и порты 6, 8, 10, 12 на switch2.

Можно было бы соединить свичи тремя патч-кордами (пунктирные линии) по одному на каждый VLAN, но не всегда это возможно и здорово.

Сейчас свичи соединены патч-кордом, который воткнут в 16 порт на первом свиче и в 8 порт на втором. Чтобы по нему могли ходить пакеты всех трёх VLAN'ов и не перемешиваться, патч-корд должен быть воткнут в тегированные порты. Порты 16 на первом и 8 на втором свиче — тегированные, каждый из них принадлежит всем трём VLAN'ам (и наоборот – 3 VLAN есть в этих портах))). На входе в эти порты метка может и ставиться (если нету), но на выходе не снимается!!!

Порты, помеченные одним цветом — нетегированные, в них нет меток, так как метки внутри одной сети не нужны.

Полезная ссылка: http://xgu.ru/wiki/VLAN

VLAN на компе

Можно настраивить с помощью vconfig или с помощью ip link.

Создать VLAN на интерфейсе eth0. Имя VLAN'a – eth0.80, оно по-хорошему должно быть именно в таком формате: интерфейс.тег, ну и тег тут 80, уже понятно:

```
# ip link add link eth0 name eth0.80 type vlan id 80
# ifconfig eth0.80 up
```

Добавить IP:

```
# ip a a 172.16.1.1/24 dev eth0.80
```

VLAN на D-Link

Создать VLAN с названием 45 и тегом 45:

```
# create vlan 45 tag 45
```

Добавить тегированные порты с 8 по 11:

```
# config vlan 45 add untagged 8-11
```

Добавить нетегированный порт 14:

```
# config vlan 45 add tagged 14
```

Удалить порт из VLAN:

```
# config vlan 45 delete 10
```

Посмотреть на VLAN:

```
# show vlan
```

VLAN на Cisco

У Cisco тегированные порты называются транковыми (trunk), а нетегированные native.

создать VLAN:

```
# vlan 2
# name vlan2
```

Добавить тегированные порты:

```
# config vlan2 add tagged 2,25
# exit
```

Изменить что-то с VLAN'ом:

```
# interface vlan2
```

добавить IP:

```
# ip address 10.1.2.1 255.255.255.0
```

Чтоб перевести порт в тегированный режим:

```
$ conf t
# interface fa0/24
# switchport trunk encapsulation dot1q
# swichport mode trunk
```

Разрешить VLAN'ы 1 и 2:

```
# switchport trunk allowed vlan 1,2
```

Включение нетегированного VLAN'а:

```
# switchport trunk native vlan 1
```

Режим по умолчанию, в этом порту не может быть тегированных пакетов:

```
# interface fa0/1
# switcport access vlan 1
```

Ссылки:

http://www.cisco.com/en/US/tech/tk389/tk815/technologies_configuration_example09186a008015f17a.shtml

http://www.cisco.com/en/US/tech/tk389/tk815/technologies_configuration_example09186a008019e74e.shtml

Cisco

У нас в лабе свич серии 3560 (на данный момент).
Как сбросить пароль:
http://www.cisco.com/en/US/products/hw/switches/ps628/products_password_recovery09186a0080094184.shtml

Прицепить циско-кабель (обычно - голубой) в циску и в COM-порт на компе
Войти:

```
$ sudo -i
# minicom
$ en
```

Настройки COM-порта:
9600 (скорость порта)
8-N-1
Чтоб перейти в другой режим, нужно набрать enable или en
команда

```
show
```

тоже что и

```
sh
```

Посмотреть конфигурацию интерфейсов:

```
$ sh ip interface
сводка по интерфейсам:
$ sh int status
```

Посмотреть конфигурацию:

```
$ sh running-config
```

default gateway на Cisco:

```
# ip route 0.0.0.0 0.0.0.0 192.168.1.100
```

ACL (Access Control List) на Cisco
0-99 — стандартные ip'шные acl
100-... — расширенные
… - по типу протокола
… - по MAC
1300-2000 — стандартные, если не хватит.
2000-... - расширенные.

```
# access-list deny 10.10.10.0 0.0.0.255
$ show access-lists
```

По дефолту, что не указано в acl, то drop

Правильная последовательность действий:

1) убрать ip access group из интерфейса
2) снести лист
3) write — запись конфигурации

Что-то:

```
# int fa0/1
# no ip access-group
# access-list 100 deny tcp 1.0.0.0 0.255.255.255 any syn
```

```
# access-list 100 permit ip any any
```
Конфигурировать интерфейсы с 1 по 100:
```
# interface range fa0/1-10
# ip access-group 100 in
# show running-config
```

GRE tunnel на Cisco:

```
# interface tun1
# ip address 10.0.0.1 255.255.255.0
# tunnel source 192.168.1.55
# tunnel destination 192.168.1.115
```

И направлять в туннель всё с 192.168.2.0 в туннель:

```
# ip route 192.168.2.0 255.255.255.0 tun1
```

Полезная ссылка:
https://supportforums.cisco.com/docs/DOC-2569

Как настраиваются WEB-сервера:

Есть свободный web-сервер **Apache**.

```
# apt-get install apache2
```

Настройки в /etc/apache2/apache2.conf
Есть три ветки Apache: 2.0, 2.2, 2.4. Первая уже устарела.
Применимость:

1) Когда нет нагрузки. Он всё умеет, но большую нагрузку не потянет.
2) MySQL + PHP + Apache — стандартная связка. Производительность на PHP — одна из максимальных. Но PHP дыряв.

Конфигурация распилена на кусочки по файлам. Прежде, чем настраивать что-то (особенно – нестандартное) стоит почитать инструкцию к нужной версии.
Файлик apache2.conf настраивается, обычно, один раз, потом добавляются сайты в sites-avaliable.
Есть modes-avaliable и modes-enabled. В modes-enabled — symlink'и, очень удобно включать/выключать настройки, не удаляя файлы с конфигами. (можно командами a2ensite xxx.ru)
Некоторые настройки:

```
Timeout – столько сервер ждет сообщение.
KeepAlive on – соединение живет не один запрос (не устанавливается лишнее соединение). Всего один реальный TCP-коннект.
```

MPM — multi processing module для Apache.
mpm_profork_module — заточен на CGI, делает fork()+exec(), не течет память. Есть, например, 5 процессов. 1 отдали, он отработал, умер, fork'нули новый. Если стоит 0 — то их неограниченное количество.
mpm_worker_module — основной сейчас. Смешанный: многопоточный и многопроцессный.
MaxRequestsPerChild — ставить не 0, если работаем с PHP, ибо обычно программисты PHP не умеют работать с памятью и она «течет». Процесс столько запросов отработал, умер, запустили новый. Но не мельчить с числом, чтоб не дергало слишком часто.))

Для Apache:

Ports.conf — файлик. Listen 80 — сервер будет слушать на 80 порту. Можем поменять на 8080, например.
Mods-avaliable — там файлики с конфигурацией различных модулей.
Есть mods-enable — символические ссылки на файлики в директории в mods-avaliable.
a2enmod
a2dismod
a2dissite

apache2.conf
ServerRoot — основная директория, где лижит вся конфигурация

PidFile — имя файлика, где будет лежать proccessID, чтоб его можно было легко прибить. $ - это означает переменную окружения.
Timeout — сколько ждать до сброса соединения.
KeeoAlive On -
MaxKeepAliveRequests — на одно соединение — столько request'ов.

User
group — от каких пользователя и группы будет запущен процесс.
AccessFileName — внутри файла, кот. указан можем переопределить права какие-то.
Там в файликах директивы какие-то можно писать.
DefaultType text/plain — обычно, text/html бывает и jpg и еще что-то. Если этого заголовка программа не написала, то этот заголовок будет подставлен по умолчанию.
HostameLookups Off — надо записывать не Apache.org, а IP. Если включим, даст лишнюю нагрузку на сервер.
Include /etc/apache2/mods-enabled — можем включать файлики в конфиг.
LogFormat — объявление того, что и как писать в лог.
ErrorDocument — можно поставить обработчик на ошибки. Можно указать текст, html'ку, скрипт.

/etc/apache2/sites-enabled — включаем эту папку и лезем смотреть теперь туда.
Virtual хосты. Чтоб несколько сайтов было на сервере.
ServerAdmin
ServerAliad
DocumentRoot — все файлы, связанные с сайтом.

В mods-enabled есть dir.conf — попытается найти файлики в этом порядке и выполнить (когда?!)

reload — перечитать конфигурацию без остановки.

Директива Alias. Пишем алиас на директорию, потом описываем, как ведет себя сервер в этой директории.
LocationMatch — для всех файликов, подходящих регулярке делаем zip-архивы.
RewriteEngine On — можем переписать немного запрос, прежде, чем он пройдет на сервер.

OSPF

Протокол динамической маршрутизации, протокол по состоянию канала.
Работает по алгоритму Дейкстры.
С помощью Hello-пакетов строит базу данных канального уровня, где описана структура сети. Еще есть база соседних роутеров и база маршрутизации (маршрутов). База маршрутизации вычисляется на основе БД канального уровня. БД канального уровня одинакова для всех роутеров а area, а БД маршрутизации каждый сам вычисляет для себя.
Чтоб не убить сеть тучей Hello-пакетов, можно выделить назначенный и резервный маршрутизаторы (DR - designated router и BDR — backup designated router). Все устанавливают соединение только с ними. Эти маршрутизаторы разошлют необходимые маршруты другим маршрутизаторам.
Нам от OSPF нужно... ну, та самая карта сети. Ну, и соответственно, маршруты. И все маршрутизаторы должны поддерживать OSPF.

BGP

Для обмена между автономными системами. Для обнаружения маршрутных петель.
Соединение между сетями по tcp, port 179.
Может использоваться для маршрутизации и внутри автономных систем.
Основное, для чего нужен — обмен инфой о доступности отдельных сетей или хостов. Эта инфа должна содержать маршруты до сети (промежуточные системы).
Периодически хосты сообщают о своей работоспособности (например, в случае ошибки). BGP не требует периодического обновления всей маршрутной таблицы, хотя BGP поддерживает маршрутную таблицу всех возможных трактов к какой-нибудь конкретной сети, в своих сообщениях о корректировке он объявляет только об основных — оптимальных маршрутах.
Есть метрики. С помощью них определяются предпочтения маршрута. Они задаются администратором в конфигах.
Вместо Hello, открывающие сообщения. Есть сообщения о корректировках.

Настройка OSPF и BGP

Сначала нужно скачать Quagga — пакет, поддерживающий протоколы динамической маршрутизации, то есть OSPF и BGP в их числе. Динамическая маршрутизация — это когда вы описываете только сети рядом с собой, а маршруты появляются до всех. Вы не пишете эти маршруты руками.
Итак:

```
# apt-get install quagga
# cd /etc/quagga
# ls
```

Смотрите, какие файлики есть. Должны быть: bgpd.conf, daemons, debian.conf, ospfd.conf, zebra.conf. Если какого-то нет, создаете его. Например:

```
# gedit bgpd.conf
```

Если вы сохраните этот файл, он создастся.
В daemons:

```
zebra=yes
bgpd=yes
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

В bgpd.conf:

```
hostname place9 !означает имя, мы брали номер узла на рисунке
router bgp 65002 !означает автономную систему, о номерах нужно договориться между собой, ну и что мы роутер bgp
bgp router-id 172.16.91.9 !любой из IP того компа, на котором настраиваете bgp
network 172.16.89.0/24 !описываете все сети, которые есть у вас на компе, то есть, которые вы видите напрямую
network 172.16.29.0/24 !описываете все сети, которые есть у вас на компе, то есть, которые вы видите напрямую
network 172.16.91.0/24 !описываете все сети, которые есть у вас на компе, то есть, которые вы видите напрямую
neighbor 172.16.91.10 remote-as 65003 !указываете соседнюю автономную систему, если их несколько, то таких строчек будет несколько
redistribute connected !то есть, чтобы вам отдавались маршруты автоматически
```

В ospfd.conf:

```
hostname place9 !опять имя компа
router ospf !мы – роутер ospf
ospf router-id 172.16.89.9 !любой из IP того компа, на котором настраиваете ospf
network 172.16.29.0/24 area 65002 !подсеть из автономной системы. Можно описать все подсети из автономной системы, а можно только те, которые видим напрямую, если какой-то роутер ospf опишет остальные
network 172.16.89.0/24 area 65002 !подсеть из автономной системы
redistribute connected !то есть, чтобы вам отдавались маршруты автоматически
log stdout !логи в стандартный поток вывода
```

В zebra.conf:

```
hostname place9
interface eth0.29 !описываем все vlan, которые есть на компе. То есть, не обязательно vlan, просто все интерфейсы
interface eth0.910
interface eth0.89
```

После того, как написали конфиги, перезагружаем quagga:

```
# service quagga restart
```

Не забудьте включить форвардинг пакетов! (см. где-то выше)

Чтоб чего-то посмотреть, или не писать конфиги ручками, есть виртуальный терминал. Он почти как в циске.

«Скажи втыщь!» это означает:

```
# vtysh
Теперь нажмите q.
```

```
Вы в том самом терминале.
Посмотреть что-то нужное:
# show ip bgp
# show ip route bgp
# show ip ospf
# show ip ospf neigbor
# show ip ospf route

Чтоб выйти на уровень вверх (возможно, после каждой команды это
будет нужно), нажмите q.
```

Полезные ссылки:
http://www.opennet.ru/base/cisco/bgp_zebra.txt.html

http://sudouser.com/nastrojka-marshrutizacii-s-pomoshhyu-quagga-zebra.html

http://www.hilik.org.ua/%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC-ospf-%D0%B2-quagga/