Содержание

Что такое MINT ?

Аббревиатура MINT расшифровывается как Mesh Interconnection Network Technology, обозначая этим технологию построения сетей с произвольными связями. Что это такое и как это можно использовать мы и рассмотрим на примерах, от простого к сложному.

Самое главное свойство технологии MINT заключается в том, что она позволяет представить любую беспроводную (а иногда и проводную, и даже гетерогенную, как мы увидим дальше) сеть как единый одноранговый сегмент Ethernet, а радио-интерфейс, подключенный к такой сети, – как обычный интерфейс Ethernet, хотя и виртуальный. Виртуальный потому, что при его настройке нам всё же придётся учитывать специфические особенности такого интерфейса, такие как частота, тип

модуляции и прочее. Но приложения, которые будут с ним работать, не заметят разницы с обычными интерфейсами.

Начальная настройка

Итак, перейдём к делу.

Допустим, у нас есть два радиомаршрутизатора InfiNet Wireless, у каждого из которых на борту имеется один «настоящий» Ethernet-интерфейс и один радиоинтерфейс.

Для начала, попробуем установить связь между двумя устройствами по радио.

Для этого нам нужно настроить параметры радио интерфейсов таким образом, чтобы они могли физически «общаться» друг с другом. Проще всего присвоить всем параметрам одинаковые значения.

Например:

```
rf rf4.0 freq 5320 bitr 54000 sid 10101010 pwr 63
```

Теперь на обоих устройствах вводим «магическую» команду:

```
mint rf4.0 start
```

И … ничего не произошло.

Но на самом деле устройства уже обменялись серией пакетов, установили связь, выбрали оптимальные параметры для передачи данных и готовы к работе.

Установление соединения занимает меньше секунды.

Убедиться в этом можно посмотрев карту (список) соседей

командой:

```
mint map
================================================================================
Interface rf4.0
Node  00028AE1DAC1  "", Id 32456, NetId 0, (mesh)
Freq  5320, Sid 10101010, autoBitrate 54000, Noise floor -95

 1 active neighbors:
 -------------------
  Id      Name             Node     Cost   Sig     Bitr   R  E Options
----- --------------  ------------ ----  -----  ------- -- -- -------------
33521                 001195FD8FFB   30  15/18   36/36   0  0 /mesh/
----- --------------  ------------ ----  -----  ------- -- -- -------------
```

Что мы видим в этой таблице?

В первых 3 строках указано имя интерфейса (rf4.0) на котором запущен протокол MINT, его сетевой (MAC) адрес, идентификатор узла (32456), его тип (mesh), а также ключевые параметры радиоинтерфейса.

Далее мы видим, что наше устройство (в терминологии MINT – узел (node)) имеет одного соседа (active neighbor). Ниже в таблице показаны:

- Id – идентификатор соседнего устройства;
- Name – имя устройства (в данном случае оно не задано);
- Node – сетевой адрес соседа;
- Cost – стоимость соединения;
- Sig – уровни принимаемого и отправляемого сигналов в dB;
- Bitr – текущие битовые скорости (bitrate), на которых идет происходит обмен данными в эфире с рассматриваемым нодом;
- R и E – индикация уровня переповторов и ошибок в процентах;
- Options – дополнительные опции соседнего узла, речь о которых пойдет ниже (в данном случае, соседний узел работает в режиме «mesh»).

Стоимость –

это ключевой параметр каждой отдельной связи, который позволяет протоколу MINT выбирать оптимальные пути передачи данных в разветвлённой сети.

Физический смысл стоимости – это прогнозируемое время передачи пакета на данном направлении в условных единицах. Чем время (стоимость) меньше, тем лучше. Стоимость соединения будет постоянно корректироваться во время работы, учитывая энергетические параметры соединения (уровни сигналов), тип модуляции и скорость передачи, количество переповторов и ошибок, загрузку канала и другие параметры, позволяя быстро переключаться на альтернативный маршрут, если стоимость его окажется более выгодной.

Конечно, в нашем случае, когда устройства всего два, этот параметр не имеет большого значения, но мы ещё поговорим о нём позднее.

Идентификатор узла, это вспомогательный числовой параметр, который может назначаться администратором для более наглядного отображения карты соседей или узлов. Списки будут отсортированы именно по идентификаторам. Никакой функциональной нагрузки этот параметр не имеет и протоколом не используется.

Вы можете просто пронумеровать все узлы в сети подряд или придумать какую-то другую систему их идентификации. Например, можно в последней цифре хранить номер интерфейса, а в остальных порядковый номер узла. К примеру идентификатор 10 будет означать узел номер 1 использующий интерфейс номер 0. А идентификатор 321 – это узел 32 использующий интерфейс номер 1. Зачем это может пригодиться мы увидим позднее, когда будем рассматривать многомодульные и многочастотные mesh системы. Идентификатор может принимать значения от 0 до 65535. Если этот параметр не задан администратором, то он устанавливается равным заводскому серийному номеру устройства.

Итак, воспользуемся полученными знаниями и присвоим нашим узлам подходящие идентификаторы, а заодно дадим им имена.

Узел 1 получит идентификатор 10 и имя Node_10.

```
mint rf4.0 –nodeid 10
mint rf4.0 –name “Node_10”
```

Узел 2 получит идентификатор 20 и имя Node_20.

```
mint rf4.0 –nodeid 20
mint rf4.0 –name “Node_20”
```

Теперь карта соседей выглядит более осмысленно:

```
mint map
================================================================================
Interface rf4.0
Node  00028AE1DAC1  "Node_10", Id 10, NetId 0, (mesh)
Freq  5320, Sid 10101010, autoBitrate 54000, Noise floor -95

 1 active neighbors:
 -------------------
  Id      Name             Node     Cost   Sig     Bitr   R  E Options
----- ---------------  ------------ ----  -----  ------- -- -- -------------
00020 Node_20          001195FD8FFB   30  15/18   36/36   0  0 /mesh/
----- ---------------  ------------ ----  -----  ------- -- -- -------------
```

Однако возникает вопрос, связь-то мы установили, но как её использовать?

Самый простой способ убедиться, что связь действительно есть, это назначить на интерфейсы обоих устройств IP адреса из одной подсети и послать ping.

Например,

на первом устройстве

```
ifconfig rf4.0 10.0.0.1/24 up
```

а н

а втором:

```
ifconfig rf4.0 10.0.0.2/24 up
```

Можете убедиться сами, связь действительно есть.

В принципе, уже сейчас вы можете использовать этот линк для практических целей, настроив маршрутизацию между интерфейсами eth0 и rf4.0.

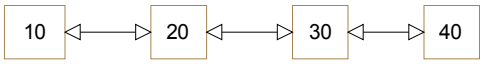
Но наша задача сейчас состоит не в этом.

Попробуем добавить к нашей сети ещё одно устройство, а лучше – сразу два.

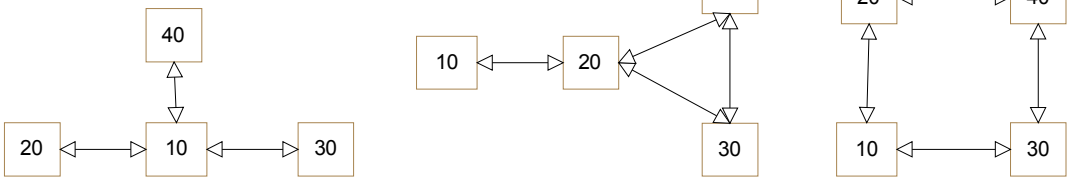
Топология MINT-сети и пути прохождения пакетов

Новые устройства настроим аналогично предыдущим, дав им идентификаторы 30 и 40, имена Node_30 и Node_40 и адреса 10.0.0.3/24 и 10.0.0.4/24, соответственно.

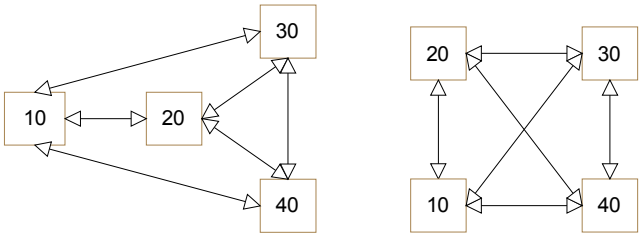
Теперь, если вы не предпринимали никаких специальных мероприятий для получения определённой топологии, наша сеть может выглядеть так:



или так:



или даже так:



Это уж как повезёт.

Соответственно в списке соседей узла 10 мы можем увидеть от одного до трёх узлов.

Однако, какую бы топологию не приняла сеть, пакеты от узла 10 к узлу 40 будут ходить как ни в чём не бывало. Но какими путями они будут ходить ?

Например, в последнем примере между узлами 10 и 40 имеется 5 альтернативных путей:

(*кстати, последние два примера тождественны, хотя и выглядят по-разному*)

- 10->40
- 10->20->40
- 10->30->40
- 10->20->30->40
- 10->30->20->40

Какой из них самый лучший?

Первый?

Может быть.

А может и нет.

Может оказаться, что самым быстрым путём будет последний, самый длинный.

Всё зависит от того, какие сложились условия на каждом конкретном участке сети.

Автомобилисты знают, что короткий путь не всегда самый быстрый, а левый поворот иногда лучше заменить тремя правыми.

Вот здесь-то и пригодится стоимость (Cost) соединения.

MINT просчитает все маршруты и выберет из них самый оптимальный по сумме стоимостей соединений на каждом маршруте. И будет делать это постоянно по мере изменения условий в сети и её топологии, так чтобы маршрут между любыми двумя узлами был всегда оптимален. Более того, MINT умеет прогнозировать ситуацию, заблаговременно переключаясь на резервные пути если есть вероятность дальнейшего ухудшения ситуации. Переключение происходит настолько быстро, что, кроме самых тяжёлых случаев, это не приводит к потере данных или разрыву соединений протоколов верхнего уровня. На самом деле, конечно, не нужно просчитывать абсолютно все пути, это было бы слишком долго и ресурсоёмко. В действительности, каждый узел доверяет своим соседям, которые уже сделали свою часть работы, и выбирает из них того, кто предлагает самый выгодный маршрут до искомого узла. Таким образом, задача сводится к тому, чтобы максимально реалистично оценить стоимость соединений со своими собственными соседями и добавить её к предлагаемым ими маршрутам.

Посмотреть полный список всех узлов сети можно командой:

```
mint map full
================================================================================
Interface rf4.0
Node  00028AE1DAC1  "Node_10", Id 10, NetId 0, (mesh)
Freq  5320, Sid 10101010, autoBitrate 54000, Noise floor -95
```

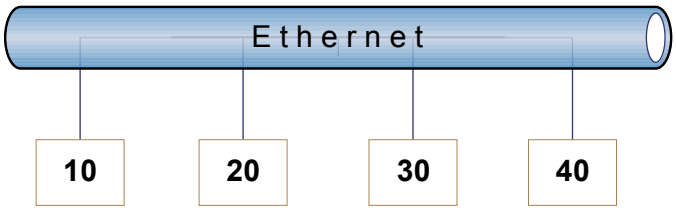
```
2 active neighbors:
 ------------------
  Id      Name              Node      Cost   Sig     Bitr   R  E Options
----- ---------------  ------------ ----  -----  ------- -- -- -------------
00020 Node_20          001195FD8FFB   40  15/18   36/36   0  0 /mesh/
00030 Node_30          000435FFC283  110  15/18   36/36   0  0 /mesh/
----- ---------------  ------------ ----  -----  ------- -- -- -------------

 4 Routes:
---------
 00010: 000435FF9359 --> 00000: 00028AE1DAC1 Cost=0     Zone=0
 00020: 001195FD8FFB --> 00020: 001195FD8FFB Cost=40    Zone=1
 00030: 000435FFC283 --> 00030: 000435FFC283 Cost=110   Zone=1
 00040: 000435FFB65B --> 00020: 000435FF9357 Cost=200   Zone=2
```

Из этой таблицы можно увидеть, что два узла 20 и 30 являются нашими непосредственными соседями, а узел 40 доступен через соседа Node_20, имеет суммарную стоимость доставки пакетов 200 и находится на расстоянии 2-ух пересылок («хопов», от англ. «hop»). Непосредственно доступные соседи находятся на расстоянии одной пересылки.

Важно помнить, что каждый узел самостоятельно рассчитывает стоимость соединения со своими соседями и стоимость передачи пакетов до каждого узла сети, поэтому, прямой и обратный путь между двумя узлами могут не совпадать. Более того, даже соседи, имеющие непосредственную связь друг с другом, для реальной передачи данных могут выбрать не прямой путь, а через промежуточный узел (узлы), если при этом скорость и надёжность доставки пакетов окажется выше.

Но что самое главное, независимо от текущей и даже меняющейся топологии сети, её внутренняя структура будет скрыта от протоколов и приложений верхнего уровня. Технология MINT позволяет представить всю сеть как один локальный сегмент сети Ethernet.



Это свойство позволяет без каких-либо изменений применять в MINT-сети любые приложения и сетевые технологии, работающие в Ethernet сетях, в том числе, протоколы, требующие возможности многоадресной или широковещательной рассылки.

Основные способы передачи траффика в MINT

Архитектура MINT поддерживает два основных способа передачи трафика – маршрутизация и коммутация.

Каждый входящий в систему пакет сначала попадает под пристальный взгляд коммутатора. Если тот признает пакет «своим», то он будет скоммутирован в соответствующую группу. Если же нет, то пакет отдаётся модулю маршрутизации для принятия окончательного решения, что же с ним делать. Учитывая это свойство можно строить комбинированные схемы, когда часть трафика маршрутизируется, а другая коммутируется. Всё будет определяться правилами коммутатора.

Маршрутизация

Маршрутизатор может работать над любыми сетевыми интерфейсами, поддерживающими протокол IPv4, посредством соответствующих протоколов передачи IP датаграмм через конкретные типы сетевых интерфейсов. При этом распространение широковещательных broadcast и multicast пакетов ограничено только локальным сегментом соответствующего интерфейса. Это позволяет полностью отделить друг от друга пространство маршрутизации и коммутации (свитчевания).

Маршрутизация – традиционный способ передачи трафика для устройств InfiNet Wireless. Архитектура MINT унаследовала от RMA все возможности подсистемы маршрутизации и дополнила их новыми. За дополнительной информацией рекомендуем обратиться к руководству по ОС WANFleX.

Коммутация

По-умолчанию, устройства InfiNet Wireless являются маршрутизаторами, однако в рамках архитектуры MINT появилась возможность передавать часть или весь трафик путем коммутации пакетов.

Иногда это бывает нужно для передачи немаршрутизируемых протоколов, иногда просто в силу более простой конфигурации или нежелания администратора по каким-либо причинам использовать маршрутизацию.

Оба этих режима, и маршрутизация и коммутация могут быть активны одновременно.

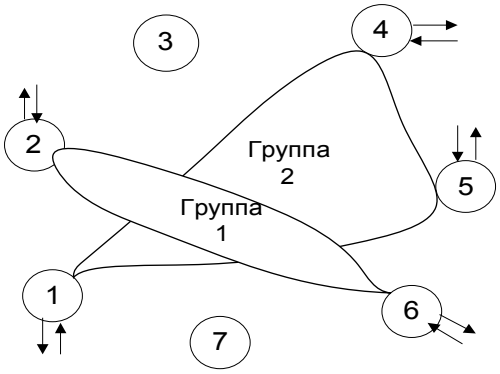
Принципы работы коммутатора

Коммутатор (switch) работает над любыми сетевыми интерфейсами типа Ethernet, поддерживающими адресацию и заголовки в формате Ethernet II (DIX) или 802.3, в том числе виртуальный Ethernet, образованный с помощью технологии MINT.

Настройка коммутатора сводится к описанию так называемых групп коммутации, обозначаемых уникальным числовым идентификатором ИКГ (идентификатор коммутационной группы, 1-4094).

Каждая группа включает в себя 2 и более сетевых интерфейсов типа Ethernet (ethX, rfX, tunX) и набор правил (фильтров) позволяющих максимально точно выбрать из общего потока ту часть данных, которая будет обрабатываться данной

группой коммутации. Каждое устройство может иметь несколько групп коммутации, в том числе построенных на одинаковых или пересекающихся множествах сетевых интерфейсов. Группы коммутации, созданные на разных узлах сети MINT и имеющие одинаковый идентификатор, образуют зону коммутации. Понятие зоны коммутации существует только внутри сегмента сети MINT.



Таким образом вся совокупность узлов сети MINT может рассматриваться как единый виртуальный распределённый коммутатор, где граничные узлы играют роль внешних портов. **Задача коммутатора – обеспечить прозрачную передачу пакетов из одного внешнего порта в другой** (другие). Важно помнить, что группы коммутации нужно создавать только на тех узлах, на которых они действительно нужны, то есть на тех, где пакеты из внешней сети попадают внутрь MINT и наоборот. На промежуточных узлах настройка групп коммутации (и коммутатора вообще) не требуется, независимо от топологии и от того, имеют эти узлы непосредственную связь или нет.

Направить входящие пакеты в соответствующую группу поможет гибкая система правил (фильтров) позволяющая классифицировать поступающие данные, путём анализа любой информации которую можно извлечь из протокольных заголовков пакета (VLAN tag, тип протокола, адреса (MAC и IP), порты, специфические опции и т.д.). На основании полученной информации можно назначить пакету подходящую группу коммутации. Будучи однажды помещённым в конкретную группу, **пакет никогда не выйдет за пределы соответствующей зоны коммутации**, кроме как покинув сеть через один из внешних портов. Отсюда, в частности следует, что **правила для назначения пакету соответствующей группы применяются только в тот момент когда пакет попадает внутрь MINT сети через один из внешних портов**. При выходе из сети никаких правил не требуется, поскольку пакет уже принадлежит какой-то группе, то он будет автоматически скоммутирован во внешний порт(ы) входящий в соответствующую зону коммутации.

Важное исключение: собственные пакеты порождаемые внутренними узлами сети MINT (например RIP / OSPF или ping) не принадлежат никакой группе коммутации, поэтому они не могут покинуть сеть **путём коммутации** ни через один внешний порт. Однако, если это всё таки требуется, то на соответствующем узле может быть назначена специальная **admin** группа, для передачи такого рода трафика.

Любую группу можно перевести в режим **repeater**, в этом случае, коммутатор превращается в простой необучаемый повторитель/ретранслятор (HUB), прозрачно передавая фреймы между всеми интерфейсами входящими в группу,

без какой-либо оптимизации. Этот режим наиболее эффективен на соединениях точка-точка.

Кроме того, специальные правила интерфейсов позволяют выполнять гибкое управление VLAN тегами, по желанию администратора назначая, удаляя или не меняя значения VLAN ID.

Коммутатор обеспечивает прозрачный проброс фреймов Ethernet (в том числе 802.1q VLAN, а также широковещательных broadcast и multicast пакетов) в пределах своей зоны коммутации между всеми интерфейсами входящими в соответствующую группу. Важно помнить, что **коммутатор работает только с номерами групп**, хотя пакеты внутри группы могут принадлежать разным стандартам (802.1Q VLAN, 802.3, Ethernet II, 802.2 и т.д.)

Опираясь на уровень конвергенции MINT, коммутатор выполняет оптимизацию передачи широковещательного трафика через сеть MINT, а также позволяет выбирать оптимальный путь доставки при наличии альтернативных путей.

Если проводить аналогию с проводными коммутаторами, имеющими несколько физических портов, аналогом порта в беспроводном коммутаторе является пара «сетевой интерфейс/шлюз». Для проводных интерфейсов шлюз (gateway) отсутствует, а для виртуального интерфейса Ethernet/MINT, шлюзом является любой (подходящий) узел беспроводной MINT сети входящий в соответствующую зону коммутации. Коммутатор автоматически распознаёт появление и изменение новых шлюзов сети и использует их для оптимизации доставки трафика.

Существует два основных типа групп коммутации – обычная и транковая.

Обычная группа – универсальна, он способна передавать любые типы пакетов (с тегами и без них) без всяких изменений. Можно сказать, что обычная группа это труба, в которую можно закачивать что угодно. Узлы находящиеся в одной группе подключены к этой трубе с разных сторон. Некоторым более понятным кажется сравнение с куском провода – узлы в одной группе как бы соединены проводами (патчкордами) друг с другом.

Транковая группа – это такая же труба, только на входе у неё находится сепаратор, который разделяет входящий из проводного сегмента поток по разным подгруппам в зависимости от VLAN тега каждого пакета. Это существенно облегчает конфигурацию коммутатора когда нужно «раздать» VLAN-ы нескольким абонентам.

Однако, перейдём от слов к делу.

Настройка коммутатора для простейших топологий

Проверим работу коммутатора на примере самой простой топологи сети: точка-точка

Связь у нас уже настроена, приступим к настройке коммутатора.

На обоих устройствах вводим команды:

```
sw group 1 add eth0 rf4.0
sw group 1 start
sw start
```

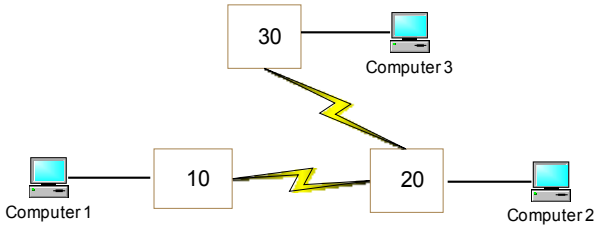
И

пока этим ограничимся. В результате мы создали группу с номером 1, подключили к ней два интерфейса, сделали группу активной и запустили коммутатор. После выполнения этих команд на обоих беспроводных устройствах, компьютеры 1 и 2 оказываются связаны в единый сегмент локальной сети.

Это самый простой вариант использования коммутатора, когда в одну группу объединяются два (или более) интерфейса и никаких правил фильтрации не используется. В этом случае все пакеты, входящие в коммутатор через проводной интерфейс Ethernet со стороны Computer1, будут доставлены на противоположный конец и доставлены адресатам находящимся в сети Computer2. И наоборот.

Такая **группа является абсолютно прозрачной** для передаваемого трафика, независимо от типов протоколов, наличия или отсутствия VLAN-тегов и их количества.

Можно усложнить задачу, добавив ещё один беспроводной узел, с такими же настройками коммутатора.

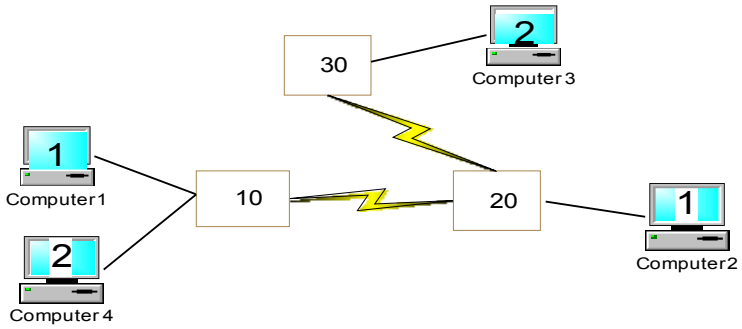


При этом независимо от получившейся топологии все три компьютера будут объединены в один единый сегмент локальной сети.

Изоляция трафика при помощи правил классификации пакетов

Предположим, что мы хотим объединить попарно две независимых подсети.

Свяжем Computer1 с Computer2, а Computer3 с Computer4, но так чтобы их трафик не перемешивался.



Поместим компьютеры C1 и C2 в группу коммутации 1, а C3 и C4 в группу 2.

Тогда на устройстве 30 мы имеем:

```
sw group 2 add eth0 rf4.0
sw group 2 start
sw start
```

На устройстве 20:

```
sw group 1 add eth0 rf4.0
sw group 1 start
sw start
```

Вроде всё просто.

Но вот на устройстве 10 нам понадобится 2 группы коммутации, что-то вроде этого:

```
sw group 1 add eth0 rf4.0
sw group 1 start

sw group 2 add eth0 rf4.0
sw group 2 start

sw start
```

Как же разделить поток входящий с Ethernet на две разные группы?

Воспользуемся правилами фильтрации.

Сначала нужно решить, по какому критерию мы будем различать эти две группы.

Предположим, что компьютеры C1 и C2 находятся в IP подсети 10.0.1.0/24, а C3 и C4 в сети 10.0.2.0/24

Тогда на устройстве 10 можно составить такие правила:

```
switch list RULE1 match add 'net 10.0.1.0/24'  # создаём правило фильтрации 1
switch list RULE2 match add 'net 10.0.2.0/24'  # создаём правило фильтрации 2

switch group 1 add eth0 rf4.0
switch group 1 rule 10 permit match RULE1      # свитчуем только сеть 1.0/24
switch group 1 deny                            # всё остальное запрещено
switch group 1 start

switch group 2 add eth0 rf4.0
switch group 2 rule 10 permit match RULE2      # свитчуем только сеть 2.0/24
switch group 2 deny                            # всё остальное запрещено
switch group 2 start

switch start
```

Для нашей задачи этого достаточно.

Пакет с компьютера 1, попав на узел 10 будет иметь адрес источника из сети 10.0.1.0/24, поэтому он, в соответствии с нашими правилами, будем помещён в группу коммутации 1, доставлен на узел 20 и там скоммутирован в проводной интерфейс Ethernet для доставки компьютеру 2. Обратный пакет, попав на устройство 20, без всяких проверок (поскольку других источников нет) будет помещён в группу 1 и доставлен на узел 10, а затем и компьютеру 1.

Цель достигнута.

Если требуется разделять потоки по иным критериям нежели IP адреса, то можно составить другие правила.

Например, на узле 10 мы могли бы использовать для идентификации пакетов VLAN-теги, если таковые используются в этой сети.

Например, предположим компьютер 1 посылает пакеты с VLAN-тегом 100, а компьютер 4 с тегом 400. Тогда правила могут выглядеть таким образом:

```
switch list VL100 numrange add 100             # создаём правила фильтрации
switch list VL400 numrange add 400             # имена могут быть любыми

switch group 1 add eth0:100 rf4.0:0
switch group 1 rule 10 permit VLAN VL100       # свитчуем только VLAN 100
switch group 1 deny                            # всё остальное запрещено
switch group 1 start

switch group 2 add eth0:400 rf4.0:0
switch group 2 rule 10 permit VLAN VL400       # свитчуем только VLAN 400
switch group 2 deny                            # всё остальное запрещено
switch group 2 start

switch admin-group 1
switch start
```

Обратите внимание, добавляя в группу интерфейсы мы написали их в виде rf4.0:0 и eth0:100. Это сделано для демонстрации возможностей манипулирования VLAN тегами. Такая запись означает, что при отправке в интерфейс rf4.0 пакеты соответствующей группы будут отправлены без VLAN тега (untagged), даже если

он был во входящем пакете. Это может понадобиться, например, если на другой стороне сети (компьютер 2) не поддерживаются VLAN теги. Того же эффекта можно было добиться сделав соответствующую запись на устройстве 20 при отправке в проводной Ethernet:

```
switch group 1 add eth0:0 rf4.0
```

С другой стороны, пакет приходящий из MINT сети с номером группы 1, будет отправлен в проводной сегмент помеченным VLAN ID 100, как того и требуют наши правила. **Манипуляции с VLAN тегами производятся при отправке пакета** через соответствующий интерфейс.

Если модификатор не используется, то пакеты проходят без изменения, модификатор **:0** сбрасывает любые теги, если они были. Модификатор **:N** принудительно маркирует пакет тегом **N**, независимо от его исходного значения.

Применение транковых групп

Последний пример можно упростить, создав одну **транковую** группу:

```
switch list VLANS numrange add 100,400          # создаём правила фильтрации

switch group 1 add eth0 rf4.0:0
switch group 1 VLAN VLANS                       # свитчуем только VLANS
switch group 1 trunk on                         # транковая группа
switch group 1 start

switch admin-group 1
switch start
```

На устройстве 30:

```
sw group 100 add eth0 rf4.0
sw group 100 in-trunk 1
sw group 100 start
sw start
```

На устройстве 20:

```
sw group 400 add eth0 rf4.0
sw group 400 in-trunk 1
sw group 400 start
sw start
```

Транковая группа обладает тем замечательным свойством, что автоматически помещает пакеты имеющие VLAN tag в группу с таким же номером.

И наоборот, пакеты которые приходят из MINT сети в разных группах, отправляет в проводной сегмент маркируя их VLAN id совпадающим с номером группы в которой пришёл пакет. Это свойство позволяет сильно упростить конфигурацию когда требуется раздать много VLAN по разным направлениям.

В нашем примере, пакет вошедший в устройство 10 через интерфейс eth0 и помеченный VLAN тегом N, будет автоматически помещён в группу коммутации с

номером N (создавать её на этом устройстве не требуется) и отправлен в MINT сеть. С другой стороны, пакет пришедший из MINT сети с уже назначенной группой N, будет отправлен в проводной интерфейс помеченным VLAN тегом N. Собственный номер транковой группы (1) является её уникальным идентификатором в пределах MINT сети. Поскольку в сети может быть несколько транковых групп, то неизбежно возникает проблема пересечения номеров динамически создаваемых групп. Например, в сети может быть ещё один узел работающий в группе 100, но этот номер принадлежит области действия другой транковой группы. Чтобы разрешить это противоречие, на оконечных узлах следует явно указывать в состав какой транковой группы они входят: "`sw group 100 in-trunk 1`" - группа 100 находится в транковой области 1. Все транковые группы в пределах одной MINT-сети должны иметь разные номера (за исключением одного случая, который рассматривается далее) .

Обратите внимание на форму записи правила для обработки VLAN, которую мы использовали в предыдущем примере:

```
switch group 1 VLAN VLANS
```

Эта запись равнозначна следующим двум:

```
switch group 1 rule 10 permit VLAN VLANS
switch group 1 deny
```

Разница в том, что во втором случае можно, теоретически, назначить и другие правила разрешающие обработку пакетов данной группой, но на практике, при работе с VLAN это не требуется, поэтому первая форма гораздо компактнее и понятнее.

Следует иметь ввиду, что если в правилах транковой группы указан список VLAN которые она должна обрабатывать (`sw group 1 VLAN VLANS`), то нетегированные пакеты не будут обрабатываться этой группой. Если же необходимо передавать одновременно и тегированные и нетегированные пакеты, то нужно либо совсем убрать правила классификации, либо воспользоваться расширенной формой **xVLAN –** (`sw group 1 xVLAN VLANS`). В этом случае, нетегированные пакеты отправляются в MINT сеть с собственным номером транковой группы.

Локальные пакеты, порождаемые узлами MINT сети (например, OSPF или arp) не принадлежат никакой группе, поэтому по-умолчанию они отправляются в проводной сегмент нетегированными. Если же администратору требуется получать их в составе какого-либо VLAN, то его можно назначить специальной командой **switch local-tag N**. Пакеты не имеющие группы будут отправляться в проводной сегмент с VLAN tag равным параметру local-tag. И наоборот, пакет принятый через проводной интерфейс и имеющий VLAN ID равный local-tag, будет считаться локальным по отношению к сети MINT (не принадлежащим никакой группе).

«Правила хорошего тона» при настройке коммутатора

Следует подчеркнуть, что выбор группы коммутации определяется только её собственными правилами классификации. Если на устройстве создано несколько групп коммутации, то это задача администратора составить такие правила классификации, чтобы **каждый** пакет мог быть **однозначно** отнесён к той или иной группе. Например, в следующем примере часть пакетов может быть

обработана обоими группами. В этом случае правильная работа коммутатора не гарантируется.

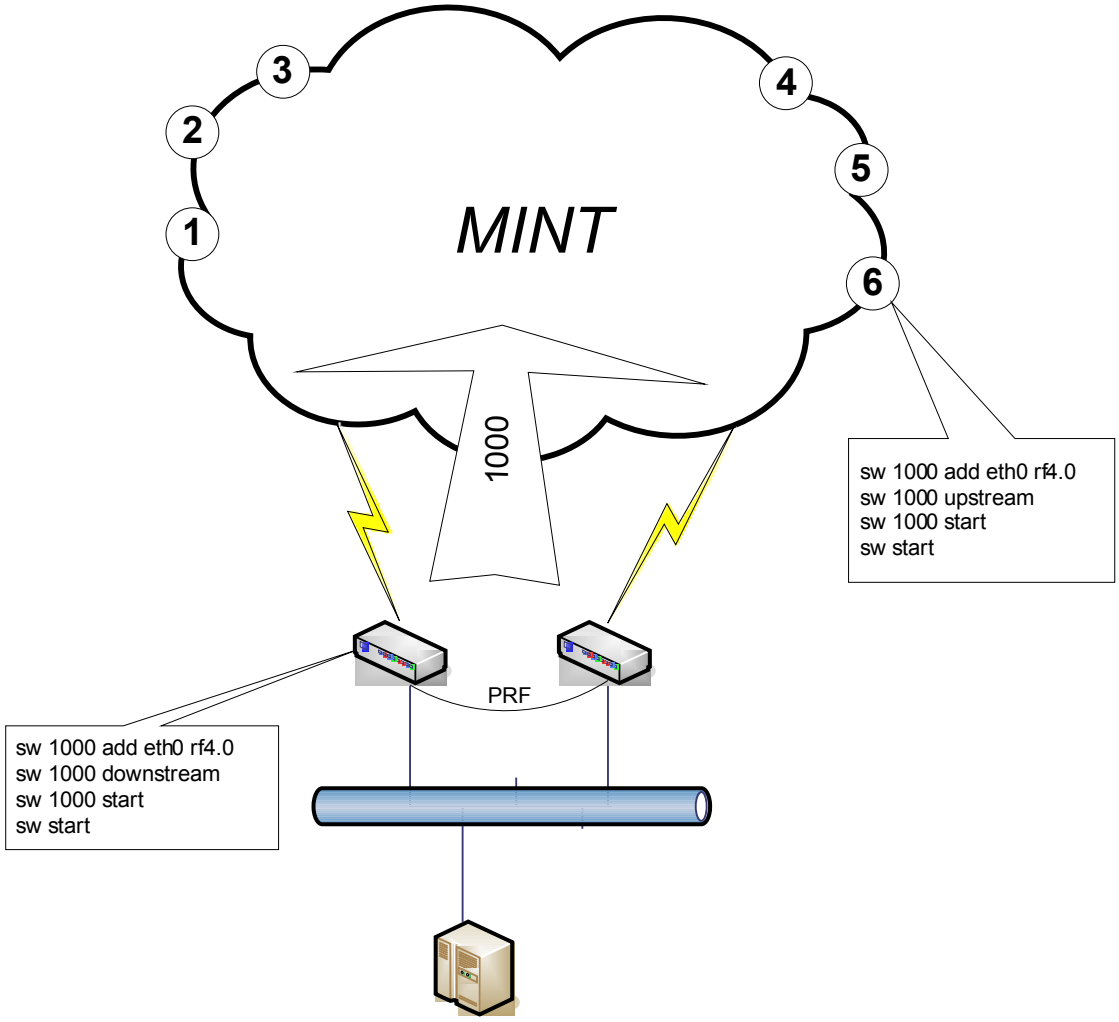
```
switch group 1 add eth0 rf4.0
switch group 1 rule 10 permit match RULE1
switch group 1 deny
switch group 1 start

switch group 2 add eth0 rf4.0
switch group 2 start
```

Порядок описания групп не имеет никакого значения, коммутатор выберет любую группу, которая подходит для передачи данного пакета. Будьте внимательны! Не создавайте групп больше, чем это действительно необходимо.

Оптимальная настройка коммутатора для передачи многоадресных потоков

Для решения часто возникающей задачи **передачи восходящих multicast потоков** в системах видеонаблюдения и регистрации в описатель группы введены два дополнительных параметра – **downstream** и **upstream**.



Предположим к узлам 1,2,3,4,5 и 6 подключены цифровые видеокамеры, которые транслируют изображение с помощью multicast пакетов и все эти потоки нужно передать на сервер наиболее оптимальным способом, не подвергая сеть ненужному “зафлуживанию” широковещательными пакетами.

Весь downstream (от сервера к камерам) трафик, если таковой имеется, передаётся в группе 1000, в которой находятся все узлы сети.

А вот upstream потоки от каждой камеры передаются непосредственно к ближайшему концентратору своей группы.

Особенностью такого решения является возможность установки нескольких концентраторов с одинаковым номером группы. Для устранения проблемы широковещательного шторма, который мог бы возникнуть из-за того, что концентраторы включены в разные порты одного проводного свитча, в MINT введено ограничение – транковые и downstream концентраторы никогда не используют друг друга для передачи трафика. Кроме того, наличие опции «upstream» гарантирует, что оконечные узлы будут выбирать для отправки пакетов только один концентратор, а именно **кратчайший путь к ближайшему концентратору**. Это позволяет устанавливать узлы и на подвижных объектах. Объединение downstream концентраторов с помощью псевдо-интерфейса prf (об этом ниже), позволяет поддерживать устойчивую связность всей сети.

И хотя решение этой задачи с использованием протокола IGMP было бы ещё эффективней, предлагаемый способ позволяет просто и быстро достичь требуемого результата.

Для того, чтобы более наглядно представлять себе какие группы имеются в сети и как они связаны друг с другом служит команда «**mint map swg**»:

```
================================================================================
Interface rf4.0

Node  001195F28E78  "NODE1", Id 1250, NetId 0, (master)(polling)
Freq  5320, Sid 10101010, autoBitrate 54000, Noise floor -96

GROUP 1 : sent 1199328 (dynamic) (in-trunk 1000)
  000435FFB7AD "Sklad CPE RF4.0   "  1 hops, Cost  68, (alive 4)

GROUP 2 : sent 2033943 (dynamic) (in-trunk 1000)
  00179AC2F4E2 "Office RF4.0      "  1 hops, Cost  51, (alive 5)

GROUP 5 : sent 18740 (dynamic) (in-trunk 1000)
  000E9B9AEB96 "Garage            "  1 hops, Cost  51, (alive 4)

GROUP 1000 : sent 347232 (trunk) (in-trunk 1000)
  000435FFB7AD "Sklad CPE RF4.0   "  1 hops, Cost  68, (alive 4)
  000E9B9AEB96 "Garage            "  1 hops, Cost  51, (alive 4)
  00179AC2F4E2 "Office RF4.0      "  1 hops, Cost  51, (alive 5)
================================================================================
```

Из этой таблицы видно, что данное устройство имеет 5 групп, одну транковую и 4 динамические входящие в область транковой зоны. Также показано, какие узлы в сети входят в состав каждой из групп.

Особенности обработки пакетов, адресованных узлу MINT-сети

Отдельно следует обратить внимание на пакеты которые адресованы непосредственно самому узлу (в том числе широковещательные пакеты, копия которых также передаётся на обработку в модуль маршрутизации собственного устройства). По умолчанию, такие пакеты передаются на верхний уровень растегированными (untagged), что позволяет модулю маршрутизации обрабатывать их независимо от установленного VLAN ID, так как будто они были приняты через интерфейс ethX. Если администратор хочет, чтобы пакеты передавались на верхний уровень с сохранённым или определённым значением VLAN ID, то нужно воспользоваться специальной командой:

```
switch self:N
```

В данном случае «self» играет роль псевдо-интерфейса, который направлен внутрь системы. Правила манипуляции VLAN тегами такие же, как и для реальных интерфейсов (кроме правила «по-умолчанию»):

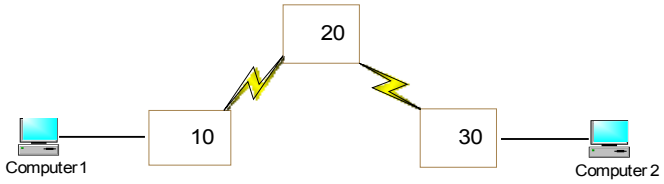
```
switch self:0   # сбрасывает любые теги (по умолчанию).
switch self     # не меняет значение VLAN тегов, если они были.
switch self:N   # принудительно устанавливает указанный VLAN тег.
```

Следует помнить, что пакеты передаваемые на верхний уровень (модуль маршрутизации) с установленными VLAN тегами, могут обрабатываться системой только при наличии в системе соответствующим образом настроенных интерфейсов vlanX. Например, если входящий пакет имеет установленный VLAN ID 100, то он сможет быть принят системой только через интерфейс vlanX, имеющем в настройках соответствующий параметр «VLAN»:

```
ifconfig vlanX VLAN 100 VLANdev eth0 up
```

Управление узлами MINT-сети

Заметьте, что для промежуточных узлов, (как узел 20 в примере ниже) не требуется настройка коммутатора, достаточно только настроить MINT. Такие узлы, теоретически даже могут не иметь вообще ни одного IP адреса.



Однако, логично будет предположить, что администратор захочет контролировать всю свою сеть. В этом случае IP-адреса назначить скорее всего придётся (впрочем, архитектура MINT позволяет контролировать все узлы MINT-сети по SNMP, даже если они не имеют IP-адреса).

Доступ к узлам MINT-сети

Итак, **как лучше организовать доступ внутрь MINT сети**? Есть два варианта: с помощью маршрутизации и путём коммутации. Какой удобнее – решать Вам.

Маршрутизируемый доступ

Поскольку все беспроводные интерфейсы MINT сети находятся в одном виртуальном сегменте Ethernet (одноранговая сеть), то можно просто пронумеровать их, назначив им адреса из одной IP подсети (это можно сделать вручную или по DHCP). Это уже хорошо, потому что попав на один из узлов сети, например по telnet, нам становятся доступны и все остальные узлы. Доступ извне тоже легко организовать, настроив маршрутизацию так, чтобы внутренняя MINT-сеть была доступна с компьютера администратора через Ethernet порт ближайшего узла, а изнутри сети, маршрут на компьютер администратора должен идти через радио-интерфейс граничного маршрутизатора.

Предположим, что все беспроводные интерфейсы MINT-сети имеют адреса в диапазоне 10.0.0.0/24. Пусть сеть администратора будет 192.168.0.0/24 и она подключена по Ethernet к узлу имеющему адрес радио-интерфейса 10.0.0.1. Назначим на Ethernet интерфейс этого узла адрес 192.168.0.1/24. Тогда на компьютере администратора (например 192.168.0.2) нужно будет настроить маршрутизацию на внутреннюю сеть:

```
route add 10.0.0.0 netmask 255.255.255.0 192.168.0.1
```

А на **всех** узлах сети обеспечить доступность сети администратора:

```
route add 192.168.0.0/24 10.0.0.1
```

Коммутируемый доступ

При коммутируемом доступе мы можем назначать беспроводным интерфейсам адреса **из той же подсети**, в которой находится сеть администратора (192.168.0.0/24) и никаких маршрутов прописывать не придётся. Но! В этом случае нам потребуется на всех узлах сети настроить коммутатор так, чтобы на нём была хотя бы одна группа в которую входит беспроводной интерфейс (на который назначен IP адрес), даже если он единственный в группе:

```
sw g 1 add rf4.0
sw g 1 start
sw start
```

Это нужно для того, чтобы коммутатор имел возможность обучаться и мог правильно выбирать направление передачи ответных пакетов. Номер группы здесь значения не имеет, можно использовать любую другую уже имеющуюся группу если в неё входит интересующий нас беспроводной интерфейс.

А вот на граничном узле, к которому подключена сеть администратора нам обязательно потребуется группа в которую входят оба интерфейса и проводной и беспроводной. И номер этой группы должен быть указан в специальном параметре **switch admin-group**:

```
sw g 1 add eth0 rf4.0
sw g 1 start
sw admin-group 1
sw start
```

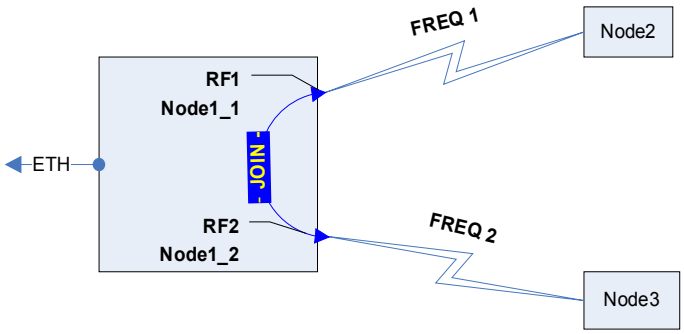
Как мы помним, путём коммутации пакеты изнутри сети могут выходить наружу только через admin группу. Их, кстати, может быть несколько, на разных узлах. Помните, что **через admin-group в ваш проводной сегмент будут попадать все**

широковещательные пакеты порождаемые внутренними устройствами MINT сети.

Объединение интерфейсов

Ещё одним интересным свойством архитектуры MINT является способность объединять (**join**) в единую mesh сеть несколько различных интерфейсов одного устройства.

Например, некоторые устройства могут иметь два или более радио-интерфейсов различных типов. Каждый такой интерфейс сам по себе может являться узлом самостоятельной MINT сети. Однако узлы из разных сетей никогда не смогут установить связь друг с другом из-за различий в параметрах используемых радио-интерфейсов (частоты, типы модуляций, различные стандарты) и административных ограничений (параметры аутентификации, ключи доступа и т.д.). Функция **JOIN** позволяет двум (или более) интерфейсам одного устройства установить связь друг с другом, так, как будто они являются двумя узлами одной сети. Никакие различия в параметрах настройки этих интерфейсов и протоколов не являются препятствием для объединения.




```
mint join rf4.0 rf4.1

mint map

Interface rf4.0
Node  000000000011  "Node1_1", Id 10, NetId 0, (mesh)
Freq  5320, Sid 10101010, autoBitrate 54000, Noise floor -95

2 active neighbors:
 --------------------
  Id      Name             Node      Cost   Sig     Bitr   R  E Options
----- ---------------  ------------ ----  -----  ------- -- -- -------------
00020 Node_2           000000000002   40  15/18   36/36   0  0 /mesh/
00012 Node_1_2         000000000012    3   0/0     0/0    0  0 /join/
----- ---------------  ------------ ----  -----  ------- -- -- -------------

Interface rf4.1
Node  000000000012  "Node1_2" Id 12, NetId 0, (mesh)
Freq  6320, Sid 10101010, autoBitrate 54000, Noise floor -110
```

```
2 active neighbors:
 ------------------
  Id      Name             Node      Cost   Sig     Bitr   R  E Options
----- ---------------  ------------ ----  -----  ------- -- -- -------------
00030 Node_3           000000000003   40  15/18   36/36   0  0 /mesh/
00011 Node_1_1         000000000011    3   0/0     0/0    0  0 /join/
----- ---------------  ------------ ----  -----  ------- -- -- -------------
```

Как видно из примера, каждый интерфейс считает, что к нему подключено два соседних узла. Поскольку реальный обмен информацией между объединёнными узлами не затрагивает физических интерфейсов, то энергетические параметры соединения (амплитуды сигналов и скорости передачи) не показываются (равны нулю). И такое соединение имеет постоянную, очень низкую стоимость.

Последнюю иллюстрацию можно представить в несколько ином виде.

[ретранслятор]

Таким образом становится очевиднее еще одно очень полезное свойство функции JOIN для устройств с двумя радиоинтерфейсами, а именно, перевод их в режим повторителя (repeater) при построении каналов «точка-точка» или организации магистральных соединений между удаленными сегментами MINT-сети, когда одного пролета оказывается недостаточно.

В отличие от объединения нескольких сегментов сети с помощью коммутатора, JOIN дает возможность создания действительно единой сети, в которой будут работать все механизмы MINT, включая оптимизацию маршрутов и контроль топологии.

Важное замечание: если несколько интерфейсов объединены с помощью функции **join**, то при включении их в группу коммутации следует указывать **только один из них** (любой).

Правильно:

```
mint join rf4.0 rf4.1
switch group 1 add eth0 rf4.0
```

НЕ правильно:

```
mint join rf4.0 rf4.1
switch group 1 add eth0 rf4.0 rf4.1
```

Коммутатор работает только с независимыми интерфейсами. При объединении нескольких интерфейсов посредством **join**, любой из них будет представлять всю группу. Остальную работу возьмёт на себя MINT.

Но и это еще не все...

«Псевдо» радиоинтерфейсы

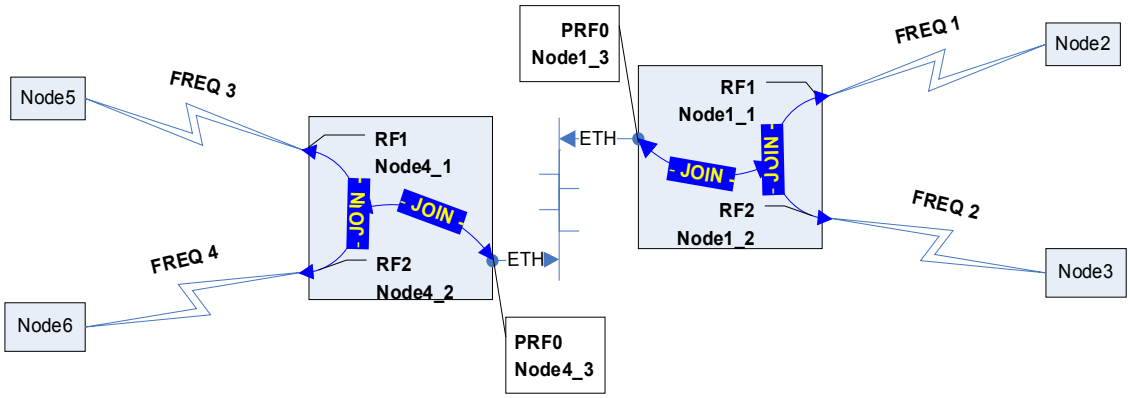
Протоколы архитектуры MINT могут работать не только по радио, но и через проводной интерфейс Ethernet. Для этого в системе имеется «псевдо» радио-интерфейс (**prf**), который можно ассоциировать с любым проводным интерфейсом подобно тому, как это сделано для интерфейсов vlanX.

```
prf 0 parent eth0
```

```
ifconfig prf0 up
```

Такой псевдо радио-интерфейс можно использовать для настройки на нём узла MINT сети и даже для объединения с другими интерфейсами. С точки зрения протоколов MINT, это будет обычный радио-интерфейс, через который узел сможет найти соседей и установить с ними связь.

```
mint prf0 start
mint join rf4.0 rf4.1 prf0
```



В этом примере нам удалось объединить в единую MINT сеть несколько обособленных, и возможно, территориально удалённых сегментов сети. Разумно комбинируя перемычки **join** и псевдо радио-интерфейсы можно обеспечить наличие в сети достаточного количества надёжных альтернативных путей для обеспечения оптимальности распространения трафика и устранения узких мест.

В качестве «родительского» интерфейса для prf можно использовать и псевдоинтерфейсы других типов – VLAN (vlanX) и туннели (tunX). Если prf запущен над туннельным интерфейсом, последний автоматически переключается в режим Ethernet-over-IP.

Управление топологией

Посмотрим, как мы можем управлять топологией и направлением потоков трафика в сети, для получения оптимальной для нас конфигурации.

Типы узлов в MINT-сети

Прежде всего, каждый узел сети MINT может быть одного из трёх типов: **master, mesh или slave**.

Узлы типа Master

Может устанавливать соединение со всеми типами устройств. Друг с другом и с устройствами типа **mesh** может образовывать сеть любой топологии.

На устройстве типа **master** может быть включен маркерный доступ. Только один master в сегменте сети может быть активным по отношению к маркерному доступу, образуя при этом сеть с топологией точка-многоточка (звезда). Все остальные узлы при этом разрывают свои соединения с другими устройствами (кроме связей установленных с помощью функции join).

Тип «master» обычно используется на ключевых устройствах сети, положение которых в пространстве относительно статично и которые используются в качестве опорных узлов сети передачи данных.

Узлы типа Mesh

Устройство может быть участником сети с произвольной топологией. Устанавливает связи с устройствами типа **mesh** или **master**. Отличие **mesh** от **master** в том, что узлы типа **master** будут стараться избегать передачи трафика опорной сети (master-master) маршрутами которые проходят через узлы **mesh** (если есть другой путь через опорную сеть), устанавливая стоимость соединения master-mesh (со стороны master) заведомо выше (параметр ***meshextracost***), чем с другими устройствами. Таким образом, тип **mesh** можно использовать на мобильных устройствах с неустойчивыми или часто меняющимися условиями связи, не опасаясь, что это нарушит работу опорной сети.

Узлы типа **mesh** могут работать в режиме маркерного доступа под управлением **master**-а. При этом, если **master** включает маркерный доступ, то **mesh** узел разрывает связи со всеми остальными узлами (кроме тех, что установлены посредством функции join). При исчезновении **master**-а (или отключении на нём маркерного доступа) узел **mesh** восстанавливает соединения с остальными соседями (если они были).

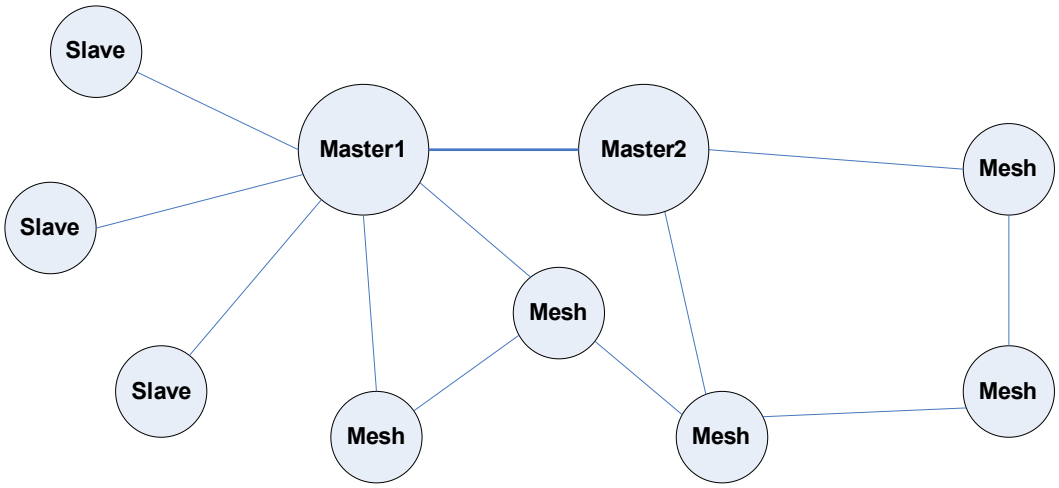
Узлы типа Slave

Устанавливает единственное соединение с устройством типа **master**. При потере соединения выполняет сканирование сети в поисках нового или утраченного **master**-а. Под управлением **master**-а устройство типа «**slave**» может работать в режиме маркерного доступа. Режим **slave** используется для построения

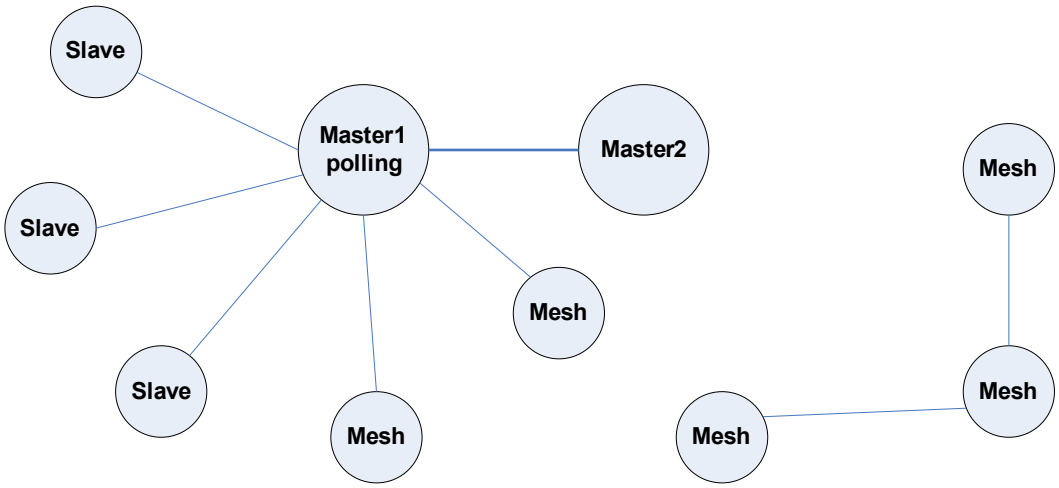
«классической» топологии «звезда» («точка-многоточка» или «базовая станция – клиенты» или «master-slave»).

Особенности применения поллинга

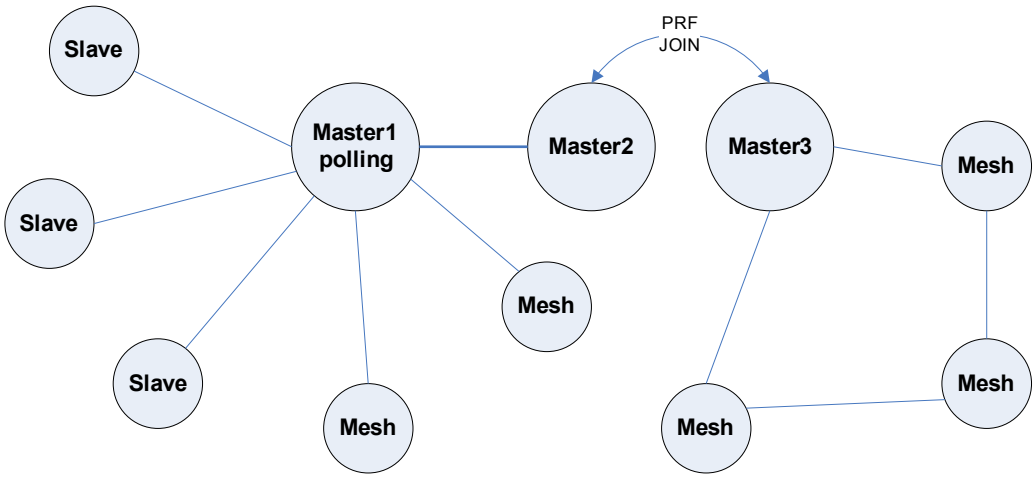
Сеть с использованием всех трёх типов узлов может выглядеть так:



Если теперь на устройстве Master1 включить маркерный доступ, то сеть распадётся на 2 несвязанных сегмента:



Очевидно, это не то, что мы хотели получить. Исправить ситуацию можно подключив «отпавший» сегмент с помощью перемычки **join**, используя вместо Master2 двухмодульное устройство или установив дополнительный узел, соединив его по Ethernet с Master2 и связав их в единый сегмент с помощью интерфейса **prf**.



Управление топологией через стоимости соединений

Кроме указания типа устройства мы можем вручную корректировать значения стоимости соединений на интерфейсе или на каждом конкретном направлении. Существует четыре поправочных коэффициента:

extracost – устанавливается на интерфейс. Задаёт добавочную стоимость для всех соединений на этом интерфейсе. Это значение прибавляется к стоимости соединения вычисленному автоматически протоколом MINT, либо установленному любым другим способом. Может быть только положительным. Значение 0 (ноль) отменяет действие этого параметра.

```
mint rf4.0 -extracost 60
```

meshextracost – устанавливается на интерфейс. Задаёт добавочную стоимость для всех соединений узла типа master с узлами типа mesh. По умолчанию – 500.

```
mint rf4.0 -meshextracost 300
```

joincost – устанавливается на интерфейс. Задаёт стоимость всех соединений на этом интерфейсе, полученных с помощью функции join (по умолчанию 3). Значение 0 (ноль) отменяет действие этого параметра.

```
mint rf4.0 -joincost 60
```

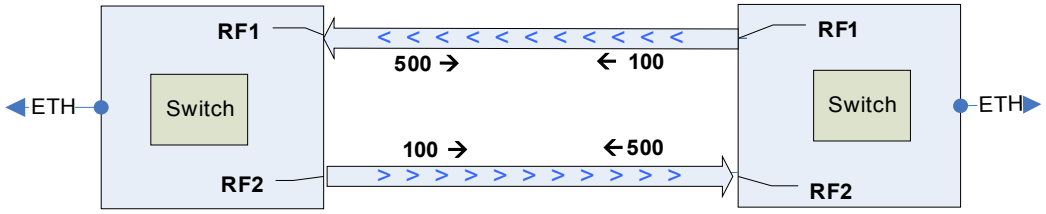
fixedcost – устанавливается на интерфейс. Присваивает все соединениям на этом интерфейсе (кроме join) фиксированное значение стоимости. Значение 0 (ноль) отменяет действие этого параметра.

```
mint rf4.0 -fixedcost 500
```

При совпадении нескольких условий, поправочные коэффициенты назначаются в том порядке, в каком они перечислены выше.

Организация полнодуплексных каналов связи

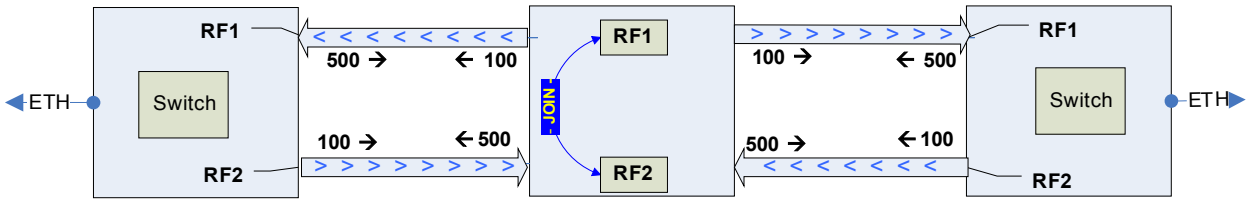
Поправочные коэффициенты можно использовать для решения специфических задач, создавая конфигурации с заранее определёнными или предпочтительными направлениями потоков. Например, в следующем примере, расставив соответствующим образом стоимости соединений (fixedcost), мы создали схему имитирующую полнодуплексную (**full duplex**) линию связи, в которой противоположные потоки идут по разным каналам, что позволяет удвоить суммарную емкость такой линии.



Так, поток слева - направо будет использовать нижний канал, поскольку его стоимость (100) меньше, чем у верхнего канала (500). В обратную сторону наоборот. Коммутатор (switch) знает о существовании параметра стоимости соединений и учитывает его при выборе направления.

Реализовать такую схему можно двумя способами, либо включив в группу коммутации все три интерфейса (`sw g 1 add eth0 rf4.0 rf4.1`) , либо объединив радиоинтерфейсы функцией join и коммутировать только один интерфейс (`mint join rf4.0 rf4.1; sw g 1 add eth rf4.0`). Оба этих варианта практически одинаковы.

Развив эту идею, можно построить, например, длинную линию с ретранслятором, не имеющую характерной для ретрансляторов проблемы потери половины производительности на каждом промежуточном узле.

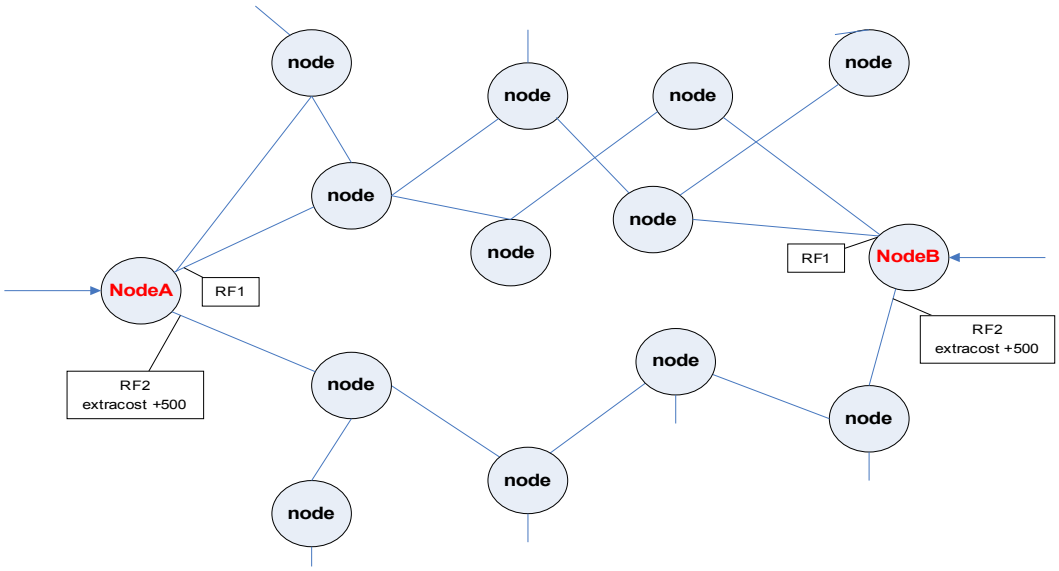


Здесь мы применили внутреннюю перемычку join для изменения направления потоков на ретрансляторе таким образом, что для приёма и передачи пакетов всегда используются разные интерфейсы.

Как видно из примера, верхний интерфейс ретранслятора всегда используется только для передачи, поэтому он, очевидно, не имеет проблем с коллизиями и потерей производительности. Однако нижний интерфейс только принимает пакеты, причём с разных сторон, поэтому здесь, вероятно, и будет центр коллизий, вызванных проблемой скрытого узла, поскольку крайние источники не слышат друг друга и не могут координировать свои действия. Но мы знаем хороший способ борьбы со скрытыми узлами – маркерный доступ. Присвоить ретранслятору тип master (на нижнем интерфейсе), включить маркерный доступ, и проблема будет решена.

Приоритезация заданных направлений

Другой пример использования поправочных коэффициентов – выделение предпочтительных направлений распространения потоков:



Например, мы хотим чтобы потоки между узлами NodeA и NodeB в основном использовали верхний сегмент сети, а нижний был зарезервирован для передачи потоков внутри и внутрь этого сегмента. Как это сделать показано на схеме. Конечно, величину заградительной поправки придётся подбирать эмпирически, исходя из реально сложившихся условий. На практике, такие сегменты сети могут быть разделены не территориально, а, например, по разных частотным диапазонам, образуя несколько слоёв для предоставления услуг разного качества. Однако, в случае необходимости, например при критической загрузке или отказе одного из сегментов (или его ключевых элементов), другой сегмент сможет автоматически взять на себя всю или часть нагрузки.

Роуминг в MINT

Частотный роуминг

Для облегчения процесса миграции между несколькими независимыми сегментами сети архитектура MINT поддерживает режим частотного роуминга (**roaming**). По-умолчанию, данная функция неактивна и устройство работает с фиксированными параметрами радиоинтерфейса, заданными в конфигурации.

Любой узел сети (один или несколько) можно назначить опорным. Это значит, что данный узел будет задавать требуемые радиочастотные параметры данной сети. В терминологии MINT такой узел называется **roaming leader**.

Roaming leader также работает с фиксированными параметрами радиоинтерфейса, однако информация о его существовании распространяется по сети, так что любой узел может определить, подключен ли он к опорному узлу или к сети, в которой такой узел уже имеется. Если в одном сегменте сети имеется несколько опорных узлов, то их параметры должны быть идентичными.

Roaming leader, кроме того, поддерживает функции DFS и Radar Detection (если установлена соответствующая лицензия).

Остальные узлы сети могут использовать данный механизм для поиска подходящего опорного узла или сети, в которой такой узел уже имеется. Непосредственно поиск выполняется путём перебора радиочастотных и административных параметров, определяемых с помощью системы профилей. Каждый профиль определяет некий фиксированный набор параметров радиоинтерфейса и сети, которые будут устанавливаться в системе перед каждым очередным этапом поиска.

Эвристический алгоритм поиска быстро оценивает общую остановку в эфире и, сосредоточившись на ключевых параметрах профилей, выбирает из числа обнаруженных сетей наиболее подходящую. При этом целью поиска является не сам опорный узел, а именно сеть в которой он имеется. (Достаточно найти любой узел который уже подключен к сети где есть опорный узел).

При изменении параметров опорного узла, либо при потере всех связей с существующей сетью, поиск будет выполнен вновь.

Основные параметры профилей:

- freq XXX – частота радиоинтерфейса. Может быть указано ключевое слово auto, в этом случае для поиска будут использованы все частоты поддерживаемые данным радиомодулем, с учётом имеющейся лицензии.
- sid XXX – сетевой идентификатор для работы в конкретной сети.
- bitr XXX – битовая скорость радиоинтерфейса. Играет роль верхнего ограничения скорости при включенном режиме autobitrate
- bandwith {full|half|quarter} – ширина полосы пропускания радиомодуля.
- key XXX – ключ доступа к сети

Пример конфигурации:

```
mint rf4.0 profile 1 -freq 5920,5960 -sid ABCDE –key mykey
mint rf4.0 profile 2 -freq auto -sid DEAD –key secret
mint rf4.0 roaming enable
```

IP-роуминг

Для провайдера услуг важно, в конечном итоге, обеспечить не частотный, а IP-роуминг, т.е. возможность предоставления услуг независимо от места подключения клиента. Эту задачу лучше решать на более высоком уровне, например, с помощью IP-маршрутизации. Для достижения большей гибкости при решении этой задачи технология MINT предлагает несколько уникальных возможностей.

Многие компоненты системы интегрированы в общую архитектуру MINT и умеют пользоваться её преимуществами. Рассмотрим это на примере простейшего сценария:

устройство с включенной опцией «roaming enable» сканирует эфир и подключается к выбранной сети, став полноценным MINT-узлом.

DHCP-клиент, получив информацию о подключении к сети, немедленно активизируется и начинает процедуру поиска сервера и получения конфигурации. DHCP-сервер может находиться как на одном из узлов MINT сети (или нескольких узлах), так и вне её (в этом случае пакеты с запросами от узла и ответами сервера должны быть скоммутированы через admin-group).

После установки на радиоинтерфейс IP-адреса в дело вступает OSPF. Используя свойство «**autointerface**», модуль OSPF обнаруживает появление нового адреса, немедленно создаёт логический интерфейс и начинает поиск соседей, обеспечивая тем самым интеграцию в существующую Internet (IP) инфраструктуру.

Для повышения эффективности использования сети, специально для технологии MINT был введён **новый тип OSPF интерфейса** – «**mesh**», который позволяет представить все связи узла со своими OSPF соседями как многоточечную (point-to-multipoint) сеть, предусмотренную протоколом OSPF. Но в отличие от стандартной многоточечной сети, для **mesh** интерфейсов не требуется вручную корректировать стоимость соединений между соседними маршрутизаторами (с помощью команды neighbor cost). Соседние маршрутизаторы будут обнаружены автоматически штатным алгоритмом поиска соседей OSPF, а стоимость соединения с каждым соседом будет динамически регулироваться на основании информации получаемой им от сетевого уровня MINT. Теперь OSPF стоимости соединений всегда оптимальны!

OSPF-интерфейс типа «mesh» может быть использован для устранения проблемы «бутылочного горлышка» (bottleneck), когда трафик всей сети выходит наружу через один единственный канал. Теперь в сети может быть несколько default маршрутов (впрочем, как и всех остальных), при этом каждый узел (OSPF router) сам выберет для себя наиболее подходящий шлюз(ы) и будет поддерживать его оптимальность.

Оба этих свойства (autointerface и mesh) позволяют эффективно использовать OSPF маршрутизацию на мобильных устройствах и в условиях роуминга.

Сложно ли воплотить такой сценарий в конфигурации конкретного устройства? Вот пример такой конфигурации (показаны только ключевые параметры):

```
ifconfig eth0 1.2.3.4/24 up   # Client LAN network

#MINT configuration
   mint rf4.0 -type mesh
   mint rf4.0 –name Client-NNN
   mint rf4.0 –nodeid NNN
   mint rf4.0 prof 1 -freq auto -sid 10101010 -bitr 36000 –key mykey
   mint rf4.0 roaming enable
   mint rf4.0 start
```

```
#DHCP configuration
   dhcpc -k test
   dhcpc rf4.0 start

#OSPF configuration
   ospf start
   ospf interface eth0
   ospf interface rf4.0
   ospf  network mesh
   ospf router
   ospf  redistribute-connected
   ospf  auto-interface rf4.0 area 0.0.0.1
   ospf end
```

Продолжение следует…

Fixed & Autobitrate + loamp + hiamp

MINT Monitor & Antenna Alignment

Authentication + Authorization

Load Balancing and Link Redundancy

Over the Air Upgrade

Сценарии использования

Управление по SNMP