Делим Интернет или QoS на Mikrotik

Автор: Кузьмицкий Алесандр

09/19/06 02 2006 ?., 21:02

Принцип делить все и вся существует столько, сколько миллиардов лет существует Вселенная. Чаще всего объекты делятся в одинаковых пропорциях и неравный отхваченный кусок кем или чем-либо вызывает недовольство поделенных. Подобные ситуации часто возникают тогда, когда объект деления имеет весомое значение в системе. Перенесемся в 21 век. Наблюдая процессы в обществе и, в первую очередь в информационной сфере, можно заметить, что разумно делить все-таки научились. И, порой алгоритмы настолько неподдельны и устойчивы, что не оставляют никакой возможности взять больше или меньше дозволенного. Применив все вышеизложенное к передаче данных можно понять о чем идет речь.

Современное общество и каждый человек в нем требует к себе одинакового отношения во всех сферах, что зачастую становится головной болью для тех, кто предоставляет услуги. В частности касаясь предоставления услуг доступа к Интернету за последние несколько лет произошел огромный прорыв, позволивший решить ряд проблем, порожденных появлением всемирной паутины у нас в домах и на работе, превратив ее из некого деликатеса в фаст-фуд для сотен тысяч людей.

Несколько раньше на нашем сайте мы опубликовали статью о RouterOS Mikrotik, в которой говорилось о возможностях, установке и первоначальной настройке этой операционной системы. Судя по количеству вопросов и ответов на нашем форуме стало ясно, что в Интернет тема настройки и управления системой раскрыта плохо и не полно.

Не секрет, что зачастую администраторам, в особенности начинающим, сложно разобраться во всех приведенных алгоритмах и принципах работы шейпера, вынуждая учиться методом проб и зачастую не безобидных ошибок. Так сложилось, что для этой категории пользователей очень мало простой и доступной русскоязычной документации, одним из первых начинаний которой и станет эта статья.

Теория

Для начала рассмотрим несколько понятий, которыми мы будем пользоваться в дальнейшем.

Технология, которая позволяет ограничивать скорость и качество доступа в Интернет, называется **шейпинг** (от англ. Shape - форма). Образно говоря - это технология придания некой формы графику загрузки канала.

Шейпер - это алгоритм, который помимо управления очередностью пакетов позволяет отбрасывать не удовлетворяющие условиям. К таковым относятся алгоритмы PCQ и HTB (о них поговорим несколько позже).

Существует ещё один тип алгоритмов, используемых для управления движением пакетов внутри шейпера **Schedulers**. Их задача состоит только в формировании очередей согласно приоритетам пакетов, адресу источника, получателя и другим параметрам. К этому типу алгоритмов относятся PFIFO, BFIFO,SFQ, PCQ, RED.

Под-очередь - очередь, сформированная из пакетов по тому или иному признаку.

Queuing discipline (qdisc - дисциплина очереди) - алгоритм, который захватывает пакеты и точно определяет в каком порядке и каким образом они будут двигаться.

HTB

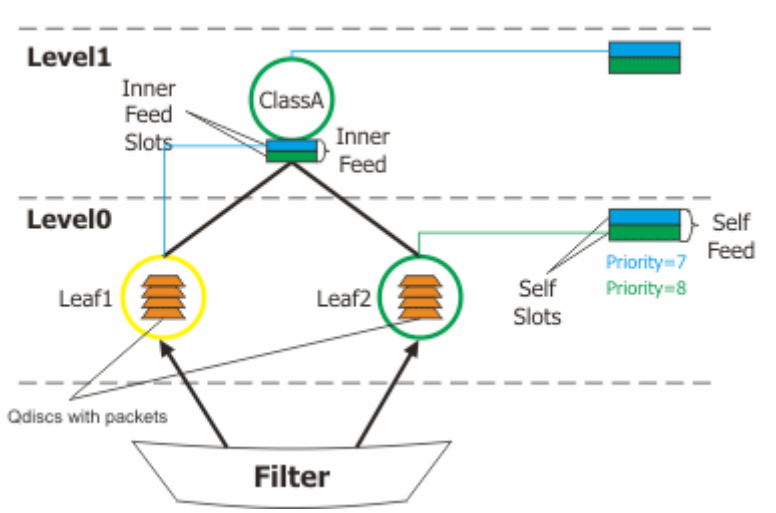
В основе шейпинга, используемого в Mikrotik, лежит дисциплина очереди HTB, реализованная во многих Linux-системах. Ее изучение является достаточно сложной, однако необходимой задачей для новичка, потому как без этих знаний дальше неудачных попыток и копирования правил из документации мало кто заходит.

Список основных возможностей по управлению трафиком в Mikrotik выглядит следующим образом:

- ограничение скорости по IP-адресам, подсетям, протоколам, портам, времени суток и другим параметрам;
- ограничение P2P-трафика (BitTorrent, eMule);приоритизация одних потоков пакетов над другими;
- использование пиковых скоростей для быстрого WEB-браузинга;
- разделение канала между пользователями поровну или в других пропорциях;
- возможность задания гарантированной скорости.

Ключевым понятием для HTB является класс. Приставка Hierarhical в аббревиатуре HTB означает, что дисциплина позволяет строить иерархию классов.

Схематически иерархию классов (для упрощения будем называть классы правилами) HTB можно представить в виде некого гибридного разделенного уровнями дерева, конечными вершинами которого являются клиенты. Классы, которые не имеют дочерних, будем называть клиентами или листьями. Обычно они находятся на нулевом уровне иерархии и первыми захватывают относящийся к ним трафик, передавая его родителям. Два или более класса, имеющие одного прямого родителя находятся на одном уровне и подсоединены к одной локальной выходной очереди.



Схематическое изображение структуры HTB

На схеме выше изображена иерархия классов, в которую из файервола (Filter) поступают пакеты с данными. В зависимости от приоритета, параметров классов и загрузки канала они попадают или в локальные очереди (Self Feed), или передаются в очереди родительских классов (Inner Feed).

Класс характеризуют следующие параметры:

- **limit-at** – гарантированная скорость;
- **max-limit** – ограничение скорости;
- **priority** – приоритет класса.

Класс может находиться в одном из трех состояний:

- **Зеленый** - пропускная способность правила не превышает параметр limit-at. В этом случае пакеты не двигаются вверх по иерархии, а перемещаются сразу в выходной поток своего уровня

согласно приоритетам.

- **Желтый** - пропускная способность правила больше limit-at, но меньше max-limit. В этом случае класс отключается от выходного потока своего уровня и подключается к родительскому классу.
- **Красный** - пропускная способность правила больше max-limit. В этом состоянии класс отключается от родительского и подключается к локальной очереди.

Пользуясь уже даже этими данными, можно составлять правила, однако на практике некоторые вещи могут выглядеть несколько иначе.

В Mikrotik предусмотрены два типа правил, разнесенные на разные закладки в графической утилите Winbox (с ее помощью можно конфигурировать Mikrotik из-под Windows):

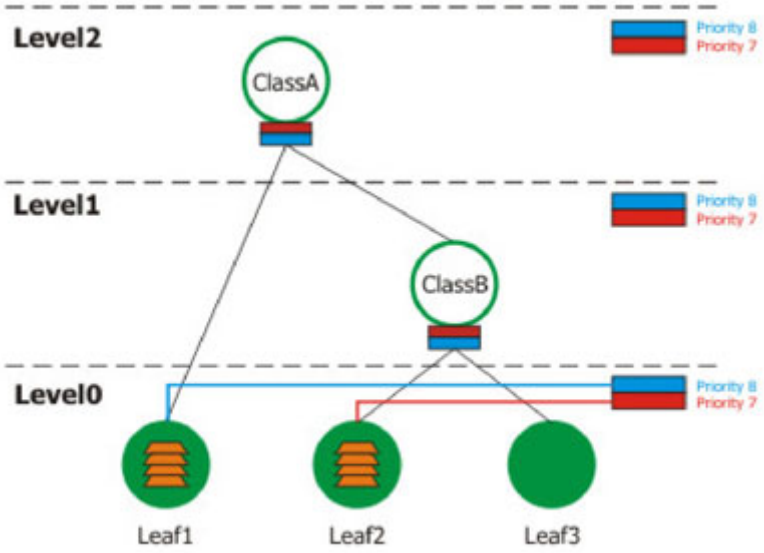
- **Simple Queues**;
- **Queue Trees**.

О них мы поговорим несколько позже, а сейчас рассмотрим несколько примеров работы HTB

Создадим несколько правил

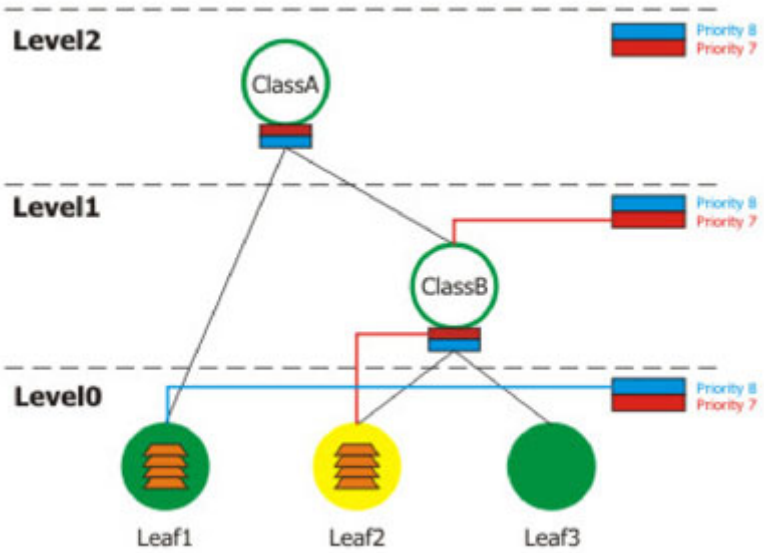
```
[admin@MikroTik] queue tree> add name=ClassA parent=Local max-limit=2048000
[admin@MikroTik] queue tree> add name=ClassB parent=ClassA max-limit=1024000
[admin@MikroTik] queue tree> add name=Leaf1 parent=ClassA max-limit=2048000 \
\... limit-at=1024000 packet-mark=packet_mark1 priority=8
[admin@MikroTik] queue tree> add name=Leaf2 parent=ClassB max-limit=1024000 \
\... limit-at=256000 packet-mark=packet_mark2 priority=7
[admin@MikroTik] queue tree> add name=Leaf3 parent=ClassB max-limit=1024000 \
\... limit-at=768000 packet-mark=packet_mark3 priority=8
[admin@MikroTik] queue tree> print
Flags: X - disabled, I - invalid
 0   name="ClassA" parent=Local packet-mark="" limit-at=0 queue=default
     priority=8 max-limit=2048000 burst-limit=0 burst-threshold=0
     burst-time=0s

 1   name="ClassB" parent=ClassA packet-mark="" limit-at=0 queue=default
     priority=8 max-limit=1024000 burst-limit=0 burst-threshold=0
     burst-time=0s

 2   name="Leaf1" parent=ClassA packet-mark=packet_mark1 limit-at=1024000
     queue=default priority=8 max-limit=2048000 burst-limit=0
     burst-threshold=0 burst-time=0s

 3   name="Leaf2" parent=ClassB packet-mark=packet_mark2 limit-at=256000
     queue=default priority=7 max-limit=1024000 burst-limit=0
     burst-threshold=0 burst-time=0s

 4   name="Leaf3" parent=ClassB packet-mark=packet_mark3 limit-at=768000
     queue=default priority=8 max-limit=1024000 burst-limit=0
     burst-threshold=0 burst-time=0s
[admin@MikroTik] queue tree>
```

1. Рассмотрим первый случай, когда клиенты 1 и 2 передают данные со скоростью меньше, чем указано в параметре limit-at, а клиент 3 не работает.



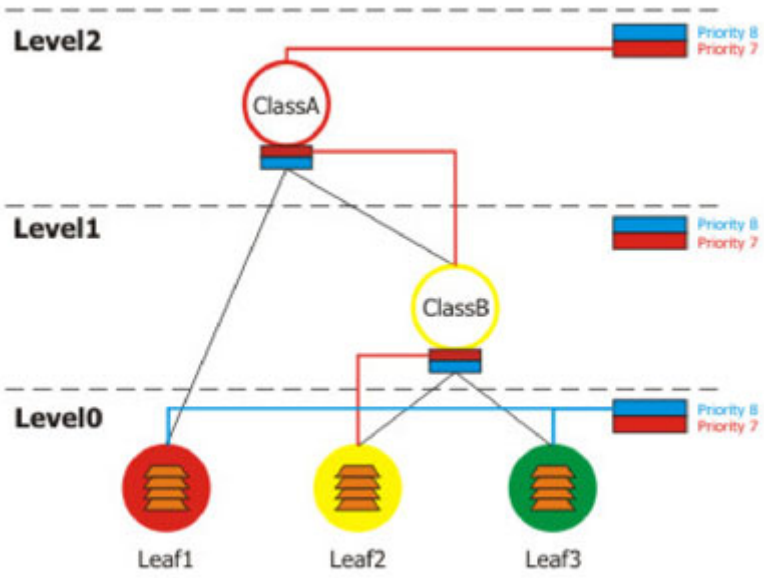
Как видим, пакеты с данными от leaf1 и leaf2 (клиенты) не передаются в родительские классы, а выстраиваются в локальную очередь в соответствии со своими приоритетами

2. Сейчас посмотрим что будет, в случае если клиент leaf2 будет передавать данные со скоростью больше limit-at, но меньше max-limit указанных в его параметрах и меньше limit-at в параметрах ClassB, к которому он прикреплен. Одновременно с ним leaf1 будет передавать данные со скоростью не превышающей limit-at.



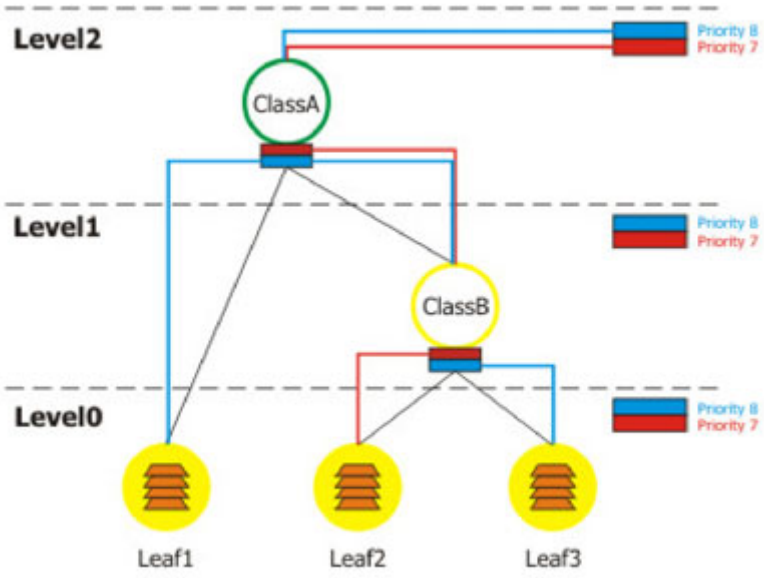
В данном случае получается интересная ситуация: клиент leaf1 будет иметь больший приоритет, чем leaf2, хотя в параметрах последнего указан больший приоритет. Это связано с тем, что передавая данные со скоростью более limit-at leaf2 подключился к родительскому классу, имеющему приоритет 8. При этом существует правило, что на нижних уровнях приоритет пакетов при одинаковых условиях больше, чем на верхних.

3. Рассмотрим следующий пример: скорости передачи данных для leaf1 превысила допустимое max-limit, клиент leaf2 передает данные на скорости больше limit-at и меньше max-limit, клиент leaf3 работает на скорости меньше limit-at.



Это весьма интересный случай. В данной ситуации видно, что ClassA перегружен данными из Leaf1, поэтому ClassB не получит разрешения на передачу.В результате работоспособным окажется только клиент leaf3, подключенный в локальную очередь на нулевом уровне.

4. Теперь рассмотрим пример, когда данные будут одновременно передавать leaf1, leaf2, leaf3, ClassB будет желтым, а ClassA зеленым.



В результате этого на втором уровне leaf2 попадет в очередь первым (так как имеет больший приоритет), а leaf1 и leaf3 подвергнутся случайному выбору для определения порядка следования.

Как видим алгоритм работы HTB весьма логичен и не так уж сложен, как могло показаться сразу. Он был принят многими производителями программного и аппаратного обеспечения за свою гибкость, надежность и отсутствие свойственных его предшественникам недостатков. Благодаря огромной универсальности с помощью него можно строить практически любые возможные иерархии правил, в точности разграничивая и давая возможность управлять потоками данных на достаточно низком уровне.

Bursts

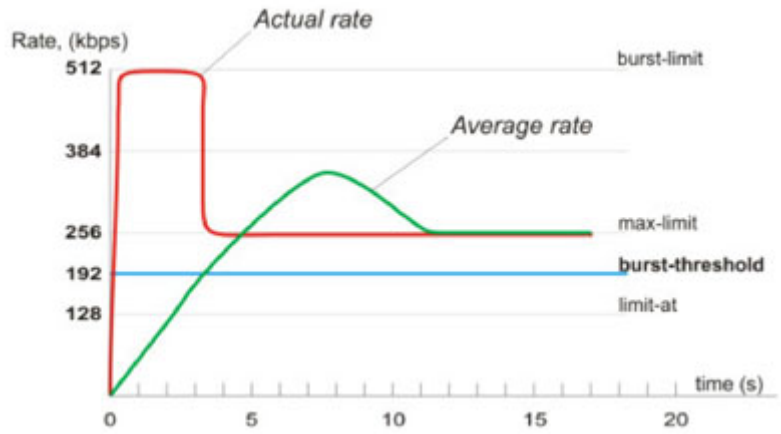
Часто возникает необходимость выдать так называемую пиковую скорость клиенту на определенный промежуток времени. К примеру, иногда требуется ускорить загрузку страниц, оставив при этом среднюю скорость на закачку файлов не выше отведенной. Или нужно редко посылать запросы и принимать данные, но делать это с максимальной скоростью. Глупо было бы в таком случае выдавать клиенту большую скорость, так как в один прекрасный момент из-за неосторожности или злого умысла остальные пользователи могут остаться или вовсе без Интернета, или получать его не таким, каких хотелось и за сколько заплачено.

Разработчики Mikrotik предоставили в распоряжение все необходимые инструменты для управления описанной выше пиковой скоростью. Следующие параметры характеризуют ее поведение:

- **burst-limit** - скорость, которая будет доступна сразу при подключении;
- **burst-threshold** – средняя скорость за последние **burst-time** секунд;
- **burst-time** - время для подсчета **burst-threshold**.

Момент, когда клиенту или классу нужно выдать максимальную скорость, определяется следующим образом. Раз в 1/16 времени burst-time вычисляется загрузка канала на указанное число секунд. Если средняя загрузка составила менее burst-threshold, то клиенту или классу выделяется указанная в burst-limit скорость до тех пор, пока она не превысит burst-threshold. После этого действует ограничение max-limit до тех пор, пока снова не случится понижение скорости менее burst-threshold.

Установим следующие параметры **limit-at**=128000/128000, **max-limit**=256000/256000, **burst-limit**=512000/512000, **burst-treshold**=192000/192000, **burst-time**=8, и понаблюдаем что случится с графиком загрузки канала от одного клиента:



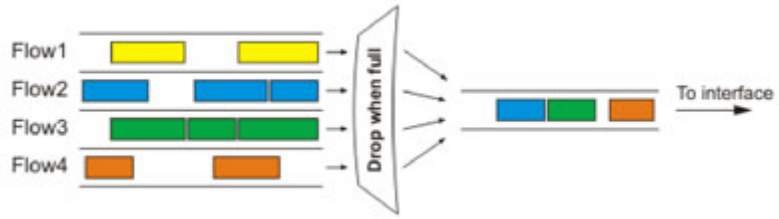
Данный график характерен для случая с закачкой большого файла по протоколу http. После первой секунды средняя загрузка канала будет равна (0+0+0+0+0+0+0+512)/8=64 kbps, что менее установленного нами параметра **burst-threshold**. После второй секунды средняя скорость будет равна (0+0+0+0+0+0+512+512)/8=128kbps. После третьей секунды средняя скорость превысит показатель **burst-threshold**. В этот момент скорость резко упадет до значения параметра **max-limit** и будет держаться на этом уровне до тех пор, пока средняя загрузка канала не станет меньше **burst-threshold** и снова не произойдет выдача burst скорости.

Schedulers

Рассмотрим алгоритмы так называемых Schedulers, о которых упоминалось выше. Обычно они применяются вкупе с шейперами, однако некоторые из них так же обладают функциями ограничения скорости. Физически Scheduler предшествует шейперу и представляет ему уже подготовленные очереди пакетов, к которым следует применять ограниения.

PFIFO/BFIFO

Packet/Bytes (FIFO) алгоритм, основанный на принципе первый пришел-первый ушел. Используется для ethernet-интерфейсов. Единственный параметр, используемый для конфигурирования данного алгоритма, - это **pfifo-limit** (bfifo-limit). Он указывает на количество байт, которые могут храниться в выходном буфере. Не попавшие в буфер пакеты будут разрушаться. Графически алгоритм можно изобразить с помощью следующей схемы. По сути дела PFIFO/BFIFO ничего особенного из себя не представляет и никаких приемуществ не дает. Он просто есть и используется там, где его использовать целесообразно…

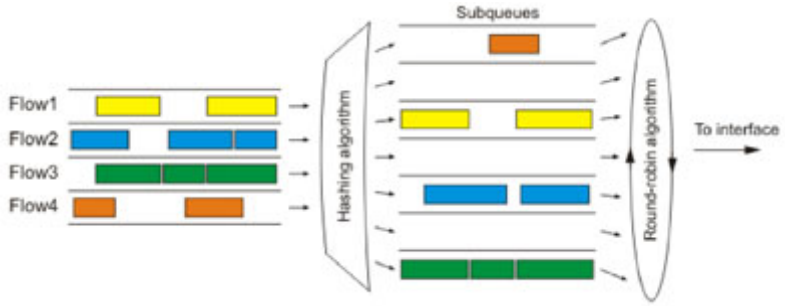


SFQ

SFQ (Stochastic Fairness Queuing) – этот алгоритм можно назвать "случайно-честным". Он применятся тогда, когда требуется предоставить всем TCP/UDP-подключениям одинаковую возможность по передаче данных. Для конфигурирования **SFQ** используется два параметра:

- **sfq-perturb** – указывает через какое время нужно менять хэширующий алгоритм, который определяет как будут формироваться под-очередь запросов;
- **pcq-allot** – определяет количество байт в под-очереди.

SFQ работает по следующему принципу: алгоритм изымания пакетов из под-очередей одновременно выпускает в выходной интерфейс pcq-allot количество байт, а хэширующий алгоритм добавляет к каждый под-очереди **pcq-allot** байт, сохраняя при этом равновесие и одинаковую длину всех подочередей. Схему работы SFQ можно сравнить с мясорубкой, в которой через выходную решетку одновременно изо всех дырок в одинаковом количестве выходит фарш :).



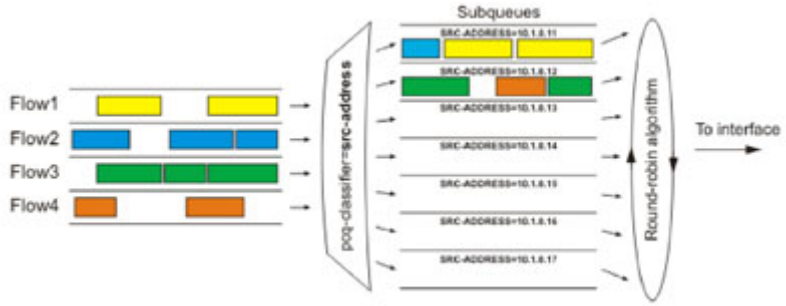
Алгоритм **SFQ** рекомендуется использовать в случаях, когда канал сильно загружен и необходимо предоставить приложениям одинаковую возможность по передаче данных. Единственным его недостатком является то, что одно приложение, открыв много потоков, может заглушить остальные подключения.

PCQ

PCQ (Per Connection Queuing) является частным случаем **SFQ** за тем исключением, что формирование потоков в под-очереди будет происходить в соответствии с неким правилом. Это может быть адрес источника/получателя и порт источника/получателя. Таким образом можно равномерно распределить скорость между участниками вне зависимости от количества открытых подключений. Алгоритм предоставляет следующие параметры для конфигурирования:

- **pcq-classifier** – параметр для формирования очередей. Может принимать следующие значения:
  - **src-address** – параметром для группировки в субочереди служит адрес источника;

    - **src-port** – параметром для группировки в субочереди служит порт источника;
    - **dst-address** – параметром для группировки в субочереди является адрес назначения;
    - **dst-port** – параметром для группировки в субочереди служит порт получателя.
- **pcq-rate** – число, которое указывает в какой пропорции разделять трафик по очередям. По-умолчанию 0.
- **pcq-limit** – длина под-очереди;
- **pcq-total-limit** - общее количество пакетов во всех очередях.

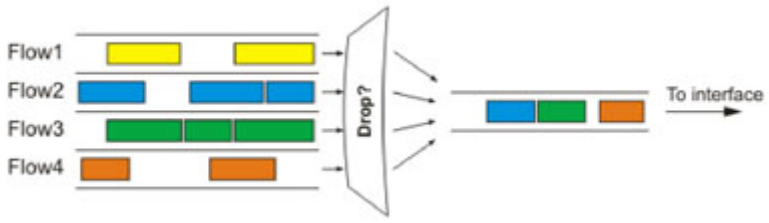


Данный алгоритм является основным при необходимости разделить пропускную способность поровну между классами или клиентами. С его помощью можно организовать динамический шейпинг, о котором много где говорят, но толковая реализация автору ещё не встречалась ни в одном продукте.

Классифицировав под-очереди по адресу источника мы получим отдельную очередь для каждого адреса, соответственно количество потоков с одного адреса не будет играть роли при доступе к выходному интерфейсу.
Стоит отметить гибкость такой классификации. Применив ее по источнику к внешнему интерфейсу, в под-очереди будут попадать внешние адреса, так как в этом случае параметр **src-address** будет все равно содержать адрес, являющийся источником передачи данных. Применив эту же классификацию к внутреннему интерфейсу, в очереди попадут адреса клиентов или классов. Таким образом задав два правила, отличающиеся одним параметром, можно разделить поровну как входящий так и исходящий каналы.

RED

RED (Random Early Detection) – алгоритм, призванный выравнивать пропускную способность и сглаживать скачки, контролируя средний размер очереди. Когда ее размер достигает значения **red-min-threshold** алгоритм удаляет случайно выбранный пакет. Число удаленных пакетов растет с увеличением среднего размера очереди. Если размер достигает значения **red-max-threshold** все пакеты удаляются. Однако случаются ситуации, когда реальный размер очереди (не средний) значительно больше **red-max-threshold**. В таком случае все пакеты, выходящие за рамки предела **red-limit**, удаляются.



Использование алгоритма крайне не желательно при присутствии UDP-трафика, так как в связи с неразборчивостью алгоритма при удалении пакетов из очереди и принципом работы UDP-протокола, данные могут не дойти до получателя.

От теории к практике

Сейчас мы знаем все необходимое для того, чтобы построить необходимые нам правила. Так как на практике применение алгоритмов **SFQ** и **RED** используется крайне редко, то на примерах их работы мы останавливаться не будем.

Queue Trees

Queue Trees - особый тип очередей, который прямо отражает устройство шейпера HTB. Он позволяет строить деревья правил (классов) и на самом низком уровне управлять пакетами.

Вкратце объясним значение основных элементов управления, присутствующих в Queue Trees:

- **burst-limit** (целое) – максимальная burst-скорость;
- **burst-threshold** (целое) – средняя загрузка канала, при которой разрешено выдать burst-limit;
- **burst-time** (время) – используется для подсчета средней загрузки канала;
- **flow** (text) – поток, маркированный в /ip firewall mangle;
- **limit-at** (целое) – гарантированная скорость;
- **max-limit** (целое) – максимальная скорость;
- **name** (text) – имя очереди;
- **parent** (text) – родитель в иерархии классов HTB;
- **priority** (целое: 1..8) – приоритет очереди;
- **queue** (text) – тип очереди. Задается в /queue type.

Примеры

I. Итак, создадим правило, которое позволит получить клиентам локальной или виртуальной сети максимальную скорость и минимальное время отклика при обращении к сайту www.x-drivers.ru , однако разделит скорость между всеми поровну.

1. Пометим все пакеты идущие от пользователей на адрес 66.148.73.54 и обратно. Для этого нужно создать 4 правила, два из которых пометят подключения в прямом и обратом направлении, а другие два пометят пакеты в этих подключениях. Необходимо обратить внимание, что очереди работают именно с пакетами, а не помеченными подключениями. Зачастую это создает у новичков вопросы плана: "Я пишу все правильно, а оно не работает. Может это глюки?" Для таких шаманов ниже приведен пример, как нужно делать, чтобы ОНО работало.

```
/ip firewall mangle add chain=forward src-address=192.168.11.0/24 dst-
address=66.148.73.54/32 action=mark-connection new-connection-mark=users-con-up

/ip firewall mangle add connection-mark=users-con-up action=mark-packet
new-packet-mark=users-up chain=forward
/ip firewall mangle add chain=forward src-address=66.148.73.54/32 \
   action=mark-connection new-connection-mark=users-con-down

/ip firewall mangle add connection-mark=users-con-down action=mark-packet \
   new-packet-mark=users-down chain=forward
```

2. Создадим два типа PCQ очереди, подключения в одной из которых мы будем классифицировать по входящему, а другие по исходящему адресу.

```
/queue type add name=pcq-download kind=pcq pcq-classifier=dst-address
/queue type add name=pcq-upload kind=pcq pcq-classifier=src-address
```

3. Создадим очереди для входящего и исходящего трафика

```
/queue tree add name=Download parent=Local max-limit=10240000 burst-limit=200000 burst-
time=10
/queue tree add parent=Download queue=pcq-download packet-mark=users-down
/queue tree add name=Upload parent=Public max-limit=160000 burst-limit=200000 burst-
time=10
/queue tree add parent=Upload queue=pcq-upload packet-mark=users-up
```

Как видим, пара правил и такая сложная задача, как динамический шейпинг на определенные адреса с предоставлением пиковой скорости была реализована. На практике обычно приходится применять более сложные правила в сочетании друг с другом, однако для демонстрации возможностей этот очень хорошо подошел.

Практически в Queue Trees мы можем оперировать только ограничением скорости. Остальные параметры, такие как адрес источника, получателя, время суток, протокол, порты и т.д. указываются в разделе Mangle-файервола.

Подобным способом также можно сделать хороший пинг на определенные адреса, даже во время серьезной загрузки канала. Приведем пример таких правил для сервера www.cybernet.by.

Для этого нужно создать четыре правила в Firewall:

- в первом пометим все подключения, у которых адрес получателя 195.222.70.250 именем cybernet-connection-up;
- пакеты в подключениях cybernet-connection-up именем cybernet-packet-u;
- в первом пометим все подключения у которых адрес источника 195.222.70.250 именем cybernet-connection-from;
- пакеты в подключениях cybernet-connection-from именем cybernet-packet-from.

```
/ip firewall mangle add chain=prerouting dst-address=195.222.70.250 action=mark-
connection new-connection-mark=cybernet-connection-up passthrough=yes

/ip firewall mangle add chain=forward connection-mark=cybernet-connection-up action=mark-
packet new-packet-mark=cybernet-packet-up passthrough=yes

/ip firewall mangle add chain=prerouting src-address=195.222.70.250 action=mark-
connection new-connection-mark=cybernet-connection-from passthrough=yes

/ip firewall mangle add chain=forward connection-mark=cybernet-connection-from
action=mark-packet new-packet-mark=cybernet-packet-from passthrough=yes
```

Следующим шагом создадим два правила в Queue Trees: одно для входящего потока, другое для исходящего.

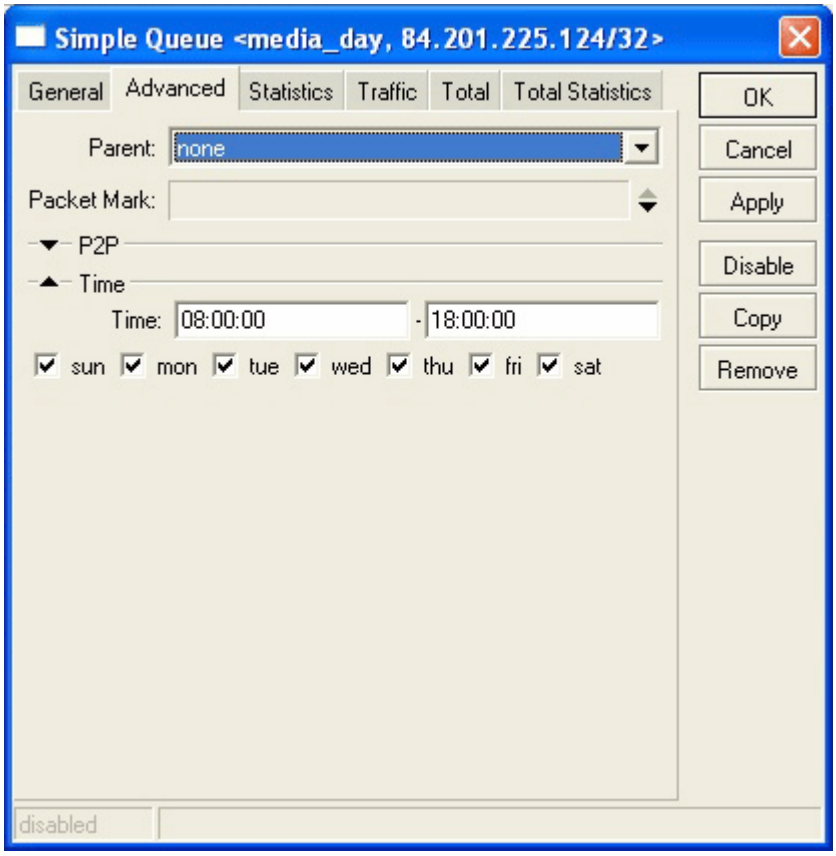
```
/queue tree add  name="queue1" parent=global-out packet-mark=cybernet-packet limit-at=0 \
    queue=default priority=1 max-limit=50000 burst-limit=0 burst-threshold=0 \
    burst-time=0s

/queue tree add  name="queue2" parent=global-in packet-mark=cybernet-packet-from limit-at=0 queue=default
priority=1 max-limit=50000 burst-limit=0 burst-threshold=0 \
    burst-time=0s
```

Вышеописанные действия позволят пакетам, приходящим и уходящим с адреса 195.222.70.250, попадать в приоритетные очереди и всегда иметь гарантированные 50 Kbit/s.

Пользуясь вышеописанным примером можно выделить для всех онлайн игр фиксированную гарантированную скорость и создать иерархию классов (правил), распределив таким образом пропускную способность между всеми поровну.

Simple Queues

Рассмотрим тип очередей Simple Queues. Эти очереди являются простыми, а если быть точнее, то упрощенными. В самом деле для их использования не нужно применять промаркированные пакеты из Firewall, однако при этом теряется некоторая гибкость. Здесь в качестве основных параметров, к которым следует применять правило, относятся адреса источника и получателя. На вкладке Advanced утилиты Winbox можно обнаружить и некоторые другие параметры, однако они не относятся к списку обязательных к заполнению.



Список текущих возможностей простых очередей выглядит следующим образом:

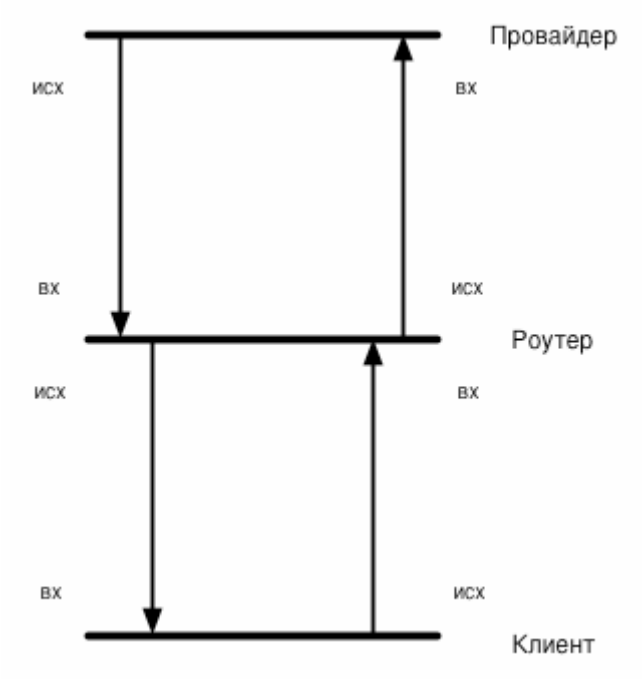
- работа с P2P-трафиком;
- применение очередей в определенных интервалах времени;
- приоритезация потоков;
- использование в качестве параметров несколько цепочек пакетов, маркированных в /ip firewall mangle;
- шейпинг бидеракционного трафика (одно правило для входящего и исходящего потока).

Стоит отметить, что начиная с версии Mikrotik 2.9 в Simple Queues стало возможным указывать параметр Parent для простых очередей. Таким образом можно строить практически аналогичные деревья классов, как и для Queue Trees за тем исключением, что здесь оперировать будем не пакетами и потоками, а адресами.

Стоит так же помнить, что Simple Queues не что иное, как частный случай Queue Trees. Поэтому создавая новые правила стоит обращать внимание не существует ли уже что-то подобное, оперирующее

с теми же адресами, портами или другими параметрами. В случае сходства, приоритет окажется на стороне Queue Trees и ваши простые очереди просто не будут работать.

Перед началом составления правил в Simple Queues необходимо понять что есть входящий и исходящий трафик для нашего провайдера, нашего роутера и наших клиентов. На рисунке выше изображена стандартная схема направления потоков трафика от провайдера к нашему роутеру и от роутера к клиенту. Стоит обратить внимание, что исходящий трафик клиента для нашего роутера будет являться входящим и наоборот. То же самое можно сказать про нашего провайдера-исходящий трафик для него-это входящий для нас.



Список параметров для конфигурирования простых очередей в Mikrototik 2.9.7 выглядит следующим образом:

- burst-limit (целое) – максимальная burst-скорость;
- burst-threshold (целое) – средняя загрузка канала при которой разрешено выдать burst-limit;
- burst-time (время) – используется для подсчета средней загрузки канала;
- dst-address (IP адрес/маска) – адрес назначения;
- dst-netmask (netmask) – маска подсети для dst-address;
- interface (text) – интерфейс, для которого предназначается правило;
- limit-at (целое/целое) – гарантированный канал;
- max-limit (целое/целое) – максимальная величина канала;
- name (text) – имя правила;
- p2p (any | all-p2p | bit-torrent | blubster | direct-connect | edonkey | fasttrack | gnutella | soulseek | winmx) – тип P2P-трафика;
- packet-marks (name; по умолчанию: "") - цепочка пакетов, промаркированных в /ip firewall mangle;
- parent (name) – имя родительской очереди;
- priority (целое: 1..8) – приоритет. 1- больший, 8-самый маленький;
- queue (name/name; default: default/default) – имя очереди из /queue type;
- target-addresses (IP address/netmask) – исходный адрес;
- time (time-time,sat | fri | thu | wed | tue | mon | sun{+}; по умолчанию: "") - применить очередь к временному интервалу;
- total-burst-limit (целое) – максимальная burst скорость в очереди global-total;• total-burst-threshold (целое) – средняя скорость в очереди global-total;
- total-burst-time (time) - используется для подсчета средней загрузки канала в очереди global-total;
- total-limit-at (целое) – гарантированная скорость в очереди global-total (входящий+исходящий каналы);

- total-max-limit (целое) – максимальная скорость передачи данных в очереди global-total.

Весьма интересной является возможность управлять входящим и исходящим трафиком вместе. Это позволяет клиентам максимально использовать проплаченный канал.

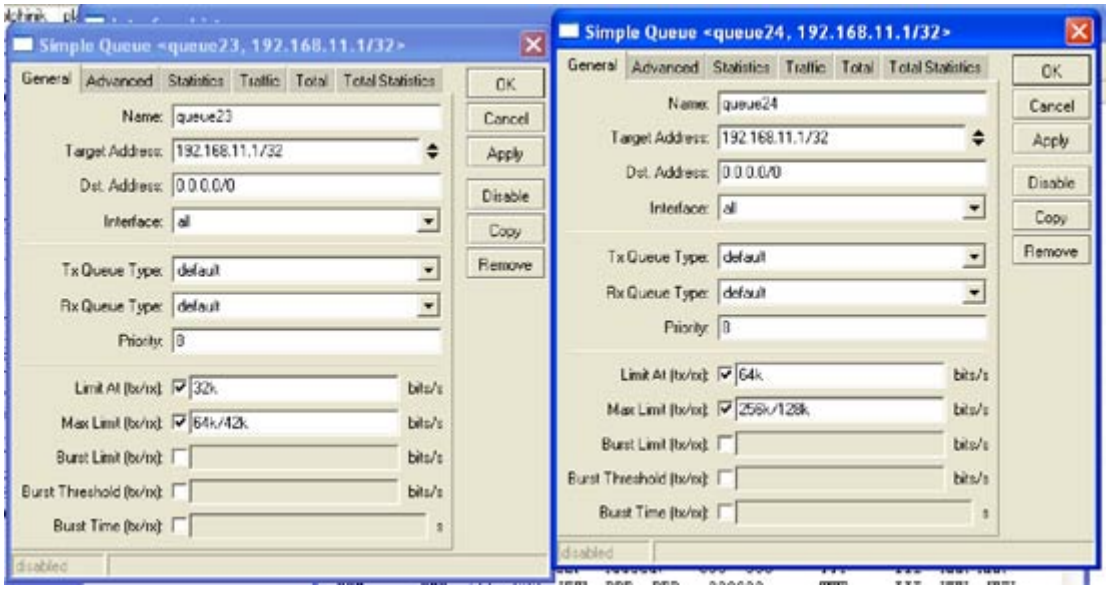
Примеры

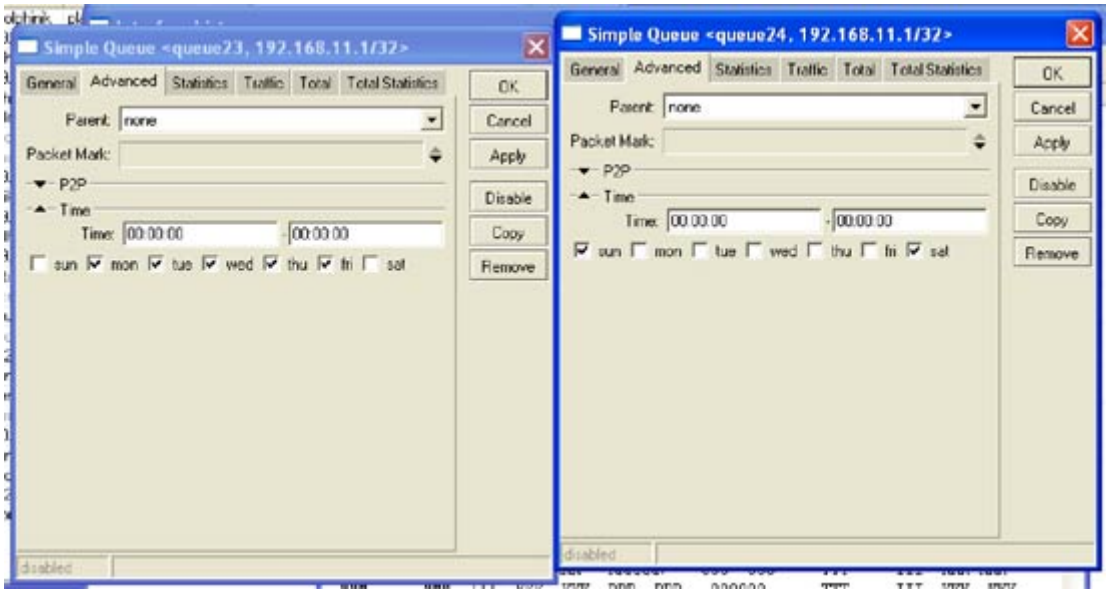
Сэмулируем для нашего клиента канал с 64/42 kbit/s с гарантированной скоростью 32/32 kbit/s в будние дни и 256/128 kbit/s с гарантированной скоростью 64/64 kbit/s в выходные.

Результатом нашей работы будет два правила:

```
/queue simple add target-addresses=192.168.11.1/32 limit-at=32000/32000 max-
limit=64000/42000  time=00:00:00-00:00:00,mon,tue,wed,thu,fri

/queue simple add target-addresses=192.168.11.1/32 limit-at=64000/64000 max-
limit=256000/128000  time=00:00:00-00:00:00,sat,sun
```

В нашем случае первая цифра в паре 32000/32000 означает входящий канал для клиента, а вторая - исходящий. Воспользовавшись нижеприведенной схемой можно инвертировать параметры относительно нашего роутера.





При желании можно позаботиться о быстром открытии страниц, добавив параметры burst-limit и burst-time.

Данное правило можно было бы несколько видоизменить, разделив указанную скорость между всей сетью 192.168.11.0.24. В этом случае для параметра queue нужно указать тип очереди pcq-download, приведенный выше в примере с Queue Trees.

Простые очереди позволяют достаточно просто реализовать возможность предоставления на определенные адреса неограниченной скорости.

Добавим к вышеприведенным правилам ещё одно, которое будет выглядеть следующим образом:

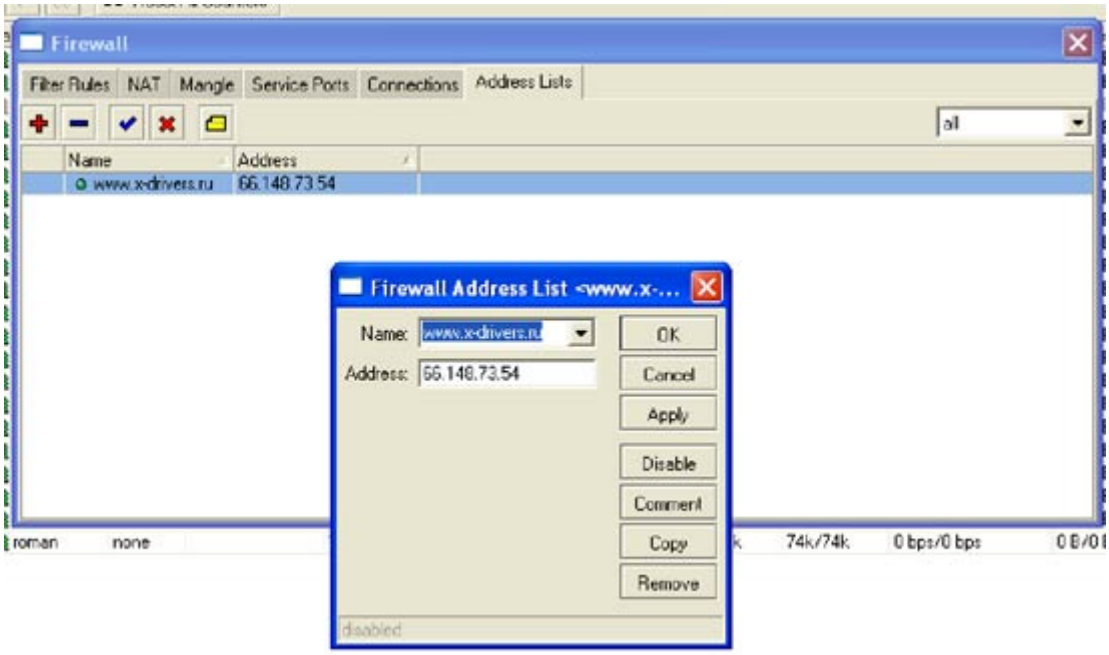
```
/queue simple add target-addresses==192.168.11.1/32 dst-address=84.201.225.124 limit-at=1024000/1024000 max-limit=2048000/2048000
```

В результате чего клиент с адресом 192.168.11.1 получит доступ к адресу 84.201.225.124 со скоростью от 1 до 2 мегабит.

При выделении гарантированной пропускной способности нужно помнить, что сумма limit-at всех клиентов должна быть меньше или равна общей пропускной способности канала. Только в этом случае можно говорить о какой-то гарантированной скорости.

Если у вашего сервера неправильно установлены часы или, к примеру, села батарейка на материнской плате, то можно воспользоваться встроенной в Mikrotik возможностью синхронизации времени с внешним источником.

Ещё одно замечание связанно с тем, что напрямую вписать URI-адрес в поле ввода IP-адреса не представляется возможным, но иногда необходимо, так как многие сайты имеют динамические адреса. Данную проблему можно решить, прописав необходимый адрес ресурса в разделе /ip firewall address-list и дав ему имя, по которому в последствии можно обратиться.



Вывод

Возможности по управлению трафиком в Mikrotik 2.9 претерпели некоторые изменения по сравнению с предыдущими версиями. Это позволило стать этой операционной системе рядом и даже несколько потеснить аппаратные решения от Cisco, сохранив при этом удобство и огромную гибкость в

конфигурировании. Объединив Mikrotik и биллинг с поддержкой протокола Radius, можно получить мощную систему, подходящую как для сети среднего провайдера, так и в качестве сервера, раздающего Интернет внутри локальной домашней сети или сети предприятия. Стоит отметить, что вышеописанные возможности шейпинга можно применить не только к разделению доступа в интернет, но и к созданию резервных каналов связи. В сочетании с беспроводными технологиями RouterOS может превратить груду уже никому не нужного металлолома и нескольких беспроводных сетевых карт в мощную Wi-Fi соту с разделением доступа и предоставлением гарантированной полосы пропускания своим клиентам.

Обсудить статью и задать вопросы автору на форуме!
Читать статью "Router OS Mikrotik - мечта сисадмина"