

# MikroTik RouterOS

## Семинар

### «Поговорим о QoS»

Перевёл на русский язык white\_crow

Если Вы нашли ошибку перевода — напишите мне на [white.crow.4096@gmail.com](mailto:white.crow.4096@gmail.com)

(все таки я учил в школе французкий, а не английский :)

Las Vegas  
MUM USA 2011

# Обо мне

- Jānis Meģis, MikroTik
- Технический специалист (НЕ продавец :)
- Техническая поддержка и обучение почти 7 лет
- Специализация: QoS, PPP, Firewall, Routing
- Обучение MikroTik RouterOS с 2005 года

# План семинара

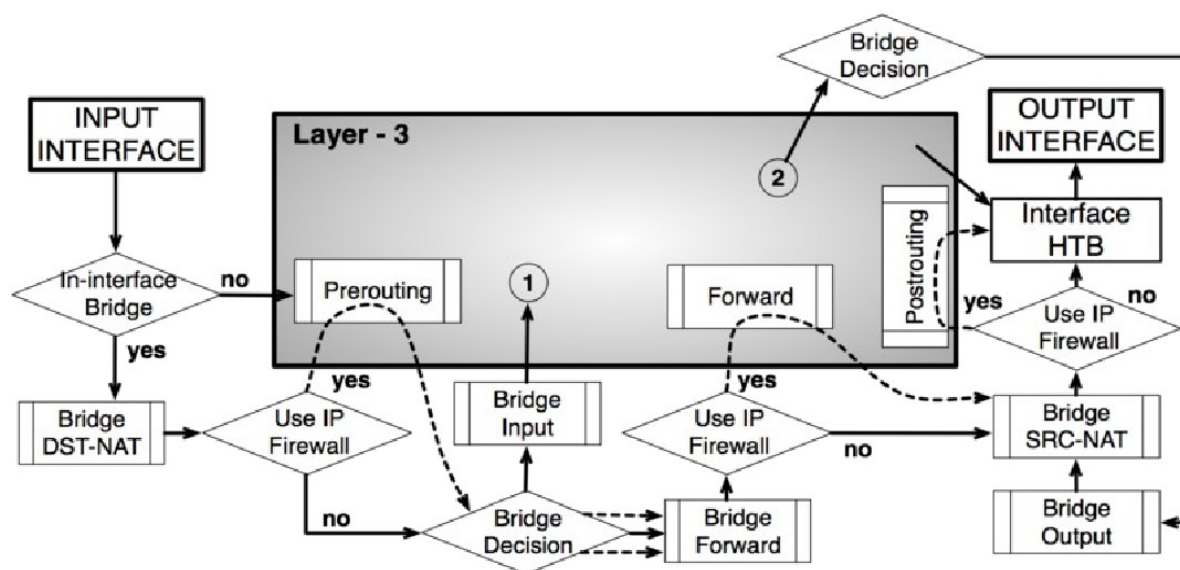
- ▶ Диаграмма движения пакетов
- ▶ Иерархия НТВ (Hierarchy Token Bucket)
- ▶ Типы очередей (PCQ, multi-queue-fifo)
- ▶ Burst - кратковременный всплеск трафика
- ▶ Размер очереди
- ▶ Дерево очередей и простые очереди

# Диаграмма движения пакетов

- ▶ Диаграмма движения пакетов — это «Большой рисунок» операционной системы RouterOS
- ▶ Невозможно должным образом управлять и поддерживать сложные конфигурации без понимания того - что, где, когда и почему происходит.
- ▶ Диаграмма движения пакетов состоит из двух частей:
  - ▶ Bridging (MAC) — эта часть диаграммы описывает движение пакетов на втором уровне семиуровневой модели OSI. Здесь часть диаграммы, описывающая маршрутизацию, упрощена до одного «чёрного ящика» - типа «Layer - 3»
  - ▶ Routing (IP) - эта часть диаграммы описывает движение пакетов на третьем уровне семиуровневой модели OSI. Здесь часть диаграммы, описывающая коммутацию, упрощена до одного «чёрного ящика» «Bridging»

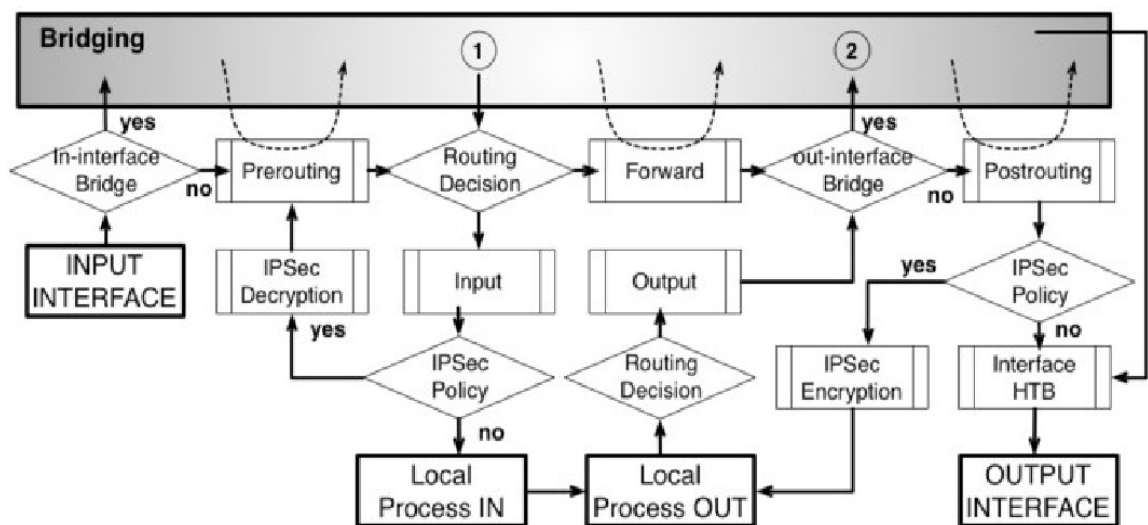
# Второй Уровень (Коммутация)

## Bridging or Layer-2 (MAC)

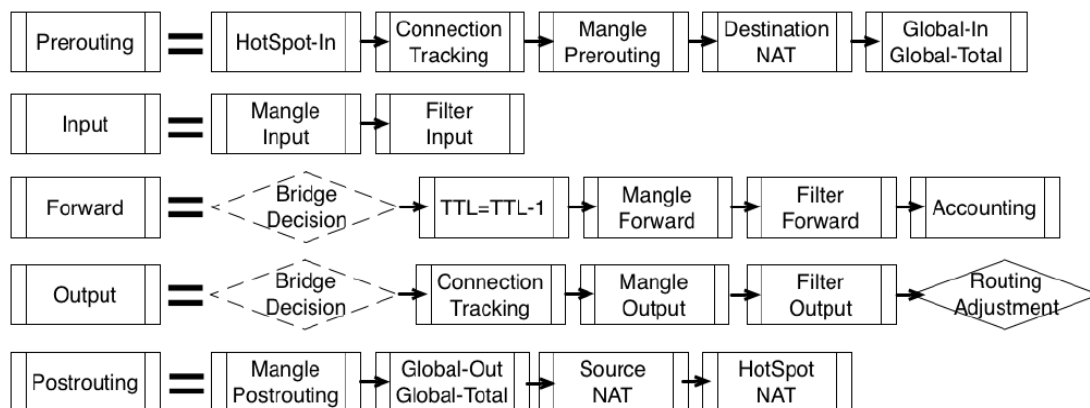


# Третий Уровень (Маршрутизация)

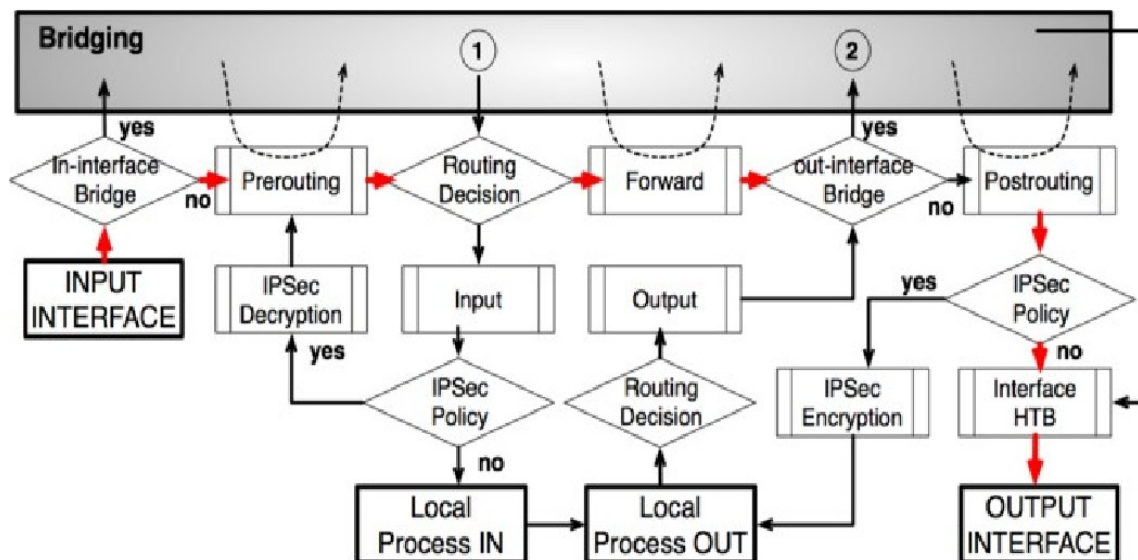
## Routing or Layer-3 (IP)



# Сокращенная диаграмма

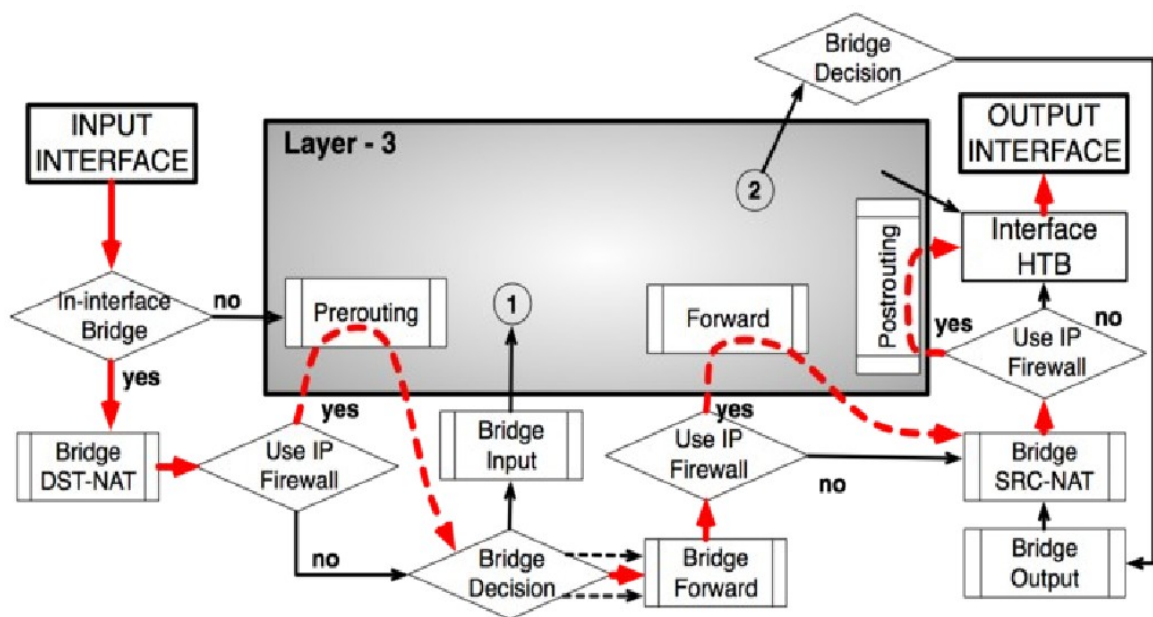


# Простая маршрутизация



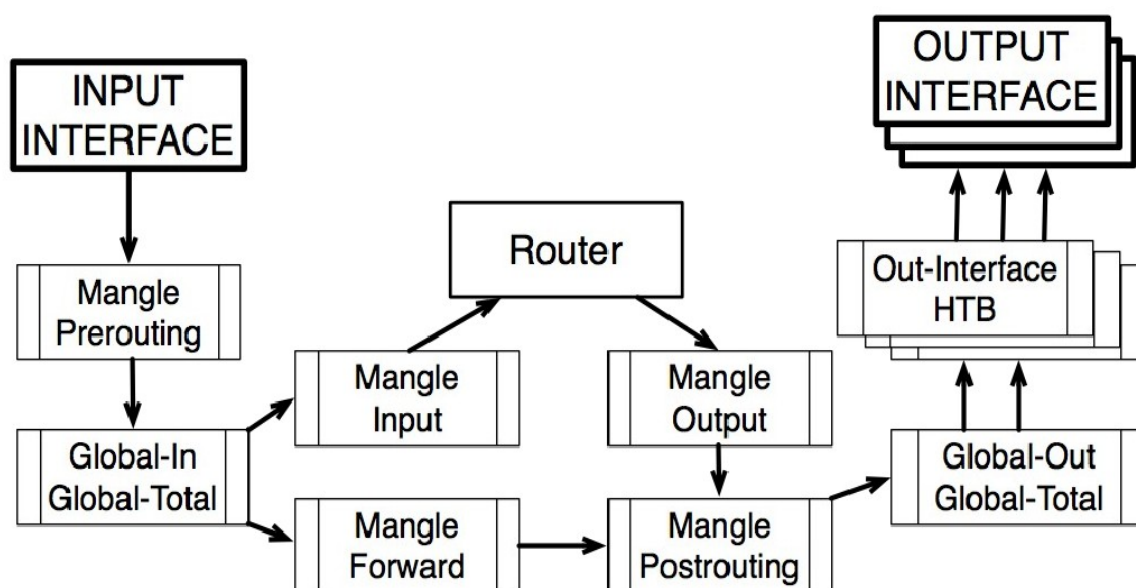


# Коммутация с использованием IP Firewall



# HTB (Hierarchical Token Bucket)

QoS Packet Flow Diagram  
Диаграмма движения пакетов

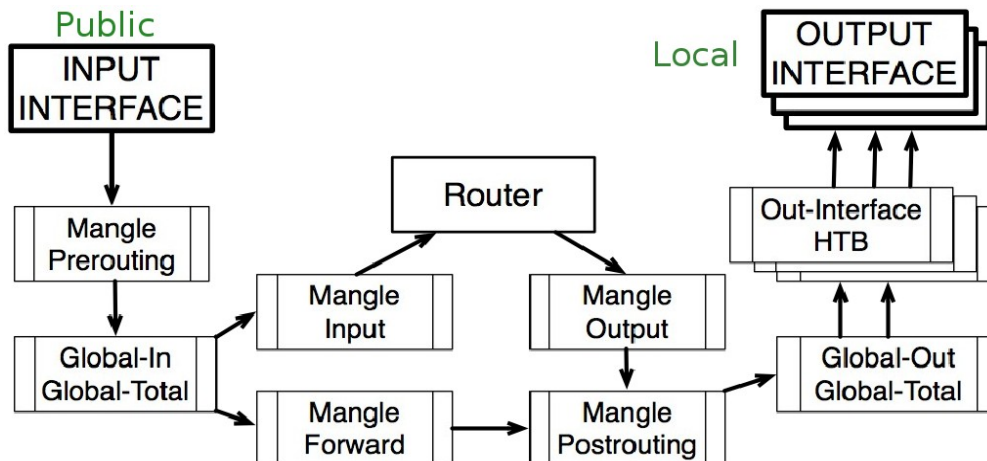


## От переводчика:

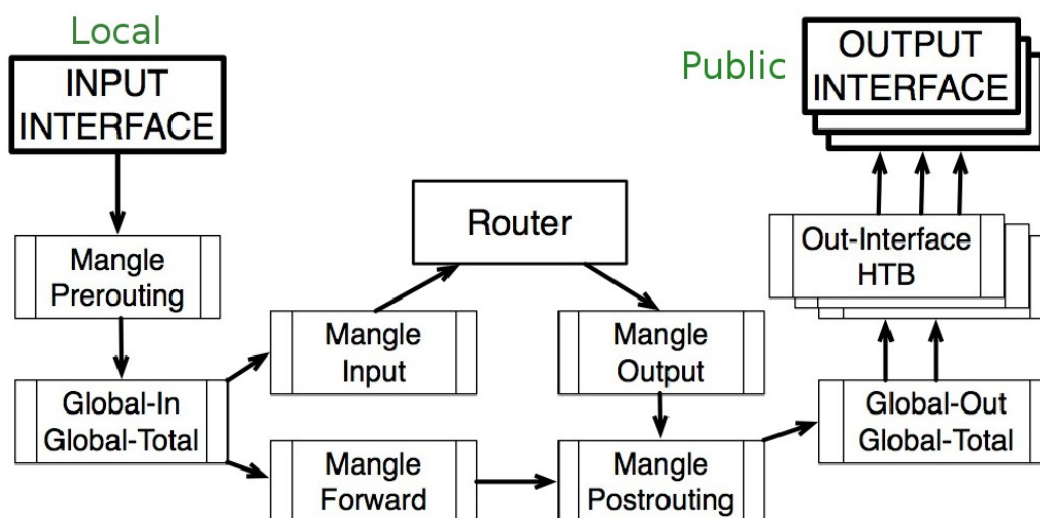
Важно понимать, что через роутер проходит несколько потоков. Например, в самом простом случае имеем один физический интерфейс роутера - Public, и один физический интерфейс - Local.

Итого - имеем два потока пакетов — проходящих через роутер - назовем условно Download и Upload потоки.

### Download поток



### Upload поток



Для Download трафика входным интерфейсом будет Public, выходным интерфейсом будет Local.

И, наоборот, для Upload трафика входным интерфейсом будет Local, выходным интерфейсом будет Public.

При этом диаграмма движения пакетов и логика обработки потоков будет абсолютно одинаковой для обоих потоков.

Другими словами - для роутера не имеет значения роль интерфейса — т.е. разделение на Public и Local - условно.

# Global-Out

## или

### Interface HTB?

Существует два важных нюанса, в зависимости от которых нужно использовать либо *Global-Out* либо *Out-Interface HTB* :

► В случае SRC-NAT (masquerade) - *Global-Out* будет знать о частных адресах клиента, но *Out-Interface HTB* **не будет!**, потому что следует после SRC-NAT.

(Прим. переводчика: если используете NAT — используйте *Global-Out*)

► Каждый *Out-Interface HTB* получает только тот трафик, который будет уходить через определенный интерфейс - нет необходимости раздельно маркировать трафик upload и download в таблице mangle

(Прим. переводчика: я не заметил от этого никакого профита - поэтому в любом случае использую Global-out и маркирую отдельно пакеты для Upload и Download - это универсальный план шейпинга - работает вне зависимости от того, используете ли вы NAT или нет. А также работает при использовании rppoe/l2tp/pptp, lpoE :), HOTSPOT.

# Mangle

- ▶ В специальной таблице Firewall Mangle возможно маркировать IP пакеты специальными метками, исходя из различных признаков трафика.
- ▶ Затем эти метки могут использоваться внутри маршрутизатора. Например: для управления полосой пропускания и/или маршрутизации.
- ▶ В добавок, средствами mangle можно модифицировать некоторые поля в IP заголовке, такие как TOS (DSCP) и TTL.

# HTB (Hierarchical Token Bucket)

- ▶ Вся реализация управления пропускной способностью в RouterOS основана на иерархии - Hierarchical Token Bucket (HTB)
- ▶ HTB позволяет вам создавать иерархические (древовидные) структуры очередей и определять отношения между ними.
- ▶ RouterOS поддерживает 3 виртуальных HTBs (global-in, global-total, global-out) и еще один прямо перед каждым выходным интерфейсом.

# НТВ

## Hierarchical Token Bucket

### (продолжение)

- ▶ Когда пакеты проходят **сквозь** роутер, они проходят все 4 НТВ дерева.
- ▶ Когда пакеты приходят **на** роутер, они проходят только global-in и global-total НТВ.
- ▶ Когда пакеты уходят **с** роутера, они проходят только global-out, global-total и interface НТВ.

# Особенности НТВ.

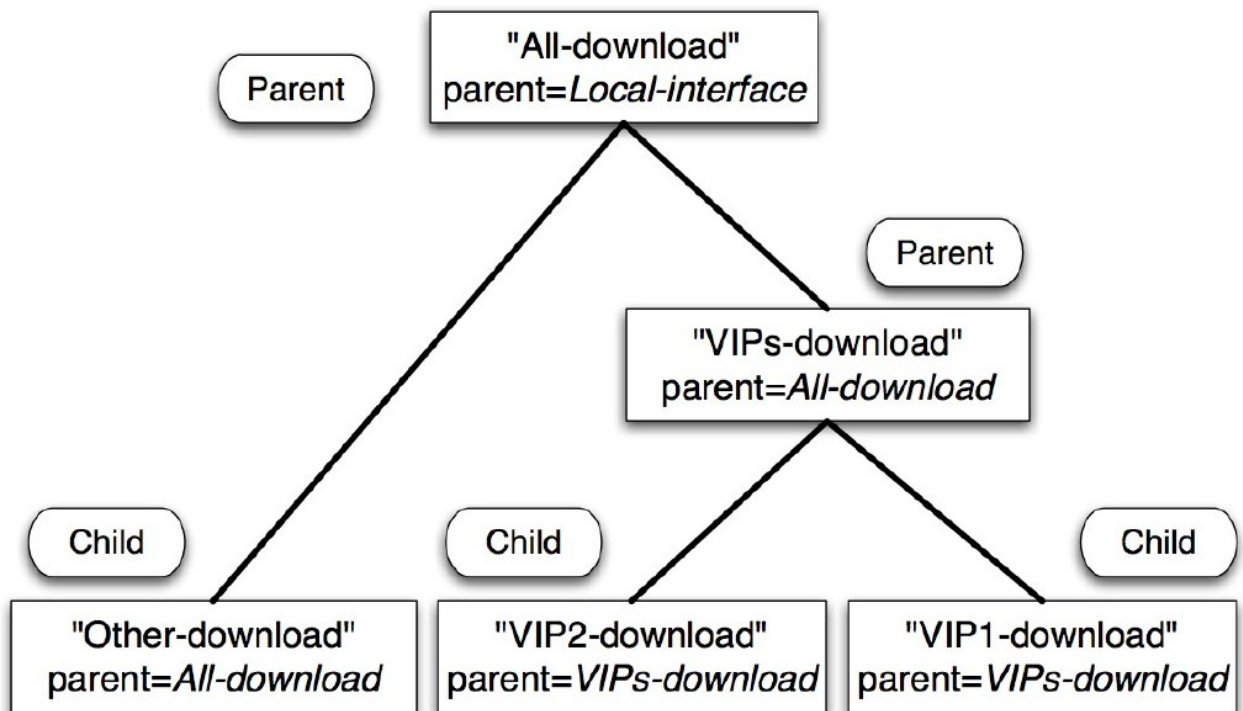
## Структура

- ▶ Если очередь имеет по крайней мере одного потомка – она является родительской очередью
- ▶ Все очереди-потомки (не имеет значения сколько уровней родителей они имеют), находятся на том же нижнем уровне НТВ.
- ▶ Потомки создают реальное потребление трафика, родители отвечают только за распределение трафика.
- ▶ Потомки получают сначала предел трафика `limit-at`, а остальной трафик будет распределен родителями.



# Особенности HTB. Структура

## HTB Features - Structure



# Особенности НТВ. Двойное ограничение.

- ▶ НТВ имеет два предела скорости:
  - ▶ **CIR** (гарантированная скорость передачи данных) - (**limit-at** в RouterOS): при худшем сценарии поток получит это количество трафика, несмотря ни на что (при условии, что мы можем на самом деле отправить так много данных)
  - ▶ **MIR** (максимальная скорость передачи данных) - (**max-limit** в RouterOS): при лучшем сценарии, скорость потока может быть повышена, если родительская очередь имеет запас полосы пропускания
  - ▶ Первым делом НТВ будет стараться удовлетворить каждого потомка скоростью **limit-at** - и только затем будет пытаться достичь **max-limit**

# Двойное ограничение

- Максимальная скорость родителя должна быть равна или больше, чем сумма гарантированных скоростей потомков

$$\text{MIR (родитель)} \geq \text{CIR(потомок\_1)} + \dots + \text{CIR(потомок\_N)}$$

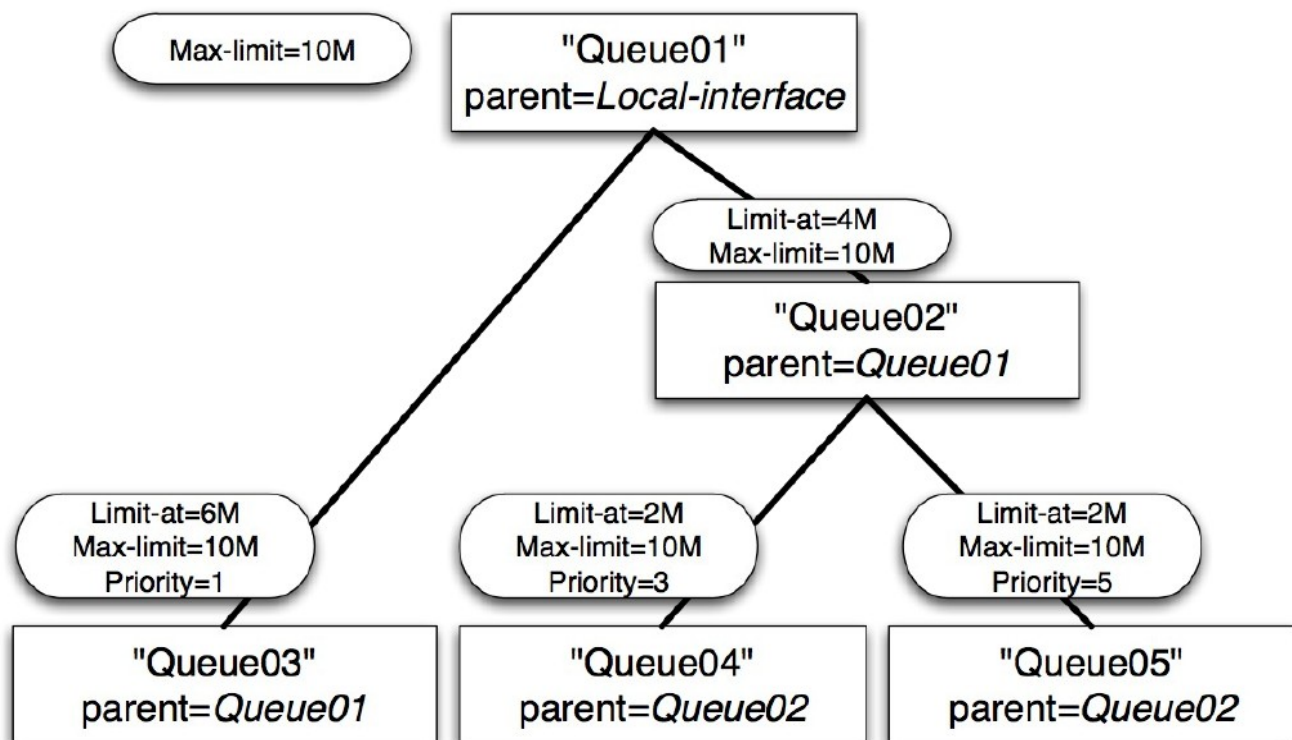
- Максимальная скорость любого потомка должна меньше или равна максимальной скорости родителя

$$\text{MIR (родитель)} \geq \text{MIR(потомок\_1)}$$

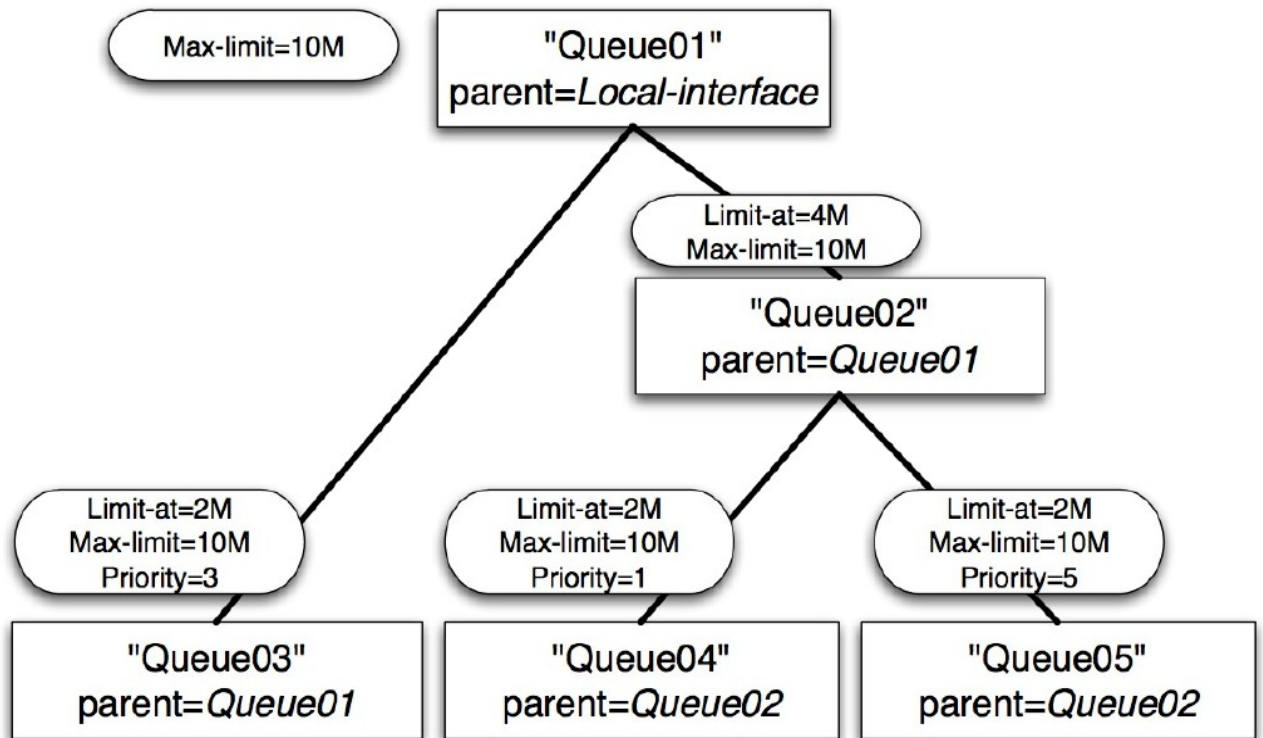
$$\text{MIR (родитель)} \geq \text{MIR(потомок\_2)}$$

$$\text{MIR (родитель)} \geq \text{MIR(потомок\_N)}$$

# HTB - limit-at



# HTB - max-limit



# Особенности НТВ. Приоритет.

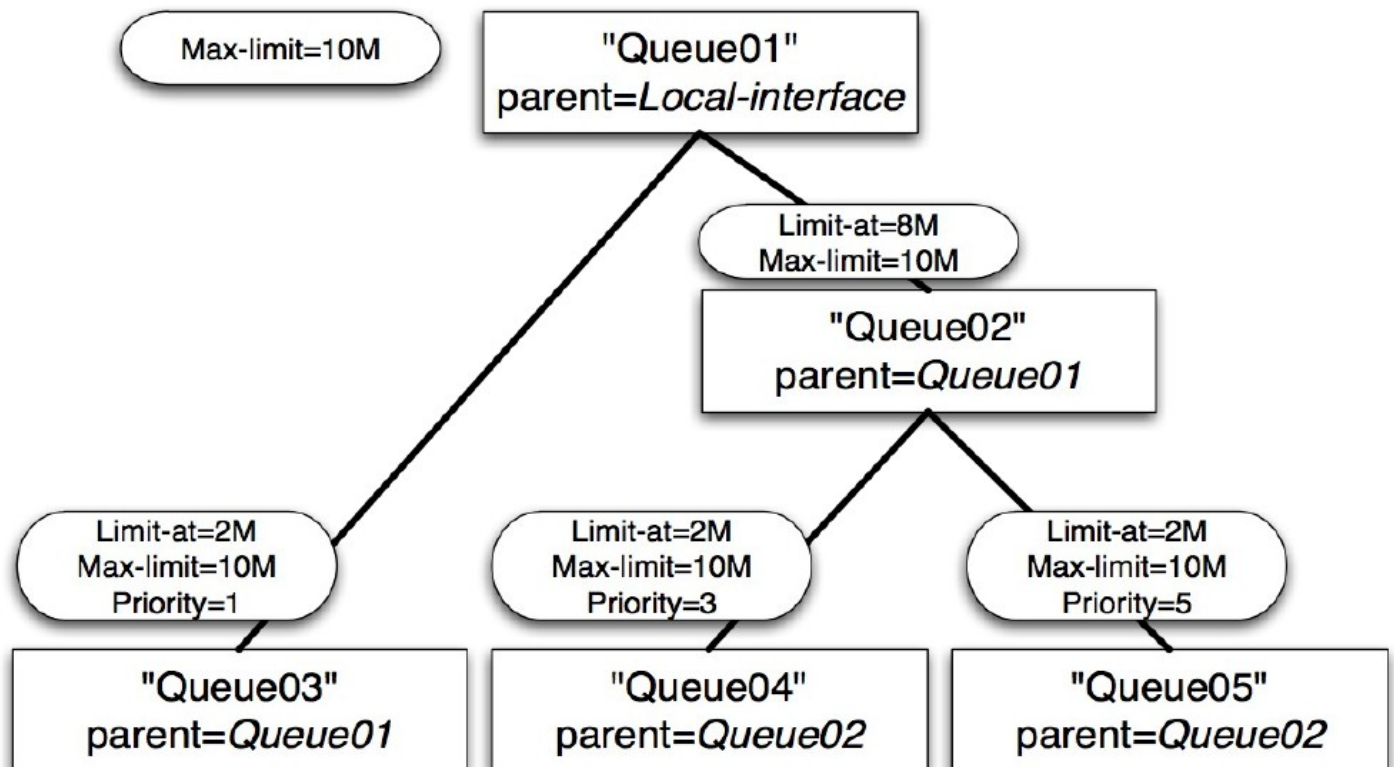
- ▶ Работает только для потомков
- ▶ 1 – высший приоритет
- ▶ 8 – низший приоритет
- ▶ Очередь с более высоким приоритетом получает возможность получить максимальную скорость (max-limit) раньше очередей с более низким приоритетом
- ▶ Фактическая приоритезация трафика будет работать, **только** если заданы лимиты. Очередь без лимитов не будет “приоритезироваться”

# QoS.

## Разрушение легенд.

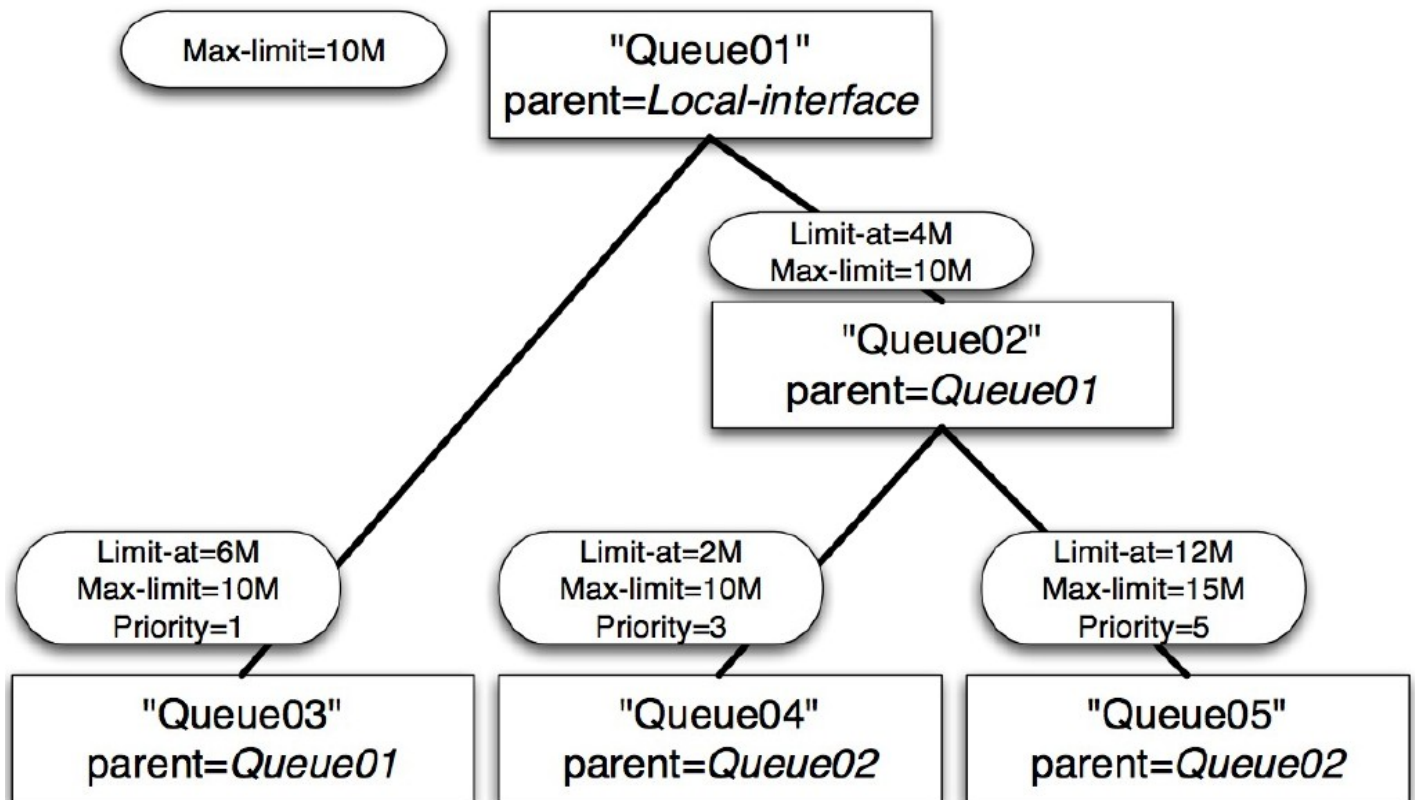
- ▶ НТВ приоритезация не меняет последовательность пакетов - не переставляет одни пакеты раньше других
- ▶ В НТВ - приоритезация является инструментом, который помогает решить - какие пакеты будут проходить дальше, а какие будут отброшены.
- ▶ Решение дропать пакет основывается на лимитах - таким образом, если пределы скоростей не установлены, то приоритеты не имеют никакого эффекта.
- ▶ Приоритет также никак не влияет на трафик пакетов, двигающихся со скоростью меньше или равной гарантированной (CIR). Пакеты просто проходят сквозь алгоритм QoS (даже если родители реально не могут обеспечить такую пропускную способность).

# HTB – limit-at of the Parent





# HTB – limit-at > parent's max-limit



# QoS.

## Разрушение легенд.

- ▶ QoS не может контролировать количество входящего трафика, который вы видите на любом из интерфейсах роутера.
- ▶ На диаграмме движения пакетов видно, что HTB global-in находится после входного интерфейса, где регистрируется к-во пришедшего на роутер трафика.
- ▶ При этом эффект снижения трафика, скорее всего, является эффектом поведения протокола TCR.
- ▶ Единственный способ увидеть QoS в действии является мониторинг передачи данных (TX) противоположного интерфейса.

### *Примечание переводчика:*

- Другими словами - Вы никак не можете повлиять с помощью своего QoS на тот реальный входящий трафик, что уже фактически пришел на любой из интерфейсов роутера - ведь ни ваши юзеры, ни "мир" - ничего не знают про ваш QoS.  
!!! Но это не значит - что QoS не работает :)  
!!! QoS работает :) (просто надо понимать как и для каких задач использовать механизмы управления трафиком)

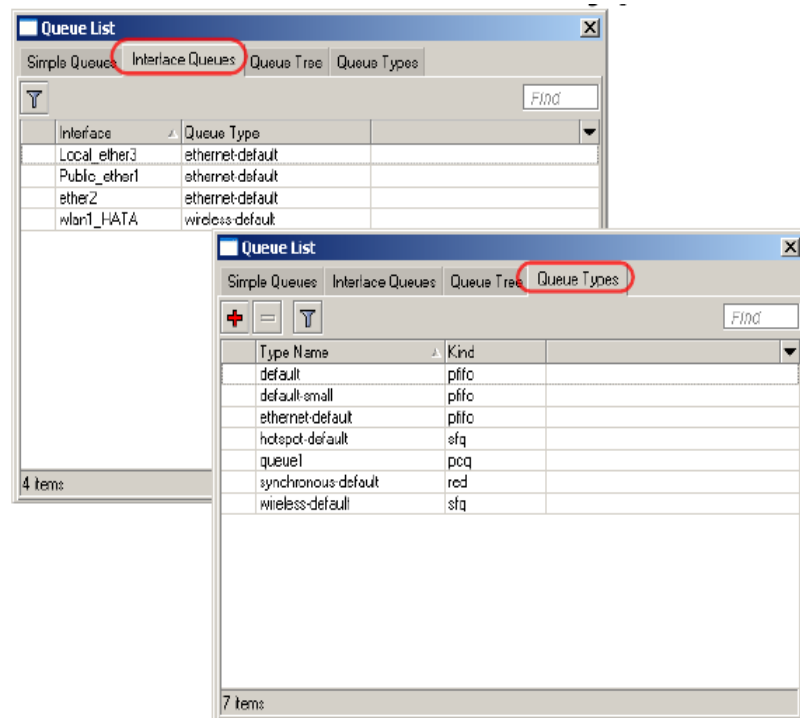
# QoS.

## Разрушение легенд.

- QoS не может знать, какова фактическая пропускная способность доступна.
- Драйвер **выходного** интерфейса является первым, кто может знать, какую пропускную способность вы фактически имеете. Но в диаграмме движения пакетов видно, что выходной интерфейс находится уже после всех НТВ.
- Драйвер интерфейса знает лишь максимальное аппаратное ограничение интерфейса, но если фактическое ограничение меньше - единственный способ обеспечить алгоритм QoS информацией о реальной пропускной способности - указать вручную явно все пределы.

# Типы Очередей (Queue Types)

## Default Queue Types



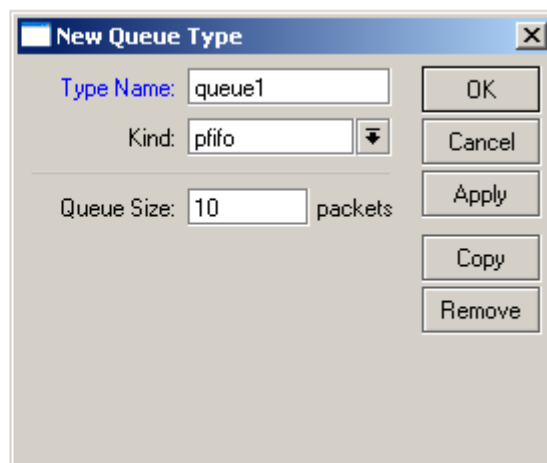
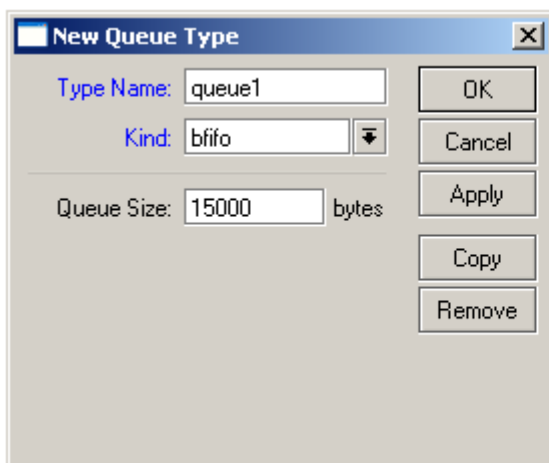
# FIFO

(first in, first out)

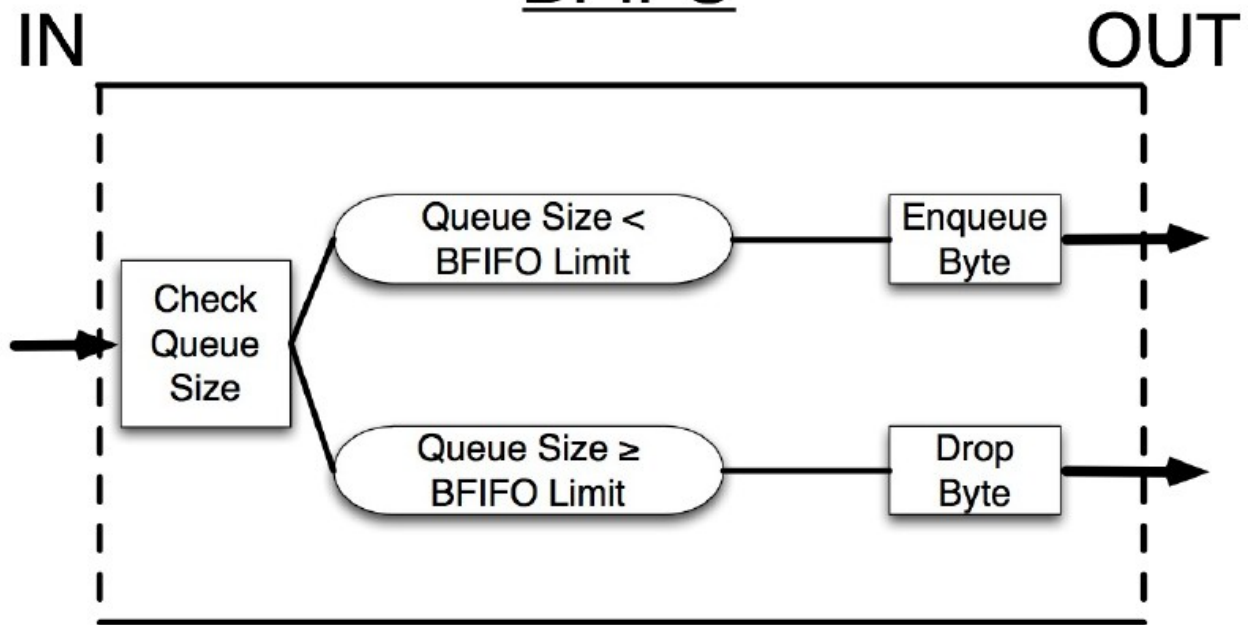
(первым вошел — первым вышел)

## Поведение:

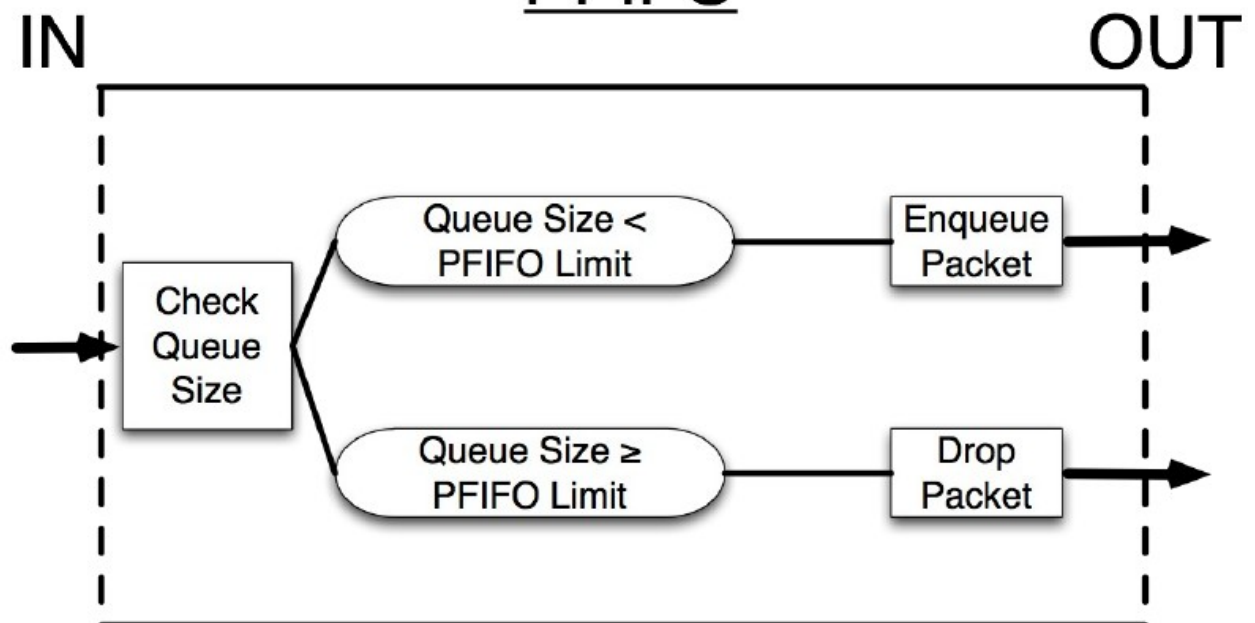
Что пришло первым — обрабатывается первым. Что пришло следующим — ожидает, пока обрабатывается первое. Количество ожидающих юнитов (пакеты или байты (PFIFO или BFIFO)) ограничено опцией «queue size» (размер очереди). Если очередь заполнена — следующая порция дропается.



## BFIFO



## PFIFO



# MQ PFIFO (MultiQueue FIFO)

MQ PFIFO очередь была разработана для многоядерных роутеров (RB1100AHx2)

MQ PFIFO должны использоваться по умолчанию для интерфейсов Ethernet, которые имеют несколько RX/TX очередей (вы можете проверить это в меню system-resources-IRQ).

MQ FIFO является альтернативой RPS (receive Packet Steering) - так что не используйте обе фишки на том же интерфейсе - это приведет к потере производительности.

# RED

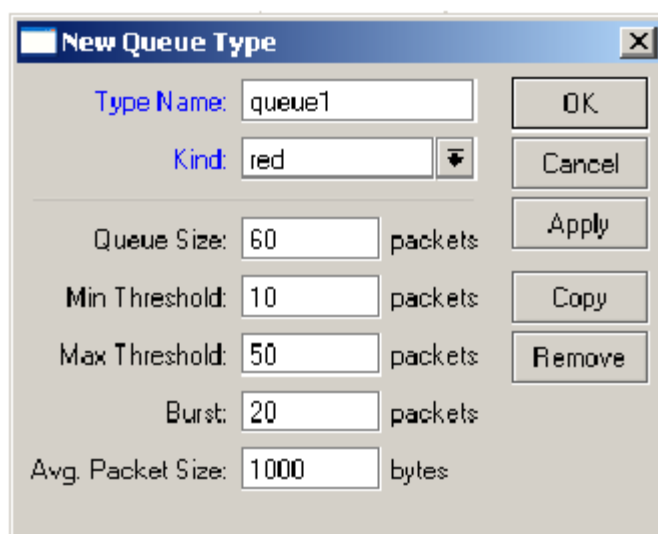
## (Random early detection)

(Произвольное раннее Обнаружение)

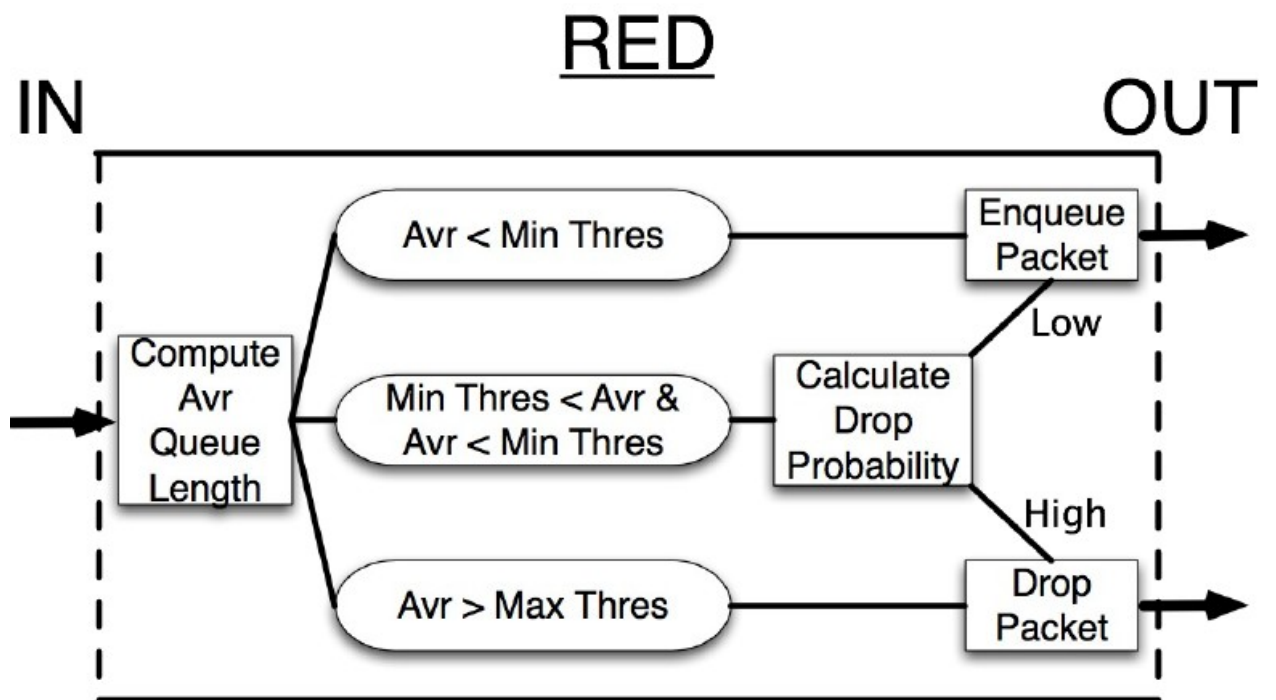
### **Поведение:**

такое же, как FIFO, только с дополнительной функциональностью — вероятностью дополнительных дропов, даже если очередь не переполнена.

Эта вероятность основывается на сравнении средней длины очереди за некоторый период времени с минимальным и максимальным порогом — ближе к максимальному порогу — больше вероятность отбрасывания пакета.





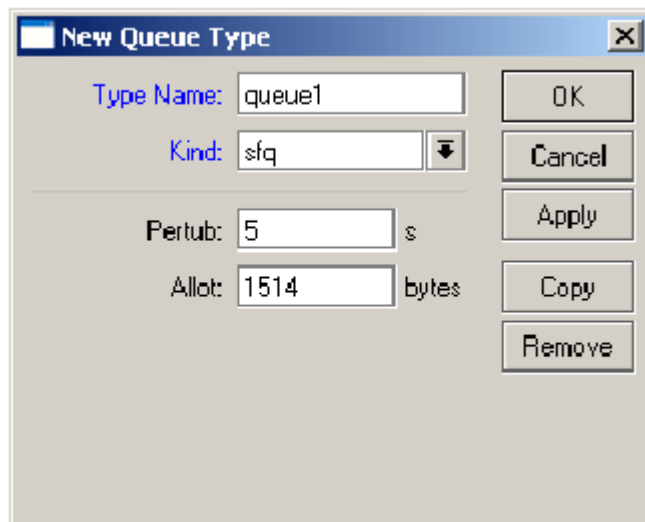


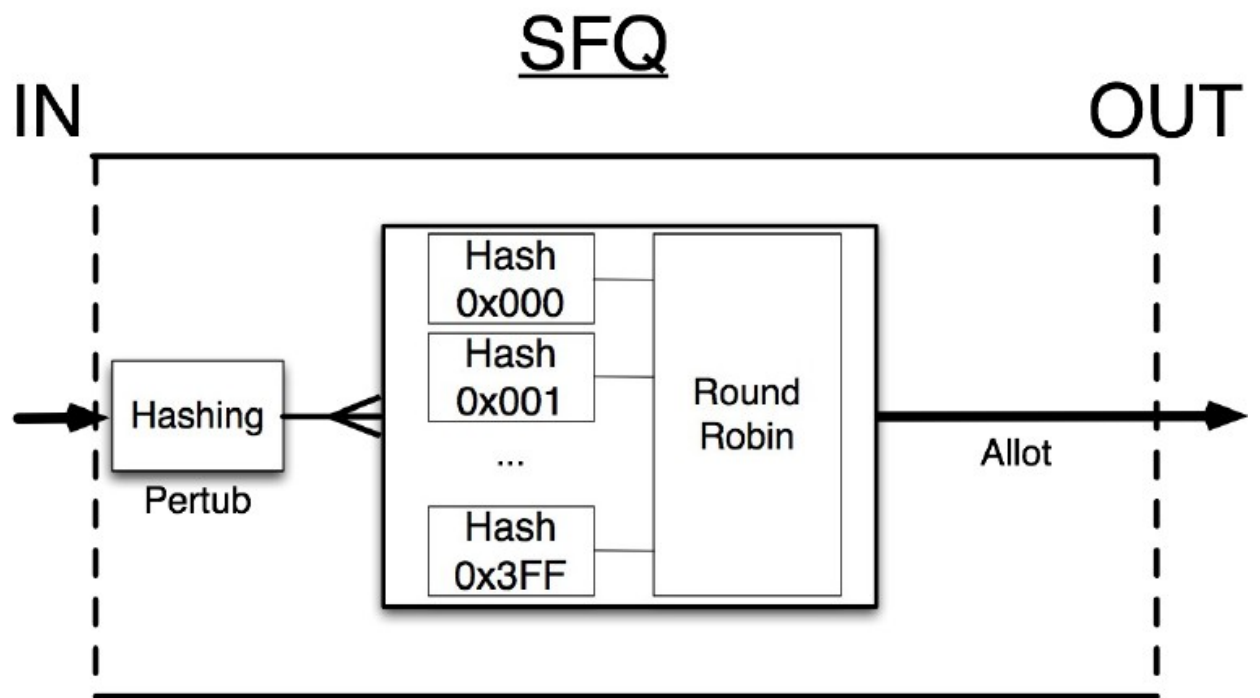
# SFQ (Stochastic Fairness Queueing)

(очередь равномерного случайного  
распределения пакетов)

## Поведение:

Базируясь на основе хэш-значения адреса источника и назначения, SFQ делит трафик между 1024 под-потоками, распределяя равное количество трафика для каждого под-потока (циклически по кругу) с помощью алгоритма Round Robin.





# Примеры SFQ

SFQ может использоваться для объединения в один под-поток аналогичных соединений (с/на один адрес).

Обычно используется для управления потоками данных от или к серверам, обеспечивая сервисом каждого клиента.

Идеально подходит для ограничений р2р трафика — можно указать строгое ограничение трафика с/на один адрес, без необходимости ограничивать и дропать количество соединений.

# PCQ

PCQ был введен для оптимизации QoS в крупных системах, где большая часть очередей одинаковы для разных под-поток.

Начиная с версии MikroTik RouterOS 5.0rc5 - PCQ имеет поддержку Burst и IPv6

New Queue Type

Type Name:

Kind:

Rate:

Limit:

Total Limit:

Burst Rate:

Burst Threshold:

Burst Time:

- Classifier

☐ Src. Address ☒ Dst. Address

☐ Src. Port ☐ Dst. Port

Src. Address Mask:

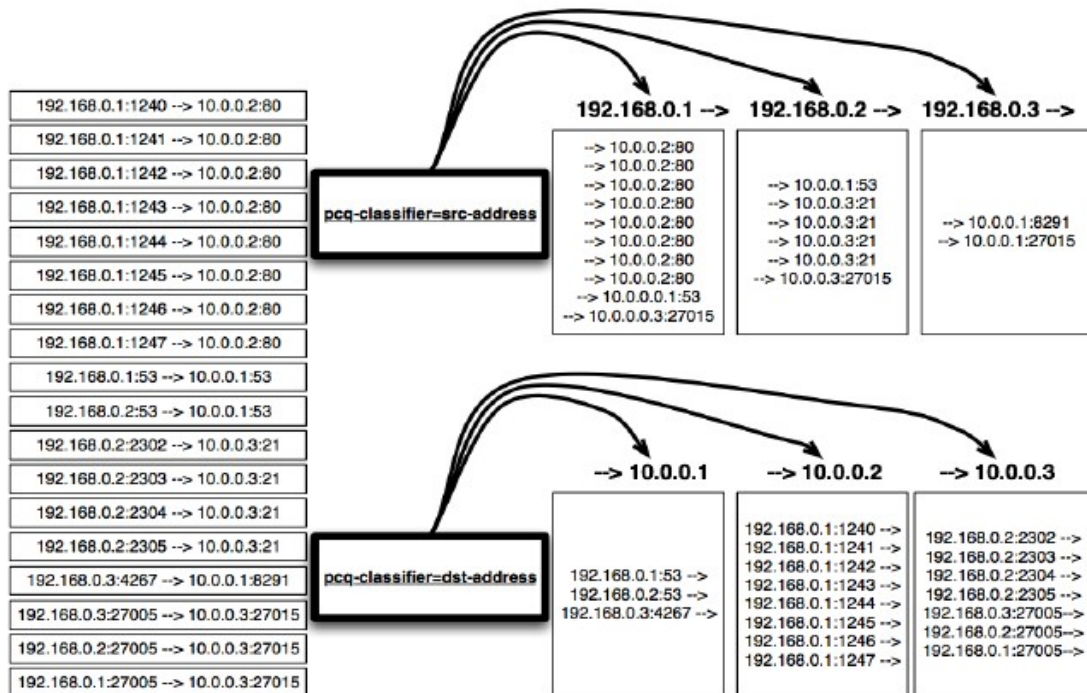
Dst. Address Mask:

Src. Address6 Mask:

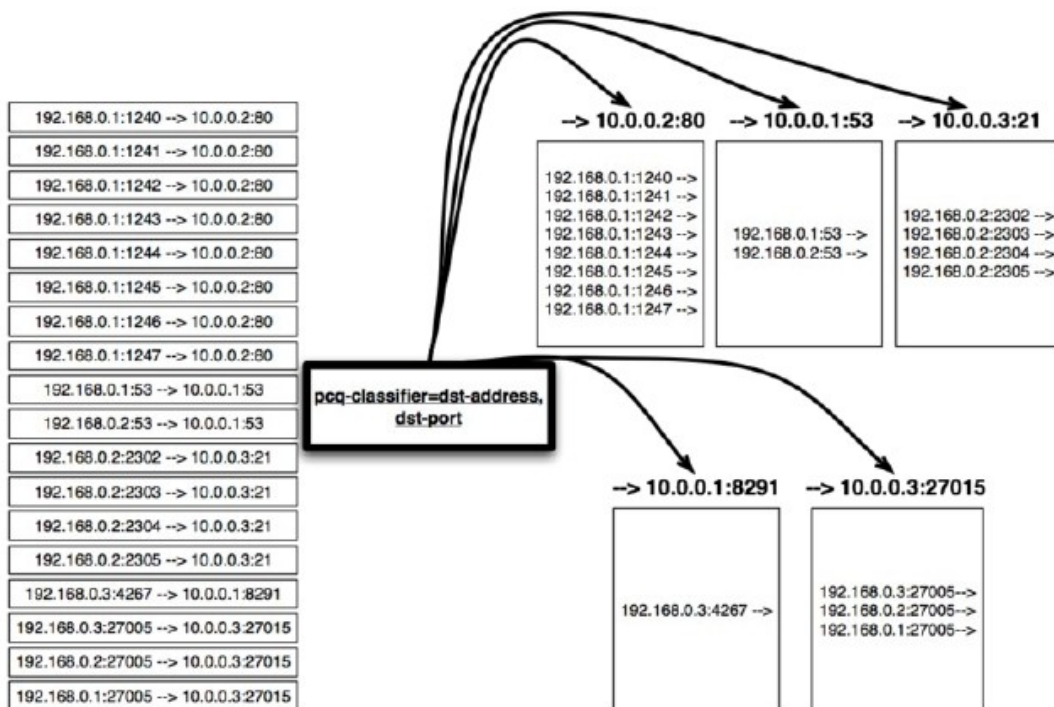
Dst. Address6 Mask:

OK Cancel Apply Copy Remove

# PCQ Classification (1)

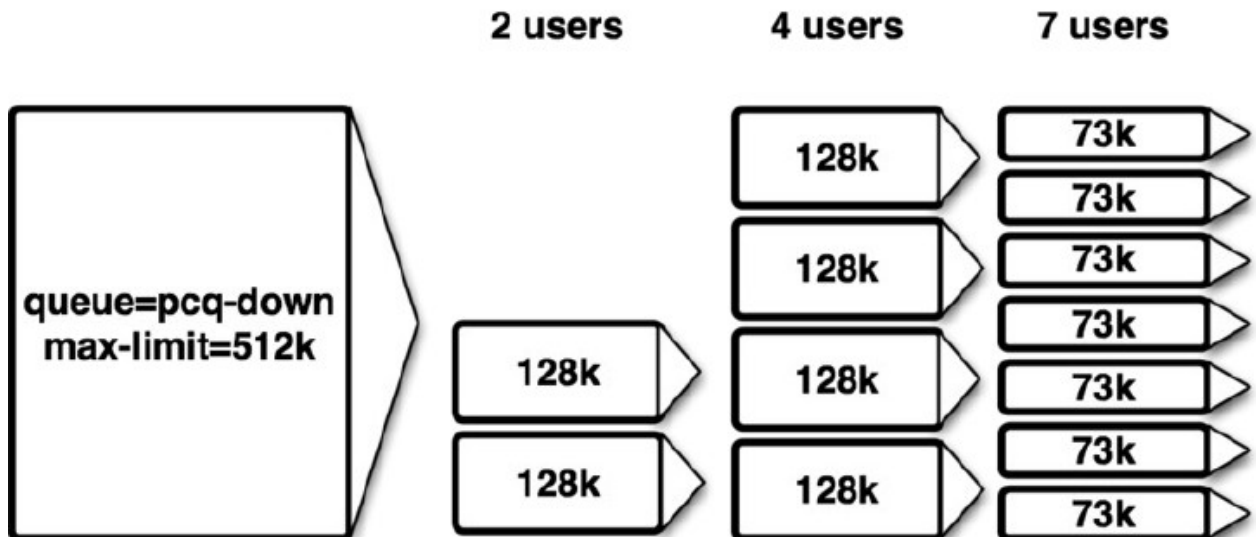


# PCQ Classification (2)



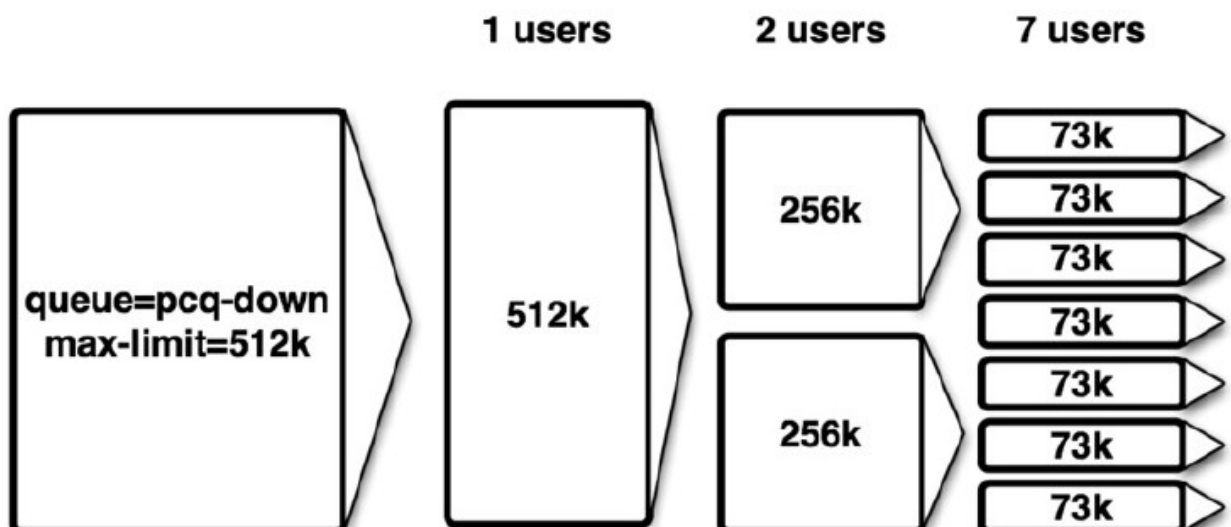
## PCQ Rate (1)

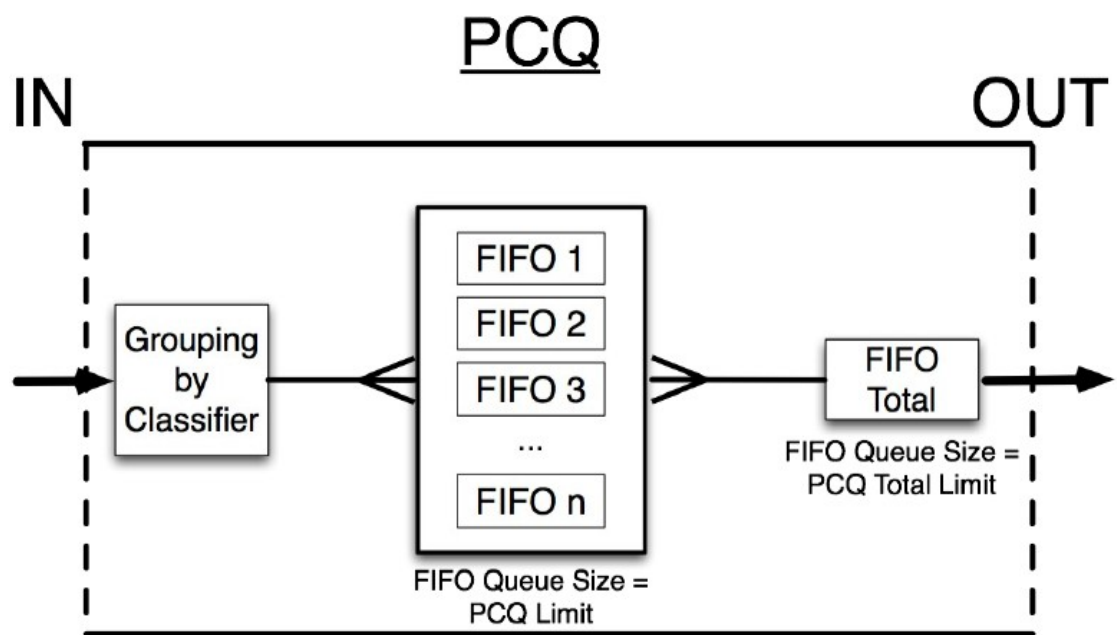
**pcq-rate=128000**



## PCQ Rate (2)

**pcq-rate=0**







# Burst

(Кратковременный всплеск (взрыв) трафика)

Функция QoS "Burst" является одним из лучших способов улучшить качество веб-сёрфинга для клиентов.

Burst позволяет обеспечивать более высокие скорости передачи данных на короткий период времени

Если средняя скорость передачи данных меньше чем Burst порог (**burst-threshold**), burst может быть использован (фактическая скорость передачи данных может достигать лимита **burst-limit**)

Средняя скорость передачи данных рассчитывается от последних **burst-time** секунд

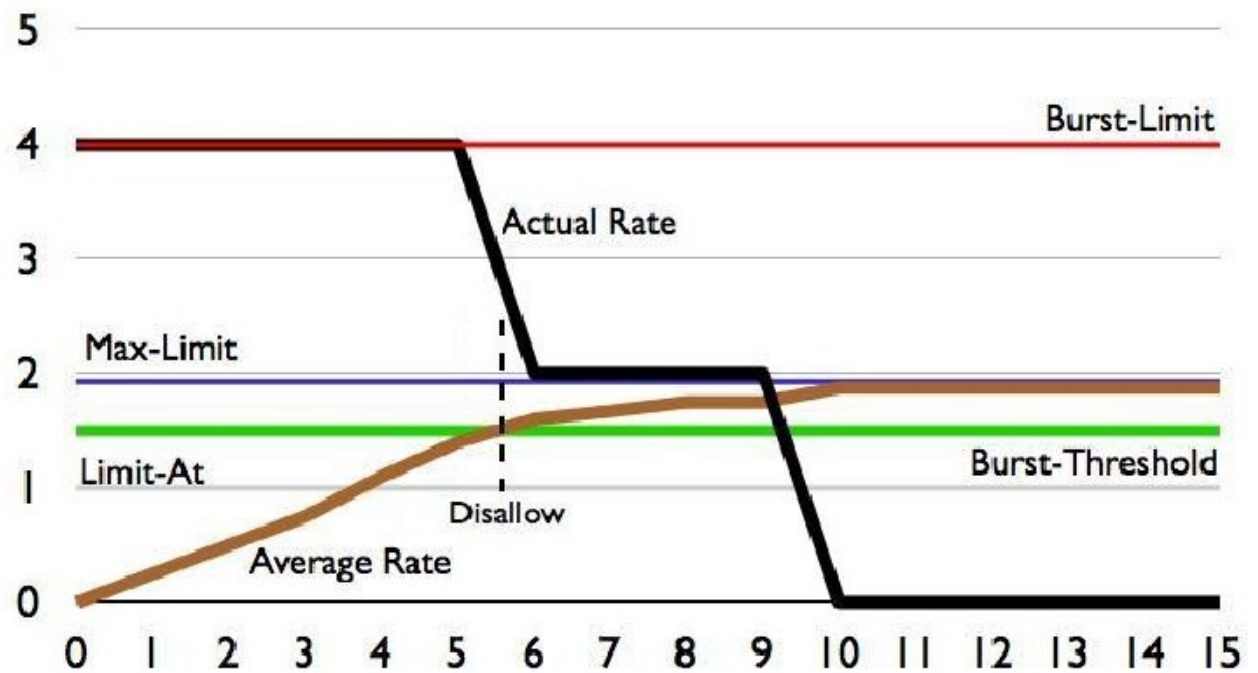
# Burst и средняя скорость передачи данных

Средняя скорость передачи данных вычисляется следующим образом:

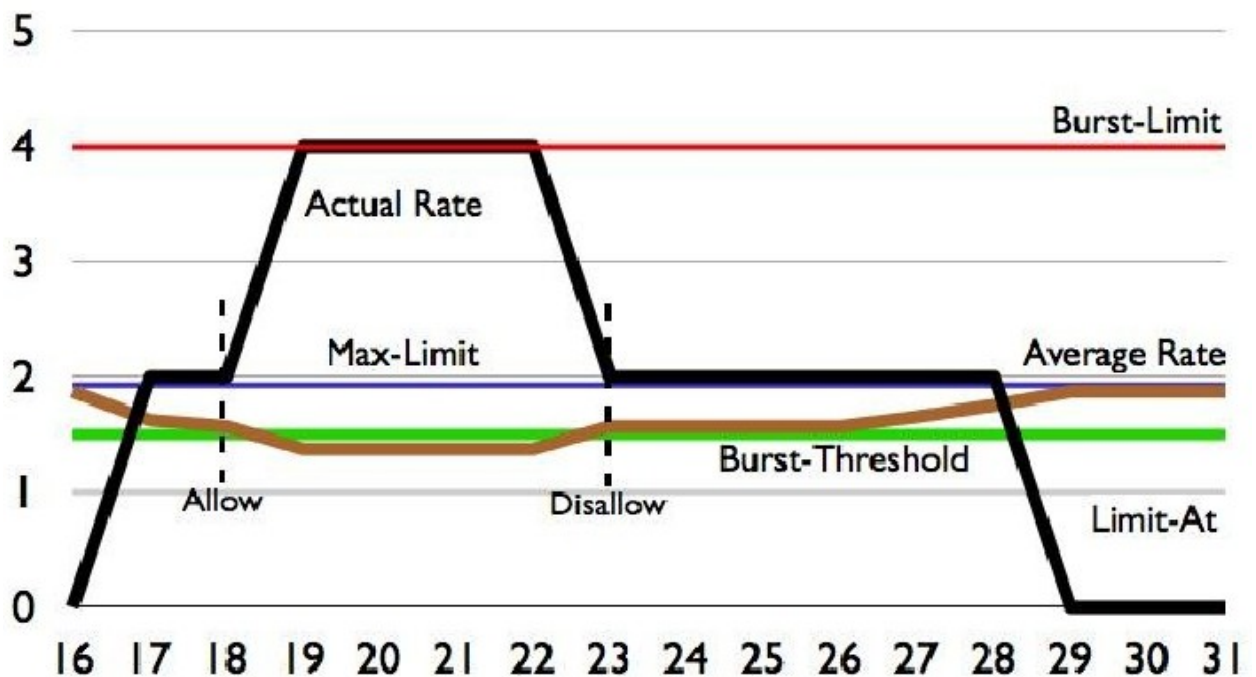
- **burst-time** делится на 16 отрезков
- роутер рассчитывает среднюю скорость передачи данных каждого класса на этих маленьких отрезках

Обратите внимание, что фактический отрезок времени burst — это не то же самое что burst-time. Он может быть в несколько раз меньше, чем burst-time, в зависимости от max-limit, burst-limit, burst-threshold и фактической истории скорости передачи данных (см. пример графика на следующем слайде)

# Burst



## Burst (Part 2)



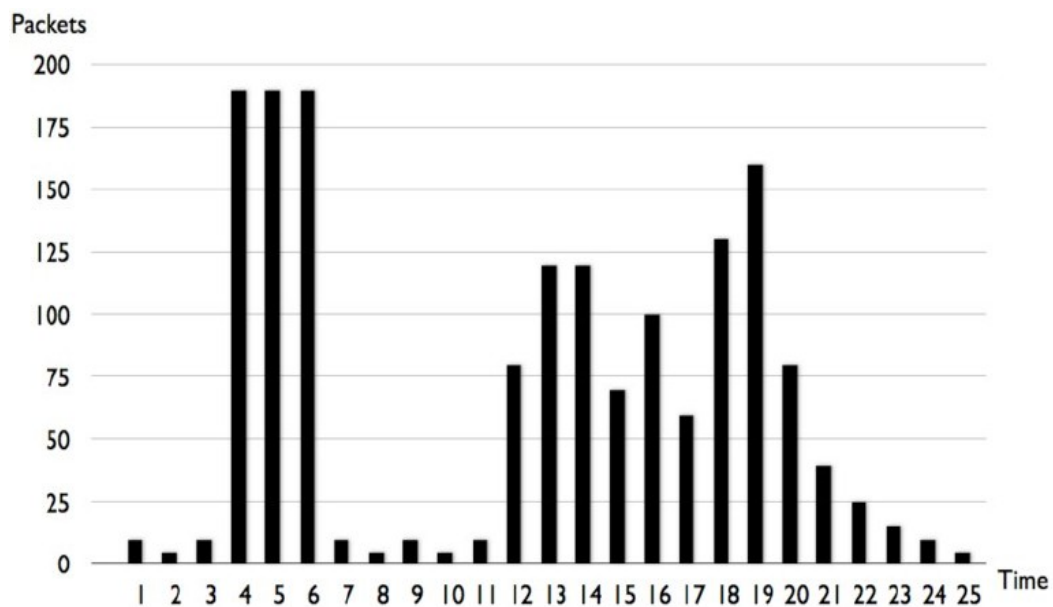
# Размер очереди (Queue Size)

Размер очереди оказывает непосредственное влияние на производительность очереди — это выбор между потерей пакетов и увеличением латентности (задержки)

В RouterOS размер очереди является общим для одного типа очереди (т.е. очередей одного типа может быть много — но размер очереди будет одинаковый — и поэтому задается в Queue Types)

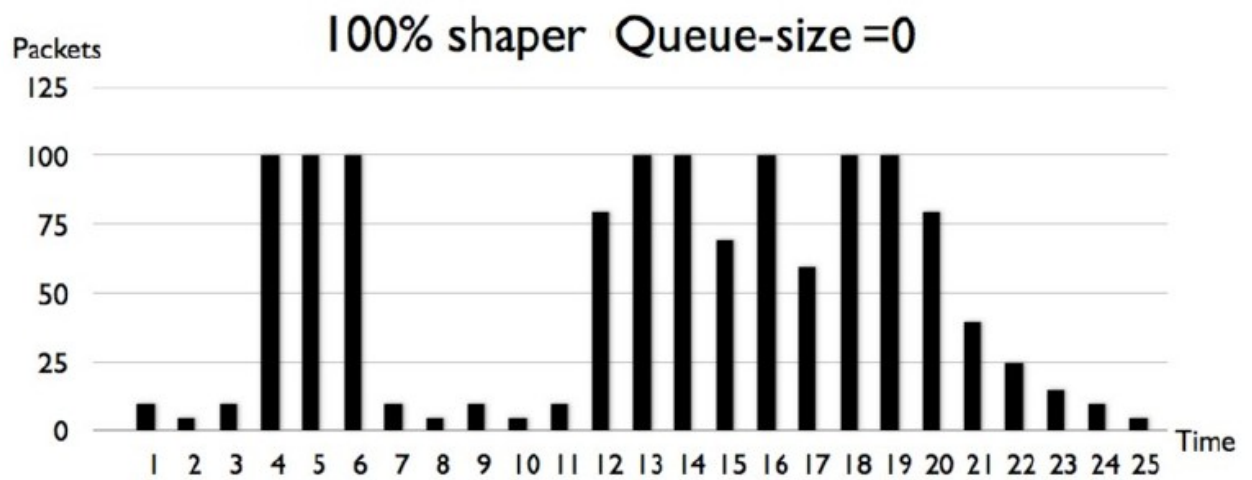
Для простоты понимания влияния размера очереди на трафик:

- не будем учитывать ретрансмит пакетов (повторную передачу)
- будем считать, что процесс, который идет непрерывно, можно разбить на шаги.



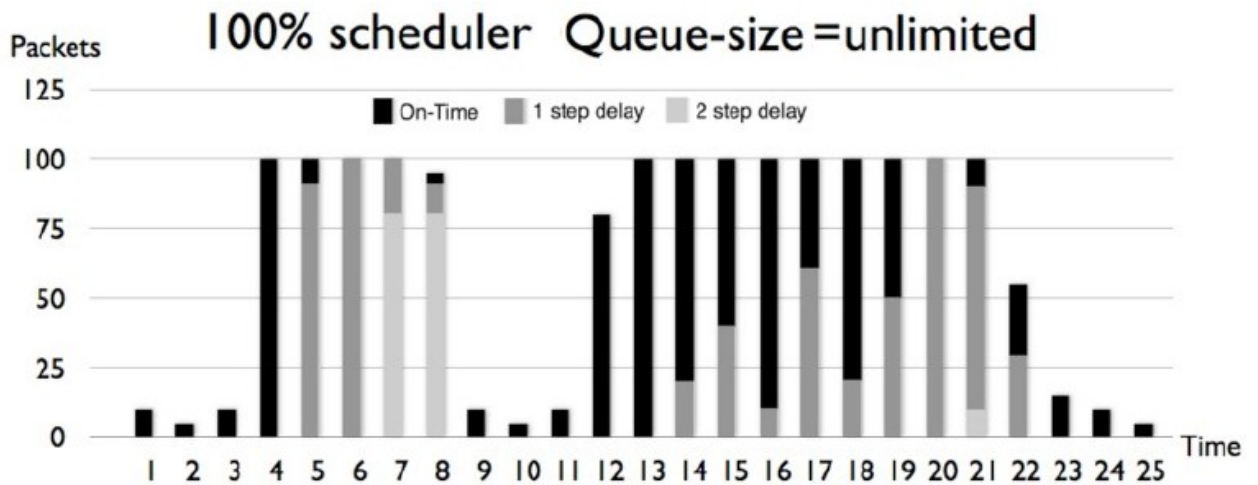
25 шагов и суммарно 1610 входящих пакетов на данном временном интервале

Размер очереди равен нулю —  
стоппроцентный шейпер.

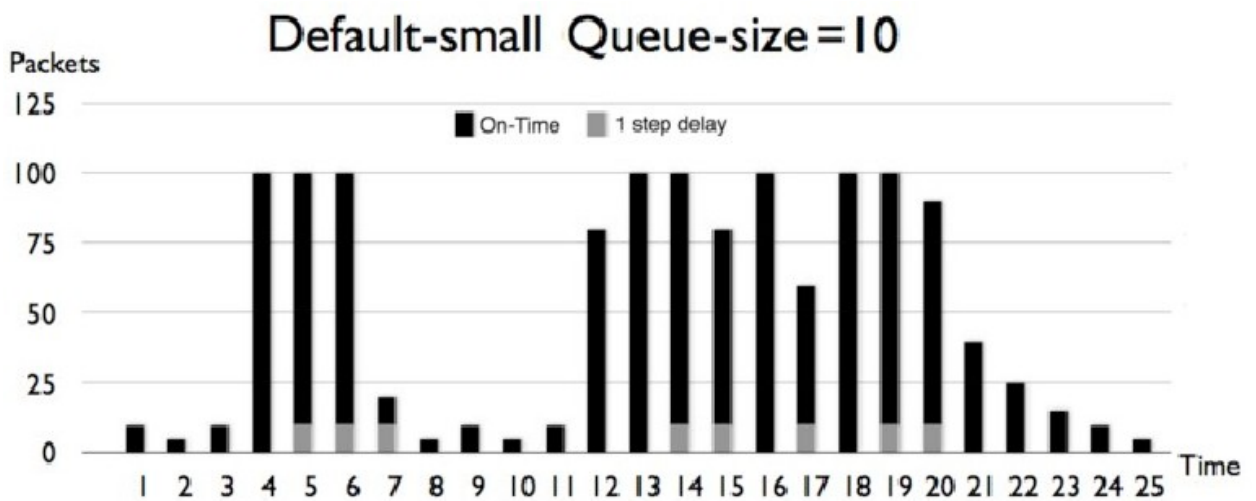


В данном случае только 1250 пакетов из 1610 прошли (22,4% пакетов отброшено), но все пакеты прошли без задержки

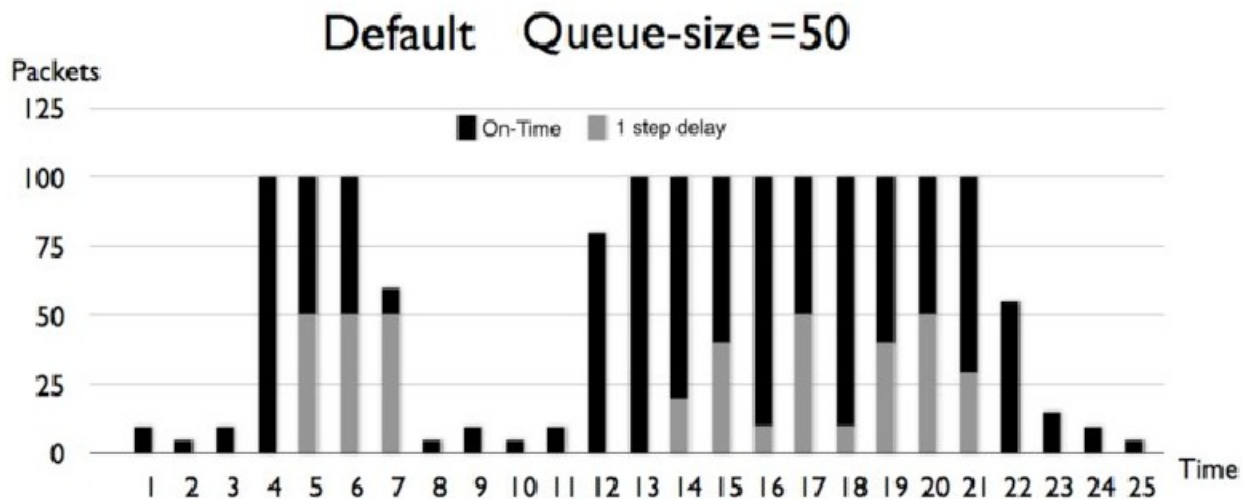
## Размер очереди равен бесконечности — стопроцентный планировщик.



Нет потерь пакетов, но 630 (39,1%) пакетов имели 1 степень задержки, и 170 (10,6%) пакетов имели 2 степени задержки (задержка = латентность)



320 (19,9%) пакетов были отброшены и 80 (5,0%) пакетов имели 1 степень задержки



190 (11,8%) пакетов были отброшены и 400 (24,8%) пакетов имели 1 степень задержки



# Простые очереди (Simple Queues)

Простые очереди идут по порядку — как правила фаервола

Пакеты 999-ой очереди будут проверены на соответствие с каждой из 998-и предыдущих очередей

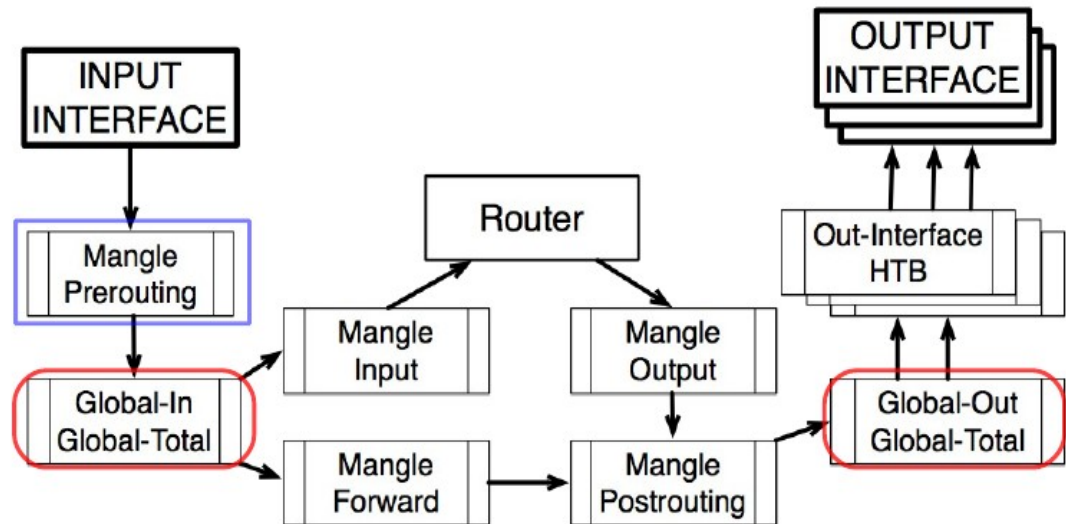
Каждая простая очередь может стоять в 3 отдельных очередях:

One in Global-in (“direct” part)

One in Global-out (“reverse” part)

One in Global-total (“total” part)

# Простые очереди и Mangle



# Queue Tree (дерево очередей)

Очереди Queue Tree бывают только однонаправленными и могут быть расположены в любом из доступных НТВ.

Очереди Queue Tree обрабатываются не по порядку – весь трафик обрабатывается одновременно.

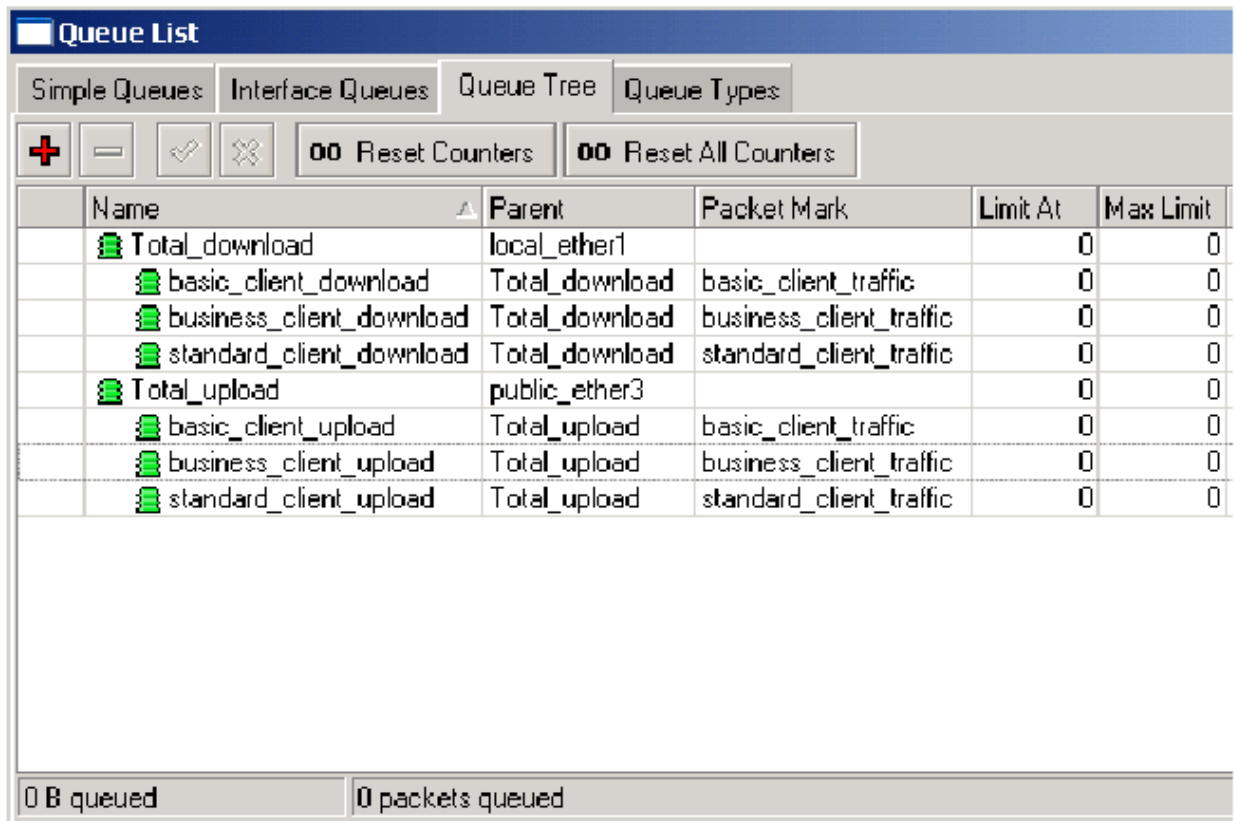
Все дочерние очереди должны иметь пакеты, маркированные с помощью средств “/ip firewall mangle”

Простая очередь (Simple queue) помещенная в тот же НТВ, будет «принимать» весь трафик от Queue Tree очереди.

# Queue Tree

## Вид в Winbox

### Queue Tree – Winbox View



The screenshot shows the 'Queue List' window in Winbox. It has tabs for 'Simple Queues', 'Interface Queues', 'Queue Tree', and 'Queue Types'. The 'Queue Tree' tab is active. Below the tabs are buttons for adding (+), deleting (-), and refreshing (circular arrows), along with 'Reset Counters' and 'Reset All Counters'. The main area contains a table with the following data:

Name	Parent	Packet Mark	Limit At	Max Limit
Total_download	local_ether1		0	0
basic_client_download	Total_download	basic_client_traffic	0	0
business_client_download	Total_download	business_client_traffic	0	0
standard_client_download	Total_download	standard_client_traffic	0	0
Total_upload	public_ether3		0	0
basic_client_upload	Total_upload	basic_client_traffic	0	0
business_client_upload	Total_upload	business_client_traffic	0	0
standard_client_upload	Total_upload	standard_client_traffic	0	0

At the bottom, there are status bars showing '0 B queued' and '0 packets queued'.

Прим. Переводчика — тут снова на слайде пример — когда в качестве родителей указывается выходной интерфейс. Я еще раз напому — что это не работает, если вы используете любой sNAT (например masquerade).

Поэтому указывайте Global-Out в качестве родителей для Total\_download и для Total\_upload и маркируйте в Mangle отдельно download и upload трафик.

Удачи в делах.