



TITANIA[®]
NETWORKS LTD

HTB Network Packet Scheduler implementation:

experimenting with the new bucket size feature

©2017 Alfredo Giordano, Matthew Ciantar
Titania Networks Limited



Alfredo Giordano

- MikroTik Certified Trainer.
- Certified consultant for MikroTik and other Brands.
- Specialized in ISP, WISP and corporate Network development.
- Working with MikroTik solutions since 2006.
- In Telecommunications since 2001.
- Active Member of RIPE and administrator for several AS
- Master degree in Electronic Engineering at Polytechnic of Turin, IT and university of Illinois of Chicago, USA.



Matthew Ciantar

- MikroTik Certified Trainer.
- Certified consultant for MikroTik – (MTCNA, MTCTCE, MTCWE, MTCRE, MTCINE, MTCIPv6E)
- Certified consultant for Cisco – (CCNA, CCNP)
- Microsoft Certified Professional - Enterprise Administrator (MCITP)
- Experience with Service Provider, and Betting Industry for providing robust and highly available infrastructures.
- Over 15 years experience with Mikrotik RouterOS and RouterBoards

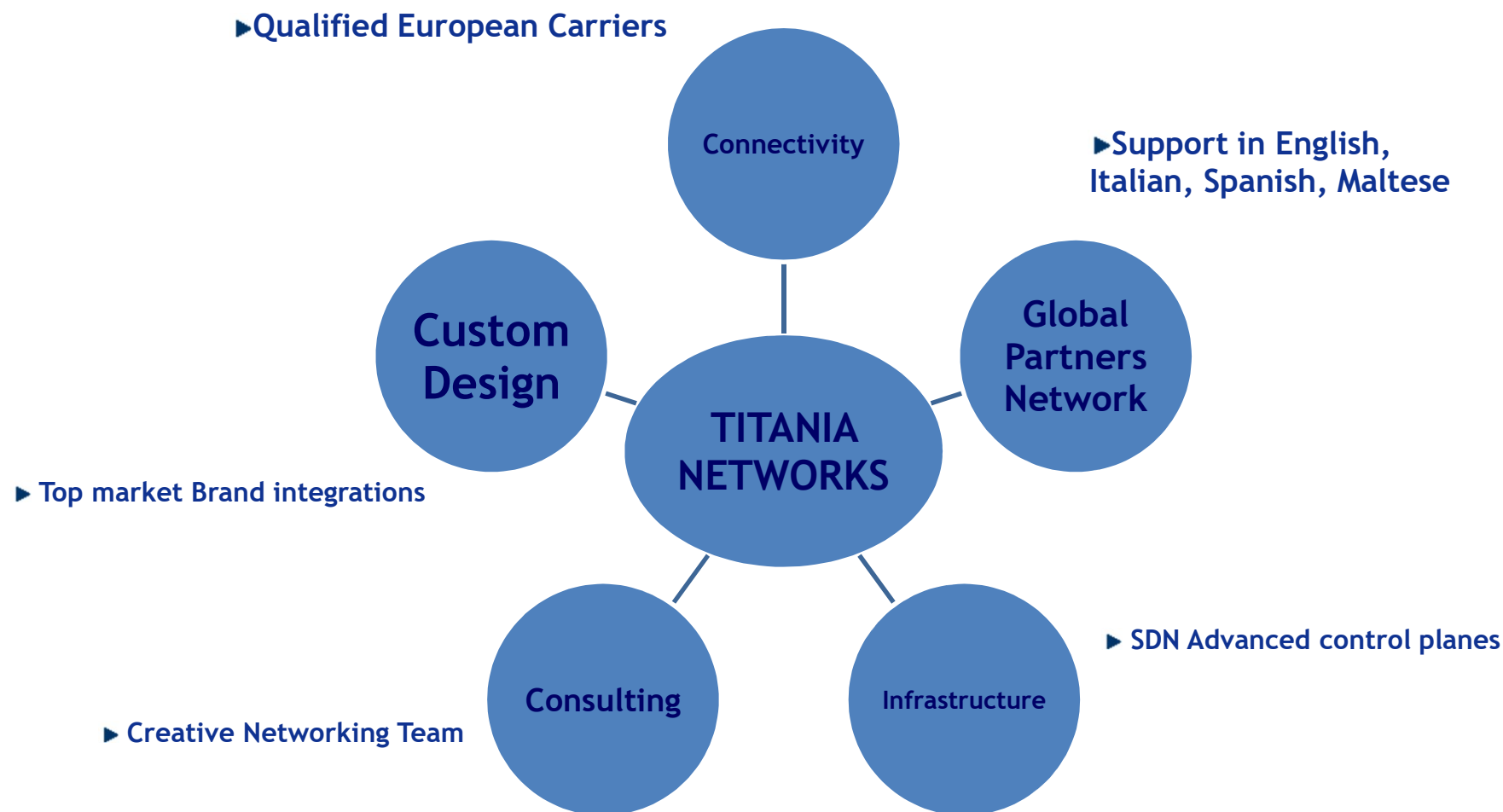
Titania Networks

- Started in 2014, by providing IT Training and Consultancy under the brand tiktrain.com
- Incorporated in 2015 as a company in Ireland started operations in Europe.
- Most requested services:
 - Mission critical Networking consulting
 - ISP Design
 - Network Training
- Operation area:
 - Europe (Ireland, Malta, Italy, the Netherlands, Spain)
 - Latin America



TITANIA
NETWORKS LTD

Operations

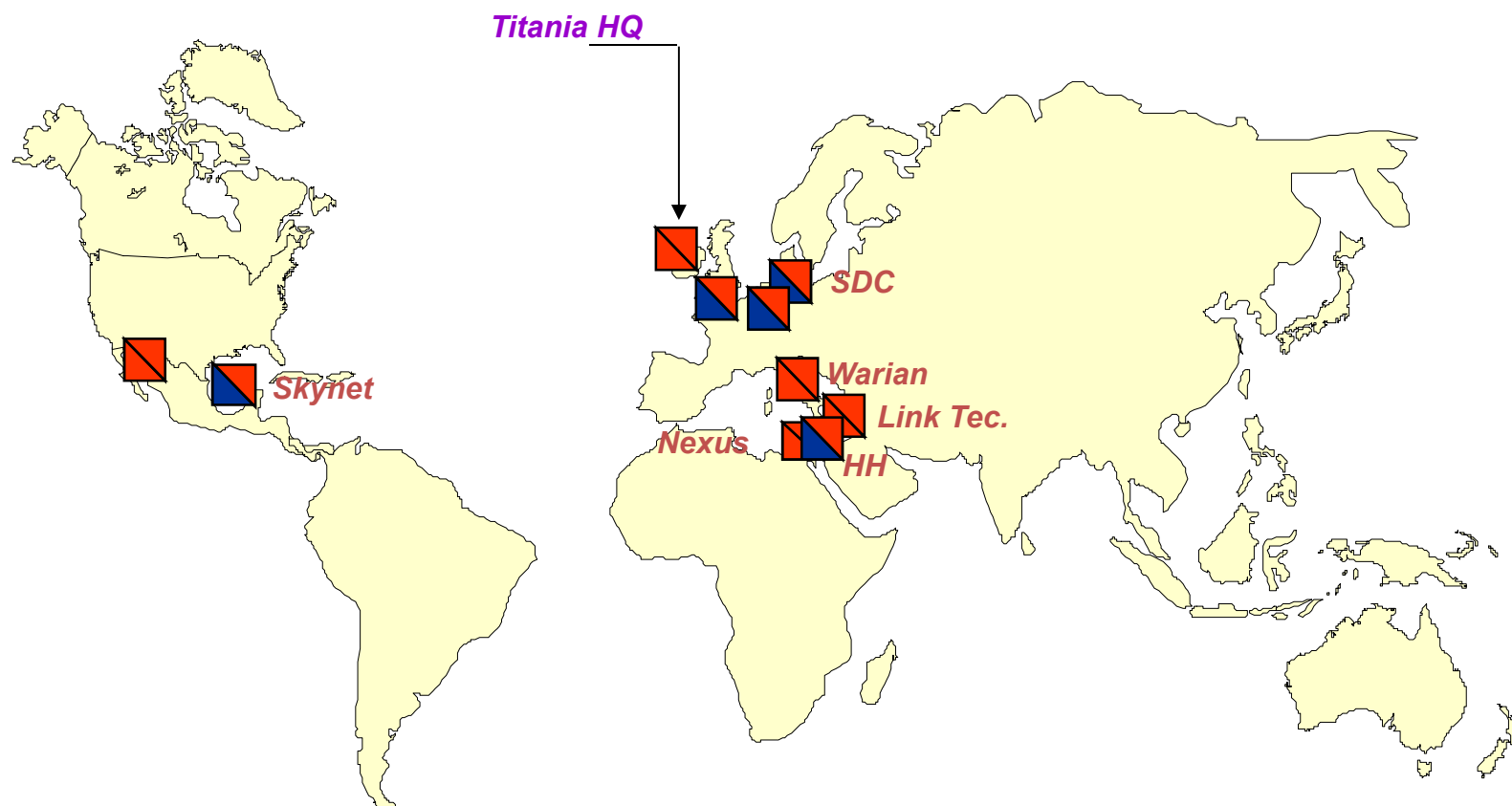


- With a single point of contact.



TITANIA[®]
NETWORKS LTD

Customers



-  **Consulting Customer**
-  **Partner with Infrastructure facility**

Goals

- With this presentation we seek to explain:
 - Concepts involving HTB
 - New features introduced
 - What you can achieve
- Trying to get a holistic picture from the available documentation



Topics

- Concepts
- The linux kernel `queue_run()`
- Queuing
 - Queue type / kind / size
- The HTB Algorithm
 - Token / Buckets
 - Classes
 - Putting everything together
- Configuration basics
- Real-time Lab demonstration



Concepts

- Scheduling
- Shaping
- Queue



The truth about Queues

- Queues are located between the system and the interface and determine how data is SENT from the interface itself.
- Queues can be used to buffer the excess of output bandwidth to prevent packet loss in case of bursts – and this is generally GOOD
- TCP/IP, because of the way it works, will try to fill any queue you offer it. Queues create latency that affects interactivity for example when your keystroke must traverse a long queue – and this is generally BAD



Proof of Concept

admin@192.168.88.1 (SOURCE) - WinBox v6.37.5 on hAP lite (smips)

Session Settings Dashboard

Safe Mode Session: 192.168.88.1 CPU: 38%

Quick Start (Running)

Test ID: 0 Start Stop Close New Window

Stream: Port: Interface: Packet Size: PPS: 15 MBPS: Tx Template:

Seq	ID	Tx Packets	Tx Rate	Rx Packets	Rx Rate	Lost Packets	Lost R
55	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
56	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
57	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
58	0	27 985	14.9 Mbps	100	76.0 kbps	27 885	14.9
59	0	27 987	15.0 Mbps	100	76.0 kbps	27 887	14.9
60	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
61	0	27 985	14.9 Mbps	100	76.0 kbps	27 885	14.9
62	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
63	0	27 984	14.9 Mbps	100	76.0 kbps	27 884	14.9
64	0	27 988	15.0 Mbps	100	76.0 kbps	27 888	14.9
65	0	27 985	14.9 Mbps	100	76.0 kbps	27 885	14.9
66	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
67	0	27 987	15.0 Mbps	100	76.0 kbps	27 887	14.9
68	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
69	0	27 985	14.9 Mbps	100	76.0 kbps	27 885	14.9
70	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
71	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
72	0	27 985	14.9 Mbps	100	76.0 kbps	27 885	14.9
73	0	27 986	15.0 Mbps	100	76.0 kbps	27 886	14.9
TOT	0	2 042 835	14.9 Mbps	7 353	76.5 kbps	2 035 482	14.9

RouterOS WinBox

admin@6C:3B:6B:1A:7C:5A (TARGET) via 192.168.88.1 - WinBox v6.37.5 on hAP lite (smips)

Session Settings Dashboard

Safe Mode Session: 6C:3B:6B:1A:7C:5A

Interface List

Interface	Name	Type	Actual MTU	L2 MTU	Tx	Rx
R	ether1	Ethernet	1500	1598	1792 bps	0 bps
R	ether2	Ethernet	1500	1598	0 bps	0 bps
R	ether3	Ethernet	1500	1598	65.5 kbps	10.4 kbps
R	ether4	Ethernet	1500	1598	79.9 kbps	16.0 Mbps
X	wlan1	Wireless ...	1500	1600	0 bps	0 bps

Simple Queue <simple-queue>

General Advanced Statistics Traffic Total Total Statistics

Target Upload Target Download

Rate: 5.4 Mbps 63.7 kbps

Packet Rate: 12 840 p/s 98 p/s

Upload: 5.4 Mbps Download: 63.7 kbps

Upload Packets: 12 840 p/s Download Packets: 98 p/s

RouterOS WinBox



The Linux Kernel

NK Installation of HTB on Slac... X +

lartc.vger.kernel.narkive.com/liqTFums/installation-of-htb-on-slack-v8-0

NARKIVE
MAILINGLIST ARCHIVE

lartc@vger.kernel.org

Search...

Discussion:
Installation of HTB on Slack v8.0 (too old to reply)

Grzegorz Skrzypczak 15 years ago

Hi All,

I'm looking for something to give me possibility to manage the bandwidth inside my FastEthernet LAN. I found HTB code but I don't know how to install it on my Slack v8.0 with kernel version 2.4.18 (without any QoS functionality compiled in to kernel). Right now my network is running without any bandwidth limitations. Could any one help me to understand the idea of HTB's class hierarchy? What external pckeges should I download and install? Do I really need "iproute2"? How to compile HTB code (where is the source code to compile?)?

Hope for helping me

Best regards,
Grzegorz "Greg" Skrzypczak

Alfredo Giordano 15 years ago

Try following the following step:

- 1 Install the kernel source package
- 2 Read all the Doc about how compile the kernel
- 3 download the HTB patch code and apply it to your kernel source tree
- 4 download the patched tc binary and replace your original tc
- 5 compile the kernel with the QoS and HTB options compiled in the kernel (not as module)
- 6 read all the lartc howto

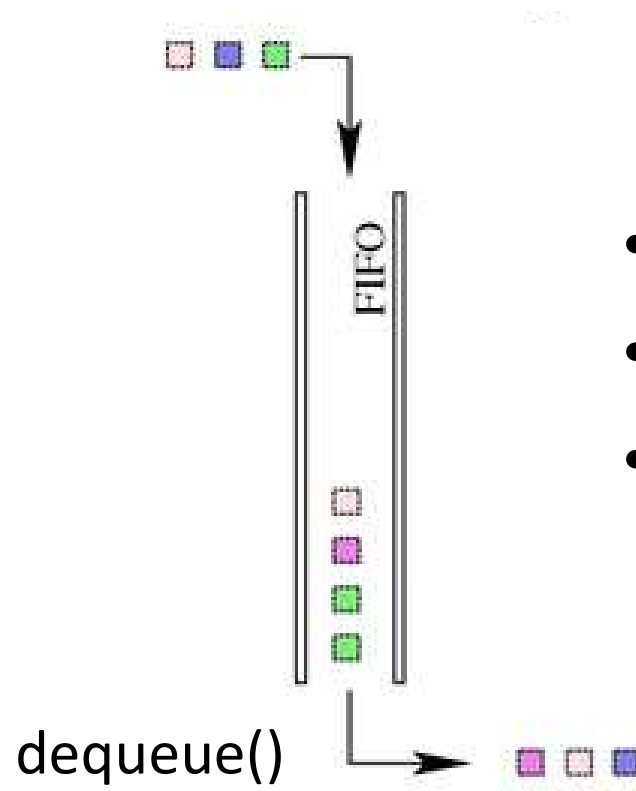
you should get trough

cheers
alfredo

The Linux Kernel

- The forwarding stack or the local process sends data to the kernel.
- Kernel enqueues data to the *queue-type* selected for the queue and immediately tries to run the queue to the hardware using `queue_run()`
- The function will call `dequeue()` according to the *queue-kind* algorithm to send to hardware (provided hardware can take as much).

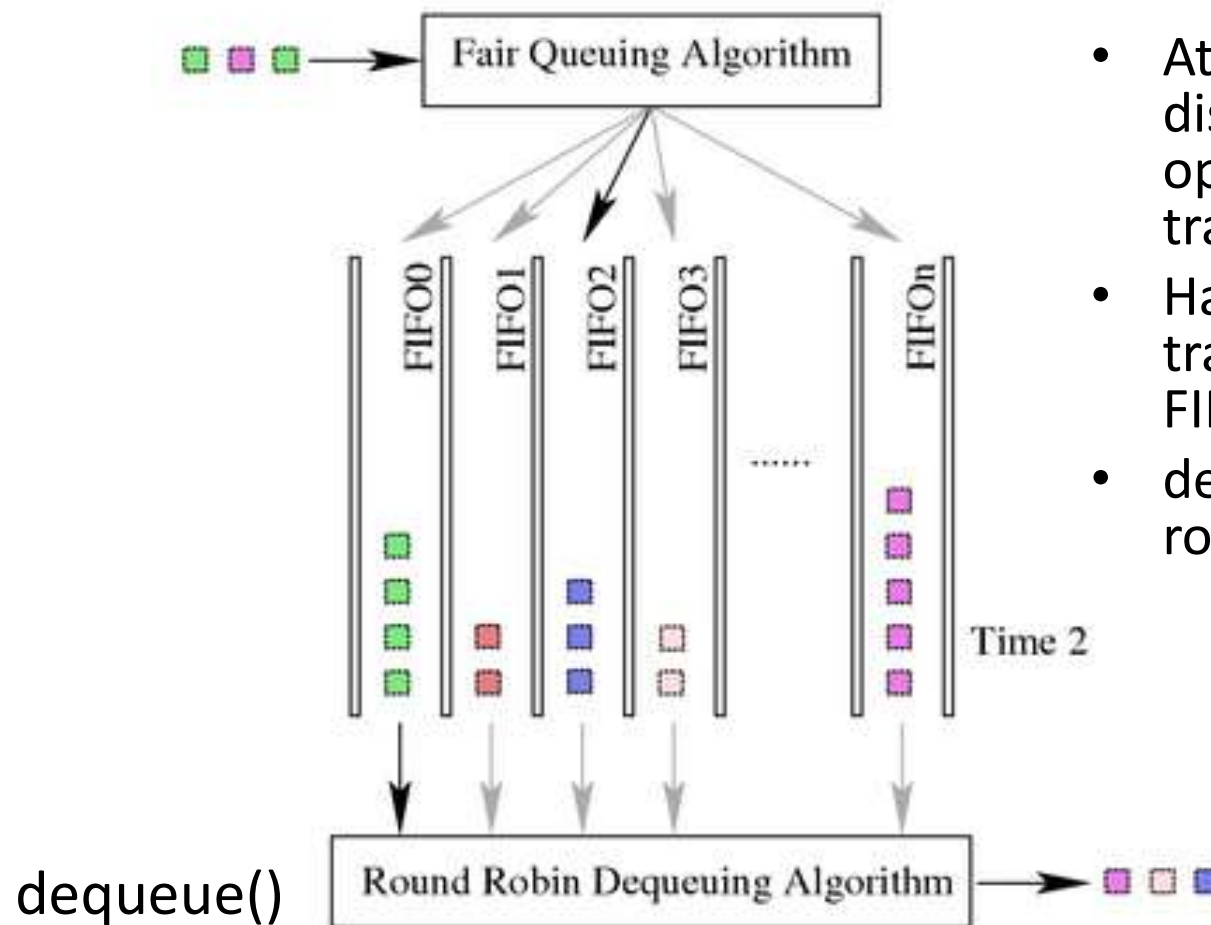
PFIFO



- Simple Buffer
- Defined by *queue-size*
- Can be bytes or packets



SFQ



- Attempt to distribute opportunity to transmit fairly
- Hash function to fit traffic in separate FIFOs
- dequeue() with round robin

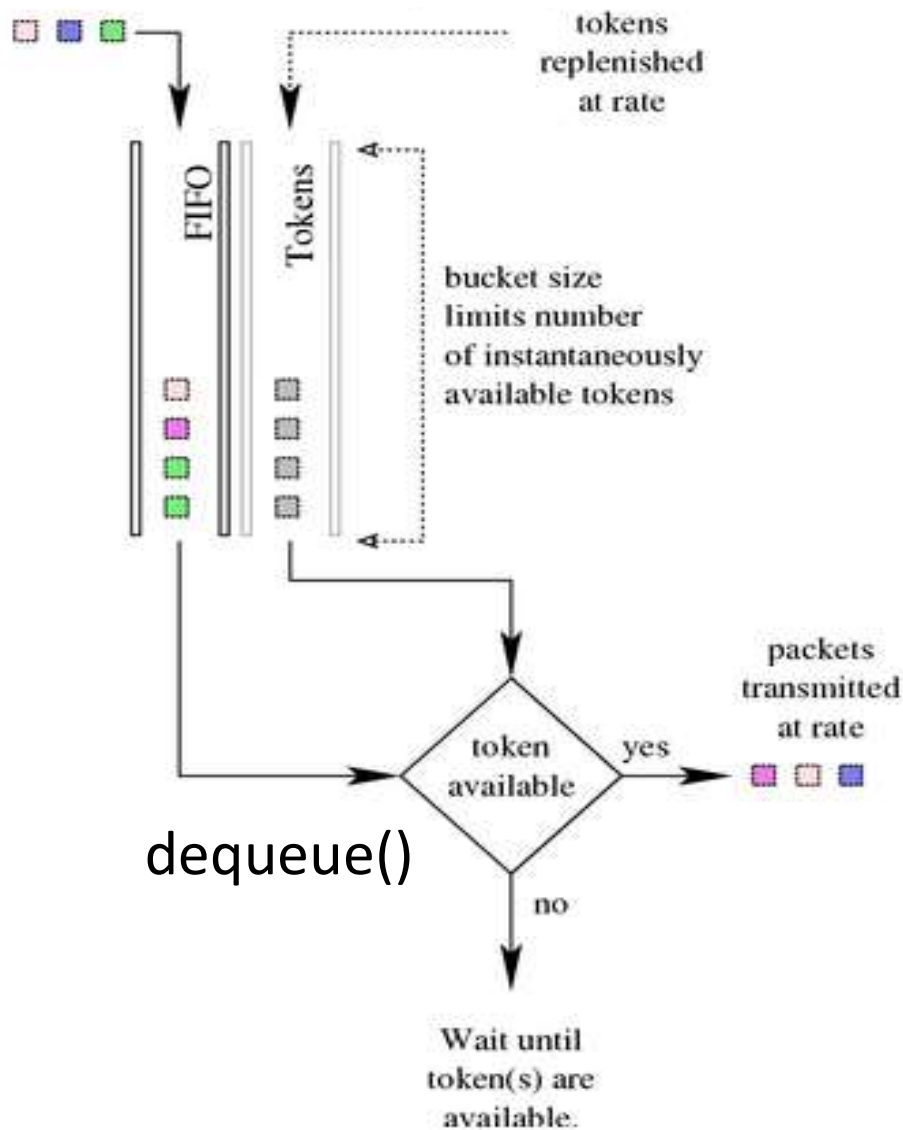


Tokens

- Control the rate of dequeuing counting the number of packets/bytes dequeued is complex and timers dependent.
- Instead of calculating the current usage, one method, used widely in traffic control, is to generate tokens at a desired rate, and only dequeue packets or bytes if a token is available.



Simple TBF



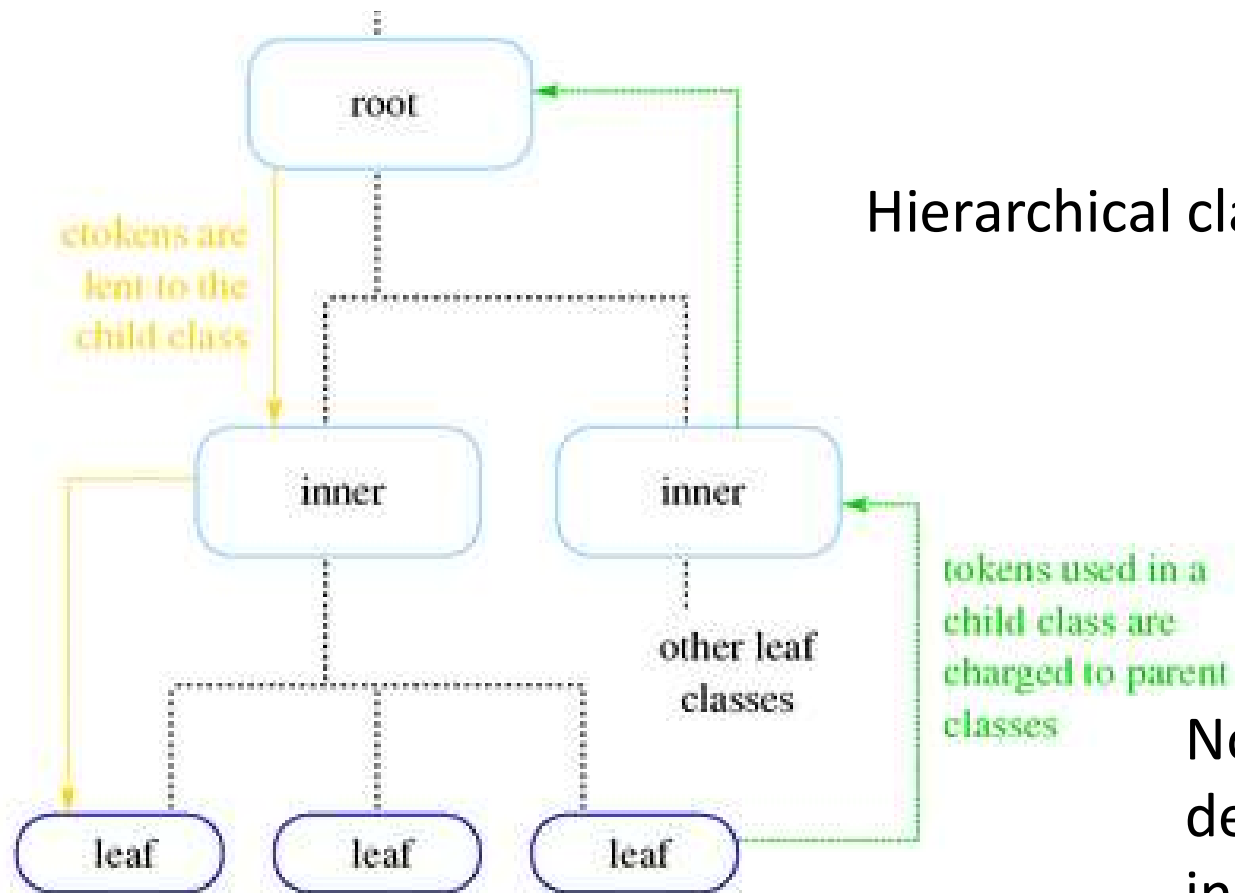
- Built on tokens and buckets
- Packets are only transmitted if there are sufficient tokens available.
- Otherwise, packets are deferred.
- It will introduce an artificial latency



Buckets

- In the case that a queue does not need tokens immediately, the tokens are collected until they are needed.
- The number of unused tokens that can be stored depends on the size of the bucket.
- A queue that has tokens available can initially dequeue a larger number of packets or bytes before tokens are depleted.

HTB - Classes



Note: HTB will not delay packets at inner level only leaf

dequeue()



HTB - Classes

- Children classes borrow tokens from their parents once they have exceeded *limit-at*.
- A child class will continue to attempt to borrow until it reaches *max-limit*
- It will then begin to queue packets for transmission until more tokens are available.

type	rate	kernel action
child	< limit-at	dequeue() based on all available tokens
child	limit-at < rate < max-limit	dequeue() try to borrow tokens from parent
child	> max-limit	delay packets
parent	< limit-at	lend tokens to children
parent	limit-at < rate < max-limit	try to borrow from parent if any, lend to children
parent	> max-limit	no borrow, no lend



HTB - Burst

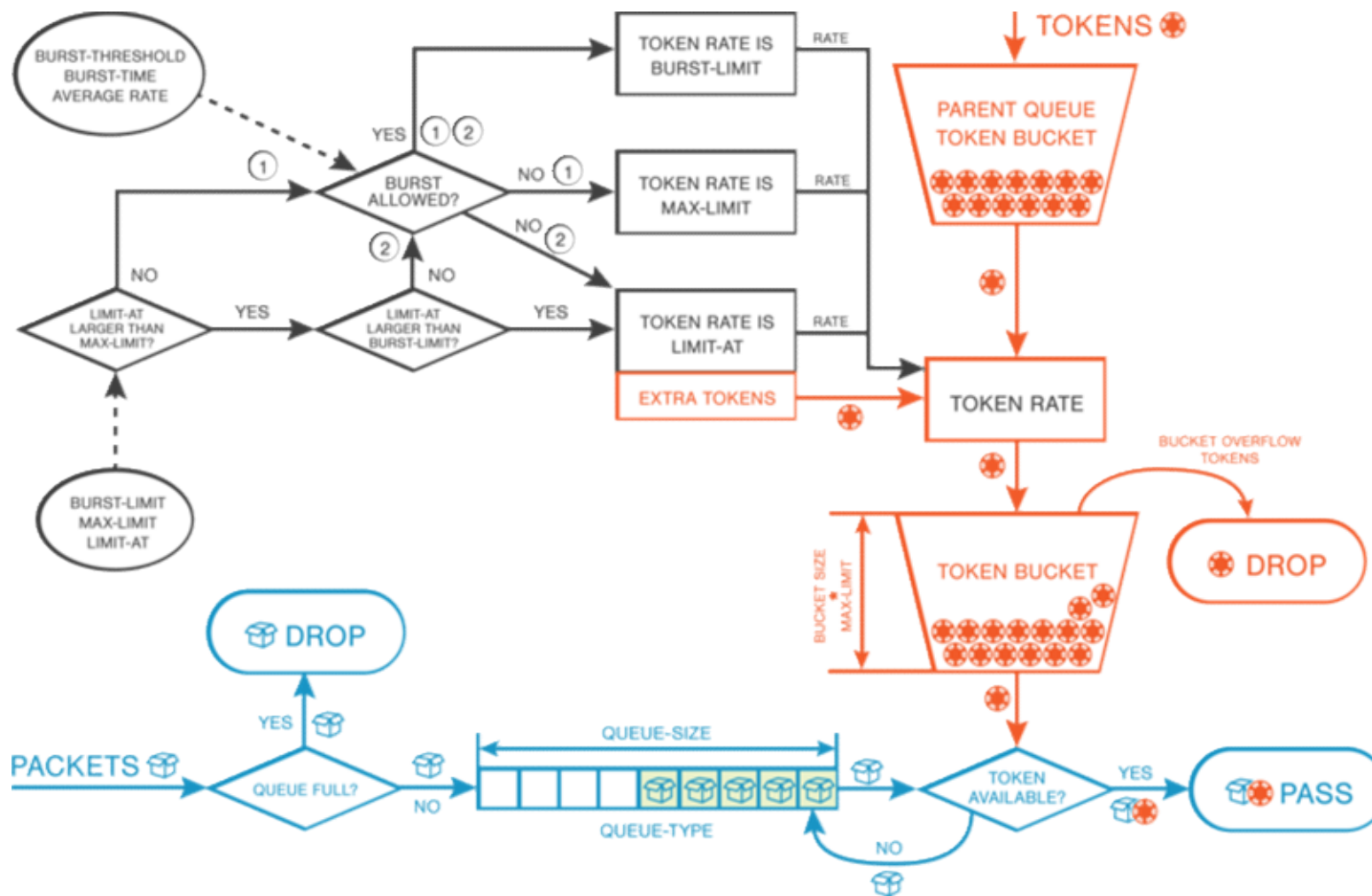
- Burst is a feature that allows to satisfy queue requirement for additional bandwidth even if required rate is bigger than *max-limit* for a limited period of time.
- Burst can occur only if average-rate of the queue for the last *burst-time* seconds is smaller than *burst-threshold*.
- Burst mechanism is simple - if burst is allowed queue will receive tokens at *burst-limit* rate. When burst is disallowed queue will receive tokens at *max-limit* rate.

HTB – Wrap up

- At the end of the day the amount of readily available tokens in the child class will define its behavior.
- We can set how fast the token replenish with the *max-limit* or *burst-limit*.
- Until RouterOS version 6.35 bucket size was hardcoded to $\text{max-limit}/10$. This is why default value is set to 0.1.
- **Now it will accept values from 0 to 10**



The New HTB Diagram





How much traffic will pass Unrestricted?

This is calculated as follows:

Bucket Capacity = *bucket-size* * *max-limit* (or *burst-limit*, if *burst-limit* is being used)

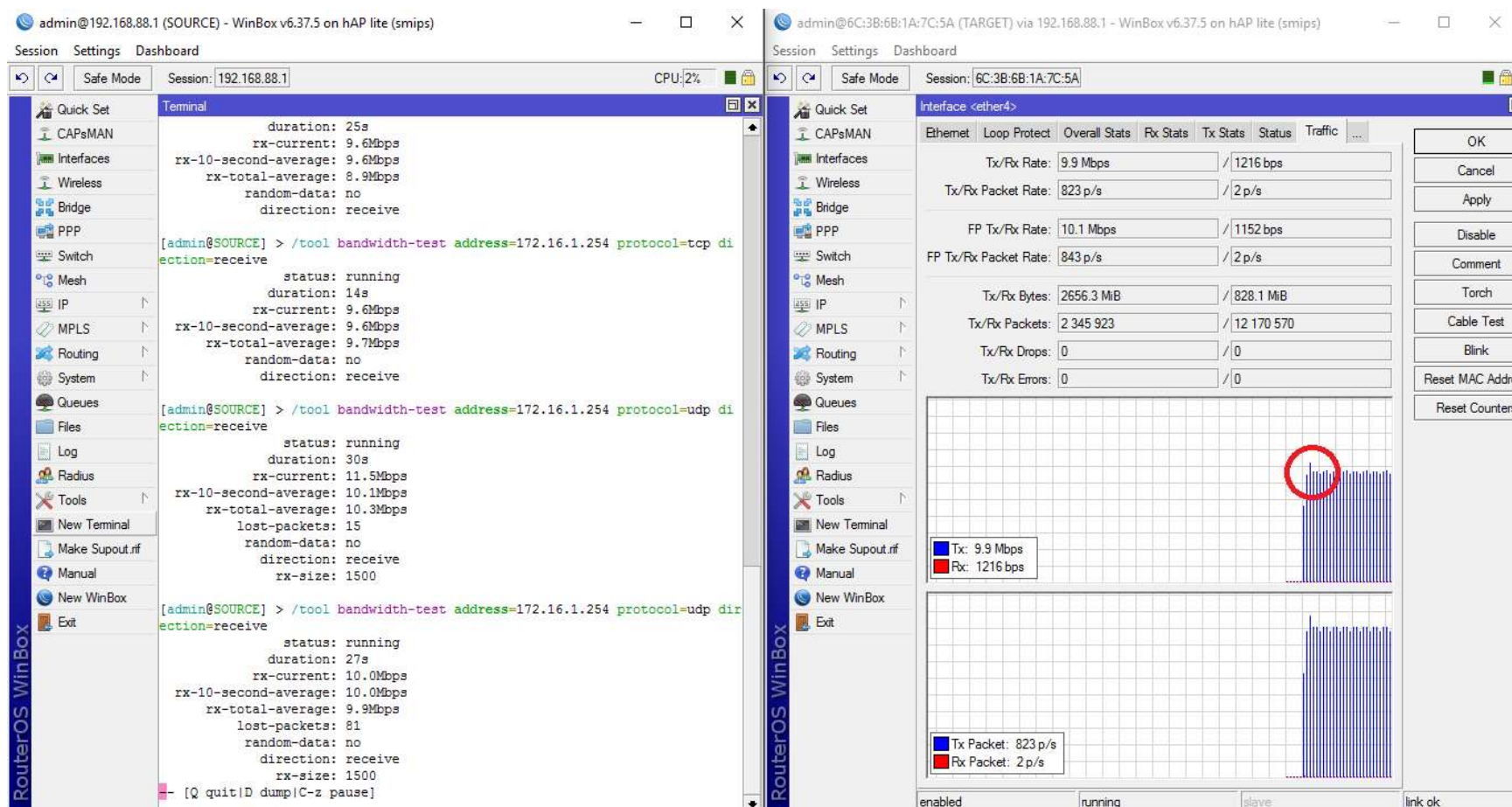
Default Bucket with 10M *max-limit*

$$0.1 * 10M = 1M$$

This will allow 1M of Data (Not bandwidth!) to go at
unrestricted speed!



Default Bucket Size





How much traffic will pass Unrestricted?

Large Bucket with 10M *max-limit*

$$10 * 10M = 100M$$

This will allow 100M of data to go at
unrestricted speed!!!



Large Bucket Size

The screenshot displays two Mikrotik WinBox windows. The left window, titled 'admin@192.168.88.1 (SOURCE) - WinBox v6.37.5 on hAP lite (smips)', shows the 'Terminal' tab with the output of the command `/tool bandwidth-test address=172.16.1.254 protocol=udp direction=receive`. The output indicates a status of 'running', a duration of 37s, and a current rate of 10.0Mbps. The right window, titled 'admin@6C:3B:6B:1A:7C:5A (TARGET) via 192.168.88.1 - WinBox v6.37.5 on hAP lite (smips)', shows the 'Interface <ether4>' tab. It displays statistics for the Ethernet interface, including a Tx/Rx Rate of 9.9 Mbps / 656 bps and a Tx/Rx Packet Rate of 822 p/s / 1 p/s. A red circle highlights a sharp spike in the traffic graph, corresponding to the test results.

Terminal Output (Left Window):

```
MikroTik RouterOS 6.37.5 (c) 1999-2017 http://www.mikrotik.com/

[?] Gives the list of available commands
command [?] Gives help on the command and list of arguments

[Tab] Completes the command/word. If the input is ambiguous,
a second [Tab] gives possible options

/ Move up to base level
.. Move up one level
/command Use command at the base level
[admin@SOURCE] > /tool bandwidth-test address=172.16.1.254 protocol=udp dir
ection=receive
status: running
duration: 37s
rx-current: 10.0Mbps
rx-10-second-average: 10.0Mbps
rx-total-average: 12.6Mbps
lost-packets: 84
random-data: no
direction: receive
rx-size: 1500
[Q quit|D dump|C-z pause]
```

Interface Statistics (Right Window):

Interface	Loop Protect	Overall Stats	Rx Stats	Tx Stats	Status	Traffic
Ethernet						
Tx/Rx Rate:		9.9 Mbps		656 bps		
Tx/Rx Packet Rate:		822 p/s		1 p/s		
FP Tx/Rx Rate:		10.1 Mbps		1776 bps		
FP Tx/Rx Packet Rate:		839 p/s		3 p/s		
Tx/Rx Bytes:		2930.1 MiB		828.7 MiB		
Tx/Rx Packets:		2 535 527		12 179 155		
Tx/Rx Drops:		0		0		
Tx/Rx Errors:		0		0		



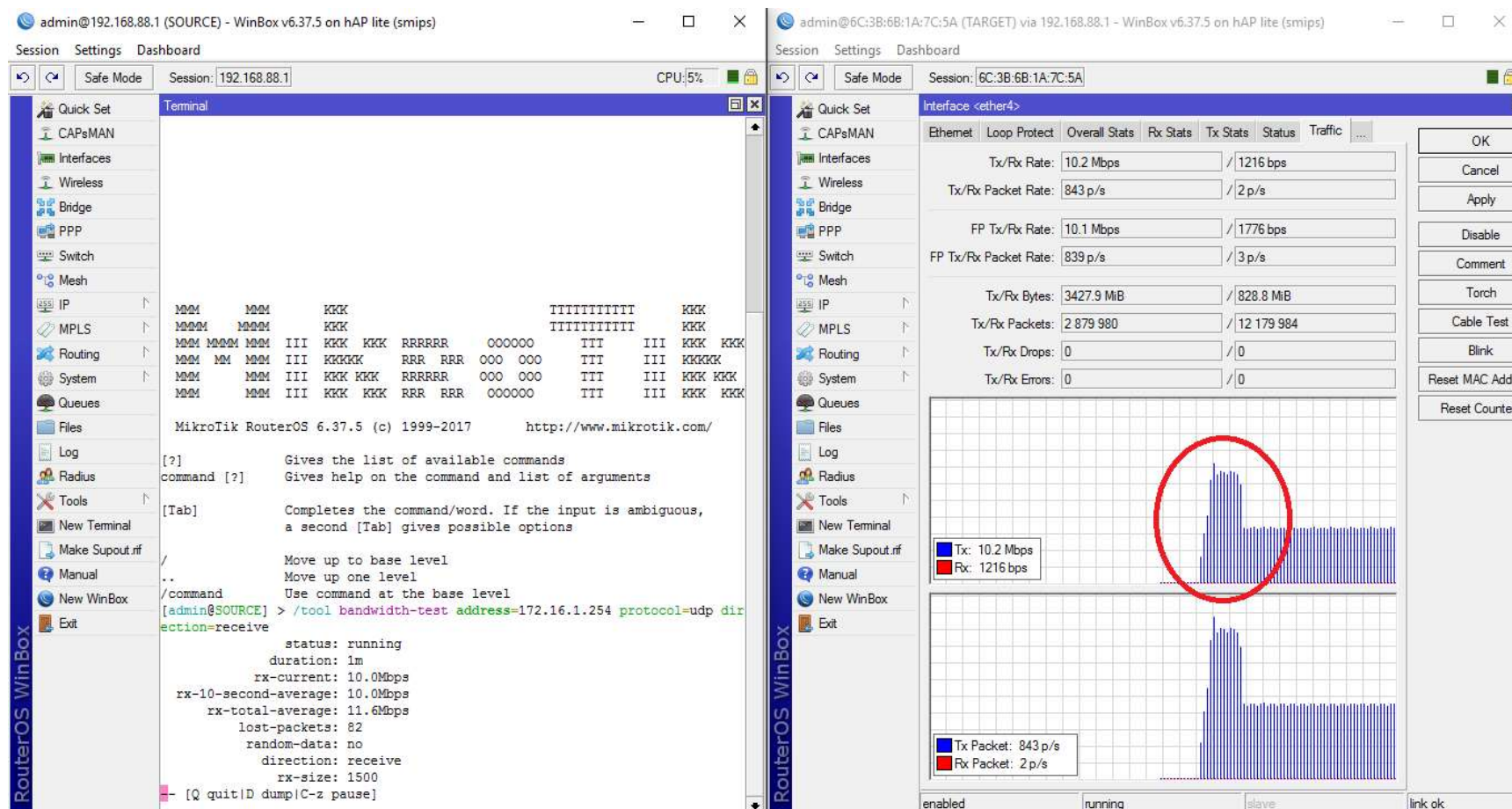
What if one wants a ceiling for the burst?

Bucket size work like burst but without a burst-limit. To be able to force a ceiling, we can set a max-limit on the parent queue.

Small Bucket with 20M *max-limit* on the parent
with a child having a Large Bucket with 10M
max-limit.



Burst with Ceiling



Burst with Ceiling

Large Bucket at the child with 10M *max-limit* will still allow $10 * 10M = 100M$ of traffic.

But the Parent is replenishing the bucket at 20Mbit rate. So it will take ~5seconds to be able to empty the 100M bucket, before the queue settles at the actual Token Rate of 10M.



HTB

- Very predictable regular traffic can be handled by small buckets. Larger buckets may be required for burstier traffic, unless one of the desired goals is to reduce the burstiness of the flows.



Credits

- [https://wiki.mikrotik.com/wiki/Manual:HTB-Token Bucket Algorithm](https://wiki.mikrotik.com/wiki/Manual:HTB-Token_Bucket_Algorithm)
- [https://wiki.mikrotik.com/index.php?title=Manual:Queues - Burst](https://wiki.mikrotik.com/index.php?title=Manual:Queues_-_Burst)
- <http://linux-ip.net/articles/Traffic-Control-HOWTO>
- <http://www.docum.org/>

Grazie!

Thank You!

Grazzi!

¡Gracias!

Time for questions, answers and suggestions

ag@tnxe.net

mc@tnxe.net

