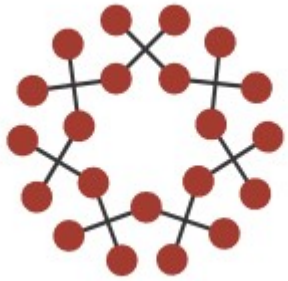


TITANIA®

NETWORKS LTD

# HTB Network Packet Scheduler implementation:

## experimenting with the new bucket size feature



Titania Networks Limited

# Alfredo Giordano

TITANIA NETWORKS LTD

- 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

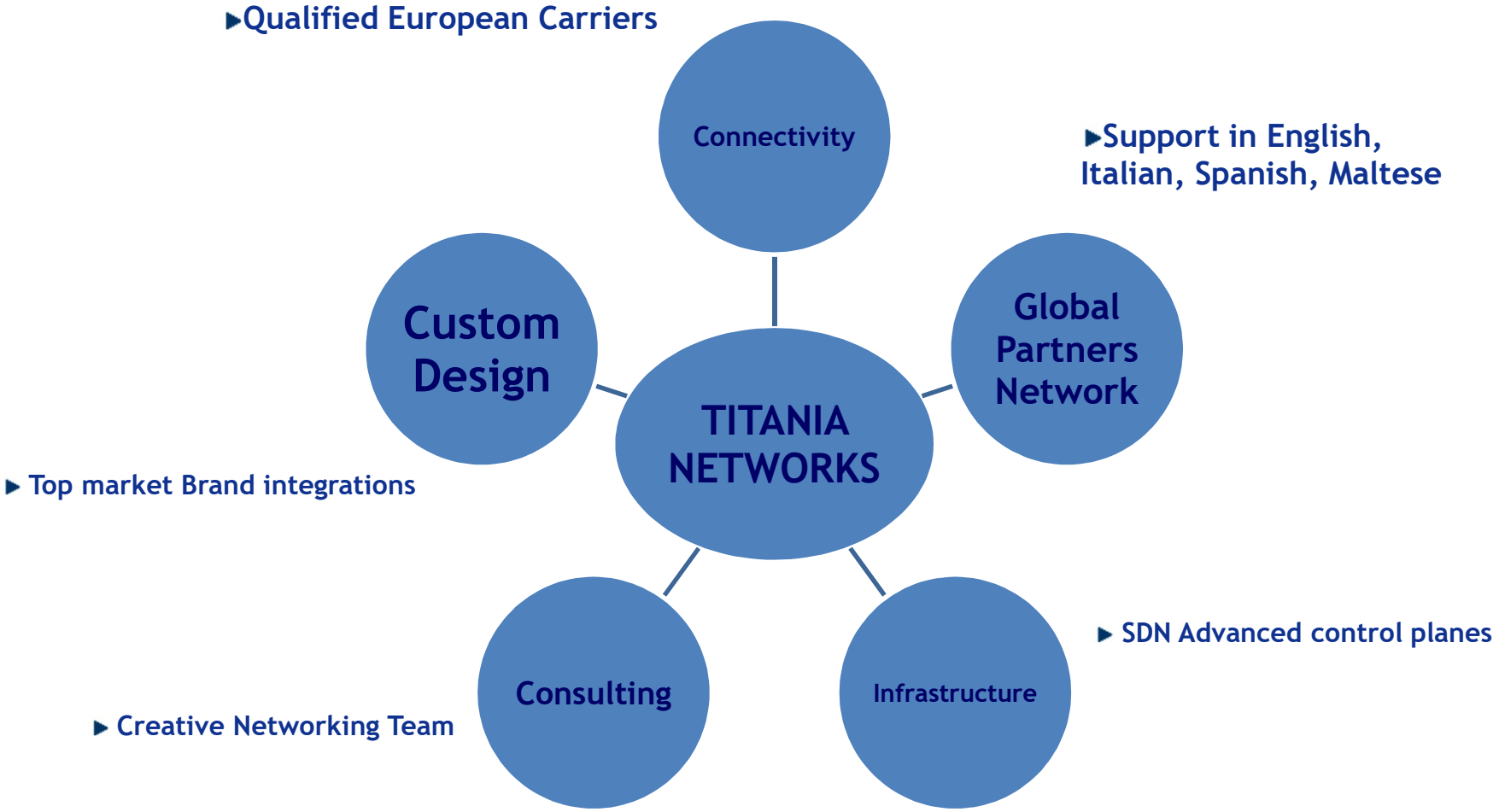
TITANIA NETWORKS LTD

- 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

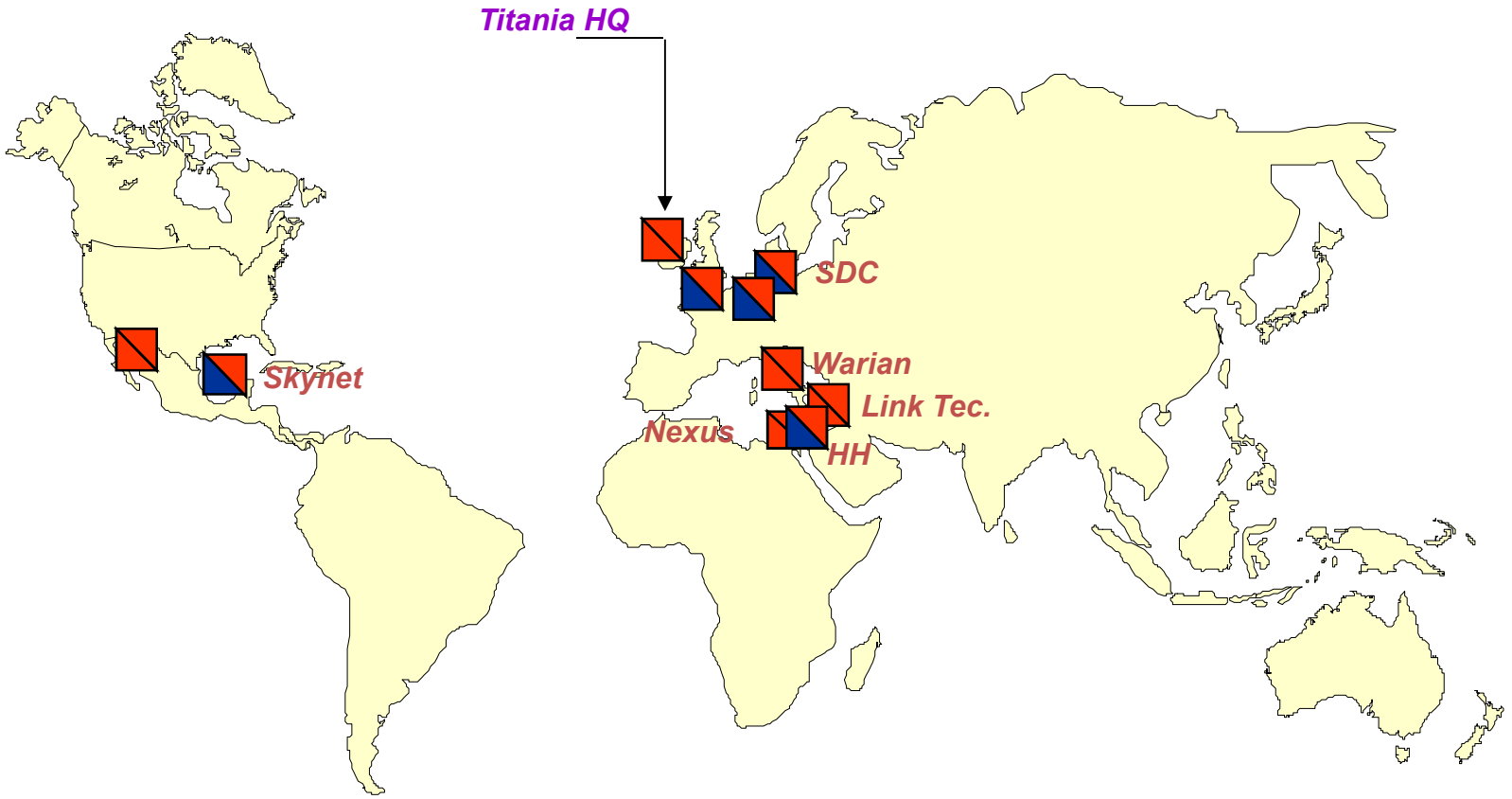
- Started in 2014, by providing IT Training and Consultancy under the brand tiktrain.com
- Incorporated in 2015 as a company in Ireland started operations in Europe.
- Most requested services:
  - Mission critical Networking consulting
  - ISP Design
  - Network Training
- Operation area:
  - Europe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

# Operations




- With a single point of contact.

# 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

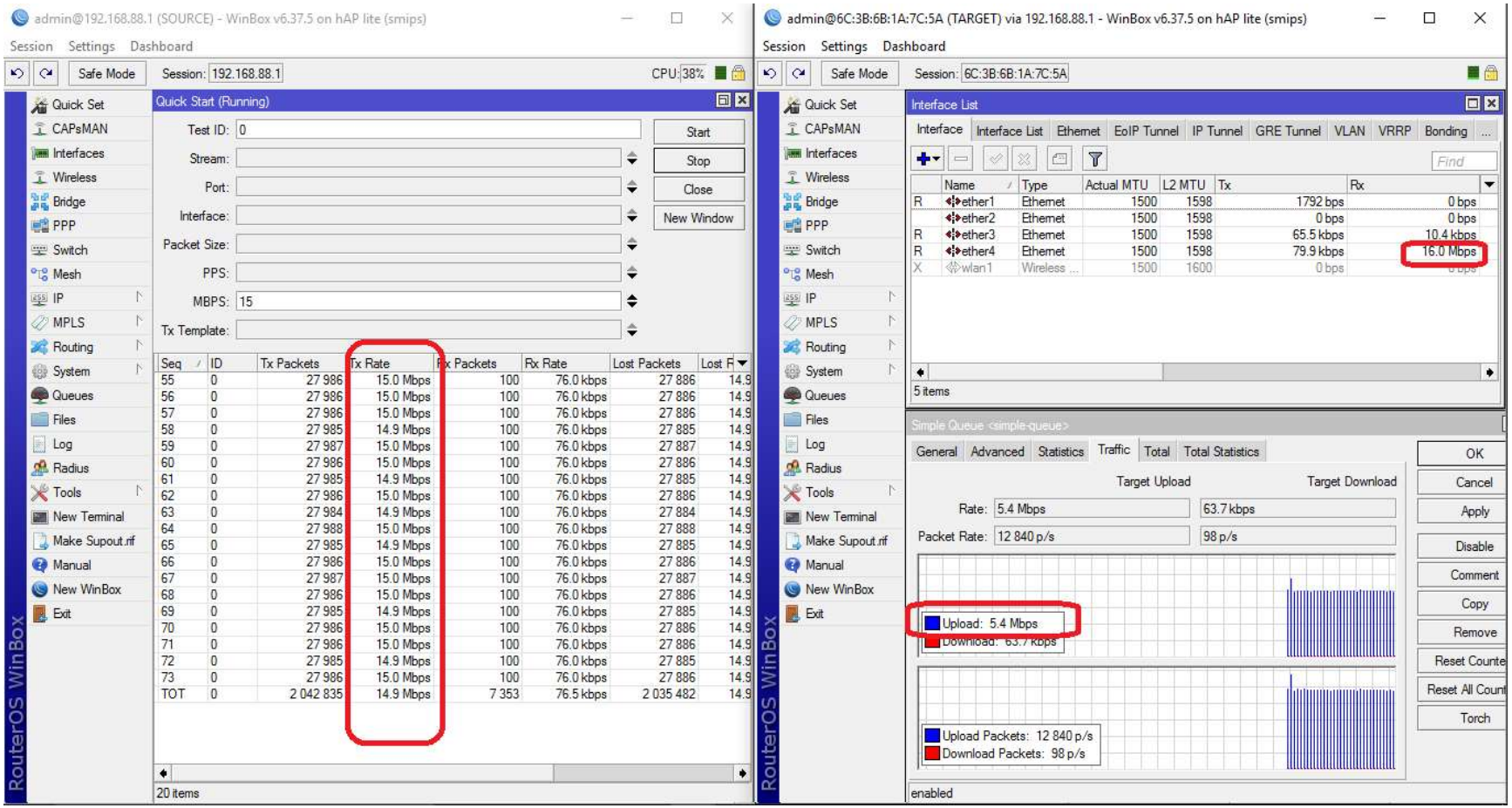
# Concepts

- Scheduling
- Shaping
- Queue

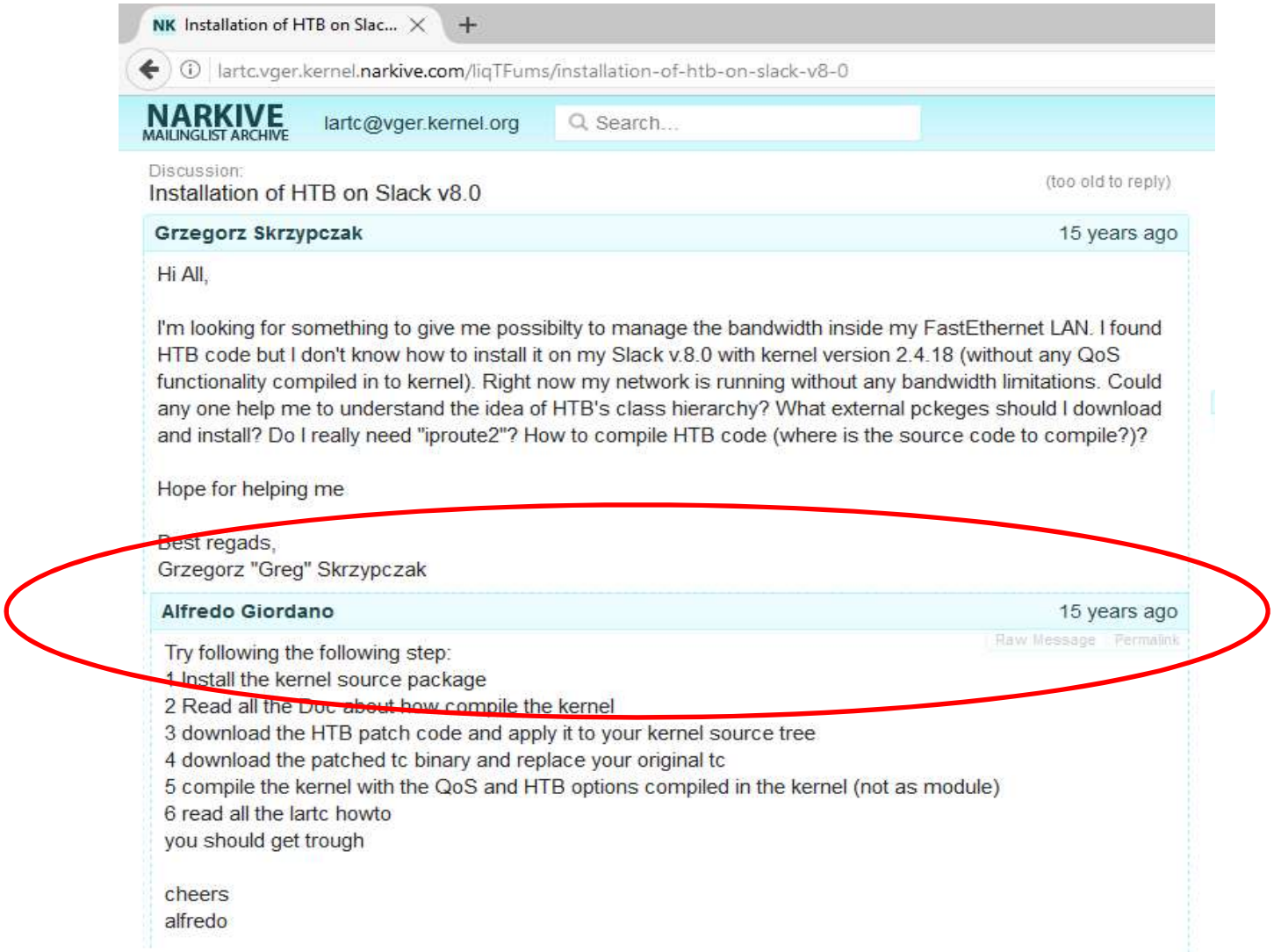
# The truth about Queues


- 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



# The Linux Kernel

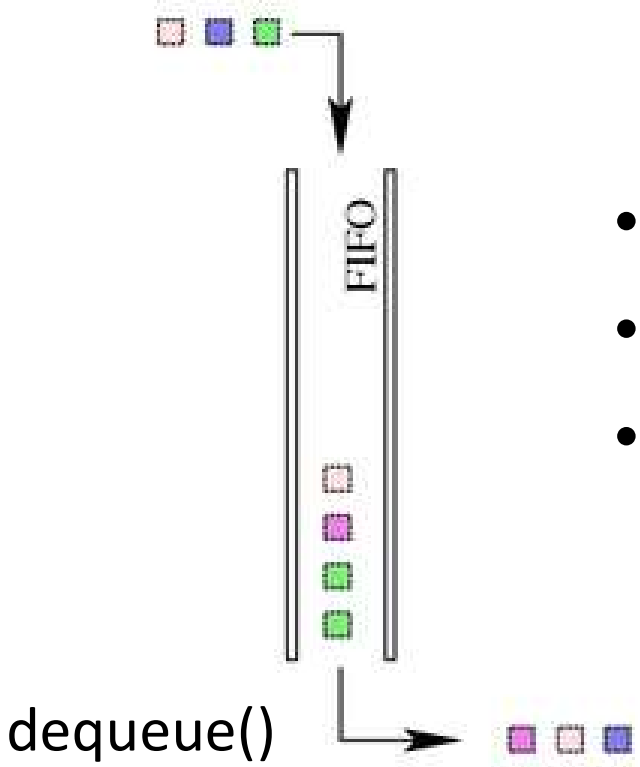


# The Linux Kernel

TITANIA NETWORKS LTD

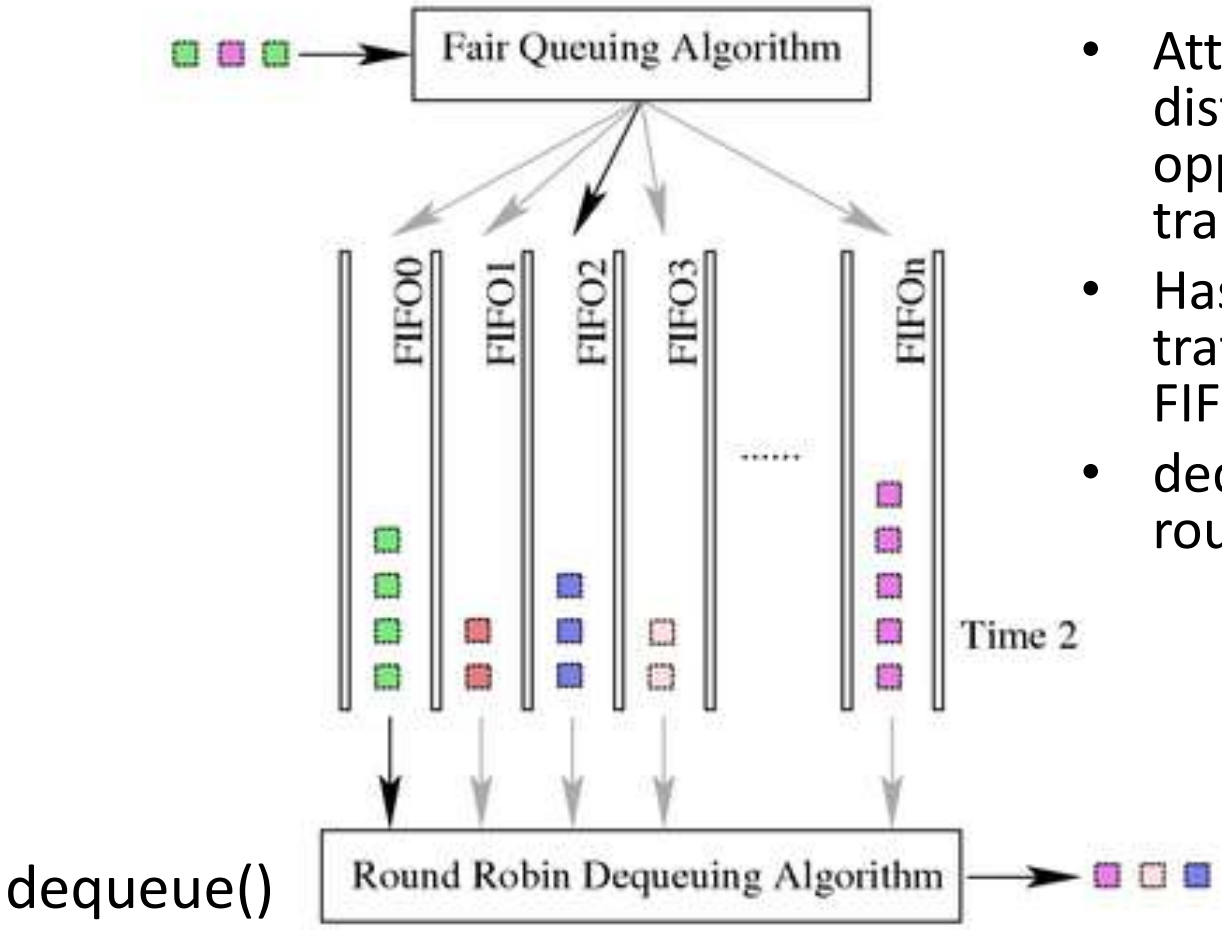
- The forwarding stack or the local process sends data to the kernel.
- Kernel enqueues data to the *queue-type* selected for the queue and immediately tries to run the queue to the hardware using queue_run()
- The function will call dequeue() according to the *queue-kind* algorithm to send to hardware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

# PFIFO




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

# SFQ

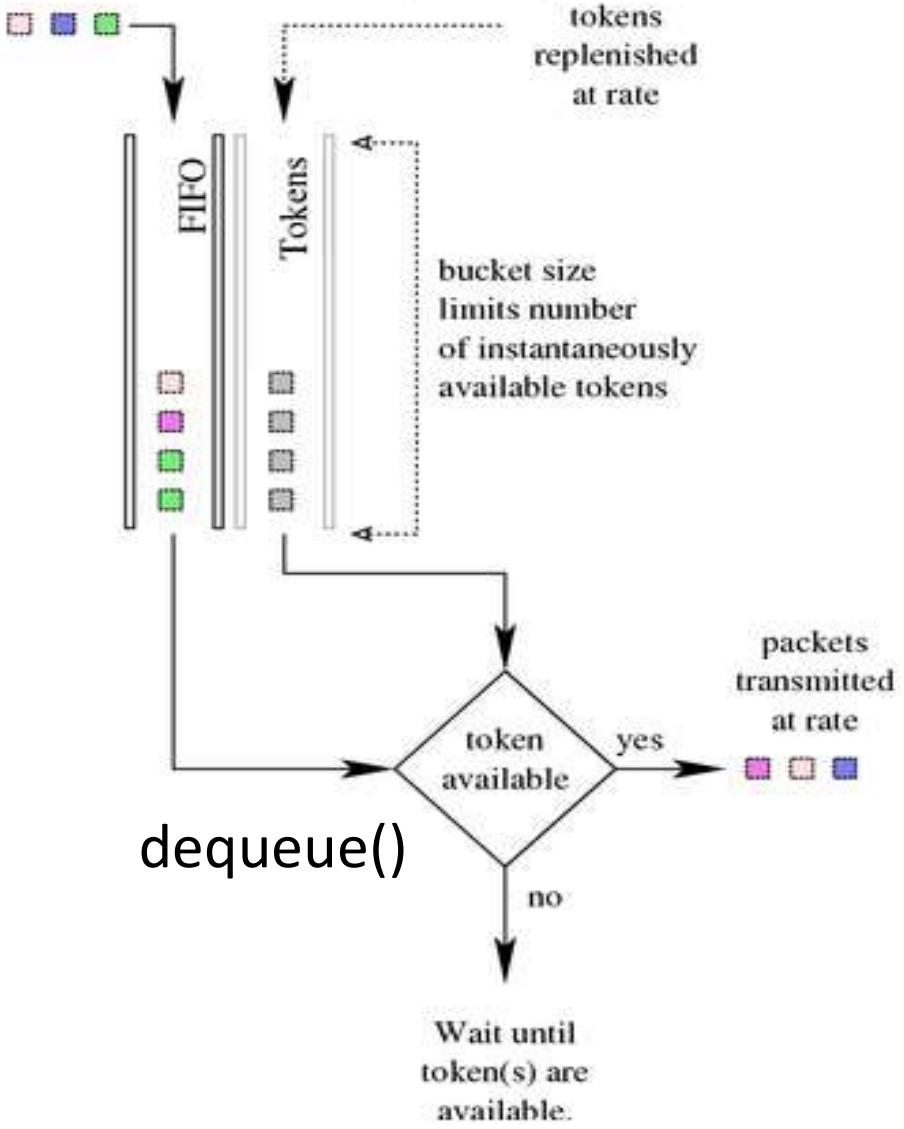



- Attempt to distribute opportunity to transmit fairly
- Hash function to fit traffic in separate FIFOs
- dequeue() with round robin

# 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

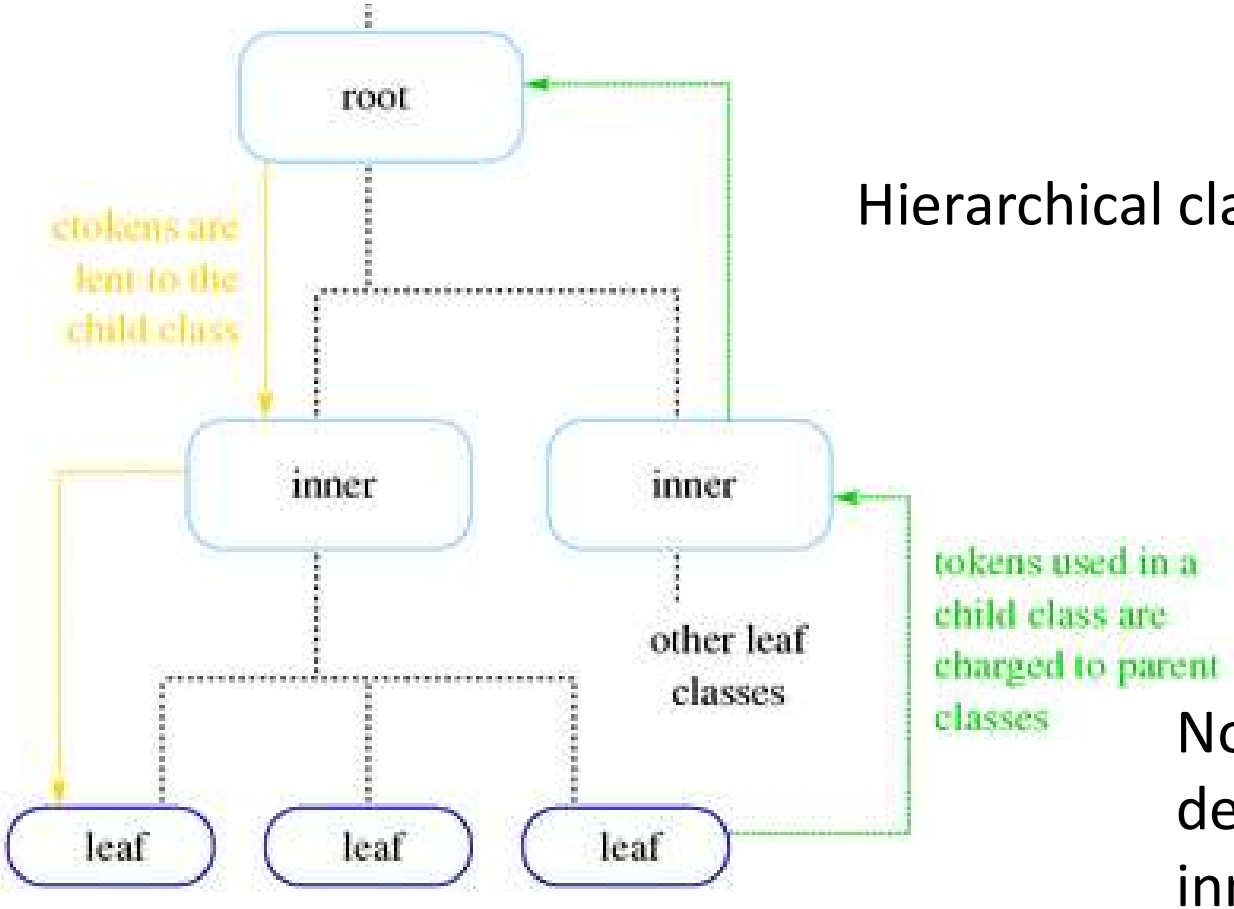



- Built on tokens and buckets
- Packets are only transmitted if there are sufficient tokens available.
- Otherwise, packets are deferred.
- It will introduce an artificial latency

# 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




Hierarchical class structure

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 |

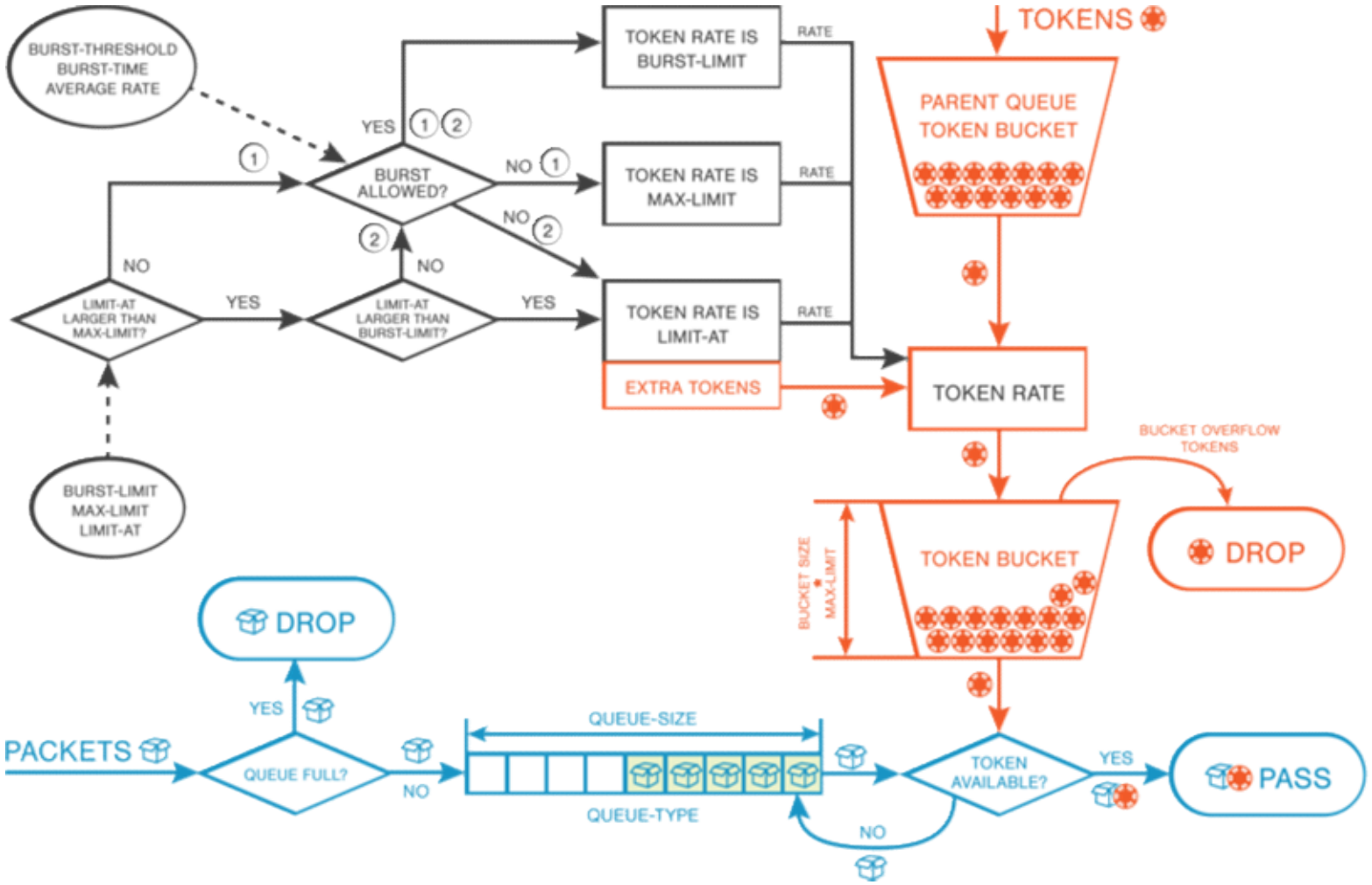
# HTB - Burst

- Burst is a feature that allows to satisfy queue requirement for additional bandwidth even if required rate is bigger that *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 that *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 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:

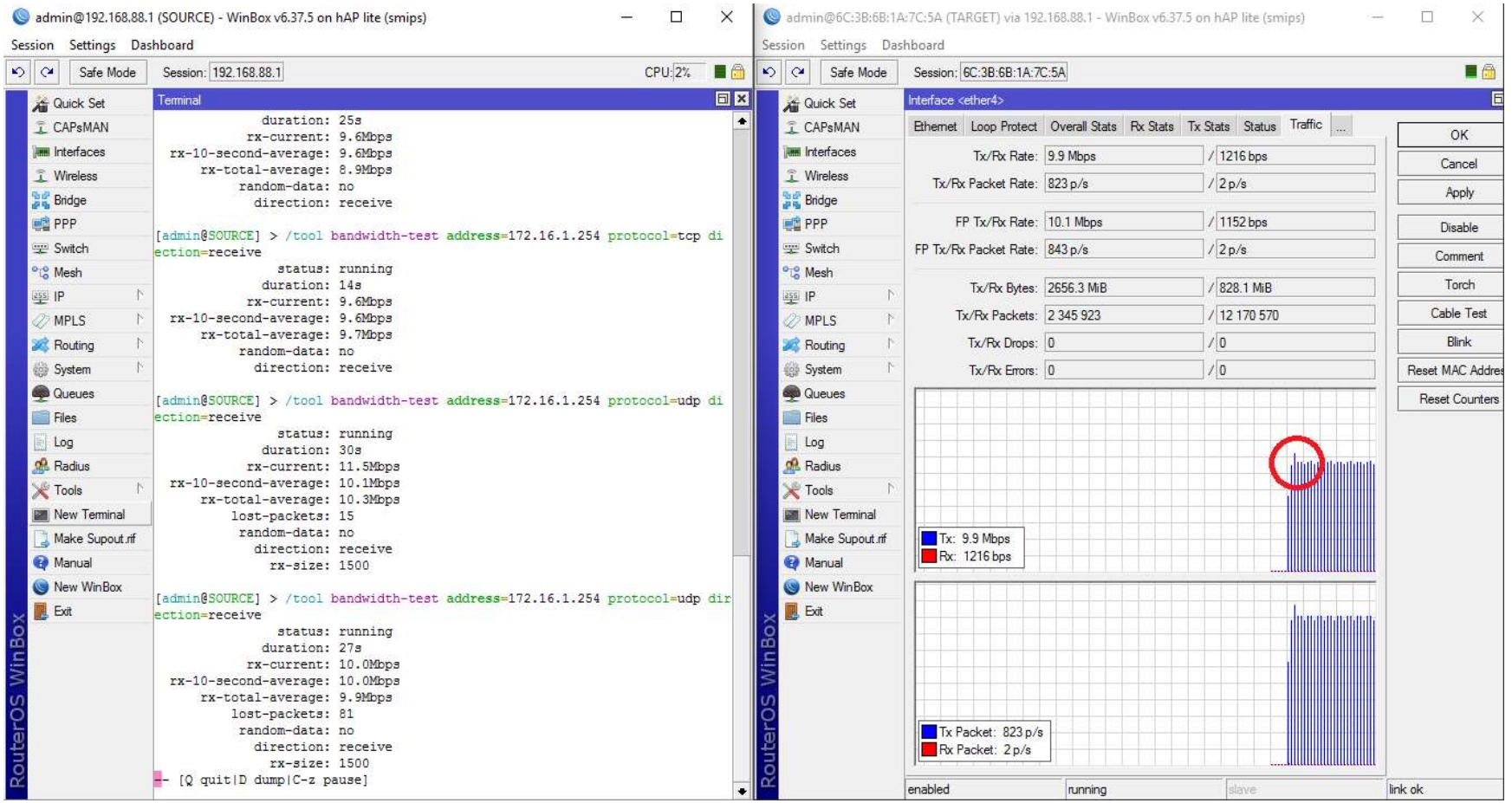
Bucket Capacity = *bucket-size* * *max-limit* (or *burst-limit*, if *burst-limit* is being used)

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



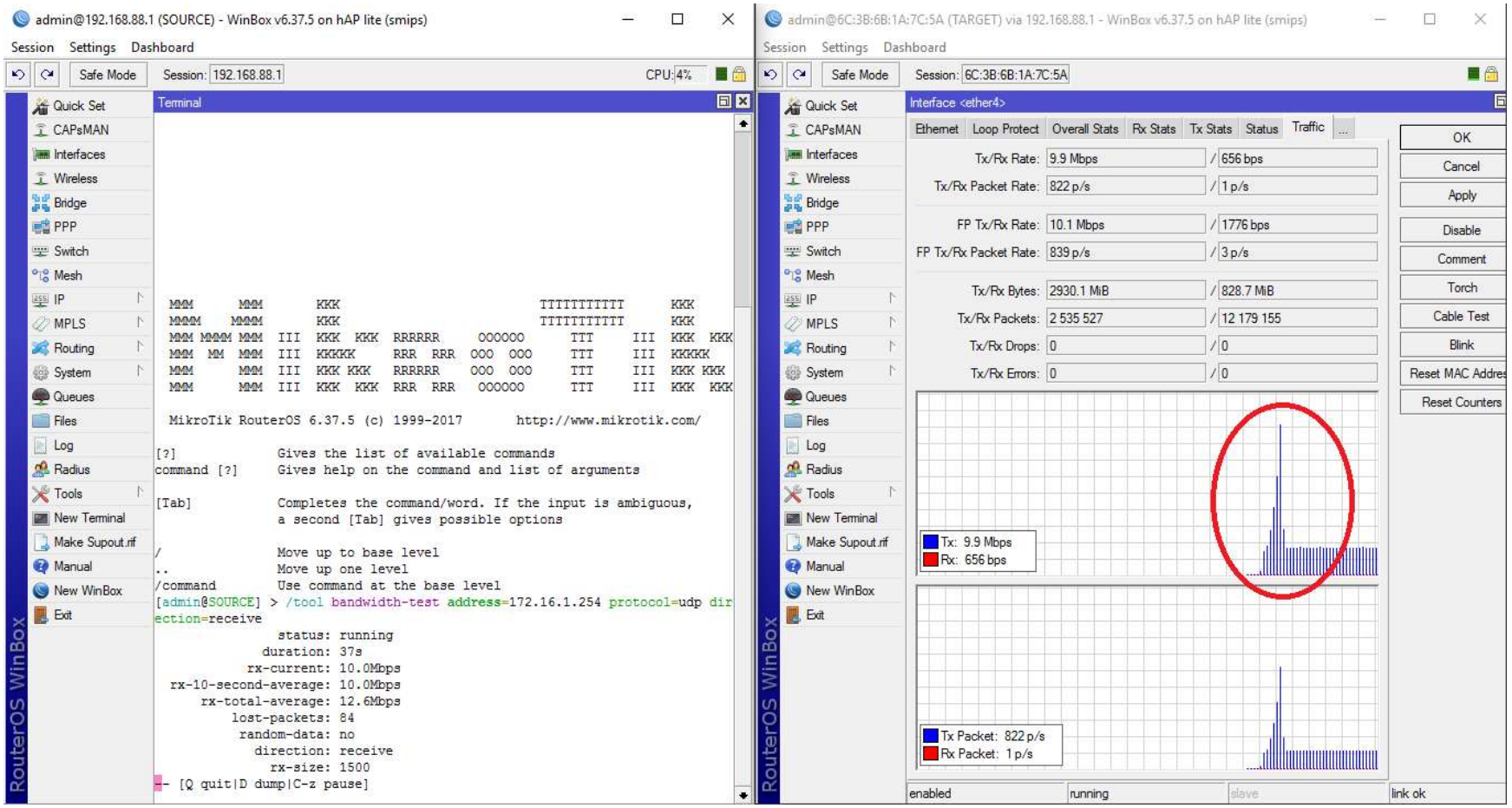
# How much traffic will pass Unrestricted?

Large Bucket with 10M *max-limit*

10 * 10M = 100M

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

# Large Bucket Size

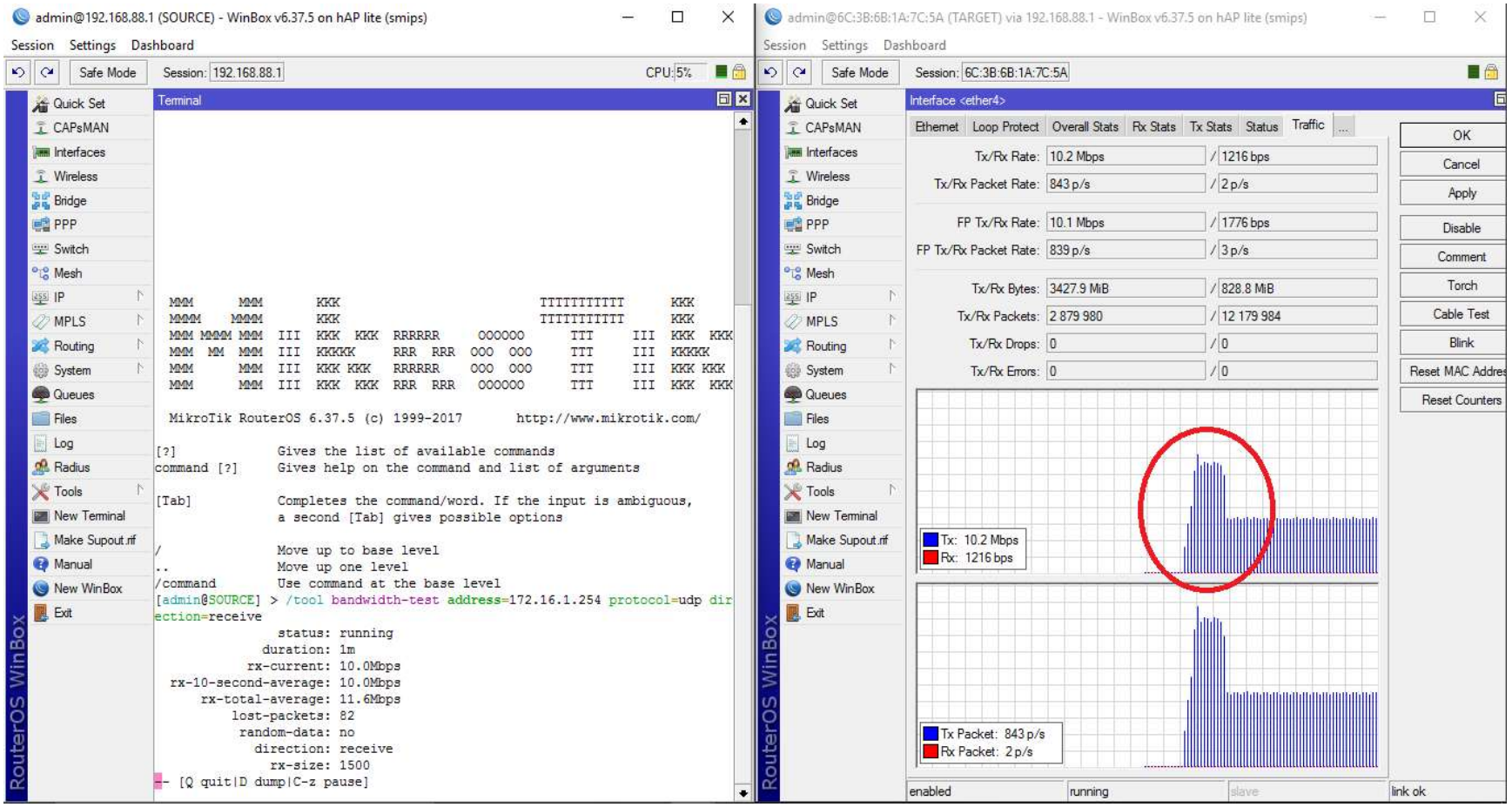


# What if one wants a ceiling for the burst?

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/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*