Mikrotik Traffic control with HTB

Who Am I

- David Attias
- Mikrotik Certified Trainer
- Team Member and sponsor of The Brothers Wisp
- Own Penny Tone LLC, a cloud hosted VoIP and phone systems provider

Todays Presentation is on

Traffic control with HTB

Who is this presentation for?

- ISP's, MSP's, Consultants, Network Engineers
- Suggested skill level: Beginner and intermediate
- Prerequisites: Network engineering & Mikrotik RouterOS experience
- The examples in this presentation are focused on customer networks

Topics in this presentation

- **Traffic control concepts**
- **What is Traffic control / QoS**
- **Classifying traffic / mangle**
- **Scheduling traffic / queues**
  - Queue Types - scheduling vs shaping
- **Shaping traffic with HTB**
  - Tokens – Buckets – bursting
  - Burst Lab

What is traffic control / QoS?

A system that:

- Regulates data flows
- Ensures sufficient though put of high priority traffic
- Promotes low latency for higher priority traffic
- How? Selectively delay or drop lower priority traffic

Important points

Important points

- We need to “tell” the Mikrotik what the total available upload and download bandwidth for the link we are going to be traffic shaping on.
- We can only “effectively” queue traffic that exits an interface
- Traffic control becomes effective when all available bandwidth of a link is maxed out.
- Simplest crudest way to overcome traffic congestion problems is to buy more bandwidth (if possible)
- Do not use Fast-Track

Three Phases of traffic control

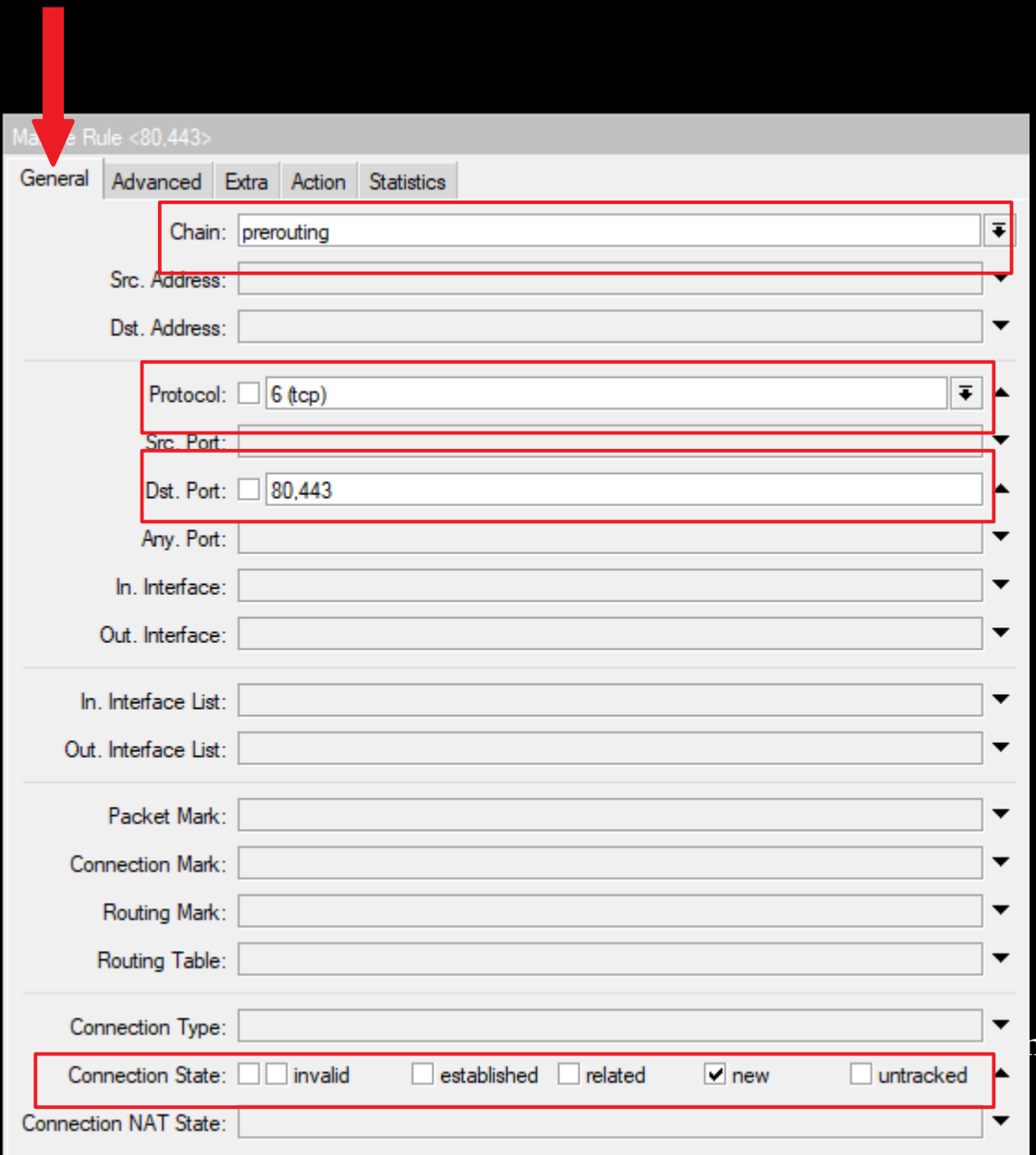
1. Classify (marking packets)
2. Schedule (enqueuing packets)
3. Shape (dequeuing packets)

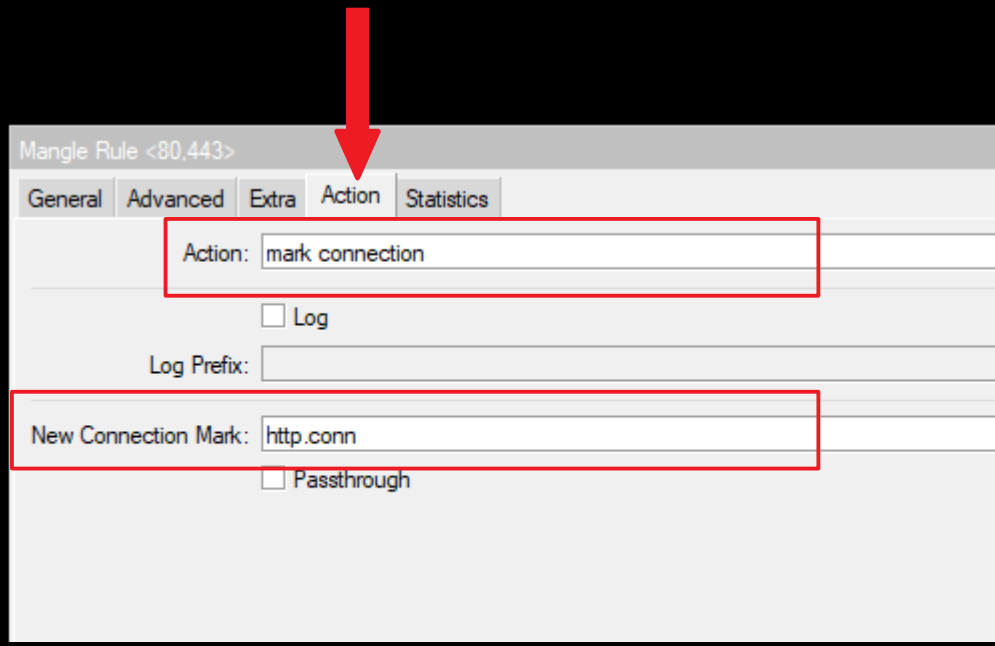
Classifying with Mangle

Classifying with Mangle

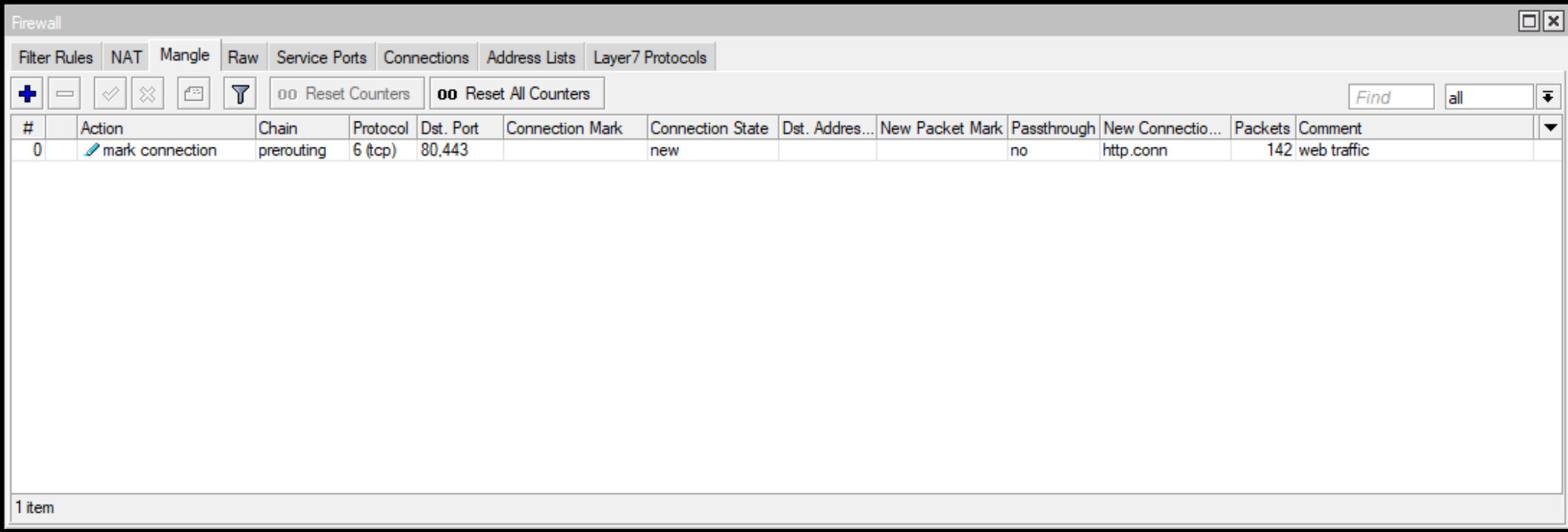
- Classify / “categorize” traffic with mangle
- Mangle is a RouterOS facility that marks packets for future processing
- RouterOS only allows one packet mark, one connection mark and one routing mark per packet.
- The mangle facility is also used to modify some fields in the IP header, like TOS (DSCP) and TTL fields.
- Mangle rules are processed sequentially (be mindful when setting passthough=yes dscp marking/ remarking)
- It’s good practice to Mark the connection first then use the “connection mark” to perform a “packet mark” (if possible / TCP ACK) (low CPU usage)

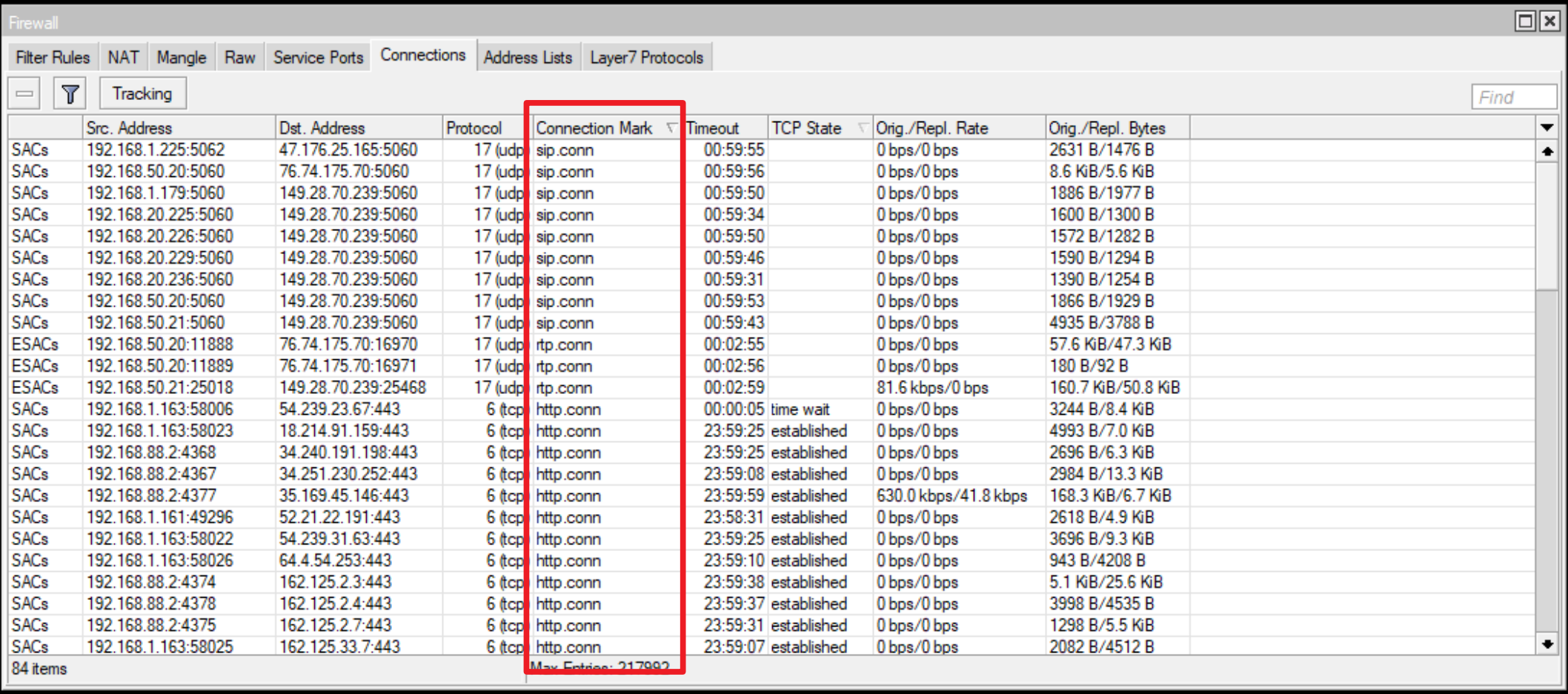
> ip firewall mangle
> add action=mark-connection chain=prerouting comment="web traffic" connection-state=new dst-port=80,443 new-connection-mark=http.conn protocol=tcp





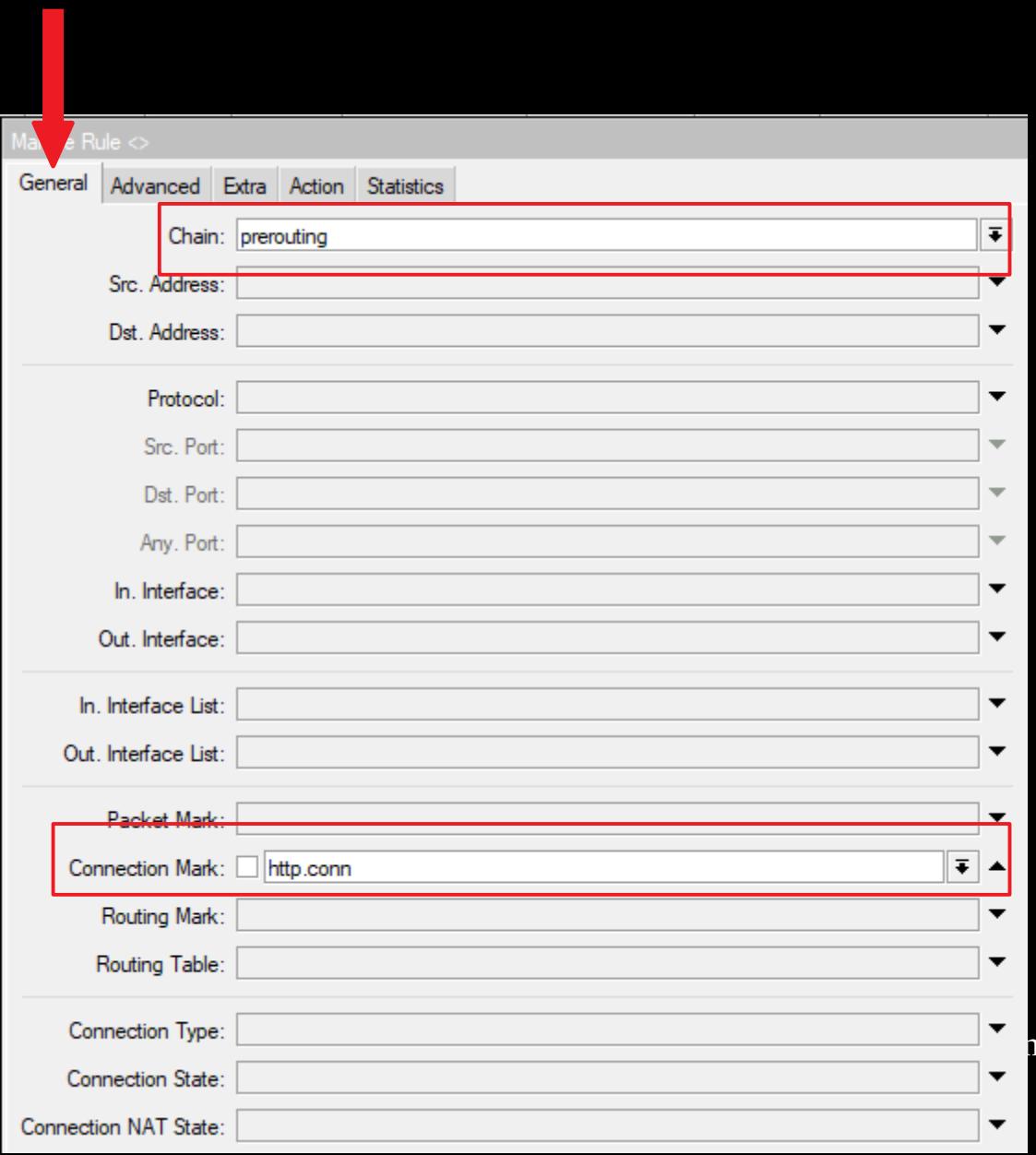
> ip firewall mangle
> add action=mark-connection chain=forward comment="web traffic" connection-state=new dst-port=80,443 new-connection-mark=http-connection protocol=tcp

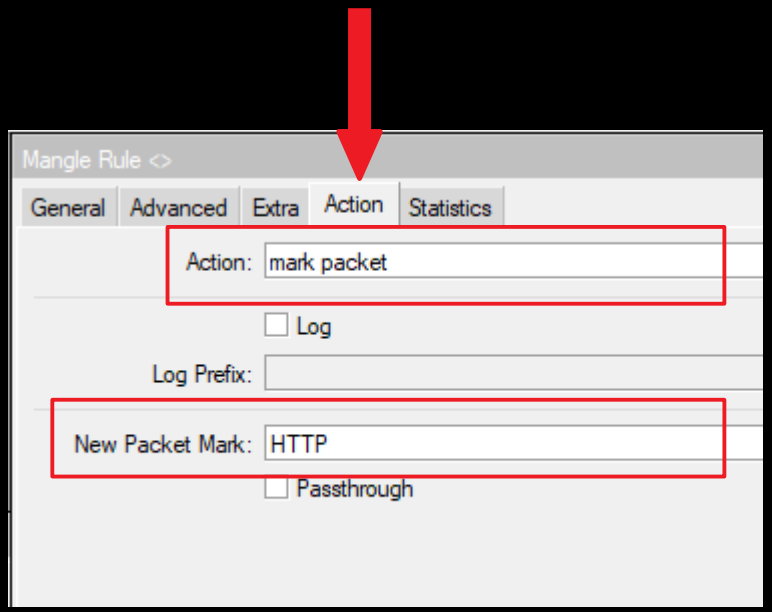




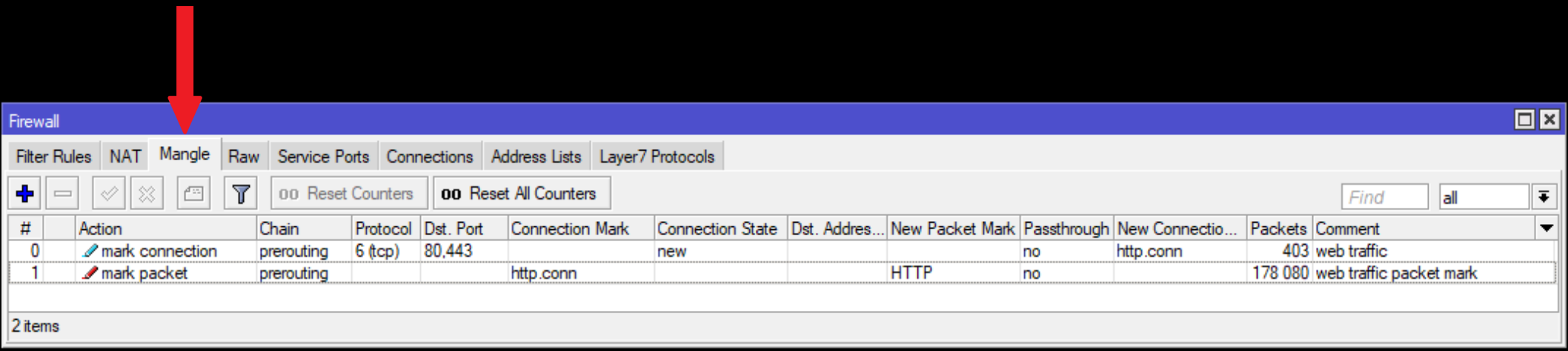
IP → Firewall → Connections

> ip firewall mangle
> add action=mark-packet chain=prerouting comment="web traffic packet mark" connection-mark=http.conn new-packet-mark=HTTP





> ip firewall mangle
> add action=mark-connection chain=prerouting comment="web traffic" connection-state=new dst-port=80,443 new-connection-mark=http-connection protocol=tcp



Firewall

Filter Rules | NAT | Mangle | Raw | Service Ports | Connections | Address Lists | Layer7 Protocols

00 Reset Counters | 00 Reset All Counters | Find | all

#	Action	Chain	Protocol	Dst. Port	Connection Mark	Connection State	Dst. Addres...	New Packet Mark	Passthrough	New Connectio...	Packets	Comment
0	mark packet	prerouting	6 (tcp)					TCP.ACK	no		19 145	tcp-ack
1	mark connection	prerouting	6 (tcp)	80,443		new			no	http.conn	150	web traffic
2	mark packet	prerouting			http.conn			HTTP	no		6	web traffic packet mark
3	mark connection	prerouting	17 (udp)	5060		new	voip-servers		no	sip.conn	14	sip connection
4	mark connection	prerouting	6 (tcp)	5060		new	voip-servers		no	sip.conn	1	sip connection
5	mark packet	prerouting			sip.conn			SIP	no		1 332	sip traffic packet mark
6	change DSCP (TOS)	postrouting							no		1 332	SIP dscp mark for packets that ...
7	mark connection	forward			sip.conn	related			no	rtp.conn	3	rtp connection
8	mark packet	prerouting			rtp.conn			RTP	no		4 816	rtp mark packet
9	change DSCP (TOS)	postrouting							no		4 816	RTP dscp mark for packets that...
10	mark connection	prerouting				new			no	management.vpn	0	
11	mark packet	prerouting			management.vpn			MGR.VPN	no		0	winbox-vpn traffic packet mark
12	mark connection	input	6 (tcp)	8291		new			no	winbox.conn	2	winbox
13	mark packet	output			winbox.conn			WINBOX	no		128	winbox traffic packet mark
14	mark packet	prerouting	17 (udp)					DNS	no		1 614	dns packet mark

Scheduling with Queue

Queues

- A queue is a facility in RouterOS that process packets prior to exiting the physical interface
- A queue is a temporary buffer that packets enter. A queue will either drop, delay, or allow packets to pass unrestricted
- Packets that enter a queue may be organized or reorganized based on a chosen algorithm (FiFo, SFQ, PCQ, RED) which will dictate how they will exit the queue
- Queues must be configured with bandwidth limits
- The RouterOS queuing implementation is based on HTB

Simple Queue vs Queue Tree

Simple Queues vs Queue Tree

Simple Queues

- Processed sequentially
- Uses multiple processor cores
- Minimum requirement is "target" and "limit"
- Can shape based on the sum total of upload and download traffic
- Can use time conditions for when a queue is in effect.
- Auto generated with PPPoE

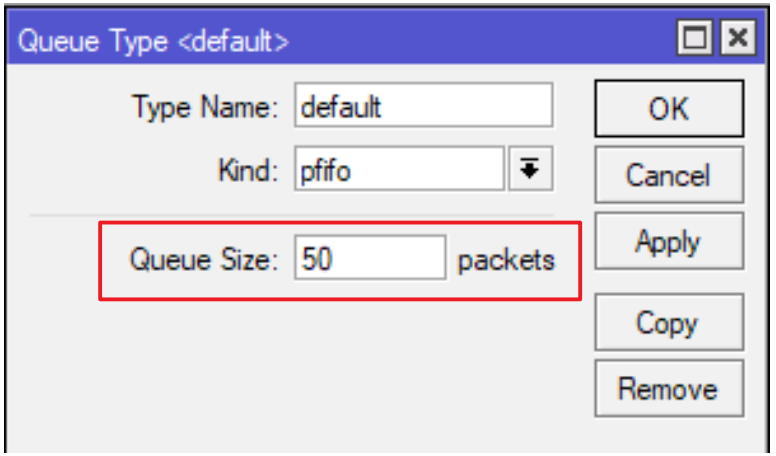
Queue Trees

- All rules processed at once
- Uses one processor core
- Only configurable with packet marks (mangle has dozens of matchers)

Queue size & Limits

Queue size

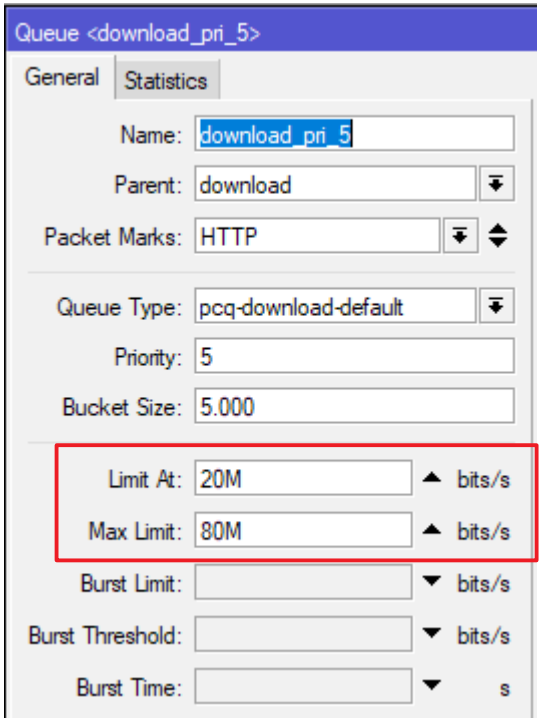
- Queue size = How many packets a queue can hold during congestion



Limits

- Limit-at = (CIR) Guaranteed bandwidth for the queue

- Max-limit = (MIR) The maximum bandwidth the queue can to pass

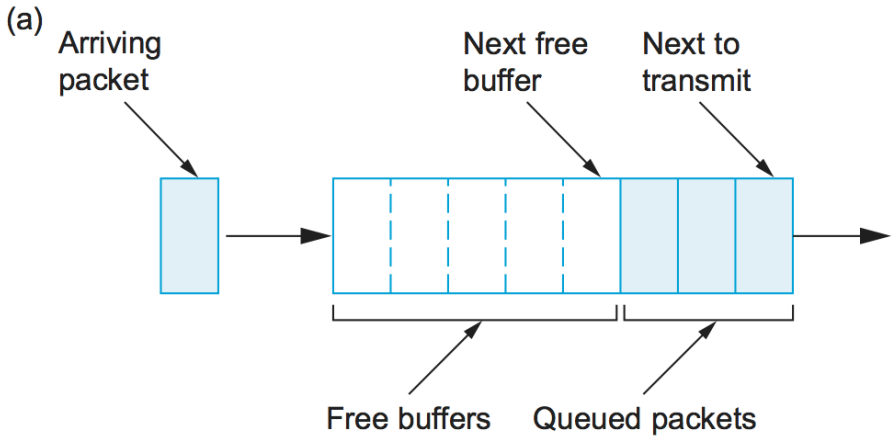


Queues will not work if max-limit is not specified

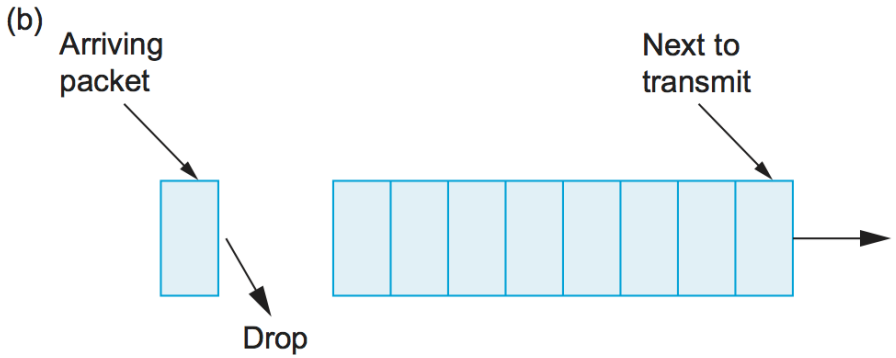
Why is max-limit important?

Why is max-limit so important?

- Once traffic exceeds max-limit, a queue can be configured to either drop or buffer packets.

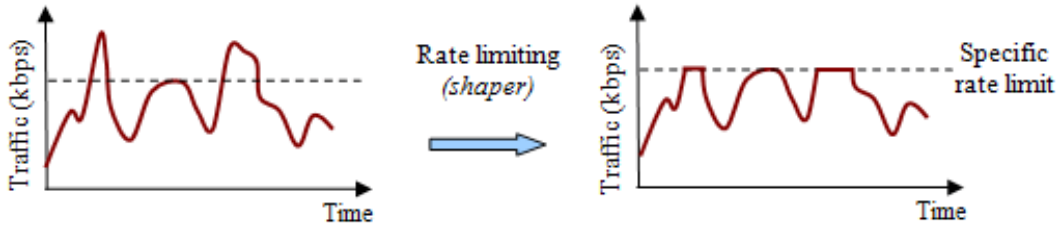


- Once the queue's buffer (queue size) is reached, packets trying to enter the queue will be dropped (tail drop)

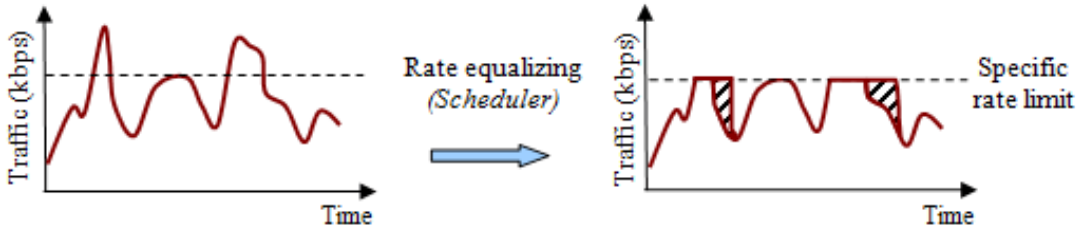


Policing VS Scheduling

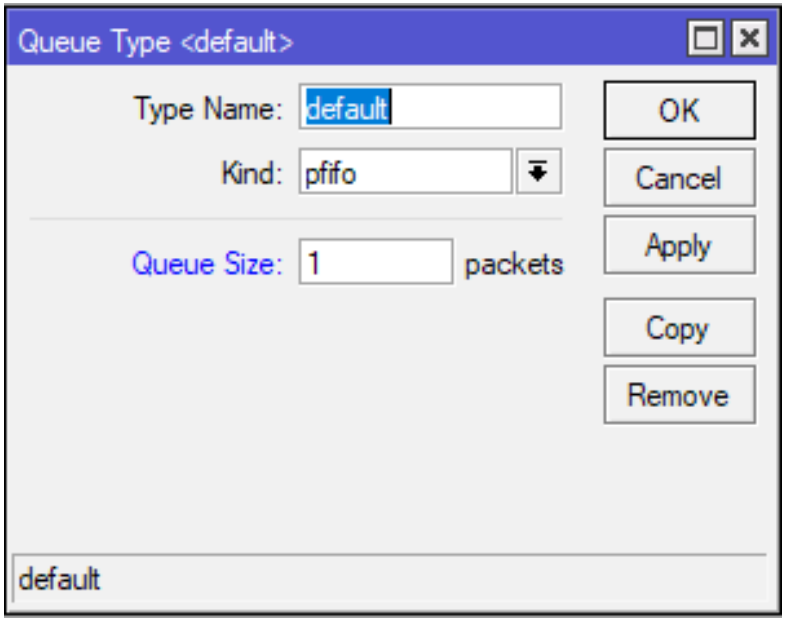
- Policing = once max-limit has exceeded, packets trying to enter this queue are dropped



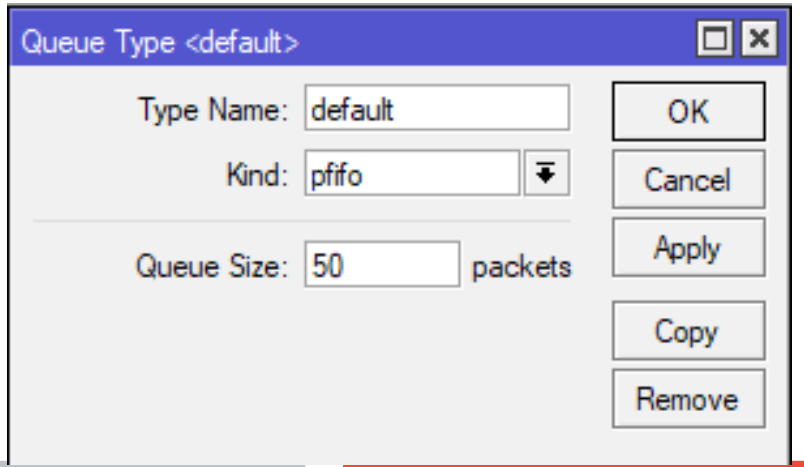
- Scheduling = Packets that exceed max-limit are enqueued. When bandwidth is available packets will dequeue



To configure a queue to police,
Set a FiFo queue size to 1



To configure a queue to schedule,
Set a FiFo queue size to >1



Shaper VS Scheduling

Policing

- Drops packets that exceed max-limit
- Lower latency for packets that are passed
- Success rate based on priority and properly sized limit-at values
- Better planning required to configure effectively

Scheduler

- Queues packets once max-limit is exceeded
- Creates delay / latency
- higher probability of packet delivery
- To configure effectively, follow parent / child limits rules & queue size

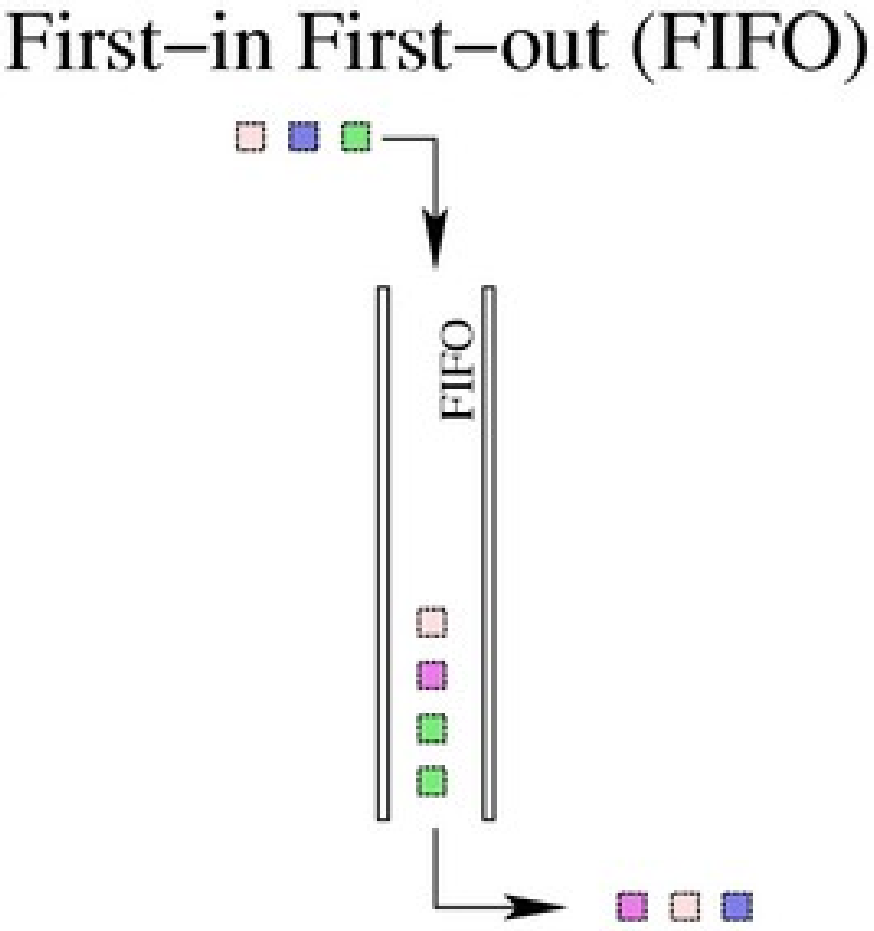
Queues Types

Queue Types (linux: Queue Disciplines)

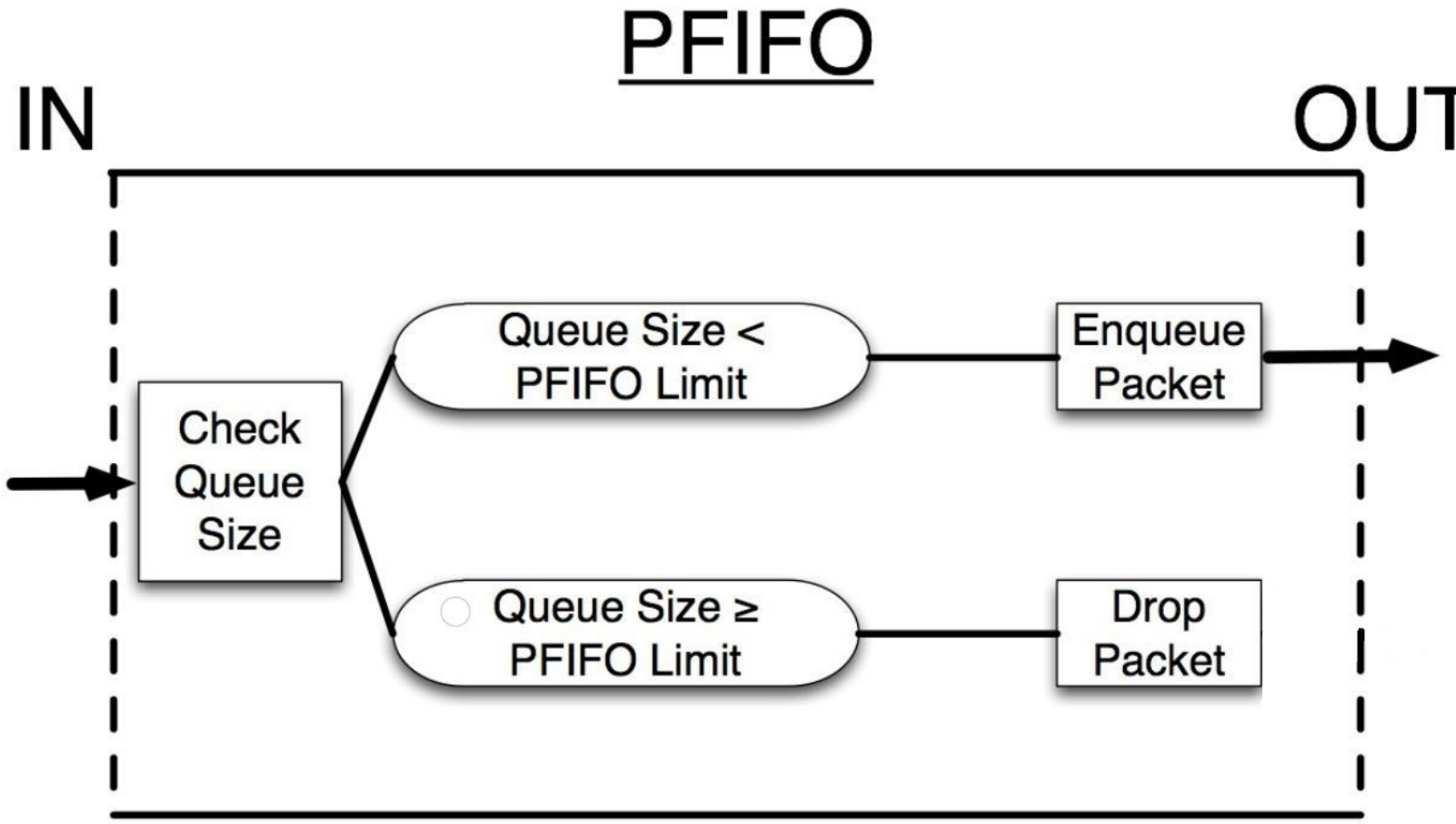
- FiFo – First in First out
- SFQ – Stochastic Fairness queuing
- PCQ – Per connection queuing
- RED – Random Early Detection

FiFo = First in First out

- The same sequence in which packets are enqueued, are dequeued

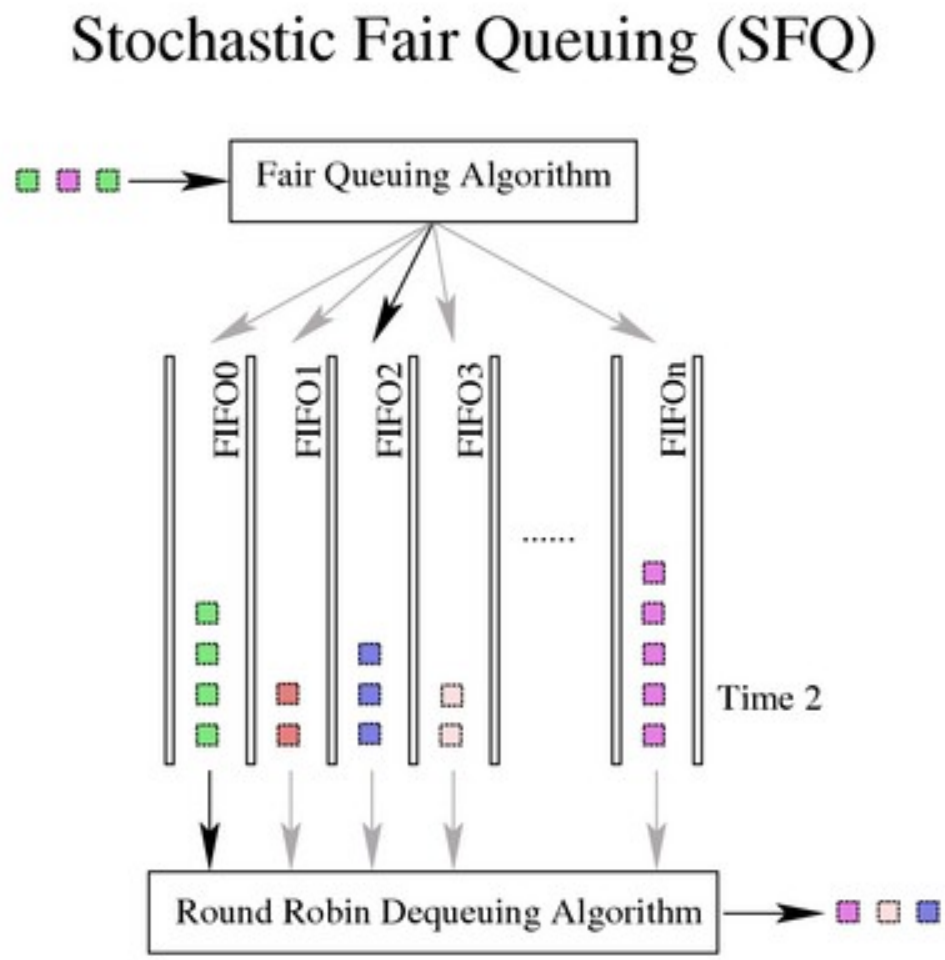


FiFo = First in First out

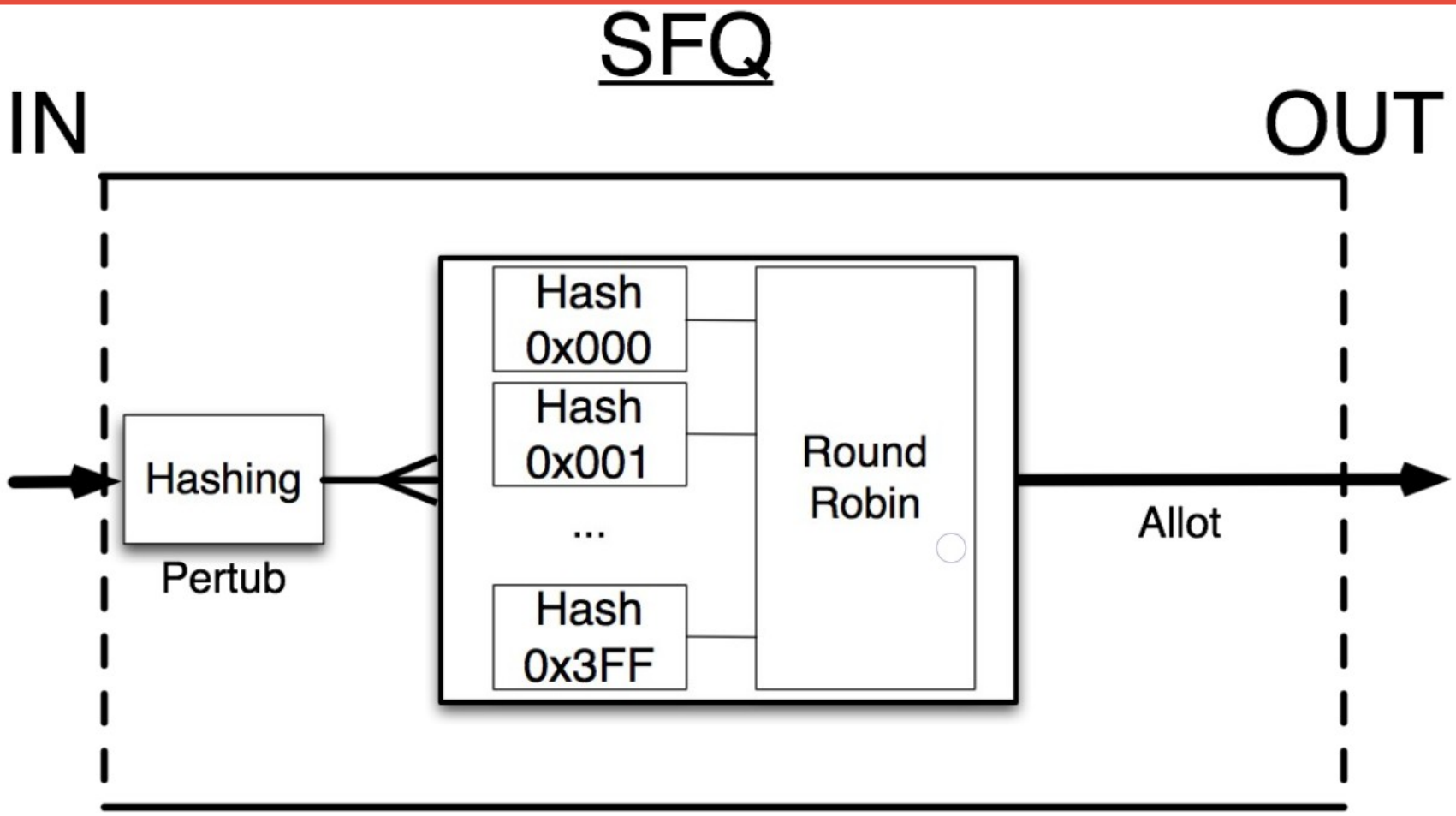


SFQ = Stochastic Fariness Queuing

- A hashing algorithm will classify traffic based on 4 identifiers, then put into any of 1024 possible sub streams
- De-queuing from sub streams will happen in a round robin fashion.



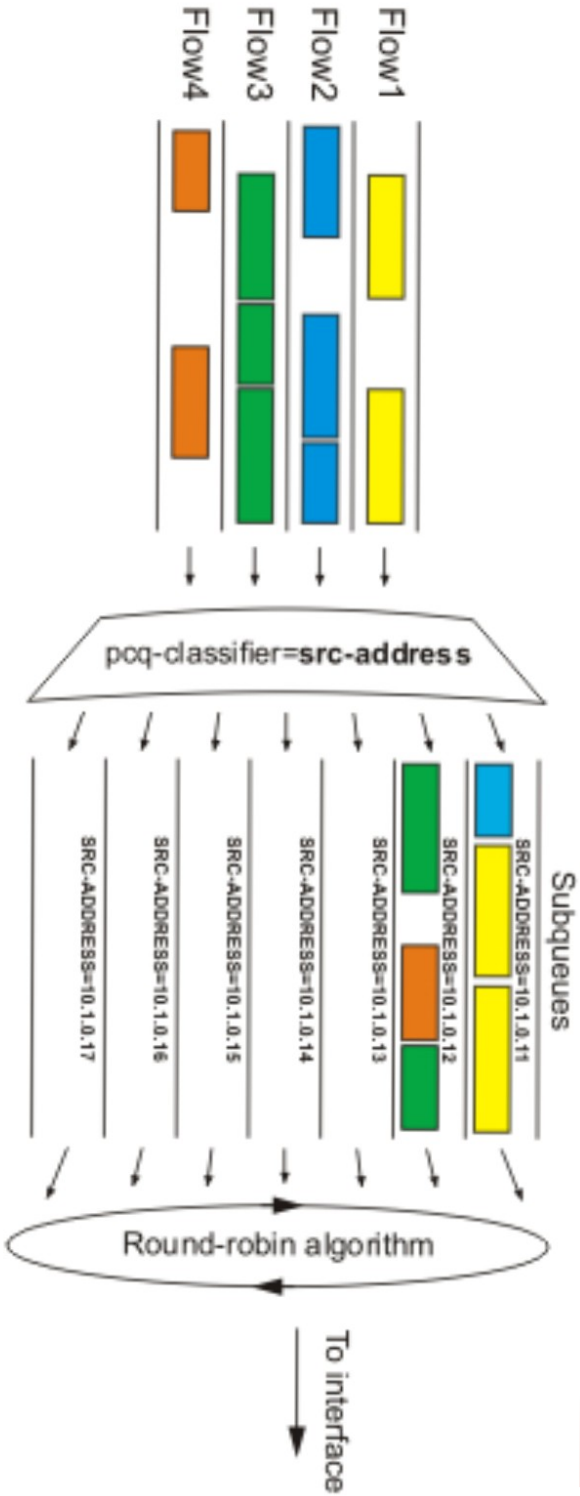
SFQ = Stochastic Fariness Queuing



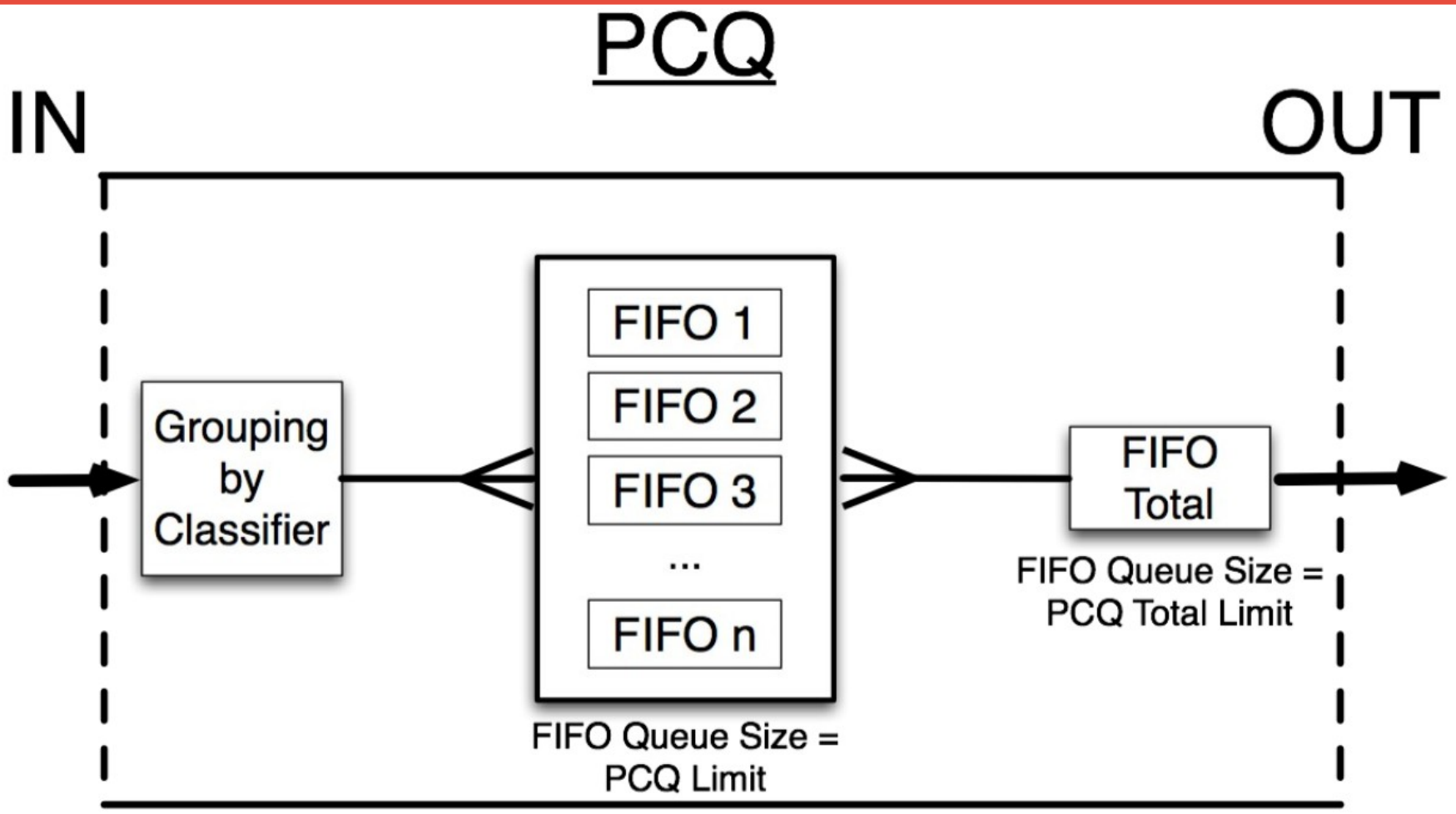
PCQ = per connection queuing

- Similar to SFQ but addresses the unfairness with SFQ by use of additional flow identifier

- Speed limitations can be applied or divided equally by number of flows



PCQ = per connection queuing

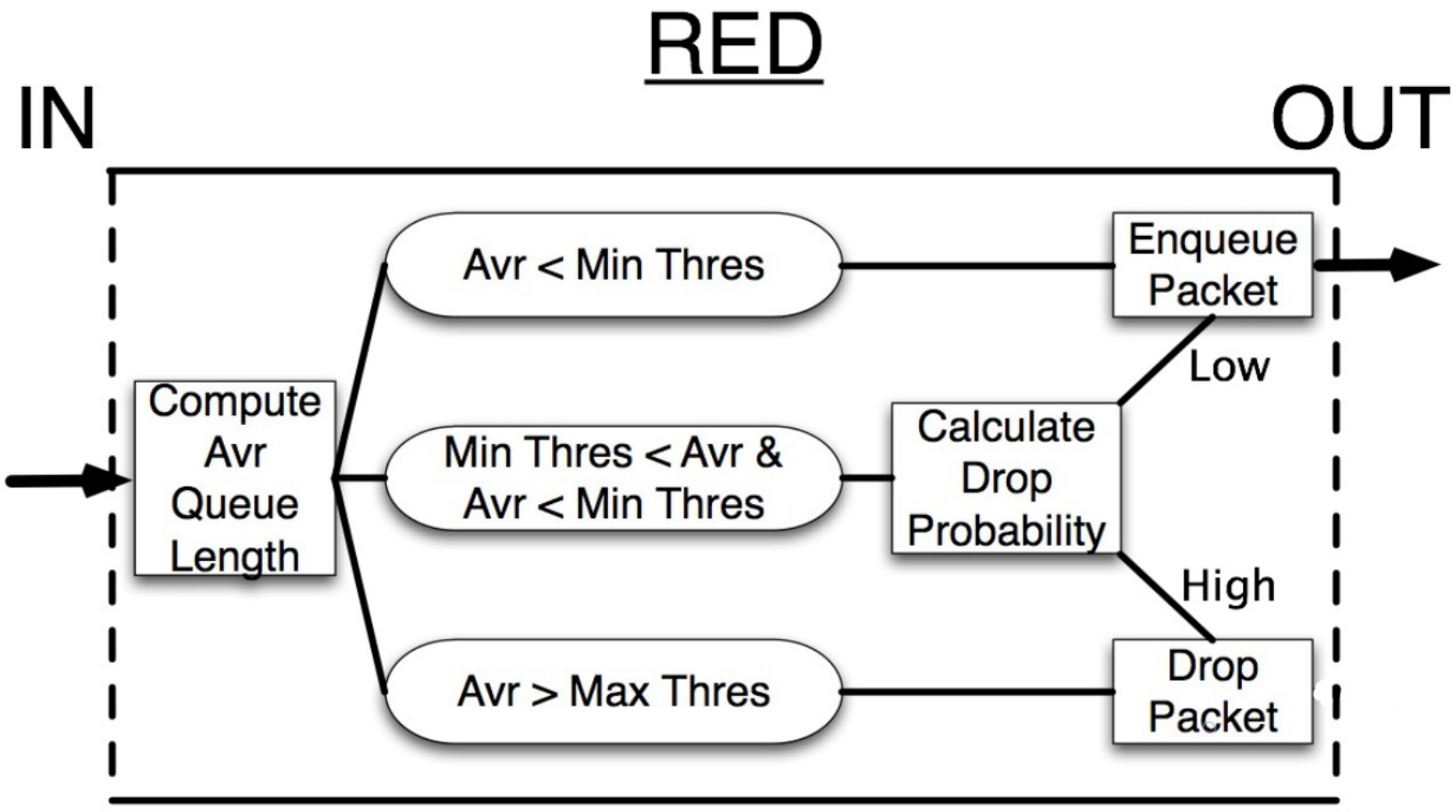


RED = Random Early Detection

- Random Early Detection is a queuing mechanism which tries to avoid network congestion by managing the average queue size.

- It helps to prevent TCP windows from collapsing and reset back to TCP slow start mode (or TCP Global Synchronization).

RED = Random Early Detection



Shaping with HTB

(The Mikrotik Sasquatch)

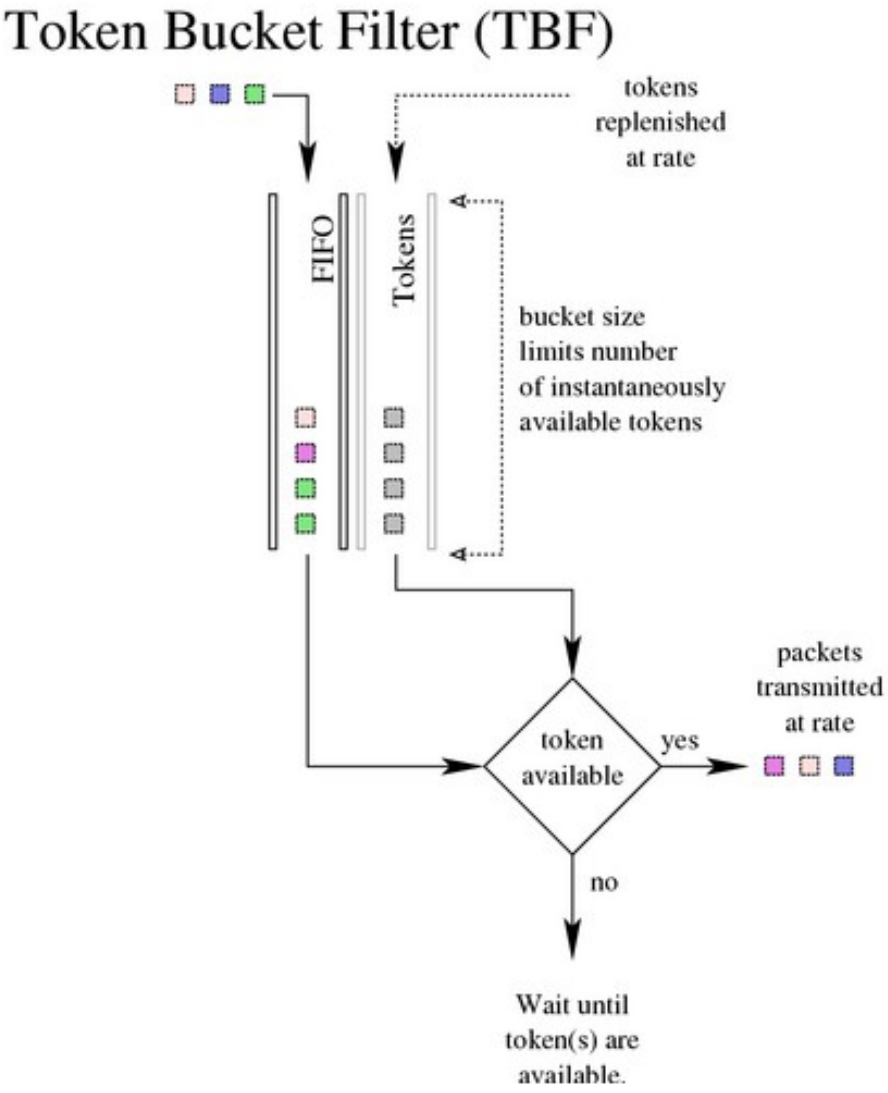
Shaping with HTB

- Shaping is act of “when” to allow a packet to exit a queue / dequeue
- Hierarchical token bucket builds relationships between queues (parents and children, priorities)
- Queues can be either parents or children (linux terms: innter queues or leaf queues)
- Setting flow limits and priorities is what determines when a packet can be dequeued.
- Each queue has a “bucket size” that hold tokens that will be used to escort packets to it’s exit interface.

Tokens

Tokens

- A packet can not dequeue without being escorted by a token
- 1 token can dequeue one 1KB of traffic
- Root parent queue is where token generation happens
- Tokens are issued at root parent's max-limit rate



Hierarchy

HTB = Hierarchical Token Bucket

- Hierarchy = Queues are configured in a hierarchy. Parent and child queues establish a “give and take relationship” for distributing and consuming bandwidth based on priority
- The Hierarchy works in one direction and is implemented on outbound interface

Name	Parent	Packet Marks
download	bridge-local	
download_pri_1	download	TCP.ACK
download_pri_2	download	RTP
download_pri_3	download	SIP
download_pri_4	download	MGR.VPN
download_pri_5	download	HTTP
download_pri_8	download	no-mark
upload	ether1-gateway	
upload_pri_1	upload	TCP.ACK
upload_pri_2	upload	RTP
upload_pri_3	upload	SIP
upload_pri_4	upload	MGR.VPN
upload_pri_5	upload	HTTP
upload_pri_8	upload	no-mark

Hierarchical Token Bucket

Parent and Child queues

Hierarchical Token Bucket

Parent Queue

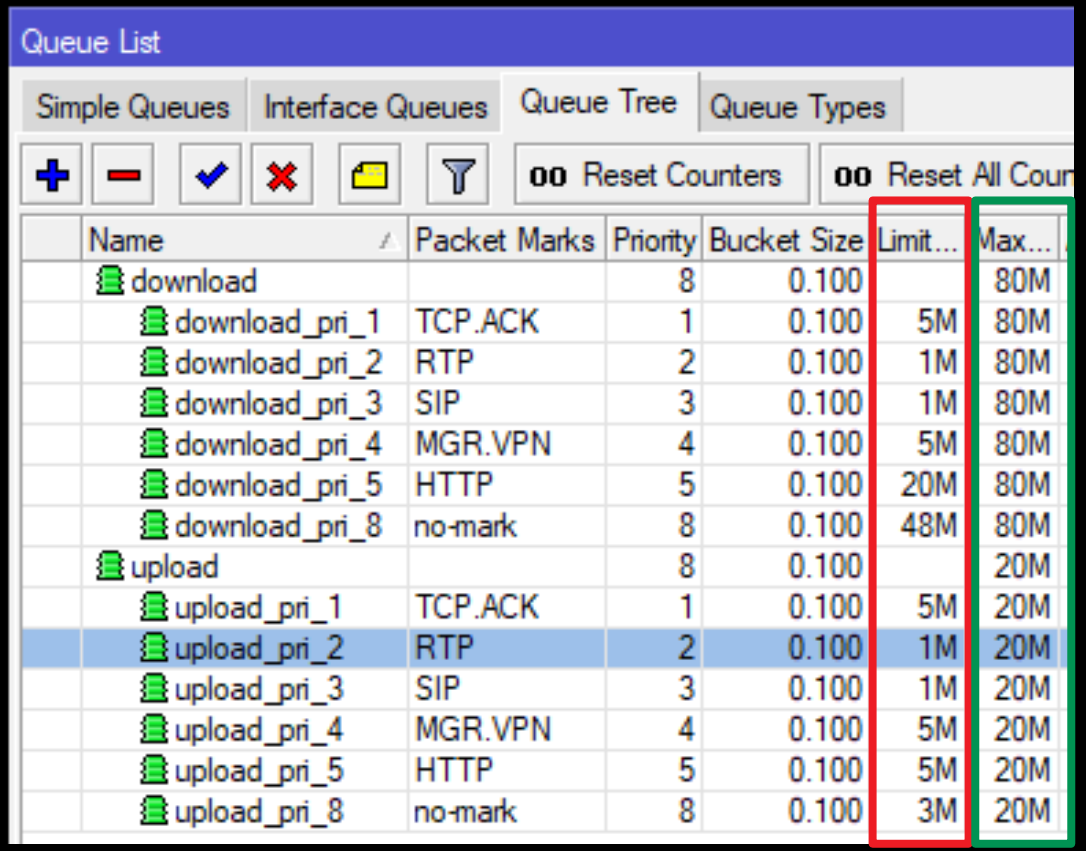
- Distribute bandwidth (tokens)
- Priority is ignored
- Parents will first satisfy the child queue's “limit-at” value then try and reach child “max-limit” in priority order

Child Queue

- Consume bandwidth / Spend tokens
- Priority dictates the order in which remaining tokens are given
- 8 is the lowest priority, 1 is the highest
- prioritization will work only if limits are specified

HTB bandwidth distribution:

- The sum of children's limit-at values should not exceed their parents 'max-limit' value
- Child's max-limit should not exceed the parents max-limit
- **The parent will satisfy the children's Limit-at values first, then any remaining bandwidth is distributed by priority to satisfy the max-limit values of each child queue.**



Name	Packet Marks	Priority	Bucket Size	Limit...	Max...
download		8	0.100		80M
download_pri_1	TCP.ACK	1	0.100	5M	80M
download_pri_2	RTP	2	0.100	1M	80M
download_pri_3	SIP	3	0.100	1M	80M
download_pri_4	MGR.VPN	4	0.100	5M	80M
download_pri_5	HTTP	5	0.100	20M	80M
download_pri_8	no-mark	8	0.100	48M	80M
upload		8	0.100		20M
upload_pri_1	TCP.ACK	1	0.100	5M	20M
upload_pri_2	RTP	2	0.100	1M	20M
upload_pri_3	SIP	3	0.100	1M	20M
upload_pri_4	MGR.VPN	4	0.100	5M	20M
upload_pri_5	HTTP	5	0.100	5M	20M
upload_pri_8	no-mark	8	0.100	3M	20M

Queue colors in Winbox:

0% - 50% of max-limit – green

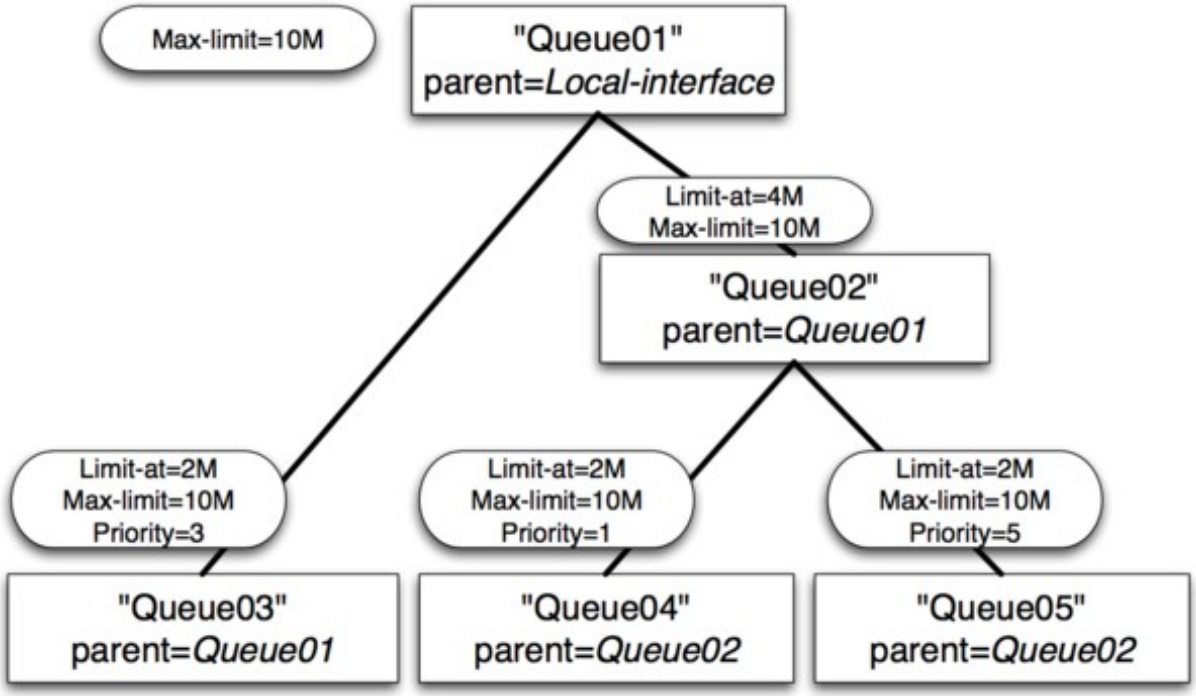
51% - 75% of max-limit – yellow

76% - 100% of max-limit - red

Queue settings

Check to verify config is correct:

- ✓ Max-limits do not exceed the parent max-limit
- ✓ Sum of child queue limit-at's do not exceed the parents max-limit
- Child limit-at will be satisfied
- 3 child queues x 2M = 6M
- 10M (max-limit) – 6M = 4M to distribute by priority
- Queue04 has highest priority so remaining bandwidth will be offered to queue04 first
- Queue04 gets 6M total



Buckets

Buckets

- A Bucket's purpose is to facilitate bursting
- "Bursting" is when traffic is allowed to exceed max-limit for a limited amount of transfer or time
- When traffic flow is less than max-limit, the bucket will fill with tokens
- A full bucket will allow bursting at an unrestricted speed, until the bucket is empty.
- If a child queue requests bandwidth from a parent queue who has a full bucket, The parent will release all tokens at once, allowing the child to burst

Bucket capacity

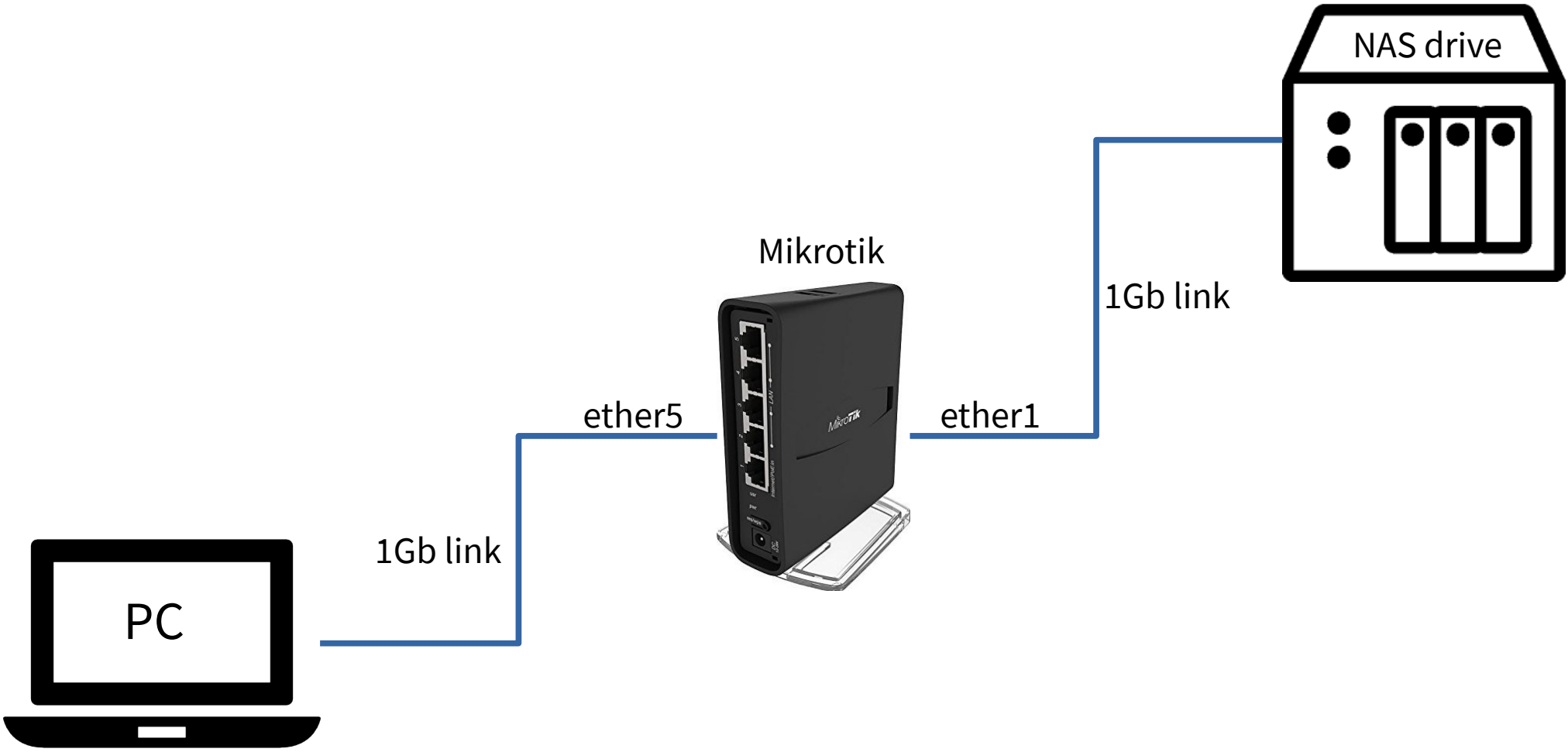
- Queues are configured with buckets that hold tokens (how many)
- max-limit x bucket size = bucket capacity
- Bucket capacity dictates data transfer. NOT FLOW or BANDWIDTH!
- All children are limited to the parents token supply

HTB LAB

Lab 1

Demonstrate a full bucket burst from a queue without any children

HTB LAB



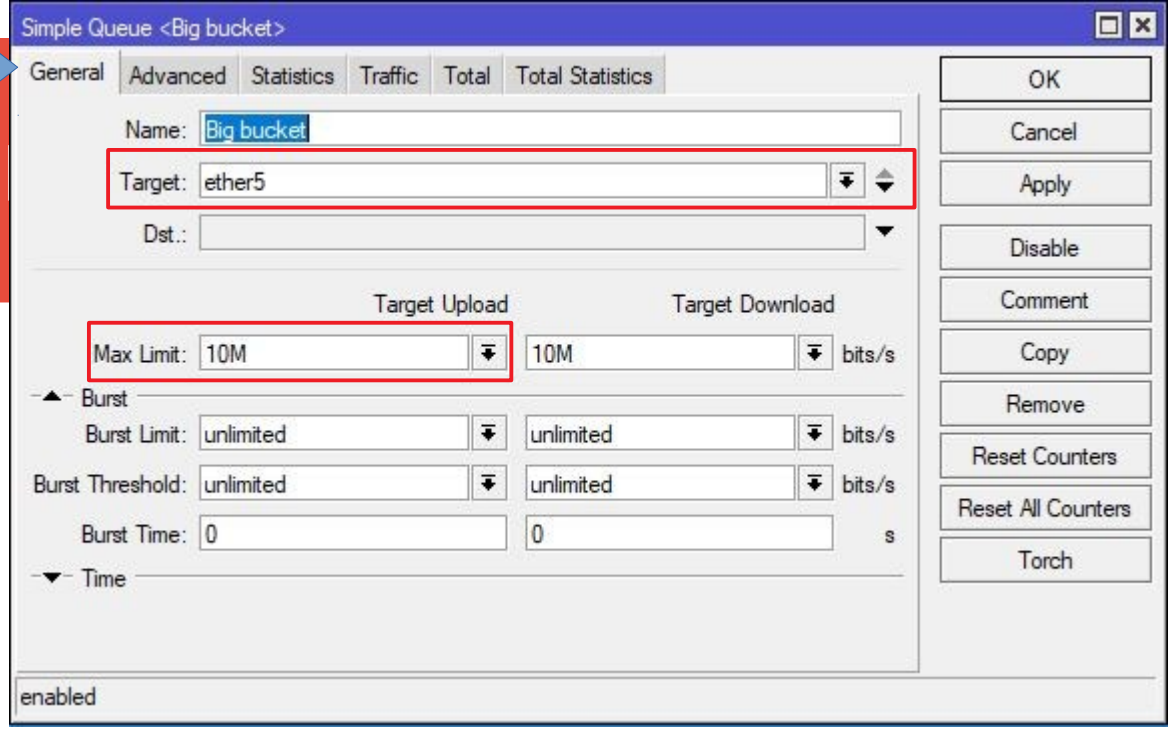
Queues set on ether5

HTB LAB

- Max-limit = 10Mbps
- Bucket size = 10
- Bucket capacity = bucket is set to burst 100Mb OF DATA TRANSFER!!!

HT

General

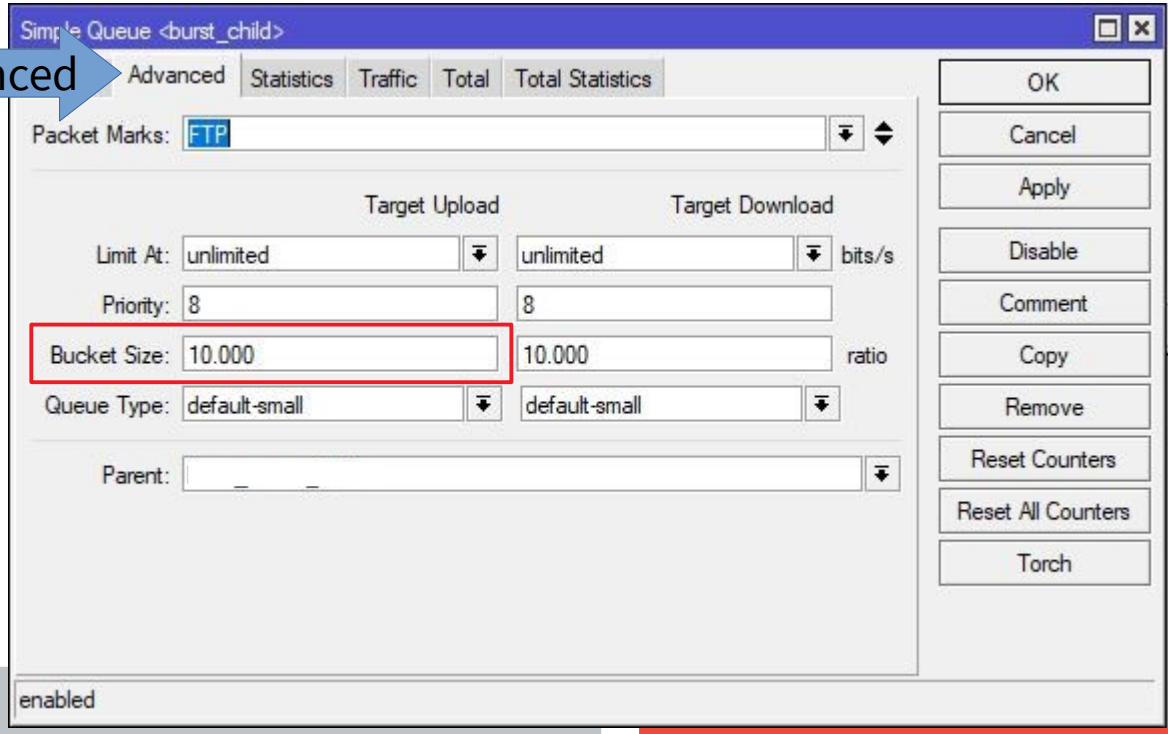


Target = ether5

Max-limit = 10M

Bucket size = 10

advanced



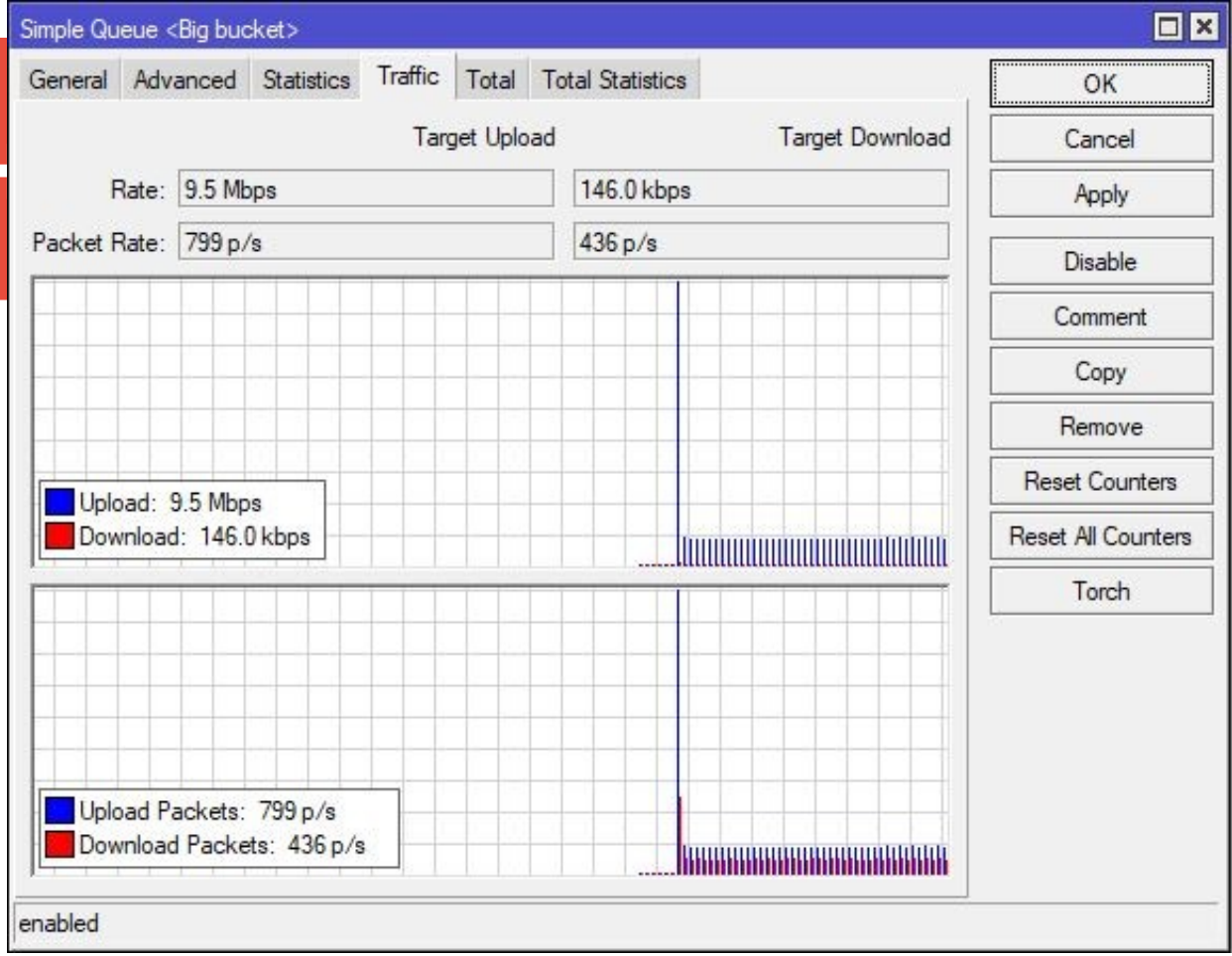
Bucket is set to burst 100Mb of DATA TRANSFER as fast as possible

H

Max-limit = 10M

Bucket size = 10

Bucket capacity = 100Mb of data transfer



268MB file = 2,248,146,944 bits

Bucket size = 104,857,760

2,248,146,944 – 104,857,760 = 2,143,289,184

2,143,289,184 / 10Mbps (10,485,760) = 204.4 seconds

204.4 seconds / 60 minutes = (3.41) 3:25 minutes + 1 second (from burst of 100Mb)

3:26 minutes for total file transfer

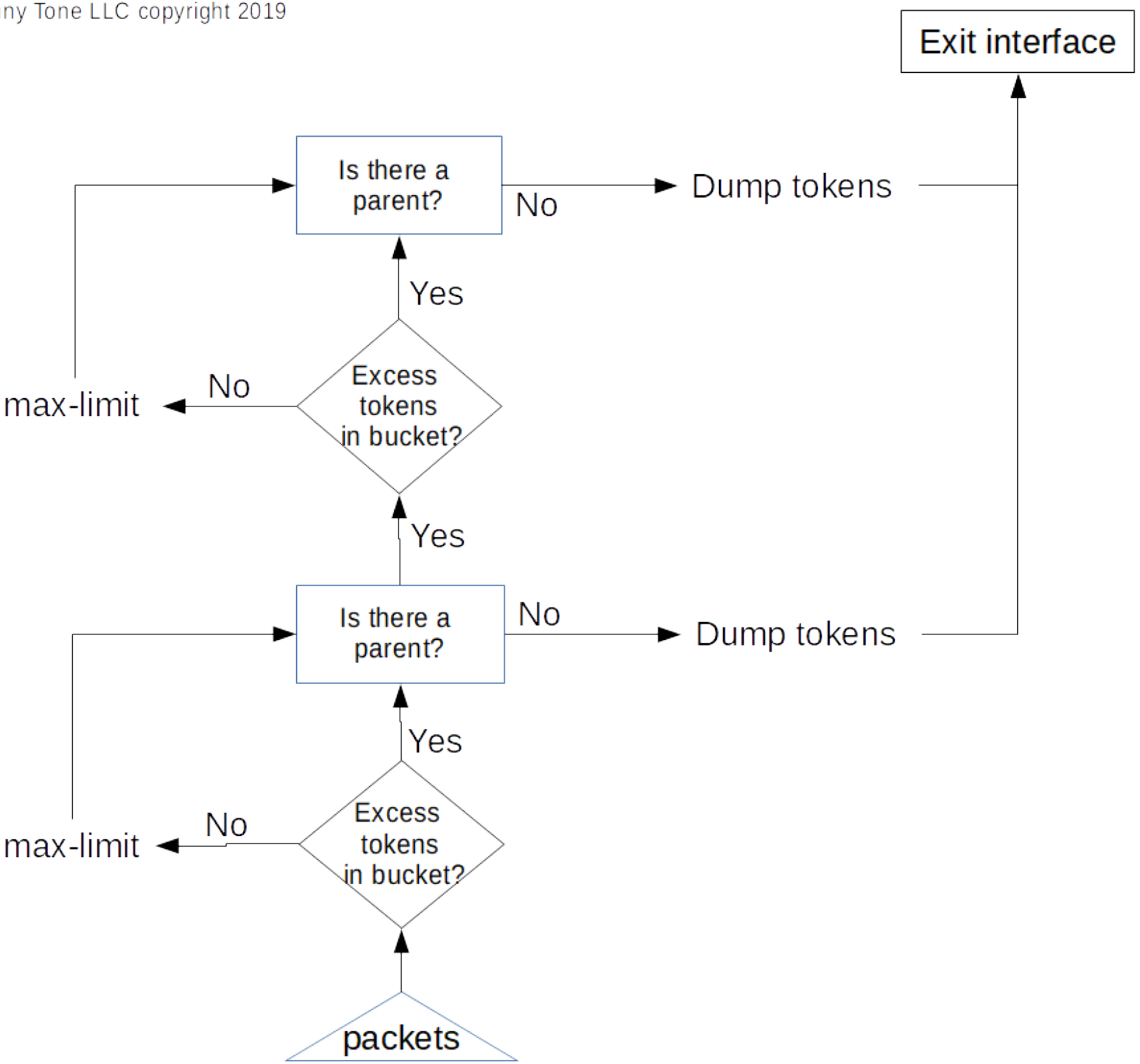
210 seconds / 60 = 3:30

Status: File transfer successful, transferred 268,435,456 bytes in 210 seconds

HTB LAB

Lab

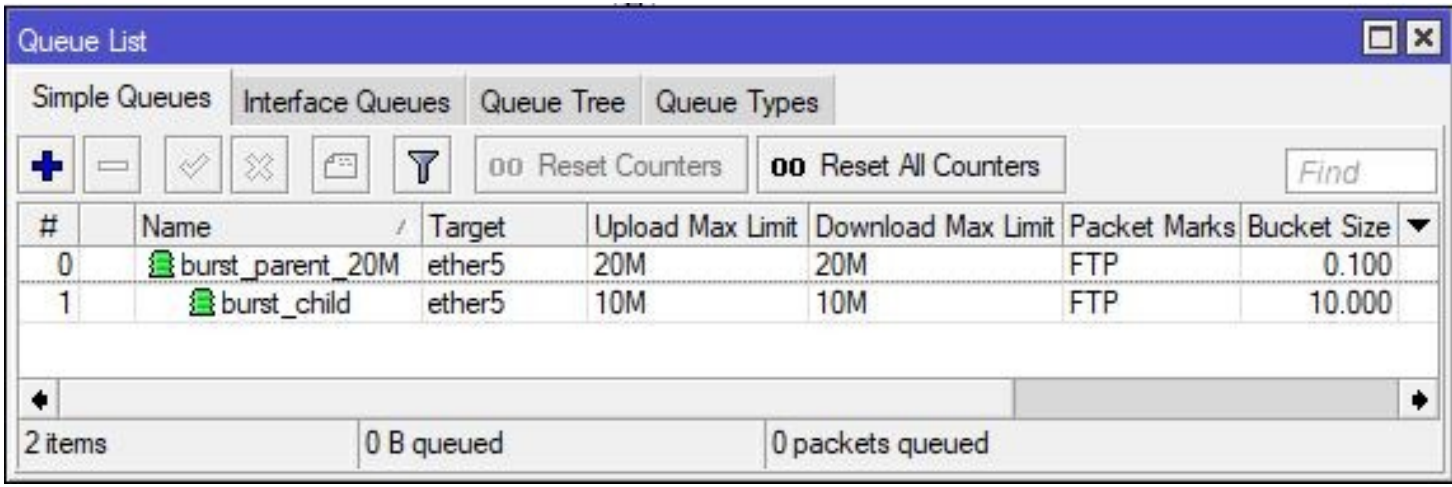
Demonstrate a burst from a child queue with a large bucket and it’s parent with a very small bucket



HTB LAB

Parent queue:

Target = ether5
Max-limit = 20M
Bucket Size = .1

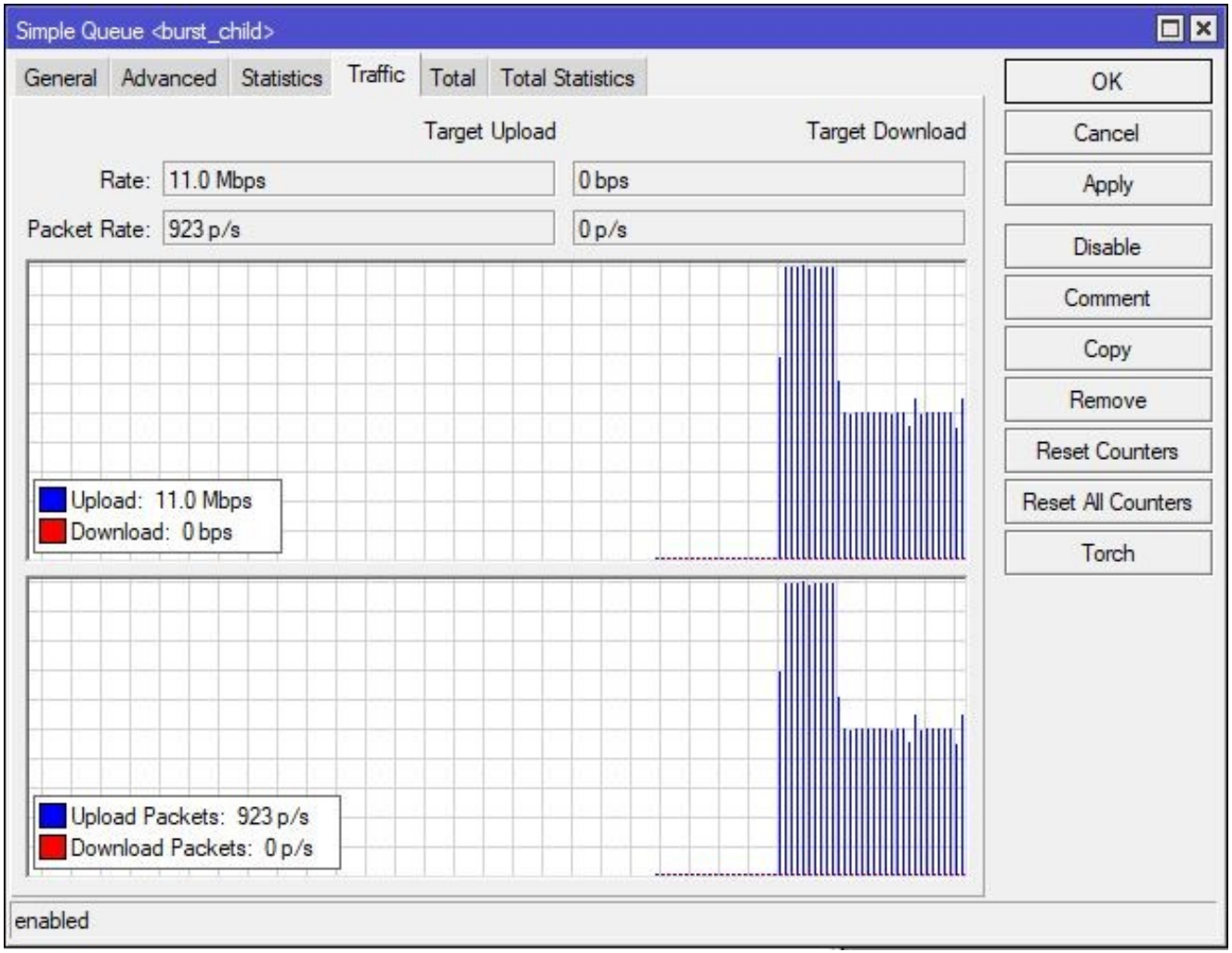


Child Queue:

Target = ether5
Max-limit = 10M
Bucket Size = 10

Bucket capacity is set for 100Mb data transfer as fast as possible

HTB LAB


Simple Queue <burst_child>
General
Advanced
Statistics
Traffic
Total
Total Statistics
OK
Cancel
Apply
Disable
Comment
Copy
Remove
Reset Counters
Reset All Counters
Torch
Target Upload
Target Download
Rate: 11.0 Mbps
0 bps
Packet Rate: 923 p/s
0 p/s
Upload: 11.0 Mbps
Download: 0 bps
Upload Packets: 923 p/s
Download Packets: 0 p/s
enabled

HTB LAB

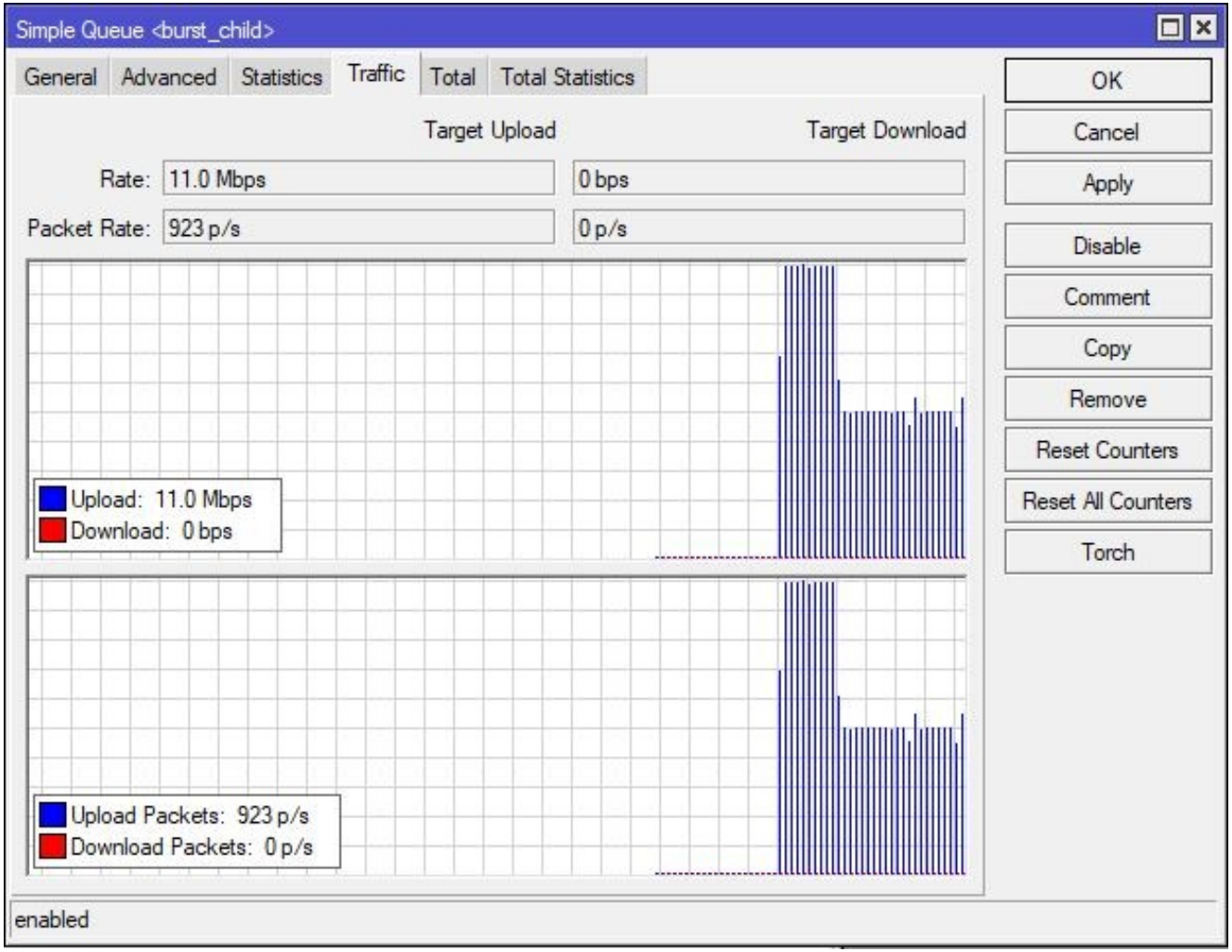
Child queue has a full bucket

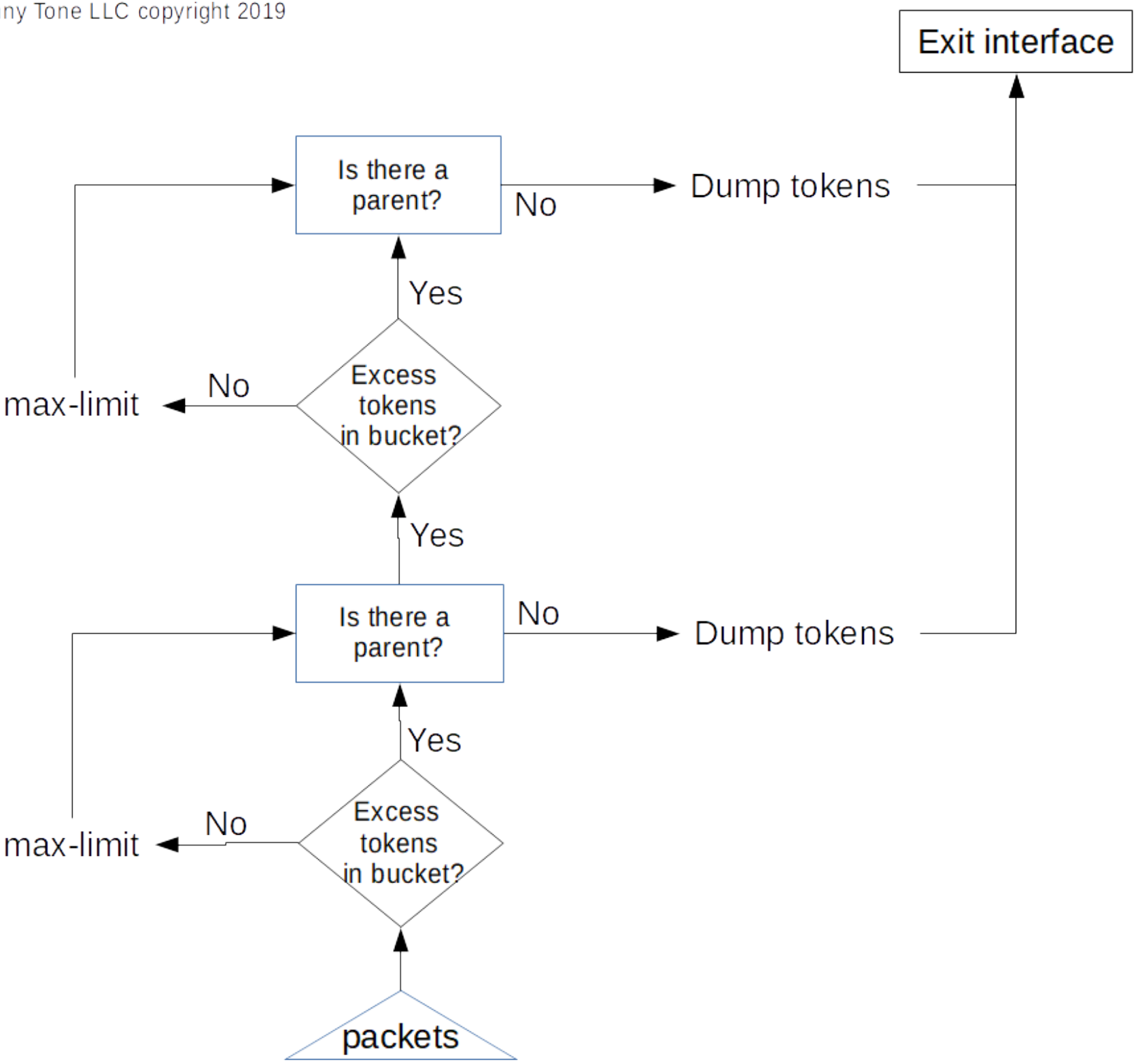
The parent has empty bucket

Child bucket is set to burst 100Mb of data transfer

But is limited to parents max-limit of 20Mbps

After 100Mb of data transfer has completed, data transfer rate returns to non bursting of child's max limit





Conclusion

We've learned about:

- Mangle
- Queuing
- Simple queues vs Queue trees
- Policing vs Scheduling
- Hierarchial Token Bucket (The Mikrotik Sasquatch)
- Bursting with buckets

Thank you'z

- The Brothers WISP / Greg Sowell
- Justin Miller – Why not to burst netflix traffic
- Nick "spock" Arellano – Telling me when I'm wrong
- Rick Frey – Token bucket theory
- Janis Megis - Token bucket theory
- Tommy "C" – Help with flow chart
- My wife and kid – being supportive and help me recharge

References

http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/06.06.htm

https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

http://linux-ip.net/articles/Traffic-Control-HOWTO/

https://wiki.mikrotik.com/wiki/Manual:HTB

https://lartc.org/howto/lartc.qdisc.html

https://lartc.org/howto/lartc.qdisc.classless.html#AEN691

https://wiki.debian.org/TrafficControl#Queueing_Disciplines

https://www.youtube.com/watch?v=dSEEwHCvOnA

https://www.youtube.com/watch?v=lXWQ3t7OL1Y