Understanding throughput

OR

Common misconceptions on what is ’real’ device *throughput*

MUM, USA 2019

Objectives

- To help you to take a deeper, different look into "well known" basic concepts

- To allow MikroTik equipment do more

- Encourage not only to update RouterOS version but also update existing configurations to use the latest features

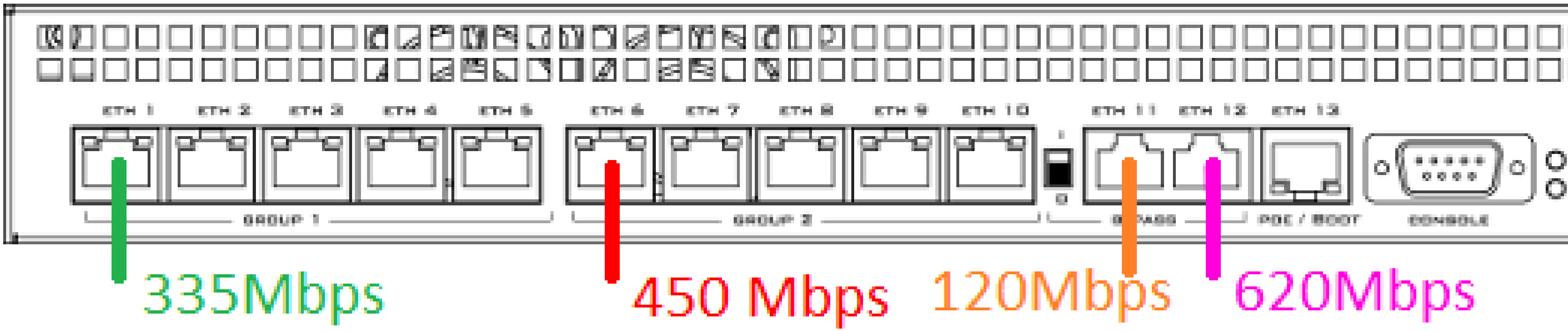
- Reduce the amount of throughput related emails to support@mikrotik.com!

What is throughput?

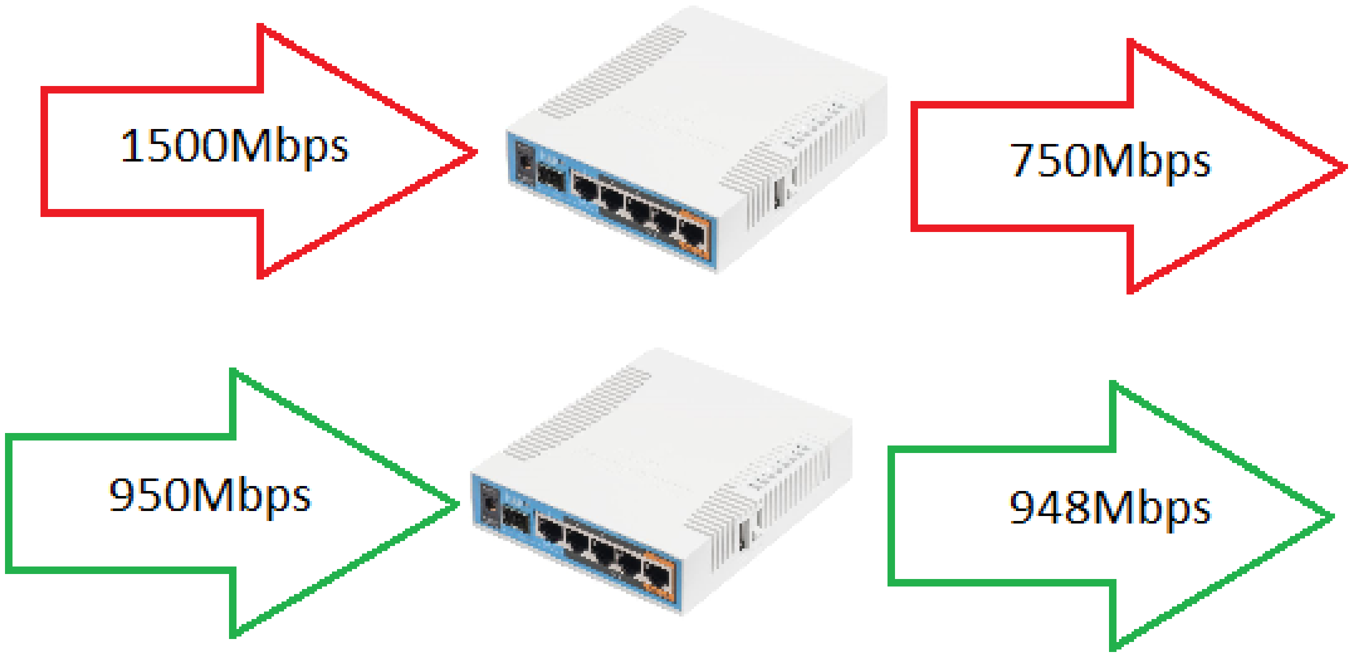
- Throughput is a measure of how many units of information a system can process, in a given amount of time
- In data transmission, network throughput is the amount of data transferred successfully from one place to another in a given time period and typically measured in bits per second (bps)

Router throughput

- Successfully transferred data through the router is equal to sum of all data that is leaving the router (not dropped in the router)



Maximum router throughput



RB962UiGS-5HacT2HnT		QCA9558 1G all port test					
Mode	Configuration	1518 byte		512 byte		64 byte	
		kpps	Mbps	kpps	Mbps	kpps	Mbps
Bridging	none (fast path)	161.9	1,966.1	345.5	1,415.2	435.6	223.0
Bridging	25 bridge filter rules	125.5	1,524.1	126.5	518.1	128.8	65.9
Routing	none (fast path)	161.9	1,966.1	334.2	1,368.9	397.9	203.7
Routing	25 simple queues	161.9	1,966.1	183.9	753.3	203.3	104.1
Routing	25 ip filter rules	78.1	948.4	77.8	318.7	77.5	39.7

Link or wire speed

- Link or Wire speed refers to the rate of data transfer a given telecommunication technology provides at the physical wire level
  - Ethernet - 10Mbps/100Mbps/1Gbps/2.5Gbps/5Gbps/10Gbps
  - SFP – 1Gbps
  - SFP+ - 10Gbps
  - SFP28 - 25Gbps
  - QSFP+ - 40Gbps
  - QSFP28 - 100Gbps

Wire-speed

- Wire-speed, as an adjective, describes any hardware or function that supports data transfer rate without slowing it down
- Functions embedded in microchips (ASIC) are more likely to run at wire speed than functions that are implemented in software
- Currently in MikroTik hardware all devices with a switch chip are capable of transferring data at wire-speed while using an impressive set of features

Wireless ”wire speed” (Data rates)

- Maximum *theoretical* wireless speed is determined by wireless protocol, number of streams, modulation and channel width

802.11n			**Channel width**				**802.11ac**
HT MCS	**Streams**	**Modulation**	**20MHz**	**40MHz**	**80MHz**	**160MHz**	**VHT MCS**
7	*1*	64-QAM 5/6	72.2	***150***	325	650	7
	1	256-QAM 5/6	n/a	200	***433.3***	866.7	9
15	*2*	64-QAM 5/6	144.4	***300***	650	1300	7
	2	256-QAM 5/6	n/a	400	***866.7***	***1733.3***	9
23	*3*	64-QAM 5/6	216.7	***450***	975	1950	7
	3	256-QAM 5/6	288.9	600	***1300***	n/a	9
31	*4*	64-QAM 5/6	288.9	***600***	1300	2600	7
	4	256-QAM 5/6	n/a	800	**1733.3**	3466.7	9

802.11ad Data rates

MCS	Modulation	NCBPS	Repetition	Code rate	Data rate (Mbps)
0	DBPSK			1/2	27,5
1	π/2 BPSK	1	2	1/2	385
2	π/2 BPSK	1	1	1/2	770
3	π/2 BPSK	1	1	5/8	962.5
4	π/2 BPSK	1	1	3/4	1155
5	π/2 BPSK	1	1	13/16	1251.25
6	π/2 QPSK	2	1	1/2	1540
7	π/2 QPSK	2	1	5/8	1925
8	π/2 QPSK	2	1	3/4	**2310**
9	π/2 QPSK	2	1	13/16	2502.5
10	π/2 16QAM	4	1	1/2	3080
11	π/2 16QAM	4	1	5/8	3850
12	π/2 16QAM	4	1	3/4	4620

Marketing numbers

802.11ac Quad Stream plus 802.11ad WiFi
Up to 4600+1733+800 Mbps† wireless speed

Performance

208 Gbps on 24-port Gigabit Ethernet model

256 Gbps on 48-port Gigabit Ethernet model

640 Gbps on 24-port Multigigabit Ethernet model

580 Gbps on 48-port 2.5G (12 Multigigabit) Ethernet model

640 Gbps on 48-port 5G Ethernet model

All models are wire-speed nonblocking performance

- Implicit/Explicit Beamforming for 2.4 & 5 GHz bands (1,733 + 866 + 400 Mbps)†

Capacity

- Network device capacity is a measure of the device structure's bandwidth
  - Example: **RBwAPG-5HacT2HnD –** has a structure of one Gigabit Ethernet and one triple stream 802.11ac wireless and one dual stream 802.11n wireless

    1.3G + 0.3G +1G * 2 (for duplex) = **4.6Gbps**
  - Example: **CRS309-1G-8S+IN** – has a structure of one Gigabit Ethernet and 8x 10Gbps SFP+ ports

    (8x10G + 1x1G) * 2 (for duplex) = **162Gbps**

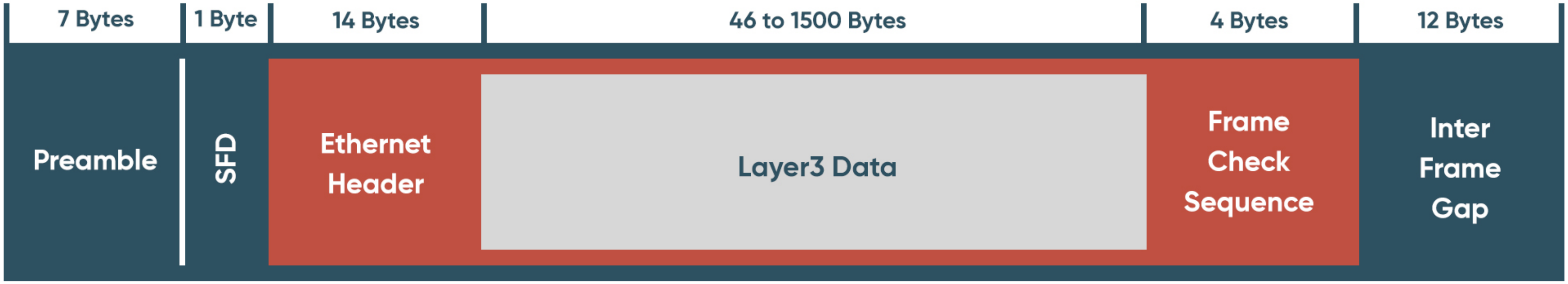
Layer1 wire-speed vs. Layer2 wire-speed

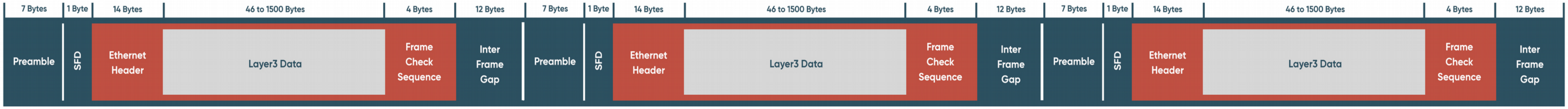
Switching results

CRS309-1G-8S+IN							
Mode	Configuration	1518 byte		512 byte		64 byte	
		kpps	Mbps	kpps	Mbps	kpps	Mbps
Switching	Non blocking Layer 2 throughput	6,583.2	79,946.7	19,032.0	77,954.9	120,535.7	61,714.3
Switching	Non blocking Layer 2 capacity	6,583.2	159,893.4	19,032.0	155,909.8	120,535.7	123,428.6
Switching	Non blocking Layer 1 throughput	6,583.2	81,000.0	19,032.0	81,000.0	120,535.7	81,000.0
Switching	Non blocking Layer 1 capacity	6,583.2	162,000.0	19,032.0	162,000.0	120,535.7	162,000.0

OSI Layer1

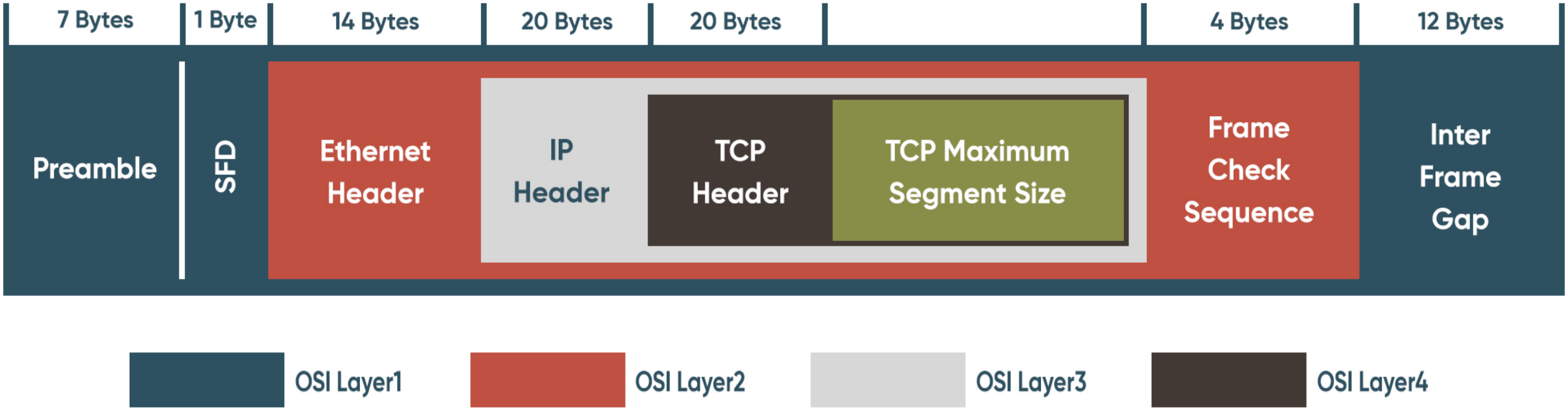
- Ethernet is a self-clocked digital protocol. The clock is synchronized from preamble field that provides a predictable 7 bytes long signal for Ethernet receiver, followed by 1 byte Start-of-Frame Delimiter
- Ethernet specifies minimum idle period between transmission of Ethernet frames known as the interframe gap – the time it takes to send 12 bytes





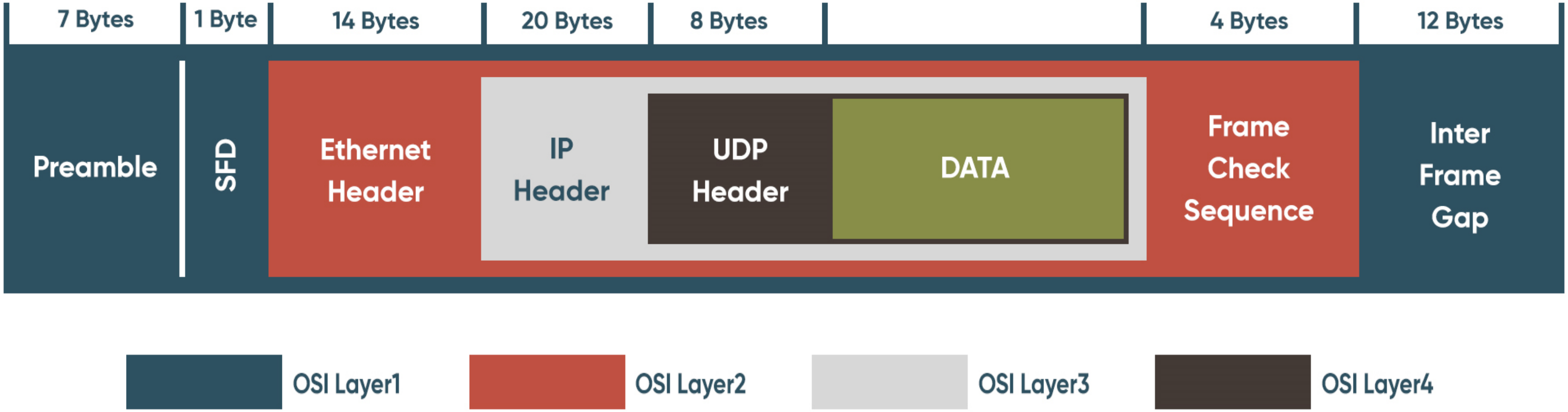
OSI Layers and wire-speed

- TCP ACK packet without any options takes
  - 40B as Layer3 packet (padded to 46B to reach minimal Ethernet payload size)
  - 46+18 = 64B as Layer2 frame
  - 64+20 = 84B as Layer1 signal on the wire
  - 40/84 = 47.62% IP transmission efficiency

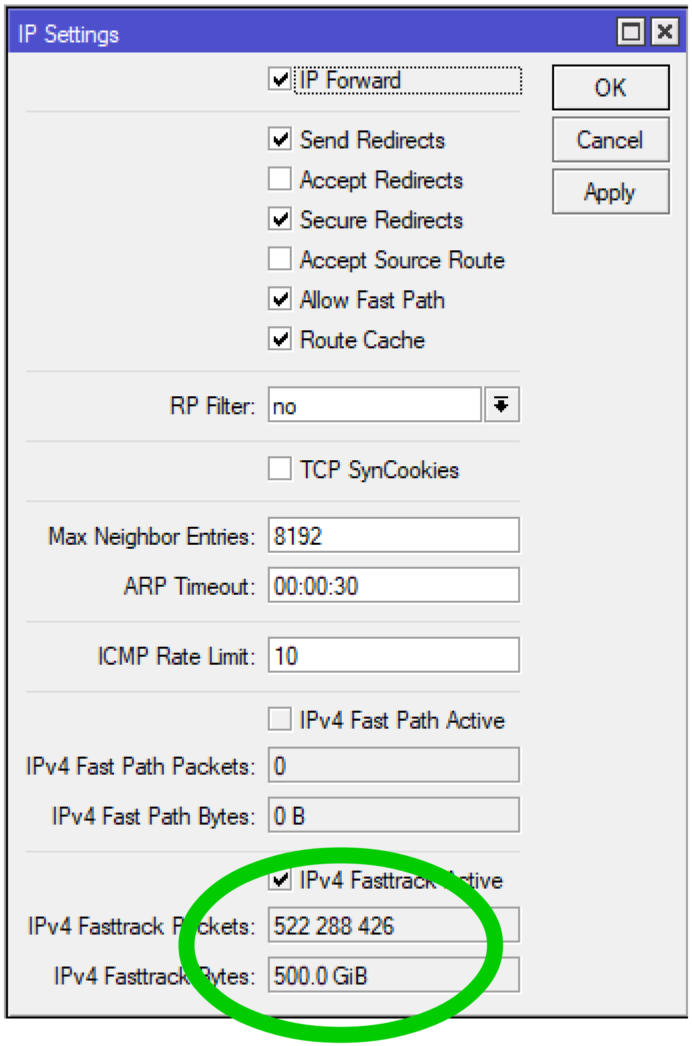


OSI Layers and wire-speed

- UDP (and TCP) data packet takes
  - 1500B as Layer3 packet
  - 1500+18 = 1518B as Layer2 frame (1514B in Wireshark)
  - 1518+20 = 1538B as Layer1 signal on the wire
  - 1500/1538 = 97.53% IP transmission efficiency



My wire-speed



- 500GiB/522’288’426 =1028B as Layer3 packet
- 1028+18 = 1046B as Layer2 frame (1042B in Wireshark)
- 1046+20 = 1066B as Layer1 signal on the wire
- 1028/1066= 96.43% IP transmission efficiency

Note: 1 GiB = 1024^3 B
1 GB = 1000^3 B

Tricky questions about throughput

- What is **1Gbps** really? Is it the same as **1Mb/ms** for 1000ms? Is it the same as **1b/µs** for 1 million microsecond?
- Are there any differences between Fast Ethernet and Gigabit Ethernet if you limit it to 60Mbps?
- Why can I get 22Mbps in my speedtest.net results if I have queue set to 20Mbps?

Lets scale up

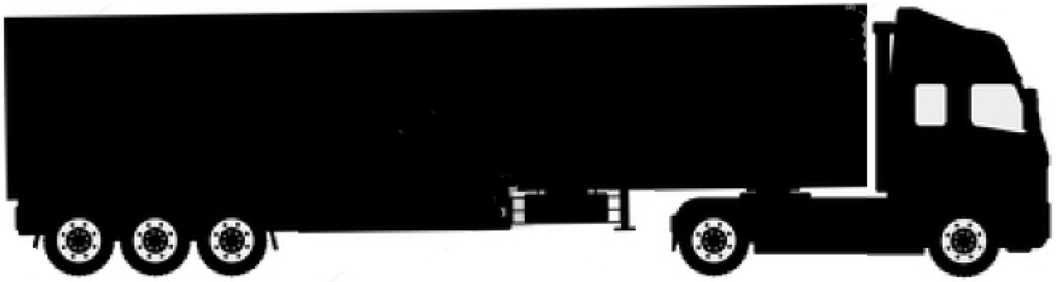
- Ethernet networks in many ways is very similar to highway system
  - Lets replace **Ethernet** with **highways**
  - Lets replace **Ethernet frames** with **cars/trucks**
  - Lets replace **Frame sizes** with **engine size**
  - Lets replace **Queues** with **tunnels**
  - Lets replace **Queues size** with **traffic jam size**
  - Lets replace seconds, with hours

Tunnel specifications

- Tunnel have limited ventilation
- It can handle 3600L/h (engine displacement Liters per hour)
- How would you control that?
- 3600L/h = 60L/min
  60L/min = 1L/s,
  right?

Limitations of scaling

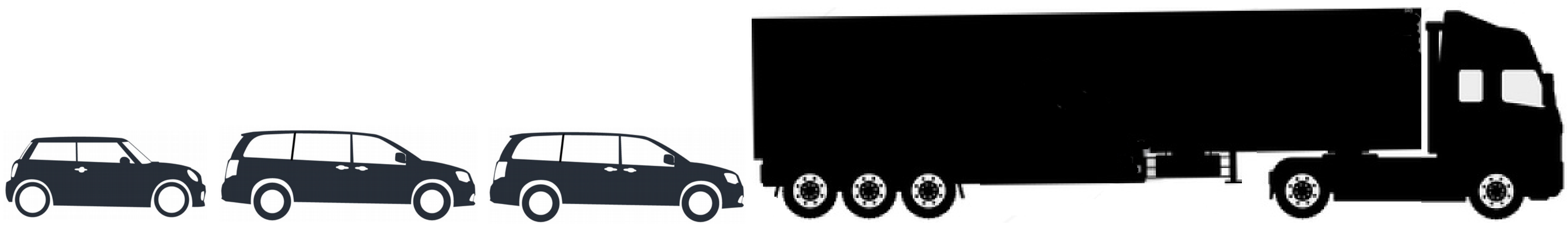
- What happens when truck arrives with a 12L engine?



- So we can’t scale down to 1L/s, we need to cope with inherit burstiness caused by randomness of engine sizes
- Lets go with 60L/min

Limitations of scaling

- What happens when for 2 minutes there are no cars, and then row of cars with worth of 150L arrive that were stuck behind lorry?



  - Should all 150L be allowed to pass based on previous 2 minutes?
  - Should 120L be allowed to pass based on previous 1 minute? (and 30L queued)
  - Should only 60L be allowed to pass based on only this minute? (and 90L queued)

Where does the measurement start?

60	60	60	60	60	60	…		…	60	60	60	60	**0**	**0**	**150**	**∑ 3570**

Past *Present*

60	**0**	**0**	**150**	60	60	…		…	60	60	60	60	60	60	60	**∑ 3570**

Past *Present*

150	60	60	60	60	60	…		…	60	60	60	60	60	60	60	**∑ 3690**

60	60	60	60	60	60	…		…	60	60	60	60	**0**	**0**	**60**	**∑ 3480**

Past *Present*

60	60	**0**	**0**	**60**	60	…		…	60	60	60	60	60	60	60	**∑ 3480**

Past *Present*

60	60	60	60	60	60	…		…	60	60	60	60	60	60	60	**∑ 3600**

Waiting time

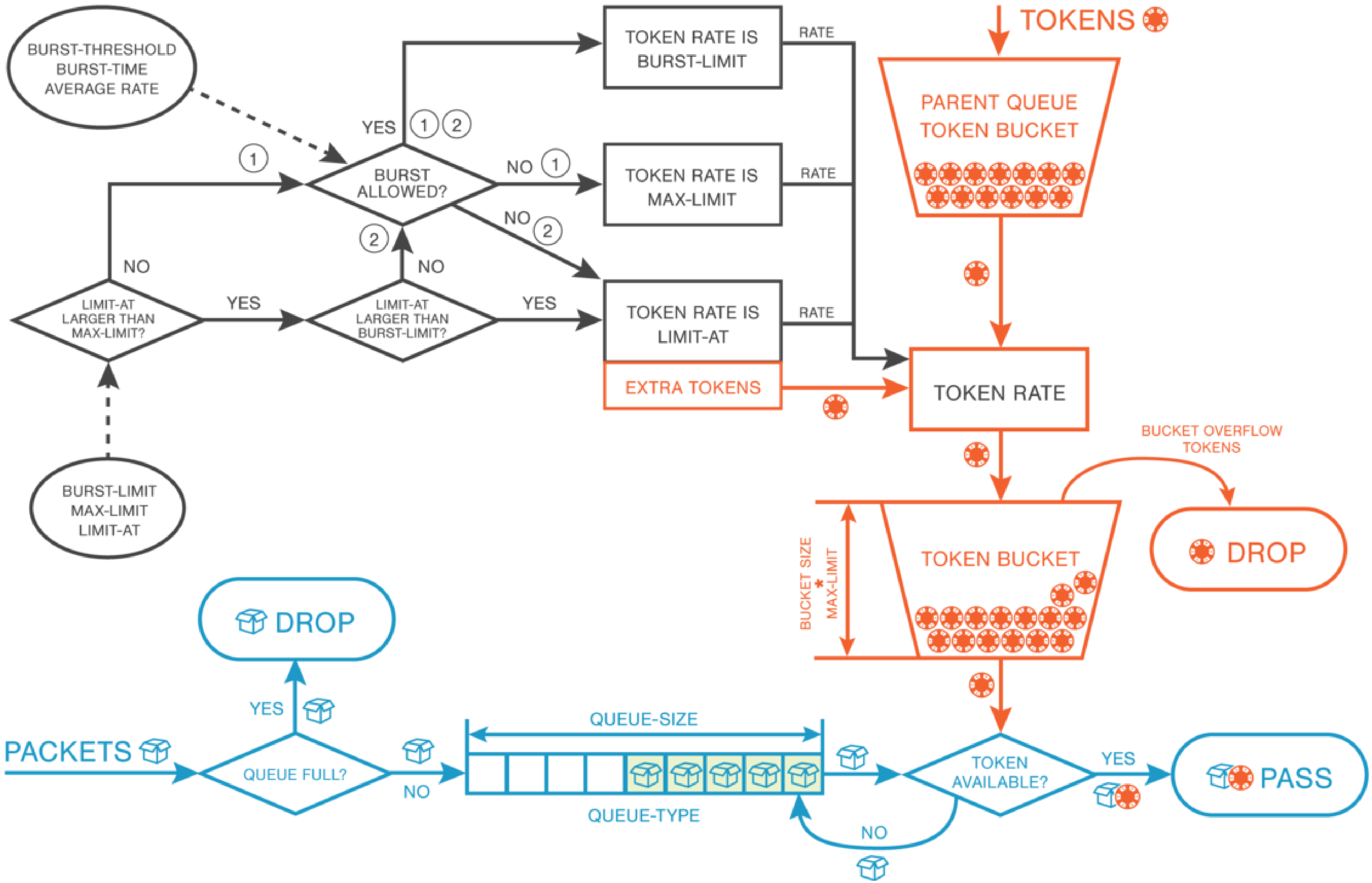
- So we can’t scale down to 60L/min ether!
- We can’t go just with 3600L/h ether, a car that will arrive with 3601st liter, will have to wait till the end of the hour, so that next 3600L are available.
- And again.. where that hour started exactly?
- Conclusion: Traffic is too chaotic and unpredictable to apply any type of scaling directly.

Solution - Tokens

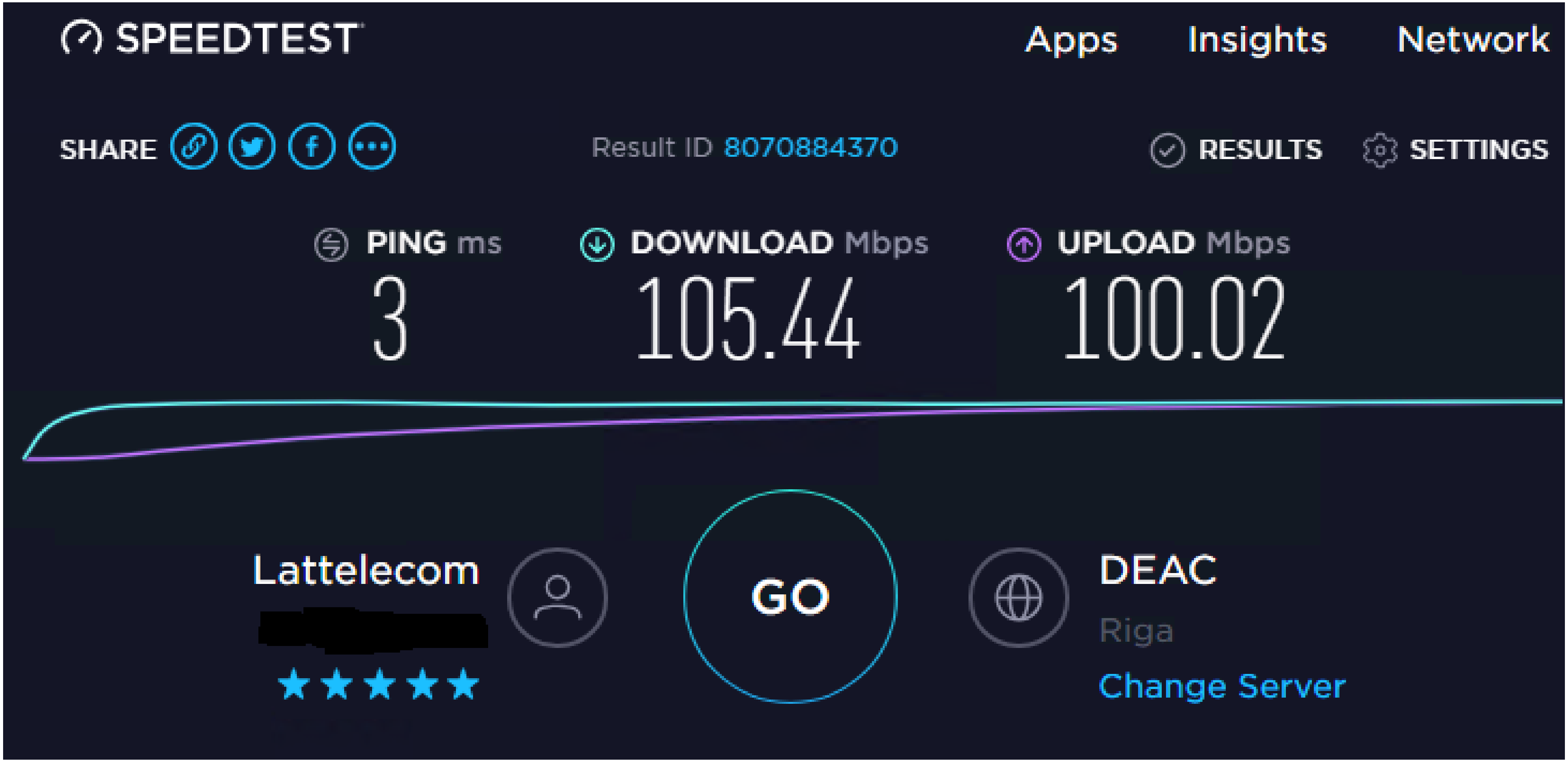
- We can place toll booth before the tunnel it will
  - generate tokens at any given scale, even ms
  - It will be able to accumulate limited number of tokens as a buffer
  - Cars will have to get tokens to pass

  - Buffer size should be taken into account when when token rate is specified, to stay below limit at any given time period

HTB Algorithm



“Real” network speed



How does speedtest.net work

- speedtest.net will use up to four HTTP threads
  - After the pre-test, if the connection speed is at least 4 megabits per second then speedtest.net will use four threads. Otherwise, it will default to two
- One side sends an initial chunk of data, the other side calculates the real-time speed of the transfers and adjusts the chunk size along with buffer size
  - All samples are sorted by speed. The two fastest results are removed and the bottom 1/4 which is then left. Everything else is then averaged to get the result

*https://support.speedtest.net/hc/en-us/articles/203845400-How-does-the-test-itself-work-How-is-the-result-calculated-

TCP theoretical throughput

TCP throughput is limited by two windows

- Congestion window - sender side
  - the amount of unacknowledged packets that may be in transit
  - it auto-tunes based on congestion control algorithms
  - impacted by packet loss and delays
- Receive window - receiver side
  - the amount of received data not processed yet by the application
  - it auto-tunes based or receive buffer size and its level of fullness

TCP theoretical throughput

- Sender and receiver continuously negotiate common transmission window (for both directions)
- Maximal single TCP stream throughput is limited to not more than one full transmission window within one round-trip time (RTT) period
- Middle devices, like routers, have only indirect impact on congestion algorithm used - by impacting RTT, jitter, packet loss

TCP theoretical throughput

```
Window size value: 148
[Calculated window size: 18944]
[Window size scaling factor: 128]
```

```
Window size value: 16504
[Calculated window size: 4225024]
[Window size scaling factor: 256]
```

```
Window size value: 1026
[Calculated window size: 262656]
[Window size scaling factor: 256]
```

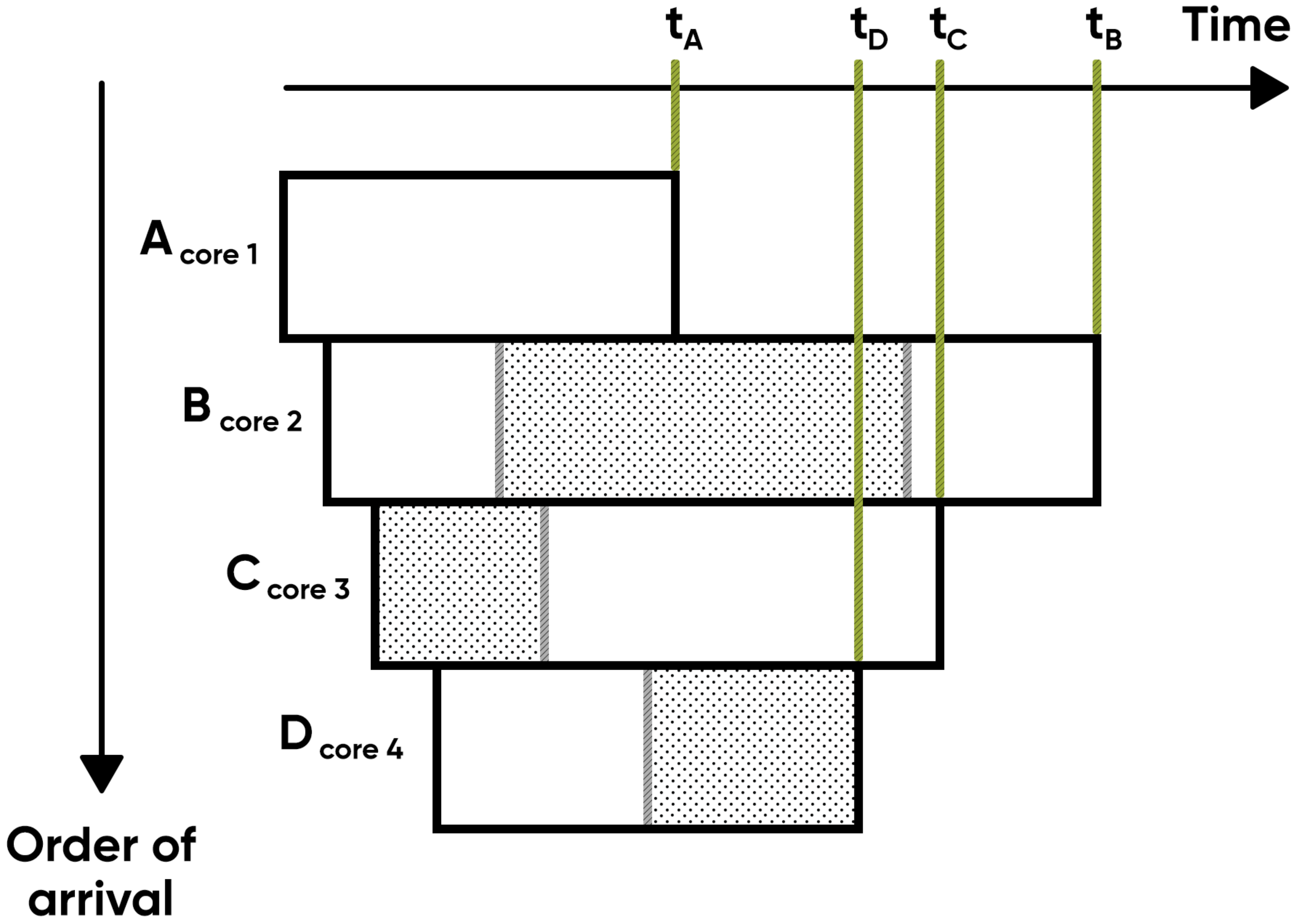
```
Window size value: 4117
[Calculated window size: 1053952]
[Window size scaling factor: 256]
```

- If window size is 4`225`024 bytes, and RTT is 60ms
  - (4`225`024*8)/60 = 563`336 bits/ms = 563Mbps
- If window size is 18`944 bytes, and RTT is 6ms
  - (18`944*8)/6 = 25`259 bits/ms = 25,26Mbps

TCP and multi-threading

- Multi-threading introduces out-of-order packets
- Usually TCP can handle out-of-order packets within a single transmission window
- Depending on the congestion algorithm used, out-of-order packets might be considered as loss and introduce delays (increase calculated RTT)

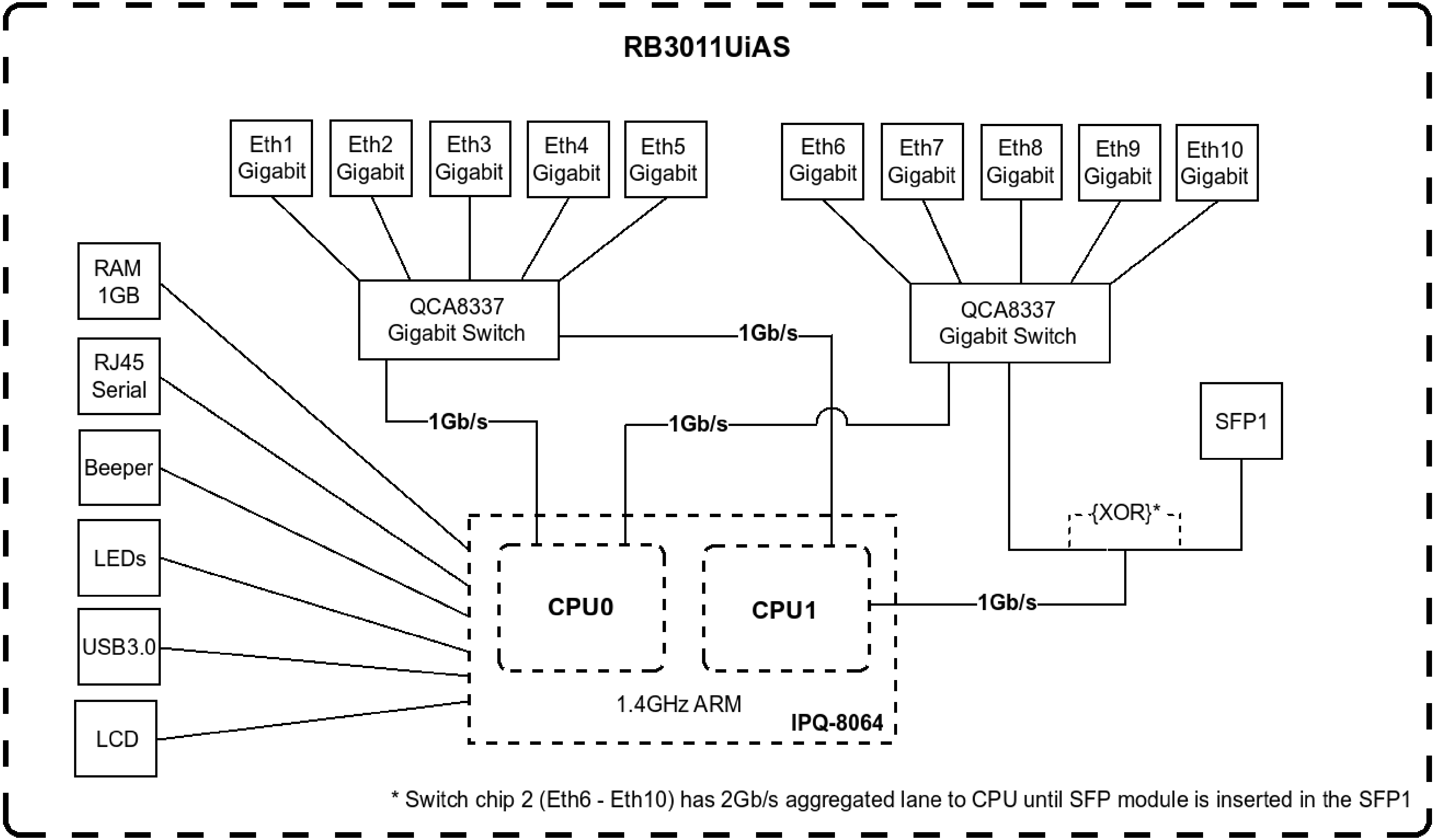
Single TCP stream and multi-threading



Flow/Packet steering

- RouterOS uses Receive Flow/Packet steering to assign incoming traffic to a specific CPU core/thread, based on the hash value
- The hashing process can be:
  - Hard-coded in the hardware
  - Configurable in the hardware
  - Implemented in the interface driver
- Receive flow/packet steering will try to keep single TCP stream bound to single CPU core/thread as long as possible

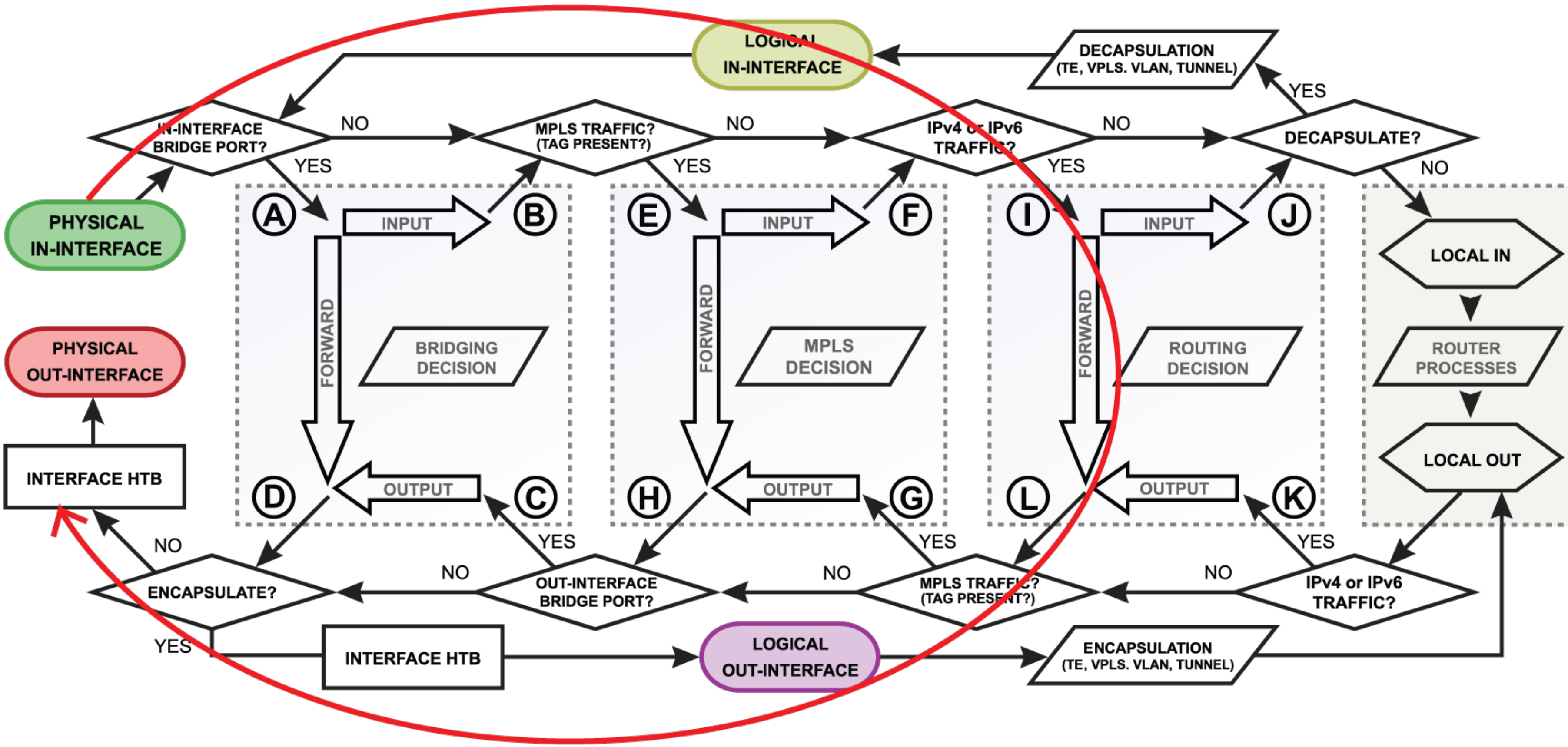
RB3011UiAS block diagram



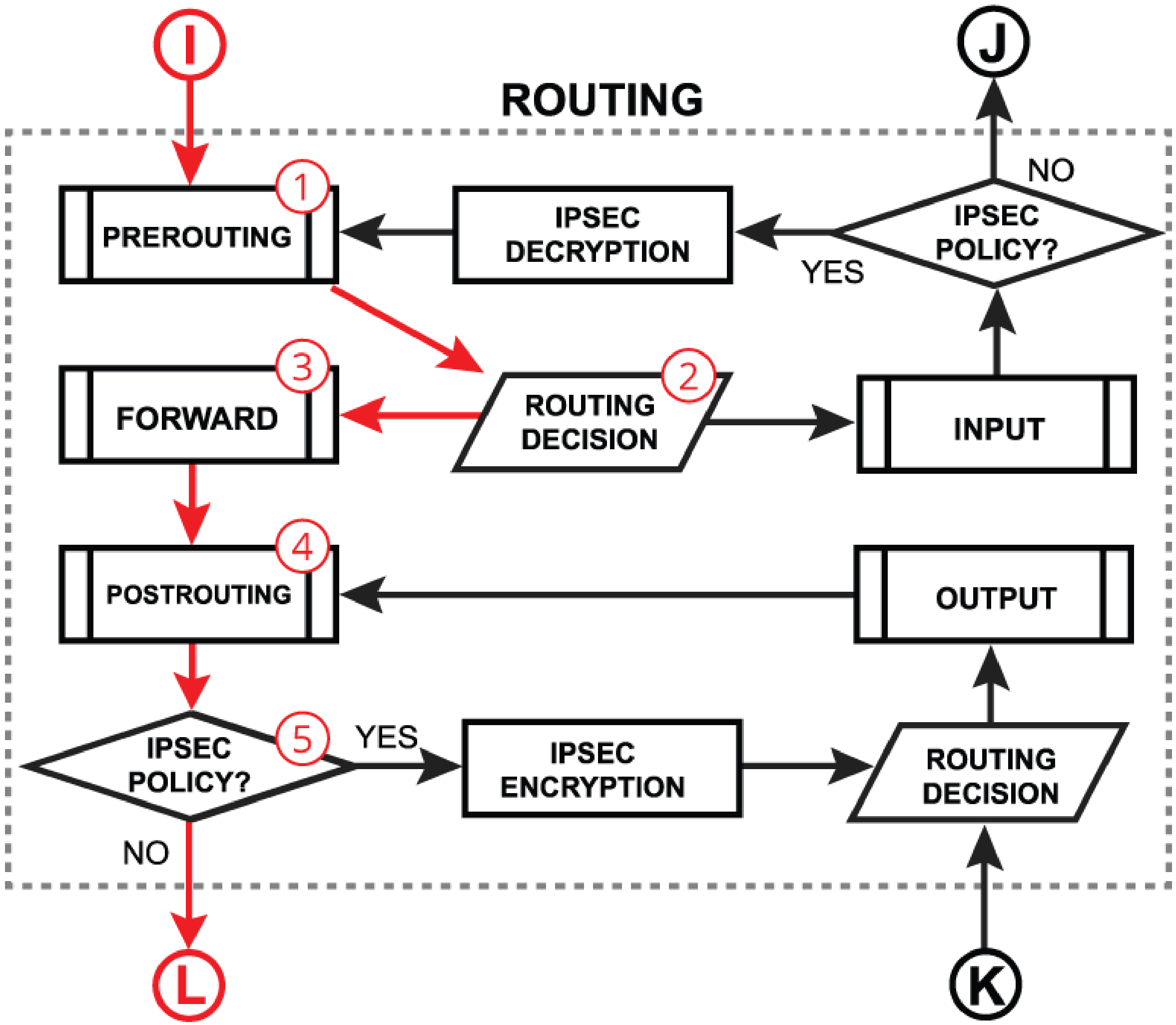
What can and can’t we do...?

  - We can’t choose the congestion control algorithm
  - We can’t determine the congestion and the receive window size of the endpoints
- We can change MSS
- We can minimize impact on RTT by reducing packet processing time
- We can impact packet loss

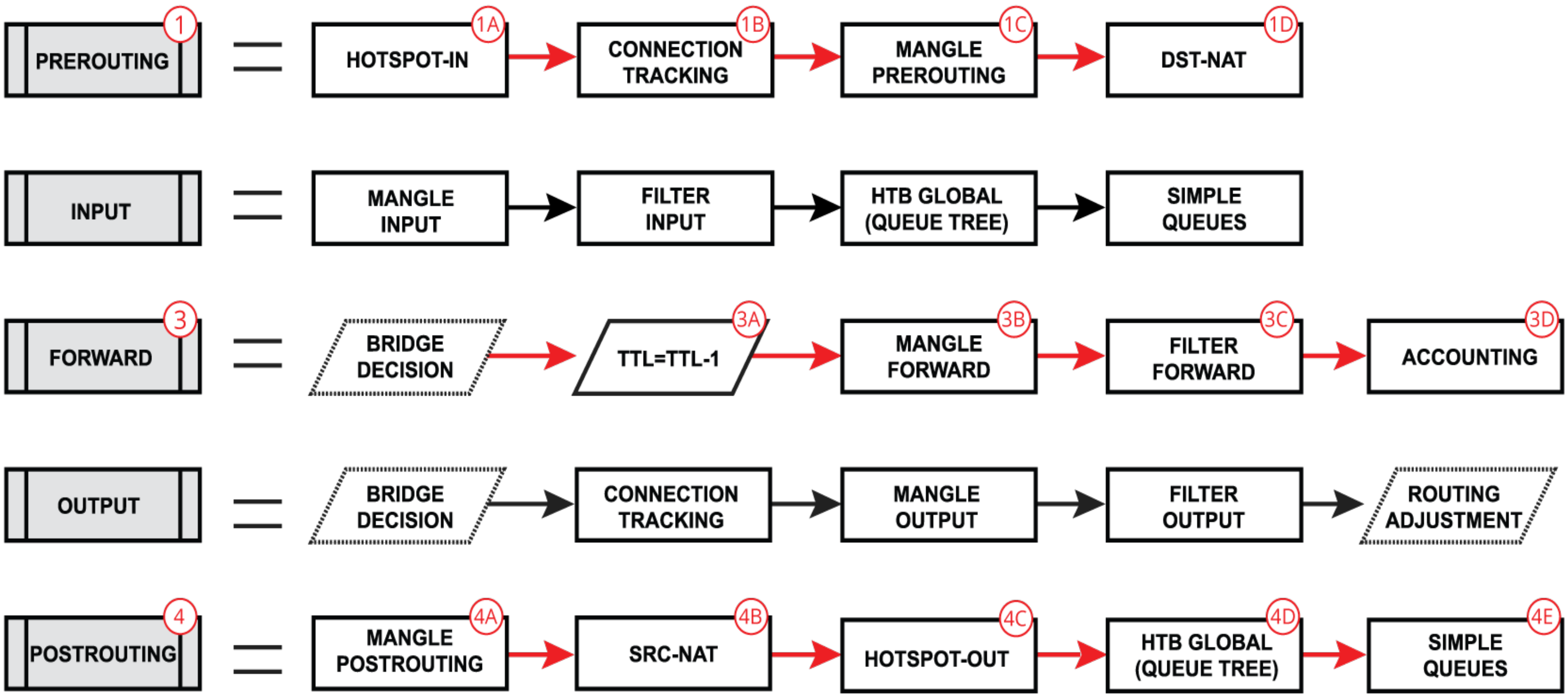
Routing forwarding



Routing forwarding



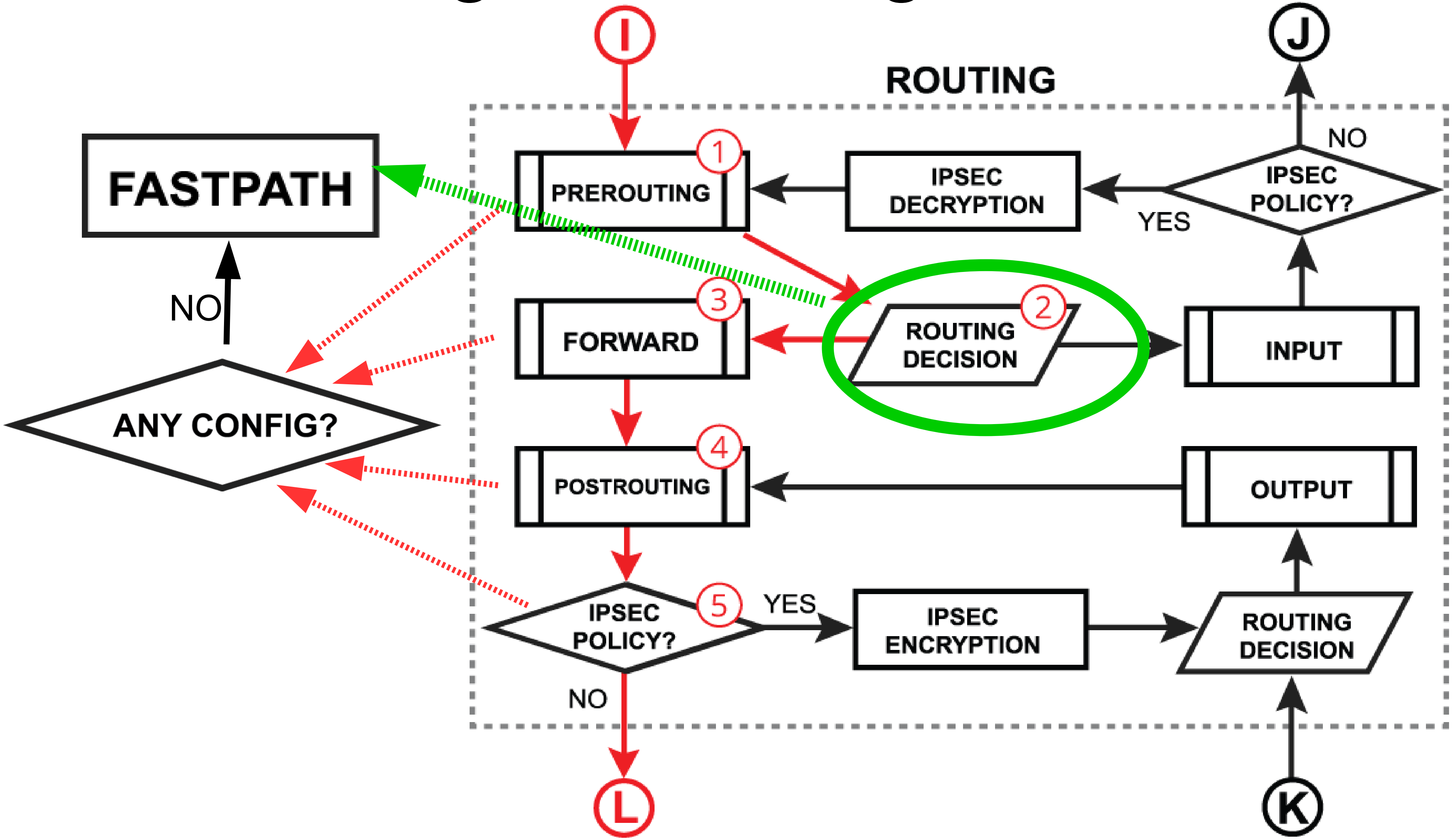
Routing forwarding



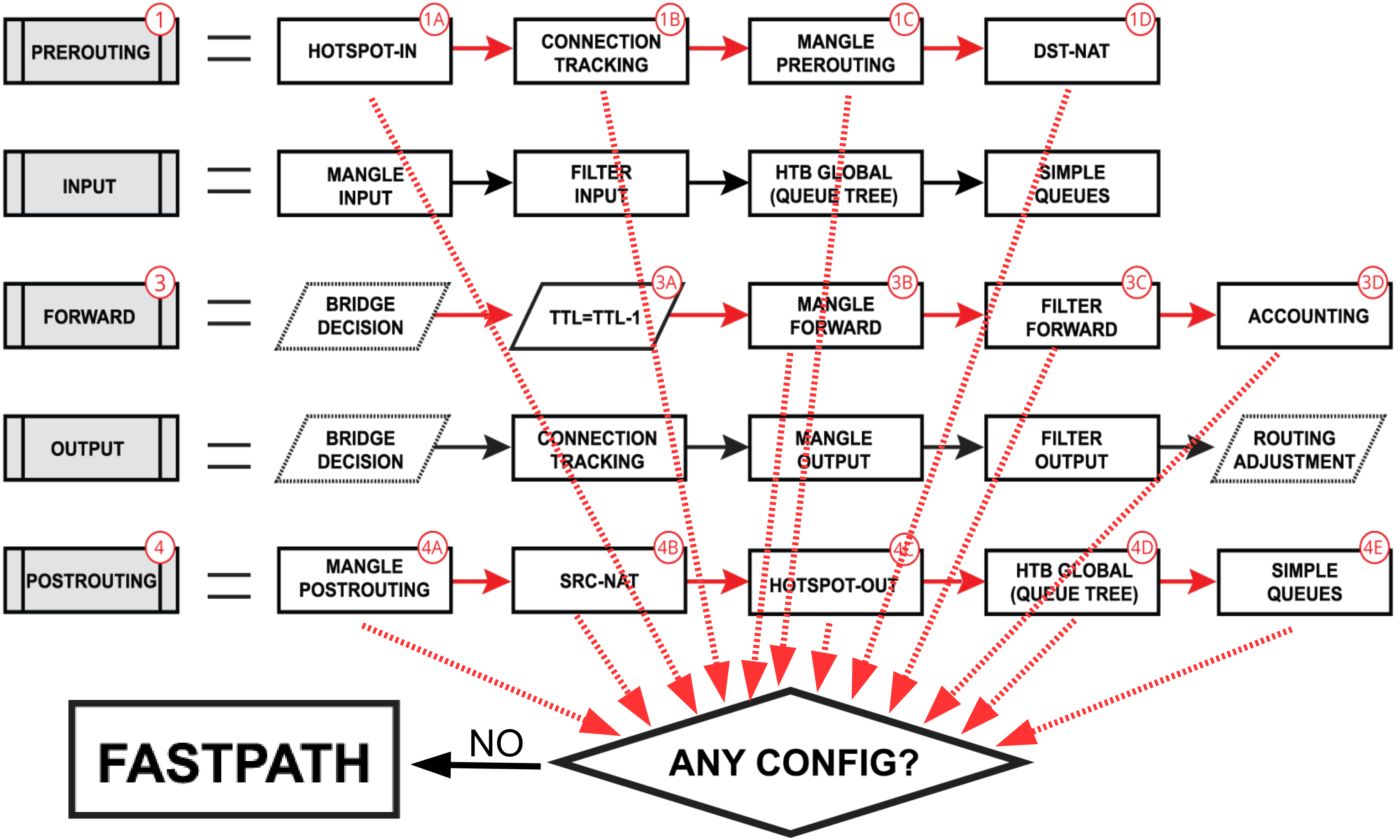
Initial FastPath implementation

- FastPath is an interface driver extension, that allows you to receive/process/send traffic without unnecessary processing
- Interface driver can now talk directly to specific RouterOS facilities - skipping others
- FastPath requirements
  - Interface driver support
  - FastPath should be allowed in configuration
  - No configuration in specific facilities

Routing forwarding FastPath



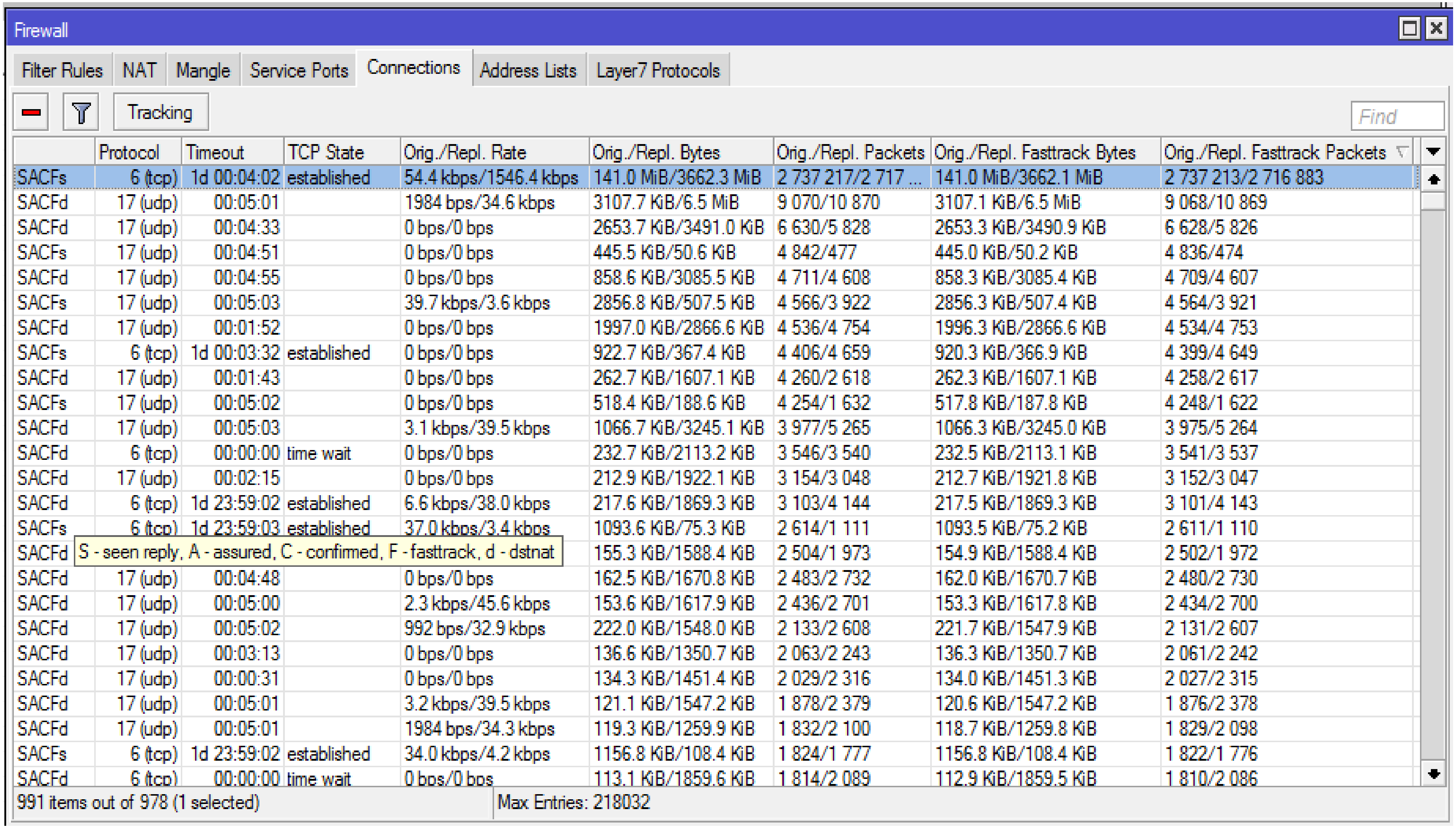
Routing forwarding FastPath



FastPath + Conntrack

- Implemented as “fasttrack-connection” action for firewall filter/mangle that adds “Fasttracked” flag to connection
- Packets from “Fasttracked” connections are allowed to travel in FastPath
- Works only with IPv4/TCP and IPv4/UDP
- Some packets will still follow the regular path to maintain timeouts in conntrack entries

FastPath + Conntrack = FastTrack



Firewall

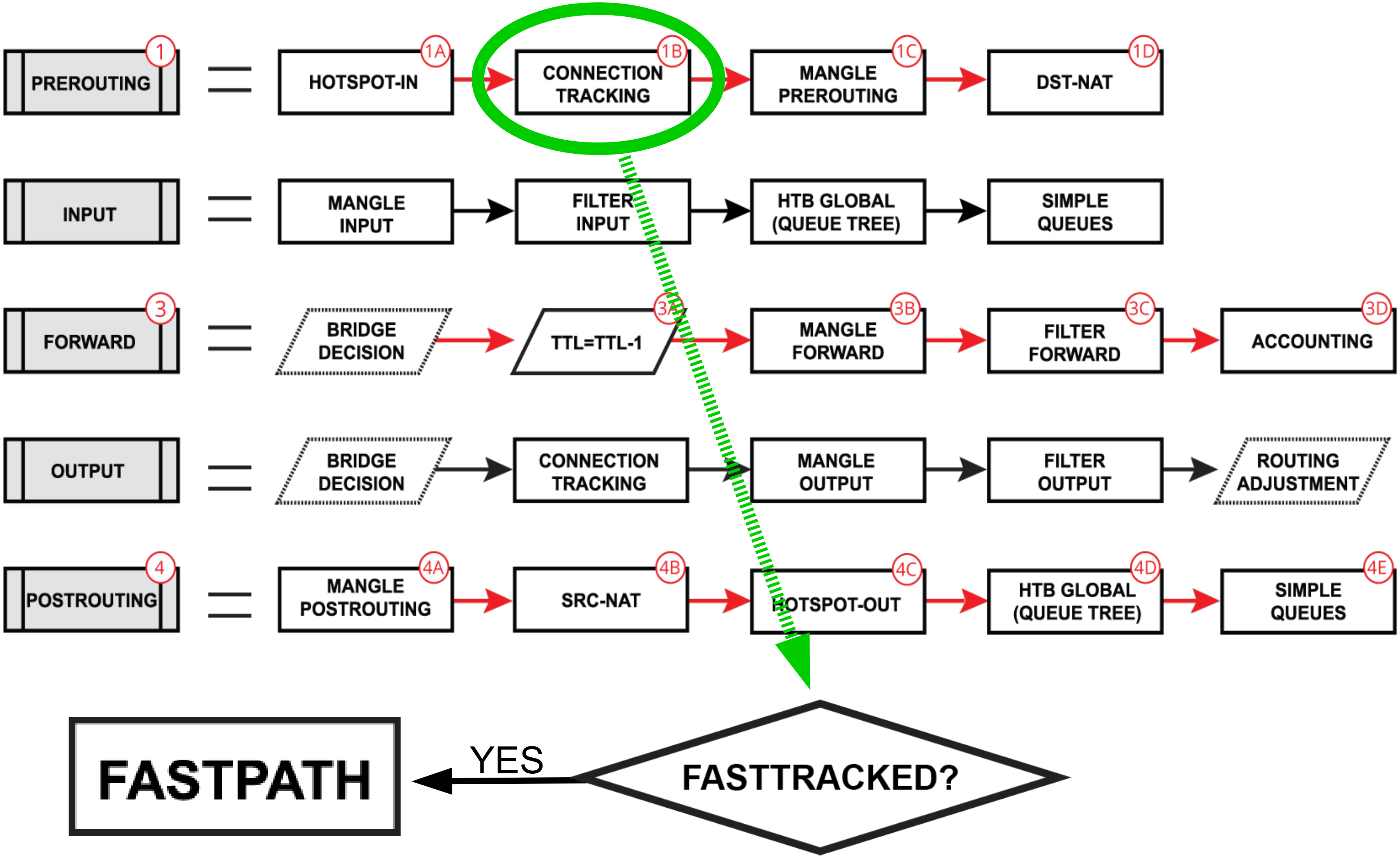
Filter Rules | NAT | Mangle | Service Ports | Connections | Address Lists | Layer7 Protocols

Tracking | Find

	Protocol	Timeout	TCP State	Orig./Repl. Rate	Orig./Repl. Bytes	Orig./Repl. Packets	Orig./Repl. Fasttrack Bytes	Orig./Repl. Fasttrack Packets
SACFs	6 (tcp)	1d 00:04:02	established	54.4 kbps/1546.4 kbps	141.0 MiB/3662.3 MiB	2 737 217/2 717 ...	141.0 MiB/3662.1 MiB	2 737 213/2 716 883
SACFd	17 (udp)	00:05:01		1984 bps/34.6 kbps	3107.7 KiB/6.5 MiB	9 070/10 870	3107.1 KiB/6.5 MiB	9 068/10 869
SACFd	17 (udp)	00:04:33		0 bps/0 bps	2653.7 KiB/3491.0 KiB	6 630/5 828	2653.3 KiB/3490.9 KiB	6 628/5 826
SACFs	17 (udp)	00:04:51		0 bps/0 bps	445.5 KiB/50.6 KiB	4 842/477	445.0 KiB/50.2 KiB	4 836/474
SACFd	17 (udp)	00:04:55		0 bps/0 bps	858.6 KiB/3085.5 KiB	4 711/4 608	858.3 KiB/3085.4 KiB	4 709/4 607
SACFs	17 (udp)	00:05:03		39.7 kbps/3.6 kbps	2856.8 KiB/507.5 KiB	4 566/3 922	2856.3 KiB/507.4 KiB	4 564/3 921
SACFd	17 (udp)	00:01:52		0 bps/0 bps	1997.0 KiB/2866.6 KiB	4 536/4 754	1996.3 KiB/2866.6 KiB	4 534/4 753
SACFs	6 (tcp)	1d 00:03:32	established	0 bps/0 bps	922.7 KiB/367.4 KiB	4 406/4 659	920.3 KiB/366.9 KiB	4 399/4 649
SACFd	17 (udp)	00:01:43		0 bps/0 bps	262.7 KiB/1607.1 KiB	4 260/2 618	262.3 KiB/1607.1 KiB	4 258/2 617
SACFs	17 (udp)	00:05:02		0 bps/0 bps	518.4 KiB/188.6 KiB	4 254/1 632	517.8 KiB/187.8 KiB	4 248/1 622
SACFd	17 (udp)	00:05:03		3.1 kbps/39.5 kbps	1066.7 KiB/3245.1 KiB	3 977/5 265	1066.3 KiB/3245.0 KiB	3 975/5 264
SACFd	6 (tcp)	00:00:00	time wait	0 bps/0 bps	232.7 KiB/2113.2 KiB	3 546/3 540	232.5 KiB/2113.1 KiB	3 541/3 537
SACFd	17 (udp)	00:02:15		0 bps/0 bps	212.9 KiB/1922.1 KiB	3 154/3 048	212.7 KiB/1921.8 KiB	3 152/3 047
SACFd	6 (tcp)	1d 23:59:02	established	6.6 kbps/38.0 kbps	217.6 KiB/1869.3 KiB	3 103/4 144	217.5 KiB/1869.3 KiB	3 101/4 143
SACFs	6 (tcp)	1d 23:59:03	established	37.0 kbps/3.4 kbps	1093.6 KiB/75.3 KiB	2 614/1 111	1093.5 KiB/75.2 KiB	2 611/1 110
SACFd	S - seen reply, A - assured, C - confirmed, F - fasttrack, d - dstnat				155.3 KiB/1588.4 KiB	2 504/1 973	154.9 KiB/1588.4 KiB	2 502/1 972
SACFd	17 (udp)	00:04:48		0 bps/0 bps	162.5 KiB/1670.8 KiB	2 483/2 732	162.0 KiB/1670.7 KiB	2 480/2 730
SACFd	17 (udp)	00:05:00		2.3 kbps/45.6 kbps	153.6 KiB/1617.9 KiB	2 436/2 701	153.3 KiB/1617.8 KiB	2 434/2 700
SACFd	17 (udp)	00:05:02		992 bps/32.9 kbps	222.0 KiB/1548.0 KiB	2 133/2 608	221.7 KiB/1547.9 KiB	2 131/2 607
SACFd	17 (udp)	00:03:13		0 bps/0 bps	136.6 KiB/1350.7 KiB	2 063/2 243	136.3 KiB/1350.7 KiB	2 061/2 242
SACFd	17 (udp)	00:00:31		0 bps/0 bps	134.3 KiB/1451.4 KiB	2 029/2 316	134.0 KiB/1451.3 KiB	2 027/2 315
SACFd	17 (udp)	00:05:01		3.2 kbps/39.5 kbps	121.1 KiB/1547.2 KiB	1 878/2 379	120.6 KiB/1547.2 KiB	1 876/2 378
SACFd	17 (udp)	00:05:01		1984 bps/34.3 kbps	119.3 KiB/1259.9 KiB	1 832/2 100	118.7 KiB/1259.8 KiB	1 829/2 098
SACFs	6 (tcp)	1d 23:59:02	established	34.0 kbps/4.2 kbps	1156.8 KiB/108.4 KiB	1 824/1 777	1156.8 KiB/108.4 KiB	1 822/1 776
SACFd	6 (tcp)	00:00:00	time wait	0 bps/0 bps	113.1 KiB/1859.6 KiB	1 814/2 089	112.9 KiB/1859.5 KiB	1 810/2 086

991 items out of 978 (1 selected) | Max Entries: 218032

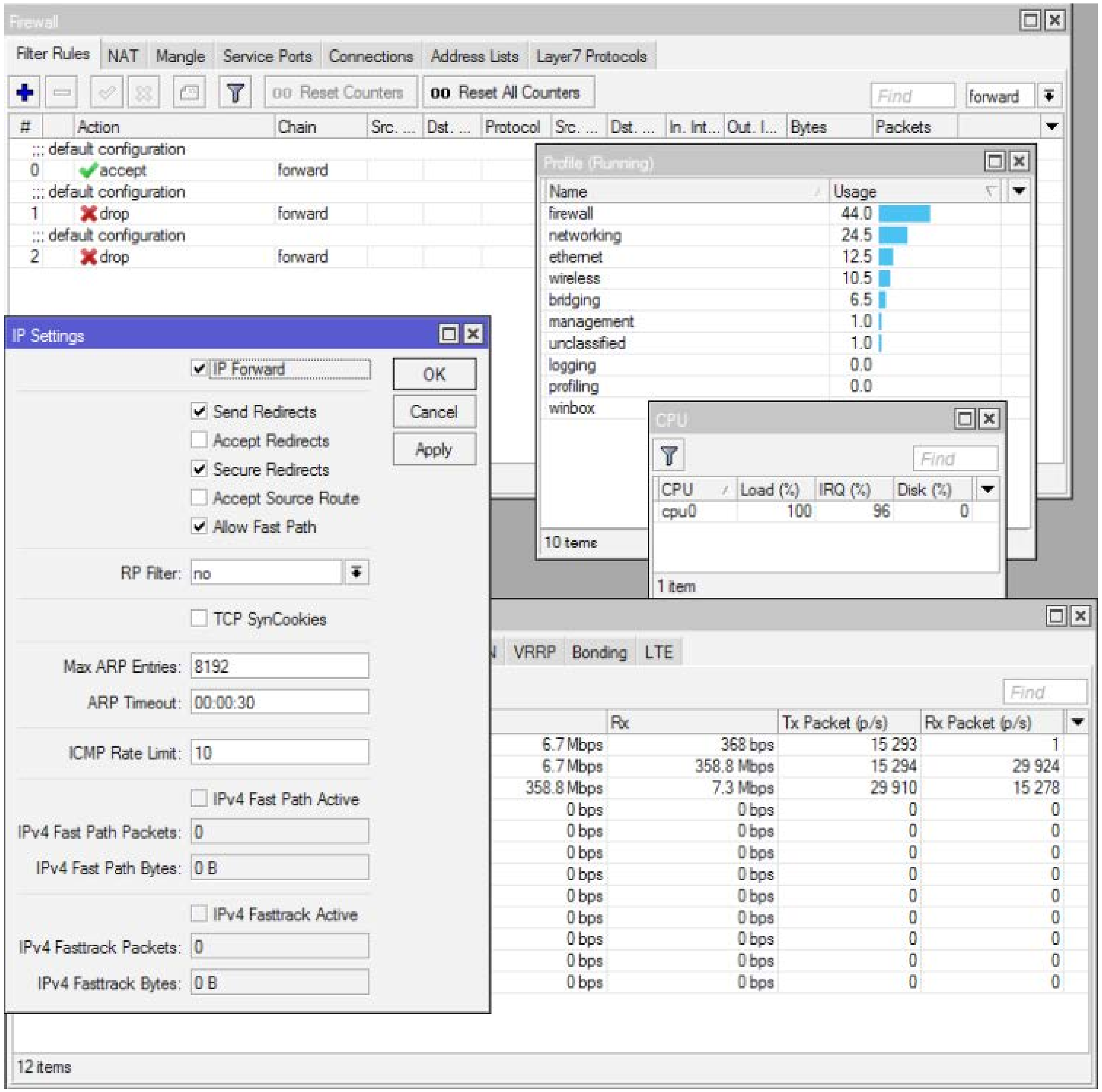
Routing forwarding FastPath



Fasttrack-connection

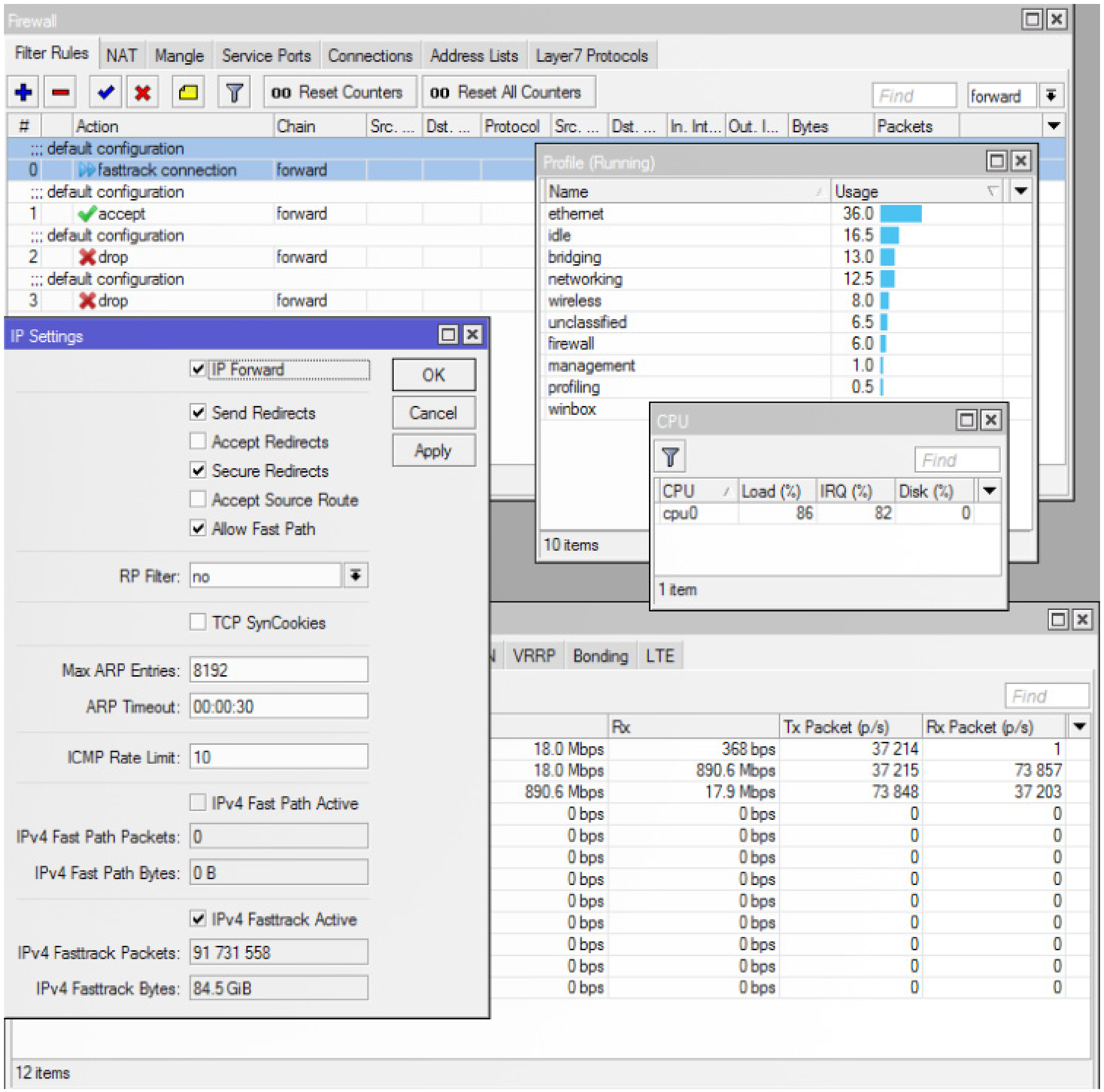
- "fasttrack-connection" action works similar to "mark-connection" action
- "fasttrack-connection" rule is usually followed by identical "accept" rule
- Most common Fasttrack implementations:
  - Fasttrack if connection reach connection-state=established and related
  - Fasttrack to exclude some specific connections from the queues
  - Fasttrack all local connections

Without Fasttrack



- Board: RB2011UiAS-2HnD
- Configuration: default Home AP
- Single TCP connection throughput: 358Mbps
- CPU load: 100%
- Firewall CPU load: 44%

With Fasttrack



- Board: RB2011UiAS-2HnD
- Configuration: default Home AP
- Single TCP connection throughput: 890Mbps
- CPU load: 86%
- Firewall CPU load: 6%

Full queues and multi-core processing

- Packets are spending most part of the processing time waiting in full queues
- In order not to waste CPU core cycles on waiting, current core will just leave the packets in the queue and take already processed packets out of the same queue
- Queued packets can be taken out of the queue randomly by the CPU core, that works on that queue at the time
- In short: full queues can shuffle packet assignments to CPU cores

Empty queues and multi-core processing

- In order not to waste CPU core cycles on waiting, current core will just leave packets in the queue and take already processed packets out of the same queue
- In the case when the queue limit is not reached, same packets will be left in and taken out of the queue by the same CPU core, making this process seamless

Conclusions

- Queues don’t slow down single TCP streams if they are not actually queuing packets (limits are not reached)
- The complexity of the configuration has a minimal impact on a single TCP stream (with the exception of deep packet inspection and queues), if the CPU core/thread doesn’t reach 100%
- Use configuration features like fastpath, fasttrack, ip firewall raw, etc. to reduce overall CPU load

Testing from router to router

- Traffic generation/elimination takes at least the same amount of CPU resources as simple traffic forwarding
- The router needs to do both generate traffic and then forward it to destination
- By default the traffic forwarding process has the highest priority in routers (the reason why ping through the router is better than to the router)

Traffic-generator tool

- Traffic Generator can:
  - Determine transfer rates, packet loss
  - Detect out-of-order packets
  - Collect latency and jitter values
  - Inject and replay *.pcap file
- “Quick” mode
- Full Winbox support (coming soon)
- Doesn’t have TCP protocol support
- Scares people

Bandwidth-test tool

- People love it!!!
- Until v6.44 was single threaded, now both UDP and TCP tests support multi-threading
- In v6.44 we added warning message when CPU load exceeds 90% (CLI only), to inform that CPU is bottlenecking results, not the link
- Can do multistream tests (but how many should you do?)

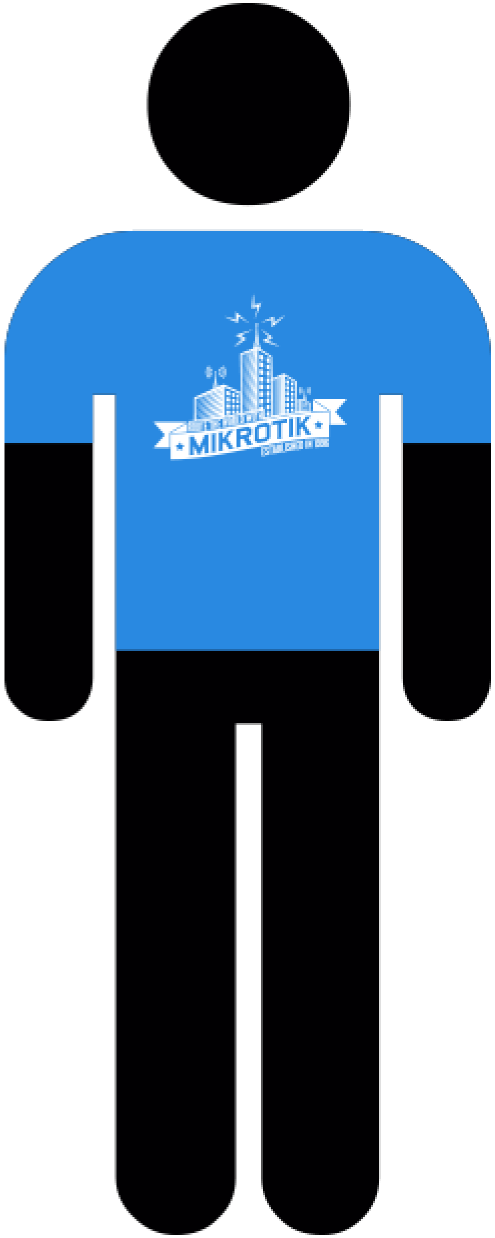
Speed-test tool

- Introduced in RouterOS v6.44 (CLI only)
- It is a simple test tool for measuring ping, jitter, TCP and UDP throughput from one MikroTik device, to another
- It is based on bandwidth-test and ping tool, to use it – the bandwidth-test server needs to be accessible
- It automatically determines optimal number of test streams based on the CPU core/thread count on both devices

Speed-test tool

```
[admin@MikroTik]] > /tool speed-test address=192.168.88.1
              status: done
      time-remaining: 0s
    ping-min-avg-max: 541us / 609us / 3.35ms
  jitter-min-avg-max: 0s / 76us / 2.76ms
                loss: 0% (0/100)
        tcp-download: 921Mbps local-cpu-load:30%
          tcp-upload: 920Mbps local-cpu-load:30% remote-cpu-load:25%
        udp-download: 917Mbps local-cpu-load:6% remote-cpu-load:21%
          udp-upload: 916Mbps local-cpu-load:20% remote-cpu-load:6%

[admin@MikroTik]] > /tool speed-test address=192.168.88.1
                 ;;; results can be limited by cpu, note that traffic generation/termination
                     performance might not be representative of forwarding performance
              status: done
      time-remaining: 0s
    ping-min-avg-max: 541us / 609us / 3.35ms
  jitter-min-avg-max: 0s / 76us / 2.76ms
                loss: 0% (0/100)
        tcp-download: 721Mbps local-cpu-load:78%
          tcp-upload: 820Mbps local-cpu-load:100% remote-cpu-load:84%
        udp-download: 906Mbps local-cpu-load:10% remote-cpu-load:54%
          udp-upload: 895Mbps local-cpu-load:55% remote-cpu-load:12%
```



Questions!!!