About Me

- Jānis Meģis, MikroTik
- Jānis (Tehnical, Trainer, NOT Sales)
  - Support & Training Engineer for almost 7 years
  - Specialization: QoS, PPP, Firewall, Routing
  - Teaching MikroTik RouterOS classes since 2005

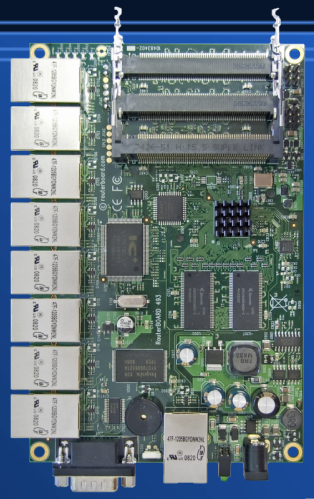
Hardware Performance and RouterOS

Phoenix, AZ

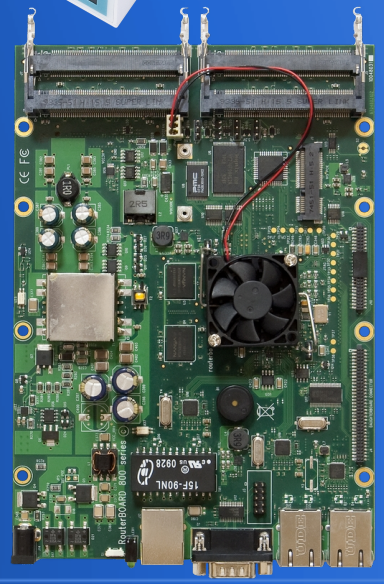
MUM USA 2010

Available Hardware Options

- CPE
  - RB4xx series
  - RB7xx series



- Core Routers
  - X86 multicore solutions
  - RB800/ RB1100



Router Performance

- Usually routers performance is measured in
  - Throughput - amount of traffic it is possible to process
  - Latency – delay that is created when packets travel thought the router
- Both of these characteristics can be increased by:
  - Faster CPU
  - Faster memory
  - Hardware acceleration

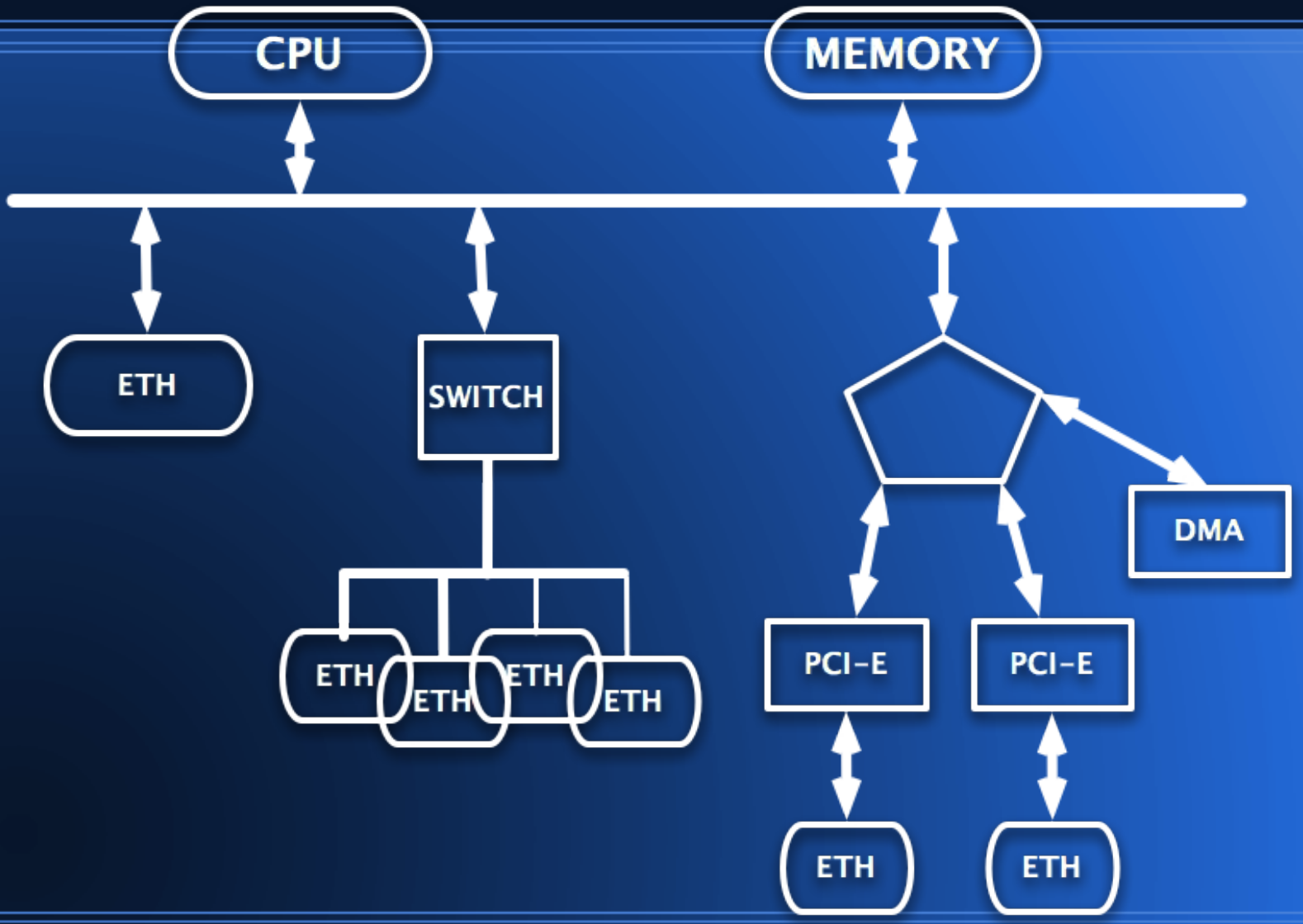
"David and Goliath"

- *“Why can a RouterBOARD with 1GHz CPU handle more traffic than my 3GHz PC???”*
- There can be several reasons:
  - RouterBOARDS have CPUs specifically designed for networking , x86 uses general purpose CPUs
  - hardware acceleration features (like IPSec acceleration or RB1000)
  - Interfaces....

Interfaces

- Directly connected to local bus
  - Have direct access to memory
  - PowerPC architecture allows to write packets directly to CPUs L2 memory
- Connected through the switch-chip
  - Allow wire speed communications between Ethernets in this switch group
- Connected via standard PCI-E
  - (standard way no shortcuts)

Ethernets



RouterBOARD Comparison

- RouterBOARD maximal Ethernet throughput comparison can be found here:
- http://www.routerboard.com/pdf/routerboard_performance_tests.pdf
- All tests was done
  - through the router
  - with minimal configuration
  - with Agilent N2X testing equipment

Guidelines for Comparison

- Results should be used to compare RouterBOARDs between each other not to production systems
  - 64 byte packet throughput reflect CPU performance
  - 1500 byte packet throughput reflects Memory performance
  - 512 byte packet reflects CPU/Memory combined performance

How to compare? (1)

- In simple bridge configuration with 64 byte packets:
  - RB7xx can handle 91000 pps
  - RB1xx can handle 14000 pps
- So RB7xx is 6-7x faster on installations with low configuration and high pps

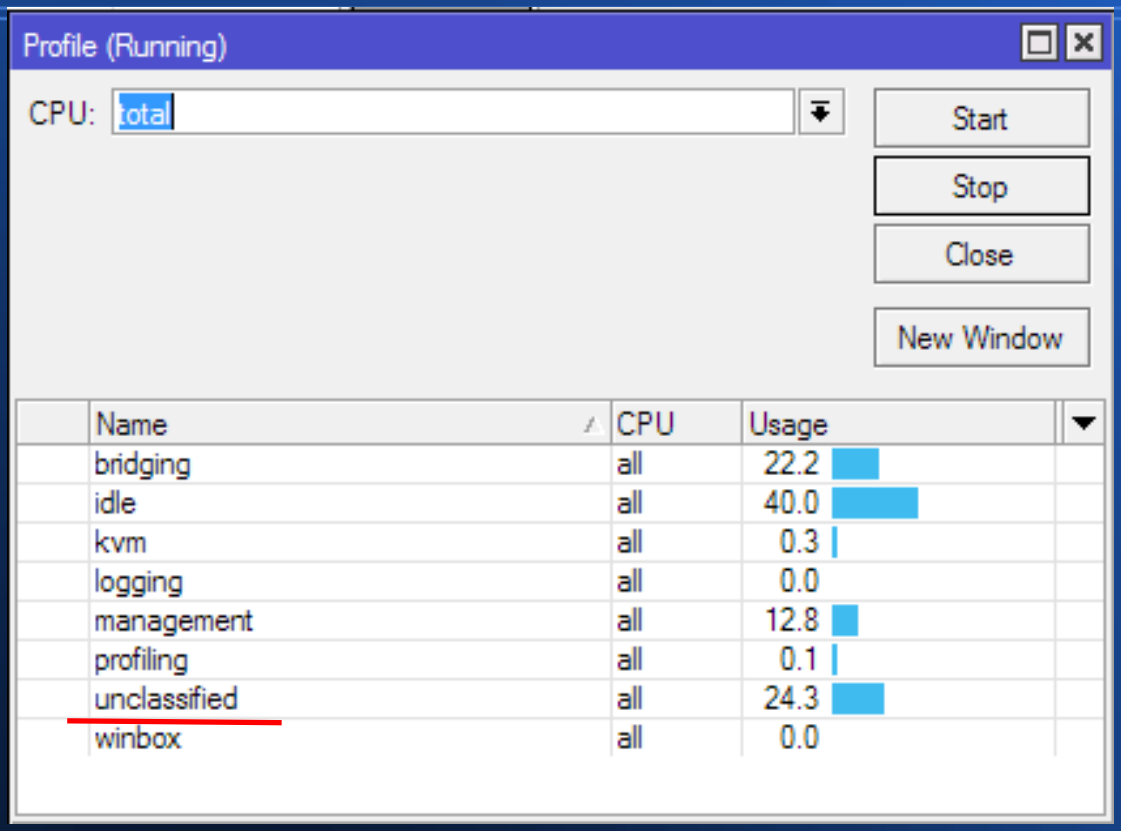
How to compare? (2)

- In bridge configuration with 64 byte packets with firewall and connection tracking enabled
  - RB7xx can handle 41600 pps
  - RB1xx can handle 4930 pps
- So as soon as we enable firewall and connection tracking (2 of most resource expensive options) RB7xx is 8-9x faster

Results of comparison

- From previous slides we can see
  - With small configuration RB7xx is 6+ times faster than RB1xx
  - As soon as configuration gets more complicated gap between board performance gets even larger
  - It is impossible to compare big packets as RB7xx meets with interface limitations (but potential can be seen in RB750G results)

CPU Load: Tools Profile(r)

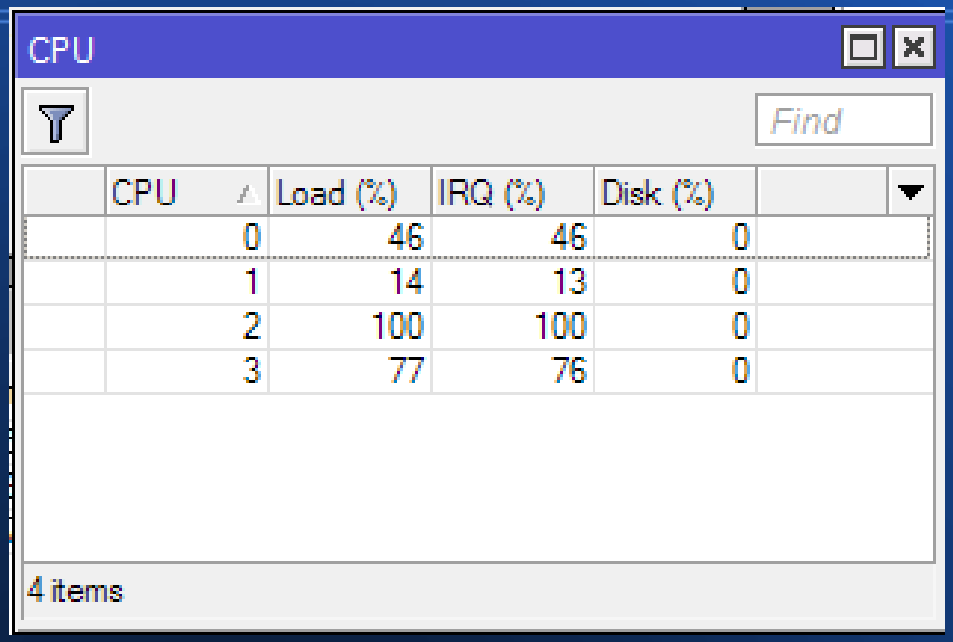


- RouterOS 5.x can report system usage by processes
- Work is still in progress
- Please, report all “unclassified” processes to support@mikrotik.com (with supout.rif)

CPU Usage

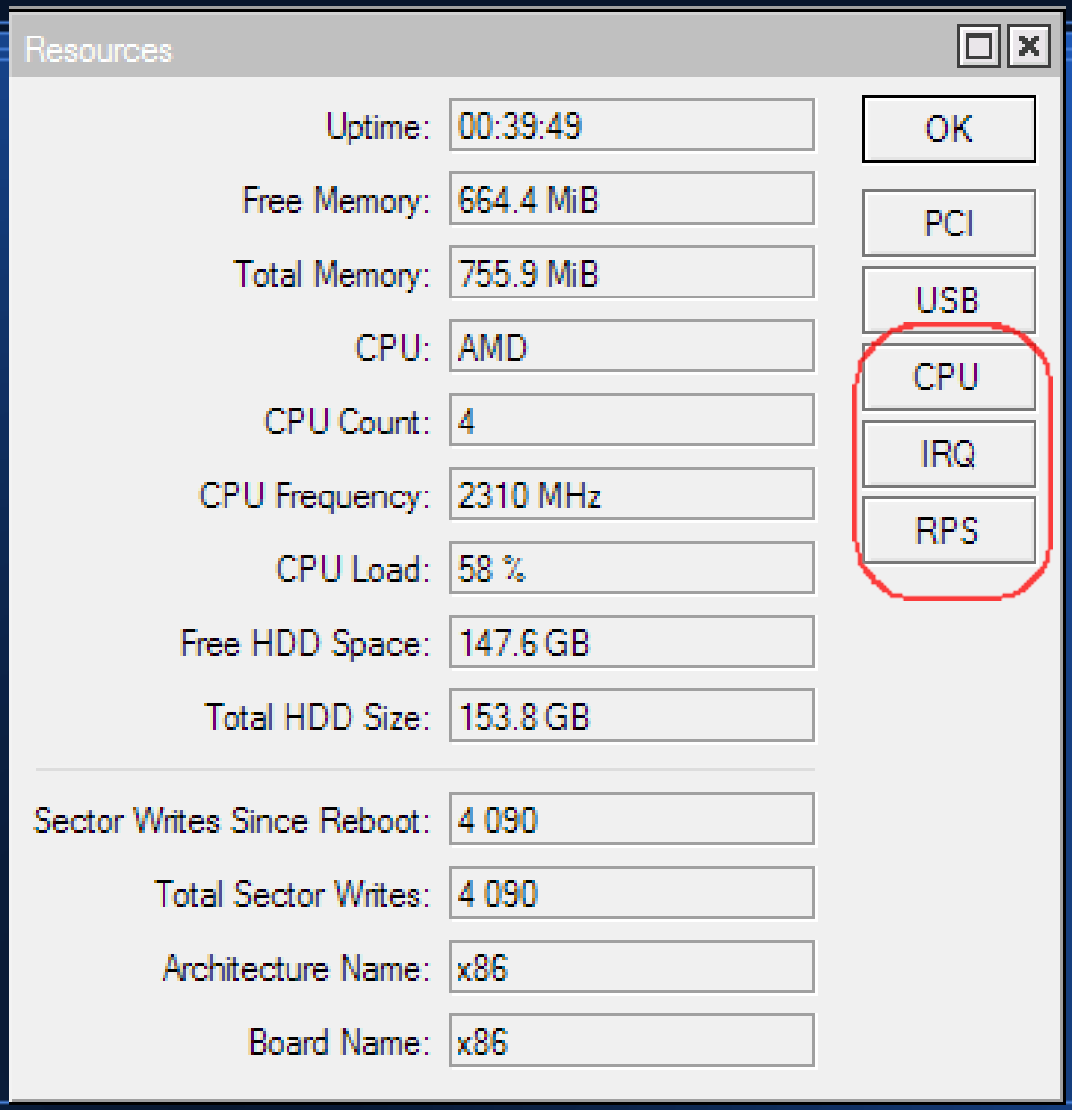
- Common **misconception** is that router works slower (bigger latency, more drops, less throughput) on 70% CPU load,then it does at 20%.
- As long as load is not 100% CPU can handle all that is thrown at it. So 1%, 17%, 50% or 98% - all the calculations are done without delays and there are no slowdowns because of CPU.
- Exceptions are multi-core systems

CPU Load per Core



- In RouterOS version 5.x multi-core routers can report CPU load per core
- At this point is is possible to recognize CPU usage from disk operation and interrupt request handling (most common usage)
- From CLI just use “/tool resources cpu print”

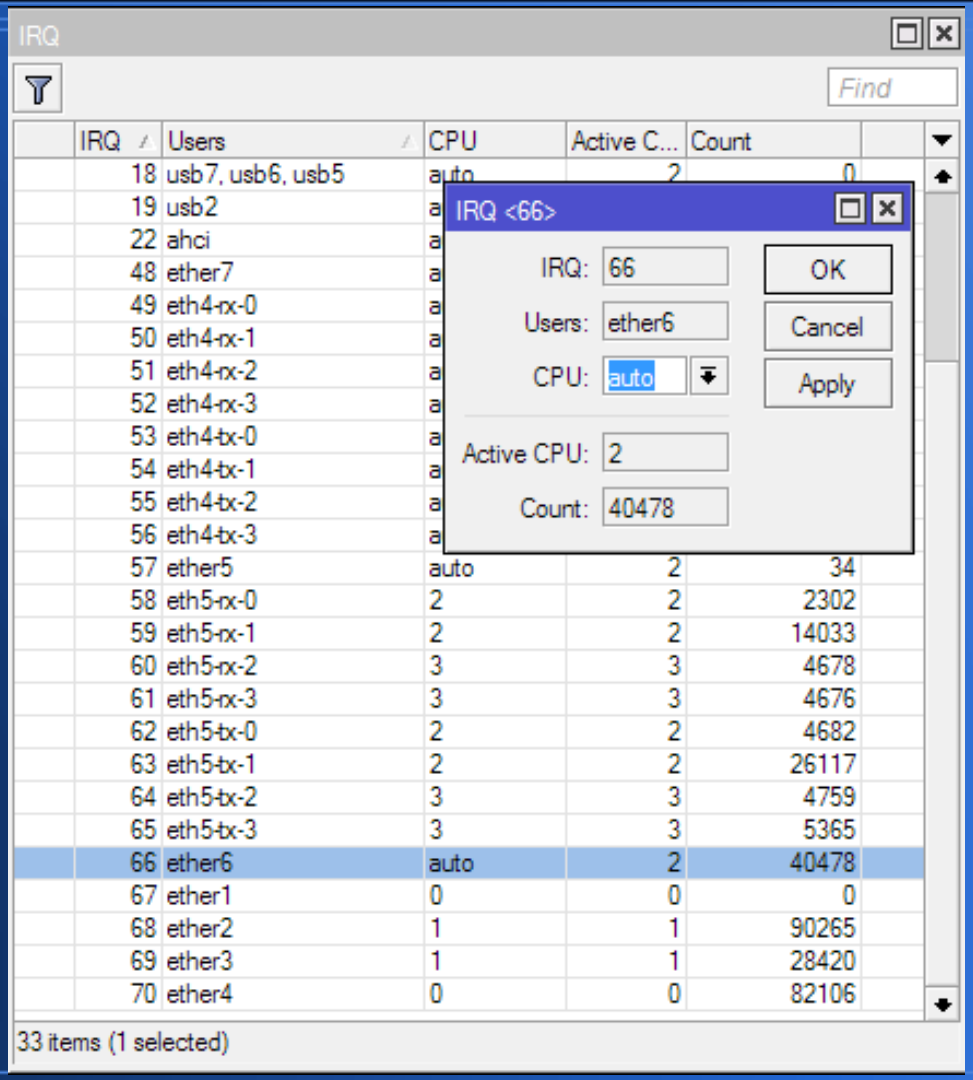
System Resources Menu



- RouterOS version 5.x users will have much more control over their multi-core routers
- Controls can be found in “/system resources” menu.

IRQ Load Balancing

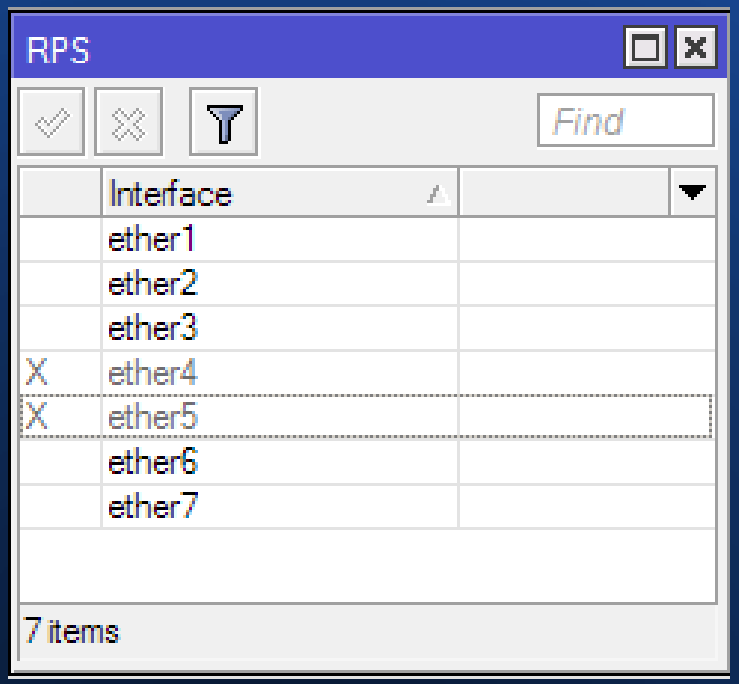
- It is possible to assign CPU cores to specific IRQ's
- Device distribution to IRQ's is done by hardware (not RouterOS)
- Auto mode works based on number of interrupts



NAPI ("New API") feature

- NAPI is an optional modification to the device driver packet processing framework.
- NAPI allows drivers to run with significantly lower number of interrupts during times of high traffic.
- NAPI-compliant drivers can often cause packets to be dropped in the network adapter itself, before the Kernel sees them at all
- NAPI can force “auto” mode to use only one core

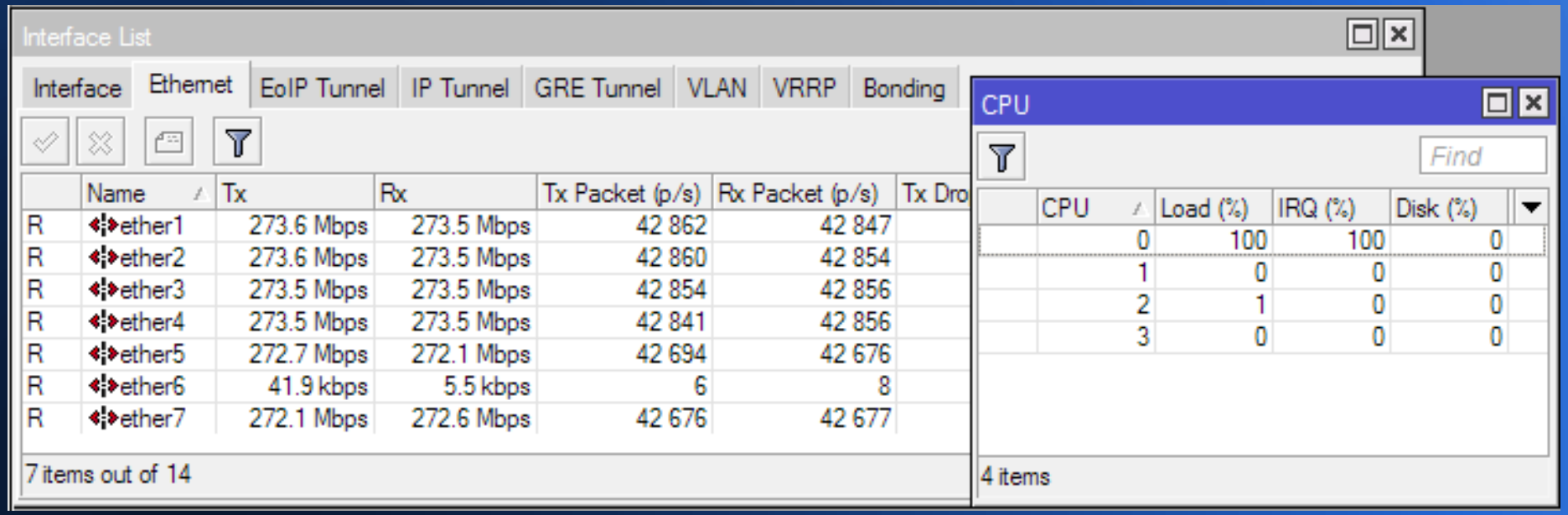
RPS: Receive Packet Steering



- NAPI can become a bottleneck under high packet load because is serialized per device queue
- RPS allows to distribute the load of received packet processing across multiple cores

Performance before optimization

- IRQ balancing = auto
- NAPI working (lots of packets)
- RPS disabled



Performance after optimization

- IRQ balancing – static assignation (mostly)
- NAPI enabled (even more packets)
- RPS enabled only on interfaces with “RX drops”