# The rostune package: Monitoring systems of distributed ROS nodes

**Georgios Stavrinos and Stasinos Konstantopoulos**
National Centre for Scientific Research "Demokritos", Greece
gstavrinos@iit.demokritos.gr, konstant@iit.demokritos.gr

## Problem Statement

Multi-core distributed ROS systems can radically improve their performance by optimizing the placement of the different nodes based on their data requirements, the bandwidth of the required data, and the processing load that a node imposes. This optimization is often straightforward, but interesting cases also exist where the decision to take nodes off-boards needs to be supported by run-time analysis.

- *Vision is both CPU-intensive and a consumer of high-bandwidth data, so it not obvious if it should be performed on-board.*
- *Clustering and pattern recognition on the 3D point cloud is CPU-intensive, but if fusion with vision is needed it is not obvious if it should be off-boarded on the same processing node as vision or on a different one.*

## Characteristic Use Case

- Extensive processing, both on-board and using off-board processing units available at the home.
- An informed decision needs to be made about which nodes should be on-board and which off-board
  - → On-board NUC capabilities and battery autonomy
  - → Wifi bandwidth, latency, and availability limitations

### The RADIO Project

A home assistant/health monitor TurtleBot2, using robot perception and smart home sensor data.

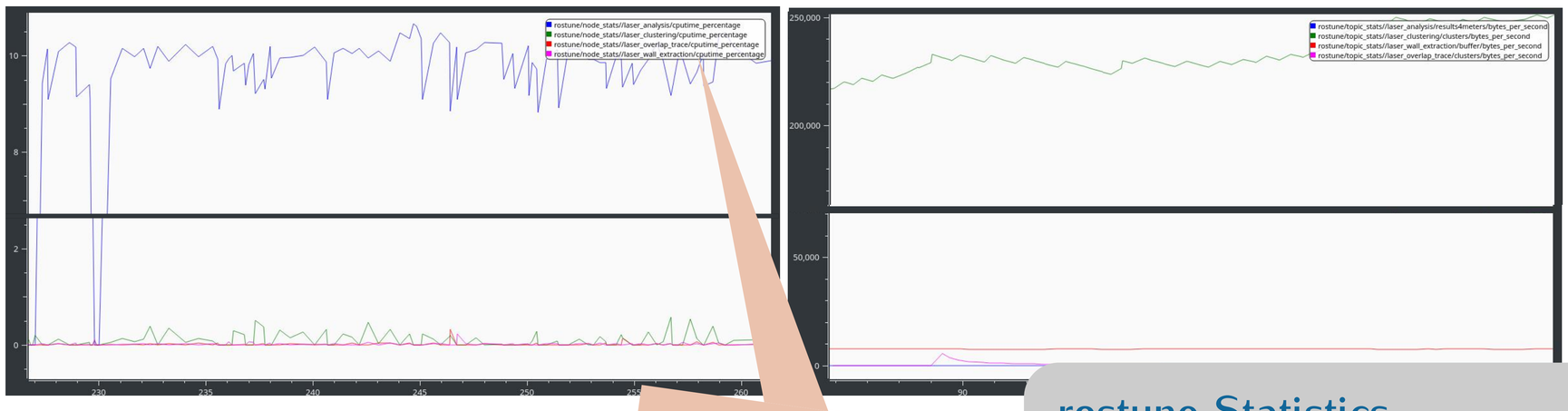
## rostune at a Glance

rostune supports the configuration of distributed ROS systems by providing CPU, RAM and network statistics. The provided data can be visualized to enhance the better understanding of the distributed system's performance strengths and bottlenecks.

- Universal topic listener
  - → No message definitions needed
  - → Collects bandwidth statistics for all topics
- Collects Linux CPU and RAM statistics for all nodes
- Behaves correctly in multi-core multi-server environments
  - → One rostune instance per machine
  - → Each instance reports statistics only for topics that nodes on this machine have subscribed to.
  - → Filters CPU and RAM statistics only for nodes that are running on the same machine as the rostune instance.

## Visualization

The provided rostune statistics can be visualized using ROS data visualization tools such as rqt_plot and PlotJuggler.




Processing pipeline:
1. laser_wall_extraction
2. laser_clustering
3. laser_overlap_trace
4. laser_analysis

## rostune Statistics

In the graph to the left (CPU time), laser_analysis appears to consume a good portion of a CPU core. In the graph on the right (bandwidth), laser_clustering feeds laser_analysis using minimal bandwidth. Based on these observations, we can experiment with sending laser_analysis off-board on a machine with a higher-end CPU.

## Acknowledgements and References

THE FRAMEWORK PROGRAMME FOR RESEARCH AND INNOVATION
HORIZON 2020

http://radio-project.eu
https://github.com/radio-project-eu
https://github.com/roboskel/rostune