Energy Efficiency of ROS Nodes in Different Languages: Publisher-Subscriber Case Studies

Michel Albonico, Paulo Júnior Varela, Adair José Rohling and Andreas Wortmann
Agenda

- Energy-efficient/Green Software
- ROS 2 Programming
- Research Design
- Results
- RQ Answers
- Limitations and Future Directions
Energy-efficient/Green Software
Battery, Budget and Environmental Sustainability

- Software (ROS) is a key-element in robotic systems [1];
- ICT is estimated to be responsible for 20% of global energy usage by 2025 [4];
- Software energy efficiency has been a recurrent research topic [2,3];
- Greener/More energy-efficient software have been researched with different goals:
  - Saving battery (smartphone apps) [5];
  - Affordable budgets/better resource allocation (infrastructure) [6];
  - Reducing the environmental impact [7].
  - Less greenhouse emissions.
- ROS community discussion on energy efficiency is an open file/no ground truth [8].

ROS 2 Programming
Shared Underlying Layers

- ROS Client Support Library

ROS 2 Programming

rcl API

- ROS Client Support Library
### Goal

Analyze ROS programming with C++ and python **with the purpose of** understanding the extent **with respect to** energy efficiency **from the point of view of** robotics researchers and practitioners **in the context of** publisher and subscriber ROS nodes.

<table>
<thead>
<tr>
<th>RQ1: What is the effect of C++ and python on the energy efficiency of a publisher/subscriber ROS node?</th>
<th>RQ2: How does the underlying software stack influence ROS nodes’ energy efficiency?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption (mW)</td>
<td>CPU Usage (%)</td>
</tr>
</tbody>
</table>
## Research Design

### Experiment Planning


<table>
<thead>
<tr>
<th>Node</th>
<th>Description</th>
<th>Dependencies</th>
<th>LLOC Python</th>
<th>LLOC C++</th>
<th>MCC Python</th>
<th>MCC C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.Simple Publisher (Node1)</strong></td>
<td>ROS node (talker) that continuously sends messages to a topic for a constrained period of time.</td>
<td>rclpy/rclcpp, std_msgs</td>
<td>34</td>
<td>39</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>2.Simple Subscriber (Node2)</strong></td>
<td>ROS node (listener) that subscribes to a topic and reads messages published by (Node1) until no more messages are received.</td>
<td>rclpy/rclcpp, std_msgs</td>
<td>34</td>
<td>42</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>3.Teleoperation Publisher (Node3)</strong></td>
<td>ROS node that sends coordinates via teleoperation messages to supposedly drive a simulated robot forward and backward.</td>
<td>rclpy/rclcpp, geometry_msgs</td>
<td>35</td>
<td>38</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>4.Teleoperation Subscriber (Node4)</strong></td>
<td>ROS node that reads coordinates published by Node 3.</td>
<td>rclpy/rclcpp, geometry_msgs</td>
<td>30</td>
<td>36</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Research Design
Experiment Execution

- Instrumentation:
  - Linux Ubuntu 22.04 operating system, kernel version 6.2.0-33-generic, with 8GB of RAM, and an Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz.
  - ROS humble distribution.
  - Algorithms are executed via ROS run command.
  - Power saving mode: performance.
  - Process priority level: -20.
  - 1 minute cool-down period between executions.
**Research Design**

**Experiment Execution**

- Resource profiling instrumentation:
    - pyRAPL wrapping up the ROS run command.
    - Validated against partial measurements with existing RAPL-based tool by Pereira et al. *
  - Power Consumption: \( P = \frac{E}{t} \)
  - CPU Usage: `ps` Linux command.
  - Memory Usage: `pmap` Linux command.

---

Results

Simple Publisher and Subscribers

Figure 2: Power consumption of Nodes 1 (n1) and 2 (n2).

(a) Distribution of power consumption (mW) with 1 subscriber.

(b) Distribution of power consumption (mW) of n1 with 10 subscribers (n2 instances).

Figure 3: Power consumption scalability for Node 1.

Figure 4: CPU and memory scalability of Node 1.
Results
Teleoperation Publisher and Subscribers

Figure 5: Power consumption (mW) of Nodes 3 (n3) and 4 (n4) with one instance each.

Figure 6: Power consumption scalability of Node 3.

Figure 7: CPU and memory scalability of Node 3.
RQ Answers

- **RQ1:** What is the effect of C++ and python on the energy efficiency of a publisher/subscriber ROS node?
  - The choice of programming language, has an impact, with C++ being more energy-efficient.

- **RQ2:** How does the underlying software stack influence ROS nodes’ energy efficiency?
  - Memory tends to be the more determinant factor, despite not being exactly proportional.
  - There may be other architectural aspects of rcl client library implementations may influence the energy efficiency.
    - e.g., multithreading, memory allocation, garbage collection…
Limitations and Future Directions

- The study reveals that carefully deciding about the programming language can help in having a greener/energy-efficient ROS code;
- The experiments focus on very specific nodes, which may not be completely representative:
  - Different communication layers;
  - Different message type formats;
  - Composed nodes.
- Part of the second experiment is inconclusive (overload), which requires further investigation.
- It is important to have a more detailed understanding of the implementation of each of the rcl client libraries.
Energy Efficiency of ROS Nodes in Different Languages: Publisher-Subscriber Case Studies

Michel Albonico, Paulo Júnior Varela, Adair José Rohling and Andreas Wortmann

2024 ACM/IEEE 6th International Workshop on Robotics Software Engineering (RoSE '24), April 15, 2024, Lisbon, Portugal