ROS 2 for Consumer Robotics
The iRobot use-case

ROSCon
November 1st, 2019

Soragna, Alberto
Oxoby, Juan
Dhiraj, Goel
ROS 2 for Consumer Robotics

Agenda

01  iRobot’s use-case
02  ROS 2 Performance Framework
03  Performance Evaluation
04  New Intra-Process Manager
05  Scalability
Our Portfolio

- **m6**
  - Braava jet®

- **s9+**
  - Roomba®

- **Terra™ t7**
  - Robot Mower
ROS 2 to rule them all?

Is ROS 2 suitable for Linux-based low-cost embedded platforms?
iRobot’s use-case
ROS 2 for Consumer Robotics

- Low-cost Linux-based embedded hardware
- Single process application
- ~1000 robots in the same network (dev stage)
Defining a target platform

The goal is to run our entire navigation system on a RPi2

Raspberry Pi 2 Model B specs:
- 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
Defining our performance requirements

Adding a ROS 2 layer to our existing system should add a small overhead:

- less than 20% CPU
- less than 20MB RAM
- have acceptable latency
- zero lost messages
ROS 2 Performance Framework
The iRobot ROS 2 Performance Framework

- Arbitrary user-defined ROS 2 topologies
- Simple to use but highly configurable
- Human-readable results
- Open-source

https://github.com/irobot-ros/ros2-performance
The iRobot ROS 2 Performance Framework

**INPUT:**
- topology file(s)

**OUTPUT:**
- RAM
- CPU%
- latency
- events

(run ...
INPUT - Topology .json file

```json
{
    "nodes": [
        {
            "node_name": "node_A",
            "publishers": [
                {
                    "topic_name": "topic_A",
                    "msg_type": "1Kb",
                    "period_ms": 10,
                    "qos": "reliable"
                }
            ]
        },
        {
            "node_name": "node_B",
            "subscribers": [
                {
                    "topic_name": "topic_A",
                    "msg_type": "1Kb",
                    "qos": "best-effort"
                }
            ]
        }
    ]
}
```
### OUTPUT – Resources

<table>
<thead>
<tr>
<th>time[ms]</th>
<th>cpu[%]</th>
<th>arena[KB]</th>
<th>in_use[KB]</th>
<th>mmap[KB]</th>
<th>rss[KB]</th>
<th>vsz[KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>27</td>
<td>61080</td>
<td>60712</td>
<td>0</td>
<td>24948</td>
<td>305212</td>
</tr>
<tr>
<td>1000</td>
<td>51</td>
<td>167860</td>
<td>167539</td>
<td>0</td>
<td>47060</td>
<td>544012</td>
</tr>
<tr>
<td>1500</td>
<td>50</td>
<td>202844</td>
<td>202753</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>2000</td>
<td>43</td>
<td>202860</td>
<td>202769</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>2500</td>
<td>39</td>
<td>202876</td>
<td>202786</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>3000</td>
<td>36</td>
<td>202900</td>
<td>202801</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>3500</td>
<td>34</td>
<td>202928</td>
<td>202822</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>4000</td>
<td>33</td>
<td>202936</td>
<td>202837</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>4500</td>
<td>31</td>
<td>202948</td>
<td>202853</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>5000</td>
<td>31</td>
<td>202968</td>
<td>202868</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>5500</td>
<td>30</td>
<td>202992</td>
<td>202893</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
<tr>
<td>6000</td>
<td>29</td>
<td>203012</td>
<td>202909</td>
<td>0</td>
<td>55496</td>
<td>660724</td>
</tr>
</tbody>
</table>
## OUTPUT – Events

<table>
<thead>
<tr>
<th>Time[ms]</th>
<th>Caller</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>SYSTEM</td>
<td>0</td>
<td>[discovery] PDP completed</td>
</tr>
<tr>
<td>151</td>
<td>SYSTEM</td>
<td>0</td>
<td>[discovery] EDP completed</td>
</tr>
<tr>
<td>156</td>
<td>amazon-&gt;lyon</td>
<td>1</td>
<td>msg 0 late. 4300us &gt; 2000us</td>
</tr>
<tr>
<td>156</td>
<td>danube-&gt;mandalay</td>
<td>1</td>
<td>msg 0 late. 4082us &gt; 2000us</td>
</tr>
<tr>
<td>250</td>
<td>danube-&gt;ponce</td>
<td>1</td>
<td>msg 10 late. 2303us &gt; 2000us</td>
</tr>
<tr>
<td>250</td>
<td>danube-&gt;geneva</td>
<td>1</td>
<td>msg 10 late. 2617us &gt; 2000us</td>
</tr>
<tr>
<td>338</td>
<td>arkansas-&gt;arequipa</td>
<td>2</td>
<td>msg 0 too late. 184382us &gt; 50000us</td>
</tr>
<tr>
<td>338</td>
<td>arkansas-&gt;arequipa</td>
<td>2</td>
<td>msg 1 too late. 136936us &gt; 50000us</td>
</tr>
<tr>
<td>339</td>
<td>arkansas-&gt;arequipa</td>
<td>1</td>
<td>msg 2 late. 37064us &gt; 50000us</td>
</tr>
<tr>
<td>2314</td>
<td>parana-&gt;geneva</td>
<td>1</td>
<td>msg 216 late. 2097us &gt; 2000us</td>
</tr>
<tr>
<td>2314</td>
<td>parana-&gt;osaka</td>
<td>1</td>
<td>msg 216 late. 2131us &gt; 2000us</td>
</tr>
<tr>
<td>3614</td>
<td>parana-&gt;osaka</td>
<td>1</td>
<td>msg 346 late. 2044us &gt; 2000us</td>
</tr>
<tr>
<td>3615</td>
<td>parana-&gt;geneva</td>
<td>1</td>
<td>msg 346 late. 2682us &gt; 2000us</td>
</tr>
<tr>
<td>3644</td>
<td>nile-&gt;hamburg</td>
<td>1</td>
<td>msg 349 late. 2234us &gt; 2000us</td>
</tr>
<tr>
<td>3645</td>
<td>tigris-&gt;hamburg</td>
<td>1</td>
<td>msg 349 late. 2081us &gt; 2000us</td>
</tr>
</tbody>
</table>
## OUTPUT – Latency

<table>
<thead>
<tr>
<th>node</th>
<th>topic</th>
<th>size[b]</th>
<th>received[#]</th>
<th>late[#]</th>
<th>lost[#]</th>
<th>mean[us]</th>
<th>sd[us]</th>
<th>min[us]</th>
<th>max[us]</th>
</tr>
</thead>
<tbody>
<tr>
<td>lyon</td>
<td>amazon</td>
<td>36</td>
<td>12001</td>
<td>11</td>
<td>0</td>
<td>602</td>
<td>145</td>
<td>345</td>
<td>4300</td>
</tr>
<tr>
<td>hamburg</td>
<td>danube</td>
<td>8</td>
<td>12001</td>
<td>15</td>
<td>0</td>
<td>796</td>
<td>233</td>
<td>362</td>
<td>5722</td>
</tr>
<tr>
<td>hamburg</td>
<td>ganges</td>
<td>16</td>
<td>12001</td>
<td>10</td>
<td>0</td>
<td>557</td>
<td>119</td>
<td>302</td>
<td>4729</td>
</tr>
<tr>
<td>hamburg</td>
<td>nile</td>
<td>16</td>
<td>12001</td>
<td>18</td>
<td>0</td>
<td>658</td>
<td>206</td>
<td>300</td>
<td>5258</td>
</tr>
<tr>
<td>hamburg</td>
<td>tigris</td>
<td>16</td>
<td>12000</td>
<td>17</td>
<td>0</td>
<td>736</td>
<td>225</td>
<td>310</td>
<td>5994</td>
</tr>
<tr>
<td>osaka</td>
<td>parana</td>
<td>12</td>
<td>12001</td>
<td>32</td>
<td>0</td>
<td>636</td>
<td>236</td>
<td>346</td>
<td>4343</td>
</tr>
<tr>
<td>mandalay</td>
<td>danube</td>
<td>8</td>
<td>12001</td>
<td>16</td>
<td>0</td>
<td>791</td>
<td>189</td>
<td>418</td>
<td>6991</td>
</tr>
<tr>
<td>mandalay</td>
<td>salween</td>
<td>48</td>
<td>12001</td>
<td>1</td>
<td>0</td>
<td>663</td>
<td>297</td>
<td>391</td>
<td>6911</td>
</tr>
<tr>
<td>ponce</td>
<td>danube</td>
<td>8</td>
<td>12001</td>
<td>15</td>
<td>0</td>
<td>882</td>
<td>203</td>
<td>437</td>
<td>7270</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>received[#]</th>
<th>mean[us]</th>
<th>late[#]</th>
<th>late[%]</th>
<th>too_late[#]</th>
<th>too_late[%]</th>
<th>lost[#]</th>
<th>lost[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>126496</td>
<td>744</td>
<td>204</td>
<td>0.161</td>
<td>2</td>
<td>0.0016</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
OUTPUT – Latency

Message classification by their latency

+----------------------------------------+----------------------------------------+----------------------------------------+
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
|                                        |                                        |                                        |
+----------------------------------------+----------------------------------------+----------------------------------------+

<---------------------      <---------------------      <--------------------->
| on_time                | late                                 | too_late                              |

<---------------------
| period               |

<---------------------
ROS 2 Performance Evaluation
ROS 2 Performance – Benchmark Topology
ROS 2 Performance – Fast-RTPS

Remember that this system is just doing message passing (no extra computation)
ROS 2 Performance – CycloneDDS

- CPU usage [%]:
  - Current: 13.12%
  - Target: 20%

- Physical RAM (RSS) [Mb]:
  - Current: 15.82 Mb
  - Target: 20 Mb

- Late msgs [%]:
  - Current: 1.9%
  - Target: 0.02%

- Lost msgs [%]:
  - Current: 0%
  - Target: 0.0%

- Too late msgs [%]:
  - Current: 0.1%
  - Target: 0%
Reducing the ROS 2 RAM usage

The DDS specification requires that nodes create proxy objects for EACH entity they discover.

What if we know in advance that some entities don’t need to know about some others?

BLACK-LISTING??

Considering our system, the RAM usage decreased to 45% of the original!
New Intra-Process Manager
ROS 2 Intra-Process Communication

Dashing Intra-Process Manager
• Sends meta-messages through the RMW layer!
• Latency and CPU improvement only noticeable for big messages

New Intra-Process Manager (now on master!)
• The communication is performed exclusively inside the RCLCPP layer
• Reduces CPU usage and latency for any message size
New Intra-Process Manager
ROS 2 Performance – Fast-RTPS + new IPC

45% CPU% decrease!
Scaling ROS 2
Problem Statement

~1000 robots in the same network
~35 topics from each robot
~5 different subnetworks across the globe

We want to be able to individually access any robot's messages from a remote debugging machine
Addressing Scalability

Proposed solution 1:
Use default DDS configuration and a different namespace for each robot

Problems:
ALL robots discover each other!
Slow/unreliable discovery, network saturation, multicast requires forwarding between subnets
Addressing Scalability

Proposed solution 2:
Solution 1 + ROS_DOMAIN_ID

Problems:
Limited numbers of domain IDs. What about assignment? Still considerable network usage and multicast limitations
Addressing Scalability

Proposed solution 3:
Disable multicast discovery.
When a remote machine wants to connect to a robot, add the robot’s IP to the initial peers list (unicast)

Problems:
The user needs direct access to the DDS configuration. This can be solved by new APIs.
Solution 3 with CycloneDDS

```xml
<CycloneDDS>
  <Domain>
    <Id>any</Id>
  </Domain>
  <DDSI2E>
    <General>
      <NetworkInterfaceAddress>auto</NetworkInterfaceAddress>
      <AllowMulticast>false</AllowMulticast>
      <EnableMulticastLoopback>true</EnableMulticastLoopback>
    </General>
    <Discovery>
      <ParticipantIndex>0</ParticipantIndex>
      <Peers>
        <Peer Address="${ROBOT_IP}:7410"/>
      </Peers>
    </Discovery>
  </DDSI2E>
</CycloneDDS>
```

ROBOT_IP=10.22.22.90 ros2 topic list
ros_to_dds?

User Application

ros_to_dds

rclcpp (C++ API)
+ Exec. with std::thread
+ Intra-Process Comms
+ Type Masquerading

rclpy (Python API)
+ Exec. with Thread
+ Intra-Process Comms
+ Type Masquerading

rcjava (Java API)
+ java.lang.Thread
+ Intra-Process Comms
+ Type Masquerading

rcl (C API / optional C++ Implementation)
+ Services
+ Parameters
+ Names
+ Time
+ Console Logging

rmw (C API)
+ Pub/Sub with QoS
+ Services with QoS
+ Discovery
+ Graph Events

FastRTPS or
RTI Connext or
RTI Connext Dynamic or

* Intra-Process Comms and Type Masquerading could be implemented in the client library, but may not currently exist.