μROSnode – running ROS on microcontrollers

Martino Migliavacca, Andrea Zoppi, Matteo Matteucci, Andrea Bonarini
migliavacca@elet.polimi.it, andrea.zoppi@mail.polimi.it
AIRLab - Artificial Intelligence and Robotics Laboratory
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
Background
The AIRLab

- Artificial Intelligence and Robotics laboratory active since 1973
- 11 senior researchers, 10 PhD Students, more than 60 Master theses/year
- Industrial, National and EU Projects
- Development of autonomous robots and unmanned vehicles
- Robot perception and multisensor fusion (laser, vision, inertial, GPS, etc.)
Background

R2P: Rapid Robot Prototyping

- Modular hardware/software framework
  - Distributed hardware modules
  - Real-time CAN bus protocol
  - Lightweight publish/subscribe middleware

- Open source development
  - ROS idea at hardware level
Interfacing ROS with hardware devices

Dedicated nodes
Interfacing ROS with hardware devices

Native ROS (TCPROS) devices
μROSnode
In a nutshell

• Lightweight ROS client (not a cross-compiled ROS system!)
• Runs on modern 32-bit microcontrollers (ARM Cortex-M)
• ANSI C implementation
• Supported ROS features
  ▪ XML-RPC graph management
  ▪ TCPROS data protocol
  ▪ Topics
  ▪ Services
  ▪ Global parameters
μROSnode
Framework architecture

- Modular, object-oriented style on top of ANSI C
- Multithreaded implementation
- ChibiOS/RT and Posix ports
- Abstraction layer to low-level libraries
- Simple integration with user firmware
  - Code generator tool
  - Topic/service handler functions
**Implementation details**

**Master node communication - XML-RPC**

**Features**

- Custom implementation
- No external dependencies
- Minimal ROS-oriented syntax
- Single-pass parsing
- On-the-fly processing

**Limits**

- No gzip support
- Size-limited unbuffered messages and strings

```xml
POST /RPC2 HTTP/1.1
Content-Type: text/xml
Content-Length: 178

<?xml version="1.0"?>
<methodCall>
    <methodName>getPid</methodName>
    <params>
        <param>
            <value><string>/rosnode</string></value>
        </param>
    </params>
</methodCall>

HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 309

<?xml version="1.0"?>
<methodResponse>
    <params>
        <param>
            <value>
                <array>
                    <data>
                        <value>i4</value>
                        <value>i4</value>
                        <value>i4</value>
                        <value>i4</value>
                    </data>
                </array>
            </value>
        </param>
    </params>
</methodResponse>
```
Code generator
Generation flow

Message types *.msg

Service types *.srv

rosmsg show

rossrv show

Configuration file

Code generator urosgen.py

Types header urosMsgTypes.h
Types source urosMsgTypes.c
Handlers header urosHandlers.h
Handlers source urosHandlers.c
[Options]
nodeName = turtlesim

[PubTopics]
rosout = rosgraph_msgs/Log
turtleX/pose = turtlesim/Pose
turtleX/color_sensor = turtlesim/Color

[SubTopics]
turtleX/command_velocity = turtlesim/Velocity

[PubServices]
clear = std_srvs/Empty
kill = turtlesim/Kill
spawn = turtlesim/Spawn
turtleX/set_pen = turtlesim/SetPen
turtleX/teleport_absolute = turtlesim/TeleportAbsolute
turtleX/teleport_relative = turtlesim/TeleportRelative

[CallServices]
# none
uros_err_t pub_tpc__turtleX__pose(UrosTcpRosStatus *tcpstp) {
    /* Message allocation and initialization.*/
    UROS_TPC_INIT_H(msg__turtlesim__Pose);

    /* Published messages loop.*/
    while (!urosTcpRosStatusCheckExit(tcpstp)) {
        /* Fill in the contents of the message.*/
        ...
        ... user code ...

        /* Send the message.*/
        UROS_MSG_SEND_LENGTH(msgp, msg__turtlesim__Pose);
        UROS_MSG_SEND_BODY(msgp, msg__turtlesim__Pose);
    }
    tcpstp->err = UROS_OK;

    _finally: /* Message deinitialization and deallocation.*/
    UROS_TPC_UNINIT_H(msg__turtlesim__Pose);
    return tcpstp->err;
}
Benchmarks

Test Platform: R2P Gateway module

- R2P—ROS Gateway module
- STM32F407 microcontroller
  - 32-bit ARM Cortex-M4 core @ 168 MHz
  - 1 MiB flash program memory
  - 192 KB RAM (112 KB main shared block)
- 100BASE-TX Ethernet port
- ChibiOS/RT 2.5.2 real-time OS
- LwIP 1.4.1 network stack
- Turtlesim clone as benchmark firmware
Benchmarks

Code and memory footprint

Code footprint
• < 40 KB for µROSnode

<table>
<thead>
<tr>
<th>Component</th>
<th>With checks [B]</th>
<th>Without checks [B]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O0</td>
<td>O2</td>
</tr>
<tr>
<td>turtlesim</td>
<td>52332</td>
<td>42020</td>
</tr>
<tr>
<td>µROSnode</td>
<td>123270</td>
<td>97691</td>
</tr>
<tr>
<td>LWIP</td>
<td>49898</td>
<td>33133</td>
</tr>
<tr>
<td>ChibiOS/RT</td>
<td>67219</td>
<td>45551</td>
</tr>
<tr>
<td>unused</td>
<td>731281</td>
<td>805605</td>
</tr>
</tbody>
</table>

Memory footprint
• < 1 KB for each topic instance
Benchmarks

Communication Benchmark

Transmission performance
- Up to 20,000 msg/s
- < 100 B limited by rostopic hz
- > 100 B limited by LwIP

Reception performance
- Up to 10,000 msg/s
- < 1 KB limited by rostopic echo
- > 2 KB Ethernet saturation
Use Case
The Triskar2 Omnidirectional Robot

![Diagram of the Triskar2 Omnidirectional Robot]

- **Power**: Battery monitor
- **Proximity**: IR sensors reader
- **DC Motor**: Motor controller
- **Gateway**: Message translation, Kinematics

- **AUX power**
- **R2P modular boards**
- **Wi-Fi**
- **CAN Bus + Power**
- **Ethernet 100 Mb/s**

**Remote Side**
- **Robot Side**

- **RJ45 Connector**
Use Case
Demo Time!
Conclusions

• μROSnode working implementation available
  ▪ https://github.com/openrobots-dev/μROSnode
• Main ROS features supported
• Turtlesim clone demo included
• Doxygen documentation

Future work
• UDPROS support
• C++ wrappers
• Open to suggestions
μROSnode – running ROS on microcontrollers

https://github.com/openrobots-dev/uROSnode

Martino Migliavacca, Andrea Zoppi, Matteo Matteucci, Andrea Bonarini
migliavacca@elet.polimi.it, andrea.zoppi@mail.polimi.it
AIRLab - Artificial Intelligence and Robotics Laboratory
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy