How to Integrate ROS Components into the Embedded Devices

Hideki Takase

(Kyoto University / JST PREST)

Contents

- **ROS: How to accelerate the development**
  - **Background & Motivation**
  - **History of robot software framework**
  - **Features of ROS**
  - **ROS 2: next-generation development platform**
- mROS: How to integrate embedded devices
  - Motivation & Our Strategy
  - Overview & Structure
  - Current Status & Case Study
- Conclusion & Announcement

Background & Motivation

- Characteristics of robot systems
  - Combination of a number of technology fields (controlling, processing, automation, planning, AI, etc,,,)
  - Interaction to the physical world
    - ✓Mixed processing of real-time and non real-time
    - ✓Huge amount of data to be operated
  - Limited computational resources and power

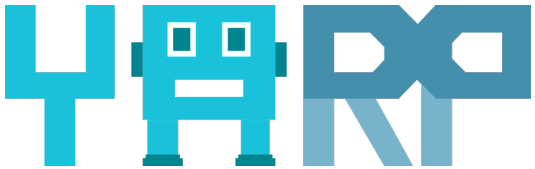
Unified design framework would accelerate development of robot systems

Robot Software Framework

- To accelerate the development,,,
  - enhancing the reusability of modules
  - managing inter-process and inter-system easily

Mainly for kinematics and dynamics
http://www.orocos.org/

CORBA-based architecture
http://openrtm.org/



Transport-neutral communication
http://www.yarp.it/

de-fact around the world!!
http://www.ros.org/

ROS History

2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

1st commit to SourceForge (Nov. 2007)
https://sourceforge.net/p/ros/code/1/log/?path=

ROS 1.0 released

Box Turtle







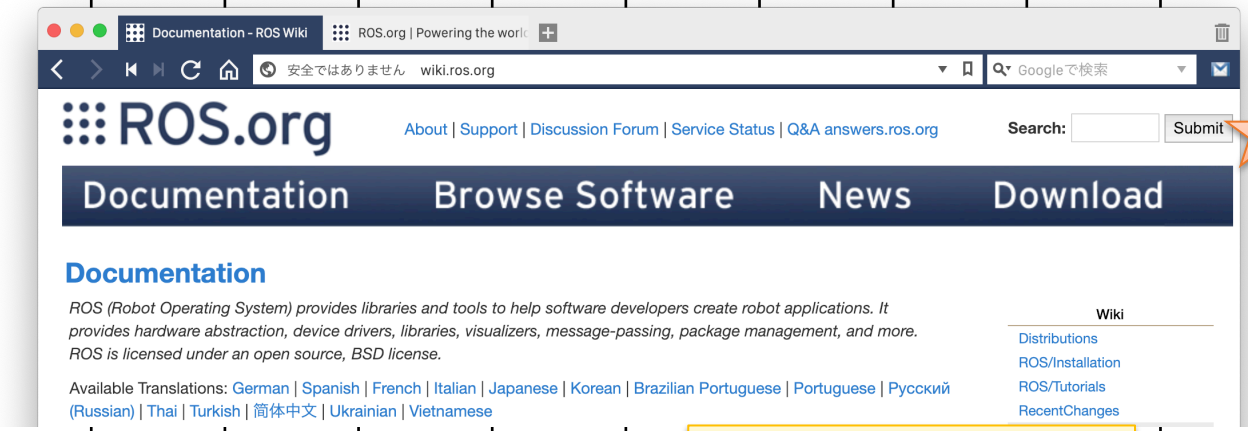

ROS 2 Alpha released







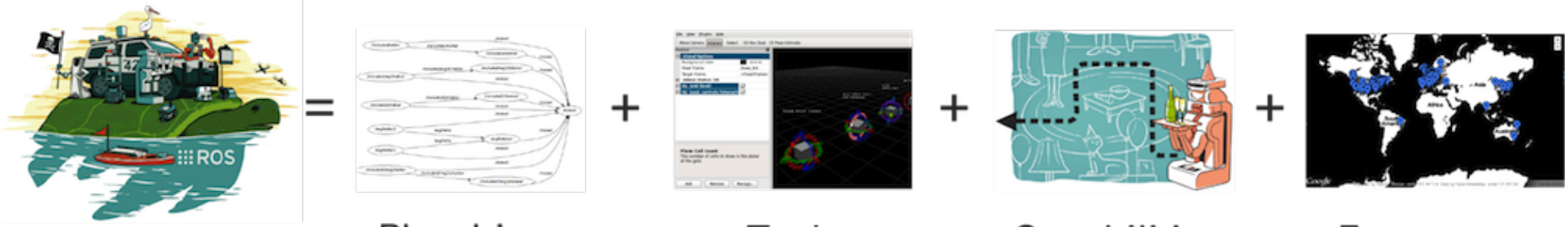
http://www.ros.org/news/2017/11/ros-turns-10.html



http://wiki.ros.org

What is ROS?

ROS is not just framework,
but design platform for robots!!

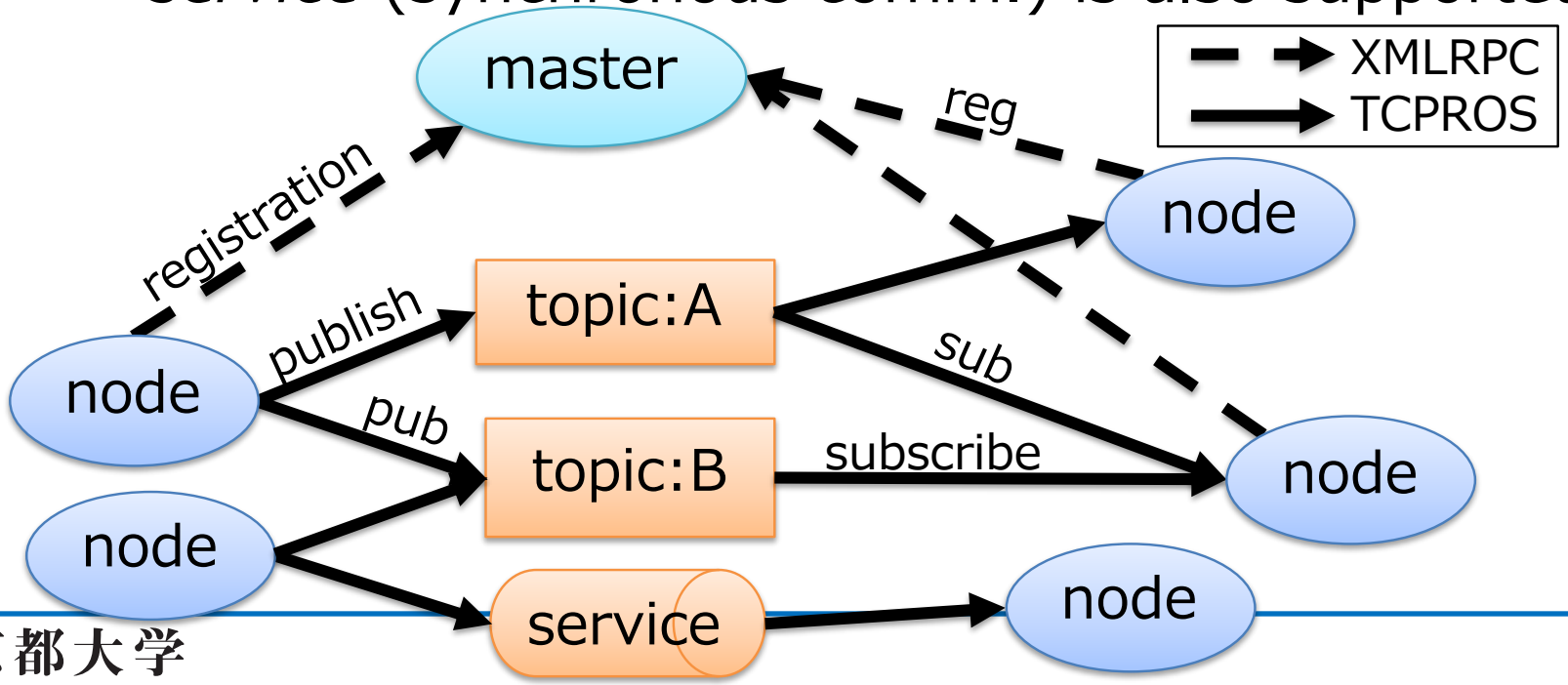


http://www.ros.org/about-ros/

- Plumbing: pub/sub messaging infrastructure
- Tools: configuring, debugging, visualizing, etc.
- Capabilities: broad collection of libraries
- Ecosystem: world-wide powerful community

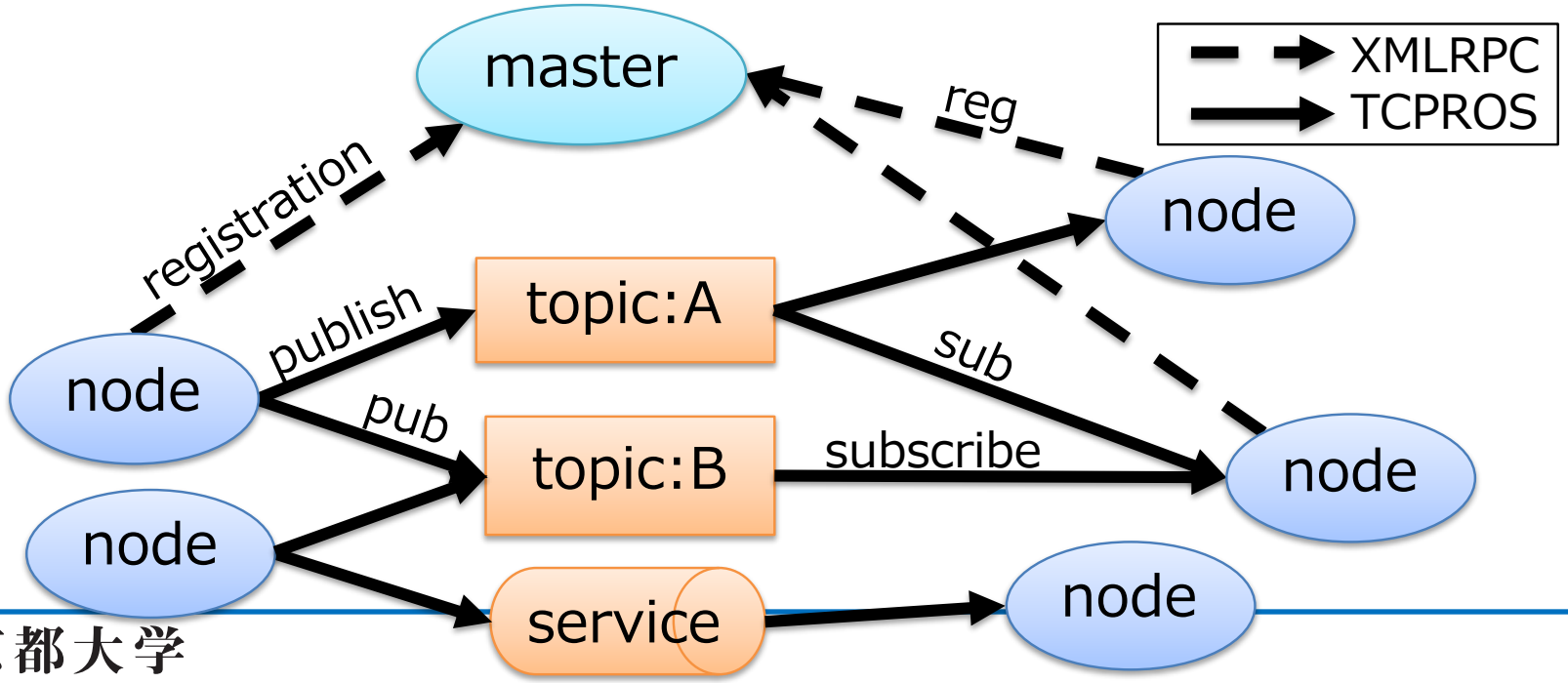
Plumbing

- Publish / Subscribe messaging infrastructure
  - ✓ ROS is not OS, but middle-ware (on Ubuntu)
  - – Asynchronous comm. between nodes via topics
  - – Master (*roscore*) manages the registration of node
    - ✓ *service* (synchronous comm.) is also supported



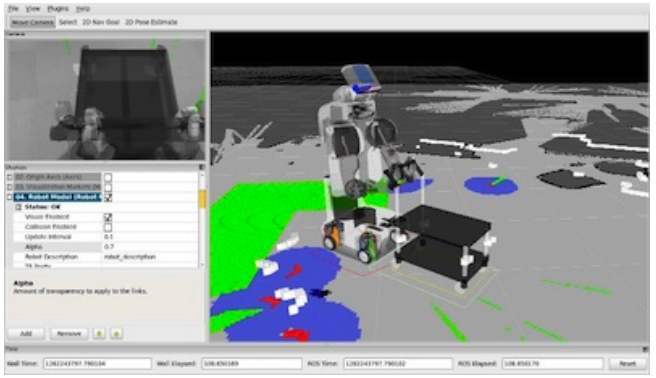
Plumbing

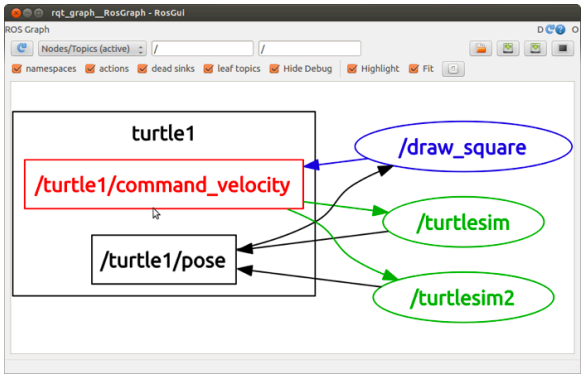
- Benefits
  - If a process/node crashes, it can be restarted
  - A functionality can be exchanged by replacing node

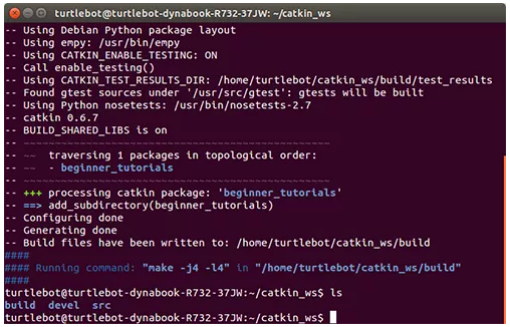


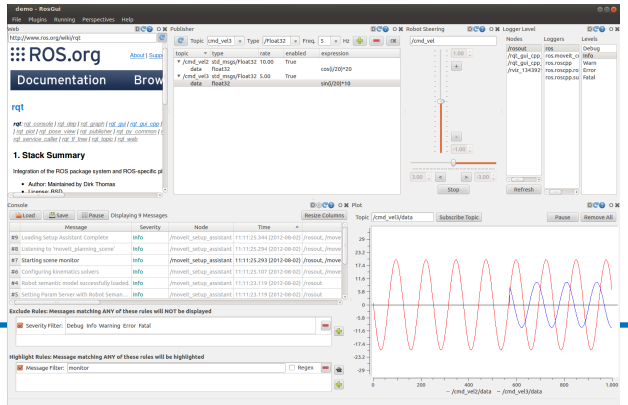
Tools

- catkin_tools: CLI of config. & build system
- rqt: Qt-based debugging framework
- gazebo: 3D physical simulation tool
- rviz: visualization tool
  - and roslaunch, rosbag, tf, etc.,,,









Capabilities

- ROS package:
  - a broad collection of libraries that implement useful robot functionality
    - ✓ mobility, manipulation, perception, etc.,,,
  - 2,000+ packages are available as open-source
- Advantages of package:
  - enhance reusability of resources
  - realize component based development approach
- Officially supported languages:
  C++, Python, LISP
  - Others: C#, Java, Lua, Go, ruby, etc.,,,

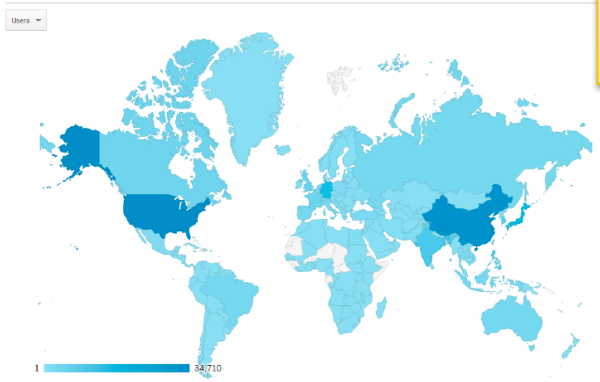
Ecosystem

- On-line community
  - ROS Wiki: documentation & download site
  - ROS Answers: Q&A community site
  - ROS Discourse: announcement of release & events
- Off-line community
  - **ROSCon**: international developers conference
  - SIG meetups, tutorial workshop, local events, etc.,,,

ROS around the world!!

wiki.ros.org visitor locations:

1.	United States	34,710 (19.08%)
2.	China	31,946 (17.56%)
3.	Japan	15,518 (8.53%)
4.	Germany	12,711 (6.99%)
5.	India	8,400 (4.62%)
6.	Philippines	7,235 (3.98%)
7.	South Korea	6,790 (3.73%)
8.	United Kingdom	4,325 (2.38%)
9.	Taiwan	4,233 (2.33%)
10.	France	3,725 (2.05%)
11.	Canada	3,354 (1.84%)
12.	Spain	2,955 (1.62%)
13.	Singapore	2,842 (1.56%)
14.	Italy	2,744 (1.51%)
15.	Russia	2,465 (1.35%)
16.	Indonesia	2,461 (1.35%)
17.	Australia	2,436 (1.34%)
18.	Brazil	2,231 (1.23%)
19.	Hong Kong	2,147 (1.18%)
20.	Turkey	1,928 (1.06%)
21.	Netherlands	1,511 (0.83%)
22.	Thailand	1,437 (0.79%)
23.	Poland	1,335 (0.73%)
24.	Switzerland	1,242 (0.68%)
25.	Vietnam	1,125 (0.62%)



Source: Google Analytics
Site: wiki.ros.org in July 2018

6

http://download.ros.org/downloads/metrics/metrics-report-2018-07.pdf

ROS Robots

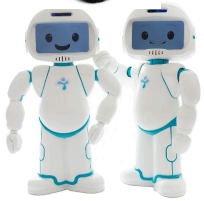
Aerial




Ground

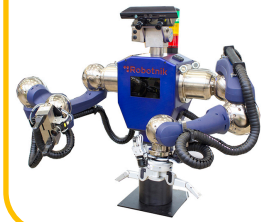
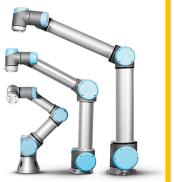




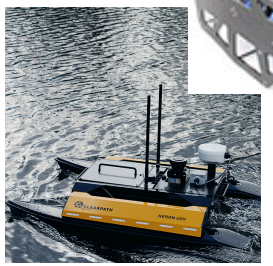
Manipulator



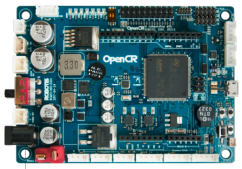



Marine





Component

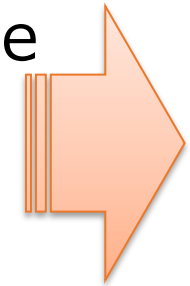

Sensor




Motor


Transition of Use Case

- Single robot
- Workstation-class computational resource
- No real-time control
- Excellent connectivity of network quality
- Application in research



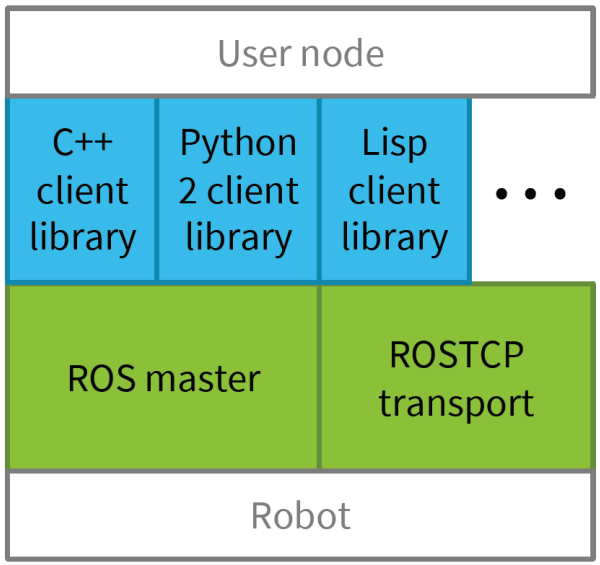
- Multiple robots
- Demands for small embedded platforms
- For real-time control
- Non-ideal networks (loss and/or delay)
- Production environment

Development of 2 has been started from the ground up!!

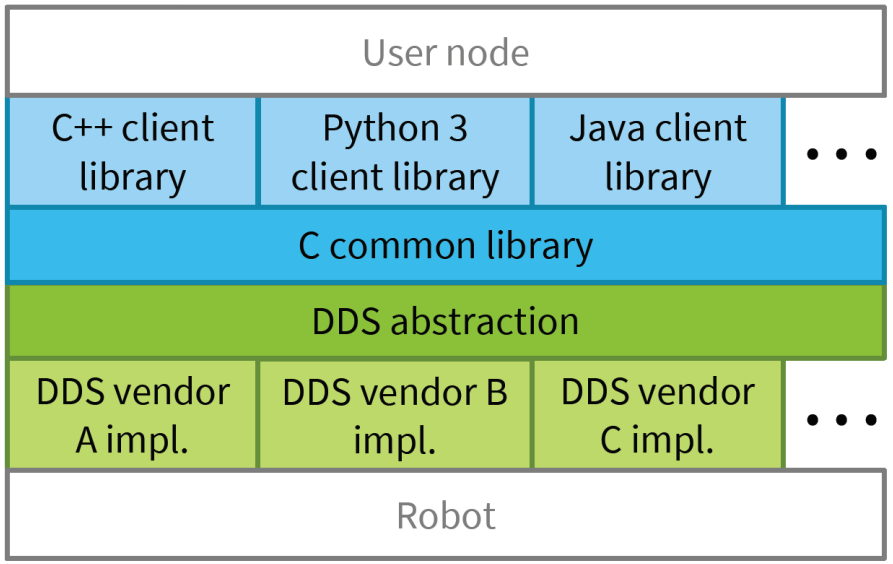
New Features of ROS 2

- DDS (Data Distribution Service)
  - ROS-like pub/sub transport protocol
    - ✓No master is needed for managing comm.
  - Technical credibility by OMG standard specification

https://speakerdeck.com/youtalk/dds?slide=12



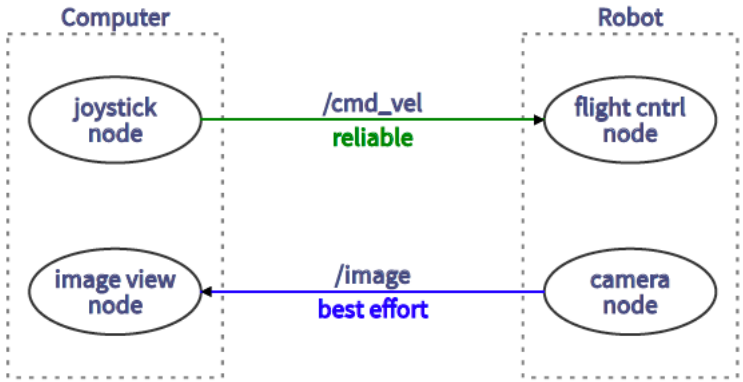
ROS 1

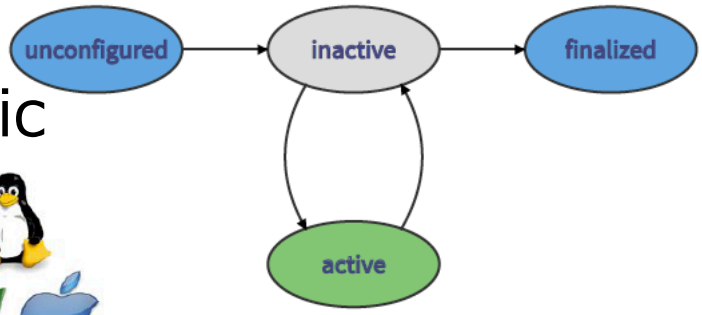


ROS 2

New Features of ROS 2

- QoS control
  - reliability, history policy, and durability for each topic
- Lifecycle state machine
  - 4 states with error handling
  - State change is notified by topic
- Multi platform support
  - Ubuntu, macOS, Windows
  - (experimental) bare-metal / RTOS with tiny DDS





- And much more improvement,,,

Contents

- ROS: How to accelerate the development
  - Background & Motivation
  - History of robot software framework
  - Features of ROS
  - ROS 2: next-generation development platform
- **mROS: How to integrate embedded devices**
  - **Motivation & Our Strategy**
  - **Overview & Structure**
  - **Current Status & Case Study**
- Conclusion & Announcement

Motivation

ROS

- Enormous amount of OSS packages
- Linux/Ubuntu is needed

ROS 2

- multi-platform support including embedded
- Not so many packages

- Embedded technology would contribute to power consumption & real-time capability
- There is no compatibility between ROS 1 and ROS 2

Our Strategy

ROS

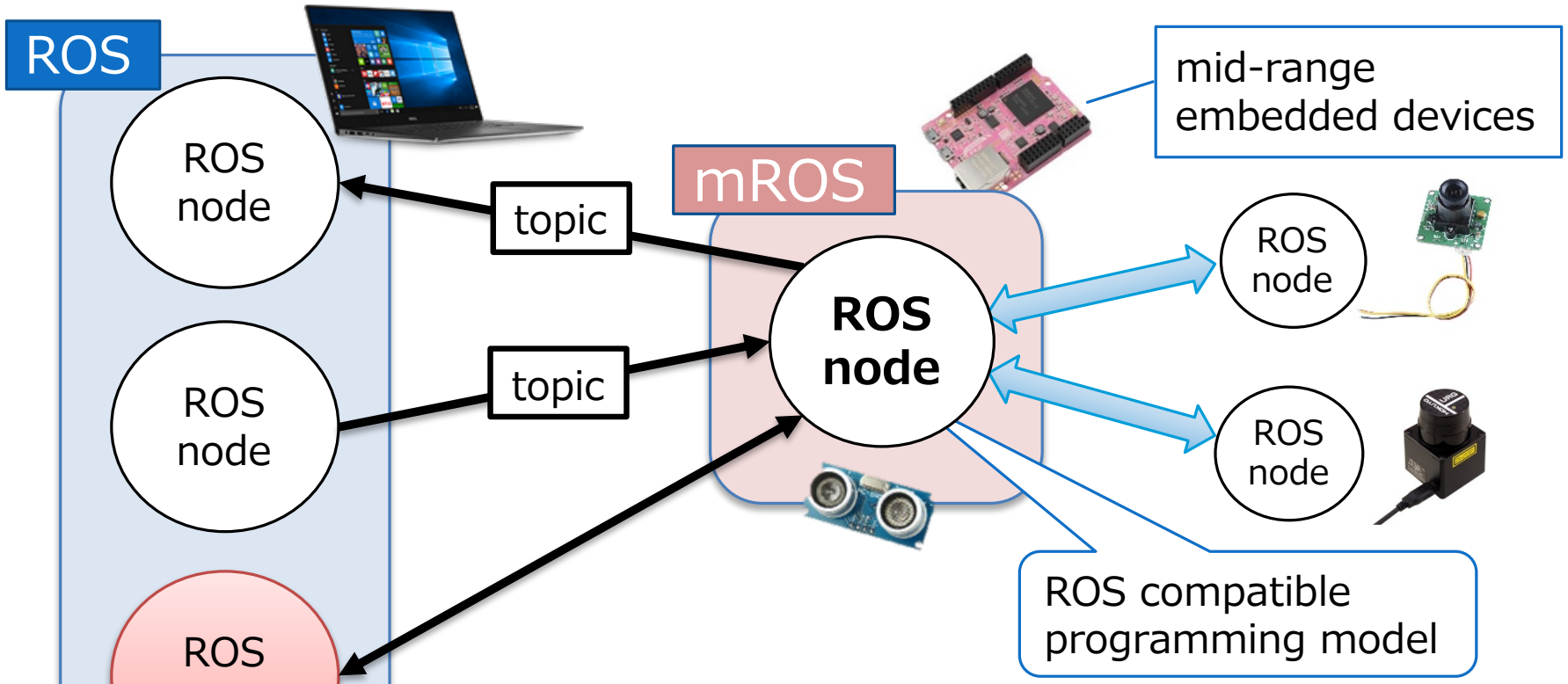
- Enormous amount of OSS packages
- Linux/Ubuntu is needed

- power-efficient
- real-time capability
- Severe resource restriction

ROS

lightweight runtime environment that enables to integrate embedded technology onto ROS 1 world

Overview of mROS



Contributions of mROS

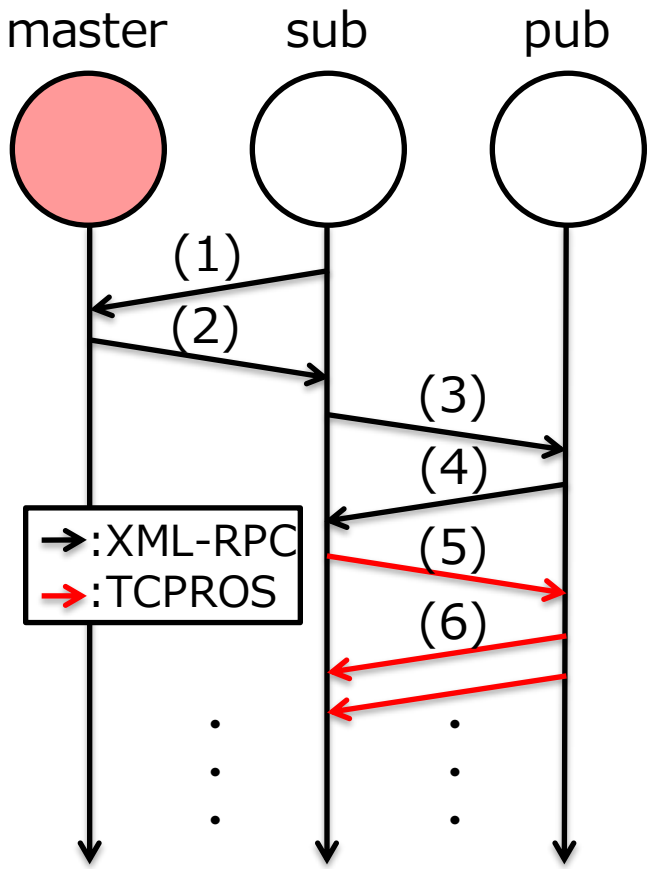
- ✓ Portability of ROS 1 packages to embedded devices
- ✓ Enhancement of power saving & real-time performance for edge nodes on distributed robot systems

Requirements & Goals

- Target: mid-range class embedded devices
  - RTOS and TCP/IP stack can be operated
  - Linux kernel cannot be operated
    - ✓No high-end control unit (e.g., MMU)
- Requirements for performance
    - ✓mROS as edge devices on distributed systems
  - Publishing data: QVGA image from camera
    -> approximately 512 KB of data at 100 ms interval
  - Subscribed data: the control instruction data set
    -> less than 1 KB in 1 ms
- Memory size for 10 MB at most

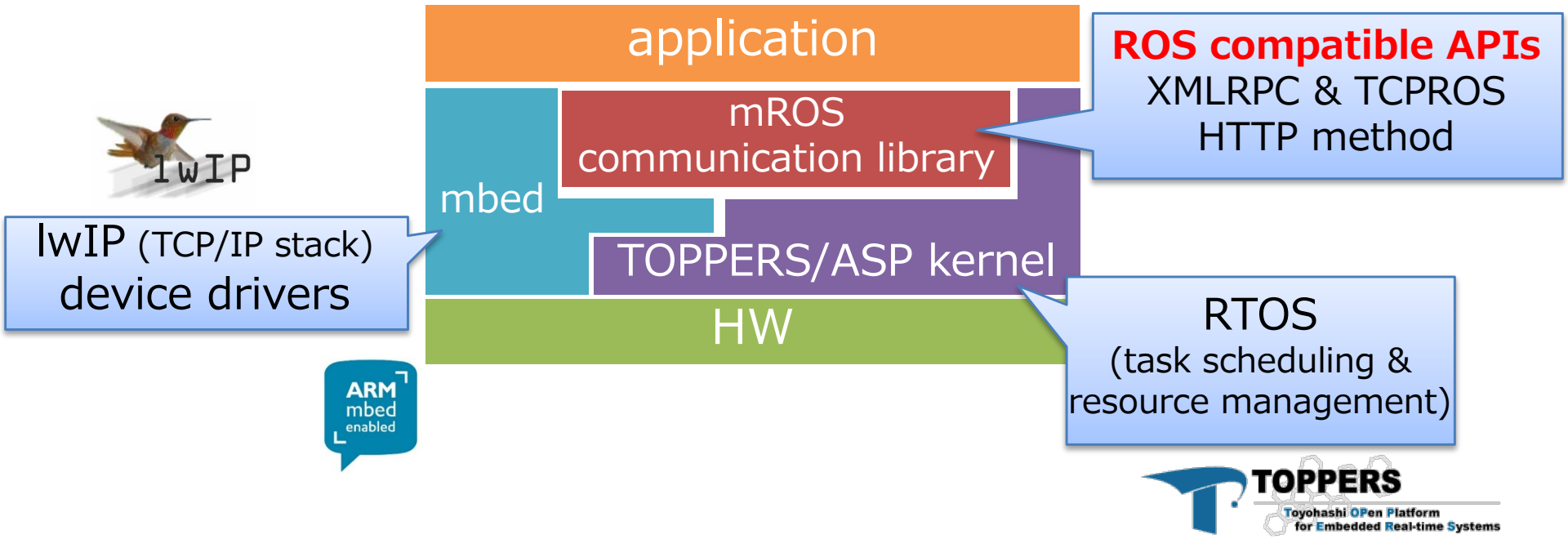
Supported Function

- data publication
- data subscription
- procedure call between the ROS master and other nodes
- acceptance of procedure call from nodes to the master

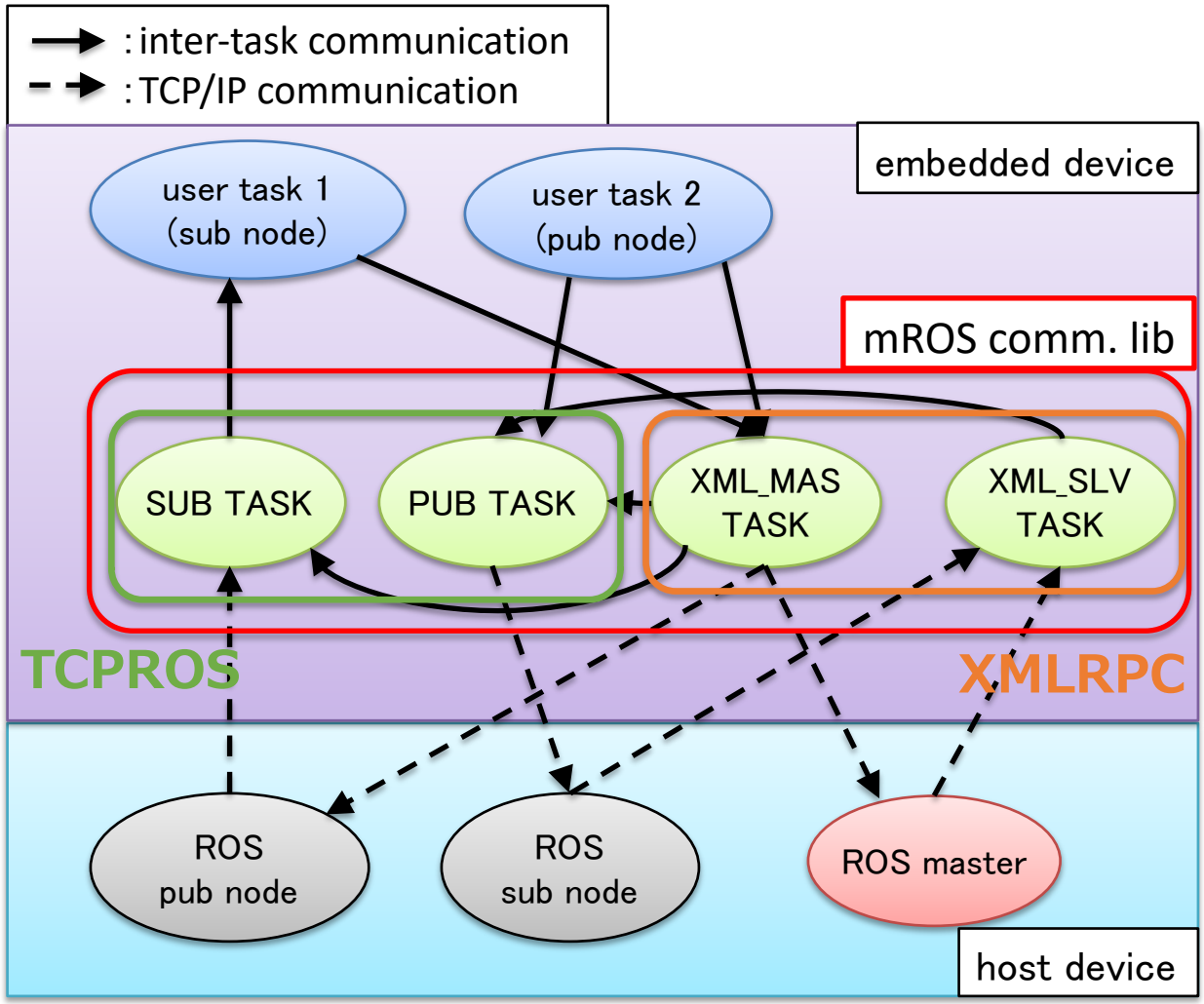


Software Structure

- mROS tasks can be designed by ROS APIs
  - Device programing can be realized with mbed lib.
  - Multi-tasks (multi-nodes) execution can be realized by TOPPERS programming model



Tasks for mROS comm. lib.

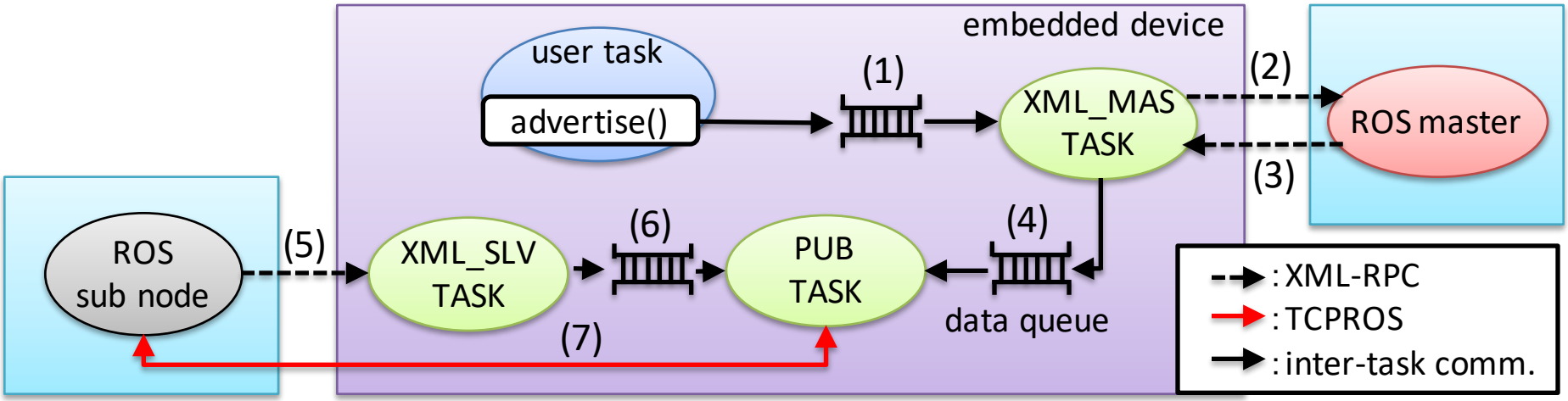


Shared memory and data queue for task communication

SUB & XML_SLV TASKs are periodically executed

Execution Flow of advertise()

- Registration of publication task (node)



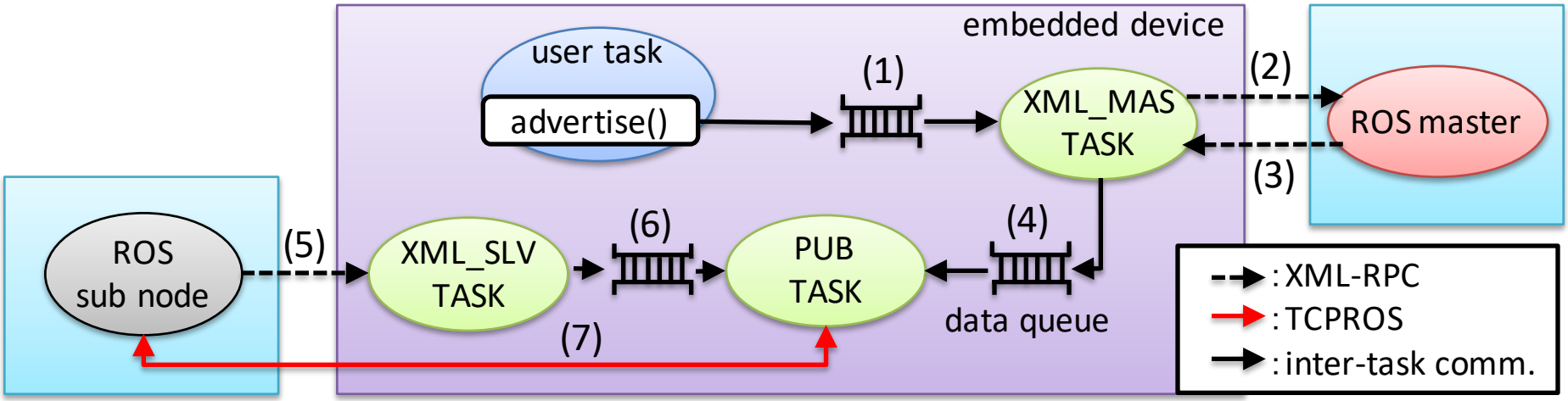
(1) initialization request of topic name via data queue and assignment of node ID

(2) generation of XML header and sending it to master

(3) response of the registration result

Execution Flow of advertise()

- Registration of publication task (node)



(4) notification of message ID and generation of TCP socket

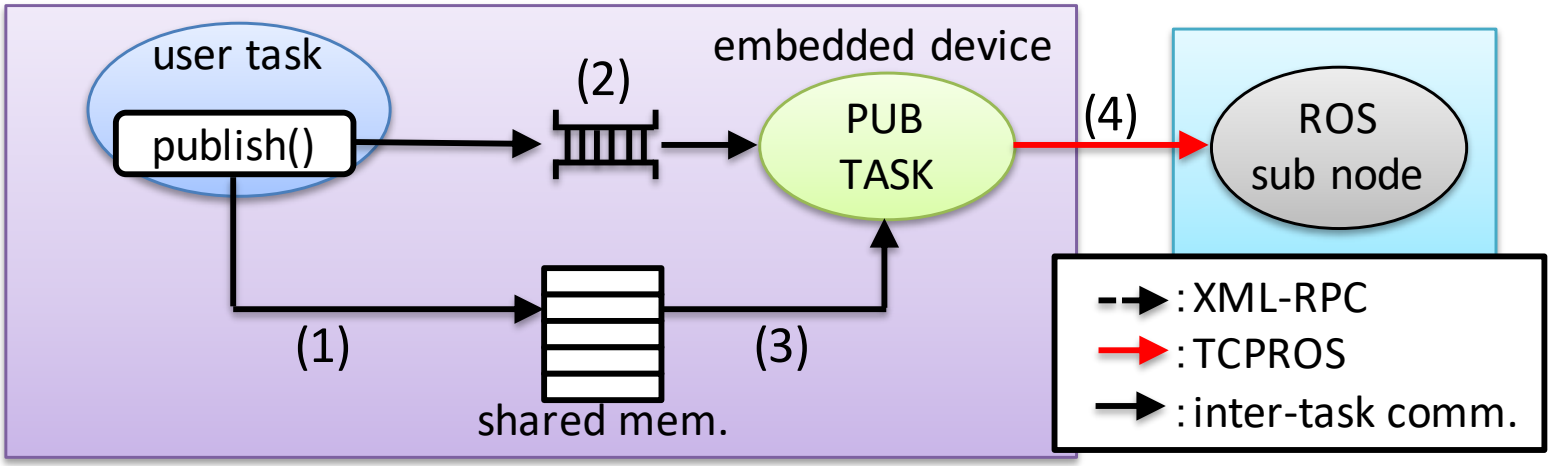
(5) notification of port number when request of topic occurs

(6) sending arrival request of topic

(7) establishment of TCPROS connection

Execution Flow of publish()

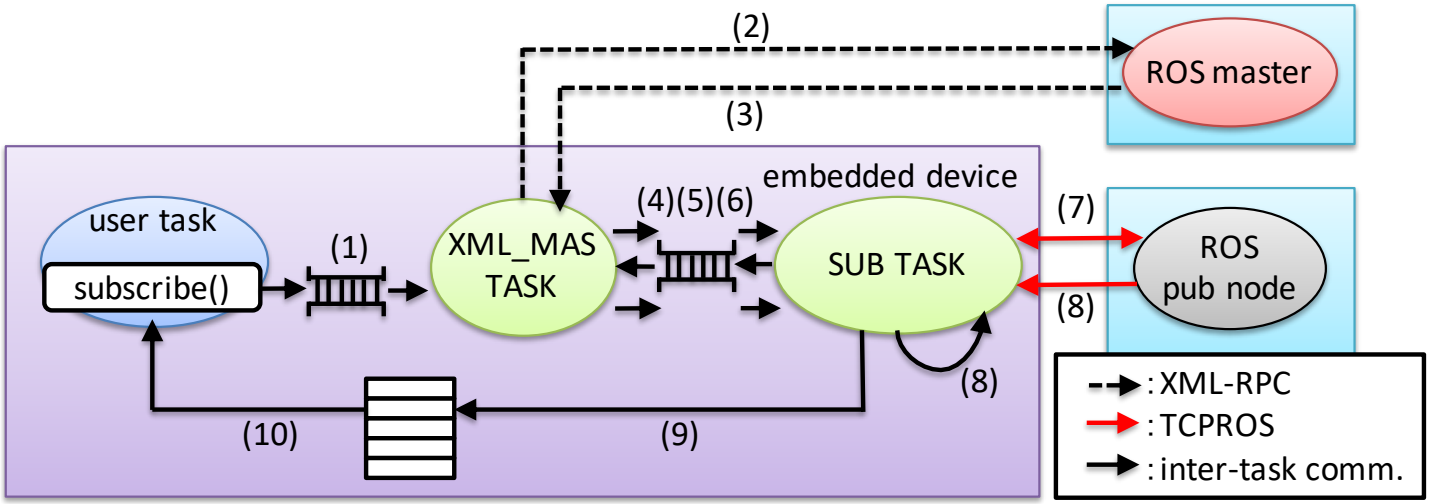
- Publication of data to external subscription node



(1) writing the publication data in shared memory
(2) generation of message ID and sending it
(3) receiving and encoding message ID
(4) searching TCP socket, and publishing data to sub node

Execution Flow of subscribe()

- Registration of subscription task (node)



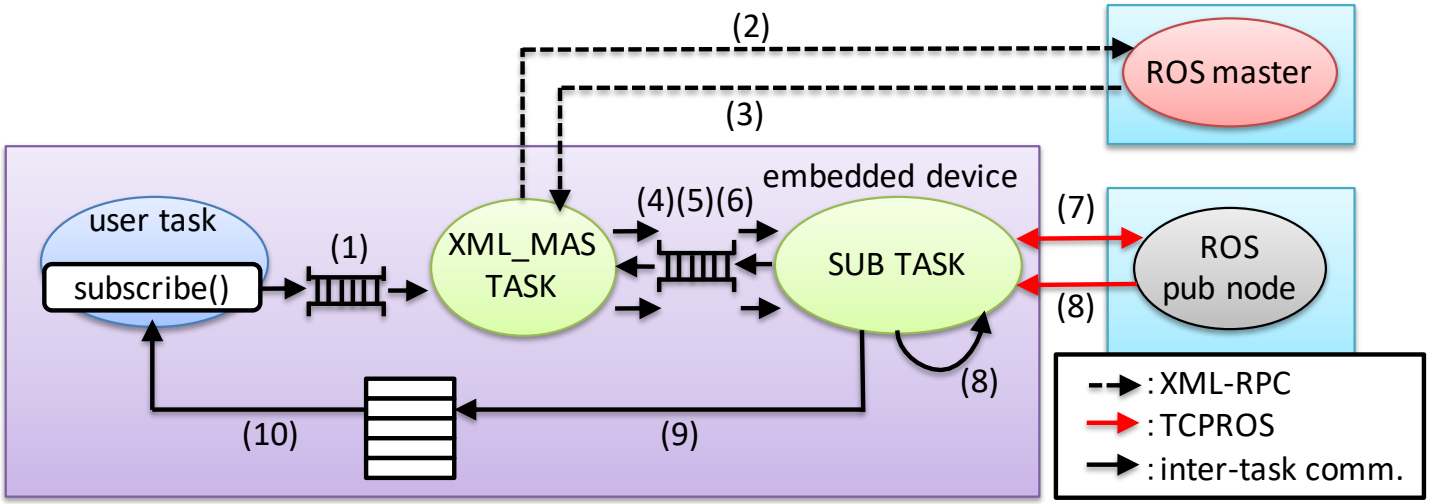
(1) notification of the initialization request

(2) generation of XML header and sending it to master

(3) response of the registration result and sending URI of external pub node

Execution Flow of subscribe()

- Registration of subscription task (node)



(4) notification of initialization request of user task

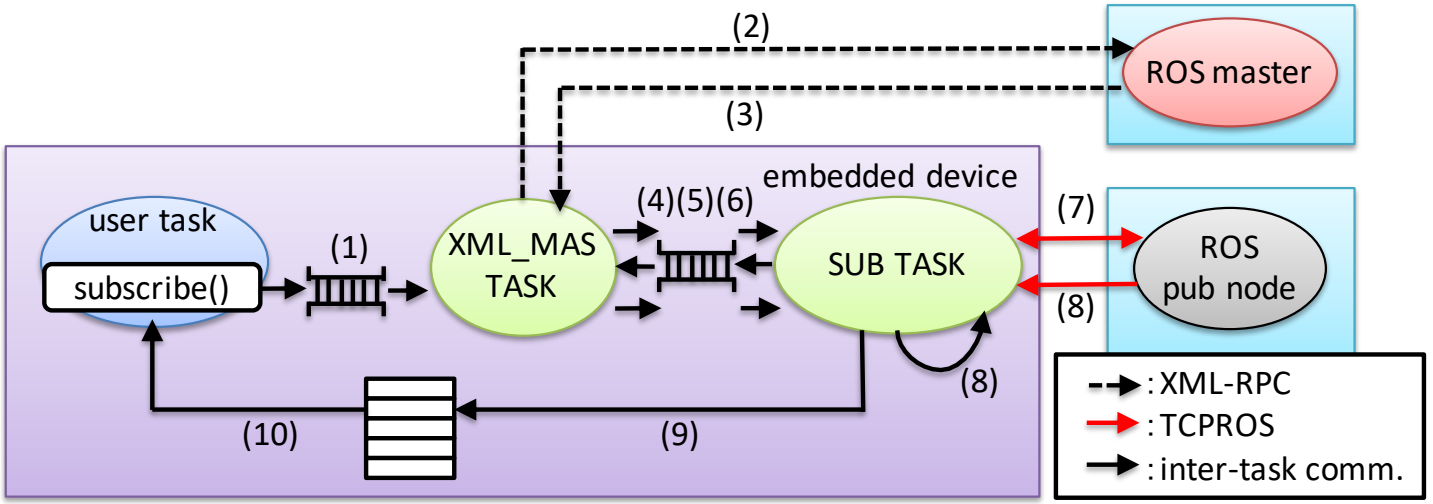
(5) sending the request for subscription

(6) notification of the port number of the external pub node

(7) generation of TCPROS connection header for subscription request

Execution Flow of subscribe()

- Subscription of topic



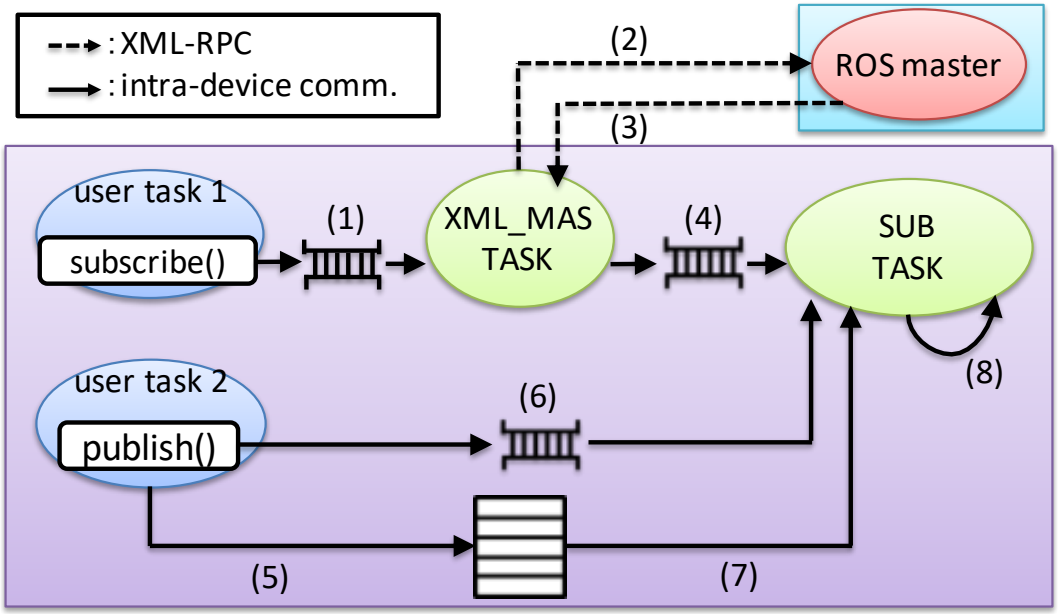
(8) activation of callback in SUB TASK

(9) writing the return value of callback

(10) getting and using the return value from shared memory

Intra-Device Communication

- Communication via shared memory



(1) notification of initialization request for subscription

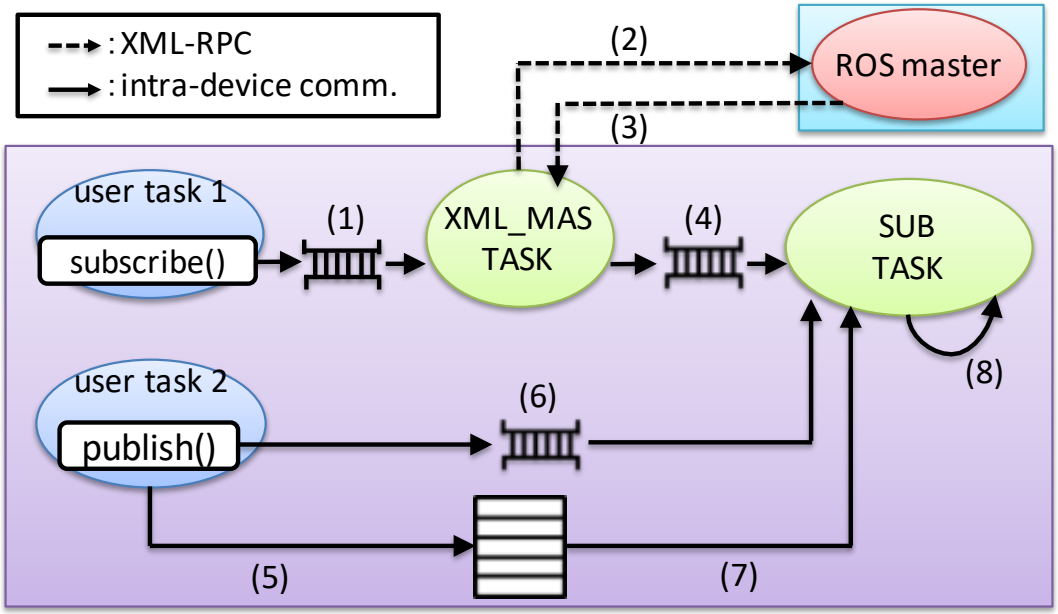
(2) generation of XML header and sending it to master

(3) response of the registration result and sending URI of pub task

(4) notification of registration result and
appending the node info. to node list where pub task is

Intra-Device Communication

- Communication via shared memory



(5) writing data in the shared memory by publish()

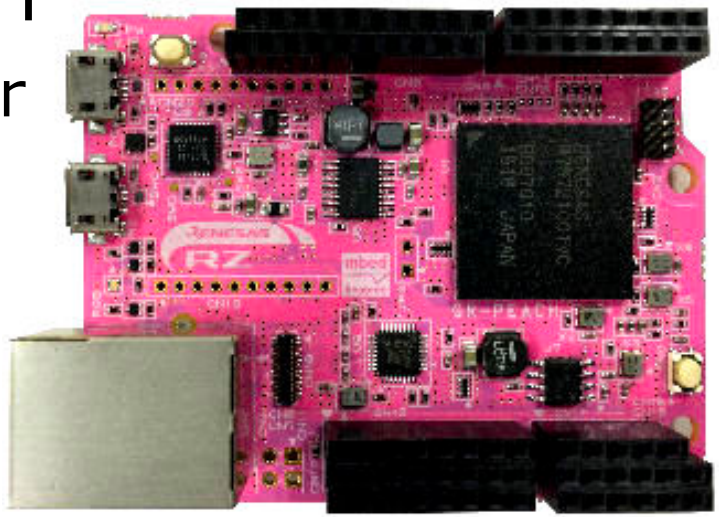
(6) sending the generated message ID

(7) reading topic data from shared memory

(8) activation of callback

Implementation Setup

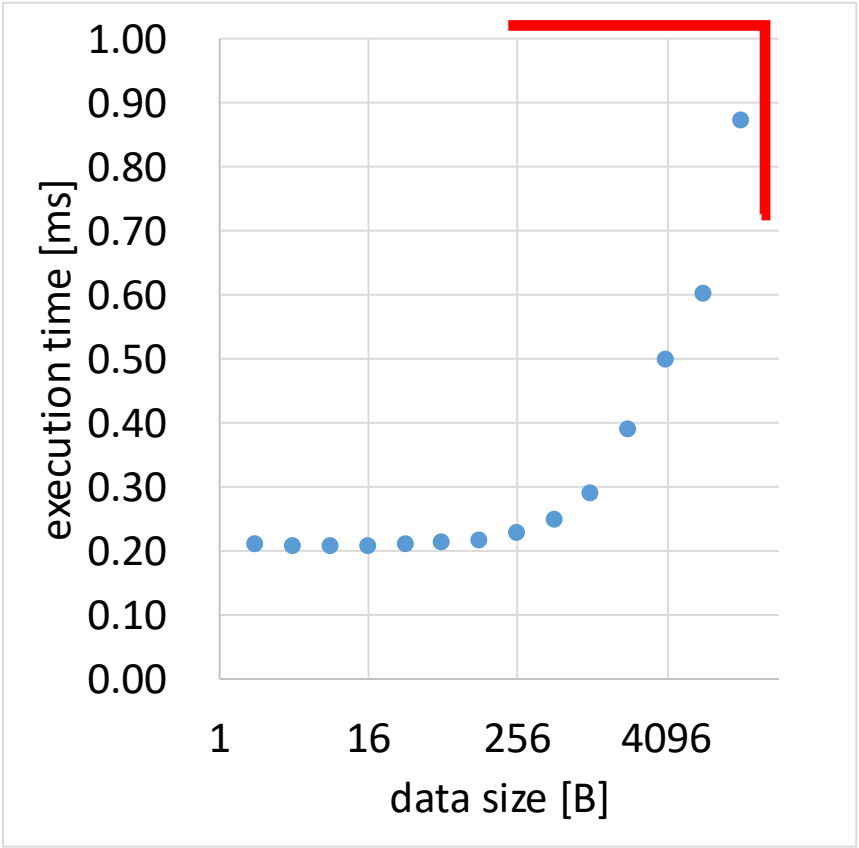
- Target: Renesas GR-PEACH
  - 400 MHz Cortex-A9 processor
  - 8 MB Flash memory
  - Arduino compatible pins
  - CMOS camera shield
  - IDE: Atollic TrueSTUDIO

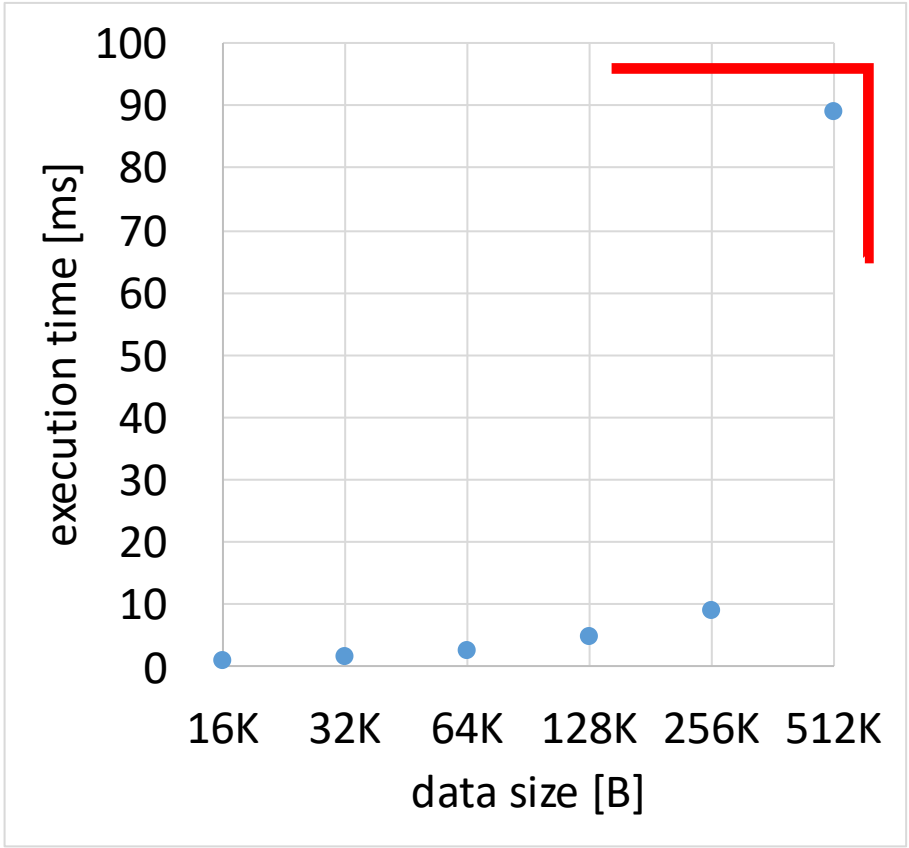


- Host: NEC's LAVIE HybrydZERO
  - Ubuntu14.04 LTS
  - ROS indigo
  - Intel Core i7 2.4 GHz, 16 GB memory

Evaluation: publish()

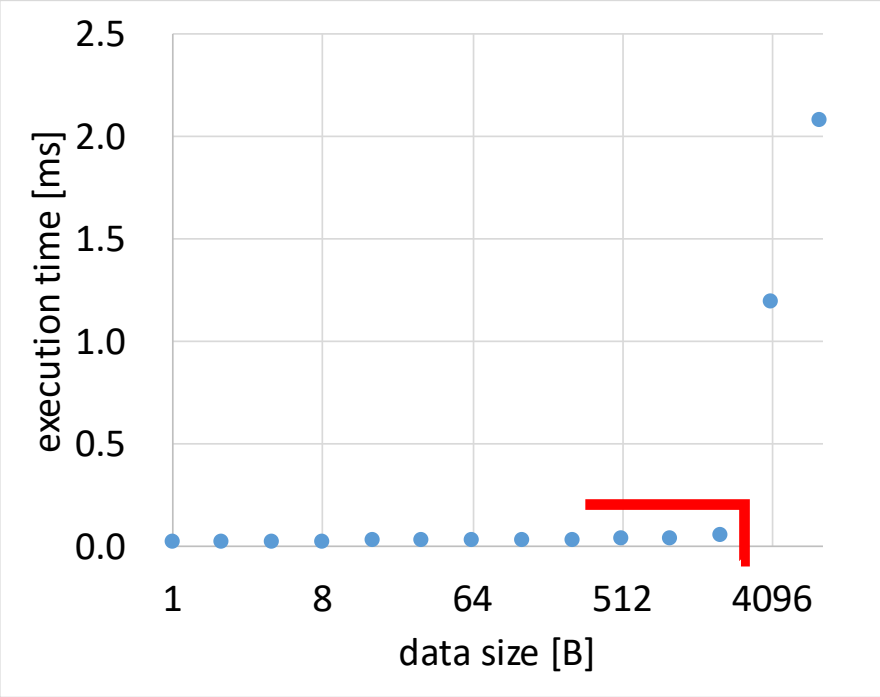
- less than 16 KB of data can be published in 1 ms
- 512 KB of data can be published in less than 100 ms

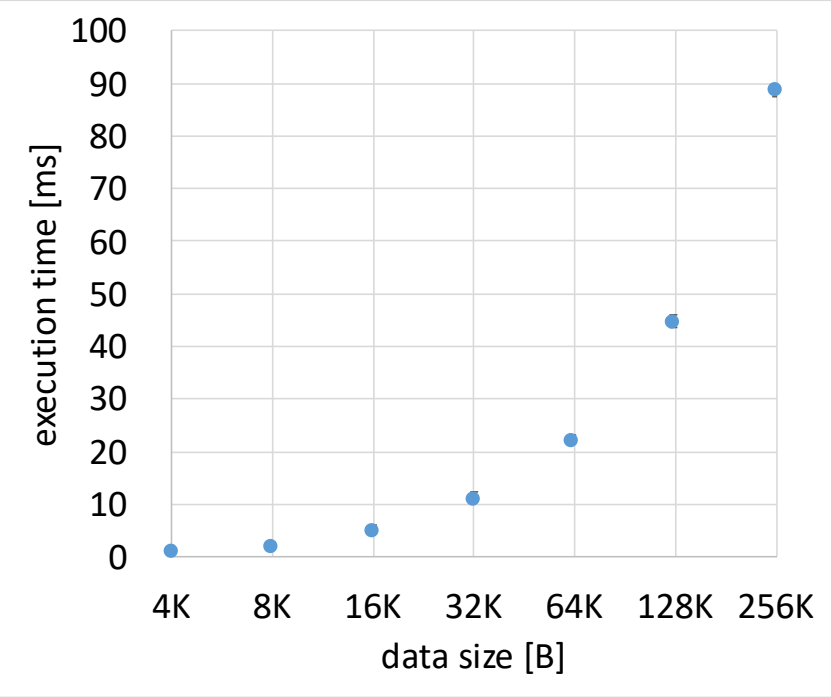




Evaluation: subscribe()

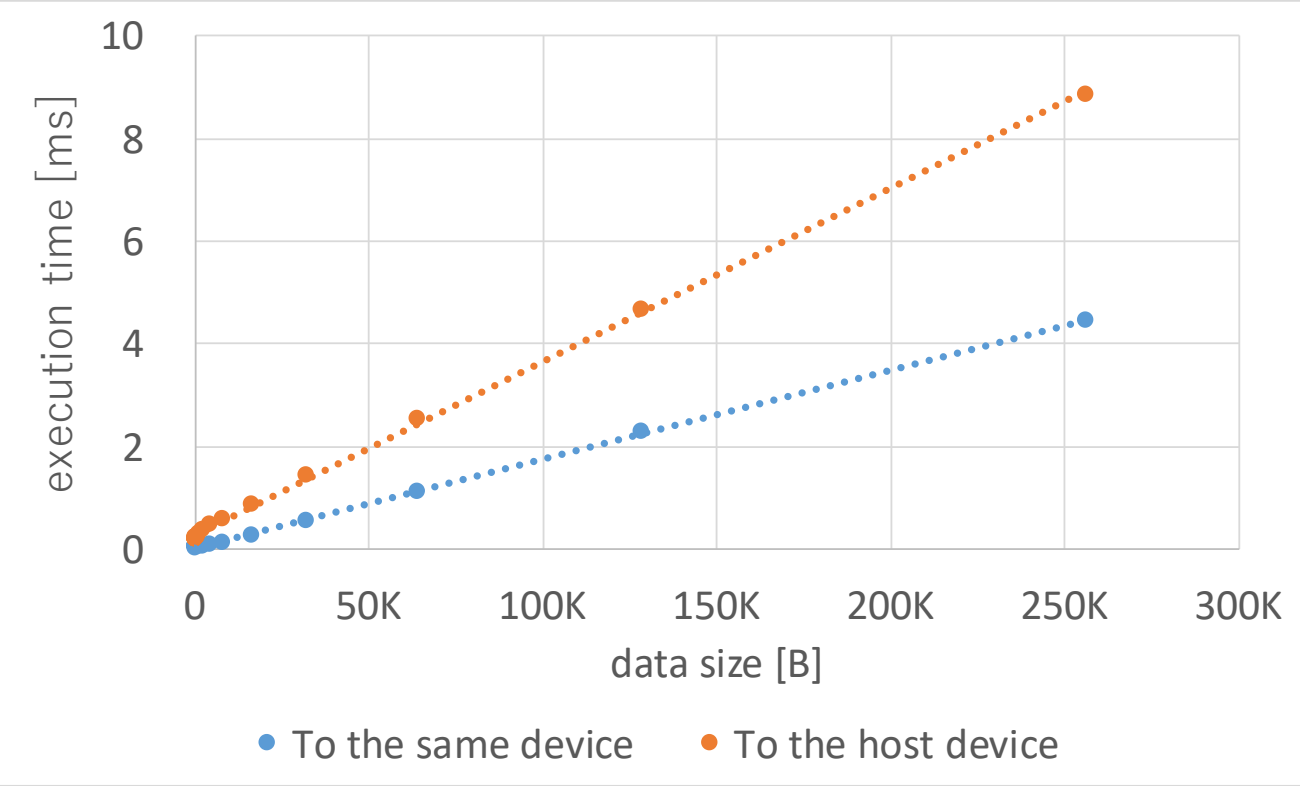
- Less than 2 KB can be executed in less than 0.1 ms
  - Large data to subscribe is not assumed





Evaluation: Intra-Device

- Data publication to the same device is much smaller regardless of the data size



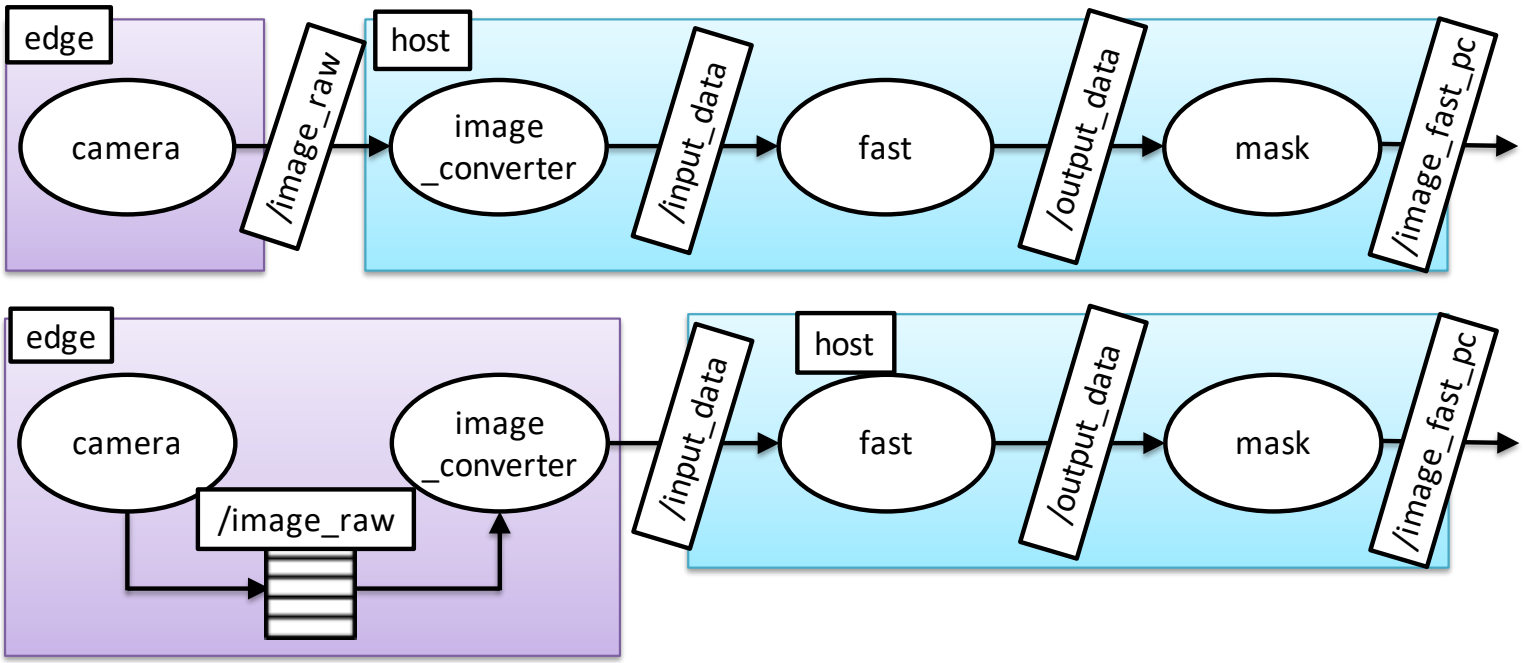
Evaluation: Memory Size

	text	data	bss	total
kernel	99,676	0	16,408	116,084
mbed lib.	264,477	52,940	45,711	363,128
mROS lib.	57,950	28	2,097,310	2,155,288
total	422,103	52,968	2,159,429	2,634,500

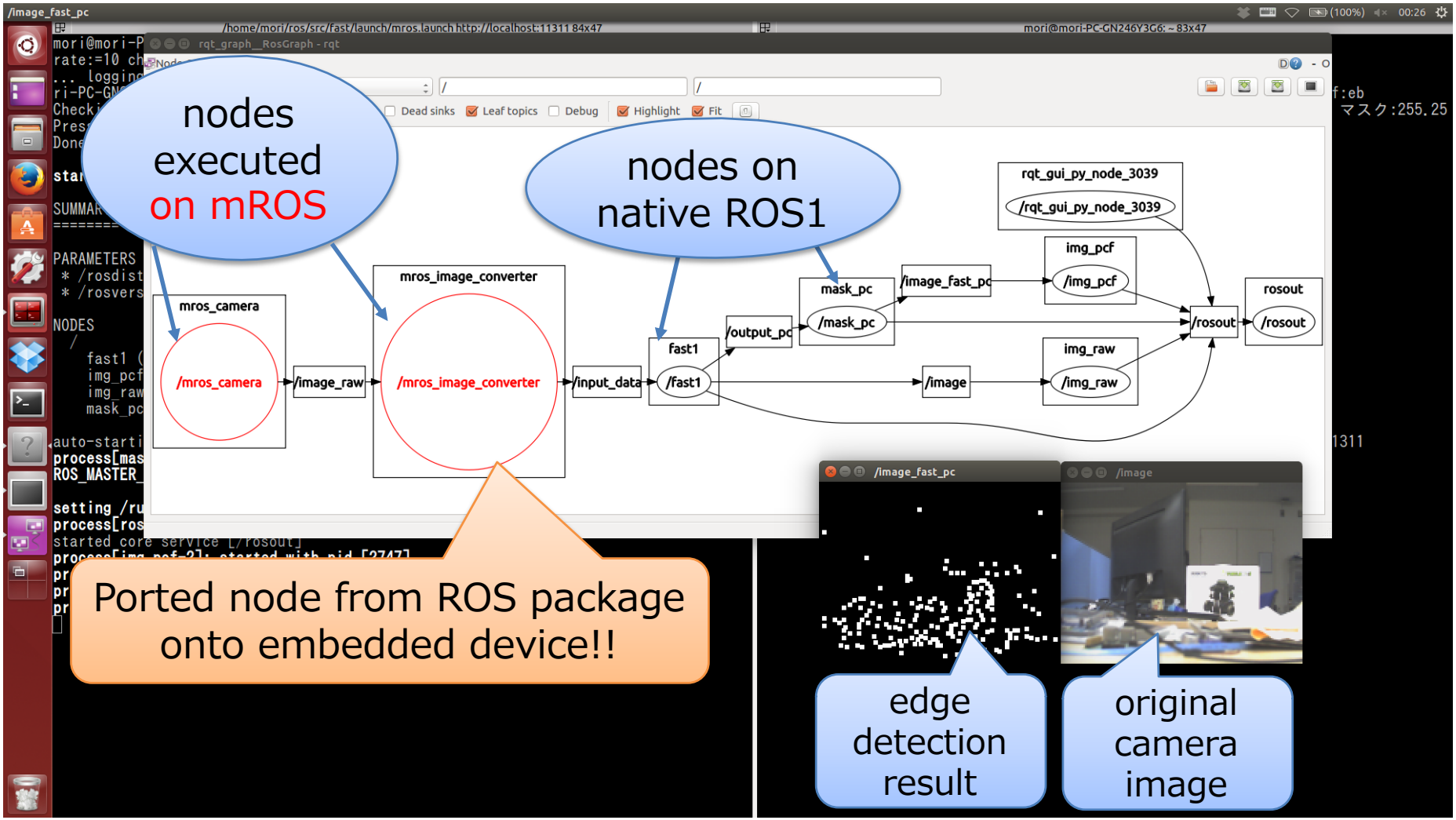
- Approximately 2.6 MB
  - sufficiently lightweight for GR-PEACH
  - It is possible to make the size smaller by adjusting the size of the shared memory by design choice

Case Study:
edge detection from camera

- Operate part of system on the edge
- 2 types of configuration are implemented
  - "image_converter" node is ported on mROS

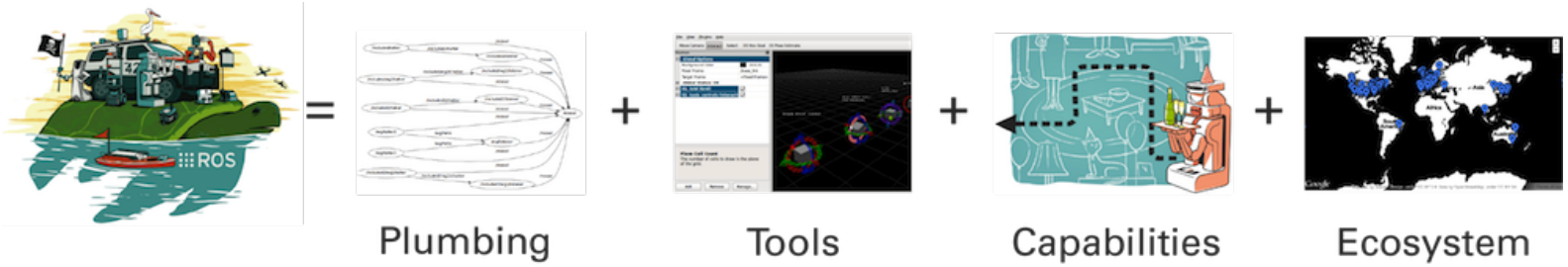


Case Study:
edge detection from camera



Conclusion

- ROS: Robot Operating System



- mROS: a light-weight runtime environment
  - Portability of ROS 1 packages to embedded devices
  - Enhancement of power saving & real-time perf.
- Future directions
  - Support for synchronous communication of ROS
  - Quantitative Evaluation of power savings
  - Integration of FPGA/MPSoC environment?

Acknowledgements & References

- **Acknowledgement**: The part of this work was supported by JST, PRESTO Grant Number JPMJPR18M8, Japan.

- Yutaka Kondo: Getting Started with ROS 2 / DDS, ROS Japan User Group #27, Dec 2018. https://speakerdeck.com/youtalk/dds
- Geoffrey Biggs: Introduction of Next-Generation Robot Framework ROS 2 (in Japanese), 20th Summer Workshop on Embedded System Technologies (SWEST20), Aug 2018. https://swest.toppers.jp/SWEST20/program/s2a.html#s2
- Tully Foote: ROS Community Metrics Report, Jul 2018. http://download.ros.org/downloads/metrics/metrics-report-2018-07.pdf
- Dirk Thomas, Mikael Arguedas: The ROS 2 Vision -For Advancing the Future of Robotics Development-, ROSCon 2017, Sep 2017. https://roscon.ros.org/2017/presentations/ROSCon%202017%20ROS2%20Vision.pdf

Acknowledgements & References

- ROS Wiki: http://wiki.ros.org
- ROS Answers: https://answers.ros.org
- ROS Discourse: https://discourse.ros.org

- Hideki Takase, Tomoya Mori, et al.: Work-in-Progress: Design Concept of a Lightweight Runtime Environment for Robot Software Components Onto Embedded Devices, Proc. of EMSOFT, Oct 2018. https://ieeexplore.ieee.org/document/8537199
- Yasuhiro Nitta, Sou Tamura, Hideki Takase: A Study on Introducing FPGA to ROS Based Autonomous Driving System, Proc. of FPT, Dec 2018.
- Takeshi Ohkawa, Yutaro Ishida, Yuhei Sugata, Hakaru Tamukoh: ROS-Compliant FPGA Component Technology - FPGA installation into ROS, ROSCon 2017, Sep 2017. https://roscon.ros.org/2017/presentations/ROSCon%202017%20ROS%20Compliant%20FPGA.pdf