A new RMW for RTI Connext DDS

Andrea Sorbini
SENIOR SOFTWARE ENGINEER

Presentation Agenda

- ROS
- RTI Connext DDS for ROS
- ROS + Connext

ROS

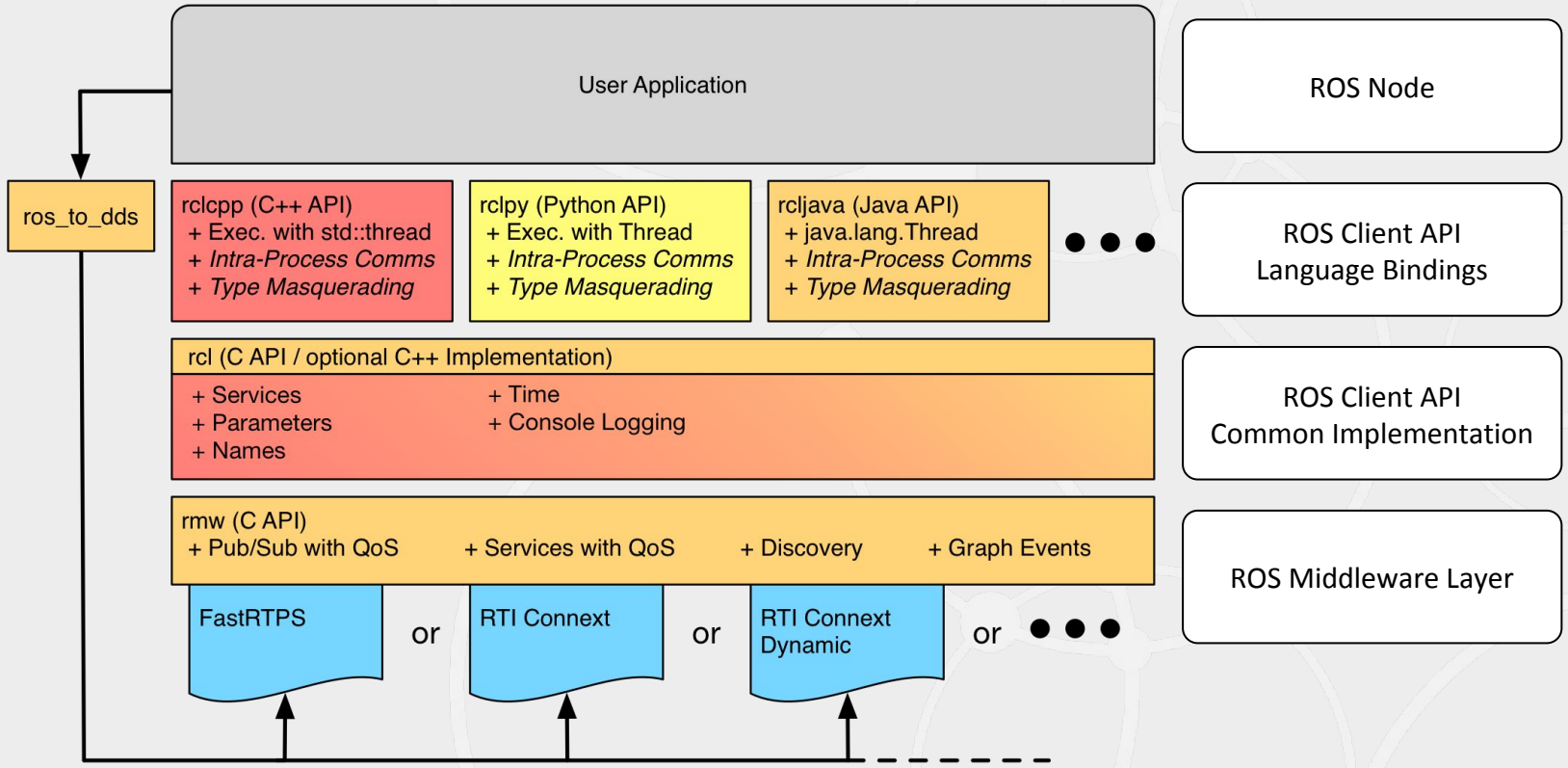
ROS

ROS2

- **R**obot **O**perating **S**ystem
- Open-source middleware for robotic applications developed by Open Source Robotics Foundation (OSRF).
  - Component-oriented (nodes).
  - Topic-based pub/sub with typed messages (ROS IDL).
  - Remote method invocation (clients/services).
  - Official API language bindings: C++, Python.
- Version 2 adopted DDS as its default communication layer.
  - Abstracted by a "middleware layer" (RMW) to support alternative communication technologies.

rti

ROS - Architecture



RMW

- C API used by rcl to create middleware entities and to access their services (e.g. message pub/sub).
- Implementation selected at runtime via environment variable `${RMW_IMPLEMENTATION}`.
- "Tier 1" implementations (Foxy release):
  - RTI Connext DDS Professional (`rmw_connext_cpp`)
  - eProsima FastRTPS (`rmw_fastrtps_cpp`)
  - Eclipse Cyclone DDS (`rmw_cyclonedds_cpp`)

RTI Connext DDS for ROS

rmw_connext_cpp

- Current RMW implementation for *RTI Connext DDS Professional*, developed by OSRF.
  - The first RMW to be implemented for ROS2.
  - *RTI Connext DDS Micro* not supported.
- Design choices cause suboptimal user experience.
  - Bad performance due to extra memory allocations and copies between ROS and DDS data representations.
  - Mangling of DDS type names (e.g. "Foo.bar" -> "Foo_.bar_") hinders out-of-the-box interoperability.

A new RMW for RTI Connext DDS

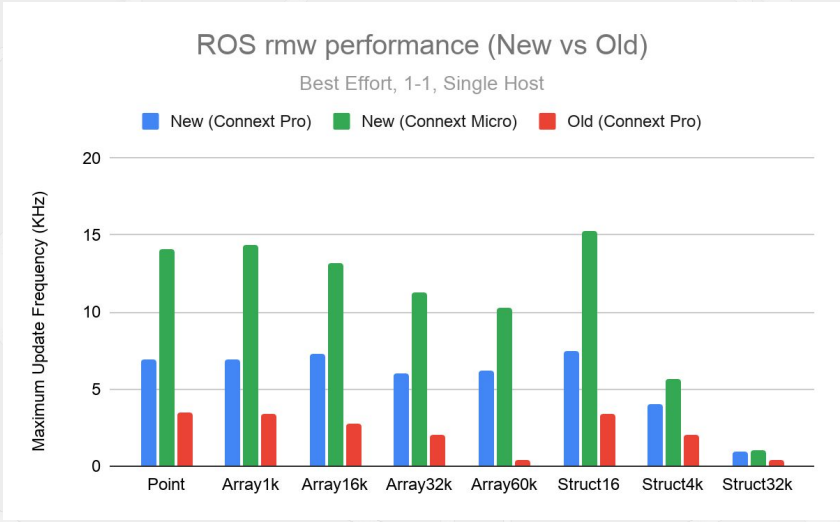
- Two new RMWs developed and supported by RTI.
  - rmw_connextdds
  - rmw_connextddsmicro
- Resolves performance issues by allowing the middleware to handle ROS messages without any transformation.
- Propagates types without name mangling.
- No Connext-specific code generation.

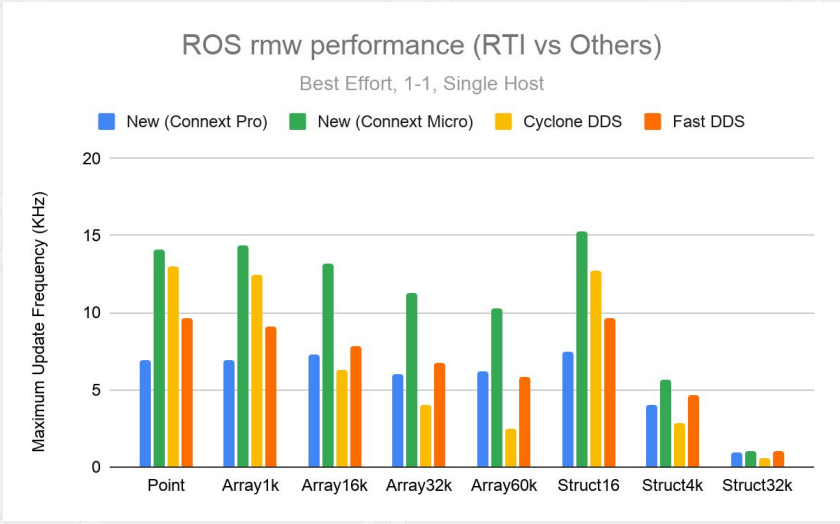
Differences between RMWs

- rmw_connextpro_cpp
  - Propagates type information over DDS Discovery.
  - Request/reply compliant with standard DDS RPC.
- rmw_connextmicro_cpp
  - Static resource limits defined at compile-time.
  - Custom request/reply implementation.
  - Incomplete support for ROS node graph.
  - Additional configuration required (via environment variables).

Improved RMW performance

- Initial results from a simple test stressing throughput between a ROS publisher and subscriber.
- Speedup vs old: 2.5x-6x
- Performance similar to other RMWs.





Roadmap

- Repository available for evaluation on GitHub.
  - Looking for feedback while completing testing and stabilization.
  - Access enabled upon request (write to robotics@rti.com).
- Replace `rmw_connext_cpp` with `rmw_connextdds_cpp` in upcoming ROS releases.
  - Requires validation and adoption by OSRF.
  - Target: Foxy patch release (TBD), Galactic (May 2021).

ROS + Connext

A ROS/Connext interoperability demo

- ROS applications can now easily interoperate with RTI Connext DDS applications and RTI Connext DDS tools.
- Two simple Connext applications interact with turtlesim.
  - Publish topic "rt/turtle1/cmd_vel" to move turtle.
  - Subscribe to topic "rt/turtl1/pose" to detect turtle's position.
- Use ros-data-types repository to simplify development.

RMW Installation

```
# Clone RMW repository in a new overlay
mkdir -p ros2_connextdds/src/ros2 && cd ros2_connextdds
git clone -b foxy \
    https://github.com/rticommunity/rmw_connextdds.git src/ros2/rmw_connextdds

# Configure environment for ROS (e.g. Foxy) and Connext
source /opt/ros/foxy/setup.bash
source ~/rti_connext_dds-6.0.1/resource/scripts/rtisetenv_x64Linux4gcc7.3.0.bash
export CONNEXTDDS_DIR=${NDDSHOME}

# Build RMW packages and load them into environment
colcon build --symlink-install
source ~/ros2_connextdds/install/setup.bash
```

ros-data-types library

- Collection of "standard" ROS data types in IDL format.
- Generates C++ (or C) interfaces for all types and links them into a single library.

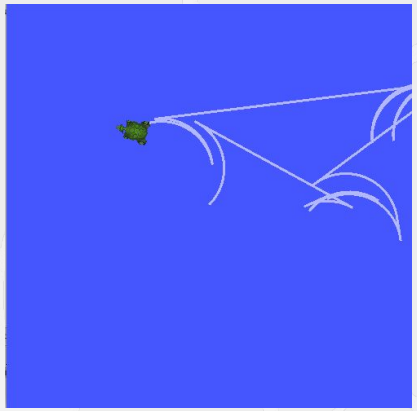
```
# Clone, build, and install ros-data-types
git clone https://github.com/rticommunity/ros-data-types.git ros-data-types

NDDSHOME=${CONNEXTDDS_DIR} cmake -Hros-data-types -Bros-data-types/build \
    -DCMAKE_INSTALL_PREFIX=ros-data-types/install -DLANG=C

cmake --build ros-data-types/build -- install

export ROS_DATA_TYPES_DIR=$(pwd)/ros-data-types
```

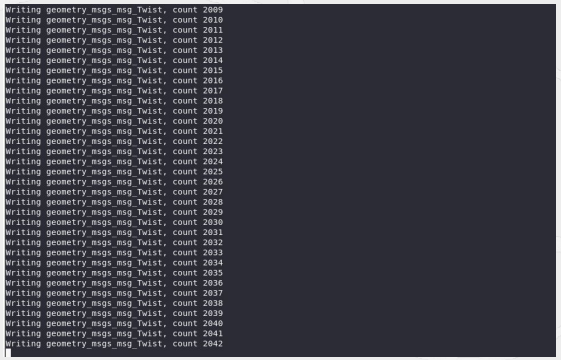
A simple turtle controller

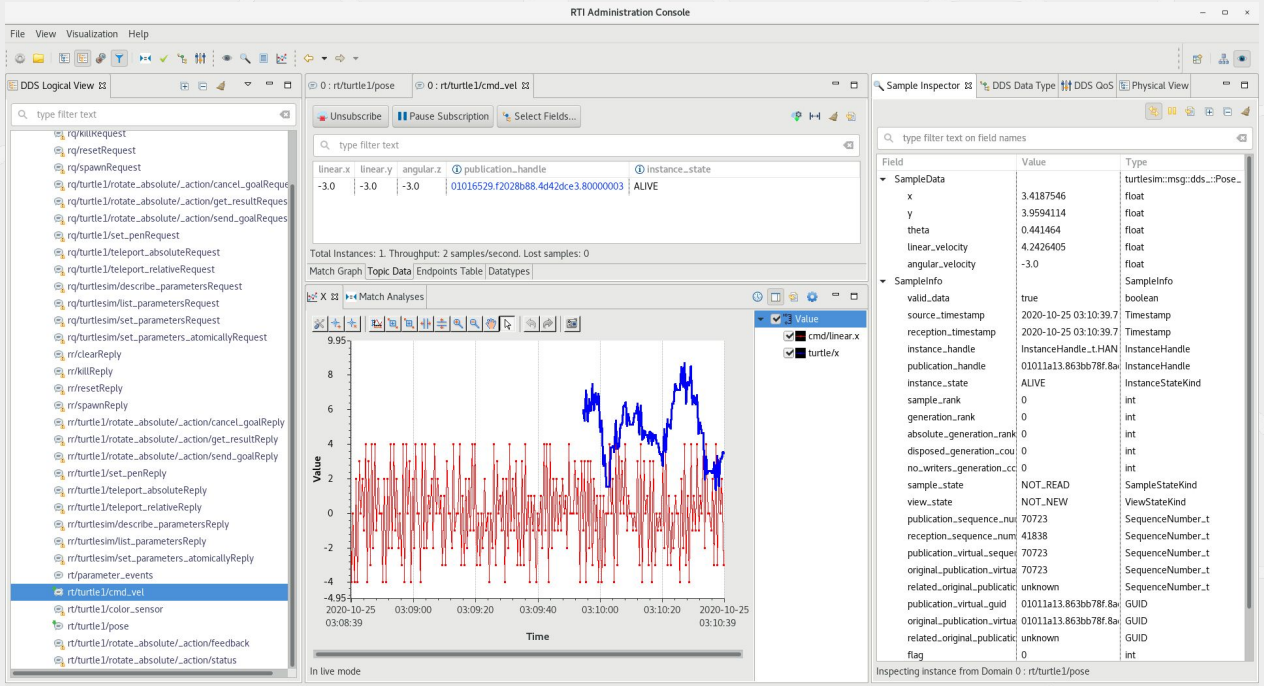
```
# Create a "workspace" directory for the example.

mkdir hello_turtle && cd hello_turtle

# Generate a publisher for geometry_msgs/msg/Twist using the IDL from
# the ros-data-types repository.

rtiddsgen -language C -example x64Linux4gcc7.3.0 -unboundedSupport \
          -d . -I ${ROS_DATA_TYPES_DIR}                            \
          ${ROS_DATA_TYPES_DIR}/geometry_msgs/msg/Twist.idl

# Edit Twist_publisher.c to subscribe to:
# - Register type with the correct name, and subscribe to correct topic.
# - Set the fields of the published sample

vim Turtle_publisher.c

# Create a CMakeLists.txt

vim CMakeLists.txt

# Build the application

cmake -H. -Bbuild && cmake --build build
```

Run and inspect

```
# Select the RMW implementation
export RMW_IMPLEMENTATION=rmw_connextpro_cpp

# Start the turtle simulator
ros2 run turtlesim turtlesim_node

# Start the publisher to control
# the turtle movements.
build/Twist_publisher

# Use rtiadminconsole to monitor
# the applications.
rtiadminconsole
```







Twist_publisher - Type and Topic

```
// Twist_publisher.c:42
#include "geometry_msgs/msg/Twist.h"
#include "geometry_msgs/msg/TwistSupport.h"

// Twist_publisher.c:118
type_name = "geometry_msgs::msg::dds_::Twist_";
retcode = geometry_msgs_msg_TwistTypeSupport_register_type(
        participant, type_name);

// Twist_publisher.c:130
topic = DDS_DomainParticipant_create_topic(
        participant, "rt/turtle1/cmd_vel",
        type_name, &DDS_TOPIC_QOS_DEFAULT,
        NULL /* listener */, DDS_STATUS_MASK_NONE);
```

Twist_publisher - Data publication

```
// Twist_publisher.c:94
struct DDS_Duration_t send_period = {0,500000000};


// Twist_publisher.c:178
double amount = (double)((rand()+1) % 5) * ((count%2)?1.0:-1.0);
instance->linear.x = amount;
instance->linear.y = amount;
instance->angular.z = (rand() % 2)? amount : 0;
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7)
project(hello_turtle C)

list(APPEND CMAKE_MODULE_PATH "$ENV{CONNEXTDDS_DIR}/resource/cmake")
set(CONNEXTDDS_DIR "$ENV{CONNEXTDDS_DIR}")
find_package(RTIConnextDDS "6.0.0" REQUIRED COMPONENTS core)

add_executable(Twist_publisher Twist_publisher.c)
target_link_libraries(Twist_publisher
   PRIVATE RTIConnextDDS::c_api $ENV{ROS_DATA_TYPES_DIR}/install/lib/libRosDataTypes.a)
target_include_directories(Twist_publisher
   PRIVATE ${CMAKE_CURRENT_SOURCE_DIR} $ENV{ROS_DATA_TYPES_DIR}/install/include)

if(CMAKE_SYSTEM_NAME MATCHES "Linux" AND CMAKE_C_COMPILER_ID MATCHES "GNU")
   set_target_properties(Twist_publisher PROPERTIES LINK_FLAGS -Wl,--no-as-needed)
endif()
```

Questions?

Thank you.

For any questions or feedback, and to request access to the GitHub repository, please write to robotics@rti.com.

Try a full version of Connext DDS for 30 days

TRY CONNEXT AT
RTI.COM/DOWNLOADS

Includes resources to get you up and running fast

Stay Connected

rti.com
Free trial of Connext DDS

rtisoftware

@rti_software

connextpodcast

@rti_software

rti.com/blog