
# R5-COP

## Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems

## Middleware Assessment Metrics (Update)

(SYN) Viatcheslav Tretyakov

| | | | |
|---|---|---|---|
| Project | R5-COP | Grant Agreement # | 621447 |
| Deliverable # | D31.12 | Dissemination Level | PU |
| Contact Person | Viatcheslav Tretyakov | Organization | SYN |
| E-Mail | vtretyakov@synapticon.com | Date | 31.01.2017 |

# Document History

| Document History | | | |
|---|---|---|---|
| **Ver.** | **Date** | **Changes** | **Author** |
| 0.1 | 20.09.2016 | Set up document frame | Rainer Buchty |
| 0.2 | Dec 2016 | collect input from D31.30/40/50 | Viatcheslav Tretyakow |
| 0.3 | Jan 2017 | Internal review | Rainer Buchty |

# Table of Contents

## List of Figures

## List of Acronyms

| | |
|---|---|
| ROS | Robot Operating System |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| TCPROS | (TCP+ROS) TCP Transport layer for ROS Messages and Services |
| UDPROS | (UDP+ROS) UDP transport layer for ROS Messages and Services |
| SSL | Secure Socket Layer |
| OMG | Object Management Group |
| RT | Real Time |
| RTPS | Real Time Publish-Subscribe |
| DDS | Data Distribution Service |
| DDSI-RTPS | Real Time Publish-Subscribe Wire Protocol DDS Interoperability |
| DDS-RPC | Remote Procedure Call over DDS |
| IDL | Interface Description Language |
| QoS | Quality of Service |
| SPI | Service Plugin Interface |

# 1 Introduction

## 1.1 Summary (abstract)

This deliverable provides a concluding summary of the WP31 activities concerning

- Middleware developments
- Assessment of ROS2.0/DDS
- Implementation and documentation of ROS bridges

These activities were greatly influenced by the Middleware Assessment Metrics (MAM) developed as part of D31.11; as such, this deliverable will summarize the above activities and illustrate how they were based and/or influenced by the MAMs.

## 1.2 Purpose of this document

According to the TA, this document serves several purposes:

- the evaluation of the developed middleware prototypes and bridges to other middlewares at the end of the project.
- The evaluation will include the results of the examination of ROS with regard to real-time, safety, transport-layer modularity aspects and comparing, optionally benchmarking it with OPC-UA.

However, with the Amendments of 2015 and 2016, this work was effectively transferred to the following deliverables:

- Examination of ROS was performed in D31.20 “ROS2.0/DDS Evaluation” and handed in in M30.
- With introduction of ROS2.0/DDS, R5-COP was adjusted to not develop a competing middleware approach. As such, only a dedicated transport-layer middleware targeting mobile operator consoles was developed and subsequenctly documented in D31.30 “Transport layer middleware” and was handed in in M24.
- Implementation and documentation of ROS bridges is part of D31.40 “Implementation and Documentation of ROS bridges” and was handed in in M32.

Hence, the purpose of this deliverable – in accordance with the TA description – is essentially a concluding summary of the aforementioned deliverables, evaluating relevance and usefulness of the metrics defined in D31.11 ”“Middleware Assessment Metrics (Initial)”’.

### 1.3 Partners involved

| Partners and Contribution | |
|---|---|
| **Short Name** | **Contribution** |
| **PIAP** | ROS2.0/DDS measurements and evaluation (D31.20) |
| | Transport-layer middleware (D31.30) |
| | ROS bridging (D31.40) |
| **SYN** | Deliverable assembly |
| **TUBS** | Support in deliverable assembly |

# 2 Middleware Assessment Metrics (MAMs)

To assess middlewares in terms of modularity, composability, configurability, reusability, real-time capability, scalability and stability define a range of specific metrics is required. The assessment shall consider the special requirements on middleware frameworks for industrial applications and embedded systems. Following a discussion on the specific topics and ranges illustrated by Figure 1, such metrics were defined in D31.11 (Cf. Section 3.1, pp.16-22). They can be summarized into four essential groups as follows:

- **Communication**
  - Node communication mechanism, i.e. how data exchange between nodes takes place (messages, topics, ports, ...)
  - Message transport protocol, i.e. via what underlying network protocol the communication is exercised (UDP, TCP, EtherCAT, ...)
  - Services used to coordinate communication between nodes (naming, lookup, and discovery service)
  - Scalability (communication distribution) of the middleware
  - Real-time capabilities
- **Programming and interoperability**
  - Host OS upon with the middleware is running
  - Programming languages supported by the middleware framework (C/C++, Python, Java, ...)
  - Message format of a middleware framework, determining interoperability between platforms and languages (XML, RDF, Java serialization, ...)

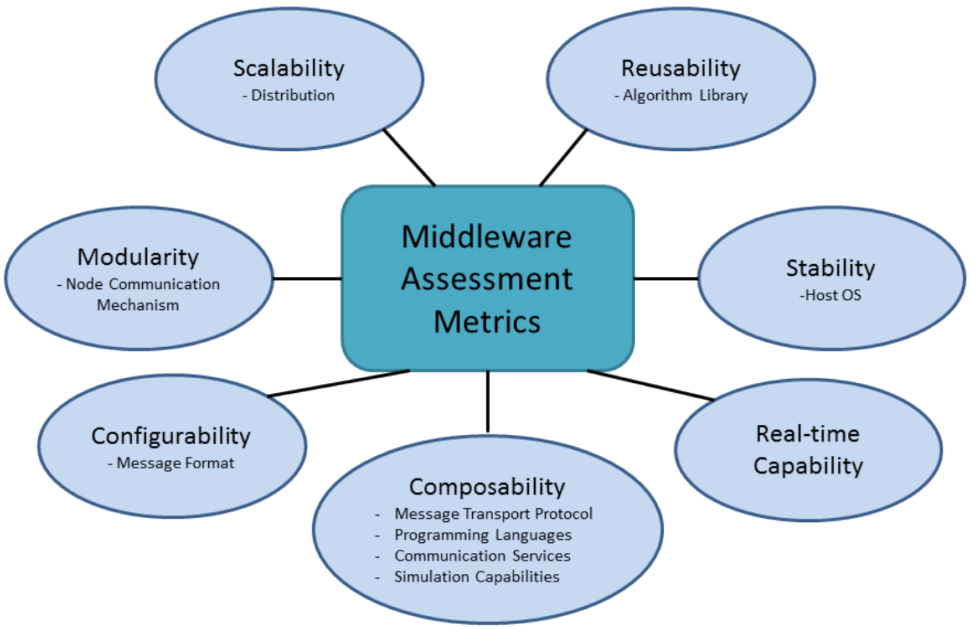


**Figure 1:** Middleware Assessment Metrics as discussed in D31.11

- **Development support**
  - Algorithm library for an existing middleware
  - Simulation capabilities permitting modeling, prototyping and simulation of the final system
- **Deployment**
  - Licensing of software components like middleware or algorithms/library

In the following, the individual work performed will be briefly summarized, and the role of the above metrics will be outlined.

## 2.1 Transport layer middleware

Documented in D31.30 *Transport Layer Middleware*, a capable communication middleware was developed. It provides handy abstractions and several different implementations for various protocols and standards, including ROS, rosbridge, ROS2 and RTRC. As such, it enables broad integration of PIAP HMI with other partner platforms, directly or through MLF. In general, support for robots form outside of PIAP is now possible. Additional benefit is that thanks to having one framework hosting several protocols, these protocols can be meaningfully compared and benchmarked. First, early results of these benchmarks are available. The architecture imposes almost no technical debt and the whole set of communication libraries can be developed further without obstacles.

The middleware has been developed as a set of libraries that provide communication abstraction as well as several implementations of it. The abstraction removes any dependency that application must otherwise have on a particular transport layer. The communication API is independent of used protocols and even of such thing as whether the protocol is connection-oriented or in a publish-subscribe model. Such characteristic allows to support a vast number of middleware protocols, such as PIAP's Real-Time Robot Control protocol (RTRC), JAUS, various ROS systems (ROS, ROS2, rosbridge package), and the R5-COP developed MLF.

This development was clearly driven by the MAMs **Programming and Interoperability**, **Development Support**, and corollary (by allowing configuration and selection of desired software components) also **Communication**. Furthermore, easy exchangeability of the underlying communication middleware allows addressing of **Deployment** by allowing to factor in also license requirements.

| MAM group | Solution provided related to MAM |
|---|---|
| Communication | Communication abstraction from underlying transport layer |
| Programming and interoperability | Removing dependencies resulting from individual transport layer |
| Development support | Provided as a set of libraries |
| Deployment | Feedback into the ROS and ROS2 communities regarding bugfixes and improvements |
| | Provision of the solution to interested parties (open-source not generally possible due to interfacing to non-open-source technologies) |

**Table 3:** MAM use within D31.30 "Transport-layer middleware"

## 2.2 ROS2.0/DDS Overview

Within R5-COP, Deliverable D31.30 *Transport Layer Middleware* highlighted shortcomings of ROS which particularly limits ROS use for security- and safety-relevant applications. This was particularly on the grounds of **Communication** shortcomings. ROS2 will address these issues either directly by the ROS API or via the DDS communication middleware. Particular identified issues were

- Multi-master issues
- Inadequacy for high-latency networks
- Blocking calls in ROS and thread-pool starvation
- No middle ground solutions and no QoS settings for publishers and subscribers
- No multicast on UDPROS level
- No persistence for latching data

ROS2 aims at directly supporting novel use cases, that particularly benefit from improving on certain ROS shortcomings. Among these are the standardization of robot-team operation, support for real-time operation, improved operation within non-ideal networks which particularly benefit from improved **Communication**.

**Deployment** is by dedication towards evolution of lab prototypes into products suitable for use in real-world applications.

Furthermore, ROS2 particularly targets **Programming and Interoperability** via prescribed patterns for building and structuring systems. With the decision to introduce a common middleware for communication concepts, mainly DDS, OS2 can leverage existing and well-developed implementations of that standard: ROS2 provides a wrapper for DDS with a simple API exposing only a small part of all DDS features. This decision was made to ensure a smooth transition

[1] `https://github.com/piappl/ros2_benchmarking`

| MAM group | Solution provided / issues addressed related to MAM |
|---|---|
| Communication | Different implementations of DDS/RTPS |
| | Types of messages |
| | General communication behavior (reliability, latency, delays, ...) |
| | Quality of Service (resistance to network disturbance, throughput, scalability) |
| | Special focus: real-time capabilities |
| Programming and interoperability | Middleware interfacing (ROS2 to underlying middleware) |
| Development support | Benchmarking suite developed |
| Deployment | Feedback into the ROS and ROS2 communities regarding bugfixes and improvements |
| | Benchmarking suite available for general use via `github`[1] |

**Table 4:** MAM use within D31.20 "ROS2.0/DDS Evaluation"

from ROS for existing user and to hide some of the DDS complexity and differences in implementations. While this is a correct approach to enable a fast adoption of ROS2, it leaves users with a subset of DDS. This directly affects the aspect of real-time capabilities and quality of service, as via the current API only limited control is possible.

With regard to **Scalability**, ROS2 alpha7 unfortunately is worse than ROS, but this is so far considered a transition/development issue to be addressed by a future version.

Details of the MAM-based ROS2/DDS assessment can be found in the M30 deliverable D31.20 *ROS2.0/DDS Evaluation*.

## 2.3 ROS bridges

Core of the bridging work were considerations regarding different communication patterns, incompatible information packaging, semantic differences in message types, and in general a differing set of standard messages and presence/absence of QoS support which is particularly relevant for real-time and secure communication. As such, the focus was particularly on the MAMs **Communication** and **Programming and Interoperability**.

The efforts are documented in D31.40 *Implementation and Documentation of ROS Bridges* (cf. Section 2). As part of this work, a number of bridging solutions were implemented and evaluated, i.e. interfacing solutions ROS1 with ROS2, JAUS, the partner-developed Real-Time Robot Control (RTRC) protocol, the Universal Robots control platform, and the R5-COP developed abstraction layer Modular Link Framework.

Also for this work, the MAMs defined through D31.11 have proven relevant and sufficient.

| MAM group | Solution provided / issues addressed related to MAM |
|---|---|
| Communication | Communication patterns |
| | Incompatible information packaging |
| | Lack of support for same-communication QoS settings |
| | Semantic differences in message types |
| | Different sets of standard messages |
| Programming and interoperability | Providing abstraction layer leveraging the communication issues identified |
| Development support | Providing individual bridging solutions |
| Deployment | Feedback into the ROS and ROS2 communities regarding bugfixes and improvements, particularly regarding ROSbridge suite and the ROS1 bridge for ROS2 |
| | Provision of ROS-JAUS bridge via `github`[2] |
| | Interfacing to the Modular Link Controller of the Modular Link Framework (cf. MLF deployment) |

**Table 5:** MAM use within D31.40 "Implementation and Documentation of ROS Bridges"

[2] https://github.com/piappl/ros-jaus-bridge

# 3 Conclusion

The Middleware Assessment Metrics (MAM) presented in D31.11 covered all relevant fields from communication and technology to programming/abstraction and also deployment/licensing. Through the course of WP31 development and comparing with the course of the externally developed ROS2/DDS (to which R5-COP provided sufficient input, particularly via the efforts of partner PIAP) as well as the project-internal work on ROS bridging (mainly pursued by partner PIAP together with partner SMR) it showed both relevance and practical orientation of the proposed metrics.

As such, the MAMs presented in D31.11 proved highly valuable for guiding and assessing the work performed in the middleware-oriented WP31.