A Communication Architecture Based on ROS2 for the Control in Collaborative and Intelligent Automation Systems

Sasanka Chaudhry[1], Mr. Navneet Ballabh Gautam[2]
[1]Department of Management, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha
[2]Department of Management, Sanskriti University, Mathura, Uttar Pradesh
Email -[1]sasankachoudhury@soa.ac.in, [2]navneet.mba@sanskriti.edu.in

ABSTRACT

In modern industry, collaborative robots are becoming a part of smart automation systems. Development and operation of these systems are distinct from conventional methods of automation and therefore contribute to new challenges. Fortunately, the Robot Operating System (ROS) offers a communication channel and a vast array of tools and utilities capable of supporting that development. However, the use of ROS in large-scale automation systems is difficult due to communication problems in a distributed environment, hence the development of ROS2. This is critical in modern industry as it allows rigorous integration into collaborative and intelligent automation systems of the state-of - the-art tools and algorithms needed for control. As these large-scale automation systems are typically distributed in nature, they require a well-structured and stable infrastructure for communication. This paper therefore presents variations of a communication architecture based on ROS2 with these characteristics. A communication architecture based on ROS2 is described in this paper along with an industrial use-case of a collaborative and intelligent automation system.

Keywords
Collaborative Robots, Communication Architecture, Robot Programming, Multi-Master, Intelligent Automation Robot Operating System.

Article Received: 10 August 2020, Revised: 25 October 2020, Accepted: 18 November 2020

Introduction

The ever-increasing demand for product consistency and quality, and the need for shorter production and delivery times, poses new production challenges. Using autonomous and collaborative robots should overcome some of those problems. Various platforms have emerged as middle-ware solutions to ease the integration and creation of collaborative and intelligent systems where the Robot Operating System (ROS) stands out.

Since Willow Garage incarnated ROS in 2007[3], there has been a substantial rise in the number of ROS users with over 16 million cumulative downloads in the year 2018[4]. ROS2[5] is currently being developed, where the communication layer is based on Data Distribution Service (DDS)[6] to allow distributed control architectures on a large scale. This development paves the way for the use of ROS2-based architectures in real-world industrial automation systems, as discussed in this paper [1], [2].

Since ROS2 is behind ROS when it comes to the number of packages and active developers, contact bridges between ROS and ROS2 are used to transfer messages. These bridges allow the community to use the strengths of both ROS and ROS2 in the same program, i.e. to have a comprehensive collection of packages built and at the same time provide a reliable way of interacting between machines. This is critical in modern industry as it allows rigorous integration into collaborative and intelligent automation systems of the state-of - the-art tools and algorithms needed for control. As these large-scale automation systems are typically distributed in nature, they require a well-structured and stable infrastructure for communication. This paper therefore presents variations of a communication architecture based on ROS2 with these characteristics [3]–[5].

Although allowing integration and communication is of tremendous benefit, it is just one part of the challenge. The architecture of overall control also needs to plan and organize the activities of robots, humans and other tools as well as keep track of everything. Mixed human-machine industrial environments involving non-traditional management methods where activities are performed either collaboratively, coactively or independently. ROSPlan, SkiROS, eTaSL / eTC and CoSTAR are just a few ROS-based frameworks that seek to manage these control strategies. However, these frameworks concentrate primarily on single robot systems, and as such lack the infrastructure to support large-scale automation systems.

Intelligent and interactive systems also involve several robots, computers, smart devices, human-machine interfaces, cameras, safety sensors, etc.

A use-case in this paper shows the difficulties of achieving scalable automation in an industrial set-up and highlights the need to use ROS due to the broad integration of smart devices and algorithms. Section 2 offers a brief description of the communication layer at ROS and ROS2. A summary of the use-case is given in Section 3, and section 5 describes the design of the communication architecture. Sequence Planner, a state-of - the-art task planner and a separate controller is explained shortly in Section 4 and the paper is finalized [6]–[8].

The Robot Operating System (Ros) And Ros2

ROS is a series of tools and libraries that assists in the creation of robotics software. The program is written in the form of nodes that exchange messages based on standard TCP / IP sockets using a transport layer called TCPROS. ROS has a centralized network configuration which means a running ROS master has to exist. The master is responsible

for naming and registration services so that other ROS nodes on the network may locate each other and connect in a peer-to-peer fashion. It is not robust to have a configuration in which all nodes rely on a central ROS master, particularly if nodes are distributed on many computers within a network.

To improve this design limitation, ROS2 developers implemented the standard DDS of the Object Management Group as the communication middleware which considers it to be scalable, reliable and well-proven in mission-critical systems. DDS thus allows ROS to be applied in industrial use cases [9]–[11].

A variety of DDS implementations are available, allowing ROS2 users to choose an implementation from a vendor that fits their needs. This is achieved by means of an ROS Middleware Interface (RMW), which also reveals the policies of quality of service. ROS2 users have the ability to choose between QoS policy options for supported RMW implementation which enables them to change the transfer layer further.

There are implementations that provide a more direct access to the wire transfer protocol for the real-time publish-subscribe (RTPS). We still have ample functionality for ROS2 while not meeting the full DDS API. An example implementation is FastRTPS from eProsima, and it is the selection implementation for the setup mentioned in this paper. Default QoS settings are maintained which define reliability, volatile durability and history of KEEP LAST. An extensive but out - of-date study compares ROS middleware results. Unlike the analysis we found FastRTPS was working without problems.

Because ROS2 uses DDS to exchange messages, the Discovery Module of the underlying RTPS protocol allows ROS2 nodes to locate each other automatically on the network, so no need for a ROS2 master.

Setting up a stable ROS multi-master system before ROS2 was introduced was rather cumbersome, and it often resulted in using other communication platforms such as Kafka or ZeroMQ for exchanging messages between different ROS systems. Several specialized ROS packages that enable communication in multiple ROS master systems already exist, e.g., multimaster_fkie, rocon, ZeroMQ-ROS and Kafka-ROS Bridge.

Currently, there are still cases where there is a need to use old ROS packages when designing new systems. In distributed cases, it is possible to use the communication layer capabilities of ROS2 together with local ROS systems. Utilizing ROS2 and several ROS masters enables us to partition a ROS system into local, single computer hubs, where a hub includes a ROS master and one or multiple local nodes. This is be exemplified with an industrial use-case and described in more detail.

An Industrial Use-Case Of Collaborative And Intelligent Automation

Benefits of utilizing both ROS and ROS2 are evident in a collaborative robot assembly station in a Volvo Trucks engine manufacturing facility located in Skövde, Sweden [20]. The aim of this use-case is to present possibilities in designing a workstation where both humans and robots work in a collaborative or coactive fashion. Particularly in this use-case, a collaborative robot and a human operator should perform assembly operations on a diesel engine, Fig. 1.

Fig. 1: A Collaborative Robot Assembly Station

The physical setup of this assembly station consists of a six Degree of Freedom (DoF) collaborative robot, an autonomous mobile platform, two different specialized end-effectors, a smart tool that can be used as an end- effector, a docking station for the end-effectors, a lifting system and docking station for the smart tool, a camera and RFID reader system and eight computers dedicated for different tasks. The collaborative assembly process is briefly explained in the next part of this section.

An Automated Guided Vehicle (AGV) carrying a diesel truck engine and an autonomous mobile platform (MiR100) carrying kitted material to be assembled on an engine enter the Collaborative Robot Assembly Station. A ladder frame, three oil filters and several oil transport pipes are to be mounted on the engine in this station. Before the collaborative execution of these operations can begin, an authorized operator has to be verified with a RFID reader. After verification, the operator is greeted by the system and instructions for tasks that the human should perform are shown on a screen. If no operator is verified, some operations can still be executed independently by the robot, however, violation of safety zones around the robot trigger a safeguard stop.

In the case of collaborative assembly, a dedicated camera system keeps track of the human assuring safe coexistence. During positioning of the AGV and the MiR100 in the assembly station, the Universal Robots (UR10) robot attaches to a special end-effector needed for manipulation ladder frame manipulation. Since the ladder frame is quite heavy, the robot and the human collaborate in transporting it from the kitted MiR100 to the engine.

After placing the ladder frame on the engine, the operator informs the control system with a button press on a smart watch, after which the UR10 leaves the current end-effector and attaches itself to a smart tool used for

Fastening bolts. During this tool change and as it was instructed on the instructions screen, the operator is placing twelve pairs of bolts needed for assembling the ladder frame on the engine and indicates completion of this task through the smart watch. Now, the control system knows that bolts are in place and the UR10 starts the tightening operation with the smart fastener tool. Since tightening of these bolts

takes some time, the operator can simultaneously mount three oil filters on the engine. If the robot finishes the tightening operations first, it leaves the smart tool in a floating position above the engine and waits for the operator in an idle home position.
The operator uses the smart watch again to let the system know that the oil filters are in place. This event makes the robot attach to the third end-effector and start performing the oil filter tightening operations. During the same time, the operator attaches two oil transport pipes on the engine, and uses the same smart tool that the robot has left floating to tighten plates that hold the pipes to the engine. After executing these operations, the AGV with the assembled engine and the empty MiR100 leave the Collaborative Robot Assembly Station.

Sequence Planner As A Discrete Event Controller

Sequence Planner (SP) is a tool for modelling, analysing and control of automation systems. In SP, automation systems are modelled using operations and variables. It includes algorithms for a variety of use cases related to modelling, for instance synthesizing control logic, verification and visualizing complex operation sequences in different projections. SP uses a hierarchical control approach based on planning of low level abilities.
Abilities are operating directly on control state which is estimated by applying transformation pipelines to different ROS2 topics. Because ROS messages are strictly typed, pipelines can be generated automatically to populate a SP model with variables that correspond to fields in the message types used. More complex pipelines can also be used, for instance for aggregating messages into a single state variable in SP or for applying some discretization function to reduce the state space of the controller.
Modelling in SP is done by composing abilities (modelled independently of each other) using specifications that disable undesired interactions between them. By disabling a minimum of undesired behaviour, a lot of flexibility is left for on-line planning algorithms to find the currently most suitable desired behaviour. This is done by employing synthesis based on supervisory control theory. On top of abilities, desired behaviour is modelled using operations. Operations are matched to sequences of underlying abilities based on on-line planning. This allows for a clear separation of two control layers, i.e. operations do not need to know about underlying detailed abilities.
Planning is continuously performed in a receding horizon fashion, allowing the control to be intelligent about which abilities to execute based on current state of the system. Fig. 2(a) shows SP during simultaneous execution of four operations (in green) where live resource usage is visualized at the bottom. In order to assure transparent and robust message transfer between Sequence Planner and the rest of the system, a well-organized communication architecture has to be designed.
Hub vs. Node type oriented communication architecture:
There are several ways to organize nodes, hubs, messages and topics in a distributed ROS system. Designing the communication architecture in the described collaborative robot assembly station was approached with considerations of possible future expansions. One possible expansion of the system would be the inclusion of other robots from other vendors, thus nodes and hubs are designed with the ability to handle equipment agnostic messages. This leads to a primarily node type oriented communication architecture since families of handler nodes could be copied on different hubs to handle different equipment while receiving messages of same type.
In a system where hubs differ in structure, it is valid to pursue a more hub oriented communication architecture. This means that each hub is looked at as a unique set of nodes that are communicating with the rest of the system. This implies that, in order to structure the communication in a large multi-hub system where hubs are structurally different, hub specific message types have to be defined.
In the hub oriented communication architecture scenario, two cases can be defined. In the first case, a pair of command-state topics per hub is communicating via a set of hub specific distributor-collector nodes, Fig. 3(a). The distributor nodes subscribe to the intended command topic, disassemble the message and publish matching message fragments to all nodes in the local hub. Nodes of a hub send their state to the collector node which assembles the total state of the hub in a state message and publishes it on the state topic of that hub. In this case, all the traffic to and from the hub is bridged with a single pair of one-directional static bridges.
In the second case, there are no gateway nodes that distribute or collect messages in a hub, Fig. 3(b). This means that all nodes of a hub subscribe either to the same command topic for that hub and evaluate only a part of the message intended for them, or subscribe to node dedicated topics. Without the state collector node in a hub, each node publishes its state on a separate topic. These two cases shouldn't be considered as mutually exclusive and should be used together according to complexities of nodes, hubs and the overall system.
In a multi-hub system where hubs contain nodes that are similar to nodes from other hubs, a node oriented communication architecture can be approached. This means that similar nodes from all hubs subscribe to a single topic and evaluate the message according to some flags that specifiy the hub. In this scenario, the state can be collected in one of the ways shown in the hub oriented scenario, or by having a dedicated node that collects states from node groups from all hubs and pass it on.
In the described use-case, the communication architecture was dominantly node type oriented without the aid of a state collector hub. This means that the stream of state messages from each node occupied dedicated topics, which made the number of topics needed for communication with SP quite big. Structure was maintained utilizing adequate namespaces for each topic group.

Conclusion

In the beginning phases of a project, particularly for a large and distributed automation system, it is not always straightforward to come up with a good solution on how to design the system architecture. This paper addressed the use of ROS and ROS2 in intelligent automation systems and suggested ways of structuring the architecture of communication. A defined use-case integrated the

mentioned proposals in the paper and demonstrated possibilities in a collaborative robot assembly station for achieving scalable automation.

Some of the key advantages of designing a system in the mentioned manner are the freedom to choose between DDS implementations and QoS policy choices, an industrial standard communication protocol, the independence of a particular operating system and ROS delivery, scalability due to a self-contained hubs architecture and a large ROS and ROS2 software library containing tools and algorithm Future research will include the further development and attempt to make stateless nodes more general and reusable.

References

[1] G. Lilis, G. Conus, N. Asadi, and M. Kayal, "Towards the next generation of intelligent building: An assessment study of current automation and future IoT based systems with a proposal for transitional design," *Sustain. Cities Soc.*, 2017.

[2] R. R. Hoffman, M. Johnson, J. M. Bradshaw, and A. Underbrink, "Trust in automation," *IEEE Intell. Syst.*, 2013.

[3] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *IEEE Transactions on Industrial Informatics*. 2011.

[4] J. Figueiredo and J. Martins, "Energy Production System Management - Renewable energy power supply integration with Building Automation System," *Energy Convers. Manag.*, 2010.

[5] L. M. Bergasa, E. Cabello, R. Arroyo, E. Romera, and Á. Serrano, "Human factors," in *Intelligent Vehicles: Enabling Technologies and Future Developments*, 2017.

[6] *Machine Learning for Cyber Physical Systems*. 2017.

[7] N. Higgins, V. Vyatkin, N. K. C. Nair, and K. Schwarz, "Distributed power system automation with IEC 61850, IEC 61499, and intelligent control," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, 2011.

[8] Y. Bestaoui Sebbane, "Control," in *Intelligent Systems, Control and Automation: Science and Engineering*, 2012.

[9] J. Bhatt and H. K. Verma, "Design and development of wired building automation systems," *Energy Build.*, 2015.

[10] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis, "Evolution of autonomous and semi-autonomous robotic surgical systems: A review of the literature," *International Journal of Medical Robotics and Computer Assisted Surgery*. 2011.

[11] T. Yang, D. Clements-Croome, and M. Marson, "Building Energy Management Systems," in *Encyclopedia of Sustainable Technologies*, 2017.