A data transmission method between the ROS-based robot and the industrial controller

Jun Zou¹,a, Zhuoxian Chen²*, Cheng Zhou¹ and Zhigang Li¹

¹ School of Mechanical Engineering, Nanjing University of Science & Technology, Nanjing, Jiangsu Province, 210094, China
²Angao Intelligent Security Technology Company Limited, Suzhou, Jiangsu Province, 215000, China

aEmail: zoujun1996@njust.edu.cn
*Email: kangta_cy@hotmail.com

Abstract. Aiming at the problems of low code reusing rate and inability to share communication content among ROS Nodes when a ROS-based robot communicates with the industrial controller, a data transmission method based on Modbus protocol and ROS service is proposed. In this method, the program responsible for communicating with the industrial controller and the program executing the robot task are encapsulated in different ROS Nodes, using the cross-process function call of ROS Service communication to realize data sharing among multiple Nodes, thereby reducing the coupling of the system and improving the code reusing rate. Finally, the feasibility of the method is verified by a robot mechanical protection system, which provides a method for the communication between the ROS-based robot and the industrial controller.

1. Introduction

Robot Operating System (ROS) is a framework for software development of robot applications[1]. With the advantages of powerful functions, multilingual support, distributed design and open-source, ROS has gradually become one of the most popular robot development platforms and is widely used in driverless cars, wheeled and humanoid robots, industrial manipulator, etc[2-3]. When developing robot applications based on ROS, PC is usually used as the master computer, the functions of the control system are realized by ROS programming. The PC is connected to the robot through the underlying interface and sends instructions to control the robot’s actions. In some application scenarios, sensors need to be installed to expand the functions of the robot. Since most sensors do not have an interface to communicate with the PC, it is necessary to send the output signals of sensors to the industrial controller first, and then establish a communication connection between the controller and ROS. The usual practice is to write data acquisition programs in all ROS Nodes that require the signals, resulting in low code utilization. In order to optimize the communication programs between ROS Nodes and controller, and realize that the communication content is shared by multiple Nodes, a data transmission method based on Modbus protocol and ROS Service is proposed, and it is applied to a robot mechanical protection system.
2. ROS communication mechanism and Modbus protocol

2.1. ROS communication mechanism
ROS is a distributed process framework in which executable programs are called Nodes, and each Node is a software module that sends and receives messages[4]. Therefore, the core technology of ROS lies in its distributed communication mechanism—the ROS Master helps different Nodes to find each other and establish connections[5]. The most basically and frequently used communication mechanisms in ROS are Topic and Service.

Compared with Topic, Service is a communication mechanism with feedback, and reduces the RPC (remote procedure call) between two communicating Nodes, so it has better real-time performance and is very suitable for processing small data volume, containing logic processing and requiring communication completed in a short time[6]. Service is based on the Server/Client model, where Client is the requester of the Service, and Server is the provider of the Service. The communication process is shown in figure 1, which is mainly includes the following steps:(1)Node 1 starts, Server registers the name and message types (contains both request and reply message type) of the Service with ROS Master, and provides a callback function to process the request of Service; (2)Node 2 starts, Client registers the name of Service to be queried with ROS master; (3)ROS master queries according to the registration information of the Client. If a matching Server is found, it sends the address information to the Client; If not found, wait for the corresponding Server registration; (4)After receiving the address information of Server, the Client establishes a connection with the Server, calls the callback function and passes in the request data; (5)After receiving the request, the Server executes the Service through the callback function, and returns the Client reply data after execution.

Before using the Service, the user needs to define the request and reply message types (corresponding to the args and return values types of the callback function), which can be provided by ROS or user-defined. After compiling and generating the code and class definition related to the Service, the user can use the Service.

![Figure 1. Service communication mechanism based on Server/Client model.](image)

2.2. Modbus protocol
Several communication protocols commonly used in industry include Modbus, BACnet, OPC, etc. Among them, Modbus has become a global industrial communication standard and is currently the most commonly used communication methods between industrial control devices[7]. Modbus has its own message structure, which can be recognized and used by controllers no matter what kind of network is used for communication. Modbus is an application layer message transmission protocol, including three communication types: ASCII, RTU, and TCP. Modbus RTU and Modbus ASCII use traditional RS232 or RS485 serial ports to transmit data, while Modbus TCP is developed based on Ethernet and TCP/IP technology, connects and communicates via Ethernet or optical fiber[8].

3. Data transmission method based on Modbus protocol and ROS Service
In some scenarios, ROS needs to communicate with the industrial controller to obtain sensor output signals stored in, which are used to adjust the robot's subsequent actions. If there are multiple running
Nodes in ROS that require the signals, the direct approach is to write programs in each Node to communicate with the controller, which leads to code redundancy and is not conducive to maintenance. In order to share the communication content between ROS and the industrial controller among multiple Nodes, this paper proposes a data transmission method combining Modbus protocol and ROS service. The process is shown in figure 2, which includes two processes: (1) Data transmission between ROS Node and the industrial controller; (2) Data transmission among ROS Nodes.

![Data transmission model based on Modbus protocol and ROS Service.](image)

3.1. Data transmission between ROS Node and the industrial controller
In ROS, “Data I/O” Node is designed for communicating with the industrial controller, and the communication process is based on Modbus protocol. After establishing the connection between PC and industrial controller through Ethernet or serial port, write a communication program in “Data I/O” Node according to Modbus protocol to read the sensor’ output signals from the industrial controller.

3.2. Data transmission among ROS Nodes
After reading sensor’ output signals from the controller, “Data I/O” Node sends it to all task Nodes that need the information. This process is completed based on the ROS Service communication mechanism, the specific steps are as follows: (1) Define the request message type of the Service according to the output of the industrial controller; (2) Define the Client of the Service in “Data I/O” Node, define the Server of the Service in each task Node and provide callback functions; (3) When a task Node starts, the Server in the Node is registered, then the Client immediately calls the callback function in the Server and passes in the sensor output signals as parameters; (4) Perform the necessary logical processing on the parameters in the callback function, and then assign them to the variables in this Node, so that the sensor’ output signals is passed from the “Data I/O” Node to the task Node through a cross-process function call; (5) If the Task node has data that needs to be sent to the controller during execution, it can be passed to the Client through the return value of the function, then the “Data I/O” Node write it to the controller. As can be seen from the above, “Data I/O” Node acts as a transfer station in the whole process, which communicates with the industrial controller and sends the communication content to each task Node.
4. Application case

As shown in figure 3, the six-axis robot on an experiment platform is developed based on ROS and uses Python as the main programming language. The end-effector of the robot has two types: gripper and sucker, which realized the sorting and transporting of materials. The robot is not equipped with a safety fence, if someone approaches the robot while it is working, there will be a danger of collision. Therefore, the safety scanner is selected to construct the mechanical protection system to monitor risky areas around the robot. When people enter the dangerous area, it sends out warning messages in time or makes the robot perform measures such as deceleration and shutdown. The safety scanner can monitor three plane areas at the same time and outputs the corresponding signal after being triggered. According to the distance between the human and the robot, the range of human activities is divided into three parts: warning zone 2, warning zone 1, and danger zone, as shown in figure 4. When people enter warning zone 2, the robot decelerates to 50% of the working speed; When people enter warning zone 1, the robot decelerates to 25% of the working speed; When people enter the dangerous area, the robot stops working.

The safety controller is used in the system to read, store and calculate the output signals of the safety scanner, and control the opening or closing of the gas-valve on gripper or sucker.

When the robot is working, it needs to communicate with the safety controller at all times to obtain the trigger status of the safety scanner. In addition, when the robot picking and placing materials, it is also necessary to send a signal to the safety controller to open or close the gas-valve on the end-effector. Therefore, in ROS project, “Data_IO” Node is designed for communicating with the safety controller, and the communication process is based on ModBus TCP protocol. First, import the pymodbus library into the Node program, and bind the IP address and port number of the safety controller through the ModbusTcpClient() method to establish a communication connection. Then, according to the address allocation of the communication fields between “Data_IO” Node the and the security controller (as shown in table 1 and table 2), call the library function to read scanner output signals from the safety controller, or write signals to it to control the gas-valve. Finally, configure the "Data_IO" node into the launch file of the ROS project, so that it starts to work after the ROS project is started.

Table 1. Address allocation of received fields for communication between ROS and the safety controller.

| Discrete input (read only) | Variable name (switching value) | Contents note (1: not triggered; 0: triggered) |
|---------------------------|---------------------------------|-----------------------------------------------|
| 8192                      | scanner_danger_zone             | state of scanner danger zone                  |
| 8193                      | scanner_warning_zone1           | state of scanner warning zone 1               |
| 8194                      | scanner_warning_zone2           | state of scanner warning zone 2               |

Figure 3. Experiment platform and the six-axis robot.

Figure 4. Schematic diagram of the robot mechanical protection system.
Table 2. Address allocation of sent fields for communication between ROS and the safety controller.

| Coil   | Variable name (switching value) | Contents note               |
|--------|---------------------------------|----------------------------|
| 4      | gripper_status                  | state of gas-valve on the gripper |
| 5      | sucker_status                   | state of gas-valve on the sucker |

In order to allow different task Nodes to use the same communication content, the ROS Service communication mechanism is used. Since the communication data are all switching values (0 or 1), define the request and response message types of the Service file as bool arrays, as shown in figure 5. Then, define the Client of the Service in the "Date_IO" Node, define the Server of the Service in each task Node and provide a callback function for processing requests. The following takes the "Robot_sort" Node that performs the sorting task as an example. After the "Robot_sort" Node starts, the "Date_IO" Node requests Service and calls the callback function in the "Robot_sort" Node, passing in the scanner output signals as parameters. Assign the parameters to global variables in callback function, then you can use them in the robot sorting program to judge whether the detection zone of the scanner is triggered, so as to decide whether the robot should slow down or stop. Meanwhile, the "Robot_sort" Node sends the signal used to control the gas-valve to the "Date_IO" Node through the return value of the function, then the "Date_IO" Node writes it to the safety controller. When the "Robot_transport" Node starts, the communication process is similar to the above. In order to ensure the timely update of data, the frequency of communication between the "Date_IO" Node and the safety controller, and the frequency of Service invocation are both set to 50 Hz. The entire communication process is shown in figure 6.

![Definition of the ROS Service file](image)

![Schematic diagram of system data transmission](image)
The real-time status of the mechanical protection system is shown in the figure 7, field tests show that no matter what tasks the robot is performing, if people enter the warning zones, the robot will decelerate as expected, and if people enter the danger zone, the robot will stop working. This indicates that the above data transmission method can realize the real-time communication between the ROS Node and the safety controller, and the communication content can be shared by multiple ROS Nodes.

Figure 7. Real-time status of the mechanical protection system.

5. Conclusion
In order to solve the problem that the communication content cannot be shared by multiple ROS Nodes when ROS-based robot communicates with the industrial controller, this paper proposes a data transmission method based on the Modbus protocol and ROS Service and gives a detailed description. This method can realize the sharing of communication content among ROS Nodes through cross-process function calls, thereby improving the code reusing rate. Finally, the method is applied to a robot mechanical protection system, and the predetermined functions of the system are realized, which proves the feasibility of the proposed method.

References
[1] Crick C, Jay G, Osentoski S, et al. (2012) ROS and Rosbridge: Roboticists out of the loop.In: Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction. New York. pp. 493-494.
[2] Zhang J.W., Zhang L.W., Hu Y., et al. (2012) Open Source Robot Operating System—ROS. Science Press. Beijing.
[3] Wang Y., Wang B.Q., Guan Y., et al. (2021) Differential Fuzz Testing of Robot Operating System. Journal of Software, 32(06): 1867-1881.
[4] Li M.N., Shao Z.Z., Qu Y., et al. (2018) Reliability design of interactive communication between Android and ROS. Computer Applications and Software, 35(03): 106-113.
[5] Morgan Q, Brian G, William D.S. (2015) Programming Robots with ROS. O'Reilly Media, Inc., Sebastopol.
[6] Hu C.X. (2018) Development Practice of ROS Robots. China Machine Press, Beijing.
[7] Zuo W., Chen Y.X. (2013) Principle and Security Analysis of Modbus Protocol. Communications Technology, 46(12): 66-69.
[8] Shang W.L., Zhang S.S., Wan M. (2014) Modbus TCP Communication Anomaly Detection Algorithm Based on PSO-SVM. Acta Electronica Sinica, 42(11): 2314-2320.