International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2021, 11(4): 505–520, doi: 10.21655/ijsi.1673-7288.00258


International Journal of Software and Informatics

Research Article

# Modeling and Analysis of Data Flow-Oriented ROS2 Data Distribution Service

Qian Lu (芦倩)[1,2], Xiaojuan Li (李晓娟)[1,2], Yong Guan (关永)[1,3], Rui Wang (王瑞)[1,4], Zhiping Shi (施智平)[1,4]

[1] (College of Information Engineering, Capital Normal University, Beijing 100048, China)
[2] (Beijing Engineering Research Center of High Reliable Embedded System (Capital Normal University), Beijing 100048, China)
[3] (Beijing Advanced Innovation Center for Imaging Theory and Technology (Capital Normal University), Beijing 100048, China)
[4] (Beijing Key Laboratory of Light Industrial Robot and Safety Verification (Capital Normal University), Beijing 100048, China)
Corresponding author: Xiaojuan Li, lixj@cnu.edu.cn

**Abstract** Robot Operating System (ROS) is an open-source, meta-operating system, which can provide a structured communication layer based on the message mechanism on heterogeneous computing clusters. To improve the unsatisfactory real-time performance and reliability of data distribution in ROS1, ROS2 is proposed with a data flow-oriented data distribution service mechanism. This study validates the real-time performance and reliability of the ROS2 data distribution mechanism by means of probabilistic model checking. Firstly, a formal validation framework is put forward for the data flow-oriented ROS2 data distribution service system, and the probabilistic timed automata model is set up for the communication system module. Secondly, the probabilistic model checker PRISM is used to verify the real-time performance and reliability of data flow-oriented ROS2 data distribution service through the analysis of data loss rate and system response time. Finally, depending on the retransmission mechanism and Quality of Service (QoS) policy analysis, different data requirements and quantitative performance analysis of the transmission mode are achieved through the setting and adjustment of QoS parameters. This study can provide references for ROS2 application designers and the formal modeling, validation, and quantitative performance analysis of the distributed data distribution service based on the data flow.

**Keywords** ROS2; data distribution service; QoS; probabilistic timed automata; PRISM; formal modeling and analysis

**Citation** Lu Q, Li XJ, Guan Y, Wang R, Shi ZP. Modeling and analysis of data flow-oriented ROS2 data distribution service, *International Journal of Software and Informatics*, 2021, 11(4): 505–520. http://www.ijsi.org/1673-7288/258.htm

---

This is the English version of the Chinese article “面向数据流的 ROS2 数据分发服务形式建模与分析. 软件学报, 2021, 32(6): 1818–1829. doi: 10.13328/j.cnki.jos.006251”.
Funding items: National Key Research and Development Program of China (2019YFB1309900); National Natural Science Foundation of China (61876111); Capacity Building for Sci-Tech Innovation-Fundamental Scientific Research Funds (00620530290073); Research Fund from Academy for Multidisciplinary Studies of Capital Normal University (0062155087)
Received 2020-08-30; Revised 2020-10-26; Accepted 2020-12-19; IJSI published online 2021-12-23

Robot Operating System (ROS) is an open-source, meta-operating system, which has been widely used in the research and development of robot software. With the increasing popularity of robotics-related technologies, the requirements for robot application are increasing accordingly. Thus, ROS should be analyzed in order to timely make corresponding responses to the external signals and other information. However, there are two major problems in ROS, namely high communication delay and low reliability. The performance of ROS is limited by its high-latency serialization methods and socket communication of TCP/UDP, which can cause multiple memory replications during message transmission[1]. Therefore, the real-time performance of ROS is not satisfactory, which cannot be directly applied to the real-time control and mission-critical applications. In addition, the ROS1 communication system is based on TCPROS/UDPROS and strongly relies on the handling of the master node. Once the master node breaks down, the reliability of the system will be greatly affected, which limits the practical application of ROS to a certain extent.

On the basis of ROS, the ROS2 is improved and adopts many new technologies, thereby bringing the reformation of the overall architecture. To improve the real-time performance and reliability of ROS1, ROS2 adopts data flow-oriented Data Distribution Service (DDS) as its communication mechanism. DDS, proposed by the Object Management Group, is a new generation of the data-centric data specification of distributed systems, which allows the use of the publish/subscribe mechanism for reliable and real-time data collection and delivery and supports the dynamic discovery of nodes, the topic-based data distribution, and the space-time decoupling of the data flow. Therefore, this data specification guarantees the real-time performance and reliability of data distribution and has been widely used in distributed real-time systems due to its flexible configuration method and its scalability. Compared with other publish/subscribe middleware, DDS has a main feature of possessing abundant QoS supports[2]. QoS policy provides guarantee for data transmission[3]. According to these QoS parameters, system designers can build distributed applications in line with specific requirements and usability. Compared with the message-based data distribution mechanism with the master node as the center, the data flow-oriented ROS2 DDS mechanism can well meet the real-time requirement of data communication between distributed nodes and improve the efficiency of data transmission. Therefore, ROS2 is good at handling huge and complex data. By controlling the QoS parameters, modules having different requirements for the update rate, reliability, and bandwidth control can be well integrated into the system. However, as the internal data interaction of the system becomes frequent and the scale of the system expands, the number of publishers and subscribers in the system, the data types, and the QoS demands also increase. Consequently, the system performance will drop, and different applications have different requirements for the performance of DDS. These problems will bring severe tests to the data distribution and cause hidden dangers to the design. Therefore, they should be analyzed in the early stage of system development. However, the system model cannot be completely verified by traditional testing and simulation methods. Formal methods use mathematical and logical methods to describe and verify the system. Compared with traditional verification methods, formal verification[4] performs auxiliary design and verification with tools based on formal methods and is of great help to improve the credibility of the system. Moreover, it can verify all possible situations specified and described, thus effectively overcoming the shortcomings of the simulation verification. Probabilistic model checking integrates probabilistic analysis and general model checking technology, which is suitable for the verification of the non-deterministic systems and can realize quantitative analysis. Therefore, the probabilistic model checking is utilized in this paper to analyze the correctness of the system, and the parametric analysis of the system performance is further made on the basis of a formal model, so as to provide valuable

references for system designers and ROS program developers.

Scholars have conducted research on the formal verification of DDS. He *et al*.[5] took the lead in the formal modeling of the publish/subscribe system based on Probabilistic Timed Automata (PTA). Liu *et al*.[6] validated the vitality of DDS on ROS2. Yin *et al*.[7] used the Communication Sequential Process (CSP) to model the components of multiple modules in the Real-Time Publish/Subscribe (RTPS) protocol and verified the non-divergence, confirmation mechanism, data consistency, and other attributes by using the model checking tool PAT. All the above studies carried out formal modeling of the data distribution mechanism, but most of them focus on the message-oriented data distribution mechanism, without the formal description of the QoS policy and the system performance analysis. QoS controls the communication mechanism between all aspects and the bottom layer and is a transmission control policy used to solve delay and blocking. When critical and distributed systems are running, the reliability needs to be guaranteed and the required performance should be satisfied. The QoS parameters can be adjusted to meet the data application requirements in different scenarios, which have a vital impact on the real-time performance and reliability of the system. In the research on the analysis of QoS, Maruyama *et al*.[8] clarified the data transmission performance of ROS1 and ROS2 in various situations. The performance of ROS1 and ROS2 is compared in terms of delay, throughput, number of threads, and memory consumption. The QoS adaptive method is an effective method for the development of real-time systems based on publish/subscribe middleware, which is of growing importance to develop the large-scale real-time distributed systems in a dynamic environment. When plenty of data is transmitted among many heterogeneous entities, strict constraints of QoS policy are required. Inglés-Romero *et al*.[9] proposed a method that can safely, automatically, and transparently adjust QoS attributes in middleware based on DDS, so that the system can operate with the best performance with the available resources. On this basis, Casini *et al*.[10] proposed a new method that can automatically adjust the QoS policy in real-time systems in dynamic environments to achieve the adaptive control of QoS. The performance of real-time systems was improved and became more stable by the reduction of the computing power. Although all the above studies described and analyzed the QoS, most of them adopted the methods of testing and simulation. It is less reported to analyze the performance of the ROS2 communication system through the quantification of QoS parameters from the perspective of formal validation.

In this paper, the formal modeling of the data flow-oriented ROS2 data distribution mechanism is conducted, in which the data flow form is expressed by a data sequence composed of multiple data blocks and the relevant QoS policies DEADLINE, RELIABLITY, DURABILITY, and HISTORY in ROS2 are formally described. The PTA model is established, and the communication characteristics of data flow, data retransmission mechanism, confirmation mechanism, and the uncertainty of environmental factors are considered in the model. Finally, the quantitative analysis of the system performance is carried out through the adjustment of QoS parameters, and the reliability and real-time performance of the ROS2 communication system is verified. This study can provide valuable references for system designers and ROS program developers and make ROS2 safer and more reliable.

Section 1 describes the ROS2 data distribution mechanism and constructs an abstract model of the ROS2 communication system based on DDS. Section 2 divides the abstract model of the communication system into sub-modules and builds a PTA model for each sub-module. Section 3 verifies the real-time performance and reliability of the ROS2 communication system with the probabilistic model checker PRISM and analyzes the critical attributes of the robot communication process. In light of the communication characteristics between ROS2 systems, the quantitative analysis is carried out by the introduction of QoS policies, and different data requirements and transmission modes can be realized through the setting and adjustment of the

QoS parameters. In addition, the quality of communication can be guaranteed in a specific communication environment. Section 4 presents the conclusions and prospects, summarizes the main work and innovations of this paper, and makes suggestions for further study.

## 1 System Description

The ROS1 communication system is based on TCPROS and UDPROS, and this communication mode depends on the master node. Once the master node breaks down, the entire communication system will be affected. Relying on DDS, the ROS2 communication system[11] provides the implementation of the DDS abstraction layer inside the system. Users do not need to know the API of DDS and the ROS2 is allowed to adopt the advanced configuration options to optimize the use of DDS. In addition, the ROS2 communication system does not need a master node and thus has truly become a decentralized distributed system, which makes ROS2 have stronger fault-tolerant ability than ROS1. ROS2 can be built on Linux, Windows, Mac, and real-time operating systems. The ROS2 framework is shown in Figure 1.

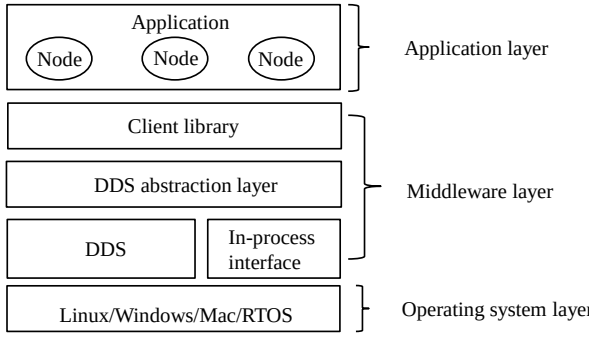


**Figure 1** ROS2 system framework

DDS is composed of Data-Centric Publish/Subscribe (DCPS) layer and Data Local Reconstruction Layer (DLRL)[12, 13]. DLRL is the upper layer of the DDS specification, which enables the distributed data to be shared by local and remote objects. DCPS is the core and bottom layer of the DDS specification, which is responsible for the data transmission and the guarantee of QoS control. DCPS can efficiently deliver data from publishers to subscribers, thus ensuring the reliability of data transmission. DDS is a data flow-oriented communication paradigm. Data-centric communication can add various parameters on the basis of each data, including publishing rate, subscription rate, data valid time, and many other parameters. These QoS parameters allow system designers to build distributed applications in line with the requirement and usability of each specific data. The critical abstract concept of DDS is Global Data Space (GDS). The DDS specification requires that the GDS in the communication model should be implemented in a fully distributed form, which allows applications that need to obtain data in the GDS to dynamically join or leave the system at any time. When an application on a single node breaks down, it will not cause the entire system to crash[14]. Therefore, the loose coupling of the system is ensured in time and space, and the flexibility and the scalability of the system are also greatly improved.

In the DDS communication model, each DDS entity has a set of QoS policies. Through the control of the QoS policies, the performance of data transmission can be dynamically adjusted to meet the diverse requirements of the system for data transmission. Figure 2 shows the QoS policy of the ROS2 communication system, and there are four QoS policies in ROS2[8]:

(1) DEADLINE: This QoS policy requires data writers and data readers to update the data

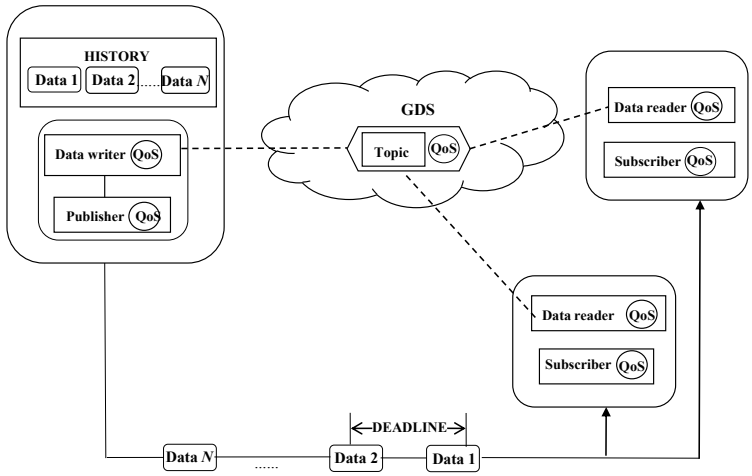


**Figure 2** Quality of service (QoS) policy for the ROS2 communication system

once within each deadline.

(2) HISTORY: This QoS policy controls whether data transmission transmits only the latest value, all intermediate values, or the values within the foregoing two situations. The depth of stored samples can be set by the queue depth setting.

(3) RELIABILITY: There are two DDS communication modes of reliability: Best_Effort and Reliable. The Best_Effort communication mode requires data to be sent as quickly as possible, but data loss may be caused. In the Reliable communication mode, the lost data can be retransmitted. In other words, the data transmission is guaranteed.

(4) DURABILITY: This QoS policy specifies whether the publisher saves unreceived data for nodes that have not joined. The DURABILITY policy includes two attributes: Transient_local and Volatile. The Transient_local attribute stipulates that the sender saves unreceived data for nodes that have not joined, while the Volatile attribute stipulates that the publisher does not deliberately save the sample data.

In the ROS2 communication system based on DDS, different QoS policies can be added for quantitative analysis. Through the setting and adjustment of QoS parameters, different data requirements and transmission modes can be achieved, and the quality of communication can be guaranteed in a specific communication environment.

In ROS2, the DDS publish/subscribe model is mainly composed of the publisher, subscriber, data writer, data reader, and GDS, which can realize the publishing and subscription of topic and the data distribution[15]. It can be divided into publishing module, subscription module, GDS module, and communication channel module.

- Publishing module: The publisher must first create the data writer $DW_i$, and $DW_i$ publishes the topic $T_i$ of the data to be published and the available QoS to the GDS. If the publishing succeeds, $DW_i$ will enter a blocking (waiting) state. At the same time, the publishing record of $DW_i$ is synchronously included in the publishing topic table of each node in the same DDS domain. When there is subscription information successfully matched, the $DW_i$ waiting state is activated, and the value of the *Durability* attribute in the QoS corresponding to $T_i$ in the publishing topic table is checked. If its value is 1, data needs to be added to the publication buffer so that subsequent subscribers can receive the data successfully. Otherwise, the data will be deleted after being sent to the subscriber.
- Subscription module: The subscriber needs to first create a data reader $DR_j$. $DR_j$ queries

the matching publisher in the publishing topic table in its GDS and sends the subscription topic $T_j$ and QoS to the GDS of the publisher. Subsequently, $DR_j$ enters the blocking state. When receiving the subscription message of $DR_j$, the publisher needs to add its subscription record to the subscription topic table of the GDS and query the value of the *Durability* corresponding to $T_j$ in the publishing topic table. If the value is 0, the data is not stored into the publication buffer and thus the subscription fails. If the value is 1, the data information that meets the subscription conditions is stored in the publication buffer and then $DW_i$ is activated to transmit all qualified data to subscribers according to the corresponding QoS policy.

- GDS module: The GDS module includes the publishing topic table, subscription topic table, subscription failure table, and publication buffer. The publishing topic table saves the publishing topic information of all nodes in the same DDS domain. The subscription topic table saves the subscription information for subscribing to the nodes. The connection between the data writer in the publishing module and the data reader in the subscription module is realized through the topic, and the connection between publication and subscription is completed by the link of the name, data type, and QoS related to the data itself. The subscription failure table saves the historical records of subscription failures of a specific node, and the publication buffer saves the data which is published by the publisher and needs to be stored for a long time. The DDS publish/subscribe model describes the data resource status that the publisher can provide and the expectation degree of the subscriber on the data resource with QoS parameters. According to these parameters, the DDS middleware selects the transmission mode that best meets the QoS requirements of two communication parties to distribute data, which not only realizes the real-time performance of data transmission but also increases the flexibility of communication. The real-time data distribution can be achieved with the the data flow as the trigger point and guidance of the entire platform.
- Communication channel module: The communication channel is responsible for the data transmission function between the publisher and the subscriber. In the communication channel, data is transmitted in the form of data flow, which can be regarded as a set of ordered data sequences with a beginning and an end. Because the communication channel is unreliable, data loss may occur in data transmission. The performance of data transmission can be dynamically adjusted through the selection of a certain QoS policy, so that different communication requirements can be met.

The publish/subscribe model of the data flow-oriented ROS2 DDS is shown in Figure 3.

In the communication system of the data flow-oriented ROS2 DDS, one or more DDS domains can be deployed. Domain participants in the same domain publish or subscribe to data by matching the topics and the QoS policies. In the DDS communication system, the publisher does not actually transmit data, but creates and manages the data writer $DW_i$. $DW_i$ is responsible for publishing the topic and sending the data to the subscriber who subscribes to the topic and has consistent QoS. The subscriber subscribes to data by creating the data reader $DR_j$. In the ROS2 communication system, QoS includes DEADLINE, RELIABLITY, DURABILITY, and HISTORY policies. When the topic published by $DW_i$ is consistent with the topic subscribed by $DR_j$ and the QoS policy is consistent, a data publish–subscribe relationship is formed. If multiple data writers $DW_i$ satisfy the subscription topic, one of them that meets the corresponding QoS policy is chosen to be subscribed to. When $DW_i$ fails, the data of other data writers can be subscribed to. Therefore, the ROS2 communication system solves the problem of the single point of failure.

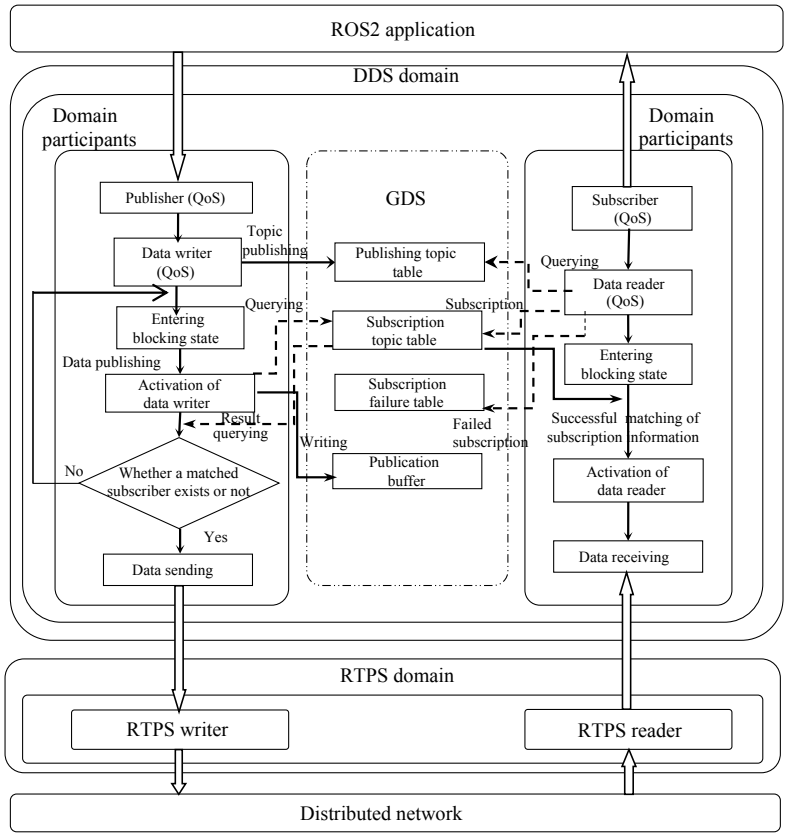


**Figure 3** DDS publish/subscribe model

## 2 System Modeling Based on PRISM

This paper uses PTA to formally model the data flow-oriented ROS2 DDS. PTA is the extension of finite automata with respect to time and probability, which represents time and probability simultaneously. Therefore, it is suitable to express the loss probability and time delay in the model. According to the system description in the previous section, this section specifically introduces the formal representation of the main models involved in the system and implements the PTA model of each model in the probabilistic model checker PRISM.

### 2.1 Probabilistic model checker PRISM

PTA[16] is a probabilistic extension of timed automata, and provide a formal framework and mechanism for modeling the real-time systems in a probabilistic environment. The probabilistic model checker PRISM[17, 18] is a tool developed by the University of Oxford to detect whether a probabilistic model satisfies the given temporal logic properties. It has been applied to analyze systems from many different fields, including communication and multimedia protocols, randomized distributed algorithms, security protocols, biological systems, and many other systems. PRISM can construct and analyze several types of probabilistic models: Discrete-Time Markov Chains (DTMC), Continuous-Time Markov Chains (CTMC), Markov Decision Processes (MDP), Probabilistic Automata (PA), and PTA. PRISM first analyzes the description of the model and then constructs the internal representation of the probabilistic model. Afterward,

it computes the reachable state space of the model and discards all unreachable states, which means that all possible combined sets may appear in the modeling system. After that, the logic specification is analyzed, and the model is verified by an appropriate model checking algorithm through inductive syntax.

## 2.2 Implementation of systems in PRISM

The communication system of the ROS2 DDS is abstracted according to the system description, and the system is formally modeled as the PTA model of the data publishing node, the PTA model of the data subscription node, the PTA model of GDS, and the PTA model of the communication channel module. For the construction of a communication system model for ROS2 DDS, the following needs to be considered: The transmission channel is unreliable, namely that data may be lost during channel transmission, and the data packet loss rate of the channel is $P$; the time delay in data transmission cannot be ignored; the order in which the data is received is consistent with the order in which it is sent; in case of buffer overflow, the new data information will be directly discarded.

### 2.2.1 *PTA model of the data publishing node and the data subscription node*

Nodes can be data publishers or data subscribers in DDS. During the modeling, the nodes are divided into data publishing nodes and data subscription nodes, whose PTA models are built respectively. Figure 4 shows the PTA model of publishing nodes communicating in the form of data flow. As the communication mechanism of ROS2, DDS distributes and transmits data through data flow. A data flow is a data sequence of bytes with an order, a beginning, and an end. In the formal model, the data flow is regarded as a data sequence composed of multiple data blocks, and a flag bit of the Boolean type is used to identify each data block. *fp* indicates whether the currently sent data block is the initial data block; *lp* indicates whether the currently sent data block is the final data block; *flag* is used to determine whether the data block is successfully sent. The initial value of *flag* is false. When the data block is successfully sent to the subscriber and the confirmation message is received from the subscriber, the value of the flag bit *flag* is replaced with true, which means that the current data block is successfully transmitted. If the data blocks are continuously sent and the publisher does not receive the confirmation message from the subscriber within a period of time *timeout*, namely that the *flag* value remains false, then the sending of the continuous data blocks is interrupted. The publisher first resets the clock variable $t$ to 0 and retransmits the lost data blocks. The number of retransmissions of each data block is bounded, and the maximum retransmission number *count* is MAX. After retransmission, the continuous transmission resumes. The state space of the PTA model of publishing node contains eight states: $S = \{Idle, Pub\_datablock, Waitack, Retransmit, Datablock\_Succ, Success, Fail, Waitsync\}$. When no data is sent, the initial state of the publisher is the *Idle* state. When there is data to be sent, the data flow of each communication is divided into multiple data blocks, and the initialization $k$ indicates that the currently sent data block is the $k$-th data block in the entire data flow. Meanwhile, *count* is initialized as the maximum number of retransmissions, and $N$ represents the number of data blocks in the entire data flow. At this time, the state of the publisher shifts from *Idle* to *Pub_datablock* and continuously sends the subsequent data blocks. After each data block is sent out, the publisher needs to wait the confirmation message from the subscriber. If the confirmation message *flag* = true is received within the time *timeout*, the state is shifted to *Datablock_Succ* , which means that the data block is successfully transmitted. In the *Datablock_Succ* state, whether the currently transmitted data block is the final data block of the entire data flow needs to be determined. If the current data block is the final data block, the state of the publisher shifts to *Success*, which means that the entire data flow is

successfully transmitted. Then the publisher returns to the *Idle* state. Otherwise, it returns to the *Pub_datablock* state, and then sends the next data block. When one of the data blocks is sent out and the confirmation message is not received from the subscriber within the time *timeout*, it will enter the *Retransmit* state. Within the maximum number of retransmissions, the current data block can be allowed to be retransmited until the confirmation message is received from the subscriber. When the number of retransmissions is exhausted but the final data block has not been sent or the final data block has been sent but no confirmation message is received from the subscriber, the state will transfer to the *Fail* state, indicating that the data flow transmission fails. When the *Fail* state is present, the publisher enters the *Waitsync* state, initializes the synchronization signal *sync_signal* to be false, and waits for the synchronization with the subscriber. When the *sync_signal* is true, the publisher state is synchronized with the subscriber state and the publisher returns to the *Idle* state.

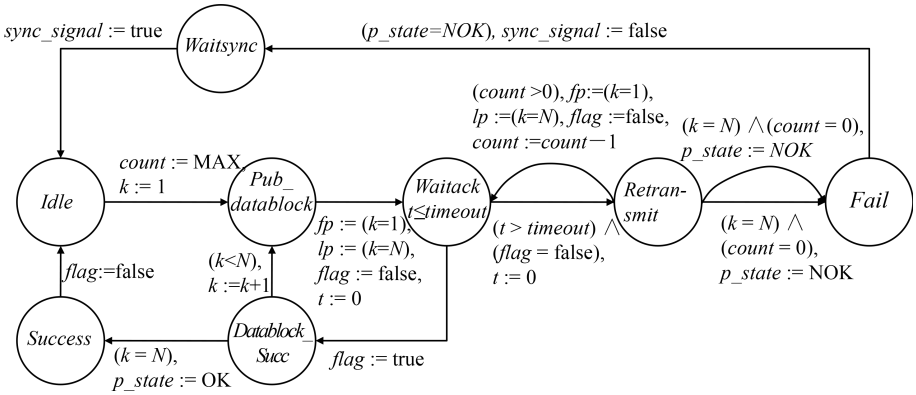


**Figure 4** Probabilistic Timed Automata (PTA) model of data publishing node

Figure 5 shows the PTA model of data subscription node.

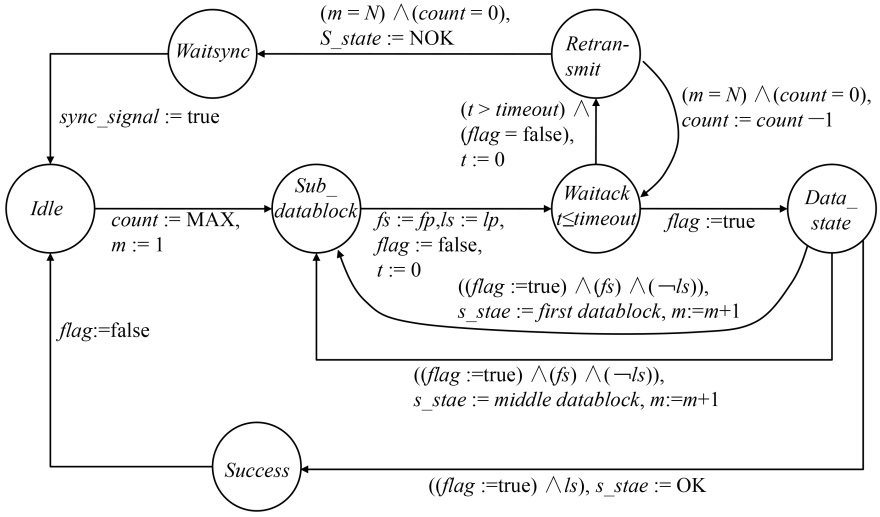


**Figure 5** PTA model of data subscription node

The state space of the model for data subscription node contains seven states: $S = \{Idle, Sub\_datablock, Waitack, Retrans\_wait, WaitSync, Success, Data\_state\}$. The subscriber

matches the data information to be subscribed with that in the publishing table of the publisher. If the matching is successful, the corresponding publisher transmits the published data to the subscriber. When no data is subscribed, the initial state of the subscriber is the *Idle* state. Once a new data block arrives, $m$ is initialized to indicate that the current data block to be received is the $m$-th data block in the entire data flow, and *count* is initialized as the maximum number of retransmissions. At this moment, the state of the subscriber shifts from *Idle* to *Sub_datablock*. In the *Sub_datablock* state, the subscriber is waiting to receive data blocks. After each data block is received successfully, the subscriber inverts the flag bit *flag* to return the confirmation message to the publisher. If the subscriber successfully receives the data block within the time *timeout*, the *Waitack* state is shifted to the data flow transmission state *Data_state*. In the *Data_state* state, whether the currently received data block is the final data block of the subscribed data flow needs to be determined: If the current data block is the final data block, the entire data flow is subscribed successfully and the subscriber returns to the *Idle* state; otherwise, the subscriber returns to *Sub_datablock* and waits to receive the next data block. If the subscriber does not receive the data block within the time *timeout*, it shifts from the *Waitack* state to the *Retrans_wait* state and waits for the publisher to retransmit the data block that is not currently received. When the number of retransmissions is not 0, the subscriber should wait the data block lost before subscribing. When the publisher has exhausted the number of retransmissions and the subscriber still does not receive some data blocks, the subscription fails and the state transfers to the *WaitSync* state. In this state, the subscriber waits for the synchronization signal. When the synchronization signal *sync_signal* is received, the state of the subscriber returns to *Idle*.

### 2.2.2 *PTA model of GDS*

The node scheduling and matching process of the publish/subscribe model is specifically implemented in the GDS module, which is mainly composed of publishing topic table *P_table*, subscription topic table *S_table*, subscription failure table *S_fail_table*, publication buffer *data_pool*, and matching flag *pair* variable. In the GDS model, the shortest time for processing published data information is *TPL* and the longest time is *TPH*; the shortest time for processing subscribed data information is *TSL* and the longest time is *TSH*. Figure 6 shows the PTA model of GDS, the state space of which contains five states:

$$S = \{Idle, PUB, P\_match, S\_match, SUB\}$$

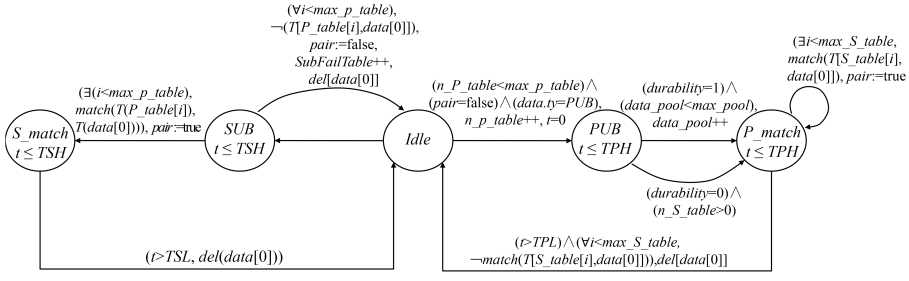


**Figure 6** PTA model of GDS

When the buffer of the publishing node in the GDS is not empty, we take out the first data information of the buffer and judge its type. If it is publishing information, whether the publishing topic table *P_table* of the publishing node is full or not should be judged. If not, the topic information of the data to be published and the corresponding QoS parameter information

are added to the publishing table of the node. The publisher changes the publishing topic table and synchronizes the publish information into the publishing topic table of each node in the same DDS domain to maintain the global consistency of the publishing topic table. At this moment, the state of GDS transfers to the *PUB*. In this state, the publishing topic information and the value of the corresponding *durability* attribute should be determined: If *durability* $= 1$, the data to be published will be saved in the *data_pool* so that the newly added nodes can subscribe later. If *durability* $= 0$, the published data does not need to be saved. In the GDS, when the state transfers to the *P_match*, the node to publish the data is matched with the subscription node in the subscription table: If the topic information and the corresponding QoS parameter information of the node to publish the data are consistent with those of the subscription node in the subscription table, the matching is successful, and the publisher sends the data to be published to the corresponding subscriber. If the matching fails, the publisher will return to the *Idle* state. During this process, *match* is a Boolean function, whose parameters are the subscribed data information in the subscription table and the published data information. If the two match, the system selects the nodes that need to publish or subscribe to data information for the communication according to the return value of the function and the QoS policies of the matched nodes. When the matching is successful, true will be assigned to the variable *pair*.

Regarding the subscription information, whether the subscription table of the subscriber is full or not needs to be determined: If the subscription table is full, the subscription message is discarded according to the overflow processing strategy; if it is not full, the subscription topic and the corresponding QoS parameter information are added to the subscription table corresponding to the subscription node. At this moment, the state of GDS transfers to the *SUB*. In this state, the subscriber wanting to subscribe needs to be matched with the publisher in the publishing topic table. If there exists a publisher whose topic information and corresponding QoS parameter information are consistent with those of the subscriber, the matching is successful. If multiple publishers meet the conditions, one will be selected as the subscription object according to a certain strategy. If there is no publisher whose topic information and QoS parameters are consistent with those of the subscriber, the subscription fails and the failure record is added to the subscription failure table *Sub_fail_table*.

#### 2.2.3 *PTA model of communication channel*

The communication channel module is mainly used for data transmission between publishers and subscribers. In the process of data flow transmission, because the channel is unreliable, data may be lost and the data packet loss rate is $p[i]$. Under this probability, the data will be retransmitted. The probability of successful data transmission is $1 - p[i]$. The variable $N$ is the number of data blocks in the entire data flow; the variable $k$ indicates that the currently transmitted data block by the data publisher is the $k$-th data block in the data flow; the variable $m$ indicates the currently received data block by the subscriber is the $m$-th data block in the data flow. Figure 7 shows the PTA model of the communication channel module. The state space of the communication channel model contains four states: $S = \{Idle, Transmit, Success, Fail\}$. When the matching between publishing and subscription nodes is successfully, the data publisher will send the data to the subscriber with the consistent topic information and QoS parameters. When the number of data blocks sent by the publisher does not reach $N$, this indicates that there still exist data blocks not sent yet. Thus the communication channel shifts its state from *Idle* to *Transmit* and sends the current data block. If the subscriber receives the data block sent from the publisher and the publisher receives the confirmation message *flag* = true from the subscriber within the clock variable *timeout*, the transmission of the current data block is successful. Then, the communication channel shifts its state to *Idle* and waits for the transmission of the next data block. Otherwise, the data transmission fails and the state of the communication channel shifts

to *Fail*. After the resetting of the clock variable, the communication channel returns to the *Idle* state and waits for the retransmission of the data block.

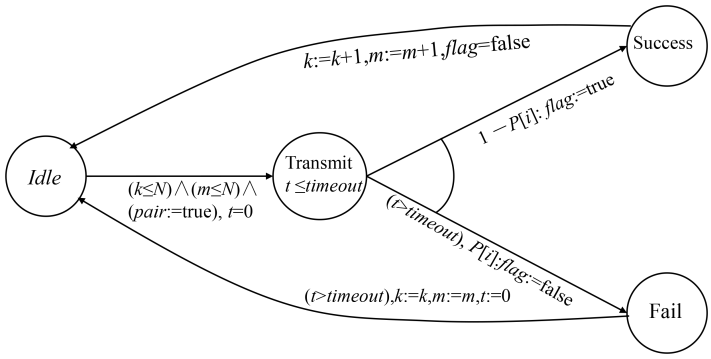


**Figure 7** PTA model of communication channel

## 3 Attribute Abstraction and Formal Expression

To verify the correctness of the system function, this paper abstracts three critical attributes. Their abstract representations are as follows:

(1) Security

For the data flow-oriented ROS2 communication system, if the system enters a deadlock state at a moment, the system cannot operate normally. In PRISM, the attribute verifying that the system is deadlock-free can be expressed as "*init*"=> $P >= 1$[true & ! "*deadlock*"].

(2) Reliability

Different applications have different requirements for DDS performance. DDS provides two transmission modes to meet the requirements of different applications, i.e., Best-Effort and Reliable. The data transmission modes can be selected through the configuration of QoS attributes. For DDS, the Reliable transmission mode should complete the distribution of the data with the specified size under different conditions.

The attribute $R =?[C <= t]$ is the cumulative reward formula in PRISM, which mainly calculates the cumulative size of data successfully transmitted by the publish/subscribe mechanism within the time $t$. In the experiment, the attribute of the successfully transmitted data size within the time $t$ is calculated to verify the data transmission situation when the QoS is reliable transmission. Two parameters $N$ and $P$ are used in the model to represent the transmitted data size and the data packet loss rate of the channel respectively (Figure 8). In the experiment, the number $N$ of data blocks in the entire data flow is set to 20. At different data packet loss rates $P$, the different expected reward values $R$ for completing the data transmission are verified. The experimental results show that when the QoS is in the Reliable transmission mode, as the data packet loss rate changes, all the curves are similar and the data transmission of the data flow with the specified size can be finally completed, namely that the expected reward value $R$ and the number $N$ of data blocks are both equal to 20, which verifies the reliability of data transmission of the DDS when the QoS is in the Reliable transmission mode. This indicates that due to the good data retransmission mechanism of DDS, the success rate of sending data will not be affected by the data packet loss rate in the Reliable transmission mode. Therefore, the system has a high reliability in the Reliable transmission mode.

The data flow transmission in the Best-Effort transmission mode is verified and presented in Figure 9. The probability attribute of successful data publishing and subscription is expressed

as:

$$\begin{cases} label\ \text{“}PubSub\_Success\text{”} = (p\_state = 1)\&(s\_state = 1) \\ P = ?[F\ \text{“}PubSub\_Success\text{”}] \end{cases}$$

where $p\_state = 1$ indicates that the publisher successfully publishes the data, and $s\_state = 1$ indicates that the subscriber successfully subscribes to the data. In the experiment, $M$ represents the size of the data flow, and different data packet loss rates $p$ are set. The experimental results show that when the data packet loss rate of the channel is constant, the probability of successful data transmission of the data flow in the Best-Effort transmission mode decreases with the increase in the number of bytes. When the size of data flow is constant, the probability of successful transmission of the data flow decreases with the increase in data packet loss rate, namely that the reliability of the system decreases.

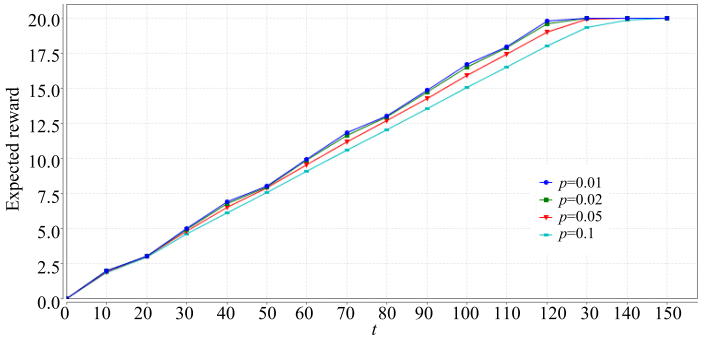


**Figure 8** System reliability in Reliable communication mode

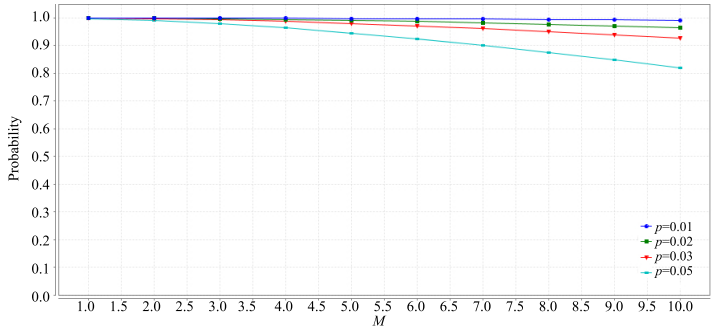


**Figure 9** System reliability in Best-Effort communication mode

(3) Real-time performance

In the calculation of the system uptime, the reward structure in PRISM can be used to calculate the time required to complete the data communication between the publisher and the subscriber. The structure of *rewards* is as below:

```
rewards ''time''

[data_transmit] true: responsetime;

[retrans] true: responsetime;

endrewards
```

The attribute $R\{\text{“time”}\} =?[F\text{“PubSub\_Success”}]$ is employed to analyze and verify the system response time of the two DDS transmission modes under different data packet loss rates. In the experiment, a constant *response time* is rewarded for each process of data flow communication between the data publisher and the data subscriber. When the communication is completed, the reward value is counted to mark the response time required by the node communication. In the experiment, the data flow size $M$ is set to 10. The data packet loss rates under the Reliable1 and Best-Effort1 policies are set to 0.05 while those under the Reliable2 and Best-Effort2 policies are set to 0.1. As shown in Figure 10, the response time of the two DDS policies in different transmission modes is verified in the experiment. We find that when the data packet loss rate is constant, due to its good retransmission confirmation mechanism the Reliable transmission mode has a longer system response time than the Best-Effort transmission mode. When the data packet loss rates are different, under the same data transmission policy, the greater data packet loss rate will lead to the longer system response time.

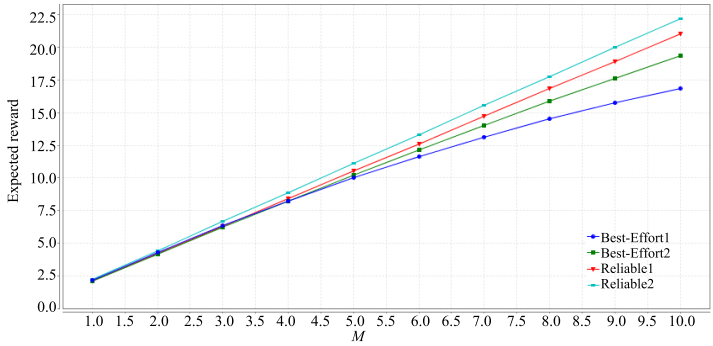


**Figure 10** Influence of QoS policy on system response time

In the data transmission of the DDS, each data transmission process and entity objects such as data writers, data readers, publishers, and subscribers are equipped with corresponding QoS policies so that different communication requirements can be met. In terms of system reliability and real-time performance, different nodes of ROS2 adopt appropriate QoS policies during the communication. Their communication services can choose Reliable or Best-Effect transmission services. However, if some specific network environments have relevant requirements for the performance of data distribution, the related QoS policies can be customized. Through the above attribute abstraction and experimental verification, the QoS policy can be changed accordingly for different service requirements and network environments. When the network environment is relatively poor, namely that the data packet loss rate in the above experiments is large, the RELIABLITY attribute in the QoS policy can be adjusted to increase the number of retransmissions in the data flow transmission of the system and realize higher stability of the system. In the same network environment, to ensure the real-time data transmission of the system (i.e., reduce the response time of the above experiments), one can select the Best-Effort QoS policy, namely that the system requires the delivery and processing of data as fast as possible. To this end, the corresponding parameters in the data transmission process, such as the size of data blocks and the number of retransmissions, need to be adjusted so as to achieve a high real-time performance as much as possible.

## 4 Conclusions

This paper proposes a formal verification framework for the abstract model of the ROS2 communication system based on DDS. The probabilistic model checker PRISM is applied to

analyze the data loss rate and the system response time to verify the real-time performance and reliability of the ROS2 communication system. Different data requirements and transmission modes can be realized by the introduction of the retransmission mechanism and the quantitative analysis of QoS policy. The formal modeling framework in this paper provides a reference for the formal modeling, verification, and quantitative performance analysis of data flow-oriented DDS. On the basis of the framework proposed in this paper, hierarchical modeling and performance analysis will be conducted focusing on the DDS of the communications between distributed nodes in different application fields, so as to provide a reference to design application systems with high performance.

## References

[1] Jiang Z, Gong Y, Zhai J, *et al*. Message passing optimization in robot operating system. Int'l Journal of Parallel Programming, 2019, 48(1): 119–136.

[2] Pardo-Castellote G, Farabaugh B, Warren R. An introduction to DDS and data-centric communications. RTI, 2005.

[3] García-Valls M, Domínguez-Poblete J, Touahria IE. Using DDS middleware in distributed partitioned systems. ACM SIGBED Review, 2018, 14(4): 14–20.

[4] Dong W, Wang J, Qi ZC. Model checking for concurrent and real-time systems. Ji Suan Ji Yan Jiu Yu Fa Zhan/Computer Research and Development, 2001, 38(6): 698–705.

[5] He F, Baresi L, Ghezzi C, *et al*. Formal analysis of publish-subscribe systems by probabilistic timed automata. Proc. of the Int'l Conf. on Formal Techniques for Networked and Distributed Systems. Berlin, Heidelberg: Springer, 2007. 247–262.

[6] Liu Y, Guan Y, Li X, *et al*. Formal analysis and verification of DDS in ROS2. Proc. of the 2018 16th ACM/IEEE Int'l Conf. on Formal Methods and Models for System Design (MEMOCODE). IEEE, 2018. 1–5.

[7] Yin J, Zhu H, Fei Y, *et al*. Formalization and verification of RTPS StatefulWriter module using CSP. Proc. of the 31st Int'l Conf. on Software Engineering and Knowledge Engineering. 2019.

[8] Maruyama Y, Kato S, Azumi T. Exploring the performance of ROS2. Proc. of the 13th Int'l Conf. on Embedded Software. ACM, 2016. 5.

[9] Inglés-Romero JF, Romero-Garcés A, Vicente-Chicote C, *et al*. A model-driven approach to enable adaptive QoS in DDS-based middleware. IEEE Trans. on Emerging Topics in Computational Intelligence, 2017, 1(3): 176–187.

[10] Casini D, Blaß T, Lütkebohle I, *et al*. Response-Time analysis of ROS 2 processing chains under reservation-based scheduling. Proc. of the 31st Euromicro Conf. on Real-Time Systems (ECRTS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[11] Kim J, Smereka JM, Cheung C, *et al*. Security and performance considerations in ROS 2: A balancing act. 2018.

[12] Schlesselman JM, Pardo-Castellote G, Farabaugh B. OMG data-distribution service (DDS): Architectural update. Proc. of the IEEE MILCOM 2004 Military Communications Conf. IEEE, 2004. 961–967.

[13] Guesmi T, Rekik R, Hasnaoui S, *et al*. Design and performance of DDS-based middleware for real-time control systems. IJCSNS, 2007, 7(12): 188–200.

[14] Corsaro A, Schmidt DC. The data distribution service—The communication middleware fabric for scalable and extensible systems-of-systems. System of Systems, 2012.

[15] Chen C. Design and implementation of data distribution system based on DDS [Master. Thesis]. Shamghai: Fudan University, 2008 (in Chinese with English abstract).

[16] Norman G, Parker D, Sproston J. Model checking for probabilistic timed automata. Formal Methods in System Design, 2013, 43(2): 164–190.

[17] Kwiatkowska M, Norman G, Parker D. PRISM: Probabilistic symbolic model checker. Proc. of the Int'l

Conf. on Modelling Techniques and Tools for Computer Performance Evaluation. Berlin, Heidelberg: Springer, 2002. 200–204.

[18] Kwiatkowska M, Norman G, Parker D. PRISM: Probabilistic model checking for performance and reliability analysis. ACM SIGMETRICS Performance Evaluation Review, 2009, 36(4): 40–45.

**Qian Lu**, master. Her research interests include formal modeling and verification of embedded systems and network protocol analysis.

**Rui Wang**, Ph.D., professor, doctoral supervisor. Her research interests include formal methods and artificial intelligence.

**Xiaojuan Li**, Ph.D., professor, doctoral supervisor. Her research interests include formal modeling and verification of embedded systems and network protocol analysis.

**Zhiping Shi**, Ph.D., professor, doctoral supervisor. His research interests include formal methods and artificial intelligence.

**Yong Guan**, Ph.D., professor, doctoral supervisor. His research interests include formal methods and software security verification.