Telecommunication Systems (2023) 84:235–250
https://doi.org/10.1007/s11235-023-01045-x

# Optimization of message delivery reliability and throughput in a DDS-based system with per-publisher sending rate adjustment

Ridlo Sayyidina Auliya[1] · Chia-Ching Chen[1] · Po-Ru Lin[1] · Deron Liang[1] · Wei-Jen Wang[1]

Accepted: 26 July 2023 / Published online: 14 August 2023


## Abstract

Data distribution service (DDS) is a communication middleware that has been widely used in various mission-critical systems. DDS supports a set of attributes and quality of service (QoS) policies that can be tuned to guarantee important performance factors in mission-critical systems message delivery (communication), such as reliability and throughput. However, optimizing reliability and throughput simultaneously in a DDS-based system is challenging. Adjusting the publisher's sending rate is a direct approach to control the performance of a DDS-based system, but to the best of our knowledge, only a few research have examined this approach. In this study, we proposed a novel algorithm that adjusts the sending rate of each publisher to optimize the message delivery reliability and throughput of a DDS-based system. We also developed a DDS-based system model and use the model to define topic-based reliability and throughput. According to our experimental results, the proposed algorithm achieves a system communication reliability of 99–99.99%, given three scenarios of different reliability issues (70–99.99% reliability). Most importantly, the proposed algorithm can slightly increase per-topic throughput while improving per-topic reliability.

**Keywords** Data distribution service (DDS) · Reliability · Throughput · QoS · Topic-based publish-subscribe

## 1 Introduction

Data Distribution Service (DDS) is an open standard communication middleware that aims at effective and high-performance publish-susbcribe data exchange. DDS was established by the Object Management Group (OMG) [1], then extended by several DDS implementations, such as Vortex OpenSplice [2, 3], RTI Connext [4], and FastDDS [5]. DDS has been widely used in various mission-critical systems in industrial sectors, including robotics [6–9], national defense [10], manufacturing [11–13], and agriculture [14]. DDS defines the standard publication-subscription mechanisms that made it possible to increase the easiness of application development, deployment, and maintenance. DDS also enables timely and dependable Quality of Service (QoS) [15], and ensures fine-grained control over key performance factors of message delivery in mission-critical systems, such as reliability and throughput.

Mission-critical systems typically require low-latency, high-bandwidth dependable data exchange. As a result, message delivery reliability and throughput become two important performance factors, where reliability is the probability that a message is successfully delivered to its target, and throughput is the amount of message that can be transmitted in a given time frame [14]. Guaranteeing reliability while maintaining high throughput is challenging. To ensure reliability, DDS-based systems must have enough computing power and bandwidth to handle each piece of transmitted data. This means that, the messages sending rates of the publishers, the computing power, and the bandwidth decide the reliability and throughput of DDS-based systems.

✉ Wei-Jen Wang
wjwang@csie.ncu.edu.tw

Ridlo Sayyidina Auliya
rsayyidinaa@gmail.com

Chia-Ching Chen
blackspeedchen@gmail.com

Po-Ru Lin
flyotlin@gmail.com

Deron Liang
deronliang@gmail.com

1 Department of Computer Science and Information Engineering, National Central University, No. 300, Zhongda Road, Taoyuan City 320, Taiwan

The OMG DDS standard provides 22 powerful QoS policies that can be used to tune performance in DDS-based systems [1]. The QoS policies enable multilevel configuration and allow higher performance of DDS than the other types of middleware [16]. Many studies [17–32], have shown how QoS policies can be used to enhance the performance of DDS-based systems in terms of various performance criteria, such as reliability, throughput, latency, and jitter. However, only a few studies have pointed out the limitation of QoS policy adjustment [21, 26]. The study by Maruyama et al. found that QoS policies are insufficient for real-time processing measurement [21]. The study by Alaerjan et al. revealed that adjusting the QoS policies while maintaining the balance between performance indicators requires additional computing resources [26]. Moreover, a large number of possible QoS policy configurations and tradeoffs among the performance criteria contribute to the complexity of QoS policy adjustment [33, 34]. In summary, although DDS supports some performance criteria-related QoS policies and those QoS policies can be used to improve the performance of DDS-based systems, these QoS policies are not adequate to address all performance problems in DDS-based systems [35].

To address the limited performance improvements obtained from the QoS policy adjustment, we propose an approach to configure DDS-based systems, which involves adjusting the publisher's sending rate, given a DDS-based system configuration and observed performance values. The sending rate is an essential parameter that directly affects publisher data delivery. The adjustment of the publisher's sending rate requires a careful calculation to obtain an appropriate value. Increasing the publisher sending rate increases throughput, but also increases workload, which may negatively affect reliability. Hence, a balance needs to be achieved between performance, computing power, and bandwidth. Meanwhile, determining the optimum sending rate in different scenarios is challenging. The sending rate should not be set arbitrarily or based on guesswork, or past experience. In the literature, the sending rate adjustment approach was only proposed for a DDS-based system that only supports unicast (point-to-point) communication [36].

In this study, we propose a novel algorithm that aims to optimize the reliability and throughput of a DDS-based system by adjusting the publisher's sending rate, given a DDS-based system configuration and observed performance values. The algorithm determines the communication capacity of each host, then calculates the optimum throughput of a multicast DDS topic on the basis of the determined capacity, and finally assigns a new sending rate for each publisher based on the calculated per-topic throughput values. To explain the concept of the algorithm, we defined a model for DDS-based systems that comprise hosts, publishers, DDS topics, subscribers, publish and subscribe relationships, and properties such as the publisher sending rate. Notice that the subscribers may experience different reception rates, even when they receive the same number of messages from the same publisher. This phenomenon can occur in DDS-based systems that adopt a publish-subscribe communication model and enables a one-to-many communication relationship (i.e., from a sender (publisher) to a topic and then to many receivers (subscribers)). Performance measurement with traditional point-to-point communication architecture is unsuitable for DDS, which adopts a one-to-many communication architecture. Consequently, we propose new definitions for per-topic reliability and throughput for performance measurements. According to our experimental results, our proposed algorithm can find a balanced point for a DDS-based system that experiences reliability issues, resulting in a system communication reliability of 99–99.99% and slightly improved throughput after adjustment of the sending rate. The two major contributions of this study are as follows:

- We proposed a novel algorithm that adjusts the sending rate for each publisher in a DDS-based system. We conducted experiments in three scenarios with varying workload requirements. The proposed algorithm achieves system communication reliability of 99–99.99%, given three scenarios of different reliability issues (70–99.99% reliability). Most importantly, the proposed algorithm can slightly increase per-topic throughput while improving per-topic reliability.
- We defined a DDS-based system model and use the model to define reliability and throughput based on the topic-based publish-subscribe model. The DDS-based system has a different communication mechanism from the point-to-point communication model. DDS-based systems have a one-to-many communication architecture in which different topics are broadcast to multiple subscribers; consequently, new performance evaluation metrics should be established for DDS-based systems. To the best of our knowledge, reliability, and throughput based on the publish-subscribe communication model has not yet been defined.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 presents the formal definitions for the DDS-based system model and per-topic reliability and throughput. Section 4 provides an explanation of the proposed algorithm for the adjustment of the sending rate. Section 5 presents the experimental results. Finally, Sect. 6 provides the study conclusion.

## 2 Related work

In the literature, there are two types of approaches for tuning the performance in a DDS-based system. The first type of approach aims to tune the QoS policies based on a guideline provided by human experts [17–32]. The second type of approach aims to tune the sending rates of the publishers automatically under a unicast communication architecture [36]. We will explain them respectively in the following subsections.

### 2.1 QoS policy adjustment

DDS supports a set of QoS policies that can be tuned to achieve the optimum performance of a DDS-based system. The OMG [1] specifies 22 standard QoS policies which were later extended by DDS implementation, such as RTI Connext with 54 QoS policies [4]. Each QoS policy governs a specific aspect of the behavior of the DDS-based system. Thus, appropriate QoS policy adjustment is required to allow a DDS-based system to achieve better performance.

Many studies [17–32] have applied the QoS policies compliant with the standard specification [1], and showed that some configurations of QoS policies can improve the performance of a DDS-based system, such as applying the RELIABILITY and HISTORY QoS to optimize reliability [23, 25]. Those existing studies have demonstrated that QoS policy adjustment can improve the performance of DDS-based systems in a wide variety of performance criteria. However, only a few of them pointed out the limitations of QoS policy adjustment. Although DDS supports some performance criteria-related QoS policies, those QoS policies are not adequate to address all performance problems in DDS-based systems [35]. The DDS-based system scenarios can vary and thus, those QoS policies configurations cannot be directly applied to all cases. Furthermore, there are many different QoS parameters and tradeoffs between the performance criteria [7, 21]. The QoS policy adjustment also requires more computing resources [26] and is time-consuming, due to the numerous attempts required to obtain the optimal value. Such approaches rely on human users' intervention and their experience to find the optimal QoS policies configuration. Therefore, it is difficult for a human user to identify an optimal QoS policies configuration that fits a certain scenario, service, or network condition.

The strategy to address the limitation of QoS policy adjustment has been discussed in a study by Yoon et al. [33]. The study proposes a mechanism namely *QoS Optimizer* to identify a suitable QoS policies configuration to improve DDS-based system performance. The suggested QoS policies configuration is generated through automatic and continuous performance value monitoring of the DDS entities. The proposed *QoS Optimizer* monitors the DDS-based system to identify performance problems, then it sent the performance information to the *Analyzers* to analyze the QoS policies and system performance related to the problem. Finally, the *QoS Optimizer* will adjust the QoS policies value. Note that this paper did not show how to identify the QoS policies and specify their value. Nonetheless, without specifying the QoS policies and their value, the *QoS Optimizer* will be an inefficient random guess mechanism.

### 2.2 Publisher's sending rate adjustment

DDS implementations, such as FastDDS [5] and RTI ConnextDDS [4] support a feature to adjust the sending rate of the publisher, namely *FlowController*. The *FlowController* feature limits the rate of messages from the publisher to avoid flooding to subscribers and determines when the publisher is allowed to send data and how much. The feature can be tuned in a specific QoS policy, simultaneously in the publisher creation. The study by Kang et al. [16] utilizes the *FlowController* feature to evaluate the performance of DDS and compare it with other publish-subscribe technologies, such as MQTT and ZeroMQ, in terms of latency and throughput. The publisher sending rate limit (flow control mechanism) is described as one of the essential performance-related properties. They performed several experiments in unicast and multicast communication with three different data flow scenarios, including high-frequency, periodic, and sporadic data flow. The sending rate value was adjusted in the *FlowController* to specify the maximum rate at which a DDS publisher may send samples. The results show that the publisher sending rate has an influence on DDS performance. However, the *FlowController* requires many experiments by humans to determine the appropriate value to perform sending rate adjustment. In addition, the vendor-specific setting might limit the applicability of the feature.

Adjusting the publisher's sending rate can be a good solution to improve the performance of the DDS-based system, however, there are only a few studies that discuss this approach. The study by Martin-Carrascosa et al. [36] proposed NAPA (Non-supervised Adaptive Publication Algorithm), a dynamic auto-tuning algorithm for unicast DDS. The algorithm focuses on dynamically adjusting the middleware parameters according to the system conditions. The algorithm adjusts the sending rate of the publisher according to the sending window size, which is determined by the threshold. The lower bound of the threshold is set as the minimum sending rate that the algorithm can tune. Meanwhile, the maximal sending rate that can be sent by the publisher is calculated from the network bandwidth, the Round-Trip Time (RTT), and message properties, such as message size. Their experiment results demonstrated that the proposed algorithm effectively improves the performance of DDS-based systems in terms of sample latency and overall

**Table 1** Comparison of the related studies

| Approach | Concept | Limitations |
|---|---|---|
| QoS Policy Adjustment: Guideline-based Adjustment [17–32] | This kind of method only suggests a guideline to adjust QoS policies, such as RELIABILITY and HISTORY [23, 25] | It is time-consuming and requires human experts to tune the QoS policies many times to obtain the optimal configuration |
| QoS Policy Adjustment: QoS Optimizer [33] | It provides a strategy to automatically try and evaluate a configuration for QoS policy adjustment | It did not provide a definite way to choose a QoS policy for performance tuning, nor did it specify a way to calculate the new value to update the target QoS policy |
| Sending Rate Adjustment: NAPA [36] | It uses two thresholds to control the sending rate of the publishers dynamically in a range of two thresholds. The thresholds are calculated from network bandwidth, Round-Trip Time (RTT), and message properties | It cannot be used in a multicast DDS-based system |

throughput. However, the proposed approach is very restrictive for DDS-based systems since DDS allows multicast communication which includes data transmission from many publishers and subscribers.

### 2.3 Summary

In summary, determining the appropriate publisher's sending rate is challenging. The sending rate adjustment should not be set arbitrarily or based on guesswork or past experiences. Meanwhile, the QoS policy adjustment and the existing approach to adjust the publisher's sending rate has several limitations (see Table 1). In this study, we proposed a novel algorithm that tunes the sending rate for each publisher in a DDS-based system. Our proposed algorithm adjusts the sending rate of each publisher according to the observed performance values, which is the reception rate of subscribers. Our proposed algorithm is based on a publish-subscribe model that can work both in unicast and multicast. In addition, our proposed model can be applied alongside the standard QoS policies, making it possible to apply across different DDS implementations and various cases.

## 3 Reliability and throughput of a DDS-based system

In this section, we first define the DDS-based system; then, based on this definition, we define per-topic reliability and throughput. The notations for the proposed DDS-based system and the proposed algorithm are listed in Tables 2 and 3, respectively.

**Table 2** Notations for the proposed DDS-based system model

| Notations | Description |
|---|---|
| $D$ | DDS-based system configuration |
| $h_j$ | The $j$th host in $D$ |
| $p_k$ | The $k$th publisher in $D$ |
| $s_k$ | The $k$th subscriber in $D$ |
| $t_i$ | The $i$th topic in $D$ |
| $ph_{jk}$ | $ph_{jk} = 1$ if publisher $p_k$ is in host $h_j$ and $ph_{jk} = 0$ otherwise |
| $pt_{ik}$ | $pt_{ik} = 1$ if publisher $p_k$ publishes to topic $t_i$ and $pt_{ik} = 0$ otherwise |
| $sh_{jk}$ | $sh_{jk} = 1$ if subscriber $s_k$ is in host $h_j$ and $sh_{jk} = 0$ otherwise |
| $st_{ik}$ | $st_{ik} = 1$ if subscriber $s_k$ subscribes topic $t_i$ and $st_{ik} = 0$ otherwise |
| $\lambda_{ik}$ | Programmed message sending rate of publisher $p_k$ to topic $t_i$ |
| $\gamma_{ik}$ | Observed message reception rate of topic $t_i$ to subscriber $s_k$ |
| $X$ | Number of publishers in $D$ |
| $Y$ | Number of subscribers in $D$ |
| $Z$ | Number of hosts in $D$ |
| $\alpha_i$ | Number of subscribers of topic $t_i$ |
| $Thrt(t_i)$ | Per-topic throughput of topic $t_i$ |
| $Rel(t_i)$ | Per-topic reliability of topic $t_i$ |
| $er_i$ | Expected message reception rate from topic $t_i$ to all the subscribers |

### 3.1 DDS-based system configuration

The DDS-based system consists of the following major components as follows:

**Table 3** Notations for the proposed algorithm

| Notations | Description |
|---|---|
| $ers_i$ | Expected message reception rate from topic $t_i$ to one subscriber |
| $\mu_j$ | Expected message reception rate of a host $h_j$ |
| $\mu'_j$ | Observed message reception rate of a host $h_j$ |
| $m_{ij}$ | Message reduction ratio of a topic $t_i$ due to the capacity of host $h_j$ |
| $Ratio_i$ | Sending rate allocation ratio of topic $t_i$ for sending rate adjustment |
| $\lambda'_{ik}$ | Newly allocated message sending rate of publisher $p_k$ to topic $t_i$ |
| $\gamma'_{ik}$ | Expected message reception rate of topic $t_i$ to subscriber $s_k$ |

1. *Topic*, an object involved in information exchange. Each topic dictates a message flow from the publishers of the topic to the subscribers of the topic.
2. *Publisher*, an object responsible for the issuance of the messages. Multiple publishers can publish to the same topic.
3. *Subscriber*, an object which receives messages from a topic. Multiple subscribers can receive messages from a single topic at DDS run-time.
4. *Host*, the computing machine where the publishers and subscribers are located.

On the basis of these major components, a DDS-based system can be defined as follows:

- **DDS-based System Configuration**

$$D = \langle H, P, S, T, PH, PT, SH, ST, \Lambda, \Gamma \rangle \tag{1}$$

- $H$ denotes the set of hosts $h_j$, where $j$ is the index of the host and the index is from 1 to $Z$.
- $P$ denotes the set of publishers $p_k$, where $k$ is the index of the publishers and the index is from 1 to $X$.
- $S$ denotes the set of subscribers $s_k$, where $k$ is the index of subscribers and the index is from 1 to $Y$.
- $T$ denotes the set of topics $t_i$, where $i$ is the index of the topic and the index is from 1 to $N$.
- $PH$ denotes the set of relationship $ph_{jk}$ between publisher $p_k$ and host $h_j$. The variable $ph_{jk}$ is a Boolean value where 1 denotes that publisher $p_k$ runs on host $h_j$, and 0 denotes otherwise.
- $PT$ denotes the set of publication relationship $pt_{ik}$ between publisher $p_k$ and topic $t_i$. The variable $pt_{ik}$ is a Boolean value where 1 denotes that publisher $p_k$ publishes to topic $t_i$, and 0 denotes otherwise.
- $SH$ denotes the set of relationship $sh_{jk}$ between subscriber $s_k$ and host $h_j$. The variable $sh_{jk}$ is a Boolean value where 1 denotes that subscriber $s_k$ runs on host $h_j$, and 0 denotes otherwise.
- $ST$ denotes the set of subscription relationship $st_{ik}$ between subscriber $s_k$ and topic $t_i$. The variable $st_{ik}$ is a Boolean value where 1 denotes that subscriber $s_k$ subscribes to topic $t_i$, and 0 denotes otherwise.
- $\Lambda$ denotes the set of programmed sending rates $\lambda_{ik}$ of all publisher-topic pairs $pt_{ik}$. $\Lambda = \{\lambda_{ik} \mid p_k \in P \wedge t_i \in T\}$, where $\lambda_{ik}$ is the sending rate of $p_k$. Variable $k$ is the index of the publisher and $t_i$ is the target topic for $p_k$. The sending rate is measured in messages per second.
- $\Gamma$, denotes the set of observed reception rates $\gamma_{ik}$ of all topic-subscriber pairs $st_{ik}$. $\Gamma = \{\gamma_{ik} \mid s_k \in S \wedge t_i \in T\}$, where $\gamma_{ik}$ is the sending rate of $s_k$. Variable $k$ is the index of the subscriber and $t_i$ is the target topic for $s_k$. The reception rate is measured in messages per second.

The concept of Eq. 1 is illustrated in Fig. 1, where a DDS-based system $D$ consists of topics $t_1$ and $t_2$. Topic $t_1$ is subscribed by subscribers $s_1$ and $s_2$ and topic $t_2$ is subscribed by subscriber $s_3$ and $s_4$. Suppose that three hosts exist. Publishers $p_1$ and $p_2$ are located in host $h_1$. Subscribers $s_1$ and $s_2$ are located in host $h_2$. Subscribers $s_3$ and $s_4$ are located in host $h_3$. Publisher $p_1$ sends a number of messages to topic $t_1$ with sending rate $\lambda_{1,1}$. These messages are then forwarded to subscriber $s_1$ with reception rate $\gamma_{1,1}$ and to subscriber $s_2$ with reception rate $\gamma_{1,2}$. Publisher $p_2$ sends messages to topic $t_2$, which then forwards the messages to subscriber $s_3$ and $s_4$ at reception rates $\gamma_{2,3}$ and $\gamma_{2,4}$. The presence of a publisher or a subscriber in a host is defined by the publisher-host relationship $PH$ and the subscriber-host relationship $SH$. The values of publisher-host relationship $ph_{1,1}$ and $ph_{1,2}$ and subscriber-host relationships $sh_{2,1}$, $sh_{2,2}$, $sh_{3,3}$, and $sh_{3,4}$ are 1 (true), indicating that the publisher and subscriber are in some hosts. The publication relationships $pt_{1,1}$ and $pt_{2,2}$ are 1 (true), indicating that the publishers are sending messages to some topics. The subscription relationships $st_{1,1}$, $st_{1,2}$, $st_{2,3}$, and $st_{2,4}$ are 1 (true), indicating that the subscribers receive messages from some topics.

### 3.2 Reliability and throughput definition

A DDS-based system adopts the publish-subscribe communication model, which is different from the traditional point-to-point model. The point-to-point model only considers one-to-one communication, whereas DDS features one-to-many communication between senders (publishers) and receivers (subscribers). An illustration of the communication of a DDS-based system is provided in Fig. 2.

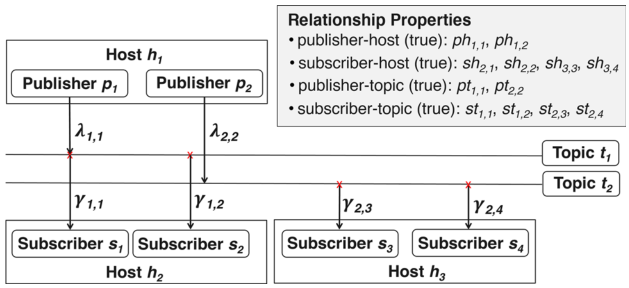


**Fig. 1** DDS-based system configuration, with programmed sending rates ($\lambda_{1,1}$, $\lambda_{2,2}$) and observed reception rates ($\gamma_{1,1}$, $\gamma_{1,2}$, $\gamma_{2,3}$, $\gamma_{2,4}$)

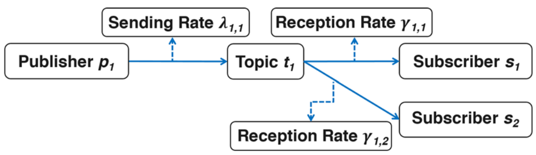


**Fig. 2** One-to-many communication pattern of a DDS-based system

As illustrated in Fig. 2, DDS-based systems do not use the point-to-point communication model. Given a number of messages sent from the publisher $p_1$ to topic $t_1$ with sending rate $\lambda_{1,1}$. Topic $t_1$ is subscribed by subscriber $s_1$ and $s_2$. The messages are forwarded by topic $t_1$ to subscriber $s_1$ with reception rate $\gamma_{1,1}$ and to subscriber $s_2$ with reception rate $\gamma_{1,2}$. The reception rates $\gamma_{1,1}$ and $\gamma_{2,2}$ can have different values. Therefore, the performance measurement used for point-to-point communication cannot be directly applied to a DDS-based system. A new performance measurement definition is required. The performance of a DDS-based system should be measured on a per-topic basis. This is because, in DDS-based systems, the topics serve as the hubs of the communication mechanism. Thus, we provide the definitions of per-topic throughput and per-topic reliability in the following subsections.

#### 3.2.1 Throughput definition

- **Per-Topic Throughput.** $\gamma_{ik}$ is the observed reception rate of subscriber $s_k$ from topic $t_i$. Note that only subscribers of topic $t_i$ can have a positive message-sending rate; non-subscribers only have a sending rate of 0. The per-topic throughput, denoted as the function $Thrt(t_i)$, can be defined as follows:

$$Thrt(t_i) = \sum_{k=1}^{Y} \gamma_{ik} \tag{2}$$

- **System Communication Throughput.** Let $D$ be the DDS-based system and $T = \langle t_1, t_2, \ldots, t_N \rangle$, where $N$ refers to the number of topics in $D$. The system communication throughput, denoted as the function $Thrt_{sys}$, can be defined as follows:

$$Thrt_{sys} = \frac{\sum_{i=1}^{N} Thrt(t_i)}{N} \tag{3}$$

#### 3.2.2 Reliability definition

- **Per-Topic Reliability.** The reliability of a topic is defined as the complement of the message loss rate of a topic. Let the expected rate of outgoing messages from topic $t_i$ be $er_i$. The gross observed message delivery rate from topic $t_i$ can be calculated as $\sum_{k=1}^{Y} \gamma_{ik}$. For this, we can calculate the reliability of a topic $t_i$ using the equation below:

$$Rel(t_i) = 1 - \frac{er_i - \sum_{k=1}^{Y} \gamma_{ik}}{er_i} \tag{4}$$

To calculate $er_i$, we need to consider the sending rate $\lambda_{ik}$ of publisher $p_k$ for topic $t_i$. In DDS, each message sent from the publisher $p_k$ to a topic $t_i$ should be delivered to all of the subscribers of topic $t_i$. Therefore, the number of messages per second from topic $t_i$ to all the subscribers should be $(\alpha_i \cdot \lambda_{ik})$. Note that only the publishers of topic $t_i$ should be considered in this case. The publish relation for topic $t_i$ can be used to select the publishers of topic $t_i$. $er_i$ is the expected message reception rate from all publishers to topic $t_i$ and can be calculated as follows:

$$er_i = \sum_{k=1}^{X} (\alpha_i \cdot \lambda_{ik}) \tag{5}$$

where $\alpha_i$ refers to the number of pairs between topic $t_i$ and all its subscribers and can be calculated based on the subscription relationship $st_{ik}$ as follows:

$$\alpha_i = \sum_{k=1}^{Y} st_{ik} \tag{6}$$

As a result, Eq. (4) can be rewritten as follows:

$$Rel(t_i) = 1 - \frac{\sum_{k=1}^{X} (\alpha_i \cdot \lambda_{ik}) - \sum_{k=1}^{Y} \gamma_{ik}}{\sum_{k=1}^{X} (\alpha_i \cdot \lambda_{ik})} \tag{7}$$

System communication reliability is defined using Eq. (8).

- **System Communication Reliability.** Let $D$ be the DDS-based system and $T = \langle t_1, t_2, \ldots, t_N \rangle$, where $N$ is the numbers of topics in $D$. The system communication reliability, denoted as the function $Rel_{sys}$, can be defined as follows:

$$Rel_{sys} = \frac{\sum_{i=1}^{N} Rel(t_i)}{N} \tag{8}$$

## 4 The sending rate adjustment algorithm

### 4.1 Algorithm design

The proposed algorithm aims to optimize the reliability and throughput of a DDS-based system by adjusting the publisher's sending rate, given a DDS-based system configuration and observed performance values. The optimization starts by determining the maximal bandwidth for the incoming messages to each host. Consequently, the maximal throughput for each topic is then determined by our algorithm. Finally, the maximal programmed sending rate for each publisher is then determined. Suppose that $D$ is the original system $\langle H, P, S, T, PH, PT, SH, ST, \Lambda, \Gamma\rangle$. Let the newly assigned sending rates calculated by the proposed algorithm be the set $\Lambda'$ and the expected reception rates of the subscribers be the set $\Gamma'$. After adjusting the sending rates, the system should replace $\Lambda$ with $\Lambda'$, and the new observed performance values should become $\Gamma'$. In addition, the new system should exhibit higher performance in terms of throughput and reliability after adjusting the sending rates.

To determine the maximal bandwidth for the incoming messages to each host, we should consider the data flow in a DDS-based system which can be abstracted into two parts: the data flow from a publisher to a topic, and the data flow from a topic to several subscribers. The variable $sh_{jk}$ reflects the presence of subscriber $s_k$ in host $h_j$. Moreover, the variable $st_{ik}$ confirms if the corresponding subscriber $s_k$ has received messages from a topic $t_i$. The values of those variables are Boolean type, and 0 indicates that the subscriber is not located in the host and has not subscribed to a topic. All the messages received by topic $t_i$ should be forwarded to all subscribers $s_k$ of topic $t_i$. Therefore, the expected message reception rate $ers_i$ from a topic $t_i$ to a subscriber can be calculated as follows:

$$ers_i = \sum_{k=1}^{X} \lambda_{ik} \tag{9}$$

For a particular host $h_j$, the expected message reception rate from topic $t_i$ to subscriber $s_k$ can be calculated as $ers_i \cdot st_{ik} \cdot sh_{jk}$, which should be 0 if subscriber $s_k$ is not located in host $h_j$. Using $ers_i$, $st_{ik}$, and $sh_{jk}$, the expected message reception rate of a host $h_j$ can be calculated as follows:

$$\mu_j = \sum_{i=1}^{N} \sum_{k=1}^{Y} (ers_i \cdot st_{ik} \cdot sh_{jk}) \tag{10}$$

In addition to the expected message reception rate of a host $h_j$, the observed message reception rate of a host $h_j$ should be calculated using Eq. (11).

$$\mu'_j = \sum_{i=1}^{N} \sum_{k=1}^{Y} (\gamma_{ik} \cdot st_{ik} \cdot sh_{jk}) \tag{11}$$

The adjustment of the programmed sending rates from the publishers is estimated based on the ratio of $\mu_j$ and $\mu'_j$. If host $h_j$ has a reliability issue, then the expected message reception rate must be more than the observed message reception rate in host $h_j$. To avoid possible message loss, we can reduce sending rates, such that the expected message reception rate is reduced to the capacity of host $h_j$, which should be the observed message reception rate $\mu'_j$. To achieve this, we can reduce the sending rate from topic $t_i$ to host $h_j$ to the message reduction ratio $m_{ij}$ of the original expected sending rate from topic $t_i$ to host $h_j$. Given topic $t_i$ and host $h_1, h_2, \ldots, h_j$ the message reduction ratio is calculated as follows:

$$m_{ij} = \begin{cases} \frac{\mu'_j}{\mu_j}, if & \exists k : (st_{ik} \cdot sh_{jk}) = 1 \\ 1, & \text{otherwise.} \end{cases} \tag{12}$$

Since the number of hosts is $Z$ hosts, the number of varying message reduction ratio $m_{ij}$ for a particular topic $t_i$ should be $Z$. To guarantee the reliability property, we should choose the minimal value of the ratios for topic $t_i$ among the different $m_{ij}$ values as follows:

$$Ratio_i = \min_{j=1}^{Z}\{m_{ij}\} \tag{13}$$

After sending rate allocation ratio $Ratio_i$ of topic $t_i$ is obtained, the new sending rate from a publisher $p_k$ to topic $t_i$ can be calculated as follows:

$$\lambda'_{ik} = \lambda_{ik} \cdot Ratio_i \tag{14}$$

Based on the aforementioned definitions and equations above, we can compose the proposed algorithm named the Sending Rate Adjustment Algorithm. The first step in the algorithm is to initialize the DDS-based system $D$ as the input, and the set of newly assigned sending rates $\Lambda'$ as the output. To determine if sending rate adjustment is necessary, we need to obtain the observed performance values. For this, we require the values of the expected message reception rate $\mu_j$ and the observed message reception rate $\mu'_j$ for each host $h_j$. The expected message reception rate $\mu_j$ is calculated using Eq. (10) and the observed message reception rate $\mu'_j$ is calculated using Eq. (11). If the value of $\mu'_j$ is equal to $\mu_j$ then the system does not have any reliability issues. Otherwise, sending rate adjustment should be performed. Because the values of $\mu'_j$ and $\mu_j$ have been obtained, in the next step, we should calculate the message reduction ratio of a topic $t_i$ due to the capacity of host $h_j$, denoted by $m_{ij}$. After the message

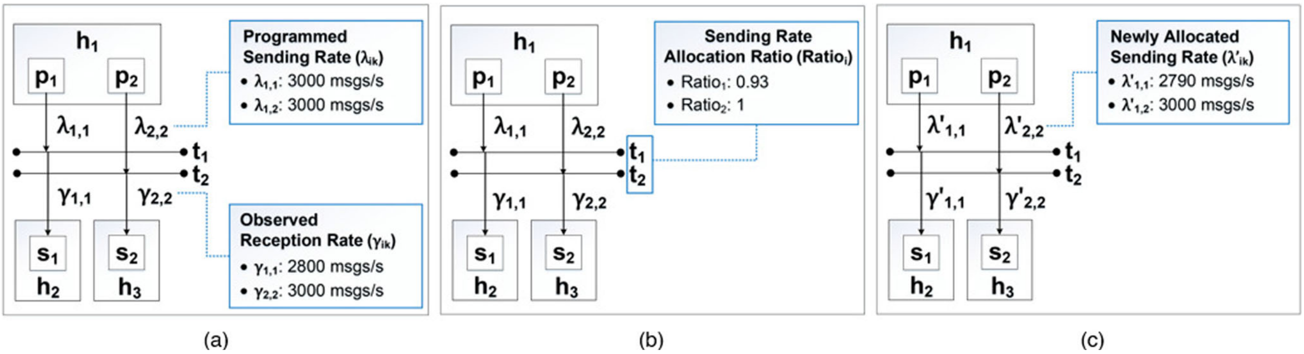


**Fig. 3** Example case using sending rate adjustment algorithm: **a** original configuration including programmed sending rate ($\lambda_{ik}$) and observed reception rate ($\gamma_{ik}$), **b** Calculating sending rate allocation ratios ($Ratio_i$): $Ratio_i$ for Topic $t_1$ and $Ratio_2$ for Topic $t_2$, and **c** Calculating newly allocated sending rate ($\lambda_{ik}$): $\lambda'_{1,1}$ for publisher $p_1$ and $\lambda'_{2,2}$ for publisher $p_2$

reduction ratio $m_{ij}$ is calculated for each host $h_j$, the minimum value of $m_{ij}$ is used to determine the sending rate assigned for topic $t_i$, which is denoted by $Ratio_i$. After determining the sending rate allocation ratio of topic $t_i$, denoted by $Ratio_i$, the new sending rate is calculated by the proposed algorithm using Eq. 14. Equation 14 results in $\lambda'_{ik}$, as the newly allocated message sending rate of publisher $p_k$ to topic $t_i$. The newly allocated message sending rate $\lambda_{ik}$ assigned to each publisher $p_k$ of topic $t_i$. Therefore, each publisher $p_k$ of topic $t_i$ will have a different sending rate based on the calculated $Ratio_i$. Finally, after all $\lambda'_{ik}$ values are obtained and assigned the variable $\Lambda'$ is returned. Note that, the new sending rate cannot be less than the allowed minimum value. Consequently, if it is less than the minimum value, the new sending rate should be the minimum value that is allowed in the DDS-based system. The sending rate adjustment is formally described in Algorithm (1).

**Algorithm 1** Sending Rate Adjustment Algorithm

**Input:** $D = \langle H, P, S, T, PH, PT, SH, ST, \Lambda, \Gamma \rangle$
**Output:** $\Lambda'$
**for** each host $h_j$ in $H$ **do**
  get $\mu_j$ using Eq. (10)
  get $\mu'_j$ using Eq. (11)
  **for** each topic $t_i$ in $T$ **do**
    get $m_{ij}$ using Eq. (12)
  **end for**
  get $Ratio_i$ using Eq. (13)
**end for**
**for** each publisher $p_k$ **do**
  **for** each topic $t_i$ **do**
    $\lambda'_{ik} = \lambda_{ik} \cdot Ratio_i$ using Eq. (14)
    put $\lambda'_{ik}$ in $\Lambda'$
  **end for**
**end for**
return $\Lambda'$

### 4.2 Example

Figure 3a shows an example of a DDS-based system configuration. Suppose that $p_1$ publishes messages to $t_1$ with sending rate $\lambda_{1,1}$ of 3000 messages/s and $p_2$ publishes messages to $t_2$ with sending rate $\lambda_{2,2}$ of 3000 messages/s. The sending rate is programmed in the publishers and can be set by the user. The publishers will send the messages to the topic, which then forward the messages to the subscribers of the topic. Let $\gamma_{1,1}$ be the reception rates of subscriber $s_1$ and $\gamma_{2,2}$ be the reception rates of subscriber $s_2$, where the reception rate values are obtained by a monitoring component or a performance profiler. Suppose that $\gamma_{1,1}$ of subscriber $s_1$ is 2800 messages/s and $\gamma_{2,2}$ of subscriber $s_2$ is 3000 messages/s. Since we have the values of programmed sending rate $\lambda_{ik}$ and observed reception rate $\gamma_{ik}$, we can calculate the expected message reception rate $\mu_j$ and the observed message reception rate $\mu'_j$ of a host $h_j$. Consequently, we can calculate the sending rate allocation ratio $Ratio_i$ for each topic $t_i$ as shown in Fig. 3b. Finally, we can calculate the new sending rate $\lambda'_{ik}$ that will be assigned to each publisher as shown in Fig. 3c. The calculation results show that we only need to modify the sending rate of publisher $p_1$.

## 5 Experiments

In this section, we introduce the experiments that were performed to validate the claimed contributions of the proposed algorithm. First, we describe the experimental design and setup. Then, we present and discuss the experimental results.

### 5.1 Experimental design and setup

We designed several experiments to evaluate the proposed algorithm. We defined three cases that represent different

**Table 4** Experiment parameters, where sending rate represents programmed sending rate to a topic from a publisher

| Case | Amount | | | | Sending rate | |
|---|---|---|---|---|---|---|
| | Pub | Sub | Topics | Hosts | High | Low |
| 1 | 3 | 3 | 3 | 4 | 7000 | 3000 |
| 2 | 13 | 17 | 13 | 4 | 7000 | 3000 |
| 3 | 30 | 34 | 30 | 4 | 7000 | 3000 |

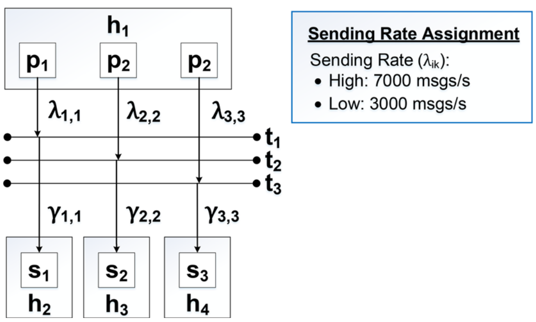


**Fig. 4** Case 1

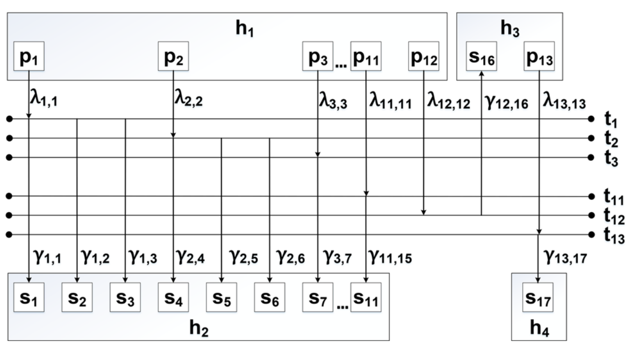

**Fig. 5** Case 2

structures of DDS-based systems with different workloads, as shown in Table 4. The first case (Fig. 4) represents a simple DDS-based system structure that consists of a small number of publishers where each publisher sends messages to a subscriber. A distinct topic is assigned to each publisher-subscriber pair, and the subscribers are deployed in three hosts. The second (Fig. 5) and third cases (Fig. 6) represent more complex DDS-based systems, that use the one-publisher-many-subscribers communication model with higher workloads than that in the first case.

In our experiments, we used two QoS policies configuration, namely *omg-def* and *omg-rel* based on the standard configuration of RELIABILITY and HISTORY QoS policies [1]. The *omg-def* QoS policy provides higher throughput and lower communication reliability for most DDS-based systems, whereas the *omg-rel* QoS policy emphasizes communication reliability. We also used two workload settings for each experiment case; the high-workload scenario had

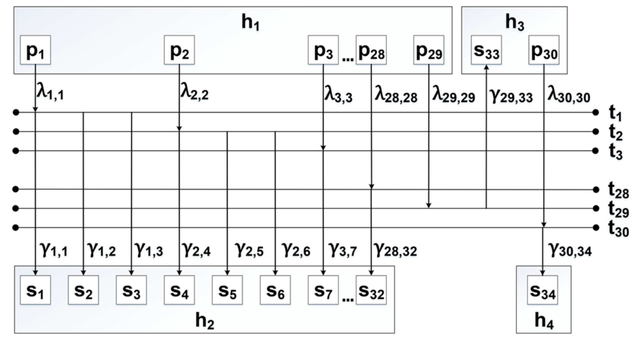

**Fig. 6** Case 3

a sending rate of 7000 msgs/s and the low-workload scenario had a sending rate of 3000 msgs/s as shown in Table 4. Note that, the sending rate values are programmed inside the publisher, and the value is achieved in a best-effort manner. In each experiment, each publisher needs to send out 10,000 messages based on their sending rate. The system then collects the reception rate, which is the number of messages successfully received by the subscribers to obtain the per-topic reliability calculated using Eq. (7). Moreover, the per-topic throughput is represented by the number of received messages per second using Eq. (2). Based on the initial result in each experiment (observed reception rates), we used the proposed algorithm to calculate a new sending rate for each publisher and reran the experiment. Then we collected the new results in terms of per-topic throughput and reliability. The detailed experimental results are presented in the following subsections.

## 5.2 Experimental results

In the first case, we ran two scenarios using the two workloads (sending rates) settings mentioned in Table 4. Figure 7a and b presents the per-topic throughput and reliability in the high-workload scenario. The system communication throughput in the high-workload scenario (7000 msgs/s for each publisher) was 6989 msgs/s for *omg-def* QoS policy and 6998 msgs/s for *omg-rel* QoS policy, as shown in Fig. 7a. The system communication throughput in the low-workload scenario (3000 msgs/s for each publisher) was 2987 msgs/s for *omg-def* QoS policy and 2989 msgs/s for *omg-rel* QoS policy as shown in Fig. 8a. Using this information, the proposed algorithm can assign a new sending rate for each publisher as shown in Table 5.

The per-topic throughput and reliability for the high-workload and low-workload scenarios are provided in Figs. 7 and 8. The system communication throughput and reliability for the high-workload and low-workload scenarios are provided in Tables 6 and 7. Figures 7a and 8a provide per-topic throughput, and Figs. 7b and 8b show per-topic reliability.

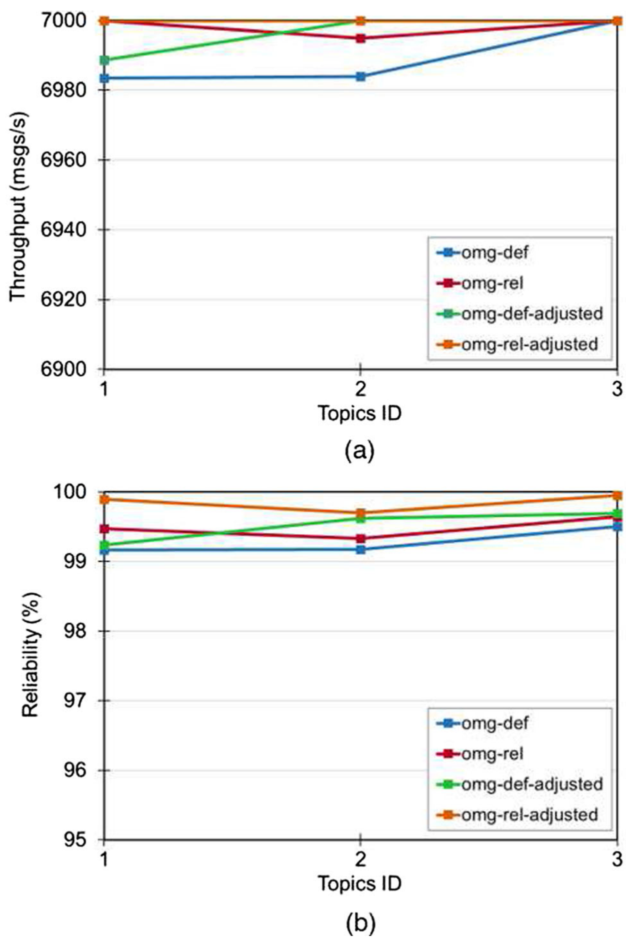


**Fig. 7** Performance comparison (before and after adjustment) in the high-workload scenario for case 1: **a** throughput and adjusted throughput **b** reliability and adjusted reliability

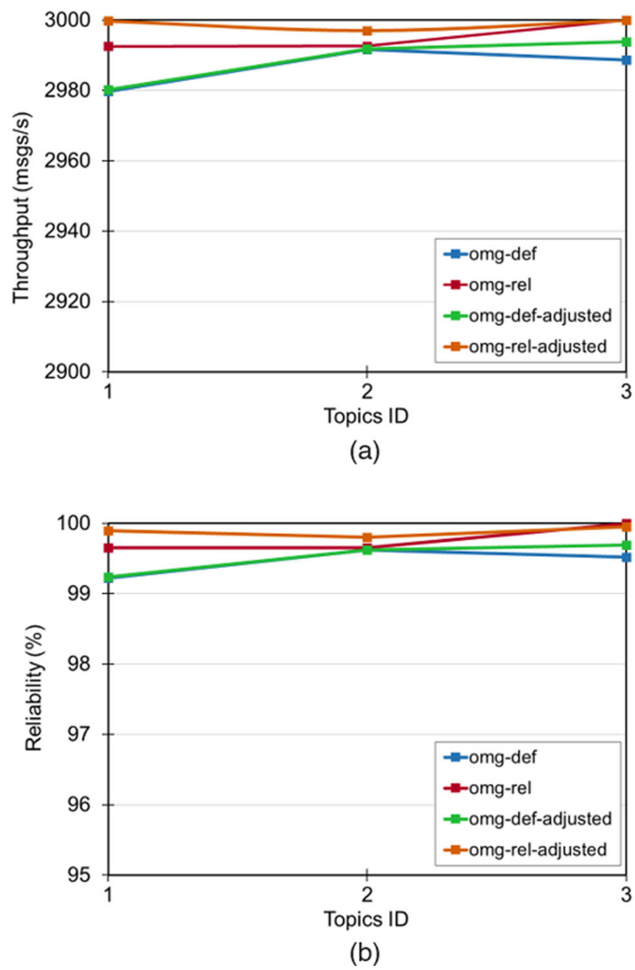


**Fig. 8** Performance comparison (before and after adjustment) in the low-workload scenario for case 1: **a** throughput and adjusted throughput **b** reliability and adjusted reliability

In the high-workload scenario, *omg-def* QoS policy results in 99.17–99.50% reliability, as shown in Fig. 7b. The per-topic reliability exhibited very little room for improvement when using the *omg-rel* QoS policy; therefore, the proposed algorithm suggested very similar sending rates. The algorithm identified a minor issue with reliability when using the *omg-def* QoS policy and the sending rates were slightly decreased to improve the performance; slight improvements were observed in both per-topic throughput and reliability as shown in Fig. 7a and b. In the low-workload scenario (Fig. 8a and b) the results after adjustment were almost identical to the original because there was very little room for improvement.

In the second case, the system communication throughput in the high-workload scenario (7000 msgs/s for each publisher) was 6872 msgs/s for *omg-def* QoS policy and 6968 msgs/s for *omg-rel* QoS policy, as shown in Fig. 9a. Meanwhile, the system communication throughput in a low-workload scenario (3000 msgs/s for each publisher) was 2993 msgs/s for *omg-def* QoS policy and 3000 msgs/s for *omg-rel* QoS policy as shown in Fig. 10a. Using this information, the proposed algorithm assigned a new sending rate for each publisher as shown in Table 8. The proposed algorithm adjusted the sending rates of publishers 1–11 to 430–460 msgs/s for the high-workload scenario and 200 msgs/s (minimal value in the system) for the low-workload scenario. The reason is that subscribers 1–15 shared the same host, and publishers 1–11 sent messages through topics 1–11 to subscribers 1–15. Therefore, the performance of topics 1–11 was strongly affected. The new sending rates were less than 7% of the original sending rates for both workload scenarios. The lower sending rates significantly improved per-topic throughput and reliability since the workload for the subscribers in host 2 was too high with the original settings.

The per-topic throughput and reliability for the high-workload and low-workload scenarios are provided in Figs. 9 and 10. The system communication throughput and reliability for the high-workload and low-workload scenarios are provided in Tables 9 and 10. Figures 9a and 10a present per-topic throughput and Figs. 9b and 10b show per-

**Table 5** Original and adjusted sending rate for case 1

| Publisher | High Workload | | | Low Workload | | |
|---|---|---|---|---|---|---|
| | Sending Rate | Adjusted Sending Rate | | Sending Rate | Adjusted Sending Rate | |
| | | omg-def | omg-rel | | omg-def | omg-rel |
| 1 | 7000 | 6980 | 7000 | 3000 | 2980 | 2990 |
| 2 | 7000 | 6980 | 6990 | 3000 | 2990 | 2980 |
| 3 | 7000 | 7000 | 7000 | 3000 | 2980 | 3000 |

**Table 6** System communication reliability and throughput of case 1 with high-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 6989 | 99.28 | 6996 | 99.52 |
| *omg-rel* | 6998 | 99.48 | 7000 | 99.85 |

**Table 7** System communication reliability and throughput of case 1 with low-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 2987 | 99.45 | 2989 | 99.52 |
| *omg-rel* | 2989 | 99.77 | 2999 | 99.88 |

topic reliability. In the high-workload scenario, the *omg-def* QoS policy resulted in system communication reliability of 97.67%. Meanwhile, the *omg-rel* QoS policy provided system communication reliability of 99.32%. The sending rate adjustment by the proposed algorithm increased the system communication reliability from 97.67 to 99.99% for the *omg-def* QoS policy, and from 99.32 to 99.99% when using *omg-rel* QoS policy. In addition, the per-topic throughputs for both QoS policies improved to 0–2%. In the low-workload scenario (Fig. 10a and b), the *omg-def* QoS policy resulted in system communication reliability of 99.72%. Meanwhile, the *omg-rel* QoS policy resulted in a system communication reliability of 99.94%. The sending rate adjustment by the proposed algorithm can increase the system communication reliability from 99.72 to 99.99% for the *omg-def* QoS policy, and from 99.94 to 99.99% for the *omg-rel* QoS policy. The sending rate adjustment can improve reliability while increasing the per-topic throughput. The performance improvement from sending rate adjustment is more obvious in the second case than in the first one.

In the third case, the system communication throughput in a high-workload scenario (7000 msgs/s for each publisher) was 6339 msgs/s for *omg-def* QoS policy and 6410 msgs/s for *omg-rel* QoS policy, as shown in Fig. 11a. Meanwhile, the system communication throughput in low workload (3000 msgs/s for each publisher) was 2996 msgs/s for *omg-def* QoS policy and 2996 msgs/s for *omg-rel* QoS policy as shown in Fig. 12a. Based on the observed performance values, the proposed algorithm assigned a new sending rate for each publisher as shown in Table 11. The proposed algorithm adjusted the sending rates of publishers 1–30 to 200 msgs/s

**Table 8** Original and adjusted sending rate for case 2

| Publisher | High Workload | | | Low Workload | | |
|---|---|---|---|---|---|---|
| | Sending Rate | Adjusted Sending Rate | | Sending Rate | Adjusted Sending Rate | |
| | | omg-def | omg-rel | | omg-def | omg-rel |
| 1 | 7000 | 460 | 460 | 3000 | 200 | 200 |
| 2 | 7000 | 460 | 460 | 3000 | 200 | 200 |
| 3 | 7000 | 460 | 460 | 3000 | 200 | 200 |
| 4 | 7000 | 450 | 460 | 3000 | 200 | 200 |
| 5 | 7000 | 460 | 460 | 3000 | 200 | 200 |
| 6 | 7000 | 460 | 460 | 3000 | 200 | 200 |
| 7 | 7000 | 430 | 460 | 3000 | 200 | 200 |
| 8 | 7000 | 450 | 460 | 3000 | 200 | 200 |
| 9 | 7000 | 450 | 460 | 3000 | 200 | 200 |
| 10 | 7000 | 450 | 450 | 3000 | 200 | 200 |
| 11 | 7000 | 430 | 460 | 3000 | 200 | 200 |
| 12 | 7000 | 6980 | 6650 | 3000 | 2970 | 3000 |
| 13 | 7000 | 7000 | 7000 | 3000 | 3000 | 3000 |

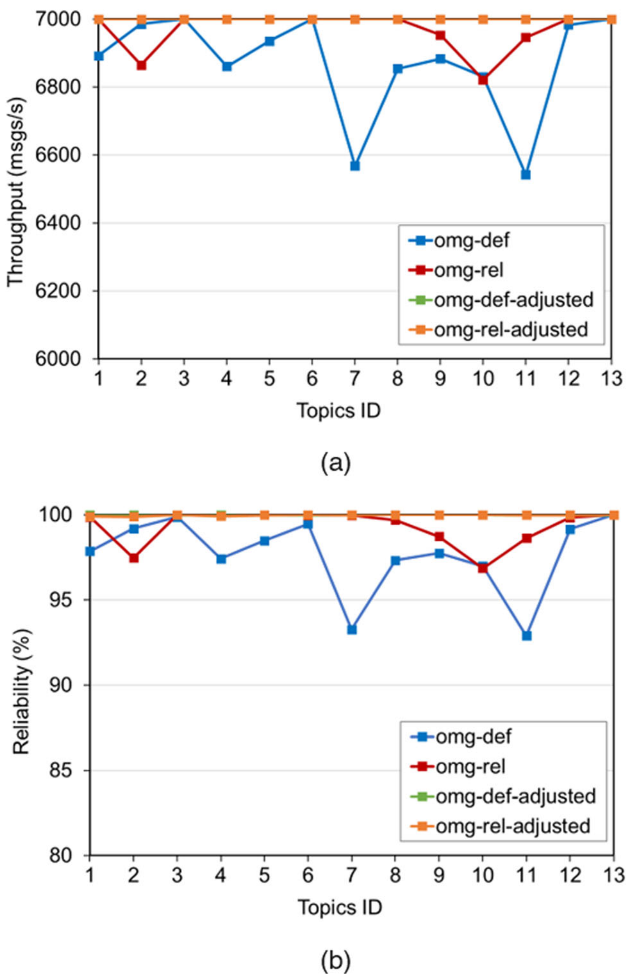


**Fig. 9** Performance comparison (before and after adjustment) in the high-workload scenario for case 2: **a** throughput and adjusted throughput **b** reliability and adjusted reliability

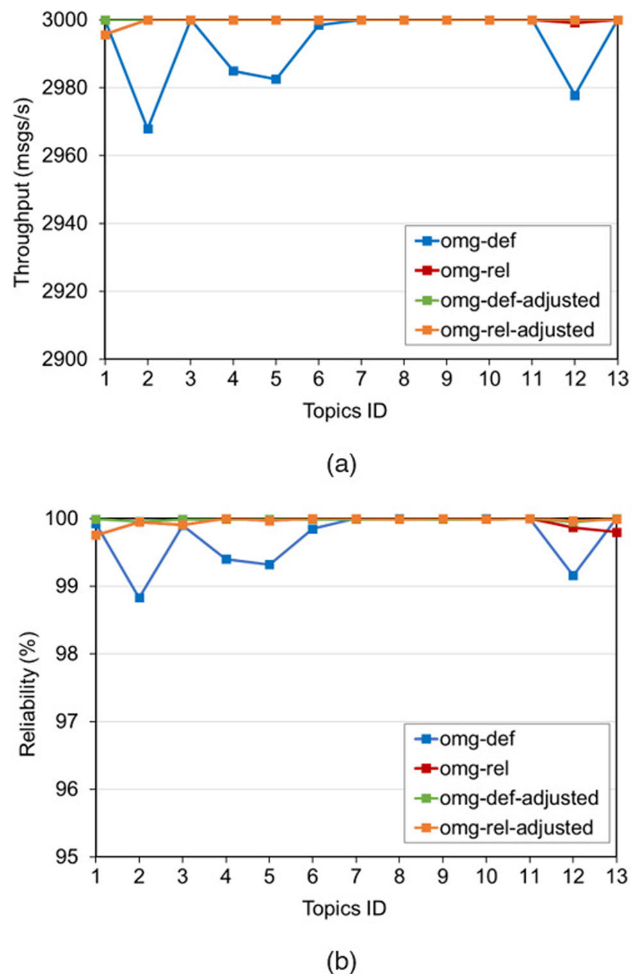


**Fig. 10** Performance comparison (before and after adjustment) in the low-workload scenario for case 2: **a** throughput and adjusted throughput **b** reliability and adjusted reliability

(minimal value in the system) in both high-workload and low-workload scenarios. The reason is that subscribers 1–32 shared the same host, and publishers 1–30 sent messages through topics 1–30 to subscribers 1–32, respectively. Therefore, the performance of topics 1–30 was strongly affected. The new sending rates were less than 10% of the original sending rates. The decreased sending rates significantly improved the per-topic throughput and reliability since the workload for the subscribers in host 2 was too high with the original settings.

The per-topic throughput and reliability for the high-workload and low-workload scenarios are provided in Figs. 11 and 12. The system communication throughput and reliability for the high-workload and low-workload scenarios are provided in Tables 12 and 13. Figures 11a and 12a present per-topic throughput, and Figs. 11b and 12b show per-topic reliability. In the high-workload scenario, *omg-def* QoS policy resulted in a system communication reliability of 90.02% and *omg-rel* QoS policy resulted in a system communication reliability of 91.04%. The sending rate adjustment by the proposed algorithm increased the system communication reliability from 90.02 to 99.99% for the *omg-def* QoS policy, and from 91.04 to 99.99% for the *omg-rel* QoS policy. In addition, the per-topic throughputs for both QoS policies improved to 0–22%. In the low-workload scenario (Fig. 12a and b), the *omg-def* QoS policy resulted in a system communication reliability of 99.85%. Meanwhile, the *omg-rel* QoS policy resulted in a system communication reliability of 99.83%. The sending rate adjustment can increase the reli-

**Table 9** System communication reliability and throughput of case 2 with high-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 6872 | 97.67 | 7000 | 99.99 |
| *omg-rel* | 6968 | 99.32 | 7000 | 99.99 |

**Table 10** System communication reliability and throughput of case 2 with low-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 2993 | 99.72 | 3000 | 99.99 |
| *omg-rel* | 3000 | 99.94 | 3000 | 99.99 |

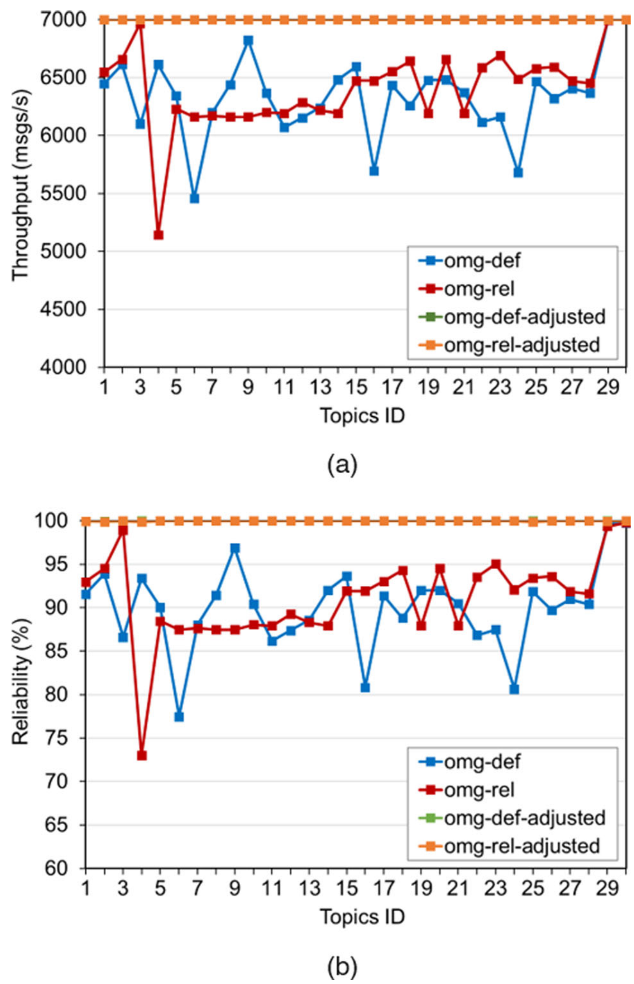


**Fig. 11** Performance comparison (before and after adjustment) in the high-workload scenario for case 3: **a** Throughput and Adjusted Throughput **b** Reliability and Adjusted Reliability

ability in the *omg-def* QoS policy from 99.85 to 99.99%, and from 99.83 to 99.99% for the *omg-rel* QoS policy. The reliability of the original configurations created the most serious reliability problem when compared with the previous two experiment cases. The proposed algorithm was able to find a set of publisher sending rates for the publishers, that increased reliability to 99.99% without sacrificing throughput.

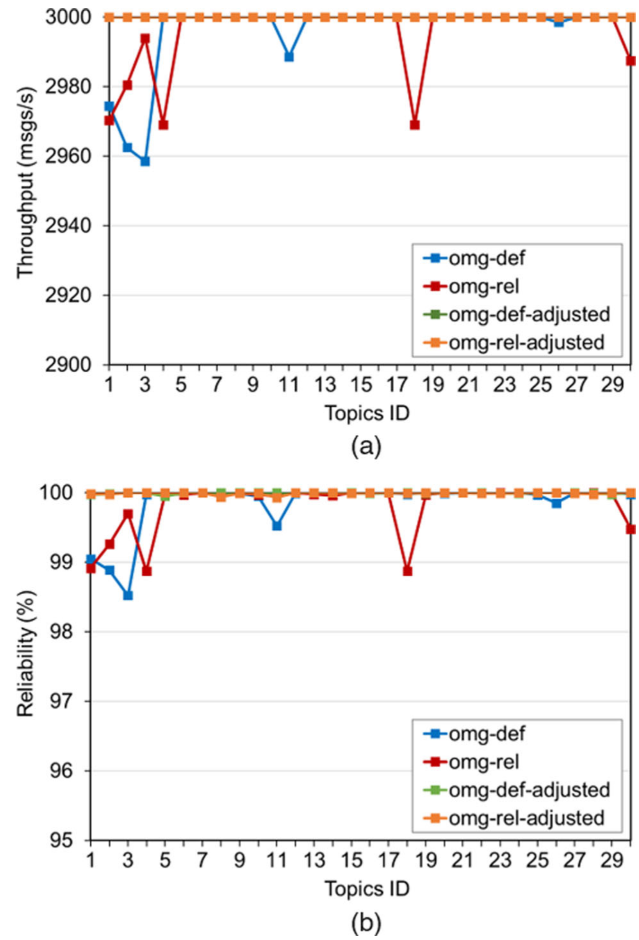


**Fig. 12** Performance comparison (before and after adjustment) in the low-workload scenario for case 3: **a** throughput and adjusted throughput **b** reliability and adjusted reliability

## 6 Conclusions

Message delivery (communication) reliability and throughput are two important factors in the DDS-based system. To improve these factors, we proposed a new algorithm to adjust the publisher sending rates, such that reliability and throughput can be optimized on a per-topic basis. We proposed a DDS-based system model and used the model to define per-topic reliability and throughput. Then, we used the definitions to compose an algorithm that adjusts the publisher sending rate based on the observed performance values of a DDS-based system. We validated our proposed algorithm through three sets of experiment cases with two workload scenarios and two QoS policies configurations. Based on our experimental results, the proposed algorithm achieves a system communication reliability of 99.52–99.99%. Most importantly, the proposed algorithm increases the per-topic throughput while improving the per-topic reliability. However, the proposed algorithm cannot further improve a DDS-based system if it already has very

**Table 11** Original and adjusted sending rate for case 3

| Publisher | High Workload | | | Low Workload | | |
|---|---|---|---|---|---|---|
| | Sending Rate | Adjusted Sending Rate | | Sending Rate | Adjusted Sending Rate | |
| | | omg-def | omg-rel | | omg-def | omg-rel |
| 1–28 | 7000 | 200 | 200 | 3000 | 200 | 200 |
| 29 | 7000 | 7000 | 7000 | 3000 | 3000 | 3000 |
| 30 | 7000 | 7000 | 7000 | 3000 | 3000 | 3000 |

**Table 12** System communication reliability and throughput of case 3 with high-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 6339 | 90.02 | 7000 | 99.99 |
| *omg-rel* | 6410 | 91.04 | 7000 | 99.99 |

**Table 13** System communication reliability and throughput of case 3 with low-workload

| Settings | Initial | | Adjusted | |
|---|---|---|---|---|
| | Throughput | Reliability | Throughput | Reliability |
| *omg-def* | 2996 | 99.85 | 3000 | 99.99 |
| *omg-rel* | 2996 | 99.83 | 3000 | 99.99 |

high per-topic reliability as shown in our first case in the experiments. Meanwhile, the performance improvement is more obvious when the DDS-based system has a low per-topic reliability problem. A limit of the proposed algorithm is that, it cannot significantly improve per-topic throughput while maintaining a system communication reliability of 99.99%. Future research may be directed toward this issue to improve the proposed algorithm based on our proposed model of a DDS-based system.

**Author Contributions** RSA was responsible for paper writing, algorithm design, paper survey, experiments, and analysis; C-CC was responsible for conceptualization and validation; P-RL was responsible for coding; DL was responsible for project supervision; W-JW: was responsible for conceptualization, algorithm design, paper review, and editing.

**Funding** This study was partially supported by the National Science and Technology Council of Taiwan, under Grants 111-2221-E-008-061- and 111-2221-E-008-059-.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

**Ethical approval** Not applicable.

## References

1. Object Management Group. (2015). OMG Data Distribution Service (DDS) Version 1.4. https://www.omg.org/spec/DDS/1.4/PDF
2. Corsaro, A. (2010). DDS QoS Unleashed. https://www.slideshare.net/Angelo.Corsaro/dds-qos-unleashed
3. Vortex OpenSplice. (2022). Vortex OpenSplice DDS Tutorial. https://www.adlinktech.com/en/vortex-opensplice-datadistribution-service
4. Real-Time Innovations. (2022). DDS: An Open Standard for Real-Time Applications. https://www.rti.com/products/dds-standard
5. eProsima. (2022). Fast DDS Documentation. https://fast-dds.docs.eprosima.com/en/latest/
6. Cruz, J. M., Romero-Garcés, A., Rubio, J. P. B., Robles, R. M., & Rubio, A. B. (2012). A DDS-based middleware for quality-of-service and high-performance networked robotics. *Concurrency and Computation: Practice and Experience, 24*(16), 1940–1952. https://doi.org/10.1002/cpe.2816
7. Fernandez, J., Allen, B., Thulasiraman, P., & Bingham, B. (2020, August). Performance study of the robot operating system 2 with qos and cyber security settings. In *2020 IEEE international systems conference (SysCon)* (pp. 1–6). IEEE. https://doi.org/10.1109/SysCon47679.2020.9275872
8. Sudhakaran, S., Mageshkumar, V., Baxi, A., & Cavalcanti, D. (2021, June). Enabling QoS for collaborative robotics applications with wireless TSN. In *2021 IEEE international conference on communications workshops (ICC Workshops)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICCWorkshops50388.2021.9473897
9. Gutiérrez, C. S. V., Juan, L. U. S., Ugarte, I. Z., & Vilches, V. M. (2018). Towards a distributed and real-time framework for robots: Evaluation of ROS 2.0 communications for real-time robotic applications. arXiv preprint arXiv:1809.02595.
10. Park, A. T., Peck, N., Dill, R., Hodson, D. D., Grimaila, M. R., & Henry, W. C. (2023). Quantifying DDS-cerberus network control overhead. *The Journal of Supercomputing, 79*(4), 3616–3642. https://doi.org/10.1007/s11227-022-04770-3
11. Calvo, I., Pérez, F., Etxeberria-Agiriano, I., & de Albéniz, O. G. (2013). Designing high-performance factory automation applications on top of DDS. *International Journal of Advanced Robotic Systems, 10*(4), 205. https://doi.org/10.5772/56341
12. Yang, J., Sandström, K., Nolte, T., & Behnam, M. (2012, September). Data distribution service for industrial automation. In *Proceedings of 2012 IEEE 17th international conference on emerging technologies & factory automation (ETFA 2012)* (pp. 1–8). IEEE. https://doi.org/10.1109/ETFA.2012.6489544
13. Al-Madani, B., Bajwa, M. N., Yang, S. H., & Saif, A. W. A. (2015). Performance evaluation of DDS-based middleware over wireless channel for reconfigurable manufacturing systems. *International Journal of Distributed Sensor Networks, 11*(7), 863123. https://doi.org/10.1155/2015/863123
14. Al-Madani, B., & Mostafa, S. M. (2021). IIoT based multimodal communication model for agriculture and agro-industries. *IEEE Access, 9*, 10070–10088. https://doi.org/10.1109/ACCESS.2021.3050391

15. Bellavista, P., Corradi, A., Foschini, L., & Pernafini, A. (2013, July). Data Distribution Service (DDS): A performance comparison of OpenSplice and RTI implementations. In *2013 IEEE symposium on computers and communications (ISCC)* (pp. 000377–000383). IEEE. https://doi.org/10.1109/ISCC.2013.6754976
16. Kang, Z., Canady, R., Dubey, A., Gokhale, A., Shekhar, S., & Sedlacek M. (2021). A study of publish/subscribe middleware under different IoT traffic conditions. In *Proceedings of the 2020 15th IEEE conference on industrial electronics and applications (ICIEA)* (pp. 7–12). https://doi.org/10.1145/3429881.3430109
17. Baldoni, R., Querzoni, L., & Scipioni, S. (2008, October). Event-based data dissemination on inter-administrative domains: Is it viable? In *2008 12th IEEE international workshop on future trends of distributed computing systems* (pp. 44-50). IEEE. https://doi.org/10.1109/FTDCS.2008.14
18. Al-Madani, B., Al-Roubaiey, A., & Baig, Z. A. (2014). Real-time QoS-aware video streaming: A comparative and experimental study. *Advances in Multimedia, 2014*, 1–1. https://doi.org/10.1155/2014/164940
19. Al-Madani, B., & Hassan, A. (2017). Data Distribution Service (DDS) based implementation of Smart grid devices using ANSI C12. 19 standard. *Procedia Computer Science, 110*, 394–401. https://doi.org/10.1016/j.procs.2017.06.082
20. An, K., Kuroda, T., Gokhale, A., Tambe, S., & Sorbini, A. (2013). Model-driven generative framework for automated omg DDS performance testing in the cloud. *ACM Sigplan Notices, 49*(3), 179–182. https://doi.org/10.1145/2637365.2517216
21. Maruyama, Y., Kato, S., & Azumi, T. (2016, October). Exploring the performance of ROS2. In *Proceedings of the 13th international conference on embedded software* (pp. 1–10). https://doi.org/10.1145/2968478.2968502
22. Lourenço, L. L., Oliveira, G., Plentz, P. D. M., & Röning, J. (2021, December). Achieving reliable communication between Kafka and ROS through bridge codes. In *2021 20th international conference on advanced robotics (ICAR)* (pp. 324–329). IEEE. https://doi.org/10.1109/ICAR53236.2021.9659422
23. García-Valls, M., Domínguez-Poblete, J., Touahria, I. E., & Lu, C. (2018). Integration of data distribution service and distributed partitioned systems. *Journal of Systems Architecture, 83*, 23–31. https://doi.org/10.1016/j.sysarc.2017.11.001
24. Al-Roubaiey, A. A., Sheltami, T. R., Mahmoud, A. S. H., & Salah, K. (2019). Reliable middleware for wireless sensor-actuator networks. *IEEE Access, 7*, 14099–14111. https://doi.org/10.1109/ACCESS.2019.2893623
25. Fu, Y., Hao, L., & Guo, D. (2019, November). Application research of distributed simulation system based on data distribution. In *2019 IEEE international conference on unmanned systems and artificial intelligence (ICUSAI)* (pp. 268–273). IEEE. https://doi.org/10.1109/ICUSAI47366.2019.9124816
26. Alaerjan, A., Kim, D. K., Ming, H., & Kim, H. (2020). Configurable DDS as uniform middleware for data communication in smart grids. *Energies, 13*(7), 1839. https://doi.org/10.3390/en13071839
27. Kronauer, T., Pohlmann, J., Matthé, M., Smejkal, T., & Fettweis, G. (2021, September). Latency analysis of ros2 multi-node systems. In *2021 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI)* (pp. 1–7). IEEE. https://doi.org/10.1109/MFI52462.2021.9591166
28. Hakiri, A., Berthou, P., Gokhale, A., Schmidt, D. C., & Gayraud, T. (2013). Supporting end-to-end quality of service properties in OMG data distribution service publish/subscribe middleware over wide area networks. *Journal of Systems and Software, 86*(10), 2574–2593. https://doi.org/10.1016/j.jss.2013.04.074
29. Agirre, A., Parra, J., Armentia, A., Ghoneim, A., Estévez, E., & Marcos, M. (2016). QoS management for dependable sensory environments. *Multimedia Tools and Applications, 75*, 13397–13419. https://doi.org/10.1007/s11042-015-2781-4
30. Saxena, S., El-Taweel, N. A., Farag, H. E., & Hilaire, L. S. (2018, October). Design and field implementation of a multi-agent system for voltage regulation using smart inverters and data distribution service (DDS). In *2018 IEEE electrical power and energy conference (EPEC)* (pp. 1–6). IEEE. https://doi.org/10.1109/EPEC.2018.8598367
31. Youssef, T. A., Elsayed, A. T., & Mohammed, O. A. (2016). Data distribution service-based interoperability framework for smart grid testbed infrastructure. *Energies, 9*(3), 150. https://doi.org/10.3390/en9030150
32. Pérez, H., & Gutiérrez, J. J. (2015). Modeling the QoS parameters of DDS for event-driven real-time applications. *Journal of Systems and Software, 104*, 126–140. https://doi.org/10.1016/j.jss.2015.03.008
33. Yoon, G., Lee, S., & Choi, H. (2016, February). Qos optimizer. In *2016 International conference on platform technology and service (PlatCon)* (pp. 1–5). IEEE. https://doi.org/10.1109/PlatCon.2016.7456819
34. Guesmi, T., Rekik, R., Hasnaoui, S., & Rezig, H. (2007). Design and performance of DDS-based middleware for real-time control systems. *IJCSNS, 7*(12), 188–200.
35. Köksal, Ö., & Tekinerdogan, B. (2017). Obstacles in data distribution service middleware: A systematic review. *Future Generation Computer Systems, 68*, 191–210. https://doi.org/10.1016/j.future.2016.09.020
36. Martin-Carrascosa, J. J., López-Vega, J. M., Povedano-Molina, J., Ramos Muñoz, J. J., & López Soler, J. M. (2014). NAPA: An algorithm to auto-tune unicast reliable communications over DDS. https://digibug.ugr.es/handle/10481/32456



**Ridlo Sayyidina Auliya** received her bachelor and master degrees in Computer and Information Science from Brawijaya University in Indonesia. Currently, she is pursuing her Ph.D. degree in Computer Science from Department of Computer Science and Information Engineering, National University, Taiwan. Her research interests are the areas of Industrial IoT systems, networking, and distributed computing.

**Chia-Ching Chen** received his bachelor and master degrees in Department of Computer Science and Information Engineering, Aletheia University, Taiwan. He is currently a Ph.D. student of Computer Science and Information Engineering in National Central University, Taiwan. His research interests are in the areas of cloud computing, IoT systems, high availability, fault tolerance, and defense technology.

**Deron Liang** received his Ph.D. degree in Computer Science from University of Maryland at College Park in the United States. He is currently a Professor and Director for Software Research Center in National Central University, Taiwan. His research interests include intelligent systems software, smart manufacturing, data analysis, software engineering, and information system security.

**Po-Ru Lin** currently an undergraduate student of Computer Science and Information Engineering in National Central University, Taiwan. His research interests are in the areas of distributed system, networking, cloud computing and virtualization.

**Wei-Jen Wang** received his Ph.D. degree in Computer Science from Rensselaer Polytechnic Institute (RPI) in the United States on December, 2006. He is currently a Professor in Department of Computer Science and Information Engineering, National Central University in Taiwan. His research interests include cloud computing, edge computing, Industrial IoT systems, and distributed computing models and algorithms.