НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2026 Том 26 № 4 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2026 Vol. 26 No 4 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2026-26-4-806-815
УДК 004.896

DDS-ориентированная отказоустойчивость передачи данных в робототехнических системах на базе ROS 2

Артем Александрович Рузметов[1]✉, Тимур Артурович Худайбергенов[2]

[1,2] Ургенчский технологический университет RANCH, Ургенч, 220100, Узбекистан

[1] ruzmetovartem@gmail.com✉, https://orcid.org/0000-0002-3491-9256
[2] timartok@gmail.com, https://orcid.org/0000-0002-5958-582X

Аннотация

Введение. Современные распределенные робототехнические системы на базе Robot Operating System 2 (ROS 2) критически зависят от качества сетевой среды. Однако при использовании беспроводных каналов связи (Wi-Fi, LoRa, 4G/5G) возникают деградации: потери пакетов, рост джиттера и микро-обрывы. Стандартные механизмы Data Distribution Service (DDS) предоставляют метрики качества обслуживания (Quality of Service, QoS), но не определяют прикладную логику адаптации. В данной работе решается проблема обеспечения непрерывности управления роботом в условиях нестабильного канала связи за счет разработки интеллектуального слоя отказоустойчивости. **Метод.** Предложена архитектура DDS-aware, интегрирующая компоненты QoS Monitor и Policy Engine. Методология базируется на онлайн-мониторинге событий DDS (DeadlineMissed, LivelinessChanged) и статистическом анализе окна доставки сообщений. Разработан алгоритм классификации состояния связи (конечный автомат) и механизмы смягчения последствий: двухпутевая доставка команд (primary/backup) с дедупликацией по серийным номерам и динамическая приоритизация трафика. Экспериментальная проверка проводилась на Linux-стенде с использованием tc netem для прецизионной эмуляции сетевых деградаций (10 % потери, задержка и джиттер, микро-обрывы) при частоте обмена 50 Гц. **Основные результаты.** Сравнительный анализ показал превосходство предложенного метода (Proposed) над стандартным подходом (Baseline). В сценарии с 10 % потерями пакетов коэффициент доставки команд *DR_c* увеличился с 99,81 % до 100 %, а *p*95 задержки снизилась с 115,52 мс до 3,51 мс. В условиях микро-обрывов предложенное решение полностью устранило провалы управления, снизив максимальное время блокировки *T_*(*blk*, *max*) с 1,052 с до 0 с. Система продемонстрировала стабильное время восстановления *T_rex* около 5 с, обусловленное политикой предотвращения осцилляций (hold-down). **Обсуждение.** Результаты подтверждают, что прикладное резервирование трафика и использование событийного анализа DDS позволяют компенсировать недостатки физического уровня связи без модификации ядра ROS 2. В отличие от стандартных методов, предложенный подход разделяет классы трафика, что гарантирует доставку критических команд управления даже при экстремальной деградации телеметрии. Ограничением метода является необходимость предварительной конфигурации профилей QoS, однако это компенсируется высокой переносимостью решения между различными реализациями ROS Middleware (Fast DDS, Cyclone DDS). Разработанная архитектура закладывает основу для создания самовосстанавливающихся коммуникационных сред в критически важных робототехнических приложениях. Робототехнические системы, работающие в динамичных и особенно беспроводных средах, подвержены деградации связи (потери пакетов, рост задержек, джиттер, кратковременные обрывы), что приводит к снижению качества телеметрии и к временным провалам доставки команд управления. В ROS 2 обмен данными реализован поверх DDS, который предоставляет политики QoS и события QoS, однако не задает прикладную политику отказоустойчивости. Предложен DDS-aware слой отказоустойчивой коммуникации для ROS 2, включающий монитор QoS и модуль мониторинга качества обслуживания, классифицирующий состояние связи и запускающий меры смягчения: приоритизации команд, адаптацию потоков телеметрии и переключение между заранее подготовленными профилями QoS. Представлена воспроизводимая методика оценки в условиях искусственно внесенных ухудшений канала связи.

Ключевые слова

ROS 2, DDS, QoS, события QoS, отказоустойчивость, деградация связи, провал управления, восстановление

Ссылка для цитирования: Рузметов А.А., Худайбергенов Т.А. DDS-ориентированная отказоустойчивость передачи данных в робототехнических системах на базе ROS 2 // Научно-технический вестник информационных технологий, механики и оптики. 2026. Т. 26, № 4. С. 806–815. doi: 10.17586/2226-1494-2026-26-4-806-815

DDS-based fault tolerance for robotic data communication in ROS 2

Artem A. Ruzmetov[1]✉, Timur A. Khudaybergenov[2]

[1,2] Urgench University Technologies RANCH, Urgench, 220100, Uzbekistan

[1] ruzmetovartem@gmail.com✉, https://orcid.org/0000-0002-3491-9256
[2] timartok@gmail.com, https://orcid.org/0000-0002-5958-582X,

Abstract

Modern distributed robotic systems based on Robot Operating System 2 (ROS 2) critically depend on the quality of the network environment. However, when wireless communication channels, such as Wi-Fi, LoRa, and 4G/5G, are used, various forms of degradation occur, including packet loss, increased jitter, and micro-outages. Standard Data Distribution Service (DDS) mechanisms provide Quality of Service (QoS) metrics, but they do not define application-level adaptation logic. This work addresses the problem of ensuring continuous robot control under unstable communication conditions by developing an intelligent fault-tolerance layer. The authors propose a DDS-aware architecture that integrates QoS Monitor and Policy Engine components. The methodology is based on online monitoring of DDS events, such as DeadlineMissed and LivelinessChanged, as well as statistical analysis of a message delivery window. A finite-state algorithm for classifying the communication state was developed, along with mitigation mechanisms, including dual-path command delivery using primary and backup channels with deduplication based on serial numbers, and dynamic traffic prioritization. Experimental validation was carried out on a Linux-based testbed using tc netem for precise emulation of network degradation scenarios, including 10 % packet loss, delay with jitter, and micro-outages, at a communication frequency of 50 Hz. The comparative analysis demonstrated the superiority of the proposed method over the standard baseline approach. In the scenario with 10 % packet loss, the command delivery ratio, denoted as *DR_c*, increased from 99.81 % to 100 %, while the 95th percentile latency decreased from 115.52 ms to 3.51 ms. Under micro-outage conditions, the proposed solution completely eliminated control gaps by reducing the maximum blocking time, *T_*(*blk*, *max*), from 1.052 s to 0 s. The system demonstrated a stable recovery time *T_rec* of approximately 5 s which was determined by the hold-down policy used to prevent oscillations. The results confirm that application-level traffic redundancy and DDS event-based analysis can compensate for physical-layer communication limitations without modifying the ROS 2 core. Unlike standard approaches, the proposed method separates traffic classes, ensuring the delivery of critical control commands even under extreme telemetry degradation. A limitation of the method is the need for preliminary QoS profile configuration; however, this is compensated for by the high portability of the solution across different ROS Middleware implementations, including Fast DDS and Cyclone DDS. This work lays the foundation for the development of self-healing communication environments for mission-critical robotic applications. Robotic systems operating in dynamic and especially wireless environments are subject to communication degradation, including packet loss, increased latency, jitter, and short-term disconnections. These factors reduce the quality of telemetry and may lead to temporary failures in the delivery of control commands. In ROS 2, data exchange is implemented over DDS, which provides QoS policies and events; however, it does not define an application-level fault-tolerance policy. This article proposes a DDS-aware fault-tolerant communication layer for ROS 2 which includes a QoS Monitor and a Policy Engine. The Policy Engine classifies the communication state and triggers mitigation actions, including command prioritization, telemetry stream adaptation, and switching between preconfigured QoS profiles. A reproducible evaluation methodology is proposed for testing the system under artificially introduced communication channel degradations.

Keywords

ROS 2, DDS, fault tolerance, QoS, wireless robotics, network emulation, latency, jitter, recovery time, tc netem

For citation: Ruzmetov A.A., Khudaybergenov T.A. DDS-based fault tolerance for robotic data communication in ROS 2. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2026, vol. 26, no. 4, pp. 806–815 (in Russian). doi: 10.17586/2226-1494-2026-26-4-806-815

Введение

Robot Operating System 2 (ROS 2) широко используется для построения распределенных робототехнических приложений. Коммуникации в ROS 2 опираются на Data Distribution Service [1–3] через слой ROS Middleware, что обеспечивает масштабируемую модель publish/subscribe и набор настроек Quality of Service для управления компромиссом между надежностью, задержкой и ресурсами [4–6]. Тем не менее, в практических развертываниях, особенно при использовании беспроводных каналов, наблюдаются деградации связи, при которых сеть формально доступна, но параметры доставки становятся непредсказуемыми. Это критично для систем управления: «связь есть» не означает, что команды управления поступают своевременно и без провалов.

Проблема, рассматриваемая в настоящей работе, состоит в том, что DDS предоставляет механизмы (политики и события QoS), но не определяет прикладную стратегию отказоустойчивости, которая обнаруживает деградацию и инициирует согласованные действия для разных классов трафика (команды управления и телеметрия). В результате разработчики либо завышают параметры надежности (что увеличивает задержки и накладные расходы), либо получают хрупкое поведение в условиях ухудшений канала.

Цель работы — предложить и описать DDS-aware слой отказоустойчивой коммуникации в ROS 2, который использует события QoS и измеримые показатели доставки для классификации состояния связи и применения мер смягчения, ориентированных на сохранение непрерывности управления.

Предпосылки: ROS 2, DDS и QoS. В ROS 2 обмен сообщениями реализован поверх DDS. DDS предоставляет набор политик QoS, включая reliability, history, deadline и liveliness [7, 8], а также связанные с ними события, на которые может подписываться приложение.

QoS позволяет задавать ожидания по частоте/срокам доставки и условия «живости» источника данных, что полезно для диагностики деградаций и отказов коммуникации. Мониторинг этих событий в реальном времени служит основой для построения адаптивных систем управления [9]. Вместе с тем многие параметры QoS фиксируются при создании конечных точек (publisher/subscription), поэтому динамическая адаптация на практике обычно реализуется через переключение между заранее созданными профилями (эндпоинтами) или пересоздание конечных точек с новым профилем [8, 10].

Для воспроизводимого измерения временных характеристик сообщений в ROS 2 может использоваться встроенный механизм topic statistics, который позволяет собирать статистику на уровне подписчика и применять ее для диагностики производительности коммуникаций. Для распределенных ROS 2 систем в научных публикациях отмечается дополнительная end-to-end задержка по сравнению с низкоуровневой передачей на уровне DDS [1, 3], что необходимо учитывать при выборе QoS и частоты публикации.

Постановка задачи и требования

Исследуется распределенная система ROS 2 с двумя основными классами трафика: команды управления (малый объем, высокая приоритетность, строгие ограничения по задержке/устареванию); телеметрия/диагностика (более высокая частота и объем, допускает деградацию качества для сохранения управления). Под деградацией понимается частичная доступность канала при росте потерь, задержки, джиттера или возникновении «пачечной» доставки. Отдельно анализируются кратковременные микро-обрывы.

Введем ключевые показатели: доля доставки команд DR_c, $p95$ задержки L_c, джиттер J_c, максимальная длительность провала управления $T\_(blk, max)$ (наиболее длинный интервал, когда команды не приходят или становятся устаревшими) и время восстановления T_rec (время возврата к устойчивой доставке после улучшения канала).

Модель системы и допущения. Описывается распределенная робототехническая система на базе ROS 2, использующая DDS через слой ROS Middleware [4, 8]. Коммуникации разделяются на два класса трафика: команды управления $u(t)$ (малый объем, высокие требования к задержке и джиттеру); телеметрия/состояние $y(t)$ (больший объем, более мягкие требования). Для исключения взаимного влияния классов трафика команды и телеметрия передаются по отдельным топикам с независимой настройкой QoS [2, 11]. В качестве наблюдаемой величины качества канала используются статистики ROS 2 (latency, rate) и/или DDS-статистика, доступная в ряде реализаций. Сквозная задержка одной команды определяется как $L = (t\_rx - t\_pub)$, где t_pub и t_rx фиксируются на стороне публикации и приема соответственно; данные записываются в файл формата «значения, разделенными запятыми» (CSV) для последующего расчета метрик. При реализации двухпутевой доставки для класса команд управления $u(t)$ учитывается объем передаваемых данных. Типовое сообщение управления в ROS 2 (например, типа *geometr_msgs*/*Twist*) имеет размер порядка 128 Б с учетом заголовков DDS. При частоте публикации 50 Гц дублирование потока создает дополнительную нагрузку на канал в размере около 6,4 КБ/с. Такая нагрузка является незначительной даже для узкополосных беспроводных каналов, что позволяет использовать резервирование [12, 13] без ущерба для передачи основного потока телеметрии.

Модель отказов и сценарии деградации. В настоящей работе рассматриваются сетевые деградации беспроводного канала, а не отказы процессов/узлов: случайные потери пакетов (Loss); увеличение задержки и дисперсии задержки (Delay+Jitter); кратковременные разрывы связи и провалы доставки (Micro-outages). Сценарии деградации воспроизводятся управляемо на стенде с использованием сетевого эмулятора tc-netem [13, 14] (Loss, Delay, Jitter, а также кратковременное «выключение» передачи). Предполагается, что вычислительная часть узлов ROS 2 работоспособна, а деградации обусловлены каналом/маршрутом передачи [15, 16].

Функциональные и нефункциональные требования. Требования формулируются к прикладному DDS-aware слою (QoS Monitor + Policy Engine) и нацелены на сохранение управляемости роботизированного объекта при деградации связи без модификации ядра ROS 2 или стандартов DDS.

— R1 (обнаружение). Деградация должна детектироваться в онлайн-режиме по событиям QoS (deadline/liveliness) и/или оконным оценкам метрик с задержкой, достаточной для предотвращения длительных провалов управления [9, 17].
— R2 (классификация). Состояние канала должно классифицироваться минимум на классы Loss, Delay+Jitter и Micro-outages для выбора различающихся реакций политики.
— R3 (реакция политики). Policy Engine должен поддерживать меры смягчения: включение резервного пути/канала, временное дублирование команд, адаптацию QoS (reliability/history/deadline) и/или частоты телеметрии, сохраняя приоритет команд управления [18–20].
— R4 (граница провала управления). Для каждого сценария должна обеспечиваться верхняя граница $T\_(blk, max)$; в Proposed ожидается уменьшение $T\_(blk, max)$ относительно Baseline во всех сценариях.
— R5 (восстановление). После устранения деградации (t_off) система должна возвращаться к штатному режиму за ограниченное время T_rec и избегать «дребезга» переключений за счет периода стабилизации (hold-down).
— R6 (совместимость). Решение должно работать поверх стандартного RMW-интерфейса ROS 2 и быть переносимым между DDS-реализациями (например, Fast DDS и Cyclone DDS) без переписывания прикладной логики.
— R7 (низкая инвазивность). Мониторинг и политика реализуются как отдельные узлы/компоненты, использующие стандартные программные интерфейсы

ROS 2 и средства статистики; изменения в прикладных узлах минимальны.

— R8 (измеримость и воспроизводимость). Методика экспериментов должна обеспечивать повторяемое введение деградаций и сбор логов для вычисления *DR_c*, *p*95, джиттера, *T_*(*blk*, *max*) и *T_rec*, что позволяет сопоставлять Baseline/Proposed на одинаковых условиях.

Критерии валидации и измеримые метрики

Для сопоставления режимов Baseline и Proposed используются следующие метрики качества канала управления и отказоустойчивости: доля доставки команд *DR_c* (%), *p*95 задержки L_c (мс), джиттер *J_c* (мс) максимальная длительность провала управления *T_*(*blk*, *max*) (с) и время восстановления *T_rec* (с). *DR_c* определяется как 100 *N_rx*/*N_tx*, где *N_tx* — число отправленных команд за интервал эксперимента; *N_rx* — число принятых уникальных команд (с учетом дедупликации по полю seq в Proposed). *p*95 задержки *L* вычисляется как 95-й перцентиль распределения *L*, а джиттер — как стандартное отклонение задержки *L* на том же интервале.

Провал управления фиксируется как непрерывный интервал времени, в течение которого отсутствуют новые команды (наблюдается разрыв по seq) или задержка превышает порог *L_thr*. В экспериментах допустимо выбирать *L_thr* = *max* (60 мс, *p*95_baseline + 5 мс) для исключения ложных срабатываний и привязки к исходным условиям стенда. *T_*(*blk*, *max*) определяется как максимум таких интервалов в рамках сценария, а *T_rec* измеряется как время от *t_off* до момента устойчивого возвращения метрик в норму (с учетом периода hold-down). Критерий приемки: Proposed не ухудшает *DR_c* и *p*95 по сравнению с Baseline и уменьшает *T_*(*blk*, *max*) по всем сценариям, при этом *T_rec* остается ограниченным заданной политикой.

DDS-aware архитектура отказоустойчивой коммуникации. Предложенная архитектура на рис. 1 предполагает интеграцию в существующую систему двух ключевых компонентов: QoS Monitor и Policy Engine. Данный подход соответствует современным архитектурным принципам обеспечения отказоустойчивости в ROS 2 системах [20].

QoS Monitor в реальном времени собирает события QoS (такие как нарушения deadline и изменения liveliness), а также измеримые показатели доставки: задержку, джиттер и оценку потерь пакетов. На основе оконных статистических оценок монитор публикует агрегированный статус состояния канала comm_health.

Policy Engine реализует конечный автомат состояния связи (NORMAL/DEGRADED/CRITICAL/RECOVERY) и формирует управляющие действия: динамическую приоритизацию команд управления, ограничение частоты или дискретизацию телеметрии, а также переключение между заранее подготовленными профилями QoS для различных потоков данных.

В качестве источника телеметрии качества используются как механизмы topic statistics на уровне rclcpp, так и специализированные статистические модули конкретных DDS-реализаций [17, 21]. Примерами могут служить Fast DDS Statistics Module или использование параметров конфигурации (например, CYCLONEDDS_URI для Cyclone DDS) для тонкой настройки поведения RMW-слоя в условиях деградации.

Алгоритм классификации и меры смягчения. Классификация состояния связи выполняется по порогам и счетчикам событий, вычисляемым на окне наблюдений: например, превышение порога потерь, рост *p*95 задержки, серии нарушений deadline, изменения liveliness. Для предотвращения «флаппинга» применяются гистерезис и минимальное время пребывания в состоянии. Переходы между состояниями представлены на рис. 2.

Меры смягчения применяются раздельно к классам трафика. Для команд управления используется профиль/эндпоинт с приоритетом доставки и ограничением устаревания (отбрасывание слишком поздних команд) [6, 22, 23]. Для телеметрии в состояниях DEGRADED/CRITICAL применяется снижение частоты, переход на облегченные сообщения или best-effort для некритичных потоков, что снижает нагрузку и уменьшает конкуренцию с управлением [16, 24].

Методика проведения экспериментов. Оценка проводится в воспроизводимых сценариях ухудшения канала связи: сканирование потерь пакетов; внесение задержки и джиттера; микро-обрывы; комбинирован-

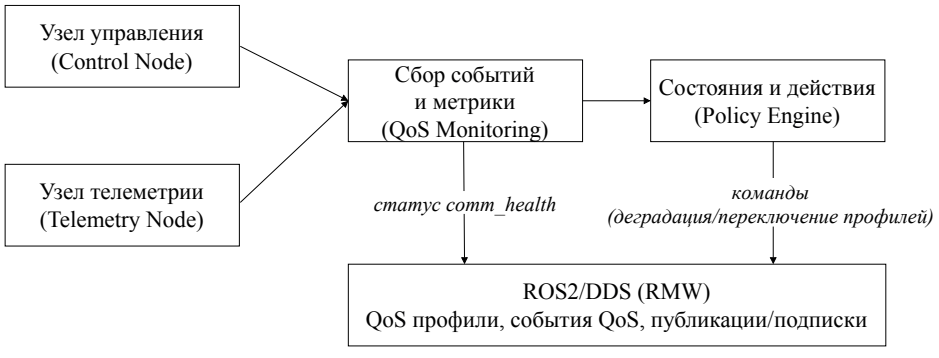

Рис. 1. Архитектура DDS-aware слоя отказоустойчивой коммуникации

Fig. 1. Architecture of the DDS-aware fault-tolerant communication layer

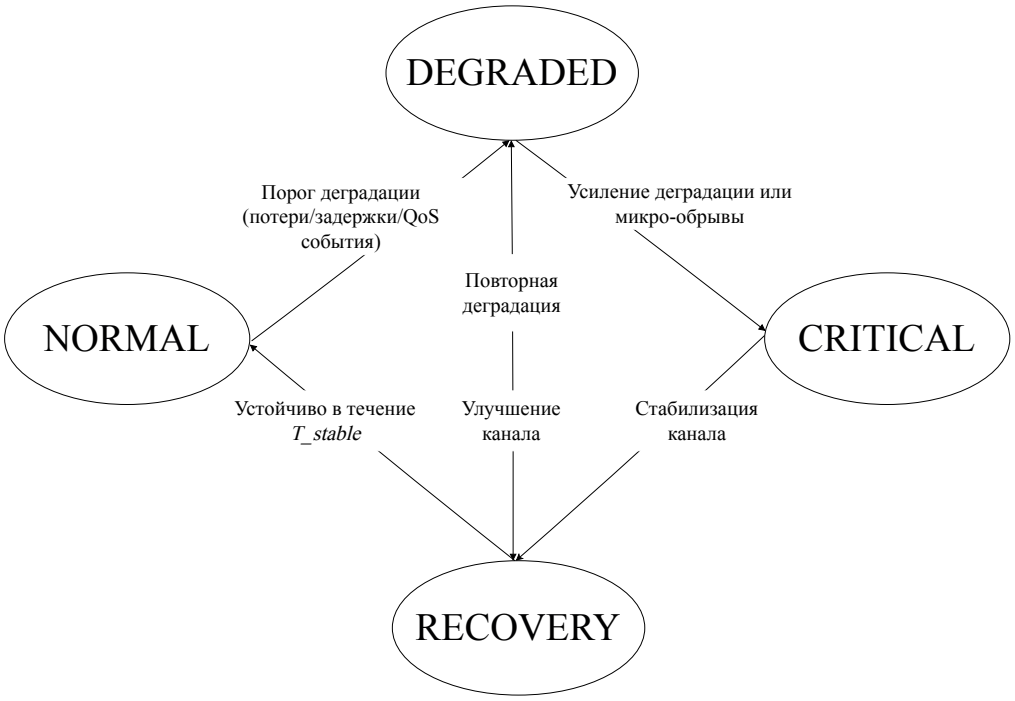

Рис. 2. Конечный автомат состояния связи (NORMAL/DEGRADED/CRITICAL/RECOVERY)
Fig. 2. Communication state finite-state machine (NORMAL/DEGRADED/CRITICAL/RECOVERY)

ные ухудшения. Сравниваются два режима: базовый (фиксированные QoS, без Policy Engine) и предложенный (QoS Monitor + Policy Engine, переключение профилей и адаптация телеметрии).

Метрики, приведенные в табл. 1, рекомендуемые для отчета: *DR_c*, *p*95 задержки *L* команд, джиттер, *T_*(*blk*, *max*), *T_rec*, а также (опционально) накладные расходы Central Processing Unit (CPU) и полоса пропускания.

Таблица 1. Экспериментальный испытательный стенд
Table 1. Experimental testbed

Компонент	Характеристики
Персональный компьютер) (host)	Intel® Core™ i5-10500H @ 2.50 GHz; 16 GB DDR4; SSD 512 GB
Операционная система	Ubuntu (Linux)
Беспроводной узел	LILYGO T-Beam (ESP32): dual-core Xtensa LX6 240 MHz; 520 KB SRAM; Flash 4 MB; GPIO/I2C/SPI/UART/PWM
Доступные каналы	Wi-Fi (ESP32), BLE/Bluetooth (ESP32), LoRa (SX127x)
Дистрибутив ROS 2	ROS 2 Foxy (*/opt/ros/foxy*)
DDS/RMW	eProsima Fast DDS (Fast RTPS); RMW: *rmw_fastrtps_cpp* (по умолчанию, *RMW_IMPLEMENTATION* не задана)
Baseline	Стандартный pub/sub ROS2 без адаптивной политики
Proposed	QoS-монитор + policy engine; слияние потоков primary/backup с дедупликацией
Деградация сети	Linux *tc netem* с network namespace и veth-парами
Сценарии	Loss 10 % (*netem loss 10 %*); Delay + Jitter (*netem delay 80 ms 20 ms distribution normal*); Micro-outages (интервальная деградация)
Профиль трафика	Частота сообщений: 50 Гц; длительность прогона: 120–180 с
Логи и метрики	CSV: *seq*, *t_pub_ns*, *t_rx_ns*, *latency_ms*; метрики: *DR_c*, *p*95 задержки *L*, джиттер (std), *T_*(*blk*, *max*), *T_rec*
Робоплатформа	Колесная платформа; контроллер Arduino (интерфейс к приводам/датчикам); контроллер передачи данных LILYGO T-Beam (ESP32)

Результаты

Результаты экспериментальной оценки качества связи и отказоустойчивости контура передачи управляющих сообщений в стенде ROS 2/DDS при моделировании деградаций канала средствами tc netem. Сравнение проводится для двух режимов: Baseline (однопутевая доставка) и Proposed (двухпутевая доставка по независимым сетевым путям с объединением потоков на стороне приемника и дедупликацией по номеру сообщения seq, а также с политикой возврата в штатный режим после устранения деградации).

Введем следующие показатели: *DR_c* — коэффициент доставки управляющих команд, определяемый как отношение числа уникальных принятых команд к числу опубликованных команд, выраженное в процентах; *p*95 — 95-й процентиль сквозной задержки доставки команд, мс; *J* — джиттер, определяемый как стандартное отклонение задержки, мс; *T_*(*blk*, *max*) — максимальная длительность провала управления, т. е. наиболее продолжительный интервал, в течение которого команды не поступают или становятся устаревшими, с; *T_rec* — время возврата алгоритма в нормальное состояние после прекращения деградации канала, с.

Табл. 2 объединяет результаты, полученные для трех классов деградации: «Потеря пакетов 10 %», «Задержка и джиттер» и «Микроразрывы связи». В предлагаемом режиме во всех сценариях коэффициент доставки управляющих команд *DR_c* достигает 100 %, а максимальная длительность провала управления *T_*(*blk*, *max*) составляет 0,000 с. В базовом режиме наиболее существенное ухудшение наблюдается при микроразрывах связи: коэффициент доставки управляющих команд снижается до 94,96 %, а максимальная длительность провала управления увеличивается до 1,052 с.

Потеря пакетов 10 %. В базовом режиме коэффициент доставки управляющих команд составляет 99,81 %, однако 95-й процентиль задержки *p*95 достигает 115,52 мс, джиттер *J* — 25,81 мс, а максимальная длительность провала управления *T_*(*blk*, *max*) — 0,260 с. В предлагаемом режиме коэффициент доставки управляющих команд достигает 100 %, 95-й процентиль задержки снижается до 3,51 мс, а джиттер — до 0,54 мс. Полученные результаты свидетельствуют об эффективной компенсации потерь за счет резервного пути передачи и объединения потоков.

Задержка и джиттер. В базовом режиме 95-й процентиль задержки увеличивается до 107,41 мс, а джиттер — до 13,61 мс, что снижает предсказуемость доставки управляющих команд. Предлагаемый режим обеспечивает снижение 95-го процентиля задержки до 4,46 мс и джиттера до 0,53 мс при сохранении коэффициента доставки управляющих команд на уровне 100 %. Это достигается благодаря переносу доставки на резервный путь при деградации основного пути и дедупликации сообщений на стороне приемника.

Микроразрывы связи. В базовом режиме микроразрывы связи приводят к снижению коэффициента доставки управляющих команд до 94,96 % и увеличению максимальной длительности провала управления до 1,052 с. При этом 95-й процентиль задержки остается низким и составляет 1,79 мс, поскольку учитывает только успешно доставленные сообщения в интервалах без обрывов. По этой причине в данном сценарии качество управления определяется прежде всего пропусками сообщений и длительностью провалов управления. В предлагаемом режиме коэффициент доставки управляющих команд достигает 100 %, максимальная длительность провала управления составляет 0,000 с, 95-й процентиль задержки — 4,08 мс, а джиттер — 0,57 мс.

Дополнительные измерения показали, что реализация модулей мониторинга и дедупликации на базе контроллера ESP32 приводит к увеличению загрузки центрального процессора на 2,6–2,7 процентного пункта по сравнению с базовым режимом. Дополнительный сетевой трафик составляет около 6,4 КБ/с и обусловлен дублированием управляющих команд при среднем размере сообщения 128 Б и частоте передачи 50 Гц. Такие накладные расходы являются допустимыми, поскольку объем управляющих сообщений существенно меньше объема передаваемой телеметрии [13, 16, 25].

Время восстановления *T_rec*. В Baseline величина *T_rec* = 0,0 с обусловлена тем, что снятие деградации (netem off) приводит к немедленному возвращению сетевых характеристик к штатным значениям, а Baseline не содержит самостоятельной логики восстановления. В Proposed *T_rec* = 5,0 с определяется политикой возврата в NORMAL (hold-down) после момента *t_off* и отражает восстановление алгоритмического состояния (стабилизацию режима), а не физическое восстановление канала. В предлагаемом режиме величина *T_rec* трактуется как время возврата системы к устойчивому режиму доставки, т. е. интервал между моментом снятия деградации канала *t_off* (например, отключение netem) и моментом, когда политика разрешает возврат в NORMAL после окна стабильности. В эксперимен-

Таблица 2. Сводные метрики качества связи и отказоустойчивости (Baseline/Proposed)
Table 2. Summary metrics of link quality and fault tolerance (Baseline/Proposed)

Сценарий	Режим	*DR_c*, %	*p*95, мс	*J*, мс	*T_*(*blk*, *max*), с	*T_rec*, с	CPU, %	Дополнительный трафик, КБ/с
Потеря пакетов 10 %	Базовый	99,81	115,52	25,81	0,260	0,0	14,1	0
	Предлагаемый	100,00	3,51	0,54	0,000	5,0	16,7	≈6,4
Задержка и джиттер	Базовый	100,00	107,41	13,61	0,111	0,0	14,3	0
	Предлагаемый	100,00	4,46	0,53	0,000	5,0	16,9	≈6,4
Микроразрывы связи	Базовый	94,96	1,79	36,55	1,052	0,0	13,8	0
	Предлагаемый	100,00	4,08	0,57	0,000	5,0	16,5	≈6,4

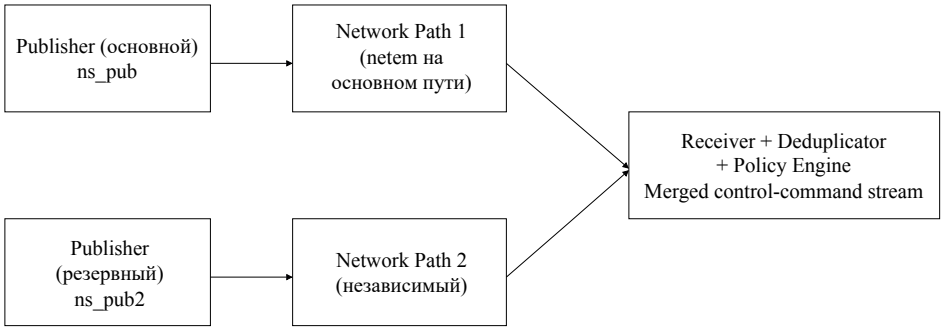

Рис. 3. Архитектура Proposed: два независимых пути доставки, объединение потоков и дедупликация по seq

Fig. 3. Proposed architecture: two independent delivery paths, stream merging, and sequence-based de-duplication

тах применяется параметр T_stable = 5 с (hold-down), поэтому ожидаемое $T\_rec \approx 5$ с при условии отсутствия повторной деградации как представлено на рис. 5.

На рис. 3 показана практическая реализация режима Proposed: сообщения управления публикуются по двум независимым траекториям доставки (primary и backup). На стороне получателя выполняется объединение потоков и дедупликация по номеру последовательности (seq), что позволяет компенсировать потери и микро-обрывы без изменения ядра ROS 2/DDS. Переход между траекториями инициируется политикой, формируемой на основе QoS-событий и мониторинга метрик.

На рис. 4 видно, что Proposed обеспечивает доставку команд DR_c = 100 % во всех сценариях и устраняет провалы управления ($T\_(blk, max) \to 0$ с). Существенное снижение $p95$ задержки L и джиттера J в сценариях Loss 10 % и Delay + Jitter подтверждает, что механизм резервирования и дедупликации повышает устойчивость контура управления к деградациям канала.

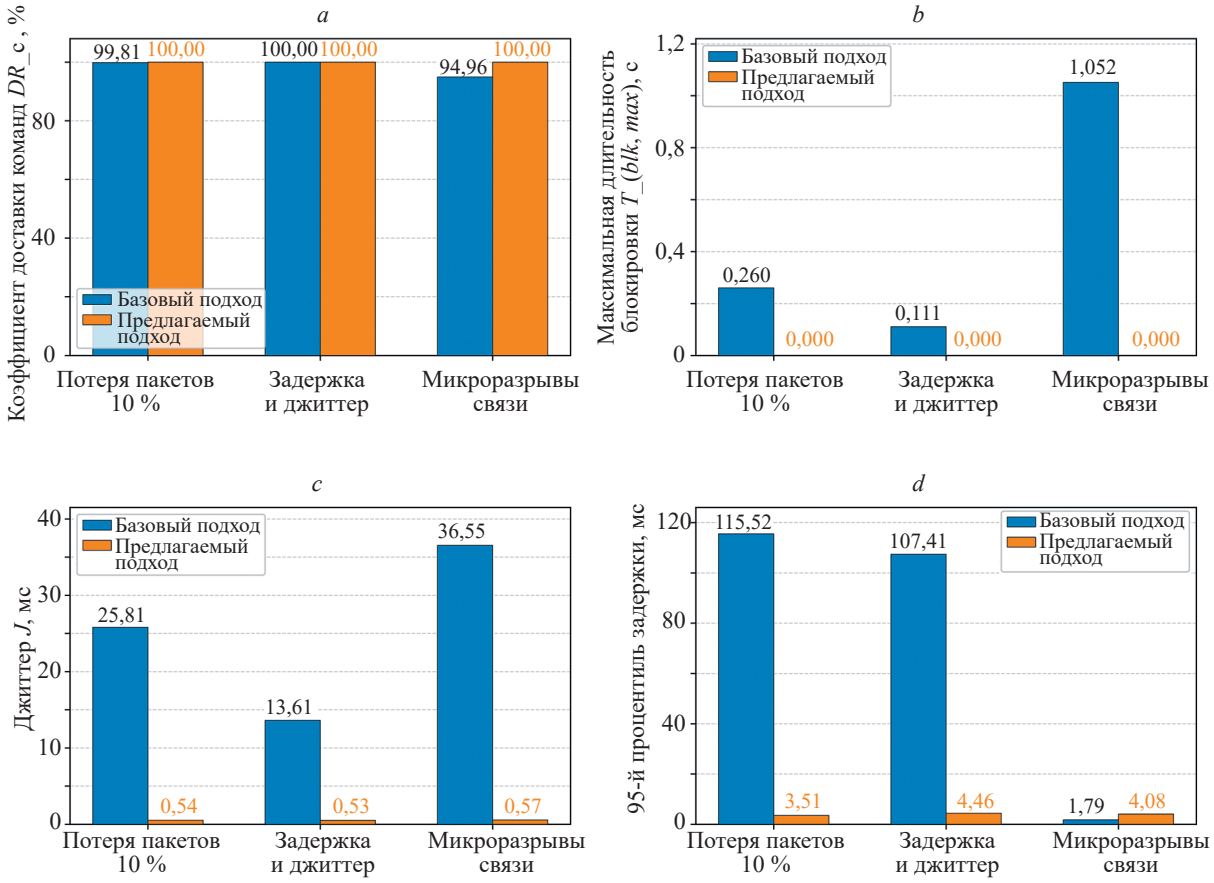

Рис. 4. Графическое сравнение сводных метрик базового и предлагаемого подхода: коэффициент доставки команд DR_c (*a*); максимальная длительность блокировки $T\_(blk, max)$ (*b*); джиттер J (*c*); 95-й процентиль задержки L (*d*)

Fig. 4. Visual comparison of the summary metrics for the baseline and proposed approaches: command delivery ratio DR_c (*a*); maximum blocking duration $T\_(blk, max)$ (*b*); jitter J (*c*); 95th-percentile of the latency L (*d*)

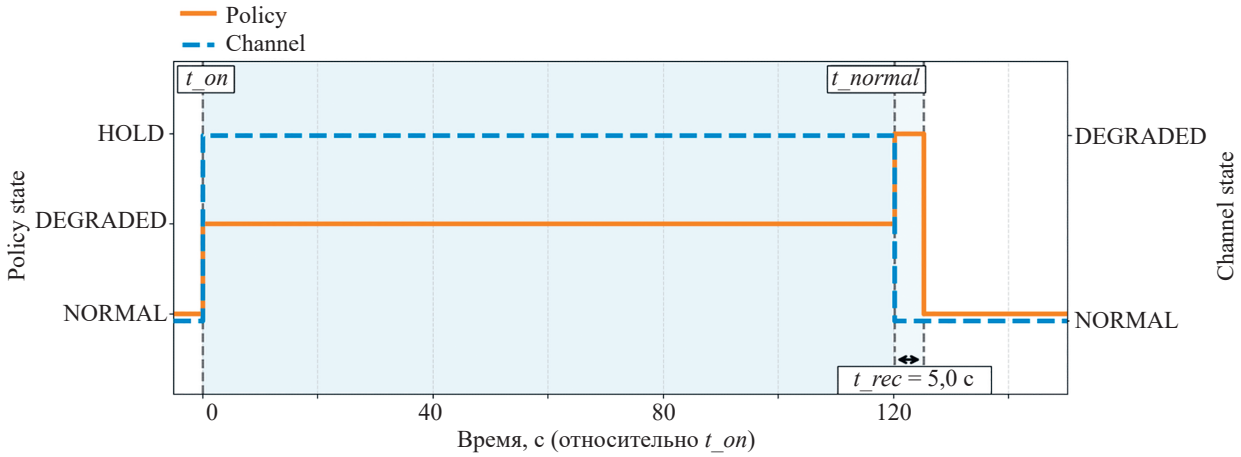

Рис. 5. Временная диаграмма определения времени восстановления *T_rec* в режиме Proposed

Fig. 5. Timing diagram for determining the recovery time *T_rec* in the Proposed mode

Обсуждение

В работе предложен DDS-aware слой отказоустойчивой коммуникации для ROS 2, который сочетает мониторинг QoS/метрик, политику переключения и прикладное резервирование доставки с объединением потоков и дедупликацией по seq [11, 12, 22]. Как отмечено в табл. 2, в Proposed обеспечена доставка команд *DR_c* = 100 % во всех сценариях, тогда как в Baseline при micro-outages *DR_c* снижается до 94,96 % [25, 26]. Максимальная блокировка управления *T_*(*blk*, *max*) в Proposed стремится к 0 с, что указывает на устранение «провалов» доставки при деградации канала [18, 19]. Фиксированное значение времени восстановления *T_rec* = 5 с было выбрано для предотвращения эффекта «дребезга» (flapping) [23, 27]. Относительная сложность предложенной архитектуры компенсируется тем, что основные операции инкапсулированы в отдельных модулях [10, 20].

Заключение

Предложенный подход не требует модификации ядра Robot Operating System 2 (ROS 2) и может быть интегрирован как надстройка на уровне узлов приложения. Экспериментальная валидация в изолированном тестбеде (Linux network namespace + tc netem) подтвердила рост устойчивости контура управления.

Использование событий Quality of Service (QoS) и оконных оценок метрик позволяет быстро фиксировать деградацию и запускать меры смягчения до того, как деградация приводит к длительным провалам управления. Разделение трафика на классы и адаптация телеметрии являются ключевыми для сохранения ограничений по задержке и непрерывности управления. Ограничением подхода является практическая невозможность бесшовно менять QoS «на лету» без переключения/пересоздания конечных точек; поэтому в реализации рекомендуется заранее подготовить несколько профилей QoS для ключевых потоков.

В дальнейшем планируется оптимизация алгоритмов дедупликации для снижения нагрузки на Central Processing Unit (CPU) и внедрение адаптивного механизма hold-down, позволяющего динамически изменять время *T_rec* в зависимости от интенсивности сетевых сбоев. Работа будет направлена на подтверждение результатов на реальных беспроводных каналах и сравнение поведения при использовании альтернативных Data Distribution Service (DDS) (например, Cyclone DDS) и иных профилей QoS. Практическая применимость ROS 2 и особенности его архитектуры систематизированы в обзорной работе, а оценка поведения ROS 2 в условиях потерь и деградации беспроводных сетей для беспилотных систем.

Литература

1. Kronauer T., Pohlmann J., Matthé M., Smejkal T., Fettweis G. Latency analysis of ROS2 Multi-Node systems // Proc. of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). 2021. P. 1–7. doi: 10.1109/MFI52462.2021.9591166
2. Macenski S., Foote T., Gerkey B., Lalancette C., Woodall W. Robot Operating System 2: Design, architecture, and uses in the wild // Science Robotics. 2022. V. 7. N 66. P. eabm6074. doi: 10.1126/scirobotics.abm6074
3. Maruyama Y., Kato S., Azumi T. Exploring the performance of ROS2 // Proc. of the 13th International Conference on Embedded Software. 2016. P. 1–10. doi: 10.1145/2968478.2968502

References

1. Kronauer T., Pohlmann J., Matthé M., Smejkal T., Fettweis G. Latency analysis of ROS2 Multi-Node systems. *Proc. of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021, pp. 1–7. doi: 10.1109/MFI52462.2021.9591166
2. Macenski S., Foote T., Gerkey B., Lalancette C., Woodall W. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 2022, vol. 7, no. 66. pp. eabm6074. doi: 10.1126/scirobotics.abm6074
3. Maruyama Y., Kato S., Azumi T. Exploring the performance of ROS2. *Proc. of the 13th International Conference on Embedded Software*, 2016, pp. 1–10. doi: 10.1145/2968478.2968502

4. Pardo-Castellote G. OMG data-distribution service: architectural overview // Proc. of the 23rd International Conference on Distributed Computing Systems Workshops. 2003. P. 200–206. doi: 10.1109/ICDCSW.2003.1203555
5. Blass T., Hamann A., Lange R., Ziegenbein D., Brandenburg B.B. Automatic latency management for ROS 2: benefits, challenges, and open problems // Proc. of the IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS). 2021. P. 264–277. doi: 10.1109/RTAS52030.2021.00029
6. Bédard C., Lütkebohle I., Dagenais M. ros2_tracing: Multipurpose low-overhead framework for real-time tracing of ROS 2 // IEEE Robotics and Automation Letters. 2022. V. 7. N 3. P. 6511–6518. doi: 10.1109/LRA.2022.3174346
7. Ye Y., Nie Z., Liu X.-J., Xie F., Li Z., Li P. ROS2 Real-time performance optimization and evaluation // Chinese Journal of Mechanical Engineering. 2023. V. 36. N 1. P. 144. doi: 10.1186/s10033-023-00976-5
8. Teper H., Betz T., von der Brüggen G., Chen K.-H., Betz J., Chen J.-J. Timing-Aware ROS 2 architecture and system optimization // Proc. of the IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). 2023. P. 206–215. doi: 10.1109/RTCSA58653.2023.00032
9. Casini D., Chen J.-J., Li J., Reghenzani F., Teper H. A Survey of real-time support, analysis, and advancements in ROS 2 // Leibniz Transactions on Embedded Systems. 2026. V. 11. N 1. P. 1:1–1:37. doi: 10.4230/LITES.11.1.1
10. Wang Z., Liu S., Ji D., Yi W. Improving real-time performance of Micro-ROS with priority-driven chain-aware scheduling // Electronics. 2024. V. 13. N 9. P. 1658. doi: 10.3390/electronics13091658
11. Sobhani H., Choi H., Kim H. Timing analysis and priority-driven enhancements of ROS 2 multi-threaded executors // Proc. of the IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS). 2023. P. 106–118. doi: 10.1109/RTAS58335.2023.00016
12. Zhang J., Yu X., Ha S., Peña Queralta J., Westerlund T. Comparison of middlewares in edge-to-edge and edge-to-cloud communication for distributed ROS 2 systems // Journal of Intelligent & Robotic Systems. 2024. V. 110. N 4. P. 162. doi: 10.1007/s10846-024-02187-z
13. Dey E., Walczak M., Anwar M.S., Roy N., Freeman J., Gregory T., et al. A Novel ROS2 QoS policy-enabled synchronizing middleware for co-simulation of heterogeneous multi-robot systems // Proc. of the 32nd International Conference on Computer Communications and Networks (ICCCN). 2023. P. 1–10. doi: 10.1109/ICCCN58024.2023.10230109
14. Hemminger S. Network emulation with NetEm // Proc. of the 6th National Linux Conference. 2005.
15. Thulasiraman P., Chen Z., Allen B., Bingham B. Evaluation of the Robot Operating System 2 in lossy unmanned networks // Proc. of the IEEE International Systems Conference (SysCon). 2020. P. 1–8. doi: 10.1109/SysCon47679.2020.9275849
16. Paul S., Lephuoc D., Hauswirth M. Performance evaluation of ROS2-DDS middleware implementations facilitating cooperative driving in autonomous vehicle // arXiv. 2024. arXiv:2412.07485. doi: 10.48550/arXiv.2412.07485
17. Gambo M.L., Danasabe A., Almadani B., Aliyu F., Aliyu A., Al-Nahari E. A Systematic literature review of DDS middleware in robotic systems // Robotics. 2025. V. 14. N 5. P. 63. doi: 10.3390/robotics14050063
18. Chovet L., Garcia G.M., Bera A., Richard A., Yoshida K., Olivares-Méndez M.A. Performance comparison of ROS2 middlewares for multi-robot mesh networks in planetary exploration // Journal of Intelligent & Robotic Systems. 2025. V. 111. N 1. P. 18. doi: 10.1007/s10846-024-02211-2
19. Lee S., Kim T., Chae J., Park K.-J. Optimizing ROS 2 communication for wireless robotic systems // arXiv. 2025. arXiv:2508.11366. doi: 10.48550/arXiv.2508.11366
20. Lee S., Park H.-S., Chae J., Park K.-J. Probabilistic latency analysis of the data distribution service in ROS 2 // arXiv. 2025. arXiv:2508.10413. doi: 10.48550/arXiv.2508.10413
21. Al-Batati A., Koubâa A., Gabr K., Abdelkader M., Aloqaily H. ROS 2 in a nutshell: a survey // ACM Computing Surveys. 2026. doi: 10.1145/3815113
22. Chen K., Hari K., Chung T., Wang M., Tian N., Juette C., et al. FogROS2-FT: Fault tolerant cloud robotics // Proc. of the IEEE/RSJ

4. Pardo-Castellote G. OMG data-distribution service: architectural overview. *Proc. of the 23rd International Conference on Distributed Computing Systems Workshops*, 2003, pp. 200–206. doi: 10.1109/ICDCSW.2003.1203555
5. Blass T., Hamann A., Lange R., Ziegenbein D., Brandenburg B.B. Automatic latency management for ROS 2: benefits, challenges, and open problems. *Proc. of the IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021, pp. 264–277. doi: 10.1109/RTAS52030.2021.00029
6. Bédard C., Lütkebohle I., Dagenais M. ros2_tracing: Multipurpose low-overhead framework for real-time tracing of ROS 2. *IEEE Robotics and Automation Letters*, 2022, vol. 7, no. 3, pp. 6511–6518. doi: 10.1109/LRA.2022.3174346
7. Ye Y., Nie Z., Liu X.-J., Xie F., Li Z., Li P. ROS2 Real-time performance optimization and evaluation. *Chinese Journal of Mechanical Engineering*, 2023, vol. 36, no. 1, pp. 144. doi: 10.1186/s10033-023-00976-5
8. Teper H., Betz T., von der Brüggen G., Chen K.-H., Betz J., Chen J.-J. Timing-Aware ROS 2 architecture and system optimization. *Proc. of the IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2023, pp. 206–215. doi: 10.1109/RTCSA58653.2023.00032
9. Casini D., Chen J.-J., Li J., Reghenzani F., Teper H. A Survey of real-time support, analysis, and advancements in ROS 2. *Leibniz Transactions on Embedded Systems*, 2026, vol. 11, no. 1, pp. 1:1–1:37. doi: 10.4230/LITES.11.1.1
10. Wang Z., Liu S., Ji D., Yi W. Improving real-time performance of Micro-ROS with priority-driven chain-aware scheduling. *Electronics*, 2024, vol. 13, no. 9, pp. 1658. doi: 10.3390/electronics13091658
11. Sobhani H., Choi H., Kim H. Timing analysis and priority-driven enhancements of ROS 2 multi-threaded executors. *Proc. of the IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2023, pp. 106–118. doi: 10.1109/RTAS58335.2023.00016
12. Zhang J., Yu X., Ha S., Peña Queralta J., Westerlund T. Comparison of middlewares in edge-to-edge and edge-to-cloud communication for distributed ROS 2 systems. *Journal of Intelligent & Robotic Systems*, 2024, vol. 110, no. 4, pp. 162. doi: 10.1007/s10846-024-02187-z
13. Dey E., Walczak M., Anwar M.S., Roy N., Freeman J., Gregory T., et al. A Novel ROS2 QoS policy-enabled synchronizing middleware for co-simulation of heterogeneous multi-robot systems. *Proc. of the 32nd International Conference on Computer Communications and Networks (ICCCN)*, 2023, pp. 1–10. doi: 10.1109/ICCCN58024.2023.10230109
14. Hemminger S. Network emulation with NetEm. *Proc. of the 6th National Linux Conference*, 2005.
15. Thulasiraman P., Chen Z., Allen B., Bingham B. Evaluation of the Robot Operating System 2 in lossy unmanned networks. *Proc. of the IEEE International Systems Conference (SysCon)*, 2020, pp. 1–8. doi: 10.1109/SysCon47679.2020.9275849
16. Paul S., Lephuoc D., Hauswirth M. Performance evaluation of ROS2-DDS middleware implementations facilitating cooperative driving in autonomous vehicle. *arXiv*, 2024. arXiv:2412.07485. doi: 10.48550/arXiv.2412.07485
17. Gambo M.L., Danasabe A., Almadani B., Aliyu F., Aliyu A., Al-Nahari E. A Systematic literature review of DDS middleware in robotic systems. *Robotics*, 2025, vol. 14, no. 5, pp. 63. doi: 10.3390/robotics14050063
18. Chovet L., Garcia G.M., Bera A., Richard A., Yoshida K., Olivares-Méndez M.A. Performance comparison of ROS2 middlewares for multi-robot mesh networks in planetary exploration. *Journal of Intelligent & Robotic Systems*, 2025, vol. 111, no. 1, pp. 18. doi: 10.1007/s10846-024-02211-2
19. Lee S., Kim T., Chae J., Park K.-J. Optimizing ROS 2 communication for wireless robotic systems. *arXiv*, 2025. arXiv:2508.11366. doi: 10.48550/arXiv.2508.11366
20. Lee S., Park H.-S., Chae J., Park K.-J. Probabilistic latency analysis of the data distribution service in ROS 2. *arXiv*, 2025. arXiv:2508.10413. doi: 10.48550/arXiv.2508.10413
21. Al-Batati A., Koubâa A., Gabr K., Abdelkader M., Aloqaily H. ROS 2 in a nutshell: a survey. *ACM Computing Surveys*, 2026. doi: 10.1145/3815113
22. Chen K., Hari K., Chung T., Wang M., Tian N., Juette C., et al. FogROS2-FT: Fault tolerant cloud robotics. *Proc. of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS). 2024. P. 1390–1397. doi: 10.1109/IROS58592.2024.10802613
23. Lauer M., Amy M., Fabre J.-C., Roy M., Excoffon W., Stoicescu M. Resilient computing on ROS using adaptive fault tolerance // Journal of Software: Evolution and Process. 2018. V. 30. N 3. P. e1917. doi: 10.1002/smr.1917
24. Keramat F., Peña Queralta J., Westerlund T. Partition-tolerant and Byzantine-tolerant decision making for distributed robotic systems with IOTA and ROS2 // IEEE Internet of Things Journal. 2023. V. 10. N 14. P. 12985–12998. doi: 10.1109/JIOT.2023.3257984
25. Ruzmetov A., Khudaybergenov T. Study on the implementation of reliable data transmission in wireless communication system // Proc. of the IEEE XVII International Scientific and Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE). 2025. P. 1–5. doi: 10.1109/APEIE66761.2025.11289272
26. Zhang J., Yu X., Westerlund T. Enhancing the resilience of ROS 2-based multi-robot systems with Kubernetes: a case study on UWB-Based relative positioning // Sensors. 2025. V. 25. N 16. P. 5067. doi: 10.3390/s25165067
27. Rossi F., Vaquero T., Sánchez Net M., Saboia da Silva M., Vander Hook J. The Pluggable Distributed Resource Allocator (PDRA): a middleware for distributed computing in mobile robotic networks // Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020. P. 4337–4344. doi: 10.1109/IROS45743.2020.9341205
28. Ruzmetov A., Khudaybergenov T. Survey of IoT application layer protocols // Modern Science and Research. 2024. V. 3. N 1. P. 1–5.
29. Рузметов А. А., Худайбергенов Т. А. Алгоритм переключения между системами передачи данных для обеспечения бесперебойной связи в робототехнике // Роль физико-математических наук в современном образовательном пространстве: материалы VII Международной научно-практической конференции, посвященной памяти Г.И. Имашева. Секция 3. Атырау: ASU Press, 2024. С. 271–275.
30. Valkov I., Trinder P., Chechina N. Reliable distribution of computational load in robot teams // Autonomous Robots. 2021. V. 45. N 3. P. 351–369. doi: 10.1007/s10514-021-09967-8

International Conference on Intelligent Robots and Systems (IROS), 2024, pp. 1390–1397. doi: 10.1109/IROS58592.2024.10802613
23. Lauer M., Amy M., Fabre J.-C., Roy M., Excoffon W., Stoicescu M. Resilient computing on ROS using adaptive fault tolerance. *Journal of Software: Evolution and Process*, 2018, vol. 30, no. 3, pp. e1917. doi: 10.1002/smr.1917
24. Keramat F., Peña Queralta J., Westerlund T. Partition-tolerant and Byzantine-tolerant decision making for distributed robotic systems with IOTA and ROS2. *IEEE Internet of Things Journal*, 2023, vol. 10, no. 14, pp. 12985–12998. doi: 10.1109/JIOT.2023.3257984
25. Ruzmetov A., Khudaybergenov T. Study on the implementation of reliable data transmission in wireless communication system. *Proc. of the IEEE XVII International Scientific and Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE)*, 2025, pp. 1–5. doi: 10.1109/APEIE66761.2025.11289272
26. Zhang J., Yu X., Westerlund T. Enhancing the resilience of ROS 2-based multi-robot systems with Kubernetes: a case study on UWB-Based relative positioning. *Sensors*, 2025, vol. 25, no. 16, pp. 5067. doi: 10.3390/s25165067
27. Rossi F., Vaquero T., Sánchez Net M., Saboia da Silva M., Vander Hook J. The Pluggable Distributed Resource Allocator (PDRA): a middleware for distributed computing in mobile robotic networks. *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4337–4344. doi: 10.1109/IROS45743.2020.9341205
28. Ruzmetov A., Khudaybergenov T. Survey of IoT application layer protocols. *Modern Science and Research*, 2024, vol. 3, no. 1, pp. 1–5.
29. Ruzmetov A.A., Khudaybergenov T.A. Algorithm for switching between data transmission systems to ensure uninterrupted communication in robotics. Proc. of the 7th International Scientific and Practical Conference "The Role of Physical and *Mathematical Sciences in the Modern Educational Space", dedicated to the memory of G.I. Imashev*, 2024, pp. 271–275. (in Russian)
30. Valkov I., Trinder P., Chechina N. Reliable distribution of computational load in robot teams. *Autonomous Robots*, 2021, vol. 45, no. 3, pp. 351–369. doi: 10.1007/s10514-021-09967-8

Авторы

Рузметов Артём Александрович — старший преподаватель, Ургенчский технологический университет RANCH, Ургенч, 220100, Узбекистан, SC 60008439800, https://orcid.org/0000-0002-3491-9256, ruzmetovartem@gmail.com

Худайбергенов Тимур Артурович — PhD, доцент, Ургенчский технологический университет RANCH, SC 57462873400, https://orcid.org/0000-0002-5958-582X, timartok@gmail.com

Authors

Artem A. Ruzmetov — Senior Lecturer, Urgench RANCH University of Technology, Urgench, 220100, Uzbekistan, SC 60008439800, https://orcid.org/0000-0002-3491-9256, ruzmetovartem@gmail.com

Timur A. Khudaybergenov — PhD, Associate Professor, Urgench RANCH University of Technology, Urgench, 220100, Uzbekistan, SC 57462873400, https://orcid.org/0000-0002-5958-582X, timartok@gmail.com

Статья поступила в редакцию 22.01.2026
Одобрена после рецензирования 16.06.2026
Принята к печати 26.07.2026

Received 22.01.2026
Approved after reviewing 16.06.2026
Accepted 26.07.2026