ROS2 Connect: A new ROS2 over WAN Solution

Daniel Schott[1], Lakshminarasimhan Srinivasan[1], Christian Herrmann[1] and Andreas Nüchter[1,2,3]

[1]Computer Science XVII - Robotics, Julius-Maximilians-Universität Würzburg, Germany
[2]ENSTA, U2IS (Visiting Chair), Institut Polytechnique de Paris, Palaiseau, France
[3]Zentrum für Telematik e.V., Würzburg, Germany

Abstract

The Robot Operating System 2 (ROS2) has become a widely adopted framework for the development of distributed robotic systems. However, its communication architecture, based on DDS and RTPS, relies on multicast discovery mechanisms that are typically unavailable in wide-area network (WAN) environments, making remote operation challenging. This work presents ROS2 Connect, a WebSocket-based communication framework that enables transparent and secure ROS2 interaction across routed networks without requiring modifications to network infrastructure or DDS configurations. The proposed client-server architecture supports bidirectional exchange of topics, services, actions, and system data while integrating authentication and access control mechanisms. Experimental evaluation over a real WAN connection demonstrates significantly lower latency, higher stability, and improved scalability compared to existing solutions, including DDS Router, rosbridge and Zenoh. Initial results show that ROS2 Connect provides a reliable foundation for teleoperation and distributed robotics applications over wide-area networks.

Keywords

ROS2, distributed robotics, wide-area networks, robot middleware, teleoperation, DDS

1. Introduction

Mobile robots have become increasingly common in distributed settings such as teleoperation, cloud robotics, and multi-site systems, where communication across wide-area networks (WANs) becomes essential. In this context, the Robot Operating System 2 (ROS2) has emerged as the de facto standard framework for the development, integration and coordination of heterogeneous robotic systems [1]. Despite its widespread adoption, ROS2 remains primarily designed for local-area network (LAN) deployments. Its communication layer, based on the Data Distribution Service (DDS) and the Real-Time Publish-Subscribe Protocol (RTPS), fundamentally relies on IP multicast for participant discovery and data exchange [1, 2, 3]. This design assumption assumes multicast-capable network infrastructure, which is commonly available in local-area networks but typically not enabled across routed WAN or Internet environments, where multicast requires explicit network-layer support [4]. Consequently, ROS2 deployments over WAN suffer from impaired automatic discovery, increased configuration complexity, and reduced scalability. In particular, DDS/RTPS discovery mechanisms, which depend on multicast-based peer discovery, introduce substantial challenges when operating across routed and heterogeneous networks. As a result, enabling efficient and stable ROS2 communication over WAN remains a non-trivial and largely unresolved problem.

To address these limitations, we propose a WebSocket-based communication framework for ROS2. Our approach consists of a client-server architecture implemented as a ROS2 package, allowing transparent bidirectional exchange of topics, services, actions, tf2 transforms, and time synchronization data between distributed systems. In addition to transport-layer mediation, the framework incorporates authentication and authorization mechanisms. This design enables controlled remote access to robotic systems over routed and heterogeneous network infrastructures, while preserving operational safety

8th International Workshop on Robotics Software Engineering (RoSE'26), June 01, 2026, Vienna, Austria
daniel.schott@uni-wuerzburg.de (D. Schott); srinivasan@informatik.uni-wuerzburg.de (L. Srinivasan); christian.herrmann@uni-wuerzburg.de (C. Herrmann); andreas.nuechter@uni-wuerzburg.de (A. Nüchter)
0009-0005-1926-9549 (D. Schott); 0009-0003-9648-5574 (L. Srinivasan); 0000-0003-3870-783X (A. Nüchter)

and system integrity. Experimental results demonstrate that ROS2 Connect exhibits lower latency while maintaining consistently low variability in comparison with existing solutions. The implementation is publicly available at https://github.com/JMUWRobotics/ROS2-Connect.

2. Existing Solutions

Existing solutions for enabling ROS2 communication across WANs are broadly categorized into network-layer approaches, DDS-level routing mechanisms, and alternative middleware frameworks. Network-layer solutions typically rely on virtual private networks (VPNs) to extend local-area connectivity across routed infrastructures, allowing native DDS discovery and communication to operate transparently. However, VPN deployment is often impractical in centrally managed environments, such as organizational networks, where users lack the privileges required to establish custom network tunnels. Consequently, VPN-based approaches are not considered further in this work. Middleware-level approaches modify DDS discovery behavior. Examples include vendor-specific mechanisms such as the centralized discovery server in eProsima's Fast DDS and static unicast peer configurations supported by Eclipse Foundation's Cyclone DDS [5, 6]. While these methods enable WAN communication, they introduce dependencies on specific DDS implementations and remain difficult to deploy across network address translation (NAT) boundaries, as these approaches rely on direct peer addressability and are not designed for traversal or dynamic endpoint negotiation. Another solution is eProsima's DDS Router, which forwards DDS/RTPS traffic over TCP/IP and supports relay-based NAT traversal [7]. Although this approach enables communication across complex network topologies, it introduces additional routing overhead and remains tightly coupled to DDS-specific infrastructure. Alternative frameworks operate outside the DDS ecosystem. The rosbridge interface enables WebSocket-based communication by serializing ROS messages into JSON [8], but it is commonly used for web-client integration and requires additional application-layer handling, while JSON serialization introduces significant communication overhead. Eclipse Zenoh provides a modern publish-subscribe protocol with an available ROS middleware (RMW) implementation and support for multiple transport mechanisms, including WebSockets and TCP/IP [9]. Similar to eProsima's DDS Router, it enables communication across complex network topologies and supports NAT traversal. However, Zenoh replaces DDS entirely and requires additional integration effort.

Despite the availability of these approaches, achieving transparent, secure, and performant ROS2 communication across WANs without requiring vendor-specific dependencies, network-layer modifications, or substantial integration effort remains an open challenge. In particular, existing solutions either rely on infrastructure-level adaptations, introduce tight coupling to specific DDS implementations, impose significant configuration overhead, or fail to provide integrated mechanisms for secure and controlled remote access. To address these limitations, this work proposes a vendor-independent communication framework that enables native ROS2 interaction across WAN environments without requiring modifications to underlying network infrastructure. The proposed approach leverages a WebSocket-based client-server architecture to mediate ROS2 communication while preserving transparency at the application level and providing integrated support for authentication and access restriction.

3. ROS2 Connect

ROS2 Connect is a communication framework designed to enable transparent and secure interaction between distributed ROS2 systems across WANs. The framework addresses the limitations of multicast-dependent DDS discovery by introducing a transport-layer mediation mechanism that decouples ROS2 communication from the underlying network topology. The proposed architecture is implemented as a ROS2 package that provides two complementary node types: a server node and a client node. The server node is deployed on a publicly reachable host within the robot's local ROS2 environment and communicates with local robotic components using the native DDS-based ROS2 middleware. The client node, running on a remote operator system, establishes a persistent WebSocket connection to the server

node across the WAN. Through this architecture, ROS2 communication data (topics, services, actions) is transparently tunneled between server and client nodes. The nodes bridge between DDS-based local communication and WebSocket-based wide-area transport, enabling remote interaction without requiring multicast support, VPN infrastructure, or vendor-specific DDS configuration. WebSockets are employed as the underlying transport protocol due to their compatibility with firewall and NAT environments, and ability to provide reliable, bidirectional communication over standard TCP/IP networks. In addition, WebSocket-based communication has previously been successfully applied in practical teleoperation systems for mobile robots, demonstrating its suitability for latency-sensitive remote control scenarios [10].

3.1. Communication Model

ROS2 Connect mediates communication between distributed ROS2 systems by explicitly managing the exchange of ROS interfaces across a client-server WebSocket connection. Rather than relying on DDS multicast discovery, the framework employs a configuration-driven model in which both server and client nodes define the set of topics, services, actions, and their associated QoS profiles to be relayed through ROS parameters. This approach ensures that only explicitly authorized interfaces are exposed across the wide-area link. For topic-based communication, ROS2 Connect utilizes generic subscription and publication mechanisms provided by `rclcpp::GenericSubscription` and `rclcpp::GenericPublisher`. These interfaces allow the framework to operate independently of compile-time message definitions by handling serialized binary message data at runtime. Upon initialization, each node creates generic subscriptions and publishers for the configured topics, enabling the transparent forwarding of arbitrary ROS message types. For transport, a lightweight protocol header containing topic identifiers and compression metadata is prepended to each message before transmission over the WebSocket connection. This enables optional per-topic data compression to reduce bandwidth usage while allowing efficient unpacking and republishing on the receiving side. To minimize bandwidth consumption and reduce processing, data forwarding is subscription-aware. Messages are transmitted only when a corresponding subscriber exists on the remote system, thereby limiting traffic to actively used interfaces and reducing load on both the DDS middleware and the WebSocket transport layer. In addition to generic topic forwarding, ROS2 Connect provides dedicated support for core ROS2 infrastructure mechanisms required for distributed operation. The framework transparently relays the transform tree by forwarding the standard `/tf` and `/tf_static` topics, ensuring consistent spatial relationships between robot and operator systems. Furthermore, time synchronization is supported by propagating the robot's ROS time through the `/clock` topic. Service and action interfaces are handled through a plugin-based mechanism due to the lack of fully generic runtime support in `rclcpp`. Type-specific service and action client and server implementations are provided as ROS2 Connect plugins, which are dynamically loaded using the ROS2 plugin infrastructure (pluginlib). This design preserves the generic architecture of the framework while enabling transparent mirroring of selected services and actions between distributed systems. Through this combination of explicit interface mediation, runtime-generic message handling, and subscription-aware routing, ROS2 Connect replaces multicast-dependent discovery with a secure communication model suitable for wide-area deployments.

3.2. Security and Access Control

ROS2 Connect incorporates authentication and access control mechanisms to enable secure remote interaction with robotic systems. Authentication is performed at the application layer following the establishment of a WebSocket connection. Each client provides a user-specific authentication token configured through ROS parameters. Upon connection, the server initially suppresses all data exchange and processes only authentication messages. After successful verification of the provided credentials, the connection transitions into operational mode and regular communication is permitted. To maintain flexibility and application independence, the authentication procedure is implemented through a plugin-based architecture. This allows system integrators to define custom authentication strategies, such

as key-based validation or external identity management integration, without modifying the core framework. Authorization is enforced through the configuration-driven interface mediation model. All topics, services, and actions that are relayed across the wide-area connection must be explicitly specified in advance, including their associated message types. The server only forwards data for these predefined interfaces and rejects any attempts to access or publish to unspecified topics, services, or actions. This approach ensures that clients cannot access arbitrary system data or inject unauthorized messages. Together, these mechanisms integrate security directly into the communication framework, enabling controlled remote operation while preserving the transparency of standard ROS 2 interfaces.

4. Evaluation

Experiments were conducted over a real WAN connection between a residential client system and a university-hosted server. The server was accessible through a publicly reachable Apache reverse proxy deployed within the university network infrastructure. Network bandwidth measurements using *iperf3* indicate an available uplink capacity of approximately 18 Mbit/s and a downlink capacity of 58 Mbit/s. All experiments were performed using ROS2 Jazzy Jalisco on Ubuntu 24.04. To evaluate communication performance, round-trip times (RTT) were measured for serialized ROS messages with payload sizes ranging from 12 B to 500 kB. The selected range reflects common robotic workloads from small scalar sensor values and control commands to medium-sized LiDAR scan data and larger payloads such as camera images. For each configuration, 1000 independent measurements were recorded to obtain statistically robust averages. Additional experiments evaluated performance under concurrent load by transmitting between one and ten parallel topics. For each configuration, 1000 round-trip measurements were recorded for message sizes ranging from 12 B to 100 kB. The proposed ROS2 Connect framework was compared against three existing approaches: eProsima's DDS Router, the rosbridge WebSocket interface, and Eclipse Zenoh. All systems were configured to operate under identical network conditions.

Figure 1 shows the average round-trip latency for increasing message sizes under sequential transmission. Across all tested configurations, ROS2 Connect consistently achieves the lowest latency compared to DDS Router, rosbridge and Zenoh. For small message sizes, all four exhibit relatively stable transmission times; however, significant differences emerge as message sizes increase. ROS2 Connect maintains near-constant latency growth over a wide range of payload sizes, indicating efficient and predictable communication behavior. In contrast, DDS Router shows earlier performance degradation, with latency increasing more rapidly for larger messages. Furthermore, DDS Router was unable to reliably transmit very large payloads beyond certain sizes in the evaluated configuration. The rosbridge interface exhibits the highest latency through all measurements due to the overhead introduced by

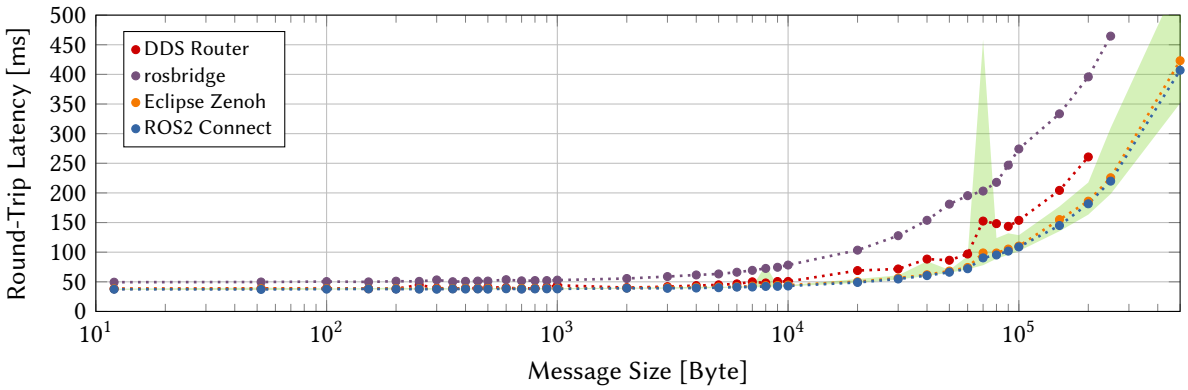

Figure 1: Average round-trip latency over a WAN for increasing message sizes. Each data point represents the mean of 1000 measurements. Large markers denote measured data points, while small markers indicate interpolated trends. The shaded region indicates the observed latency range for ROS2 Connect.

JSON serialization. Eclipse Zenoh exhibited overall latencies close to those of ROS2 Connect, differing by approximately 1ms for smaller and up to 16ms for larger payloads, with ROS2 Connect consistently achieving lower latency. In addition to achieving the lowest average latency, ROS2 Connect demonstrates substantially lower variability in round-trip times compared to DDS Router and rosbridge, while Zenoh exhibited comparable variability. The narrow latency range observed across most message sizes indicates stable communication behavior, which is particularly important for teleoperation scenarios requiring predictable timing characteristics. Overall, the results confirm that ROS2 Connect provides both improved transmission efficiency and higher latency stability compared to existing ROS2 over WAN solutions. Figure 2 shows the average round-trip latency of ROS2 Connect under concurrent load for increasing number of parallel topics. For small message sizes, latency remains nearly constant even when up to ten parallel topics are transmitted simultaneously, indicating that ROS2 Connect efficiently handles concurrent communication with minimal overhead. For larger message sizes, latency increases approximately linearly with the number of parallel topics. This behavior reflects both the limitations of available network bandwidth and the characteristics of WebSocket transport, which relies on a single ordered TCP stream and therefore transmits messages sequentially. As a result, large payloads can temporarily occupy the communication channel, increasing latency for concurrent transmissions. Despite this effect, communication remains stable and predictable across all tested configurations. These results demonstrate that ROS2 Connect scales effectively under concurrent load, with small control and sensor messages remaining largely unaffected while larger messages primarily reflect transport-layer constraints inherent to wide-area communication.
Due to space constraints, only representative results are presented here. A more extensive evaluation, including additional performance analyses and implementation details, is provided in the author's accompanying master's thesis [11].

5. Conclusions

This work presented ROS2 Connect, a WebSocket-based communication framework designed to enable secure and reliable ROS2 interactions across WANs. By decoupling ROS2 communication from multicast-dependent DDS discovery, the proposed approach enables transparent transmissions of topics, services, actions, and system infrastructure data without requiring network-layer modifications or vendor-specific configurations. Experimental evaluation demonstrated that ROS2 Connect achieves lower latency, higher stability, and better scalability under the evaluated concurrent load compared to existing ROS2 over WAN solutions, including DDS Router, rosbridge, and Zenoh. The results further showed that communication performance remains predictable across a wide range of message sizes and parallel workloads, making the framework well suited for teleoperation scenarios. Further work will focus on

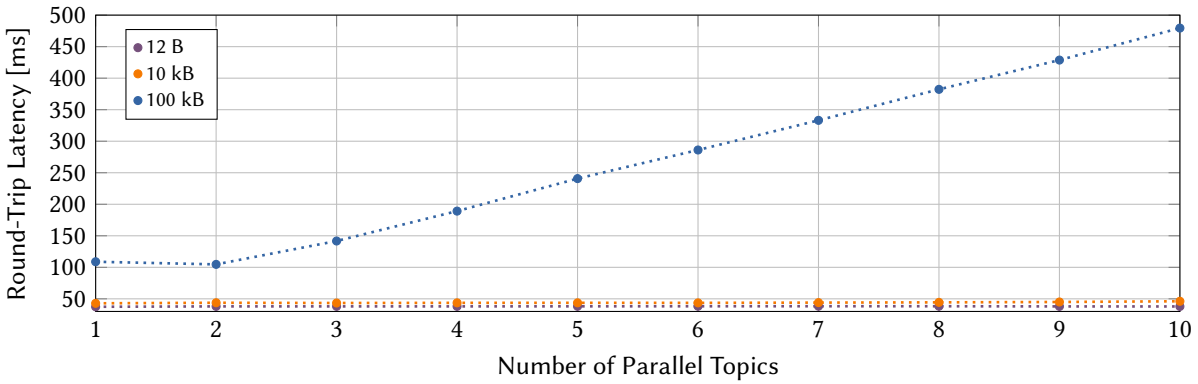

Figure 2: Average round-trip latency of ROS2 Connect under concurrent load for increasing numbers of parallel topics. Each data point represents the mean of 1000 measurements.

extending generic support for services and actions, evaluating in real robotic deployments with full application stacks, and exploring adaptive transport optimizations under varying network conditions.

Acknowledgments

The authors gratefully acknowledge the Virtual University of Bavaria (VHB 24-II-02-13-Nue1) for its continuous support since 2005 and for the grant that partially funded the implementation presented in this work.

Declaration on Generative AI

During the preparation of this work, the authors used DeepL Write and ChatGPT 5.2 in order to: Grammar and spelling check, Paraphrase and reword. After using these tools/services, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, Robot Operating System 2: Design, architecture, and uses in the wild, Science Robotics 7 (2022) eabm6074. doi:10.1126/scirobotics.abm6074.

[2] Object Management Group, Data Distribution Service (DDS), Version 1.4, Technical Report, Object Management Group, 9C Medway Road, PMB 274 Milford, MA 01757 USA, 2015. URL: https://www.omg.org/spec/DDS/1.4/PDF.

[3] Object Management Group, The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification, Version 2.2, Technical Report, Object Management Group, 9C Medway Road, Milford, MA 01757 USA, 2014. URL: https://www.omg.org/spec/DDSI-RTPS/2.2/PDF.

[4] C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen, Deployment issues for the ip multicast service and architecture, IEEE Network 14 (2000) 78–88. doi:10.1109/65.819174.

[5] eProsima, Fast DDS Documentation, Release 3.4.2, Technical Report, eProsima, Plaza de la Encina 10-11 Nucleo 4 2ª Planta, 28760 Tres Cantos, Madrid, Spanien, 2026. URL: https://fast-dds.docs.eprosima.com/_/downloads/en/v3.4.2/pdf/.

[6] Eclipse Foundation, Cyclone DDS, Version 0.10.5, 2024. URL: https://cyclonedds.io/docs/cyclonedds/0.10.5/.

[7] eProsima, DDS Router Documentation, Release 3.4.0, Technical Report, eProsima, Plaza de la Encina 10-11 Nucleo 4 2ª Planta, 28760 Tres Cantos, Madrid, Spanien, 2025. URL: https://eprosima-dds-router.readthedocs.io/_/downloads/en/v3.4.0/pdf/.

[8] Robot Web Tools, rosbridge_suite, 2026. URL: https://github.com/RobotWebTools/rosbridge_suite, last accessed: February 16, 2026.

[9] Eclipse Foundation, Zenoh, 2026. URL: https://zenoh.io/, last accessed: February 16, 2026.

[10] L. Srinivasan, J. Scharnagl, Z. Xu, N. Faerber, D. K. Babu, K. Schilling, Design and Development of a Robotic Teleoperation System using Duplex WebSockets suitable for Variable Bandwidth Networks, IFAC Proceedings Volumes 46 (2013) 57–61. 3rd IFAC Symposium on Telematics Applications.

[11] D. Schott, ROS2 Over Wide-Area-Networks: Teleoperation of Mobile Robots in an E-Learning Environment, Master's thesis, Julius-Maximilians-University, Würzburg, Bavaria, Germany, 2025. URL: https://robotik.informatik.uni-wuerzburg.de/telematics/download/MSc_Daniel_Schott.pdf.

A. Source Code

ROS2 Connect is released under the Mozilla Public License Version 2.0 and is available at: https://github.com/JMUWRobotics/ROS2-Connect