High Communication Performance Internet of Robotic Things Systems Based on RT-Middleware

Yutaka Watanobe

The 11th Robotics Symposium of the University of Aizu, May 22, 2026.

This presentation is based on the following published paper:

Daishi Yoshino, Yutaka Watanobe, Keitaro Naruse

F-Design Inc. | The University of Aizu

IEEE Access, Vol. 13, 207527 – 207540, 2025 • DOI: 10.1109/ACCESS.2025.3640266

Outline

01 **Background & Motivation**
Why IoRT?
What are the challenges?

02 **Problem Statement**
IoT bridges and their overhead

03 **Proposed Approach**
MQTT interface integrated into RT-Middleware

04 **Implementation**
MQTT communication module design

05 **Performance Evaluation**
Test setup, RTT measurements, results

06 **Conclusion & Future Work**
Findings and next steps

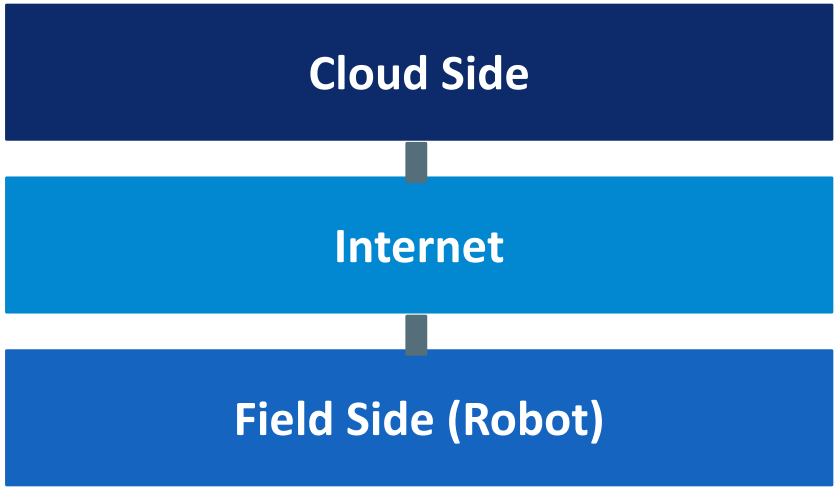
Background & Motivation

What is IoRT?

Internet of Robotic Things (IoRT) connects robots to the Internet to deliver cloud computing, remote control, and inter-robot communication services.

The Challenge

① IoRT spans multiple layers: robots, edge, cloud
② Each layer uses different middleware & protocols
③ Data conversion bridges are needed between layers
④ Bridges add latency and reduce real-time capability



Goal: Eliminate bridges — build the entire IoRT system within a single middleware framework.

Problem: IoT Bridges in ROS-Based Systems

ROS1 IoRT — requires MQTT bridge

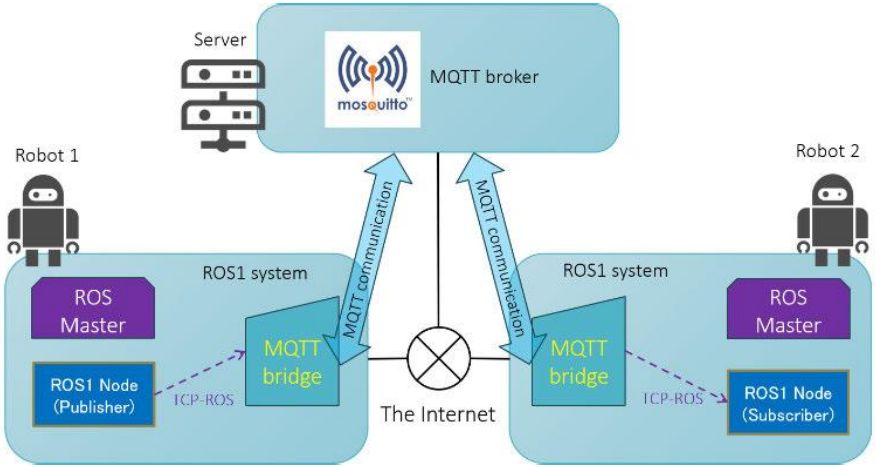


FIGURE 1. Overview of the IoRT system using ROS1

ROS2 IoRT — also requires MQTT bridge

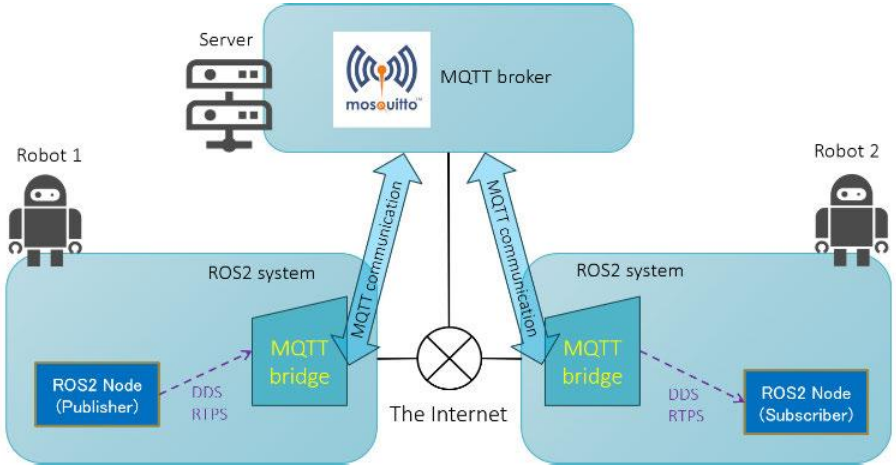


FIGURE 2. Overview of the IoRT system using ROS2

Key Issues with Bridge-Based IoRT:

- **ROS1:** Fixed TCPROS protocol; ROS Master hidden behind NAT → bridge mandatory.
- **ROS2:** DDS uses UDP multicast → NAT traversal issues → bridge mandatory.
- **Result:** Inter-process communication (IPC) between ROS nodes and IoT bridge adds latency.

RT-Middleware (OpenRTM-aist) Architecture

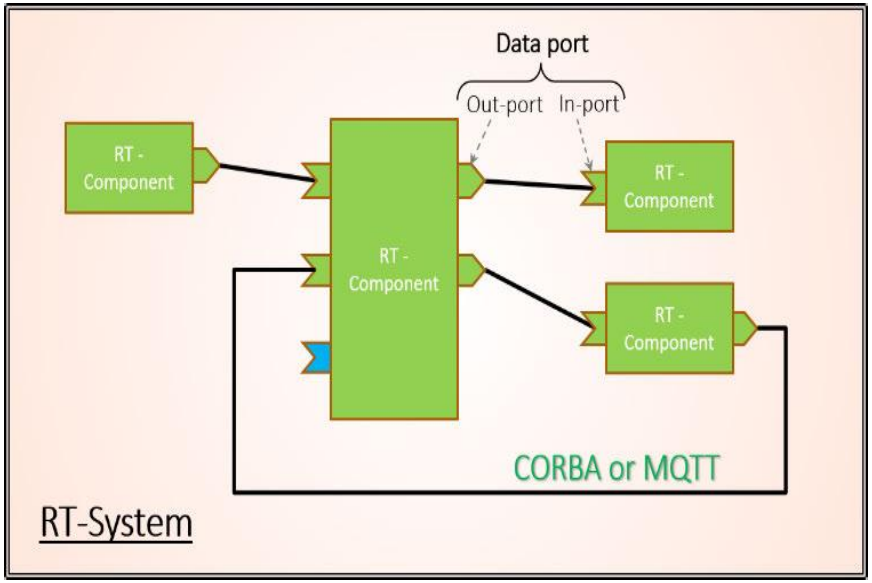


Fig. 3 — RT-System built with RT-Middleware

RT-System

Composed of RT-Components (RTCs) — reusable functional units.

Data Ports

Out-port (sender) + In-port (receiver) via plug-in interfaces.

CORBA (default)

Distributed object protocol — but blocked by NAT.

Plug-in Extension

Communication interface swappable via plug-in; no source change needed.

Key Advantage

MQTT interface → each RTC speaks IoT directly — NO bridge.

Proposed: MQTT Communication Interface for RT-Middleware

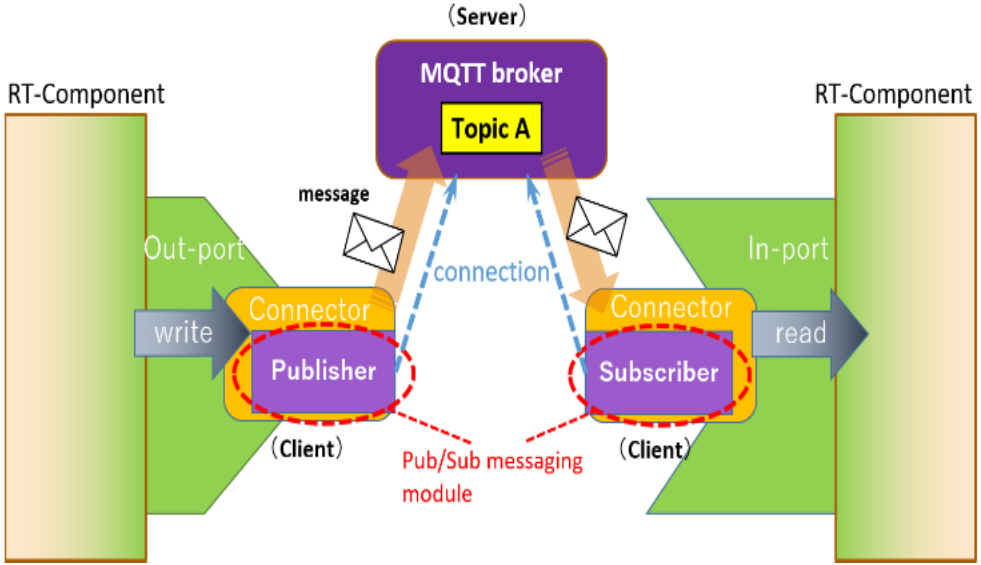


Fig. 4 — MQTT messaging interface architecture

Implementation Details

Protocol:	MQTT (libmosquittopp v1.6.9)
Out-port module:	Publisher client — invokes mosquitto_pub()
In-port module:	Subscriber client — callback on_message()
Data format:	CDR serialized byte array (same as CORBA path)
Config:	MQTT properties set via RT-Middleware environment
Broker:	Operates independently; configured separately
No bridge RTCs:	Bridging logic is inside the port module itself

Result: IoRT system built entirely within RT-Middleware — zero bridge overhead.

Performance Evaluation — Test Setup

ROS-Based IoRT System (with IoT bridge)

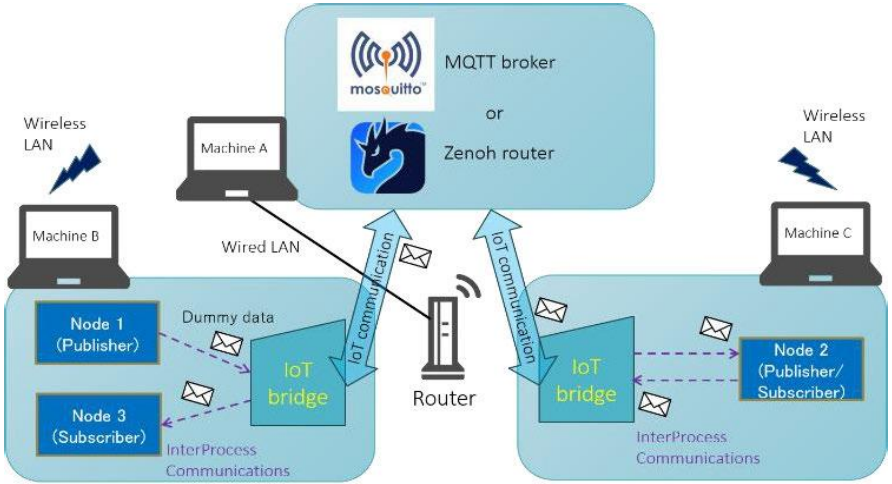


Fig. 5 — ROS test system: IPC between nodes and IoT bridge

RT-Middleware IoRT System (bridge-free)

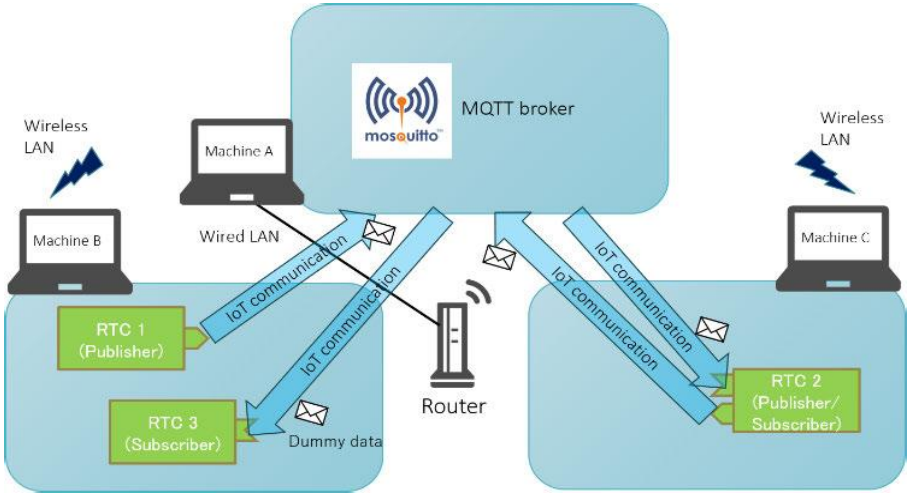


Fig. 6 — RT-Middleware: RTCs connect directly to MQTT broker

Setup: 3 PCs (Ubuntu 20.04), 1 router (5 GHz Wi-Fi). **Middlewares:** ROS1 Noetic, ROS2 Foxy, OpenRTM-aist 2.0.1. **Metric:** Round-trip time (RTT) × 1,000 trials @ 1 Hz. **Payloads:** 8 bytes – 128 KB (15 sizes; small/medium/large).

5 configurations tested:
① ROS1+MQTT ② ROS1+Zenoh ③ ROS2+MQTT (double bridge) ④ ROS2+Zenoh ⑤ **RT-Middleware+MQTT (proposed)**

Test Configurations — IoT Bridge Layouts

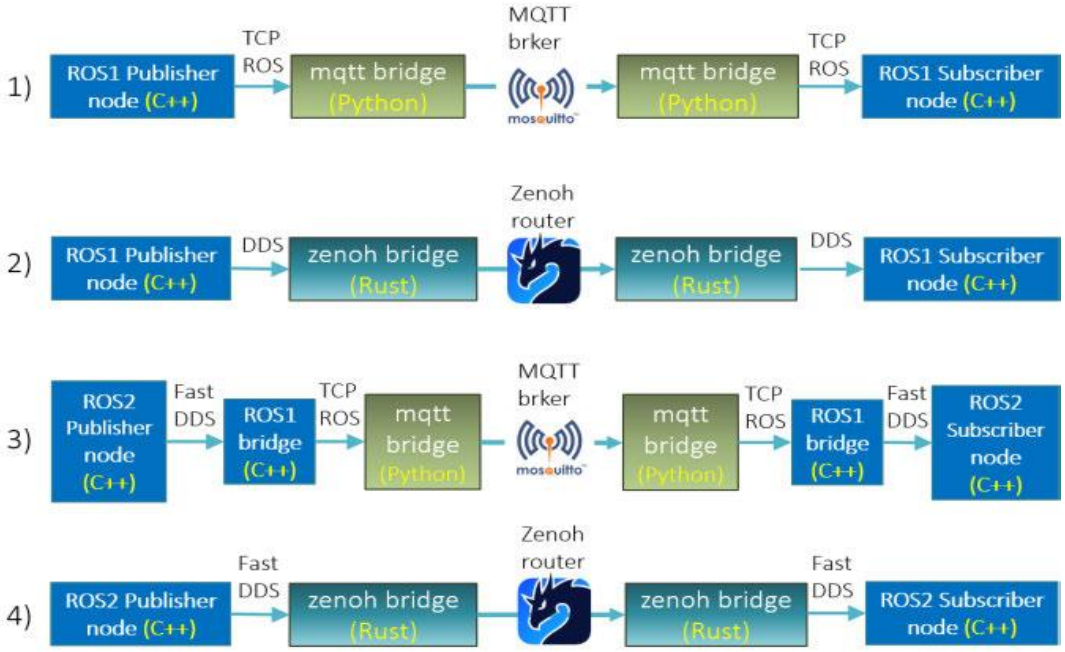


Fig. 7 — ROS-based configurations (1–4): outward communication path

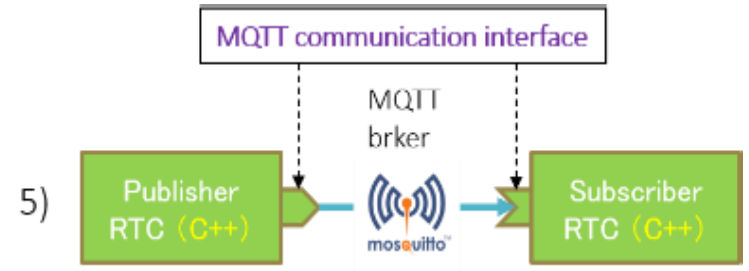


Fig. 8 — Config ⑤: RT-Middleware, no bridge

① ROS1 + MQTT bridge (Python)

② ROS1 + Zenoh bridge (Rust)

③ ROS2 + MQTT bridge (double bridge: ROS1 bridge + MQTT)

④ ROS2 + Zenoh bridge (Rust)

⑤ RT-Middleware + MQTT interface ← Proposed

Results — Intra-Machine IPC Latency (Small Payloads)

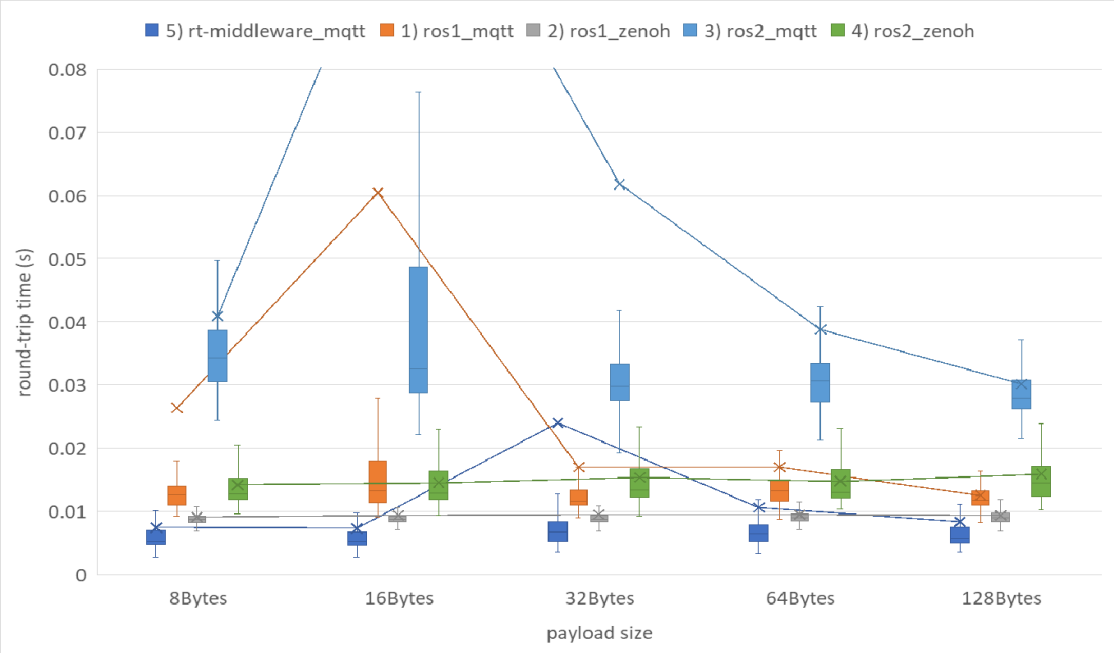


Fig. 9(a) — RTT inside Machine B: ROS node ↔ IoT bridge (8–128 bytes)

Key Findings

ROS1 ↔ IoT bridge

~2–3 ms stable; Python bridge slightly slower than C++.

ROS2 node-to-node

>1 ms higher than ROS1 (default QoS reliable policy).

ROS2 + double bridge (③)

Highest IPC cost — >3 ms vs ROS2 alone due to bridge conversion.

RT-Middleware ⑤

Lowest — no IPC between nodes and IoT layer at all.

Results — System-Level RTT: Medium Payloads (256 B – 4 KB)

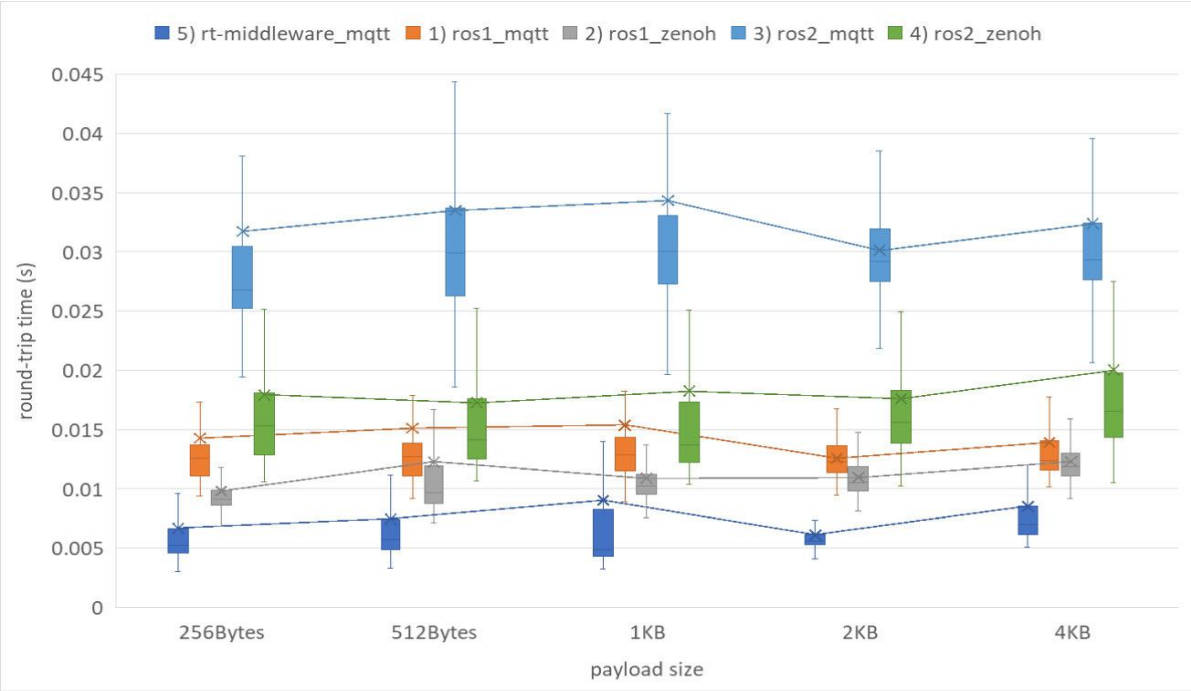


Fig. 10(b) — System RTT for medium payloads

Observations

⑤ RT-Middleware
Consistently lowest median RTT across all medium sizes.

② ROS1 + Zenoh
2nd best; ~3–5 ms higher than ⑤; stable IQR.

① ROS1 + MQTT
Moderate; IPC accounts for ~22–26% of total RTT.

④ ROS2 + Zenoh
Similar to ① for small/medium; IQR widens for large.

③ ROS2 + MQTT
>20 ms above ⑤ — double bridge amplifies IPC cost.

Results — System-Level RTT: Large Payloads (8 KB – 128 KB)

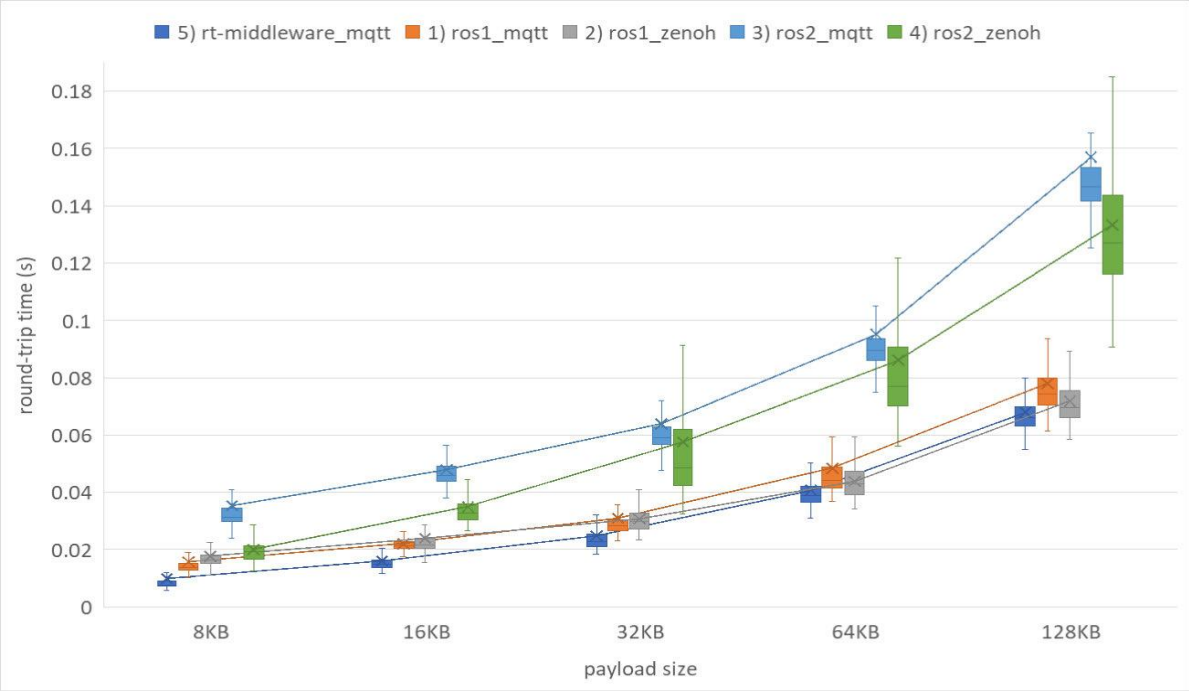


Fig. 10(c) — System RTT for large payloads

Large Payload Analysis

All configs
RTT grows with payload — network bandwidth is the bottleneck.

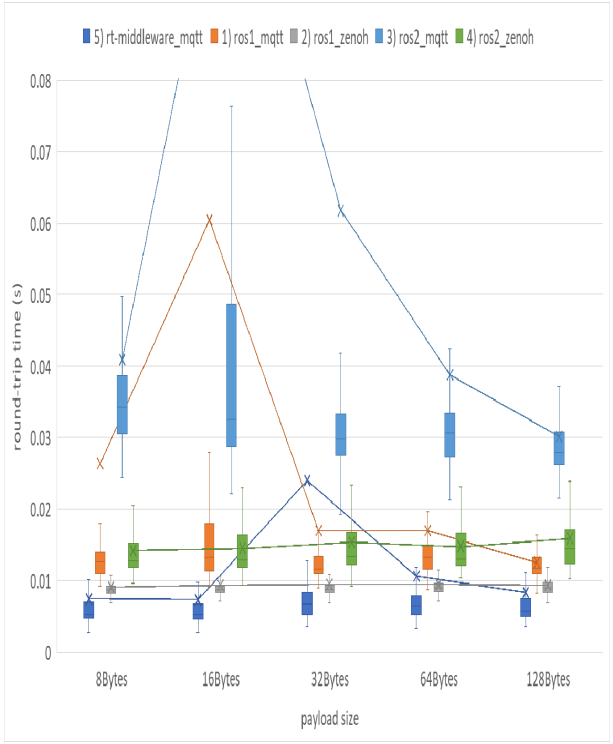
⑤ RT-Middleware
Still lowest — bridge-free design scales well.

③ ROS2+MQTT
Remains highest (~150+ ms @ 128 KB) — double-bridge cost persistent.

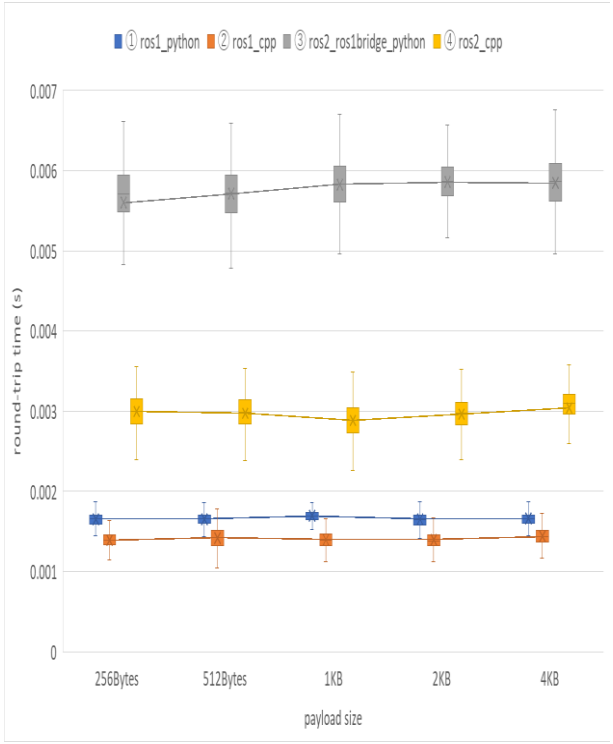
④ ROS2+Zenoh
IQR expands sharply (8–128 KB) — Zenoh router receive buffer (64 KB default) saturates.

② ROS1+Zenoh
Similar degradation at large sizes — same buffer issue.

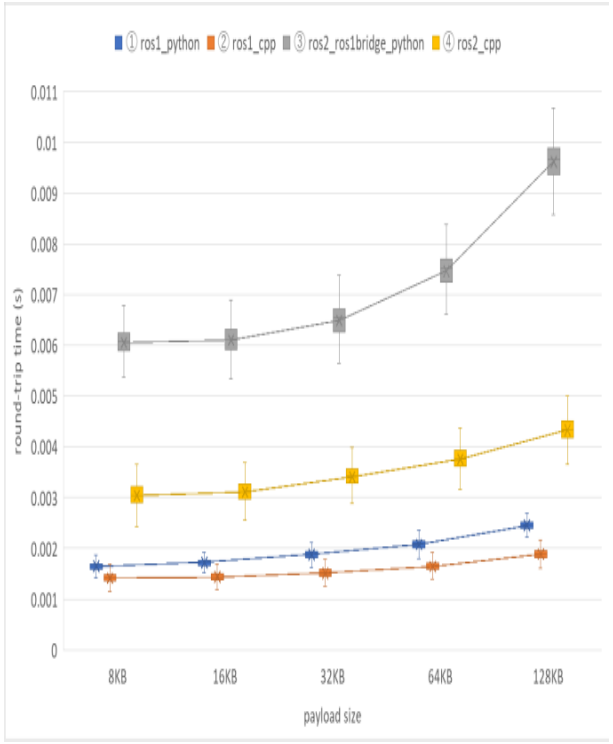
Results — Intra-Machine IPC RTT (Fig. 9: All Payload Sizes)



(a) Small payloads (8–128 B)



(b) Medium payloads (256 B–4 KB)



(c) Large payloads (8–128 KB)

Fig. 9 — RTT for intra-machine IPC (ROS node ↔ IoT bridge) on Machine B. **Config ③ (ROS2+double bridge)** shows highest IPC cost across all payload sizes. **RT-Middleware ⑤: IPC = 0% — no bridge.**

Conclusion & Future Work

✓ Bridge-free IoRT is achievable

Integrating MQTT directly into RT-Middleware data ports eliminates the need for any IoT bridge.

✓ Lower latency confirmed

RT-Middleware with MQTT achieved the lowest system RTT across all payload sizes compared to ROS1/ROS2 bridge-based systems.

✓ Bridge overhead quantified

For small/medium payloads, intra-machine IPC (bridge overhead) accounts for up to 40% of total round-trip time.

✓ Better real-time support

Eliminating bridge inter-process communication significantly improves real-time performance.

Future Work:

① Zenoh interface for RT-Middleware
② Throughput & reliability evaluation
③ Secure communication & QoS tuning
④ Real-world Internet deployment

Thank you

Paper:

Daishi Yoshino, Yutaka Watanobe, Keitaro Naruse F-Design Inc. | The University of Aizu

"High Communication Performance IoRT Systems Based on RT-Middleware"

IEEE Access, Vol. 13, 207527 – 207540, 2025 • DOI: 10.1109/ACCESS.2025.3640266