Poster: Exploring virtio-msg for Inter-OS Communication in RTOS-hosted Linux Environments on a Single MCU

Yoshifumi Shu[1] Yutaka Matsubara[1] Ryoma Hiraoka[2] Ryosuke Yamamoto[3] Atsushi Harada[2]
Keisuke Horii[2] Hiroki Masuda[3] Shinya Honda[1] Hiroaki Takada[1]

[1]Graduate School of Informatics, Nagoya University, Nagoya, Aichi, Japan
[2]Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa, Japan
[3]Engineering and Development Center, Mitsubishi Electric Corporation, Yokohama, Kanagawa, Japan

Contact: shu.yoshifumi@ertl.jp

Abstract

In Internet-of-Things (IoT) applications, co-running a real-time operating system (RTOS) and a general-purpose operating system (GPOS) on a single microcontroller unit (MCU) is a promising way to reduce the development cost of feature-rich software while meeting requirements for SWaP-C (Size, Weight, Power, and Cost), real-time performance, and reliability. EUKL, our ongoing work, is a NoMMU-Linux-based unikernel that runs as an RTOS task. It enables a NoMMU Linux environment to co-run with the host RTOS on a single MCU.

In this paper, we explore virtio-msg for inter-OS communication between the host RTOS and the RTOS-hosted Linux environment provided by EUKL so that RTOS-side and Linux-side software can cooperate closely. Our current prototype implementation, a WireGuard-based application, demonstrates that Linux network stacks can be made available to RTOS-based applications via a virtio-net interface.

CCS Concepts

• **Computer systems organization → Embedded software**.

Keywords

microcontroller unit, inter-OS communication, virtio-msg

1 Background and Motivation

Internet-of-Things (IoT) applications often impose stringent SWaP-C (Size, Weight, Power, and Cost) constraints while requiring real-time performance, reliability, and increasingly rich functionality. Microcontroller units (MCUs) are attractive platforms for such applications, but RTOS and bare-metal environments have fewer feature-rich software assets than general-purpose operating system (GPOS) environments such as Linux, increasing development costs.

One promising way to reduce this cost is to co-run an RTOS and a GPOS on a single MCU. EUKL[2], another ongoing work by the authors, is a NoMMU-Linux-based unikernel for MCUs. By running EUKL as an isolated RTOS user-space task, EUKL enables the RTOS and a NoMMU Linux environment to co-run on a single MCU without using an MMU, while preserving the real-time capability and reliability of RTOS-side software.

In such a dual-OS environment, inter-OS communication becomes essential for combining RTOS-side software and Linux-side software. There are two requirements for making inter-OS communication practical. (1) The communication must incur low overhead. (2) The interface must be well abstracted because Linux-side feature-rich software typically assumes high-level device and system interfaces.

However, conventional inter-OS communication mechanisms applicable to this RTOS-hosted Linux environment do not simultaneously satisfy these requirements. A straightforward way to connect RTOS-side software and EUKL is to directly use the inter-task communication primitives natively provided by the RTOS. Although this approach can be efficient, it exposes only low-level APIs to Linux and requires application-specific modifications to Linux-side software.

Another possible direction is to use virtio, which provides a more convenient abstraction. It exposes device interfaces already supported by Linux, allowing Linux-side software stacks to use the inter-OS communication path with little modification. However, existing virtio transport layers are not well suited to MCUs. virtio-pci assumes PCI support, which MCUs usually lack. virtio-mmio often relies on MMIO mediation mechanisms, such as trap-and-emulate handling in virtualized environments, which can introduce overhead.

To achieve the requirements discussed above in EUKL, we explore virtio-msg for inter-OS communication between the host RTOS and EUKL on a single MCU. Its message-based transport naturally maps onto RTOS-native message-based IPC mechanisms while preserving Linux-compatible virtio abstractions. Our current prototype demonstrates the ability to integrate RTOS-side and Linux-side software into one application, which is expected to help reduce development costs in IoT applications.

Table 1: Prototype environment

Item	Description
Hardware	Arm MPS3-AN547 FPGA (Arm Cortex-M55 @ 32MHz)
Assigned RAM	256MB
Host RTOS	TOPPERS/HRP3
Guest OS	EUKL (based on NoMMU Linux v6.1)

2 Applying virtio-msg to EUKL

virtio-msg is a message-based transport mechanism that enables virtio over various low-level communication buses [1]. Its specification is currently under development[1]. It separates a common virtio transport layer from bus-specific message delivery mechanisms. The common layer defines transport messages required by the virtio core, while the bus-specific layer carries these messages over a particular communication mechanism. This separation allows a system to use virtio by implementing only a relatively small bus-specific driver for the target platform. Thus, virtio-msg can provide highly abstracted inter-OS communication to the Linux side through virtio devices, while allowing the underlying low-level communication bus to be tailored to the MCU and RTOS environment.

In EUKL, we implement this bus over RTOS-native inter-task communication between EUKL and RTOS-side tasks. We place the virtio device backend side on the host RTOS and the frontend driver side in EUKL. This design reduces implementation effort because EUKL, as a Linux-based environment, can reuse the existing Linux virtio framework and frontend drivers such as virtio-net. Our bus-specific driver for RTOS inter-task communication consists of dedicated tasks and a mechanism for injecting virtual interrupts into EUKL.

3 Prototype

To demonstrate the feasibility and usefulness of the discussed inter-OS communication mechanism, we implemented a prototype application that provides WireGuard, a VPN protocol, using virtio-msg. Table 1 shows the configuration of the prototype, and Figure 1 illustrates its architecture. The RTOS-side application accesses the WireGuard service through a virtio-net interface, with the WireGuard and physical Ethernet drivers in the Linux kernel. In the current implementation, a UDP echo test through WireGuard showed an average round-trip time of 159.4 ms and a maximum round-trip time of 458 ms over 100 trials, with no packet loss. These results

[1]https://github.com/Linaro/virtio-msg-spec

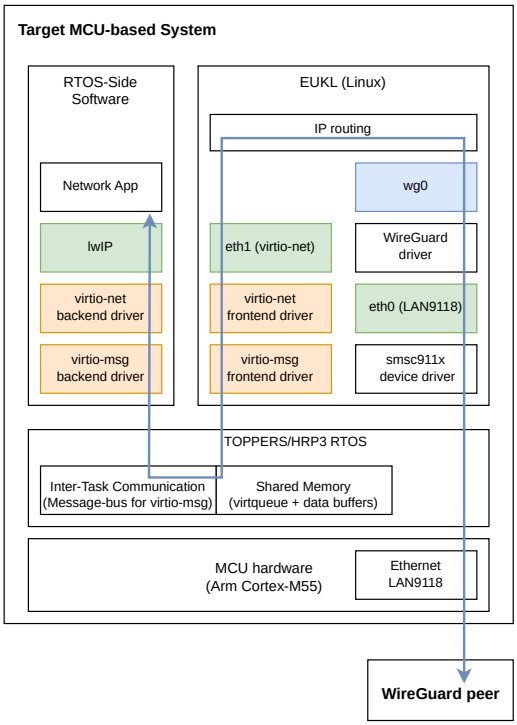

Figure 1: Prototype architecture

show that RTOS-side applications can exchange application-level packets through Linux-side WireGuard using the discussed virtio-msg-based inter-OS communication path. Performance optimization remains future work.

4 Conclusion & Future Work

In this paper, we explored virtio-msg for inter-OS communication between a host RTOS and the RTOS-hosted Linux environment provided by EUKL on a single MCU. This mechanism enables the integration of RTOS-side and Linux-side software into one application and is expected to help reduce development costs in feature-rich IoT applications. Our future work includes detailed evaluations through comparisons with alternative inter-OS communication mechanisms and extensions of the prototype to other virtio devices.

References

[1] Linaro. [n. d.]. Heterogeneous VirtIO for Automotive Computing - Confluence. https://linaro.atlassian.net/wiki/spaces/HVAC/overview. Accessed: 2026-05-22.

[2] Yoshifumi Shu, Yutaka Matsubara, Yixiao Li, and Hiroaki Takada. 2025. Towards a Linux-based Unikernel for Resource-Constrained Embedded Systems. In *Proceedings of the 19th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2025)*. Brussels, Belgium, 23–27.