

Integration of ROS and RT tasks using message pipe mechanism on Xenomai for telepresence robot

R. Delgado, B.-J. You, M. Han and B.W. Choi✉

A promising approach integrating non-real-time (NRT) robot operating system (ROS) packages and RT tasks is proposed to enhance the development of RT robot control applications. Since ROS alone does not provide RT properties essential for achieving precise control period in manipulating multiple devices and complicated software, Xenomai, an RT extension of Linux is adapted. However, using NRT ROS packages inside RT tasks triggers mode switching that causes inability to satisfy critical temporal constraints. To address this issue, a message pipe mechanism termed cross-domain datagram protocol (XDDP) is applied. In comparison to traditional inter-task mechanisms, XDDP provides a communication interface between RT and NRT tasks. This greatly improves robot application development utilising ROS tools and packages with RT tasks on the Xenomai domain ensuring priority-based scheduling and deterministic response in a multitasking environment. Feasibility of the proposed method was validated for practical use by realisation on the open embedded controller for a telepresence robot. Experiments were conducted to actuate the mobile base of the robot using ROS navigation packages. The results indicate that the robot accomplishes its objectives while satisfying RT constraints.

Introduction: Telepresence robots are mobile robot platforms designed to help people communicate across distances focused on the concepts of remote presence and telecommunication [1]. These robots are equipped with several types of sensors and actuators operated with algorithms contingent on expensive proprietary black box software that is vendor specific and prohibits integration to more complex systems. Robot operating system (ROS), the prevalent open source robotic framework, provides an easily redistributable collection of development tools and libraries to reduce user tasks for creating complex and robust robot behaviour [2, 3]. However, ROS does not operate in RT which is a critical requirement to administer precise control period and to meet necessary temporal deadlines of priority-based multi-tasking RT systems.

Some researchers proposed several approaches to make ROS RT. In [2], a host-guest system is suggested where an RT operating system (RTOS) is running on the guest hardware while the host handles the ROS packages. This is a costly approach using a proprietary RTOS and the performance is highly dependent on the communication protocol between both devices. Wei *et al.* [4] exploit the inter-core mechanisms of a multicore Intel processor; operating ROS under the standard Linux on one core and an open source RTOS on the other. However, the flexibility of this solution is questionable because the main controller of robots is mostly built on embedded platforms [5].

In this Letter, we introduce a promising solution based on Xenomai, an RT dual-kernel approach of embedded Linux integrating ROS and real-time (denoted as RT for tasks) tasks through a communication interface termed, cross-domain datagram protocol (XDDP). In comparison to the prior studies, the method is based on existing open source technologies allowing redistribution and easy integration of ROS to more complicated RT applications. Practicality and flexibility are verified on a low-cost and open embedded hardware platform, Raspberry Pi 3 (RPi 3), for the RT control of the M4K telepresence robot [1], using navigation packages available in ROS.

Integration of ROS using XDDP on Xenomai: Fig. 1 shows the overall simplified model of an RT application integrating ROS and RT tasks on Xenomai. ROS core is realised on the standard Linux kernel to handle applications denoted as ROS nodes. Xenomai runs alongside the standard Linux through a hardware abstraction layer, termed adaptive domain environment for operating systems (ADEOSs), that enables both kernels to exist in the same machine. Using standard Linux or ROS functions inside an RT task triggers mode switching (MSW) that causes non-deterministic response and stability problems of the entire system. Correspondingly, the RT task is scheduled by the standard Linux scheduler resulting in loss of RT capabilities.

To integrate ROS packages with RT tasks on Xenomai without MSW, we adopted the message pipe mechanism, XDDP. Compared to the traditional inter-task communication mechanisms, which is only feasible between RT tasks [6], XDDP is developed based on the RT driver model

to provide a communication interface for data concurrency between RT and non-real-time (NRT) tasks. In this Letter, RT tasks are connected to NRT tasks that perform original ROS operations instead of directly accessing the NRT resources. Hence, RT tasks are schedulable based on priorities that ensure deterministic response while utilising the rapid-development tools and libraries offered by ROS.

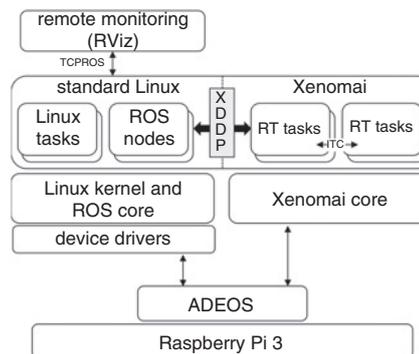


Fig. 1 Realisation of RT control application integrating ROS and RT tasks on Xenomai

Realisation of a low-cost open embedded platform: The RT environment is implemented on an open embedded hardware, RPi 3, that serves as the main controller of M4K. Unlike desktop personal computer based on the Intel architecture in [2, 4], realisation on an embedded platform is more difficult owing to lack of existing systematic documents and technical support. Therefore, the compatibility of each software should be analysed to achieve a feasible RT environment. In case of unavailability, the common solution is to commit a patch written by the user firsthand.

This Letter focuses on a redistributable and reusable solution for easier reproduction by RT developers. To establish a cross-development environment supporting the RPi 3, we utilised a toolchain available in the repository of the manufacturer. Since the bootloader of the RPi 3 has closed its licence by Broadcom and not available for redistribution, this is kept intact without any modification. The latest version of Xenomai is ported alongside the compatible version of the Linux kernel in accordance with the available ADEOS patch.

Later distribution of ROS is currently existent, but we preferred the latest stable version which has received extensive testing from the robot community. For the root filesystem, ROS highly recommends an Ubuntu distribution. However, owing to the limitation in modifying the device tree binaries in extending the serial peripheral interface (SPI) (motor and encoder support), disabling Bluetooth for full RS232 support, and disabling the sound card (pulse-width modulation for LED), we have selected Raspbian Jessie. This introduces a small trade-off of manually building the ROS sources and considering toolchain compatibility. Table 1 shows the software configuration for the RT environment with the tested version compatibility.

Table 1: RT environment for RPi 3

Item	Description
Toolchain	gcc-linaro-arm-linux-gnueabi-hf-raspbian-4.8.3
Bootloader	Broadcom Licenced Bootloader
ADEOS	ipipe-core-4.1.18-arm-4
Kernel	Linux Kernel v4.1.21 with Xenomai 3.0.2
Filesystem	Raspbian Jessie (Debian 8)
ROS	ROS Kinetic Kame

Navigation of M4K using ROS packages: Here, the M4K is operated in an environment with a static obstacle which is a common topic in conventional mobile robot navigation. However, our aim is to easily realise a navigation scheme for the robot using ROS navigation packages while satisfying RT requirements in a multi-tasking environment.

Fig. 2 shows the specifications of the M4K telepresence robot. The robot is designed to overcome the spatial limit and extend the interaction field between participants in a coexistent space [7]. The mobile base includes actuators and sensors such as a laser rangefinder (LRF), ultrasonic (sonar), and an inertial measurement unit (IMU). Standard Linux device drivers are developed to control each device from the RT tasks.

However, direct access of these drivers triggers MSW that causes a system freeze as shown in the left-hand side of Fig. 3. To address this issue, device drivers are operated in NRT tasks which are connected to the RT tasks through XDDP. The right-hand side of the figure shows the proper behaviour of the device displaying the acquired data on a terminal. Four instances of MSW are detected due to opening and binding a socket to an XDDP port. This routine is executed only once before the RT task loop and does not affect the RT performance.



robot part	specification	details
boby	dimensions	L45×W28 ×H20 cm
	weight	30 kg
	drive system	differential
actuators	speed	200 cm/s
	acceleration	37 cm/s ²
	gear ratio	12:1
	diameter	20.32 cm
wheels	width	5 cm
	position	LRF
sensors		sonar
		IMU
	status	LED

Fig. 2 Specifications of M4K telepresence robot

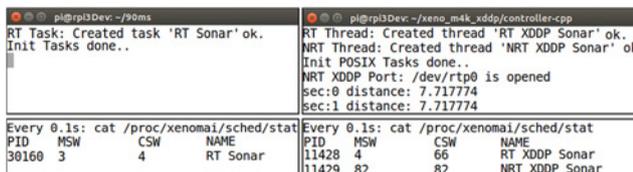


Fig. 3 Execution of standard Linux device driver in RT task

Navigation of the M4K requires a multi-tasking environment with four RT tasks with their respective cyclic periods and priorities as shown in Fig. 4. An ROS node is developed as a subscriber to the ROS navigation package, *move_base*, which produces centre velocity commands. The node is linked to the highest priority RT task (actuator) that converts the centre velocity to joint space commands for the actual actuation of M4K via XDDP. The odometry task acquires data from the encoder and IMU and calculates the current position of the robot. The calculated position is sent to the same ROS node via another XDDP, which then publishes it to a remote monitor running *RViz* (see Fig. 1), the visualisation tool provided by ROS. The status and obstacle detection RT tasks handle the LED and sonar sensor, respectively. An ROS node for the LRF is connected directly to the ROS navigation stack because the generation of the environment map does not require hard RT scheduling. As shown in the figure below, all the scheduled RT tasks were able to meet the expected temporal deadlines in accordance with their priorities.

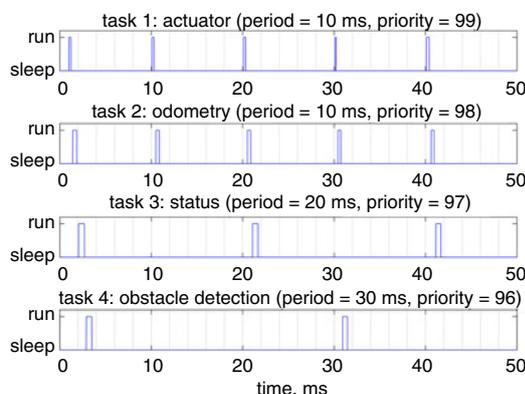


Fig. 4 Execution timeline of RT tasks for navigation of M4K

The trajectory planners [3] included in the ROS navigation package are configured according to the kinematics of M4K. To avoid any

accidents that could occur due to slip, experiments were conducted at a reduced speed of 0.25 m/s. The left-hand side of Fig. 5 shows the reference and feedback velocities of the M4K; travelling along a trajectory that avoids an obstacle as shown in the Cartesian space at the right-hand side of the figure. These results indicate that the proposed method is feasible in controlling the telepresence robot using ROS packages while satisfying real-time requirements in an embedded environment.

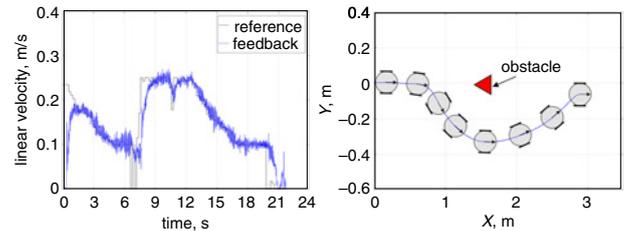


Fig. 5 Trajectory of M4K using ROS navigation packages

Conclusion: This Letter proposes a method to integrate ROS and RT tasks on Xenomai through XDDP. The experiment results obtained through implementation on an embedded platform for RT control of a telepresence robot indicate that the proposed method enhances RT application design exploiting the rapid-development tools of ROS in a multi-tasking environment that supports priority-based scheduling and deterministic response of the entire system. This approach opens up possibilities to integrate ROS in more complicated RT applications.

Acknowledgments: This work was supported by the Human Resources Development of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea Government Ministry of Trade, Industry & Energy (No. 20174030201840) and by the Global Frontier R&D Program on 'Human-centered Interaction for Coexistence' funded by the National Research Foundation of Korea grant funded by the Korean Government (NRF-2010-0029759).

This is an open access article published by the IET under the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>)

Submitted: 28 May 2018 E-first: 3 December 2018

doi: 10.1049/el.2018.5560

One or more of the Figures in this Letter are available in colour online.

R. Delgado and B.W. Choi (Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, Republic of Korea)

✉ E-mail: bwchoi@seoultech.ac.kr

B.-J. You and M. Han (Center of Human-centered Interaction for Coexistence, Seoul, Republic of Korea)

References

- Lee, H., Kim, Y.H., Lee, K.K., *et al.*: 'Designing the appearance of a telepresence robot, M4k: a case study', in Koh, J., Dunstan, B.J., Silvera-Tawil, D., *et al.* (Eds.): 'Cultural robotics. CR 2015. Lecture notes in computer science' (Springer, Cham, Switzerland, 2016)
- Bouchier, P.: 'Embedded ROS [ROS topics]', *Robot. Autom. Mag.*, 2013, **20**, (2), pp. 17–19
- Marin-Plaza, P., Hussein, A., Martin, D., *et al.*: 'Global and local path planning study in a ROS-based research platform for autonomous vehicles', *J. Adv. Transp.*, 2018, **2018**, pp. 1–10
- Wei, H., Shao, Z., Huang, Z., *et al.*: 'RT-ROS: a real-time ROS architecture on multi-core processors', *Future Gener. Comput. Syst.*, 2016, **56**, pp. 171–178
- Hu, C., Arvin, F., Xiong, C., *et al.*: 'Bio-inspired embedded vision system for autonomous micro-robots: the LGMD case', *Trans. Cogn. Dev. Syst.*, 2017, **9**, (3), pp. 241–254
- Shin, U.C., and Choi, B.W.: 'Performance evaluation of real-time mechanisms on open embedded hardware platforms', *J. Inst. Control Robot. Syst.*, 2017, **23**, (1), pp. 60–66
- You, B.-J., Kwon, J.R., Nam, S.-H., *et al.*: 'Coexistent space: toward seamless integration of real, virtual, and remote worlds for 4d+ interpersonal interaction and collaboration'. SIGGRAPH Asia 2014 Autonomous Virtual Humans and Social Robot for Telepresence, Shenzhen, China, December 2014, pp. 1–5