組込みデバイスを活用したROSコンポーネントの軽量実行環境の初期検討

森 智也[1]　高瀬 英希[1]　高木 一義[1]　高木 直史[1]

概要：近年のサービスロボット開発では，開発支援フレームワークである ROS（Robot Operating System）の活用が注目されている．ROS はソフトウェア・コンポーネント間の通信レイヤを提供するミドルウェアであり，Linux 上での動作を想定している．そのため，高機能かつ消費電力の大きいデバイスを採用する必要がある．そこで本研究では，消費電力の小さな組込みデバイス上で ROS ノードを軽量に実行可能な環境を検討する．提案する実行環境は，リアルタイム OS および ARM mbed ライブラリを使用して実現する．このため，設計対象のエッジ端末におけるコンポーネントの省電力化およびリアルタイム性確保が容易になると考えられる．

キーワード：ROS，リアルタイム OS，組込みマイコン，ロボット

A light-weight runtime environment for ROS components using the embedded device

Tomoya Mori[1]　Hideki Tasase[1]　Kazuyoshi Takagi[1]　Naofumi Takagi[1]

Abstract: Recently, ROS (Robot Operating System) has been attracted attention as a component-based development framework for robot systems. ROS provides the communication layer between software components as the middleware. Since ROS is designed to be operated on Linux, it is necessary to employ a rich device with high function and high power consumption. In this research, we study a light-weight runtime environment for the ROS component that can be driven on embedded devices. We are developing the proposed environment by using the real-time OS and ARM mbed libraries. Our study would contribute to save power consumption and ensure real-time capability of ROS components on the edge device.

Keywords: ROS, real-time OS, embedded micro-computer, robot

1. はじめに

近年，様々な場面でモバイルロボットの需要が高まっている．特に，災害救助ロボットやガイドロボット，掃除ロボットなど，社会生活においてサービスを提供するためのロボットの開発が活発になっている．これらのロボットは過去に開発されてきた産業用ロボットとは異なり，外部電源からのエネルギー供給ではなく，内部電源のエネルギーによって動作するという特徴がある．また，高度かつ多機能なサービスを要求されることが多く，一つのロボットシステムで様々な制約のもと多くの機能を実現しなくてはならない．そのため，ロボットが提供するサービス品質の向上には，省電力かつ多機能の実現が必要になる．

[1] 京都大学
Kyoto Uniersity

ROS（Robot Operating System）[1] はロボットシステムの設計生産性の向上に資する開発支援フレームワークとして注目されている．ROS ではソフトウェア部品をノードとして表現し，複数のノードを組合わせることでロボットシステムを実現する．実行ファイルと環境設定ファイルなどをまとめた ROS パッケージが数多く公開されており，それらを活用することで効率的なコンポーネント指向開発が可能になる．また，ROS はノード間での通信層を提供するミドルウェアとしての役割も担っている．しかし，ROS は Linux 上での動作を想定した設計になっており，高機能かつ消費電力の大きなデバイスを採用する必要がある．

本研究では，組込みデバイス向け ROS ノード実行環境である mROS を提案する．mROS は組込み環境向けの対 ROS システム用通信ライブラリを提供する．ROS システムにおいて主に使用される XML-RPC および TCPROS の

ⓒ 2017 Information Processing Society of Japan

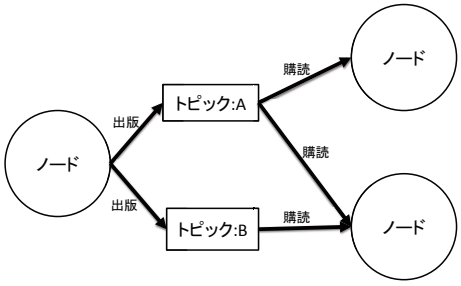

図 1 ROS におけるノード間の出版購読通信モデル
Fig. 1 Publish/subscribe communitacion model in ROS

通信プロトコルをサポートすることで，ROS システムと通信を行えるようにする．現在 TOPPERS/ASP カーネル [2]，ARM mbed ライブラリを使用して GR-PEACH を対象に mROS の開発を行っている．

提案する mROS によって，従来であれば ROS システムに組込むことができなかった消費電力の小さな組込みデバイスを活用することが可能になる．また，既存の ROS パッケージ資産を再利用することも可能になると考えられる．リアルタイム OS と TCP/IP プロトコルスタックが搭載された組込みデバイスを対象としているため，リアルタイムシステムの設計も容易になると考えられる．

2. ROS

ROS は，ロボットソフトウェアの開発フレームワークであり，開発されたソフトウェアの再利用性を高めることを目標として開発された．

ROS では，機能を実現するプログラムをノードという一つの単位として表現し，複数のノードを用いてロボットシステムを実現する．ROS はノード間における通信層を提供するミドルウェアであり，ソフトウェア開発や移植を簡単化している．ノード間の通信は，トピック名でデータの種類を分別し，出版されるメッセージを購読する出版購読通信で行われる．

図 1 にトピックによるノード間での出版購読通信のモデルを表す．扱うトピック名が同一であれば，ROS ロボットシステムにおいてハードウェアとそのノードを他のハードウェアと対応するノードに容易に変更できる．プログラム実行時には，ノードを管理するマスタとなる roscore を立ち上げることにより，それぞれのノードの名前空間や変数空間を管理する．

図 2 にデータを送信するパブリッシャーノードと受信するサブスクライバーノードが出版購読通信を始めるまでの通信の流れを示す．

(1) まず，ROS マスタに対して XML-RPC による register メソッドにより自身のノード名，扱うトピック名および URI の登録を行う．
(2) マスタから与えられた URI とポート番号に接続し，パブリッシャーノードに対して XML-RPC でトピックのリクエストを行う．パブリッシャからのレスポンスではポート番号などが送られる．
(3) 得られたポート番号に接続し，TCPROS 通信を行う．TCPROS 通信では，接続した直後に TCPROS コネクションヘッダの送受信を行う．
(4) コネクションヘッダの送受信が完了し接続が確立された後，トピックに対応したデータ転送が開始される．

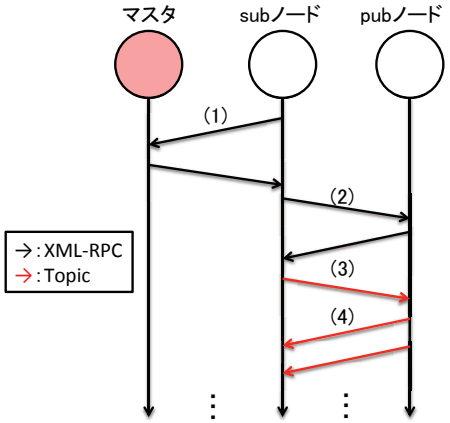

図 2 出版購読通信における通信フロー
Fig. 2 Publish/subscribe communication flow

mROS において，以上の通信フローをサポートすることで ROS ノードを組込みデバイス上で実行可能にする．また，ROS のトピック通信に使用されるプロトコルは TCP と UDP の 2 種類あるが，mROS では TCP のみをサポート対象として実装する．

[3], [4] では，ROS がリアルタイム性を持っておらず，限られた OS 上でしか動作しないため，ROS はリアルタイム分散組込みシステムの構築には向いていないと述べられている．そこで，ROS にリアルタイム性を持たせるための手法が提案されている．

ROS の次世代バージョンとして開発されている ROS2 では，リアルタイム OS を含む様々な環境で実行可能であるように設計されている．しかし，ROS2 では master ノードが存在せず，ノード間の通信システムが ROS とは異なるため ROS との間に互換性がない．

ROS システムにおいて組込みデバイスを利用する方法として，LinuxOS の代わりにリアルタイム OS を使用する手法が提案されている [5], [6]．また，組込みデバイス上で動作するプログラムとの通信をサポートする ROS パッケージもある [7], [8]．これらの方法では，組込みデバイス上で実行されるプログラムはそれぞれの環境によるものであり，ROS ノードは実行できない．そのため，オープンソースとして公開されている ROS パッケージを組込みデバイス上で活用することが難しい．

3. mROS

本研究では，ROS ノードの軽量な実行環境を検討する．

ⓒ 2017 Information Processing Society of Japan

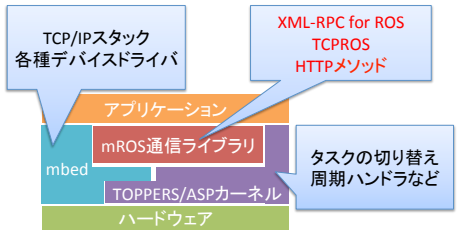

図 3　組込みデバイス上での ROS ノード実行環境
Fig. 3　Running emvironment for ROS components on the embedded device

提案する実行環境は mROS と名付けており，組込みマイコンで ROS ノードが動作できるように設計を進めている．

3.1　開発目標と設計要件

本研究では，ROS および Linux の動作するホストデバイス，および，ROS ノードの動作する組込みデバイスであるエッジ端末から構成される分散システムを想定する．ROS のノードを管理するマスタである roscore はホストデバイス上で動作していることとする．組込みデバイス上で実行されるプログラムは，ROS ノードとして振る舞う．すなわち，ROS の通信層を介して ROS システムとの通信をサポートする通信ライブラリを提供する必要がある．

汎用 PC における ROS では，Linux が提供する TCP/IP プロトコルスタックを用いた通信層を介してノード間のデータ通信を行う．Linux が搭載できない組込みマイコンにおいて ROS 通信層と通信する場合には，Linux 無しで動作できる TCP/IP プロトコルスタックを提供する必要がある．加えて，組込みデバイスからホストデバイスとの通信処理を管理する機能が必要である．

提案する実行環境である mROS を用いることで，組込みデバイス上で ROS ノードとしてプログラムが実行可能になることを目標とする．これにより，ROS パッケージとして記述されたソフトウェアを，組込みデバイス上の環境に対してクロスコンパイルを行うだけで ROS ノードとして実行可能となるようにする．

3.2　構成

mROS では，リアルタイム OS と TCP/IP プロトコルスタックを使用して mROS 通信ライブラリを提供することで ROS ノードの実行環境を実現する．通信処理の管理を担うリアルタイム OS としては，μITRON 仕様のリアルタイム OS である TOPPERS/ASP カーネルを採用する．TCP/IP プロトコルスタックには，ARM mbed ライブラリに含まれる lwIP を用いる．現在は，両者が移植済みである GR-PEACH [9] を対象として開発を進めている．**図 3** に組込みデバイスにおける ROS ノードの実行環境を示す．開発中の実行環境では，mbed ライブラリと TOPPERS/ASP カーネルを活用したアプリケーション設計も可能である．

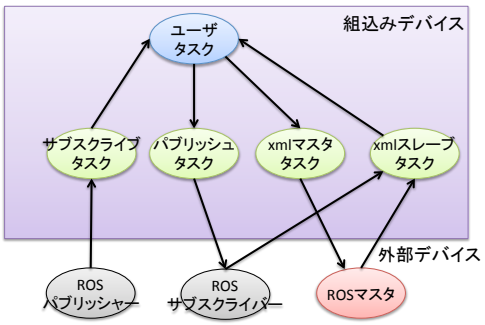

図 4　mROS におけるデータフロー
Fig. 4　Data flow on mROS

3.3　mROS 通信ライブラリ

3.3.1　実現する機能

mROS は組込みデバイス上のプログラムに対して以下の通信プロトコルをサポートすることで ROS ノードとしての振る舞いを可能にする．

- ROS 用 XML-RPC 通信
  XML-RPC は，通信データを XML でエンコードし，HTTP で転送することで手続き呼び出しを行う RPC プロトコルである [10]．ROS システムでは XML-RPC を用いてノードと ROS マスタ間およびノード間での手続き呼び出しを行う．mROS は XML-RPC マスタとして XML-RPC の生成と HTTP 転送，および，XML-RPC スレーブとして XML-RPC の要求を受け付けて応答する機能をアプリケーションに提供する．
- TCPROS データプロトコル
  TCPROS は ROS のメッセージとサービスのためのトランスポートレイヤであり，TCP/IP プロトコルでデータ転送を行う．TCPROS では，特有のフィールドが用意されておりノードの役割に応じて必要となるフィールドが異なる．ノード間での接続が発生した場合には，それらの情報を含んだコネクションヘッダを送信しなければならない．また，データに対してそのデータ長を付与してエンコードする必要がある．mROS ではコネクションヘッダを生成し送信する機能，送信するデータを TCPROS のフォーマットに変換する機能および受信したデータをデコードする機能をアプリケーションに提供する．

3.3.2　タスク構成

TOPPERS/ASP カーネルでは，アプリケーションはタスク単位で実行が管理される．mROS では，ユーザが記述するユーザタスクに対して，サブスクライブタスク，パブリッシュタスク，xml マスタタスクおよび xml スレーブタスクの 4 つのタスクが生成され，ROS システムとの通信を提供する．

図 4 にタスクの構成とデータフローを示す．ユーザタスクでは，ROS の API を使用してノード処理が記述でき

ⓒ 2017 Information Processing Society of Japan

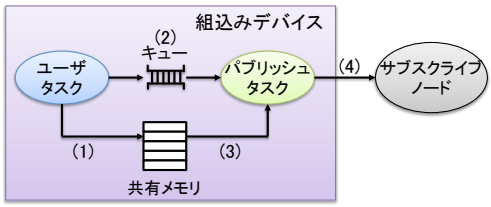

図 5 パブリッシュタスクフロー
Fig. 5 Task flow of publish

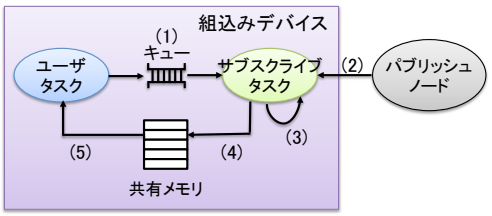

図 6 サブスクライブタスクフロー
Fig. 6 Task flow of subscribe

る．それぞれの ROS API に対応する機能は，4 つのタスクが提供する．サブスクライブタスクおよびパブリッシュタスクは，通信相手となるノードに対して TCPROS プロトコルで必要となるヘッダ情報の付与およびエンコードを行い，データを転送する．xml マスタメソッドでは，ROS マスタと他のノードに対する XML-RPC の HTTP 転送を行う．xml スレーブタスクは，マスタや他のノードからの手続き呼び出しの受付，レスポンスを行う．

mROS のライブラリでは，ROS の API の処理はそれぞれのタスクの呼び出しと定義している．つまり，ROS コンポーネントとして記述されているソースコードを修正せずに，組込みデバイス上で ROS API の実行が可能となる．なお，同一デバイス上で複数のノードの実行はユーザタスクを複数実装することで可能になる．

3.3.3 データ出版および購読の実行フロー

図 5 にデータを出版する際，**図 6** にデータを購読する際のタスク実行フローを示す．タスク間通信では，共有メモリとデータキューを使用する．

パブリッシュを行う場合は，データ出版を行うタイミングで，ユーザタスクが出版するデータが書き込まれた共有メモリのアドレスと mROSID をパブリッシュタスクに通知する．mROSID は，ノードの名前と扱うトピックの種類を表現する識別子である．パブリッシュタスクは共有メモリから取り出したデータを mROSID に対応するポート番号から出版する．

サブスクライブを行う場合は，まず，ユーザタスクはサブスクライブタスクに，mROSID とトピックを購読した際に呼び出すコールバック関数を通知する．サブスクライブタスクは周期的にトピックの購読を行い，コールバック関数を実行する．また，このときコールバック関数の実行結果をユーザタスクが必要な場合は共有メモリのアドレスを指定しておくことで，サブスクライブタスクはそのアドレスに結果を格納する．

4. まとめ

本研究では組込みデバイス向け軽量実行環境である mROS の検討を行った．TOPPERS/ASP カーネルと ARM mbed ライブラリを採用して mROS の開発を進めている．mROS 通信ライブラリで ROS の通信プロトコルをサポートすることで Linux が搭載できない組込みデバイス上で ROS ノードの実行が可能になる．

今後の課題として，ROS の通信方式の一つのサーバクライアント通信のサポートおよびすべての ROS API への対応がある．また，実行環境として TOPEERS/ASP カーネルと ARM mbed ライブラリを使用しているが，異なるリアルタイム OS やプロトコルスタックを使用した場合でも実行可能であることが望ましいと考えられる．そのため，対応デバイスの拡張や実現環境の拡張を検討する必要がある．さらに，組込みデバイスが利用できる ROS2 や，従来の ROS ノードと比較を行い，データ転送の通信時間およびメモリ消費量などを評価することを計画している．

謝辞 本研究の一部は JSPS 科研費 16H02795 の助成による．

参考文献

[1] Quigley, M., et al.: ROS: an open-source Robot Operating System, *ICRA workshop on open source software*, No. 3.2, p. 5 (2009).
[2] TOPPERS プロジェクト：TOPPERS/ASP kernel, `https://www.toppers.jp/asp-kernel.html`.
[3] Maruyama, Y., et al.: Exploring the Performance of ROS2, *Proc. of EMSOFT*, pp. 1-10 (2016).
[4] Bezemer, M., et al.: Connecting ROS to a real-time control framework for embedded computing, *Proc. of ETFA*, pp. 1–6 (2015).
[5] Migliavacca, M., et al.: μROSnode: running ROS on microcontrollers, *ROS Developers Conference* (2013).
[6] Migliavacca, M.: The R2P framework for robot prototyping: methodological approach, hardware modules, and software components, PhD Thesis, Italy (2014).
[7] Bouchier, P.: Embedded ros [ros topics], *IEEE Robotics & Automation Magazine*, Vol. 20, No. 2, pp. 17–19 (2013).
[8] Crick, C., Jay, G., Osentoski, S., Pitzer, B. and Jenkins, O. C.: Rosbridge: Ros for non-ros users, *Robotics Research*, Springer, pp. 493–504 (2017).
[9] Open-source Software Platform Based on TOPPERS/ASP Kernel, mbed and Arduino Library for Renesas GR-PEACH.: `https://github.com/ncesnagoya/asp-gr_peach_gcc-mbed`
[10] Merrick, P., Allen, S. and Lapp, J.: XML remote procedure call (XML-RPC) (2006). US Patent 7,028,312.

ⓒ 2017 Information Processing Society of Japan