組込みシステム向け軽量 ROS 環境の産業ネットワーク向け拡張

2020SC088 竹内結斗
指導教員：本田晋也

1　はじめに

ROS2 とは，ロボット制御分野で使われているプログラミング及び通信のフレームワークである．ROS2 は，異なるコンピュータ上の複数のアプリケーションを連携することが可能である．アプリケーション間は，トピックと呼ばれる単位のメッセージを出版-購読型モデルでやりとりする．これらアプリケーション間の通信は，DDS[1] と呼ばれる通信ミドルウェアにより実現される．DDS はマスターを必要とせず，動的な追加削除が可能な機器間通信を実現できる．DDS を実行するには，ソフトウェアプラットホームに多くの機能が必要となる．具体的には，POSIX API をサポートした OS（例えば Linux）が必要である．

小規模な組込みシステムはリアルタイム性やコストを重視するため，OS レスや機能が少ない RTOS で動作することが多く，DDS を使用することが不可能である．そこで，これらのシステムで ROS2 を使用可能にする micro-ROS が開発されている [2]．micro-ROS は小規模な組込みシステムを Client として，PC 上の Agent を経由して DDS に接続する．組込みシステムでは機器間の通信に様々な方法が使われるため，micro-ROS はトランスポート層を独立させることにより，通信方法を容易に変更できる構成となっている．現状，通信方法として，Serial，Ethernet (TCP/UDP)，CAN-FD などがサポートされている．

CAN-FD は主に車載や FA のネットワークとして使われているシリアル通信プロトコルである．CAN-FD は CAN-ID による優先度制御が可能であり，リアルタイム性が高い通信が実現できる．

現状の micro-ROS の通信方法として CAN-FD を使用するには，以下の問題点が存在する．

- Client と Agent が送信するメッセージの CAN-ID が同じである．

このことは，CAN-FD では違反になる．さらに，改善すべき課題として以下がある．

1. すべてのトピックで CAN-ID が同じである．
2. 同じトピックを複数の Client に送信する．

本研究では，現状の micro-ROS の通信方法として CAN-FD を使用するために，micro-ROS の CAN-FD のトランスポート層を変更することにより，問題点を解決し，また，課題を改善するをことを研究目的とする．

2　背景技術

2.1　ROS2

ROS2 とはロボット制御分野で使われているプログラミング及び通信のフレームワークである．本研究では，通信方法としてトピック通信を用いる．トピック通信は，メッセージを送信する publisher とメッセージを受信する subscriber が，トピックで通信する出版-購読型モデルでやりとりを行う．さらに，ROS2 には QoS として Best Effort と Reliable が存在する．Best Effort は ACK がないのに対して，Reliable は ACK がある．

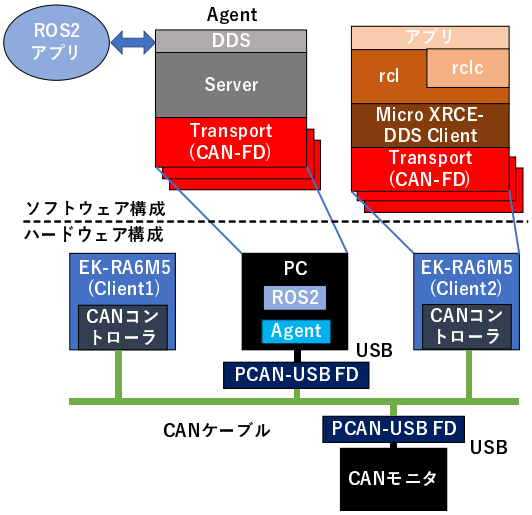

図 1　全体の接続図

2.2　micro-ROS

micro-ROS は小規模な組込みシステムで ROS2 を使用可能にする．micro-ROS は PC 上の Agent と通信して，DDS に接続する．micro-ROS は図 1 のように，通信を実現するトランスポート層が独立しており，様々な通信路に対応している．

2.3　CAN-FD

CAN-FD は CAN-ID による優先度制御が可能である．メッセージの種類ごとに固有の CAN-ID を決め，各ノードは受信したい CAN-ID のメッセージを受信する．この仕組みにより，送信元のノードは送信先のノードの情報について知らなくても，メッセージを送信可能である．さらに，複数のノードがメッセージを同時に送信し，コリジョンが発生した場合，小さい CAN-ID のメッセージが優先されるので，リアルタイム性がある．

CAN-FD には，マスク機能により，受信するメッセージを指定することが可能である．具体的には，特定の CAN-ID のみを受信，CAN-ID の特定の bit がマッチするとメッセージを受信可能である．

3　CAN-FD トランスポート層の問題点と改善すべき課題

現状の CAN-FD 向けのトランスポート層は，前述の通り，1 個の問題点と 2 個の改善すべき課題が存在する．

問題点について説明する．Client が Agent へメッセージを送信すると，Agent が Client へ応答メッセージを送信する．その時に，Client が送信するメッセージの CAN-ID を Agent が送信する応答メッセージの CAN-ID に使用している．同一 CAN-ID のメッセージが同時に CAN バス上に流れると，メッセージが正しく受信されないので，CAN-FD では違反になる．Agent ないし，Client の片方からのみ publish する場合には問題は発生しないが，双方ともに publish する場合には，この問題が発生し通信が行えない状況が発生する可能性がある．

改善すべき課題 1 は，すべてのトピックに対して同じ CAN-ID を使用しているので，CAN-ID による優先度制御に対応できない．トピックの種類ごとに，異なる CAN-ID を割り当て，優先度制御に対応させる必要がある．

改善すべき課題 2 は，同じトピックを 1 つずつ，複数の Client に送信する．この方法は，ネットワーク負荷を上げるので，同じトピックを複数の Client に同時に送信する必要がある．

4　問題点への対応

問題点を解決するために，Agent がある Client 向けに送信するメッセージには，その Client が送信するメッセージとは異なる CAN-ID を付与する仕組みを実現した．

まず，CAN-ID の上位 17bit を Client ごとに異なる値を割り当てることとする．Client はメッセージに CAN-ID の上位 17bit は割り当てられた値を，下位 12bit は 0 として送信する．Agent はメッセージは CAN-ID の上位 17bit を送信先の Client に割り当てた値を，下位 12bit は 0x100 として送信する．下位 12bit は改善すべき課題に使用する予定である．そのため，Client はマスク機能により受信時は CAN-ID の上位 17bit のみをチェックして自分に割り当てられた値のメッセージなら受信するようにする．

5　改善すべき課題 1 への対応

本研究では，通信設定が「Client が publisher かつ QoS が Best Effort」において実施した．

5.1　対応方法

トピックを publish する前に生成メッセージが送信される．仕様書 [3] から生成メッセージのメッセージフォーマットを解析し，メッセージごとに状態遷移するように Client の CAN-FD トランスポートを変更した．具体的な実現方法は，まず，生成メッセージより，トピック名とトピックを識別する ID を抽出する．その後，ユーザが設定したトピック名と CAN-ID のテーブルより，CAN-ID と ID の対応テーブルを生成する．publish フェーズでは ID から CAN-ID と ID の対応テーブルにより CAN-ID を特定し，割り当てる．

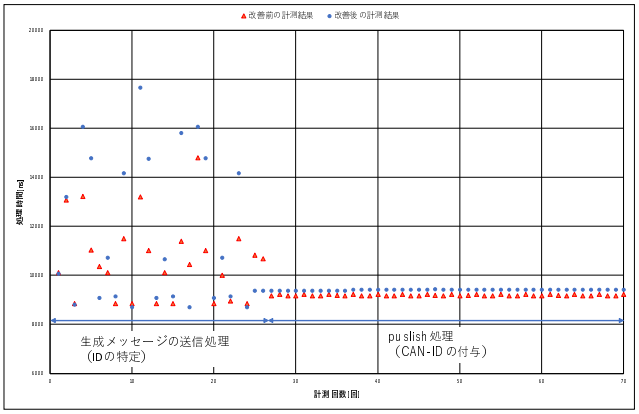

図 2　CAN-FD トランスポートの送信関数における CAN-FD 送信処理を含めた実行時間

5.2　性能評価

トピック毎に CAN-ID を割り当てる処理を Client の CAN-FD トランスポートに追記したので，プログラムの改善前に比べて，改善後の処理時間の増加について評価する．CAN-FD トランスポートでの送信関数における CAN-FD への送信時間の計測結果を図 2 に示す．生成メッセージが送信された後，トピックが publish し続けるので，生成メッセージが送信された後の結果は同じであると判断する．改善前と改善後の最大処理時間の差は 4,460ns であるが，実際にトピックを publish する際の改善前と改善後の処理時間の差の平均は 230ns であり，publish 全体の処理時間 581ms に対して 0.04% の実行オーバヘッドで実現できた．

6　おわりに

本研究では，現状の micro-ROS の通信方法として CAN-FD を使用する際に，Client と Agent が送信するメッセージの CAN-ID が同じであるという問題点を解決した．さらに，設定方法が Client が publisher かつ QoS が Best Effort である際のすべてのトピックで CAN-ID が同じであるという改善すべき課題を改善した．今後，改善すべき課題 1 において他の通信バリエーションに対応することを実施する．さらに，改善すべき課題 2 を改善する．

参考文献

[1] Object Management Group(OMG): OMG Data Distribution Service (DDS) 1.4 (2015)

[2] K. Belsare, A. C. Rodriguez, P. G. Sanchez, J. Hierro, et al. Micro-ROS. In: Anis Koubaa (ed.) Robot Operating System (ROS): The Complete Reference (Volume 7), Springer, pp. 3-55 (2023)

[3] Object Management Group(OMG): DDS for eXtremely Resource Constrained Environments 1.0 (2019)