Vol.2019-MBL-93 No.23
Vol.2019-ITS-79 No.23
2019/11/22

ROS2を用いた遠隔運転を想定した通信のリアルタイム性評価

川上 勇剛[†1]　本田 晋也[†2]　倉地 亮[†2]　中條　直也[†3]

概要：近年, 自動運転システムの開発が進んでいる. しかし, 走行が難しい場面では, 自動運転の代わりに遠隔運転の使用が予想される. 自動運転は, ロボット開発向けのミドルウェアである ROS1 を用いて開発されてきた. しかし, リアルタイム性確保の仕組みがない問題点があった. そのため, リアルタイム制御を考慮した ROS2 が開発されている. そこで本研究では、ROS2 を用いた遠隔運転を想定した場合の通信のリアルタイム性評価を行った. 評価実験の結果, ROS2 は, 遠隔運転の通信の信頼性の向上に繋がることが分かった. しかし, 現在 ROS2 は, 開発途中であり制御できるパラメータが限られている点が問題であると分かった. これから開発が進むことで, より柔軟にシステムに応じた通信の信頼性を担保できるようになると考えられる.

キーワード：ROS2, 遠隔運転, リアルタイム性

Real-time Evaluation of Communication Assuming Remote Driving using ROS2

Yugo Kawakami[†1]　Shinya Honda[†2]　Ryo Kurachi[†2]　Naoya Chujo[†3]

Abstract: In recent years, development of autonomous driving has progressed, but in situations where it is difficult to drive, it is expected that remote driving will be used instead of autonomous driving. Autonomous driving has been developed using ROS1, a middleware for robot development. However, there was no mechanism for ensuring real-time performance. For this reason, ROS2 that considers real-time control has been developed. In this study, we evaluate the real-time performance of communication assuming remote driving using ROS2. As the result of the evaluation experiment, it was found that ROS2 led to the improvement of the reliability of communication for remote driving. However, ROS2 is currently under development and the problem is that the controllable parameters are limited. As development progresses in the future, it will be possible to ensure more reliable communication according to the system.

Keywords: ROS2, remote driving, real-time

1. はじめに

近年, 自動運転の開発が進んでいる. しかし, 走行が難しい場面では, 自動運転の代わりに遠隔運転の使用が予想される. 自動運転システム用オープンソースソフトウェアとして, Tier IV 社の Autoware があり, ROS 1(Robot Operating System 1) を用いて開発されている [1].

ROS1 とは, オープンソースで提供されるロボット開発向けのミドルウェアであり, 研究開発用に広く利用されている. 一方で ROS1 の課題として, リアルタイム性の確保が難しい点がある.

そのため, 2017 年 12 月に ROS2 がリリースされた. ROS2 では, リアルタイムおよび組込みシステム向けの通信プロ

†1 愛知工業大学大学院
Graduate School, Aichi Institute of Technology
†2 名古屋大学大学院
Graduate School, Nagoya University
†3 愛知工業大学
Aichi Institute of Technology

トコルである DDS(Data Distribution Service) と, 通信の信頼性を確保するための QoS (Quality of Service) を導入することでリアルタイム性の確保を目指している [2].

ROS2 ではユースケースに合わせて, QoS をカスタマイズすることが出来る. 通信の信頼性を制御するための QoS ポリシーと呼ばれるパラメータをカスタマイズしてシステム開発ができる. 毎度, 開発をする際にどの QoS ポリシーを利用するか設定する必要がある. そのため, 代表的なユースケースに合わせて設定された QoS プロファイルが 4 種類, 提供されている. しかし, これらがパフォーマンスにどのような影響を及ぼすか明らかではない [3].

そこで本研究では, ROS2 を用いた遠隔運転を想定した通信のリアルタイム性評価を行う. QoS ポリシーの違いによる評価を行い, QoS ポリシーの選び方と ROS2 を用いた遠隔運転における QoS ポリシーを提案する.

2. ROS2 の特徴

ROS2 では, ROS1 の問題点を解決するための設計がなされている [4].

(1) ROS1 は, メッセージ通信の仲介役である ROS マスタが落ちてしまうと, すべてのプロセスが停止してしまうという問題がある. ROS2 では DDS によってこの問題はなくなった.
(2) ROS1 は, リアルタイム性を考慮していないのに対し, ROS2 では DDS と QoS によってリアルタイム性確保を目指している.
(3) ROS1 は, メッセージを暗号化せず取り扱っており, セキュリティに問題がある. ROS2 ではデータの暗号化および認証機能を搭載している
(4) ROS1 は, 非力な計算資源では, 動作が難しい問題があった. しかし, ROS2 では, 組込み向けに非力な計算資源でも動作するように開発されている.

ROS2 のアーキテクチャを図 1 に示す. ROS2 は, rcl(ROS client library) と rmw(ROS middleware) から構成されている. rcl は, C++, Python, Java といった各プログラミング言語用クライアントライブラリに共通する機能を提供する API である. rmw は, ROS2 の基盤ソフトウェアを実装するために設計された, 通信機能を抽象化するミドルウェア API である.

本章では, リアルタイム性確保のための仕組みである DDS と QoS について述べる.

2.1 DDS

ROS2 は通信層に DDS を使用している. DDS は publish/subscribe 通信を提供する国際ミドルウェア標準であり [5], ROS2 は DDS を使用して, プロセス間通信を行っている.

DDS の publish/subscribe 通信モデルを図 2 に示す.

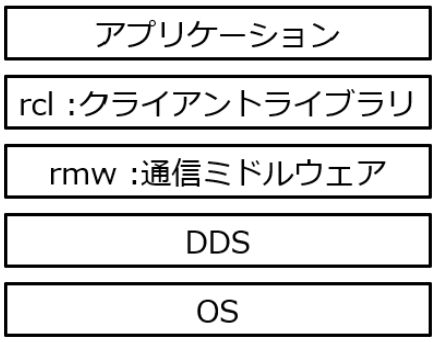

図 1 ROS2 のアーキテクチャ

Publisher から Subscriber にデータを Topic として受け渡す. Publisher, Subscriber, Topic 毎に QoS を設定することが出来る.

複数のベンダが DDS を実装している. 現在, ROS2 ではデフォルトで eProsima 社の Fast-RTPS が使用されている [6]. 他には, ADLINK 社の OpenSplice[7] や RTI 社の Connext[8] がある.

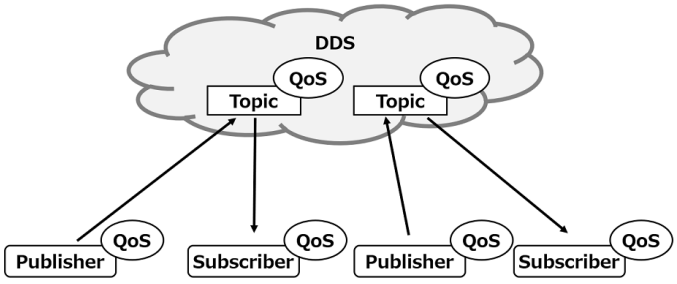

図 2 DDS の通信モデル

2.2 QoS

DDS 標準には 20 程度の QoS ポリシーがあるが, 現在, ROS2 が対応しているのはその一部である. 順次, 実装されていく予定である. 現在対応している QoS ポリシーを以下に示す.

- History
  - Keep last: Depth で指定した数だけ送信待ちメッセージをキューに保存する. 指定数を超えた場合, より新しいメッセージが保存される.
  - Keep all: DDS ミドルウェアのリソースの上限まで送信待ちのメッセージをキューに保存する.
- Depth
  - Size of the queue: Keep last を設定したときのキューに送信待ちのメッセージを保存する数を設定する.
- Reliability
  - Best effort: メッセージを一度しか送信しない. ネットワークが不安定な場合, メッセージが欠損する可能性がある.

– Reliable: メッセージの送信を失敗した場合に，メッセージを再送する．
- Durability
– Transient local: 遅いタイミングで subscribe した相手に送信するために, Publisher が古いメッセージを保存する．
– Volatile: Publisher が古いメッセージを保存しない．

代表的なユースケースに沿って定義されている QoS プロファイルを表 1 に示す．デフォルトでは, メッセージ再送を行って通信の信頼性を確保している．

表 1　定義されている QoS プロファイル

	Reliability	History	Depth	Durability
Default	Reliable	Keep last	10	Volatile
Sensor data	Best effort	Keep last	1	Volatile
Services	Reliable	Keep last	10	Volatile
Parameters	Reliable	Keep All	-	Volatile

3. 関連研究

公道において遠隔運転実証実験が行われている．KDDI は, 2019 年 2 月に国内初の公道において 5G を使用し, 運転席が無人の状態での実証実験を行った [9][10].

それまでの実証実験では時速 20 km での走行しか許可されていなかったが, 5G 回線が高速, 大容量, 低遅延の回線であることから, 時速約 30 km で運転席にドライバーのいない車両を公道走行させる実験を行った．

遠隔運転車の車内には前方確認用の高精細な 4K カメラ 1 台と右左後方確認用のフル HD カメラ 3 台とスピードメータ確認用カメラ 1 台が取り付けられている．カメラ映像を遠隔運転手が常駐する遠隔管制室に送信することで遠隔運転を行っている．

また, ROS2 の通信性能評価が行われている．文献 [11] では, ROS1 と ROS2 で文字列データ送受信のレイテンシやオーバーヘッドを測定している．256 byte から 1M byte までローカルネットワーク (1 台のコンピュータ) およびリモートネットワーク (2 台のコンピュータ) で段階的に評価を行っている．

ROS1 では, 通信開始時のメッセージが損失されるのに対して, ROS2 では損失せず通信の信頼性が高い結果が示されている．しかし, ROS2 の問題点として, リモートネットワークにおいて, 64 Kbyte を超えたあたりからレイテンシの増加量が大きいことをあげている．原因として大きいデータは, 小さいパケットに分割し送受信する必要があるためレイテンシの増加に繋がっていると述べている．

文献 [12] では, ROS2 は, 1 つの DDS しか使用できないため, 特徴の異なる通信毎に最適な DDS を使用することが出来ない点を問題点として挙げている．そこで 動的に最適な DDS に切り替える手法を提案している．データサイズ, 通信範囲, 再送確認に応じて最適な DDS を選択している．現在の ROS2 よりもレイテンシを抑えられている．

これらの評価は文字列データで行われており画像データでの通信性能が明らかになっていない．遠隔運転では, 画像データの通信が必要であるため, 本研究では, 画像データで評価を行う．

4. ROS2 を用いた遠隔運転と評価項目

本章では, ROS2 を用いた遠隔運転のシステム構成と遠隔運転における必要要件について述べる．また, 評価項目について述べる．

4.1 システム構成

本研究の ROS2 を用いた遠隔運転のシステム構成を図 3 に示す．遠隔運転車を想定した Publisher から車載カメラ画像（前後左右）を遠隔運転手を想定した Subscriber に送信する．車載カメラ画像を基にアクセル・ブレーキ・ステアリングの制御情報を遠隔運転者に送信することで遠隔運転を行う．評価環境の詳細を表 2 に示す．

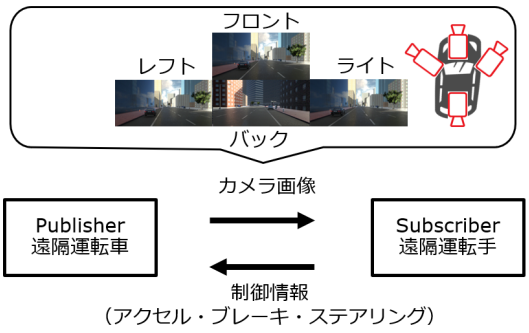

図 3　ROS2 を用いた遠隔運転のシステム構成

表 2　評価環境

		Machine
CPU	Model number	Intel Core i5-7200U
	Frequency	2.20GHz
	Cores	2
	Threads	4
Memory		8GB
Network		1Gbps Ethernet
ROS2		Dashing Diademata
DDS		OpenSplice
OS	Distribution	Ubuntu18.04
	Kernel	Linux 5.0.0.1

4.2 遠隔運転における必要要件

遠隔運転では, 映像を基に運転をするため, レイテンシが重要である．文献 [13] では, 遠隔運転で許容できるレイテンシを調査している．遠隔運転シミュレータを構築して,

カメラ映像に意図的に遅延をかけて, どの程度まで遠隔運転が可能か実験を行っている. 実験結果は, 時速 10 km の場合, 800 ms 程度までは遠隔運転が可能であると報告されている.

また, 文献 [14] の 最高速度 20 km のゴルフカートを用いた遠隔運転の実証実験の報告によると, 500 ms 以内を必要要件として挙げている. このことから時速 10 km や時速 20 km の低速の遠隔運転では, 500 ms 以内にレイテンシを抑える必要があると考えられる.

本研究では, 低速の遠隔運転を想定して必要要件は, レイテンシが 500 ms 以内とする.

4.3 評価項目

本報告では, カメラ画像送受信を 10000 回行う際のレイテンシやサンプルのロスト数の計測を行った. 平均, 最悪値, 最良値, 標準偏差を算出した. 平均と標準偏差は, ロストしたサンプルを除外して計算した. 計算に含めてしまうとレイテンシが実際よりも小さく計算されてしまうためである. 画像枚数 1〜4 枚での評価を行った. VGA の解像度のダミーデータを用いた. 画像 1 枚のサイズは, 374 kB であった. 送信周期は, 10 Hz に設定した.

正確に評価を行うために, 文献 [11] を参考に SCHED __ FIFO[15] と mlockall を使用した. SCHED __ FIFO は, デフォルトの Linux スケジューリングを使用するプロセスよりも優先して対象プロセスを実行する. また, mlockall は, プロセスの仮想アドレス空間を物理 RAM に固定することでメモリがスワップ領域にページングされることを防止する.

遠隔運転は, 街中や山道と様々な場所での使用が想定される. ネットワークの状態が常に良好であるとは限らない. ネットワークが不安定な場所では, 映像を安定して届けにくい. そのため, ネットワークが安定している場合と不安定な場合を想定して実験を行った. ネットワークが安定している場合は, ネットワークに負荷をかけず行い, ネットワークが不安定な場合は, ネットワークに負荷をかけて評価を行った.

ROS2 で定義されている QoS プロファイル [3] を参考に表 3 の QoS プロファイルを策定し実験を行った.

表 3 実験に使用した QoS プロファイル

条件	Reliablity	History	Depth	Durability
1	Reliable	Keep last	10	Volatile
2	Reliable	Keep last	1	Volatile
3	Reliable	Keep All	-	Volatile
4	Best effort	Keep last	1	Volatile

5. 評価

本章では前章で設定した内容について，評価実験の結果と結果に対する考察を述べる.

5.1 評価結果

ネットワークが安定している場合の実験結果を表 4 に示す. ネットワークが不安定な場合の実験結果を表 5 に示す. 画像枚数は, 送った画像枚数を示している. 1 枚から 4 枚での評価を行っている. 画像番号は, 何枚目の画像であるかを示している. サンプルのロスト数は, 10000 回サンプルの送受信を行った結果, 受け取ることができなかった数を示している. ネットワークが不安定な場合の実験では, 意図的にパケットロスを発生させる tc qdisc コマンドを用いて, ネットワークに負荷をかけた. 5%の確率でパケットロスを発生させた.

5.2 考察

ネットワークが安定している場合に比べてネットワークが不安定な場合は, サンプルのロスト数が増加していることが分かった.

Reliability の影響について述べる. 条件 1〜3(Reliable) は, 条件 4(Best effort) よりも, ネットワークが不安定な場合においてレイテンシが大きい結果となった. パケットがロスされた結果, 条件 1〜3 では, メッセージ再送する時間が増えたと考えられる. ネットワークが安定している場合では, 大きな差は見られなかった. ネットワークが安定しており, メッセージ再送が起きにくかったためであると考えられる.

サンプルのロスト数は, 条件 4 よりも条件 1〜3 の方が少ない結果となった. Reliable は, メッセージ再送のオーバーヘッドはあるが, サンプルのロストを削減し, 通信の信頼性の向上に繋がっていると分かった.

Depth の影響について述べる. Depth は, 送信待ちメッセージをキューに保存する数を指定するパラメータである. 条件 1 では 10 個, 条件 2 では, 1 個, 条件 3 ではリソースの上限までに設定している. 条件 2 は送信待ちメッセージを 1 個しか保存しないため, サンプルのロスト数が大きい結果となった. 条件 1 と 3 では大きな差はでなかったが条件 3 の方がリソースの上限まで送信待ちメッセージをキューに保存するため, サンプルのロスト数は少ないことが分かった. また, 条件 1〜3 において最悪値のばらつきが大きい結果となった. 条件 1〜3 は, Depth の数が違うのみである. 原因について Depth が影響しているか定かではない. Reliable のメッセージ再送時間のばらつきが影響している可能性もある.

また, ネットワークが不安定な場合での画像枚数が 4 枚の場合では, 平均が 20000 ms を超えてしまっている. これ

表 4　ネットワークが安定している場合のレイテンシ

画像枚数	画像番号	条件	平均 (ms)	最悪値 (ms)	最良値 (ms)	標準偏差 (ms)	サンプルのロスト数
1	1	1	3.5	126.1	2.6	4.7	0
		2	3.6	123.2	2.7	4	0
		3	3.3	55.3	2.7	0.6	0
		4	3.4	4.5	2.4	0.3	76
2	1	1	3.9	5.6	3.1	0.2	0
		2	4	54.2	3.2	0.5	0
		3	3.9	54	3.1	0.6	0
		4	3.9	5.5	3.3	0.2	0
	2	1	7.6	124.9	6	1.9	0
		2	7.5	111.9	4.2	1.2	0
		3	7.6	78.8	4.2	0.8	0
		4	7.5	8.4	6	0.5	4
3	1	1	5	124.8	3.5	6.4	0
		2	4.3	132.2	3.5	2.2	0
		3	4.3	124.1	3.4	2.1	0
		4	4.3	6	3.5	0.3	143
	2	1	8.1	150.4	3.8	6.7	0
		2	12.2	154.9	5.1	18.3	0
		3	13	151.2	4.9	19.6	0
		4	8.2	9.4	4.7	0.7	5
	3	1	12.4	176	4.7	6.2	0
		2	14.8	205.1	6.5	17.8	0
		3	15.2	175.6	6.4	17	0
		4	11.9	13.7	7.6	1	3
4	1	1	5.2	133.7	3.5	7.1	0
		2	5.2	277.9	3.6	7.7	0
		3	5.4	124.4	3.8	7.6	0
		4	4.8	7.2	3.9	0.5	69
	2	1	8.8	155.5	3.8	11.5	0
		2	10.7	303.7	5.1	11.4	1
		3	11.1	154.8	5	13	0
		4	9.3	14.1	4.5	0.8	82
	3	1	13.5	177.9	5.1	13.5	0
		2	15	208.7	7.8	12.5	1
		3	15.3	175.2	8	14.1	0
		4	13.5	18.8	7.6	1.2	53
	4	1	17.6	201.6	8.4	14.1	0
		2	18.8	279.9	10.6	13	3
		3	19	194.3	6.8	13.1	0
		4	17.1	24.4	10.5	1.4	88

について決定的な原因は不明である．これまでの経験から, データの転送量には, 許容できる量があることが分かっている．一定のラインを越えてしまった場合に大きいレイテンシとなることを確認している．今回においても画像枚数が 1〜3 枚では問題なかったが, 4 枚では許容量を超えてしまい, レイテンシが大きくなっていると考えられる．

今回の実験では, 4.2 で定義した要件を満たした (ネットワークが不安定な場合での画像枚数 4 枚の場合を除き)．しかし, ネットワークが不安定な場合に最悪値が 500 ms を超えてしまうデータがでていることが分かった．このとこから, 通信の信頼性とレイテンシは, トレードオフの関係にあると言える．

QoS ポリシーの選び方と ROS2 を用いた遠隔運転における QoS ポリシーについて述べる．Reliability は, Reliable で使用するのがサンプルのロストを防ぐことができるため望ましいと考えられる．しかし, メッセージ再送のオーバーヘッドによってレイテンシが大きくなってしまう場合があるため注意が必要である．

この問題については, QoS ポリシーの Deadline によって対応できると考えられる．Deadline では, Reliable のメッ

表 5　ネットワークが不安定な場合のレイテンシ

画像枚数	画像番号	条件	平均 (ms)	最悪値 (ms)	最良値 (ms)	標準偏差 (ms)	サンプルのロスト数
1	1	1	13.7	356	2.7	27.8	0
		2	15.4	401.8	2.6	30.4	9
		3	13.6	376.8	2.7	27.5	0
		4	3.3	4.3	2.5	0.2	5585
2	1	1	22.1	599.7	2.9	53.4	0
		2	31.7	957.9	2.9	85.2	22
		3	28.5	1180.5	2.8	84.8	0
		4	3.8	5.4	3.3	0.3	5560
	2	1	33.6	720.6	3.2	64.4	0
		2	42.4	975	3.4	95	63
		3	40.5	1204.8	3	95	0
		4	5.5	8.4	3.6	1.6	5313
3	1	1	232.1	2957.1	2.8	432.3	0
		2	319.5	2796.6	2.9	464.7	273
		3	451.5	4526.3	2.9	743.4	0
		4	4.1	5.8	3.5	0.3	5417
	2	1	241.3	2987.6	3.1	432.9	0
		2	336.9	2822.3	3.3	480.6	296
		3	472.1	4602.6	3.1	749.4	0
		4	6.2	11.2	4.2	1.6	5317
	3	1	260.9	2943.6	4.2	441.5	0
		2	359.4	2870.3	4	488.5	350
		3	495.9	4603.4	4.1	755.8	0
		4	8	15	4.7	2.3	5328
4	1	1	27406.8	42365.8	3.4	13081.9	7732
		2	15893.6	19561.1	3.1	3200.7	2201
		3	25600.1	40723.9	4.9	13076.2	7038
		4	4.8	10	3.8	0.5	5438
	2	1	34196.6	42458.3	7.2	7754.4	3064
		2	14657.7	19577	5.7	1964.9	2140
		3	32111.1	40885.8	381.1	8545.2	2354
		4	6.5	17.9	4.3	2	4356
	3	1	35642	42232.5	211.6	4216.9	354
		2	14665.8	19665.3	9.6	1950.8	2069
		3	34289.5	41484	434.5	4978	182
		4	8.6	23.6	5.1	2.6	4358
	4	1	35990.2	42202.7	12384	2233.1	112
		2	14661.1	19707.9	13.8	1952.5	2114
		3	35006.9	40752.8	436.7	2340.7	36
		4	10.4	27.8	5.7	3.3	4353

セージ再送時間の上限を設定できる仕様となっている．しかし, 現在, ROS2 では開発中であり実装されていない．今回の遠隔運転では, 必要要件の 500 ms に設定するのが良いと考えられる．

また, 常にレイテンシを最小にしたい場合には, Best effort を使用するのが良いと考えられる．Depth は, 1 ではロストが起きてしまうことがある．デフォルトの 10 で使用してみて状況に応じて調整すべきと考えられる．

6. おわりに

本報告では, ROS2 を用いた遠隔運転を想定した通信の QoS ポリシーの違いによるリアルタイム性の評価を行った．評価実験の結果, ROS2 は, QoS によってサンプル送受信のロストを削減することができ, 遠隔運転の通信の信頼性の向上に繋がることが分かった．

また, QoS ポリシーの選び方と ROS2 を用いた遠隔運転における QoS ポリシーについて述べた．メッセージ再送を行えばサンプルのロストは防げるがレイテンシは大きく

なってしまうため, 信頼性とレイテンシのバランスを取る必要があると言える. 現状の ROS2 では QoS ポリシーが全て対応されていない. これから開発が進むことで, より柔軟にシステムに応じた通信の信頼性を担保できるようになると考えられる.

謝辞

本研究は, 「知の拠点あいち」重点研究プロジェクト (3 期) の支援で実施されたものである.

参考文献

[1] THE AUTOWARE FOUNDATION: Autoware.AI (online), 入手先 ⟨https://www.autoware.ai/⟩(2019.10.18).
[2] Open Source Robotics Foundation: GitHub - ros2/ros2: The Robot Operating System, is a meta operating system for robots (online), 入手先 ⟨https://github.com/ros2/ros2/⟩ (2019.10.18).
[3] Open Source Robotics Foundation: About Quality of Service Settings (online), 入手先 ⟨https://index.ros.org/doc/ros2/Concepts/About-Quality-of-Service-Settings/⟩(2019.10.18).
[4] 近藤豊: ROS2 ではじめよう次世代ロボットプログラミング , 株式会社技術評論社, 2019.p.52-p.54, p.59-p.62, p.127-p.136, (2019).
[5] Object Management Group: DDS Portal - Data Dis- tribution Services, Object Management Group (online), 入手先 ⟨http://portals.omg.org/dds/⟩ (2019.10.18).
[6] eProsima: eProsima Fast RTPS (online), 入手先 ⟨https://www.eprosima.com/index.php/products-all/eprosima-fast-rtps⟩ (2019.10.18).
[7] ADLINK: ADLINK Tech — Vortex Opensplice Data Distribution Service - ADLINK Technology (online), 入手先 ⟨https://www.adlinktech.com/en/vortex-opensplice-data-distribution-service.aspx⟩ (2019.10.18).
[8] RTI: Connectivity Software Framework for the Industrial IoT — RTI (online), 入手先 ⟨https://www.rti.com⟩(2019.10.18).
[9] TIME&SPACE: 2020 年には自動運転が実用化？ 日本初の 5G を活用した公道走行に成功 (online), 入手先 ⟨https://time-space.kddi.com/au-kddi/20190322/2603⟩ (2019.10.18).
[10] KDDI IoT ポータル: 国内初、5G 等を活用した複数台の遠隔監視型自動運転の実証実験の実施 | 活用事例 | KDDI IoT ポータル (online), 入手先 ⟨https://iot.kddi.com/cases/car5g/⟩ (2019.10.18).
[11] Maruyama, Yuya, Shinpei Kato, and Takuya Azumi: Exploring the performance of ROS2., Proceedings of the 13th International Conference on Embedded Software, ACM, (2016).
[12] 森田錬, and 松原克弥: ROS2 における通信特性に応じた DDS 実装の動的選択機構の実現 (コンピュータシステム)–(組込み技術とネットワークに関するワークショップ ETNET2018)., 電子情報通信学会技術研究報告= IEICE technical report: 信学技報 117.479 : 7-12, (2018).
[13] 水島知央, 神蔵貴久, and 大前学: 遠隔型自動運転システムにおける遠隔操作時の映像遅延が操舵の操作に与える影響の評価, 自動車技術会論文集 50.3 : 970-976 , (2019).
[14] 国立研究開発法人産業技術総合研究所: 平成 29 年度 高度な自動走行システムの社会実装に向けた 研究開発・実証事業： 専用空間における自動走行等を活用した 端末交通システムの社会実装に向けた実証 成果報告書, (2018).
[15] A. Garg.: Real-time Linux kernel scheduler, Linux Journal, 2009(184):2, (2009).