ROS2 リアルタイムの最新動向の紹介と、ROS2 への期待

SWEST22
2020/8/21

システム計画研究所
長谷川 敦史

資料目次

ISP

- ROS の基本的な話
- ROS 2 のリアルタイム性
- ROS 2 への期待

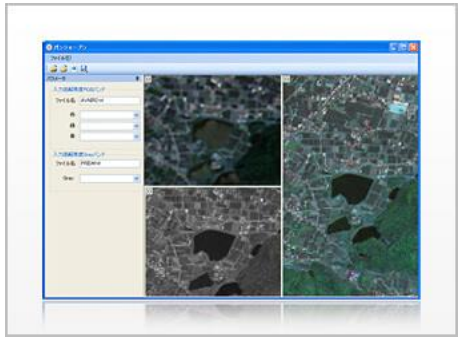
会社紹介

- （株）システム計画研究所（ISP）
  - 1977 年創業
  - 研究開発型のソフトウェア会社
- 事業分野

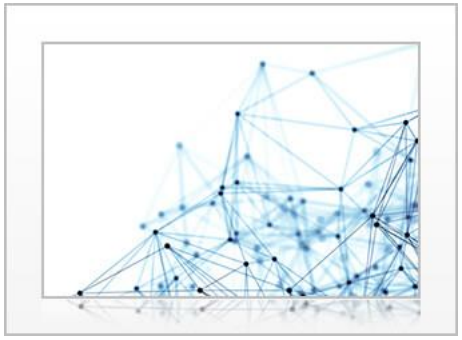
人工知能

医療情報

画像処理



宇宙・制御



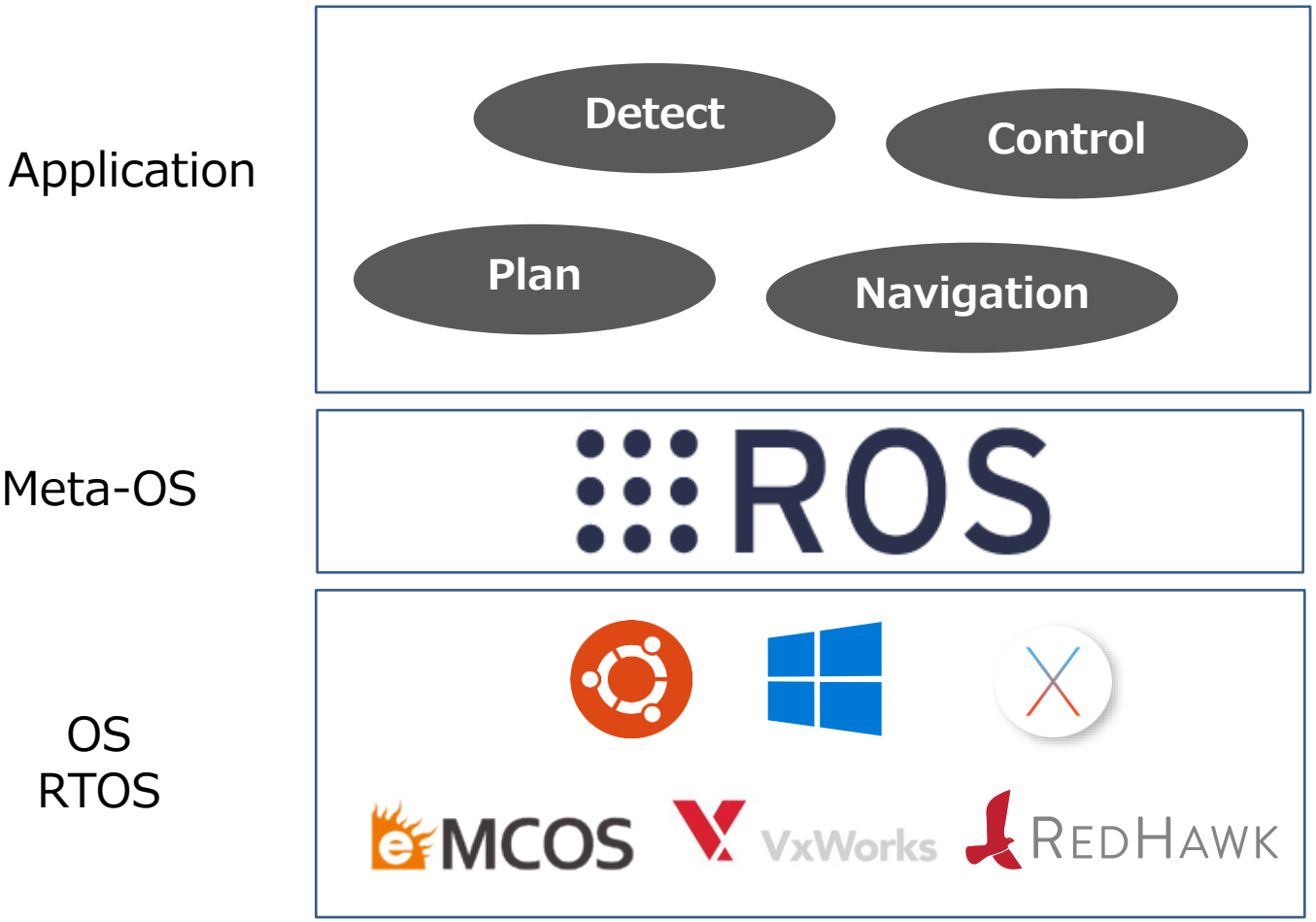
通信
ネットワーク

資料目次

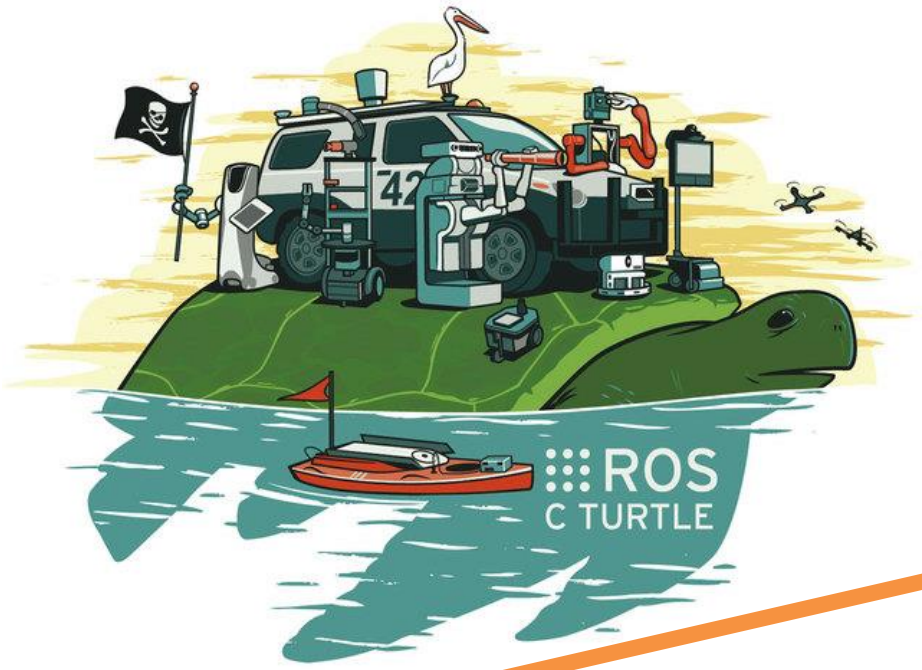
Robot Operating System (ROS) とは

ISP

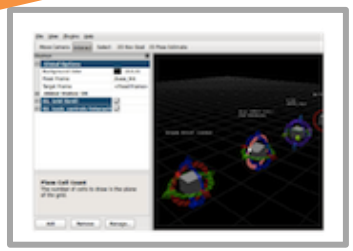
■ オープンソースで提供される
ロボット向けの メタ・オペレーティングシステム



ROS の構成要素



発表のメイン

=  +  +  + 

通信

ツール郡
シミュレーション
可視化

機能群
マニピュレータ
ナビゲーション

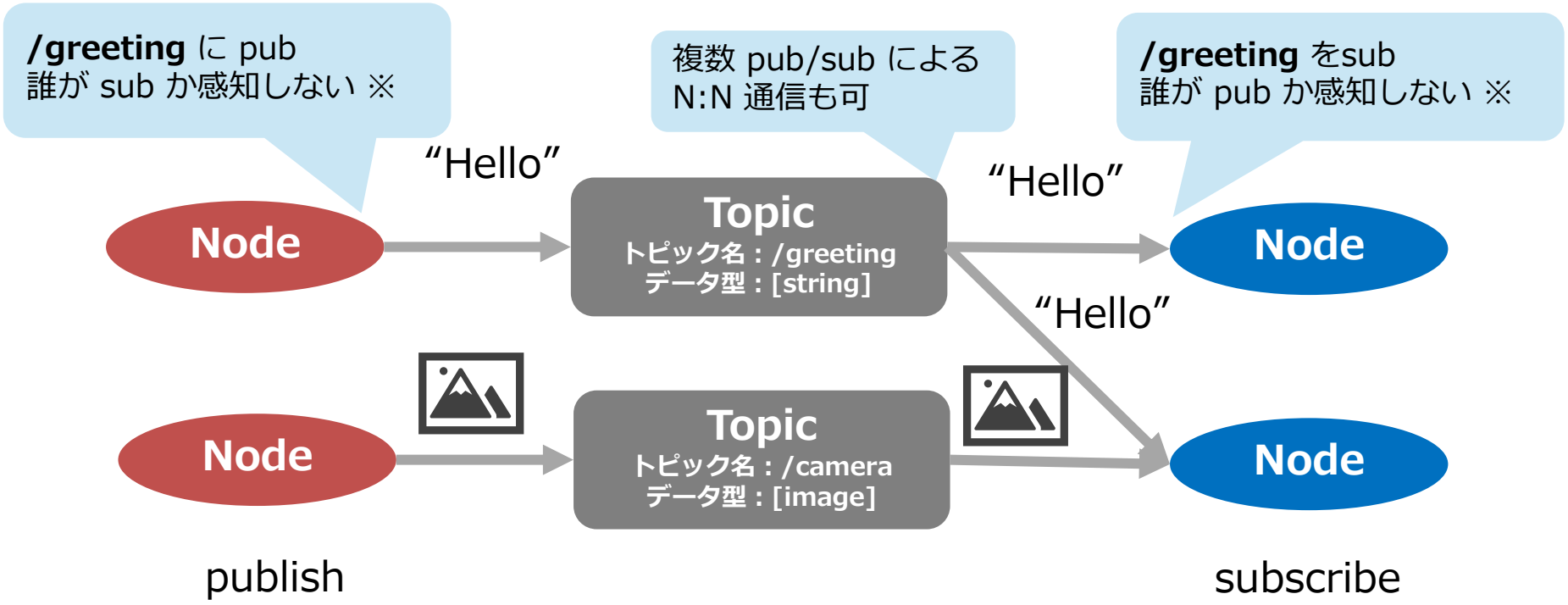
エコシステム
コミュニティ

[Open Robotics, ROS.org | About ROS]

ROS の通信

■ トピックを介した Pub/Sub 方式の通信を採用

Publish/Subscribe 型通信



※ ROS 2 ではノードの認証などのセキュリティ機能あり

ROS 1 と ROS 2

- 2025 年まで ROS 1 と ROS 2 のサポートが続く
  - 豊富な資産と実績のある ROS 1
  - 産業からのニーズに対応した ROS 2

API の変更あり。１から作り直している。
ROS2 Design | Why not just enhance ROS 1

2020 2021 2022 2023 2024 2025

Kinetic Kame (~2021/4)

ROS.org | Distributions

ROS 1

Melodic Morenia (~2023/5)

Noetic Ninjemys
ROS1 最後の LTS (~2025/5)

Dashing Diademata (~2021/5)

ROS Index | Distributions

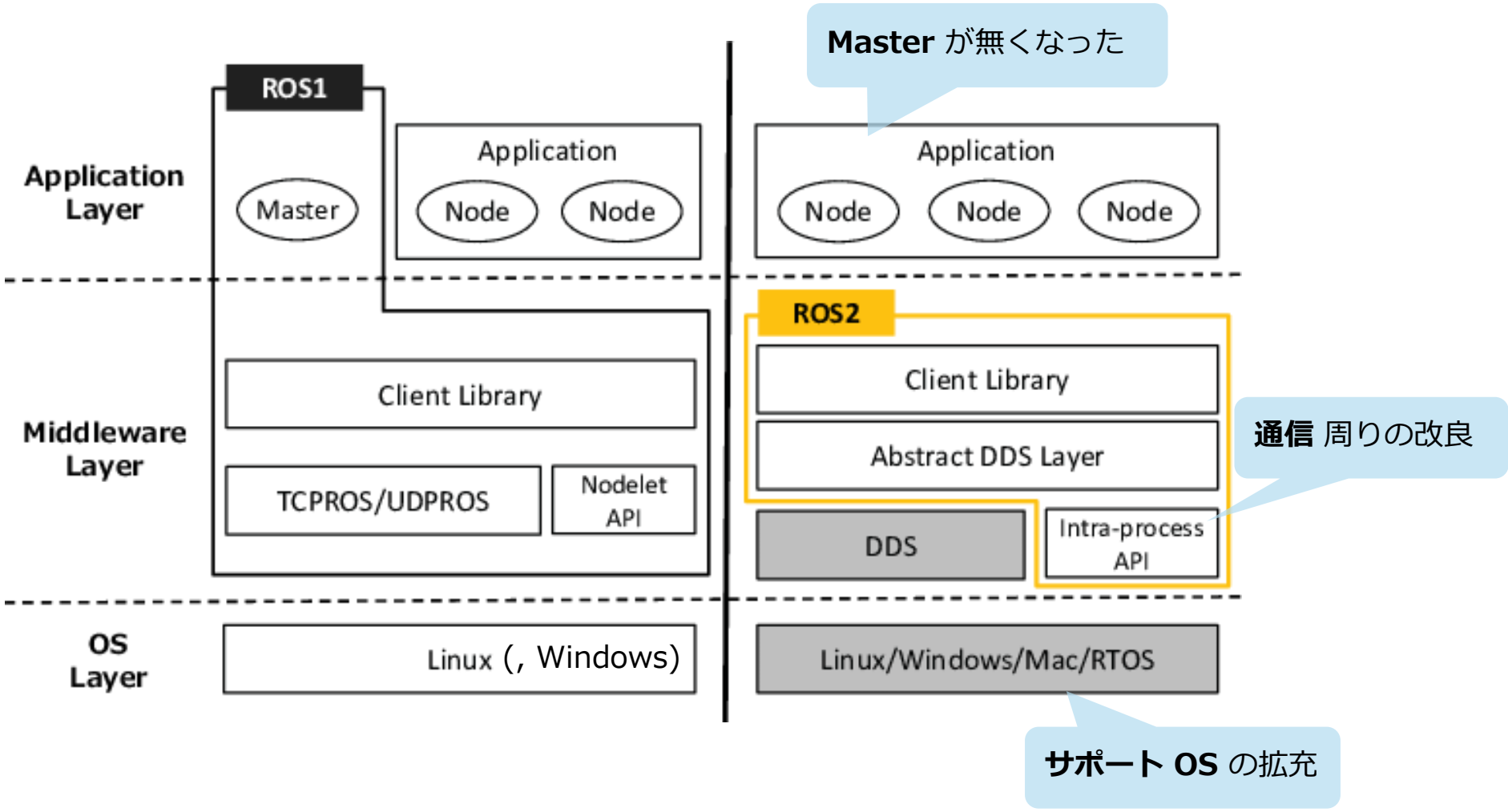
Eloquent Elusor (~2020/11)

ROS 2

Foxy Fitzroy (~2023/5)

今後もリリースが続く

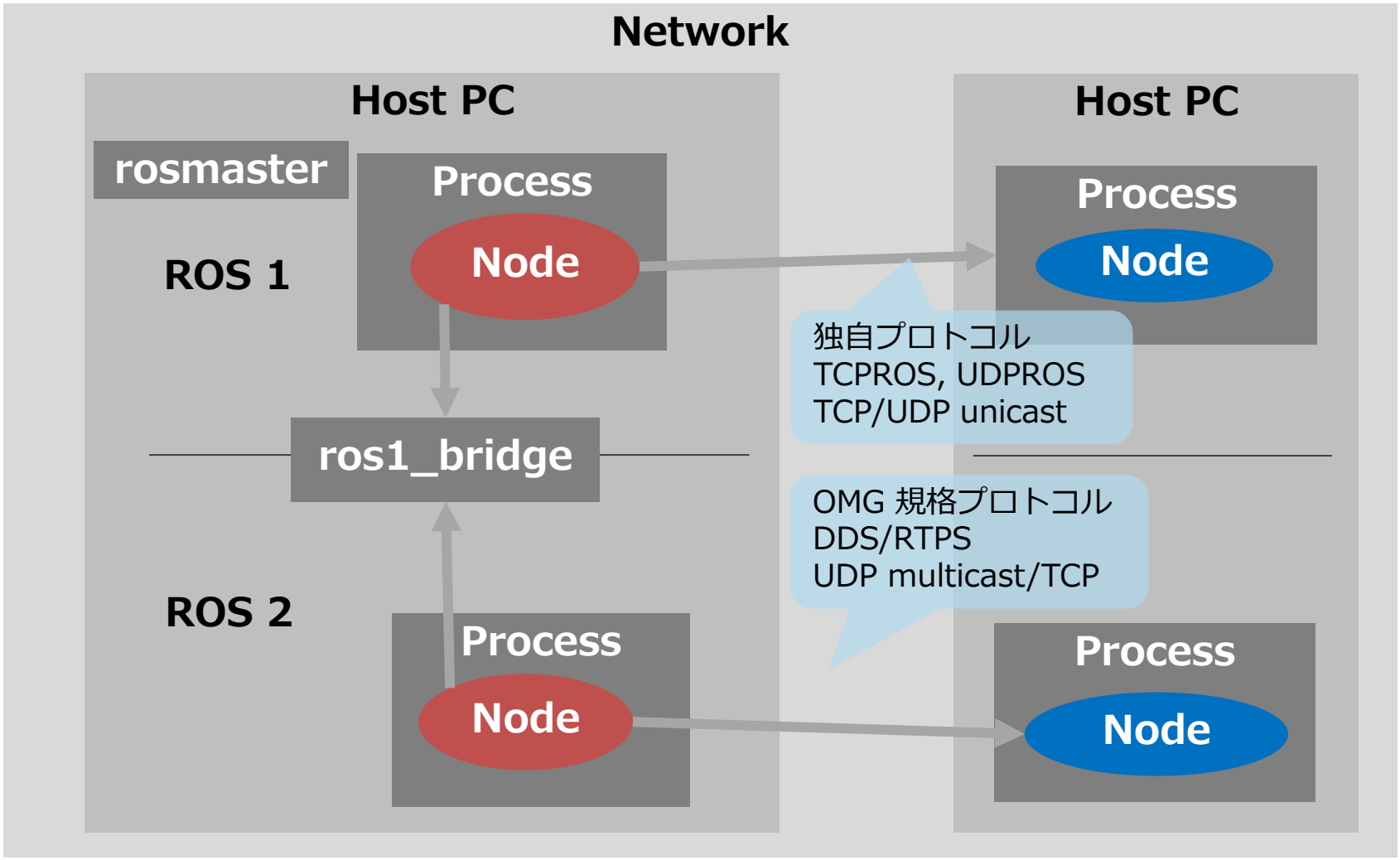
ROS 1 vs ROS 2 | アーキテクチャ



- Maruyama Yuya, Kato Shinpei, Azumi Takuya, Exploring the performance of ROS2, 2016.

ROS 1 vs ROS 2｜通信プロトコル

- ROS 2 は通信プロトコルを OMG 規格に変更
- ROS 1 と ROS 2 間の通信にはブリッジが必要



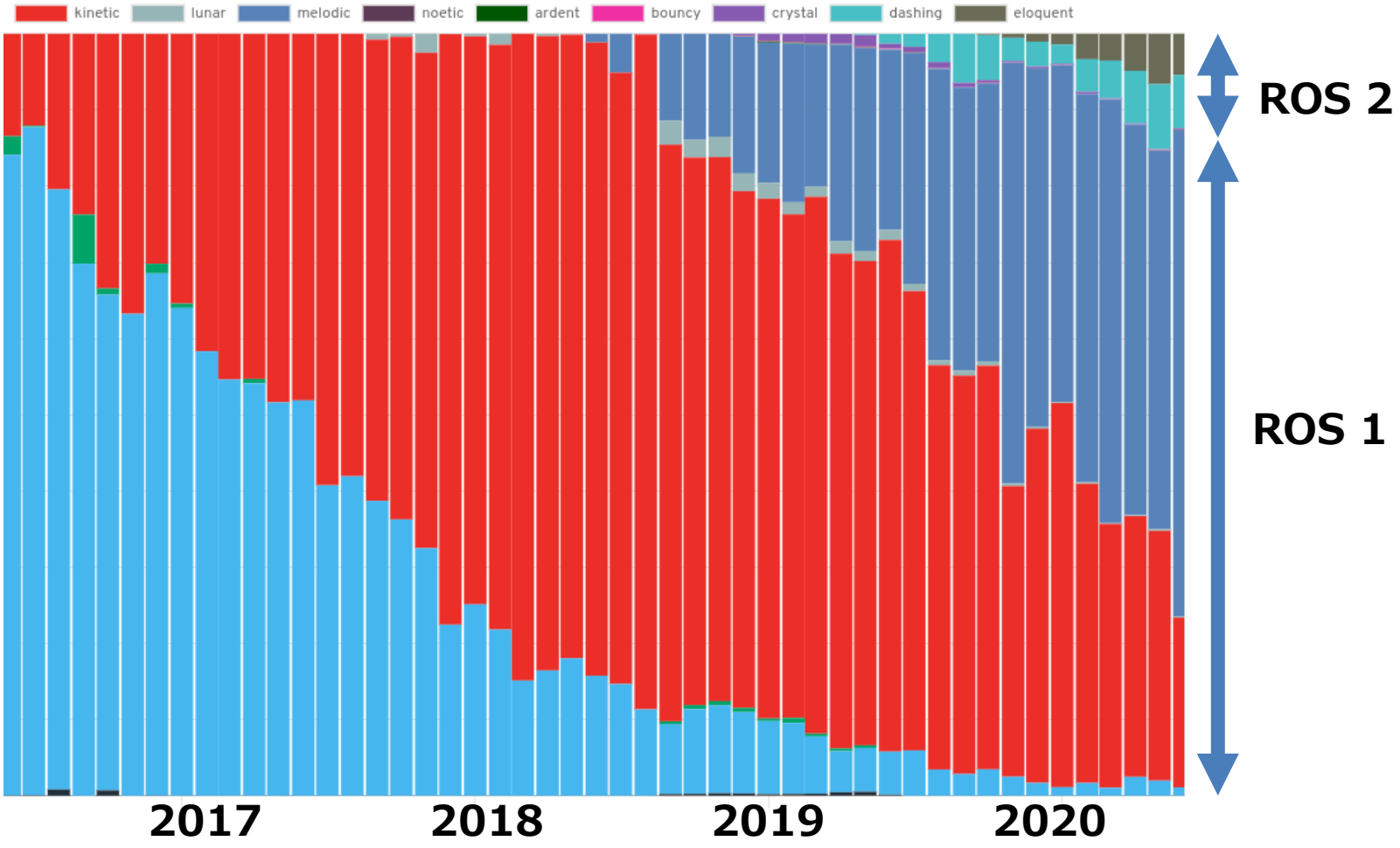
ROS 1 vs ROS 2 | ユースケース

ROS 1	ROS 2
▪ 研究用途 ▪ ロボット単体 ▪ 潤沢な計算資源 ▪ Ubuntu (/ Windows) ▪ 安定したネットワーク ▪ リアルタイム性は別の仕組みで対処	▪ 研究用途＋製品用途 プロトタイプから製品までの利用 ▪ ロボット単体～多数 ▪ 組み込み～潤沢な計算資源 ▪ Ubuntu / Windows / Mac ▪ 不安定な通信環境下での利用 ▪ **リアルタイム処理のサポート** （※ ソフトリアルタイム）

- ROSCon 2014 | Why You Want to Use ROS 2
- ROSCon 2016 | ROS2 Update
- ROS 2 Design | Why ROS 2?

ROS 1 vs ROS 2 | ダウンロード数

- ROS 1：ROS 2 = 9：1 （2020年時点）
- ROS 2 の利用はこれから増える



ROS Metrics | ROS Distro Usage by packages.ros.org traffic

ROS 1 vs ROS 2｜比較表

ISP

	ROS 1	ROS 2
特徴	研究での利用に対応 豊富な資産と実績	産業からのニーズに対応 分散システムなどのユースケースに対応可
公式リリース	2010 / 3 (Box Turtle)	2017/ 8 (Ardent Apalone)
通信方式	独自プロトコル XMLRPC, TCPROS, UDPROS rosmaster による単一故障点	OMG 規格プロトコル DDS / RTPS （各ベンダー実装の切替可） rosmaster がなくなった分散処理 不安定なネットワークへの対応
サポート OS	Ubuntu （公式サポート） Windows （Microsoft サポート）	Ubuntu, Windows, Mac （公式サポート） RTOS （各ベンダーサポート）
主なライセンス	3-clause BSD（商用利用可）	Apache 2.0（商用利用可）
主な開発言語	C++, Python	C++, Python （言語拡張が比較的容易）
Launch システム	XML	Python （ライフサイクルの追加） XML, yaml

- Github ms-iot | ROS on Windows
- ROS.org | Index of ROS Enhancement Proposals
- ROS 2 Design | Changes between ROS 1 and ROS 2

資料目次

ISP

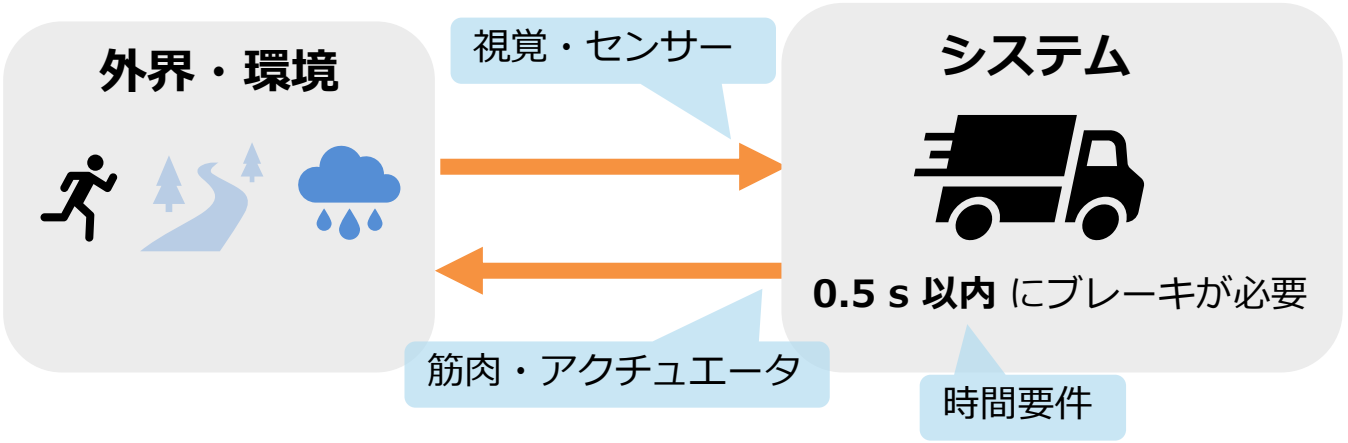
- ROS の基本的な話
- **ROS 2 のリアルタイム性**
- ROS 2 への期待

リアルタイム性とは

デッドライン

■ システムが定められた時間要件 を満たして動作する性質

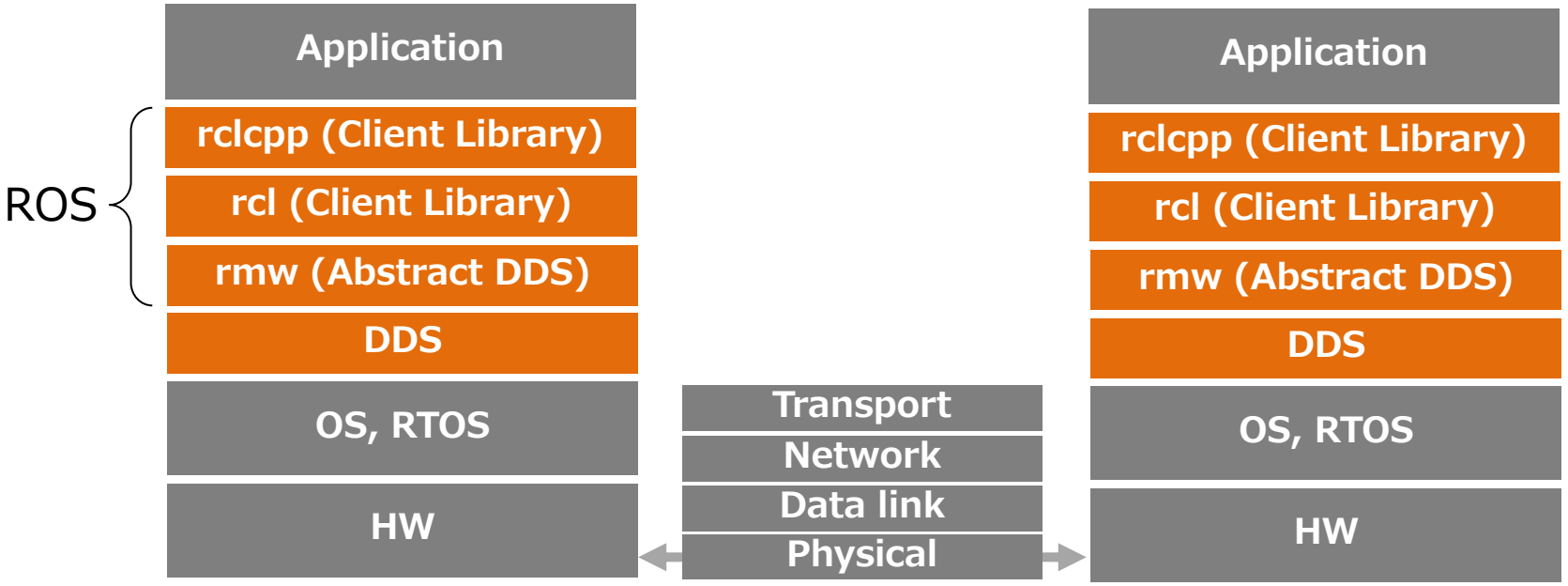
リアルタイムが求められる例（車の運転）



リアルタイム性に関する発表のスコープ

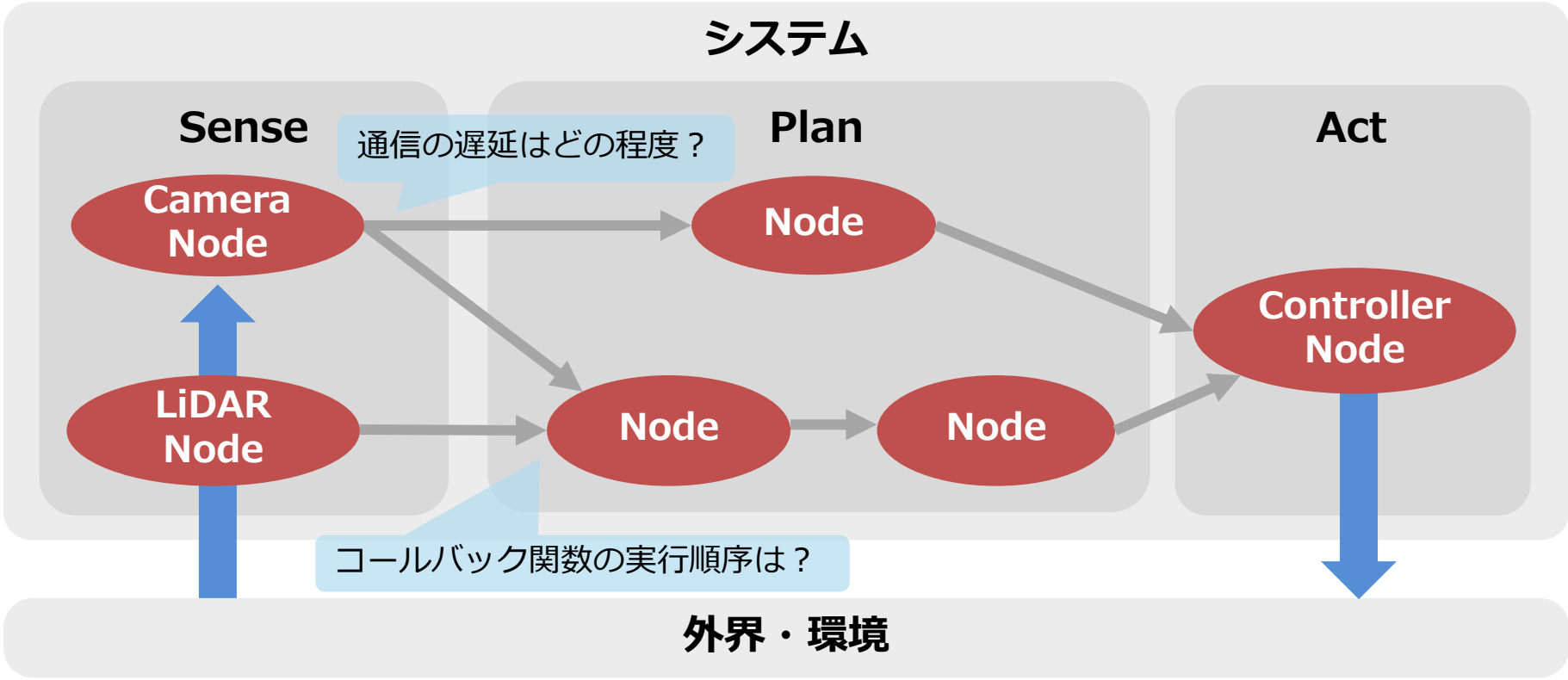
ISP

- 話題：より厳しいリアルタイム性、高速化
- 各レイヤーでリアルタイムである必要がある
  本資料の対象： ROS と DDS のレイヤー、C/C++



ROS のリアルタイム性？

- end-to-end の処理に要する時間が時間要件以内
  例：センサ入力からアクチュエータの出力まで
  - ノード間の **通信** ← 後半のメイントピック
  - コールバックの **スケジューリング**

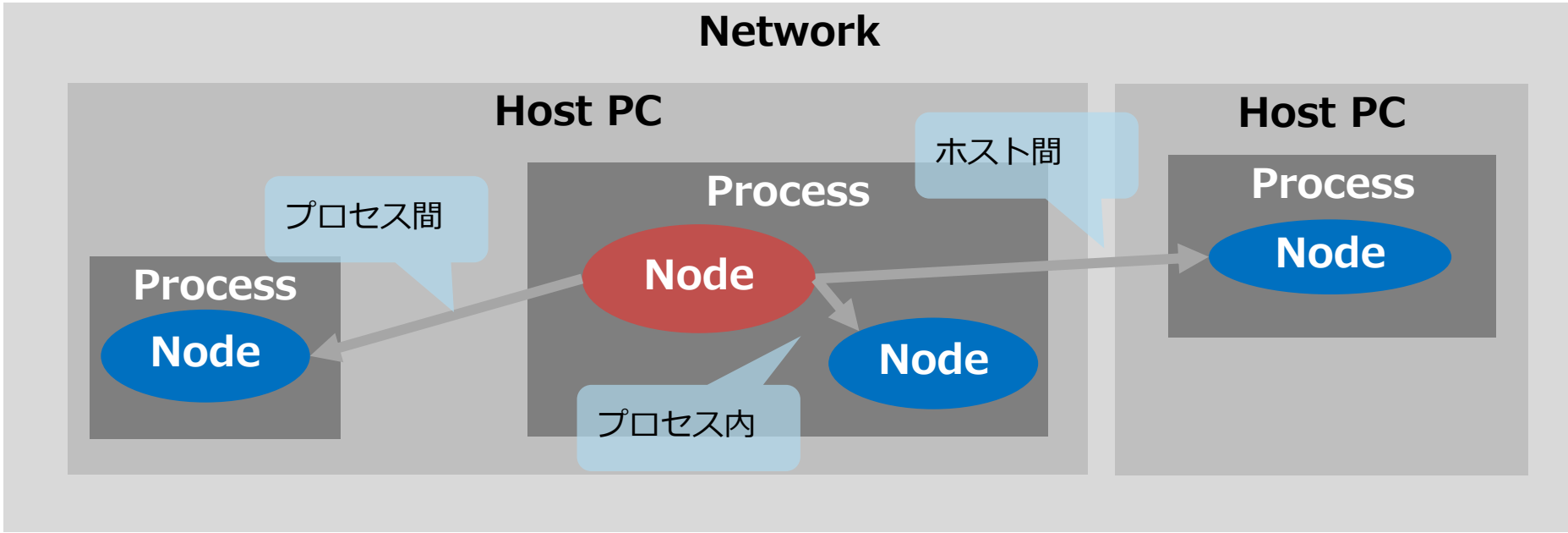


通信のリアルタイム性

ISP

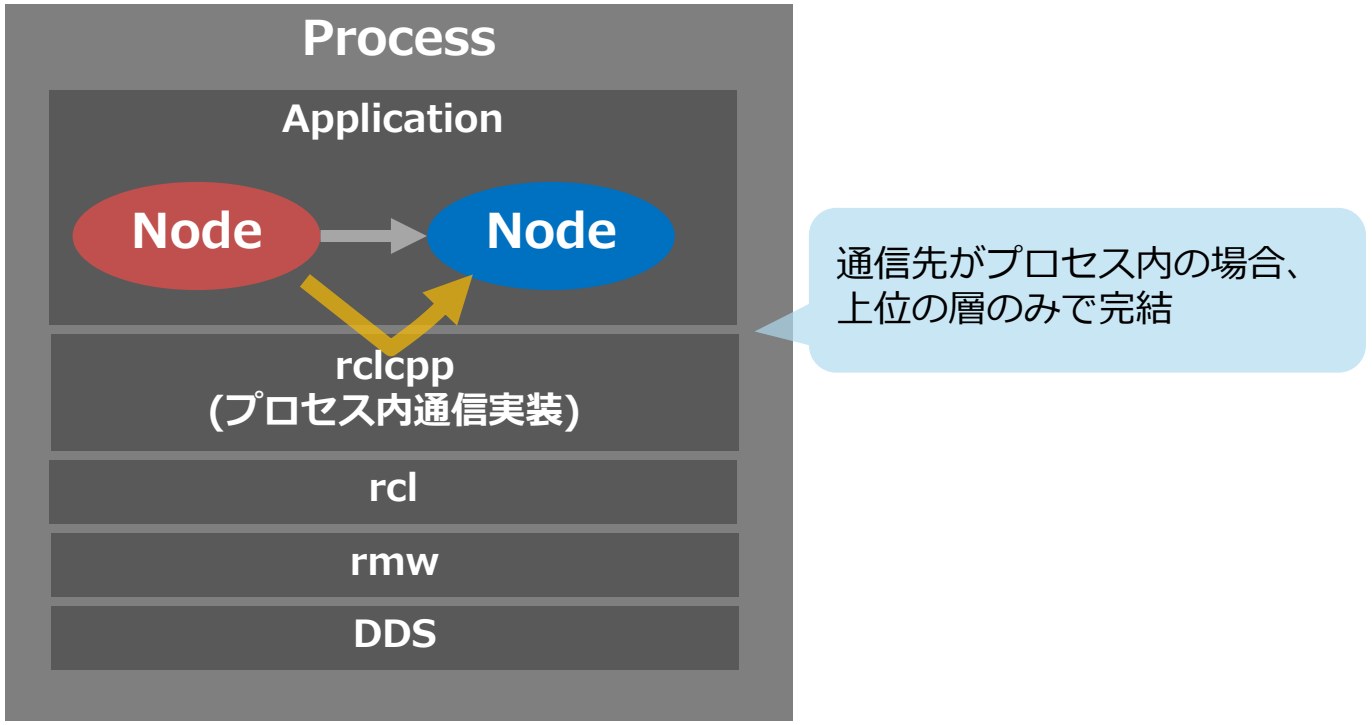
- 複数の通信手段が提供されている
  - プロセス内
  - プロセス間
  - ホスト間

論理的に近いノード間の通信ほど
高速・高リアルタイム性



通信のリアルタイム性｜プロセス内通信

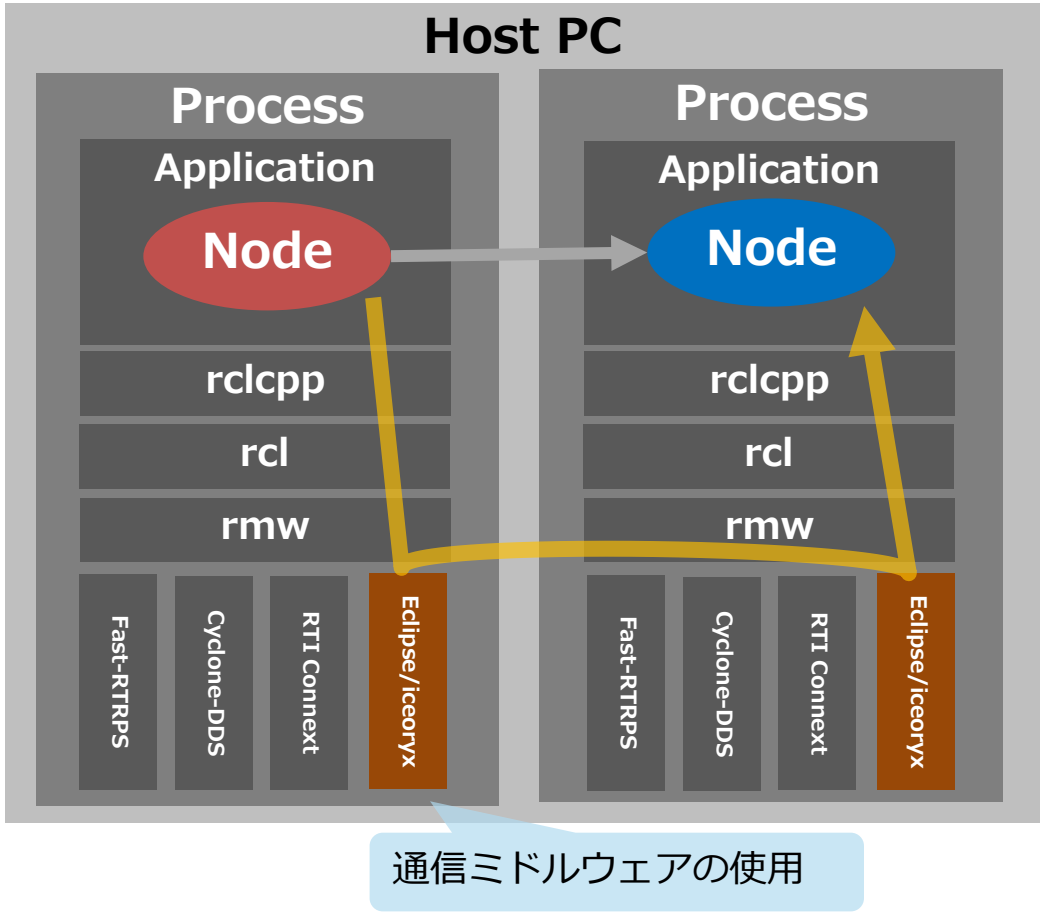
■ プロセス内：ポインタ渡し

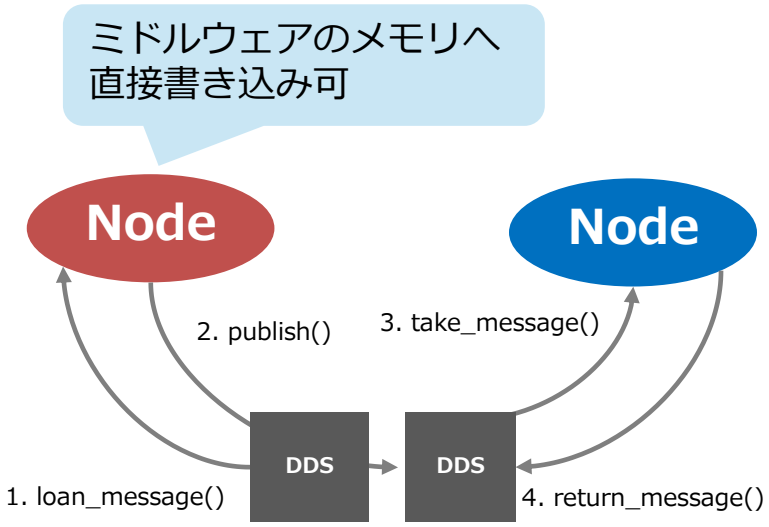


通信のリアルタイム性｜プロセス間通信

高速化

- プロセス間：共有メモリを含む通信





zero copy API

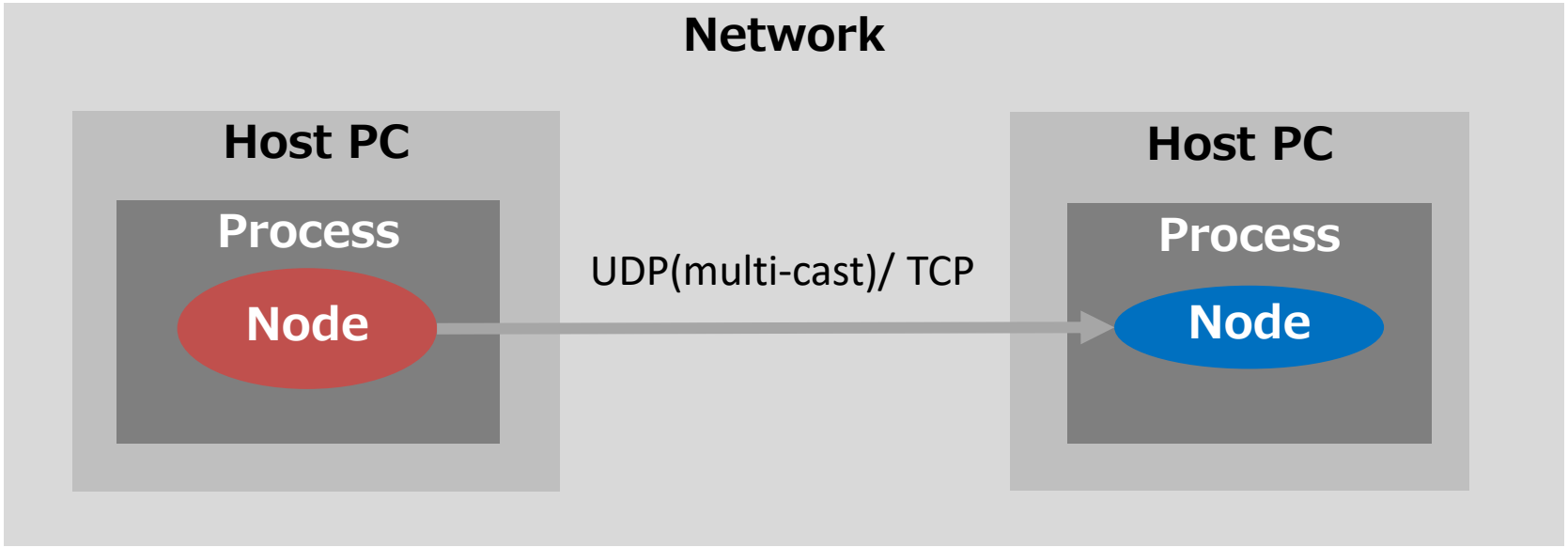
- Karsten Knese, Michael Pöhnl, A true zero copy RMW implementation for ROS2, ROSCon 19
- Fujita Tomoya, Eclipse Iceoryx Overview, ROS Japan UG #34 LT大会

通信のリアルタイム性｜ホスト間

- ホスト間
  - UDP マルチキャスト
  - Quority of Service (QoS)
    - デッドラインミス時にコールバック実行

subscriber の増加による影響小

[QoS の例]
History = KEEP_ALL / KEEP_LAST
Reliability = BEST_EFFORT / RELIABLE
Deadline = 守るべき最小周期
…

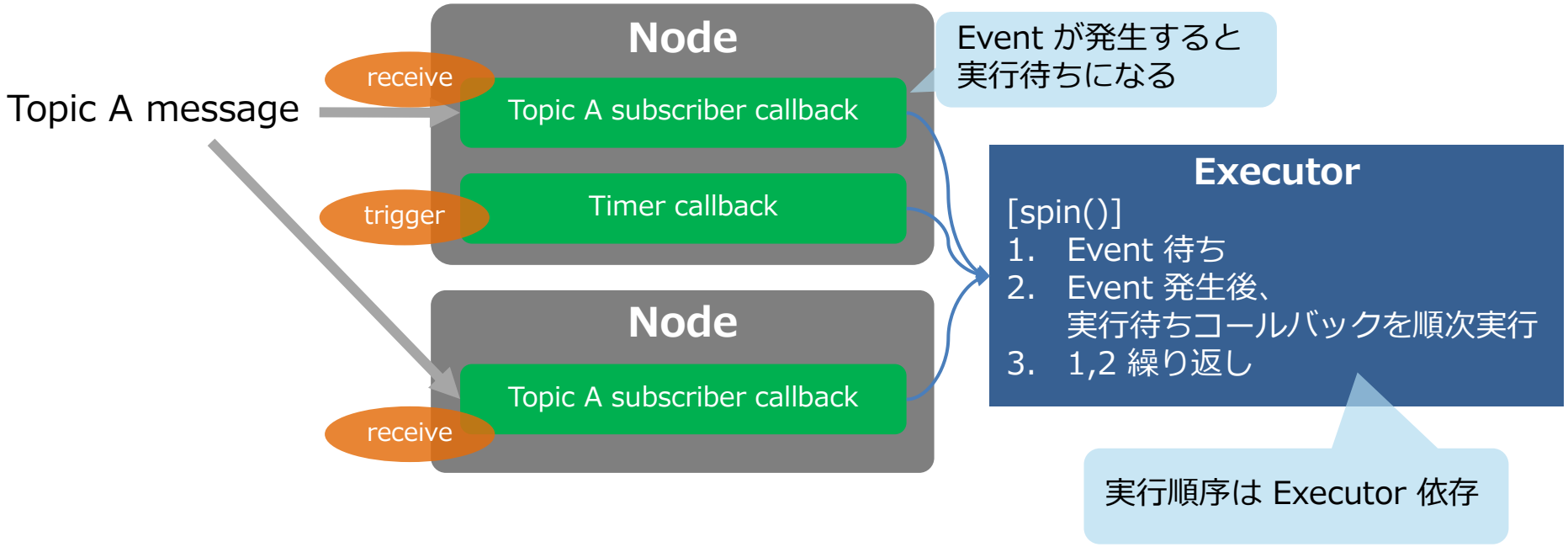


QoS を試すためのデモ

- GitHub - eProsima/ShapesDemo, RTI |Interactive Shapes Demo
- Emerson Knapp, Nick Burek , Quality of Service Policies for ROS2 Communications, ROSCon 19.

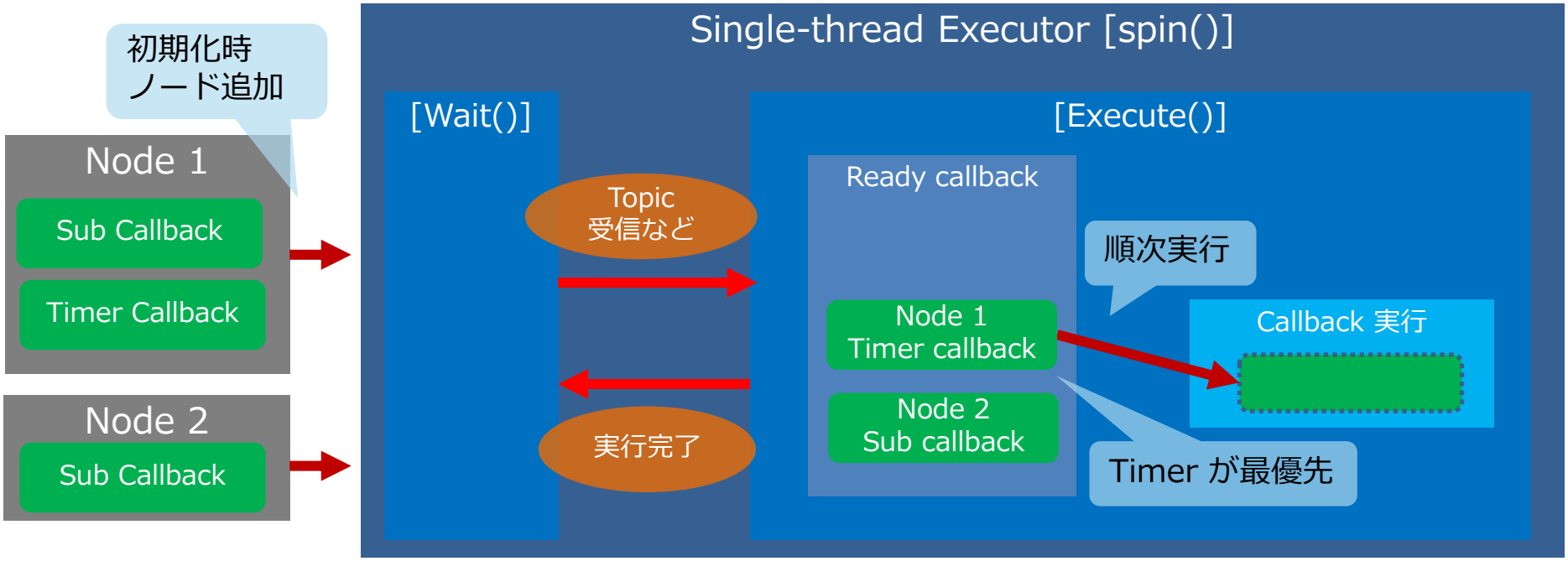
コールバックのスケジューリング

- Executor がコールバックのスケジューリングを担っている。
  - Single (static) thread executor (C++)
  - Multi thread executor (C++)
  - Cbg executor (C++)
  - RCLC executor (C)
  - ThreadedCallback (C++)



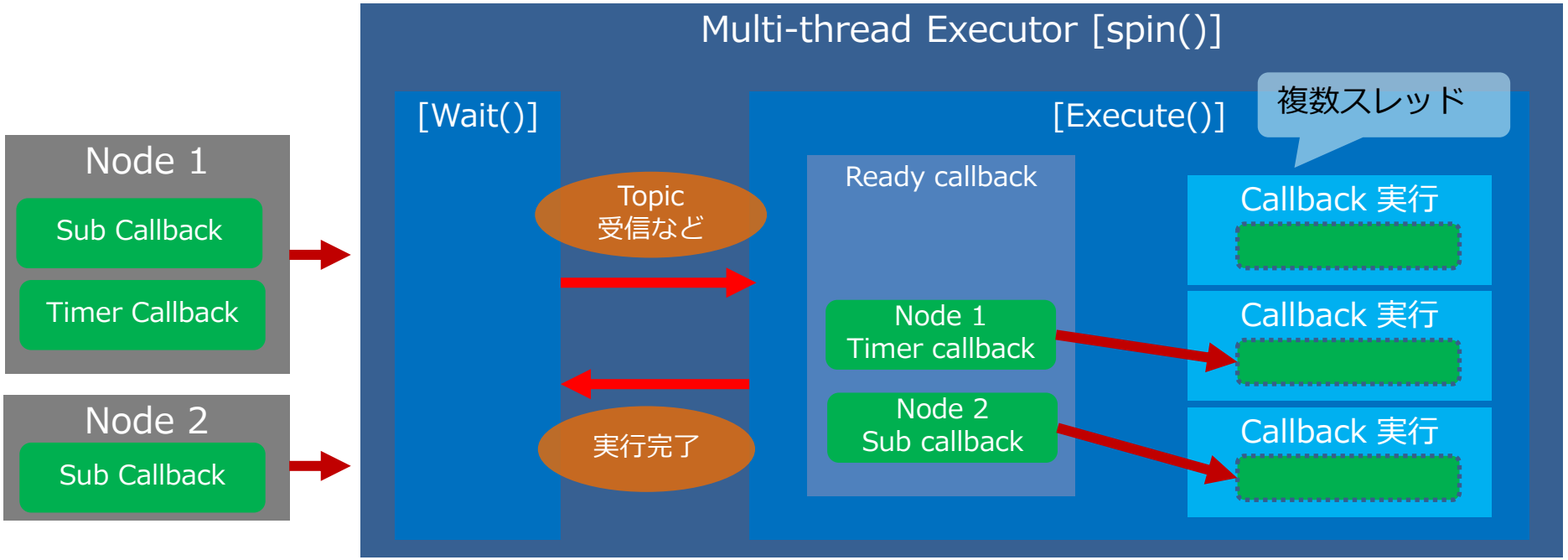
Single-thread Executor

- 特徴・長所
  - イベントによる起床後、順次コールバック実行
    （基本的な Executor ）
  - シングルスレッド
  - Foxy では CPU 利用効率を改善した
    Static-single-threaded Executor が取り込まれた
- 短所
  - コールバック実行中のイベントは他のコールバックに待たされる
  - コールバックの優先度固定・非プリエンプティブ



Multi-thread Executor

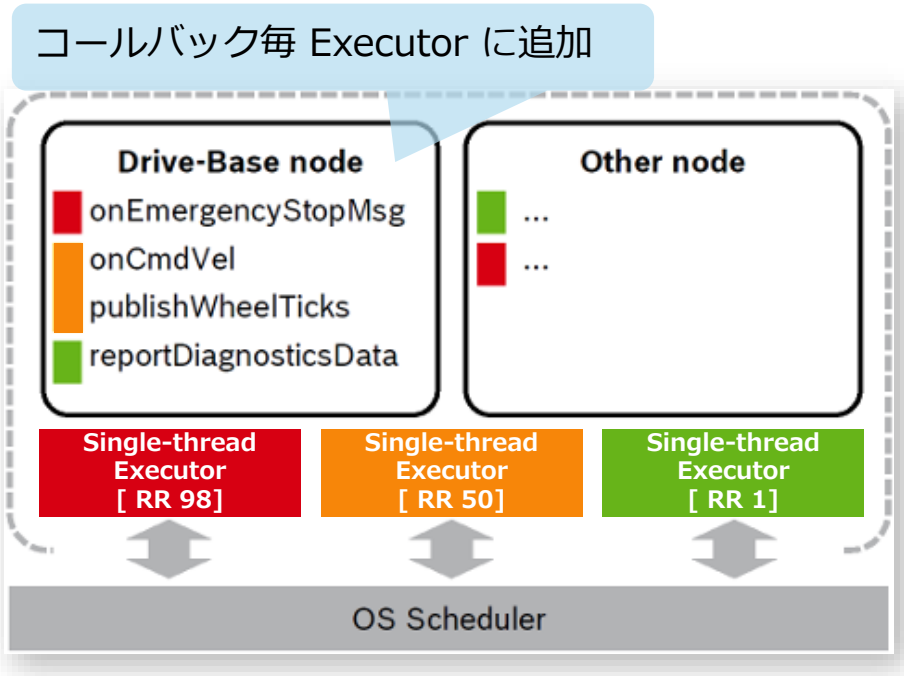
- 特徴・長所
  - イベントによる起床後、順次コールバック実行
  - 同一 Node のコールバック実行に排他処理を設定可能
- 短所
  - コールバック実行中のイベントは他のコールバックに待たされる
  - コールバックの優先度固定・非プリエンプティブ

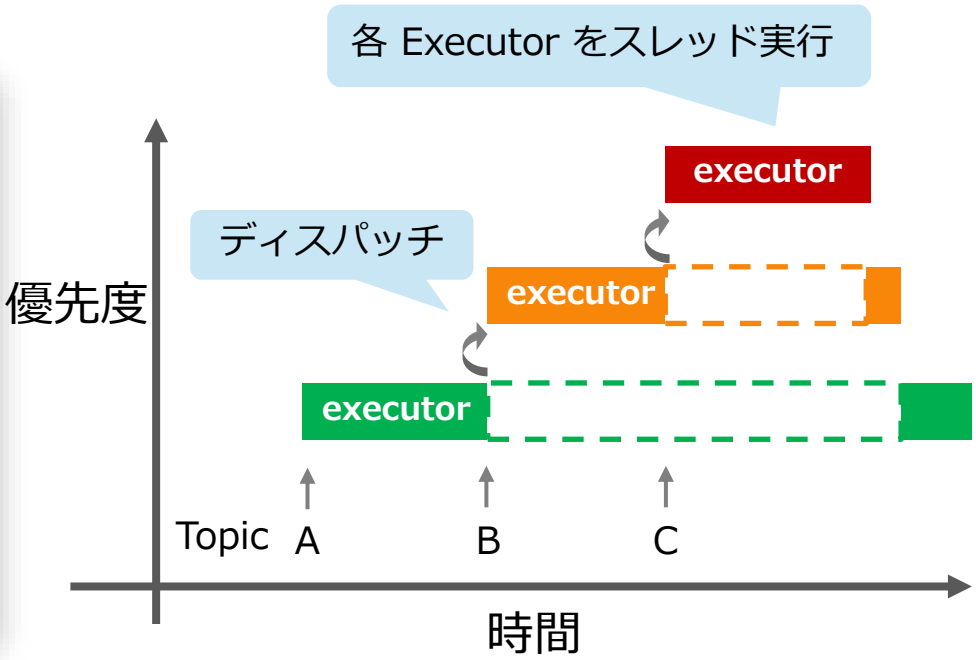


Callback-group-level Executor [micro-ROS] 26

本家 ROS2 からのフォーク。
マイコンで動作させて利用。
NuttX などの RTOS をサポート。

- 特徴・長所
  - コールバックのグループ毎に優先度付け
  - プリエンプティブ
- 短所
  - ユーザーによる排他処理・同期処理の記述が必要

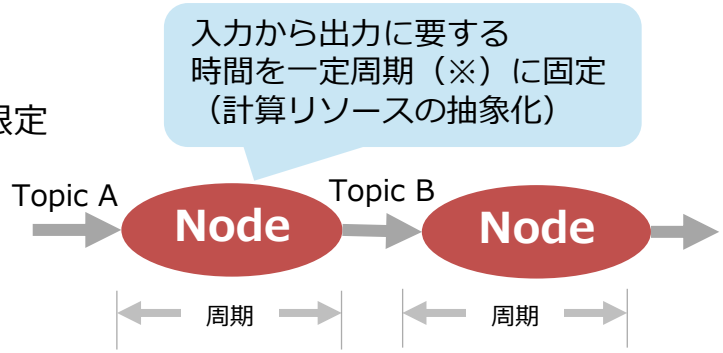


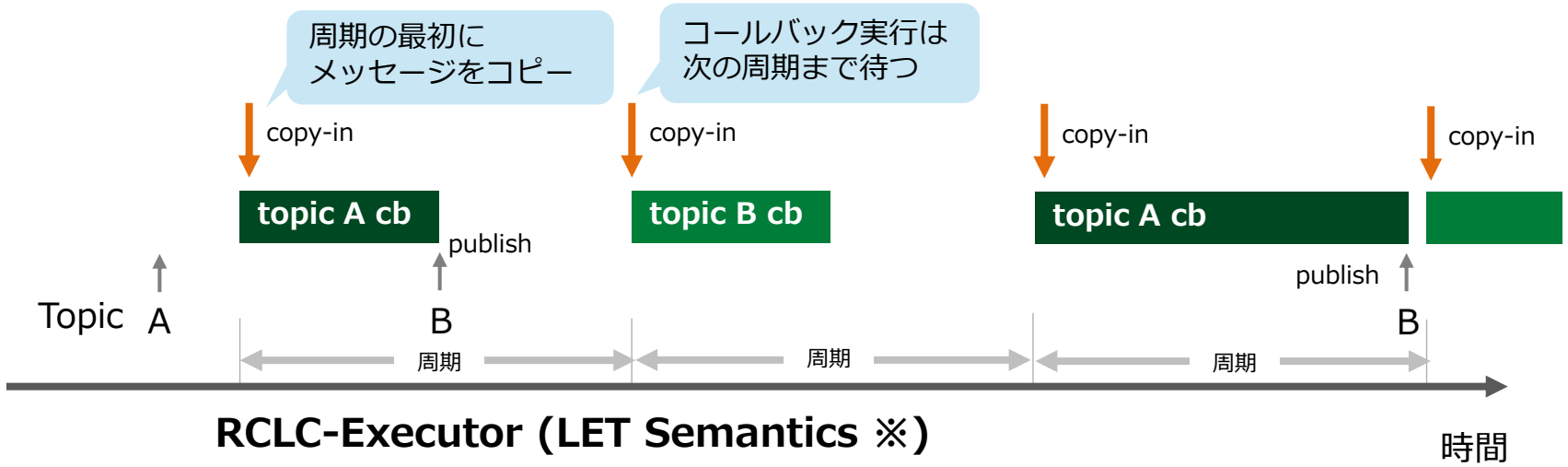


- micro-ROS | Callback-group-level Executor

RCLC-Executor [micro-ROS]

- 特徴・長所
  - コールバックの実行順序を設定可
  - Logical Execution Time (LET) semantics が使用可
    入力のタイミングを周期の開始時に入力に固定（※）
    - Event で起床するコールバックも 1 callback / 周期に限定
    - リアルタイム性の検証が比較的容易
- 短所
  - シングルスレッド動作
  - 非プリエンプティブ





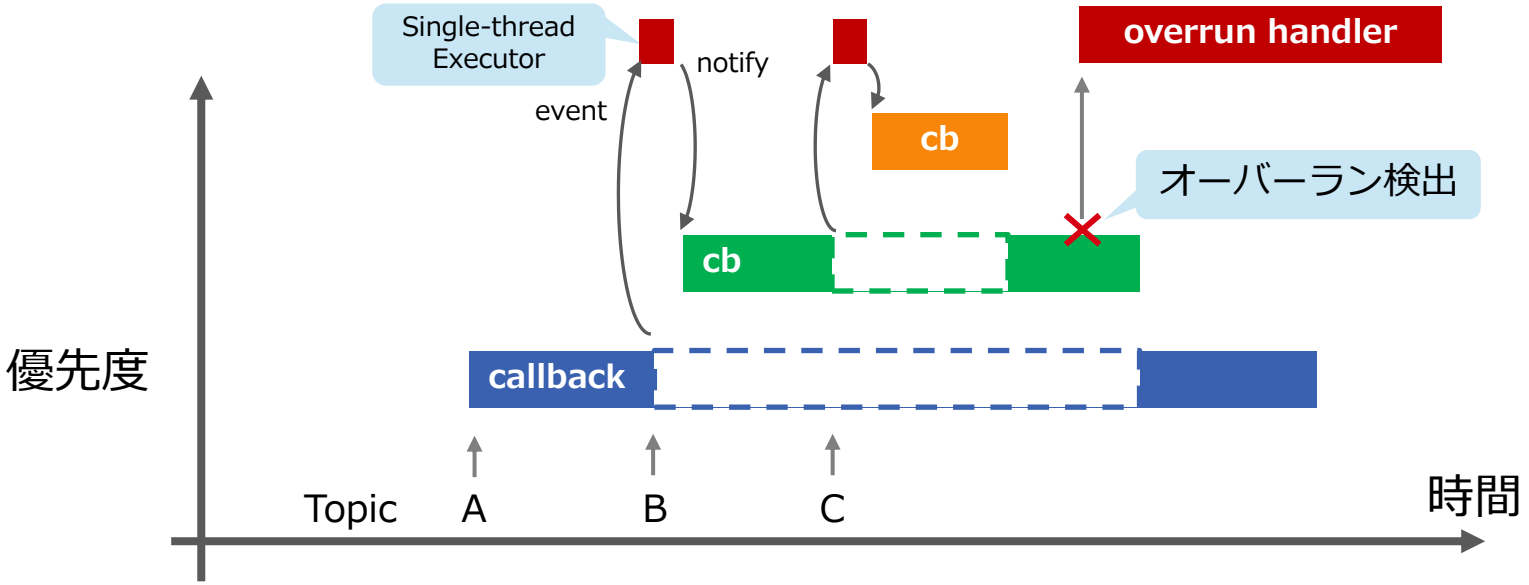
※ RCLC-Executor の実装に近い説明にしています。
オリジナルの LET が定めるのは周期開始時の入力(copy-in)と終了時の出力(copy-out)。

- micro-ROS | RCLC-Executor
- Christoph M. Kirsch, Ana Sokolova, The Logical Execution Time Paradigm, 2012.
- T. A. Henzinger, B. Horowitz and C. M. Kirsch, Giotto: a time-triggered language for embedded programming, 2003.

ThreadedCallback

現在は PoC 実装

- 特徴・長所
  - 各コールバックをスレッド化
    - コールバック毎に CPU ・優先度を割り当て可能
    - プリエンプティブ。スケジューリングを OS に委任
  - コールバックのオーバーランハンドラを追加
- 短所
  - ユーザーによる排他処理・同期の記述



- Y. Okumura, Threaded Callback with priority, affinity and overrun handler, ROS Discourse, 2020.

ROS 2 リアルタイム まとめ

ISP

- 更に厳しいリアルタイム性に向けて関連した話題を紹介
  - 高速な通信方式
  - デッドライン・オーバーランの検出
  - リアルタイム処理に優れたスケジューリング

- DDS や Executor 単体/複合 での評価・改良が必要。
  現状、個別の評価が進んでいる状態。
  - 我々のチームでは Executor の評価・提案を進めています
    - A. Hasegawa, ROS2 generated child thread scheduling policy affects timers, ROS Discourse, 2020.
    - Y. Okumura, Experiment to inhibit DDS and ROS2 child threads, ROS Discourse, 2020.
    - Y. Okumura, Threaded Callback with priority, affinity and overrun handler, ROS Discourse, 2020.

- リアルタイムは TOPPERS など日本の得意とするところ。
  是非日本勢も contribute しよう！

資料目次

ISP

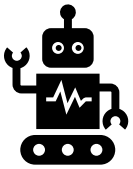
- ROS の基本的な話
- ROS 2 のリアルタイム性
- **ROS 2 への期待**

ROS 2 に抱いている期待｜共同開発

ISP

- ROS の目標
  - ロボットソフトウェアの共同開発を全世界的に推進する
- 利用者としてのメリット
  - 労働・コスト・時間 の分散
  - 知識・知恵・経験 の共有

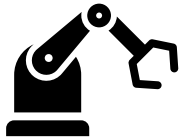
幅広い技術のうち、
専門分野で活躍すれば良い！



自律移動

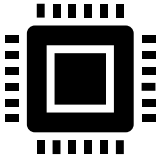
自己位置認識

センサー



マニピュレータ

画像認識・AI

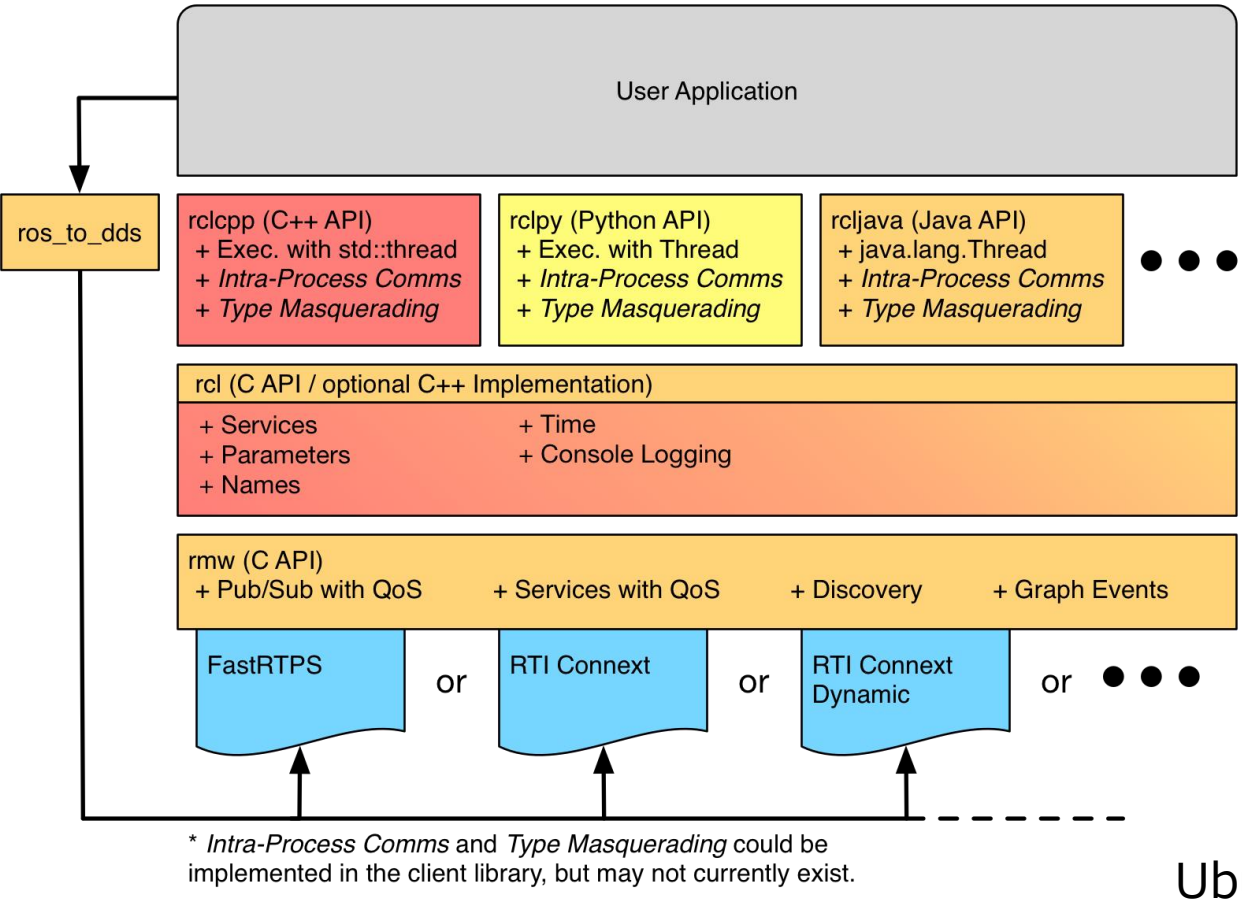


ハードウェア

■ ■ ■

ROS 2 に抱いている期待｜拡張性

- 拡張性のあるアーキテクチャ
  - コミュニティーによる拡張が可能



言語の拡張性

通信の拡張性

クロスプラットフォーム

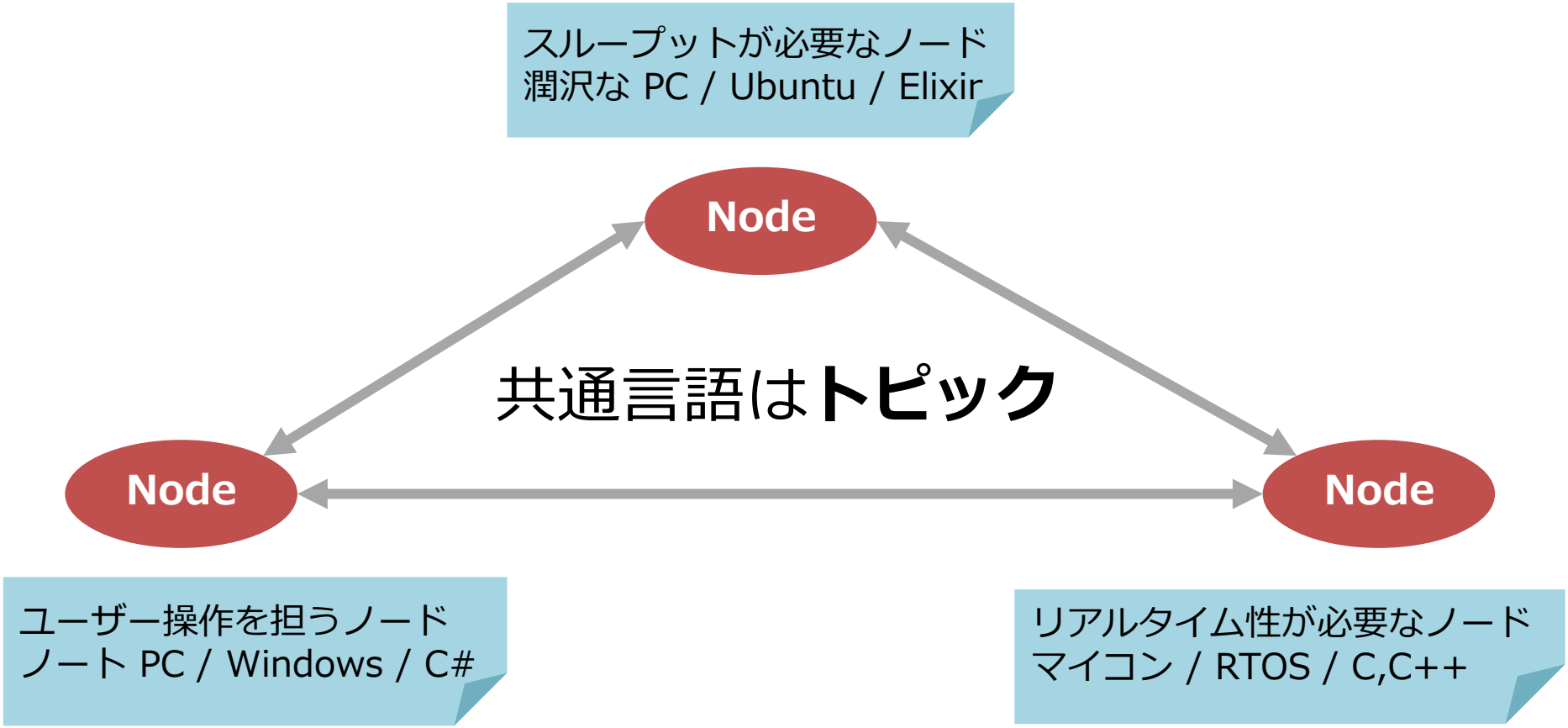
Ubuntu / Win / Mac / RTOS

[ROS 2 Overview | Internal API Architecture Overview]

ROS 2 に抱いている期待｜実装の抽象化

ISP

■ ハード/OS/言語 を抽象化する
⇒ ノードごとの目的に合わせた実装が可能



ROS 2 に抱いている期待｜アプリとハードの分離 34

- ROS の抽象化はハードとアプリの開発を分離する
  - 抽象化により、それぞれの開発が促進される

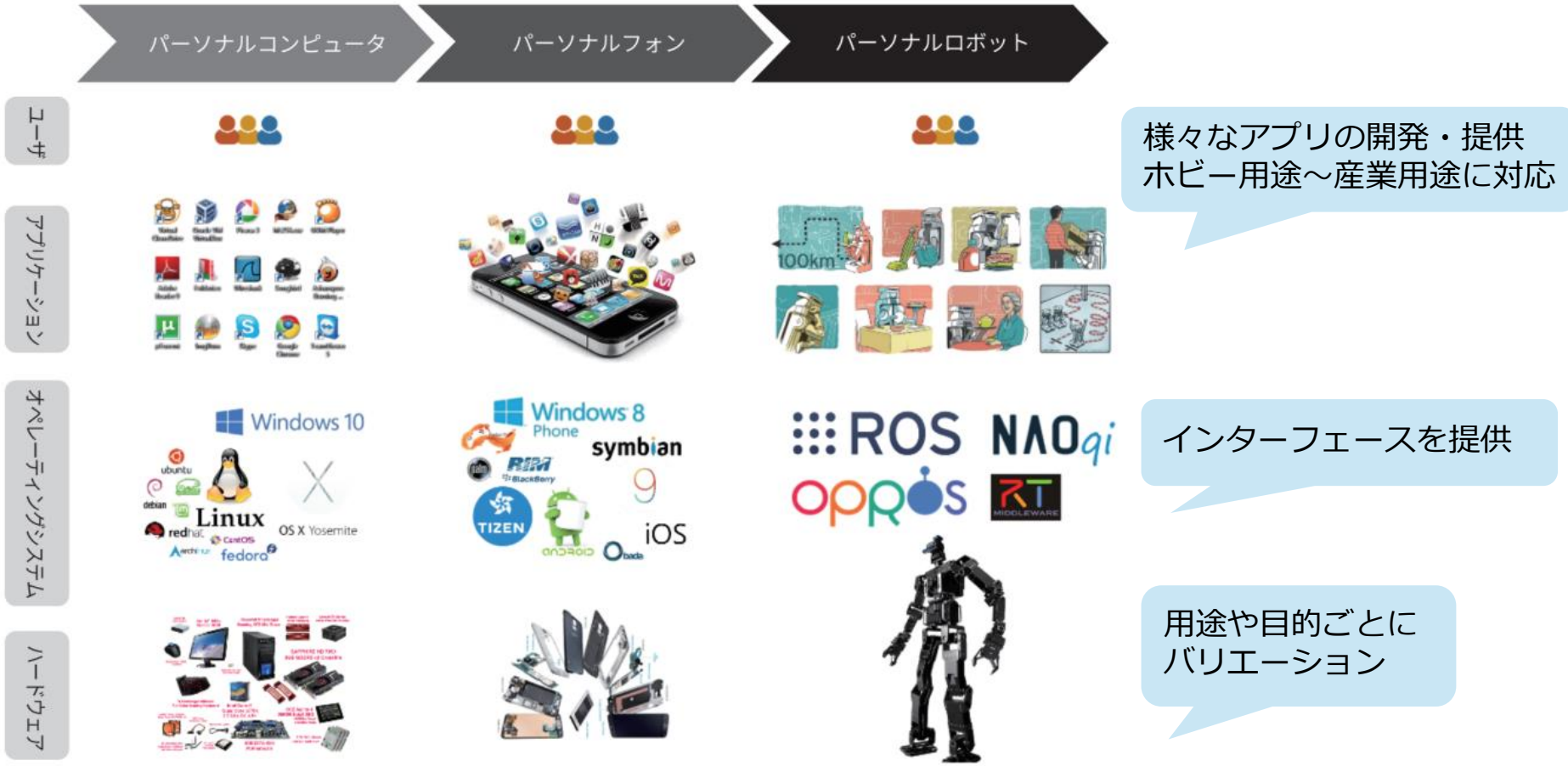
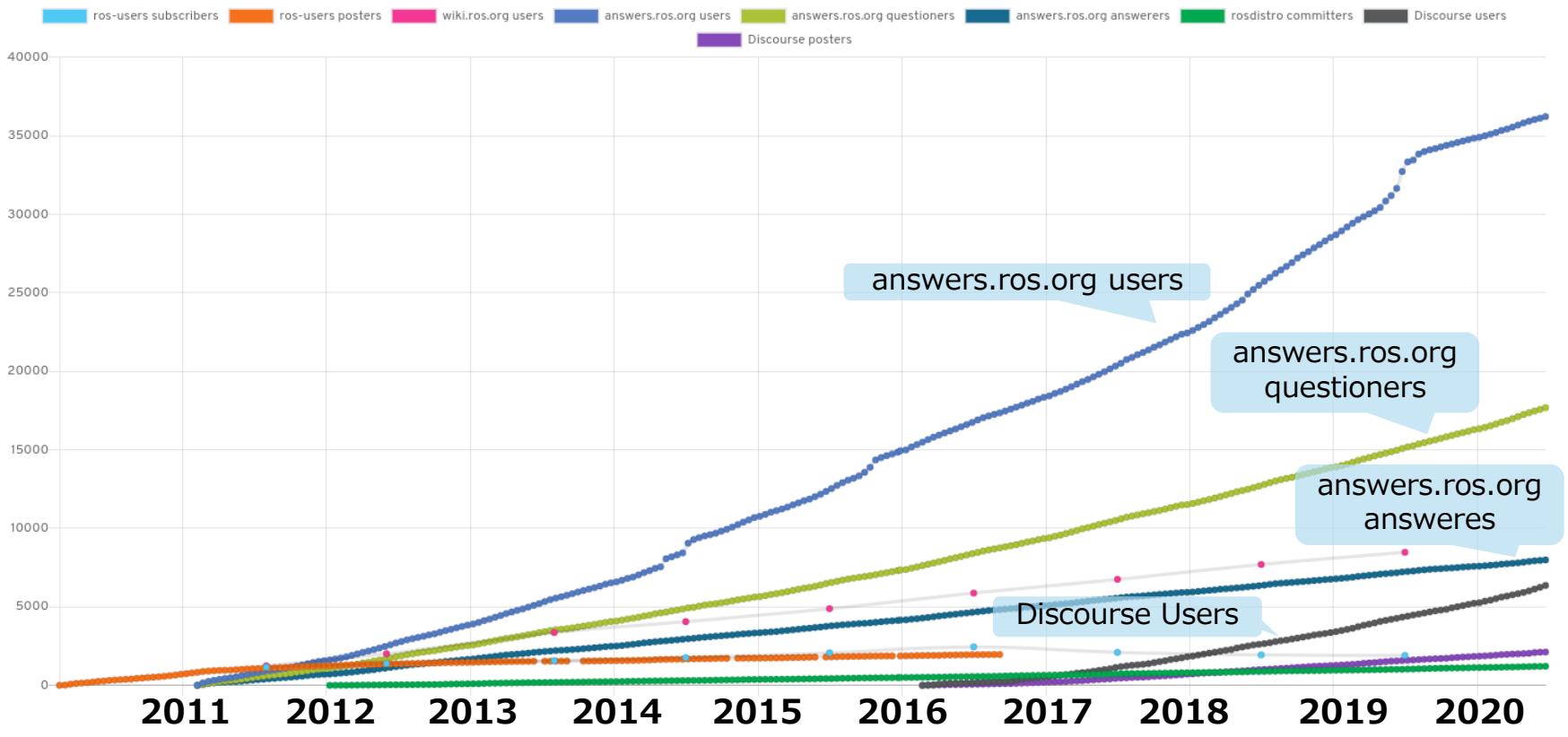


図 1.2　パーソナルコンピュータやスマートフォンエコシステムの 4 大構成要素

■ 表 允晳，倉爪 亮,鄭 黎螭, ROSロボットプログラミングバイブル, オーム社, 2018年, p2.

ROS 2 に抱いている期待｜コミュニティの拡大　35

- ROS のユーザーは年々増え続けている
- ROS を使った事例、新しい機能、関連サービスはこれからも増えていく



[ROS Metrics | Numbers of ROS Users]

Open Robotics のスポンサー

ISP

open robotics

ROS の開発・運営・管理

Products

ROS

GAZEBO

[Open Robotics | Sponsors]

Apex.AI

amazon

arm

BOSCH
Invented for life

DARPA

HITACHI
Inspire the Next

NASA

U.S. NAVAL RESEARCH LABORATORY

Office of Naval Research
ONR
Science & Technology

TARDEC

TOYOTA
RESEARCH INSTITUTE

ROS-Industrial

ISP

ROS の産業利用を目指すオープンソースプロジェクト

■ Members

ROS-Industrial | Current Members

最後に

- ROS 2 は産業のニーズが組み込まれた次世代 ROS
  機能や性能の改良が日々されている。
  世界的な共同開発によりロボット開発は更に推進される。

- ROS コミュニティーへ参加
  - ROS 2 の開発に参加
    - **Discourse** で議論
    - **ROS Answers** で質問・質問への回答
  - パッケージへの貢献
    - 自作パッケージの公開
    - Issue, PR 投稿
  - イベントに参加
    - **ROSCon**, **ROSCon JP**, 勉強会など
    - **ROS Japan Users Group** で国内イベントをチェック

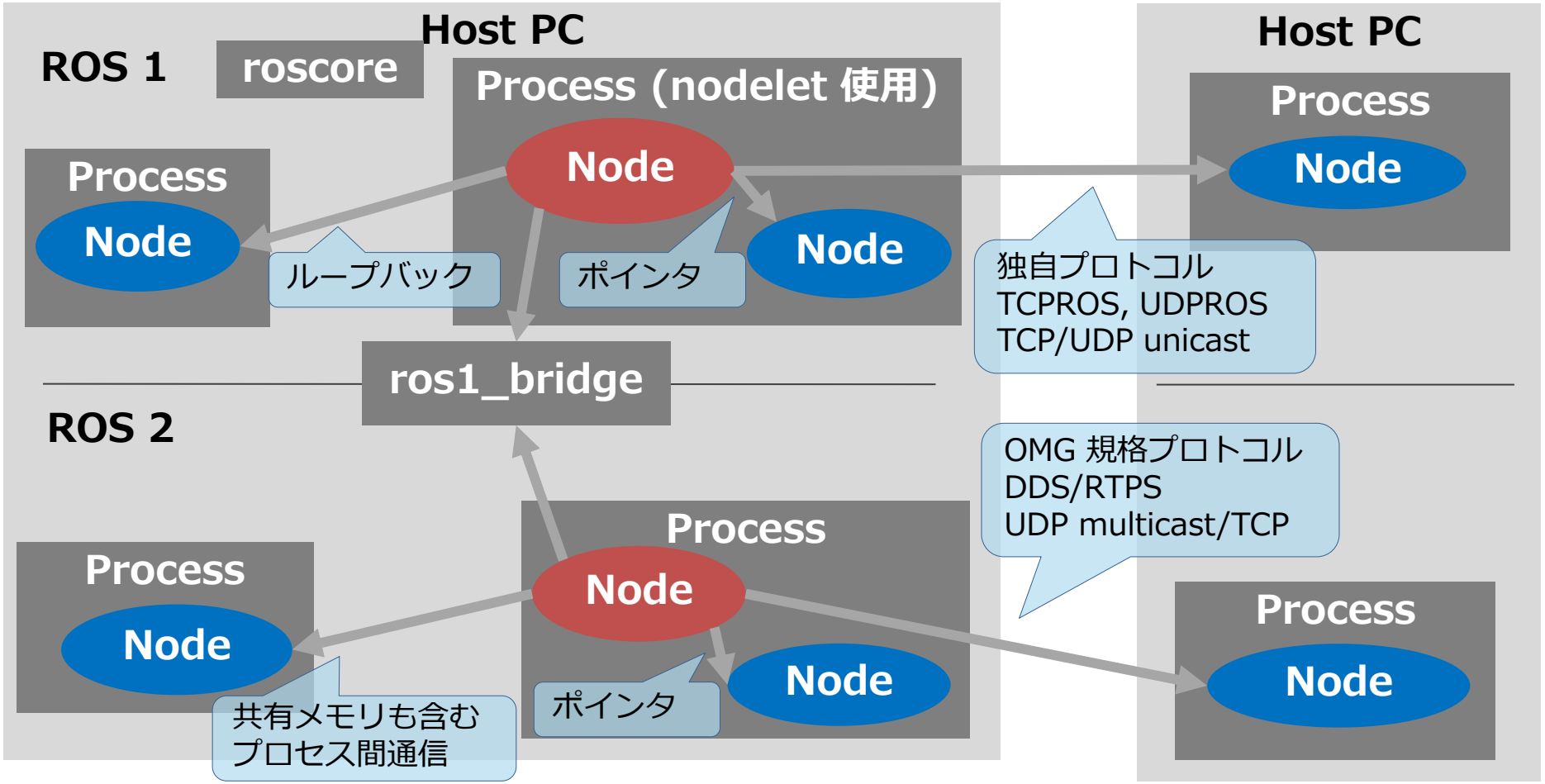
その他 参考資料

- 書籍
  - 高田 広章 ほか, リアルタイムOSと組み込み技術の基礎, CQ出版, 2003.
  - 西田 健 ほか, 実用ロボット開発のためのROSプログラミング, 森北出版, 2018.
  - 近藤 豊, ROS2ではじめよう 次世代ロボットプログラミング, 技術評論社, 2019.
- ROSCon
  - Andrei Kholodnyi, ROS2 on VxWorks - Challenges in porting a modern, software framework to RTOS, ROSCon 2019, 2019.
  - リアルタイム WS, Doing real-time with ROS 2: Capabilities and challenges, ROSCon 2019.
- ROSConJP
  - 足立 一希, REAL-TIME CONTROL IN REDHAWK AND ROS 2.0, ROSCon JP 2019, 2019.
  - 佃明彦, 組込みシステムにおける ROS 2の動向と RTOS の対応, ROSCon JP 2019, 2019.
- SWEST
  - Geoffrey Biggs, 次世代ロボットフレームワーク ROS2 の紹介, SWEST20, 2018.
  - Geoffrey Biggs, 安積 卓也, 自動運転プラットフォームの実用化： ROS2 で高信頼ソフトウェアの実装, SWEST21, 2019.
- その他
  - A. Pemmaiah et al., Performance Testing in ROS2, Apex, 2020.
  - 森田 賢, ROSによる今後のロボティクスのあり方,ロボティクスの未来 YASKAWA Mirai Tech #1, 2019.
  - 高瀬 英希, つながるロボット ～分散協調ロボットの開発を加速化するROSの紹介～,けいはんなロボット技術フォーラム2019 Autumn（第2回ミライロボット研究会）, 2019.

予備スライド

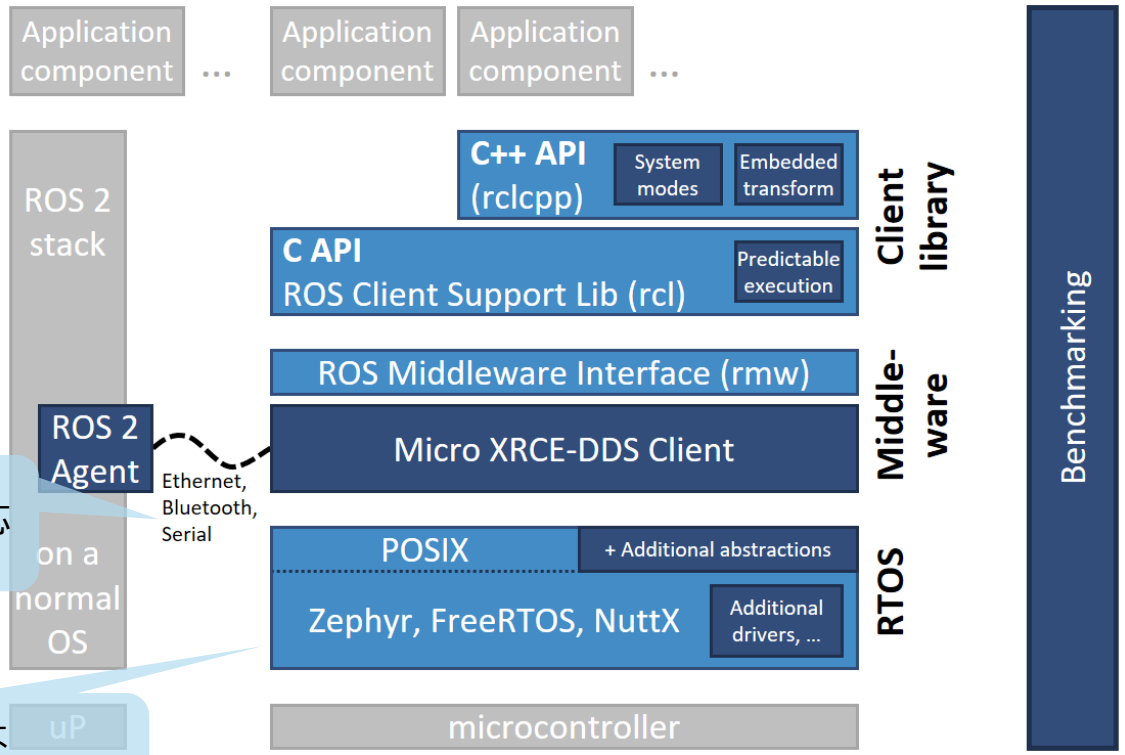
ROS1 vs ROS2｜通信の比較

- ROS 2 は通信プロトコルを OMG 規格に変更
- ROS 1 と ROS 2 間の通信にはブリッジが必要



- XRCE-DDS
  - マイコンで ROS の pub/sub 通信を行う
  - PC 側にエージェント、マイコン側に軽量のクライアントを実装
  - micro-ROS が対応

組み込み向けにフォークされた ROS 2



複数のプロトコルに対応

サポート OS は
マイコン向けのRTOS

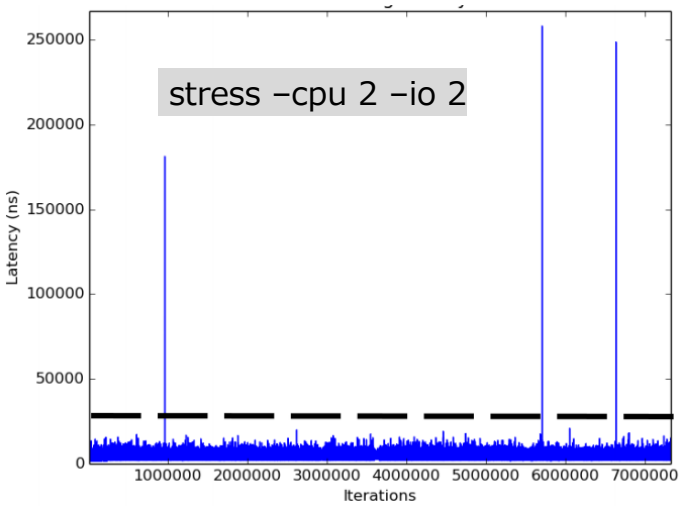
[micro-ROS | Architecture]

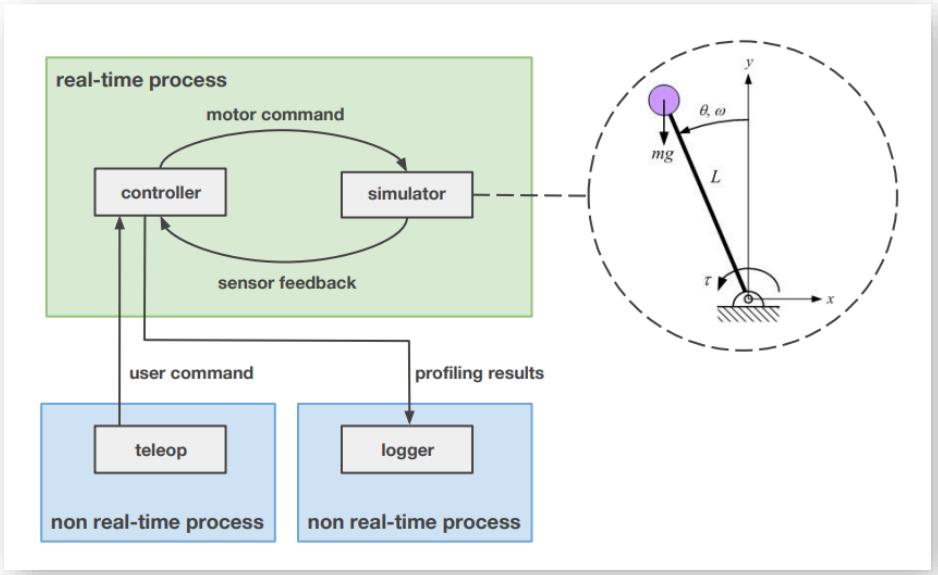
リアルタイムデモ｜pendulum_control

- RT_PREEMPT パッチ適用済みカーネル
- 優先度スケジューリングの利用 (SCHED_RR, 優先度 98)
- カスタム executor の使用
- TLSF アロケータの使用
- 実行時の allocation 排除
- 実行時の major pagefaults 排除

Real-time control in ROS and ROS 2.0

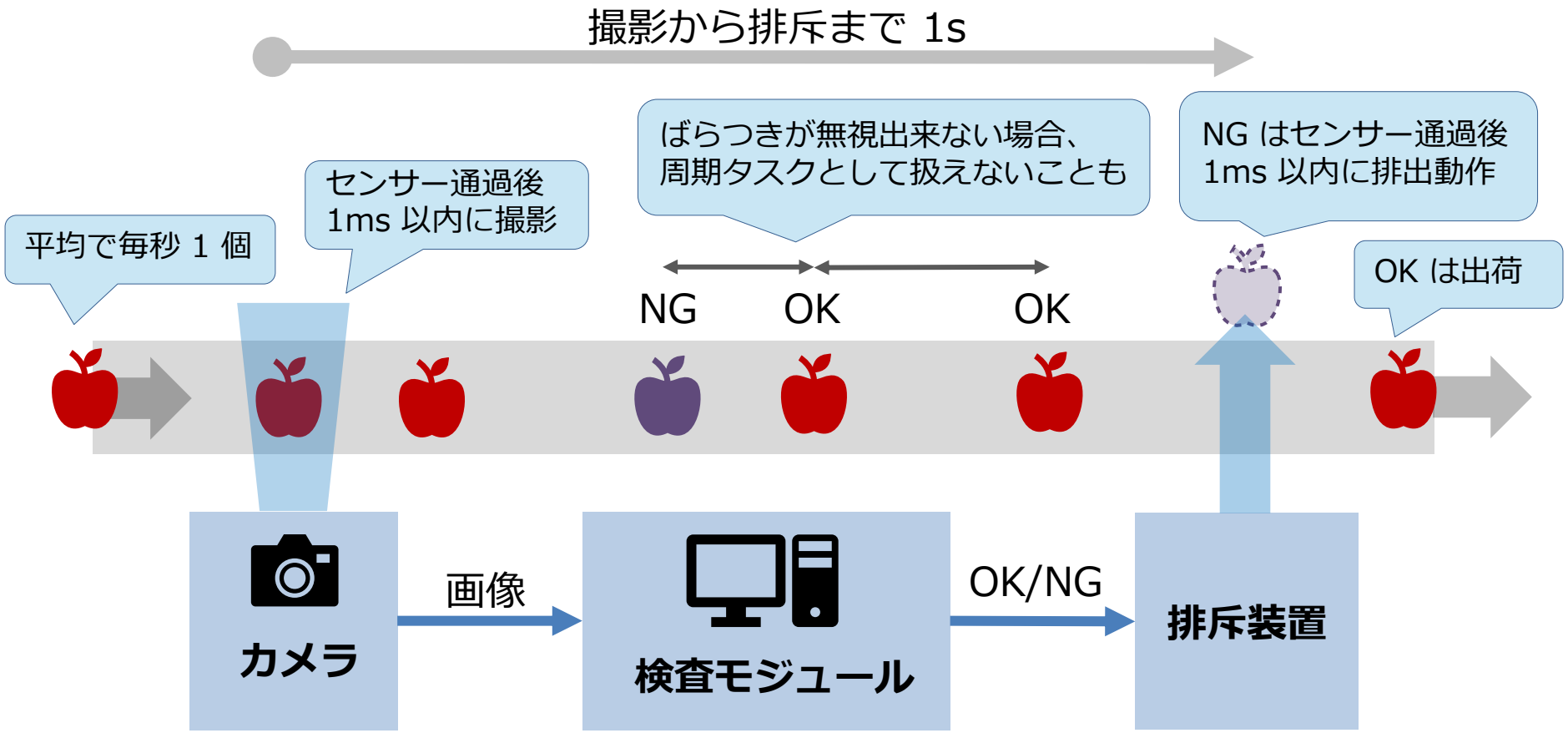
Goal
- 1 kHz update loop (1 ms period)
- Less than 3% jitter (30 µs)





リアルタイム性が必要な例｜外観検査

- 外観検査
  - システム全体： 撮影から 1s 以内
  - カメラ・排斥装置：センサー検知から 1ms 以内



リアルタイム性が必要な例（りんごの外観検査）