ROS 2スレッド属性設定拡張のご紹介

2024年02月09日

SWD Engineering Platform Development ROS2/Autoware
森田 正二

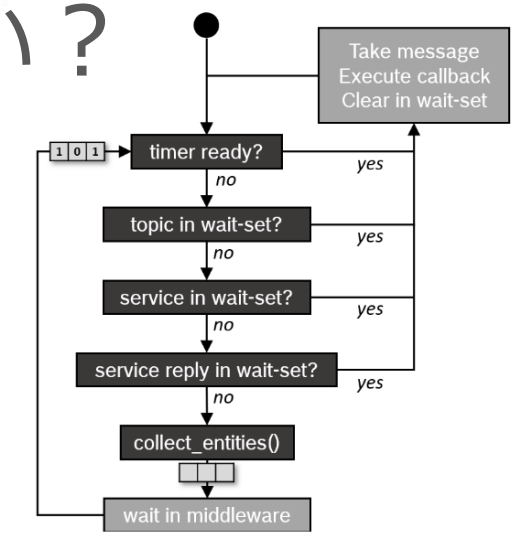
Real-time Working Groupの紹介

- 2020年から活動をしているROS公認のWG(隔週月曜日、深夜AM0:00 or 1:00)
- 主な参加メンバ
  - Andreiさん(WindRiver), Carlosさん(Apex.AI), Shuhao(NVIDIA), Janさん(BOSCH)等々
- 活動内容
  - The Real-Time Working Group's mission is to advocate for and work on memory management, real-time pub/sub, real-time DDS, and tools that allow tracing, profiling and optimizing.
  - 最近の主だったトピック
    - **ROSCon 2023でのワークショップの開催準備**
      - ROS Japan UG #53で高瀬先生が紹介
      - **昨年のメインは啓蒙活動**
    - Raspberry Pi 4ベースのリファレンス環境の提供
      - 今年は5に移行したいらしい
    - REP-2017
      - 今日の本題

https://ros-realtime.github.io/roscon-2023-realtime-workshop

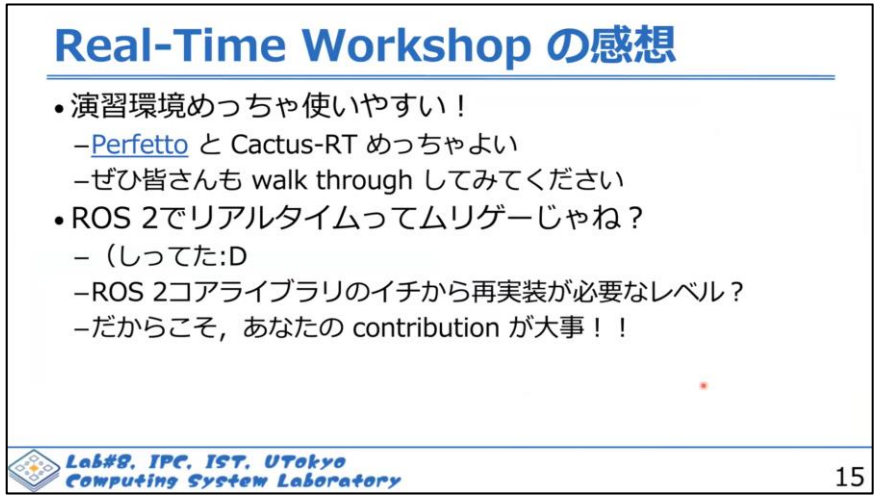
ROS 2でReal-timeはなぜ盛り上がらない？

- Real-timeの需要が無い?
  - 速い?Linuxマシンで何とかなっている?
  - 厳格なReal-timeが必要なものはROSしない?
  - Executorの振る舞いがおかしい?(Casini et al. at ECRTS 2019)
- ガチ勢によってExecutorの問題は提案・改善しつつある。
  - Static Single Threaded Executor
  - Callback groups
  - rclcpp WaitSet, Events Executor
- もっと他に問題はないか？
  - Real-timeのROS 2 アプリを書くのは面倒
    →すぐにOS依存部分が顔を出す。
  - ROS 2はReal-timeをサポートするインフラが弱い。
    → 例えば、コンフィグ方法の用意は無し。



動きが読みづらいExecutor

https://docs.ros.org/en/humble/Concepts/Intermediate/About-Executors.html



Workshopに参加した高瀬先生の感想

https://speakerdeck.com/takasehideki/roscon-2023can-jia-bao-gao-real-time-workshop-and-zenohs-current-status-and-forecast

Real-timeなROS 2 アプリを書くには

- Real-timeの基本はスケジューラの設定だが
  - (1) Executorのスレッド制御と機種依存コードが必要
  - (2) Executorのスレッド属性の受け渡し方法が必要

```
rclcpp::init(argc, argv);
 ...
rclcpp::executors::SingleThreadedExecutor
     front_exe,
 ...
 for (const auto & node : front_nodes) {
    front_exe.add_node(nodes.at(node));
 }
 ...
 std::thread front_thread {[&]() {
      set_rt_properties(hotpath_prio,
         HOTPATH_AFFINITY);
      front_exe.spin();
    }};
 ...
```

```
void set_rt_properties(int prio, const
std::unordered_set<size_t> & affinity)
{
  struct sched_param sched_param = { 0 };
  sched_param.sched_priority = prio;
  sched_setscheduler(0, SCHED_RR, &sched_param);

  cpu_set_t cpuset;
  CPU_ZERO(&cpuset);
  for (const auto cpu : affinity) {
    CPU_SET(cpu, &cpuset);
  }
  sched_setaffinity(0, sizeof(cpuset), &cpuset);
}
```

https://github.com/ros-realtime/reference-system/blob/main/autoware_reference_system/src/ros2/executor/autoware_default_prioritized.cpp

Real-timeなROS 2 アプリを書くには

- Real-timeの基本はスケジューラの設定だが
  - (1) Executorのスレッド制御と機種依存コードが必要
  - (2) Executorのスレッド属性の受け渡し方法が必要

```
rclcpp::init(argc, argv);
 ...
rclcpp::executors::SingleThreadedExecutor
     front_exe,
 ...
 for (const auto & node : front_nodes) {
    front_exe.add_node(nodes.at(node));
 }
 ...
 std::thread front_thread {[&]() {
      set_rt_properties(hotpath_prio,
         HOTPATH_AFFINITY);
      front_exe.spin();
    }};
 ...
```

```
void set_rt_properties(int prio, const
std::unordered_set<size_t> & affinity)
{
  struct sched_param sched_param = { 0 };
  sched_param.sched_priority = prio;
  sched_setscheduler(0, SCHED_RR, &sched_param);

  cpu_set_t cpuset;
  CPU_ZERO(&cpuset);
  for (const auto cpu : affinity) {
    CPU_SET(cpu, &cpuset);
  }
  sched_setaffinity(0, sizeof(cpuset), &cpuset);
}
```

https://github.com/ros-realtime/reference-system/blob/main/autoware_reference_system/src/ros2/executor/autoware_default_prioritized.cpp

Executorのスレッド制御と機種依存コードが必要

- ROS 2ではスレッドモデルの実装は言語バインディング層(Client Library)の責務

  Language-specific functionality

  Client library concepts that require language-specific features/properties are not implemented in the RCL but instead are implemented in each client library. For example, threading models used by “spin” functions will have implementations that are specific to the language of the client library.
- rclcppがベースとするstd::thread(C++標準)はスレッド属性の操作はできない
  - スレッド属性の変更やExecutorのスレッドの割当はOS固有の処理で実現する必要がある。
  - もっと言うと優先度継承のmutexもC++標準の範囲外

Real-timeなROS 2 アプリを書くには

- Real-timeの基本はスケジューラの設定だが
  - (1) Executorのスレッド制御と機種依存コードが必要
  - (2) Executorのスレッド属性の受け渡し方法が必要

```
rclcpp::init(argc, argv);
 ...
rclcpp::executors::SingleThreadedExecutor
     front_exe,
 ...
 for (const auto & node : front_nodes) {
    front_exe.add_node(nodes.at(node));
 }
 ...
 std::thread front_thread ([&]() {
      set_rt_properties(hotpath_prio,
         HOTPATH_AFFINITY);
      front_exe.spin();
    }};
 ...
```

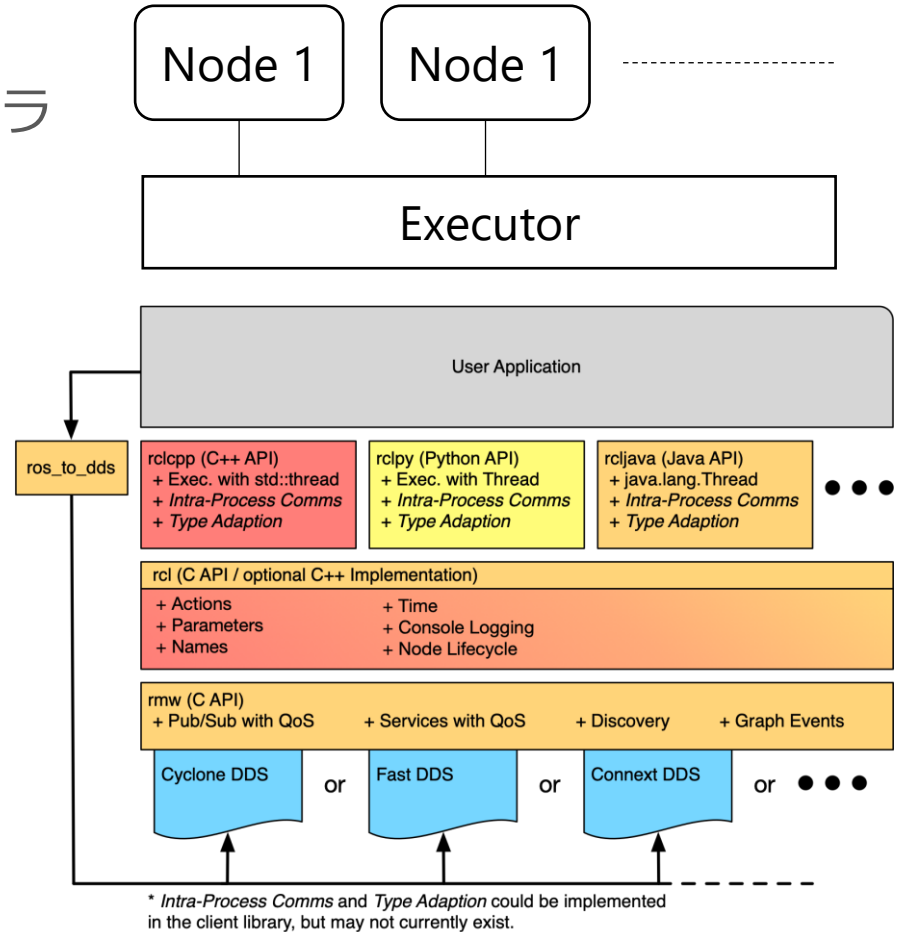
```
void set_rt_properties(int prio, const
std::unordered_set<size_t> & affinity)
{
  struct sched_param sched_param = { 0 };
  sched_param.sched_priority = prio;
  sched_setscheduler(0, SCHED_RR, &sched_param);

  cpu_set_t cpuset;
  CPU_ZERO(&cpuset);
  for (const auto cpu : affinity) {
    CPU_SET(cpu, &cpuset);
  }
  sched_setaffinity(0, sizeof(cpuset), &cpuset);
}
```

https://github.com/ros-realtime/reference-system/blob/main/autoware_reference_system/src/ros2/executor/autoware_default_prioritized.cpp

Executorのスレッド属性の受け渡し方法が必要

- ROS 2のParameterが使えそうだが・・・
  - ParameterはNodeに対する設定をするためのインフラ
  - ExecutorにはNodeを複数紐づけられるので不適切
  - アプリごとにパラメタのセットを定義するのは面倒
- Client library層で受け持つべきか？
  - スレッド・モデルはClient library毎だが、
    スレッド属性はClient libraryには依存しないはず。
    → Client library毎にパラメタセットを個別に
    用意すべきではない。
  - 将来的にはExecutor以外の実行体のスレッド属性も
    設定したくなるかも。



https://docs.ros.org/en/humble/Concepts/Advanced/About-Internal-Interfaces.html

Real-timeなROS 2開発を盛り上げるために

- ガチ勢だけでなくライト勢へも裾野を広げる。
  プラットフォームは使われてなんぼ、フィードバックがないと良くならない。
- ROS 2のプラットフォームとして、Real-timeなExecutorを利用しやすいインフラを加えるべき。
  - 環境依存の処理やC++標準に不足しているものはプラットフォームに任せる。
  - 抽象化したスレッド属性設定のインフラを用意する。

Provide features

Feedback

eSOLのROS 2スレッド設定拡張の提案

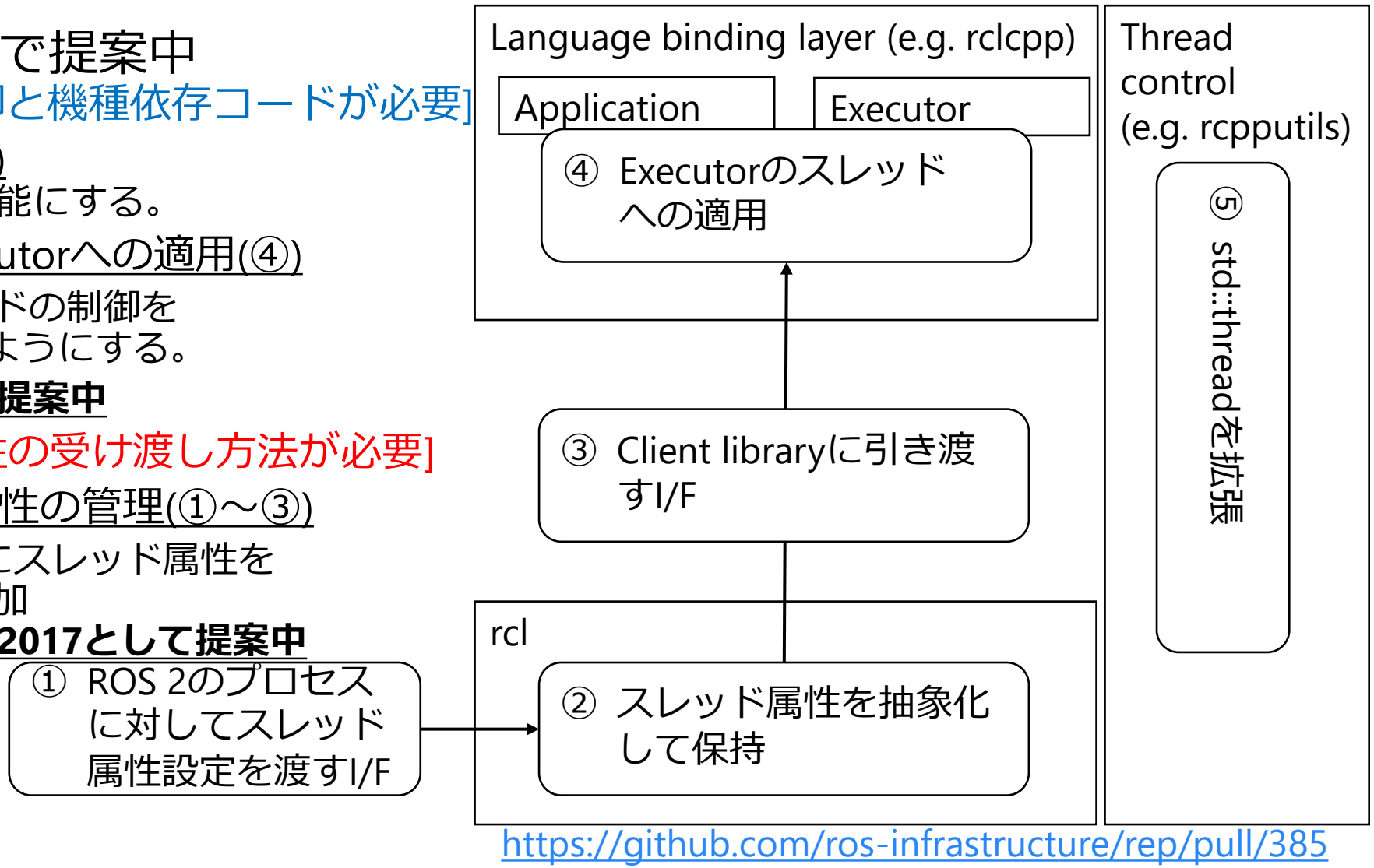
- 以下をReal-time WGで提案中

[Executorのスレッド制御と機種依存コードが必要]

1. std::threadの拡張(⑤)
   スレッド属性を設定可能にする。
2. スレッド属性のExecutorへの適用(④)
   アプリ開発者はスレッドの制御を
   Executorに移譲できるようにする。
   → **Pull-Requestとして提案中**

[Executorのスレッド属性の受け渡し方法が必要]

3. rcl層でのスレッド属性の管理(①～③)
   Client libraryやアプリにスレッド属性を
   引き渡すインフラを追加
   → **Pull-Request&REP-2017として提案中**



https://github.com/ros-infrastructure/rep/pull/385

std::threadの拡張(rcpputils::thread)

- rcpputilsへのPull-Request

  https://github.com/ros2/rcpputils/pull/179

- 既存のstd::threadを属性指定をしてスレッドを生成できるように拡張
  以下を追加しただけ、それ以外はstd::threadと同等

```
template<typename F, typename ... Args>
  Thread(Attribute const & attr, F && f, Args && ... args)
```

  - 優先度、スケジューリングポリシ、コアアフィニティが設定できる。
  - 現在はLinuxのみ、それ以外はstd::threadにフォールバックしている。

- 優先度継承付きmutexも将来的に追加することを検討中

スレッド属性のExecutorへの適用

- rclcppへのPull-Request

  https://github.com/ros2/rclcpp/pull/2205

- Executorへのrcpputils::threadの利用方法の提案

  既存のSingle/Multithreaded Executorを拡張して、
  以下のようにスレッド属性をコンストラクタ引数として設定ができる。

```
---
auto planner_attr= find_thread_attr("RCLCPP_EXECUTOR_PLANNER");
rclcpp::executors::SingleThreadedExecutor
   planner_exe(rclcpp::ExecutorOptions(), *planner_attr);
---
```

- 予約済みのスレッド属性のタグ(後述)を使って既存のコードであっても、
  コードの修正なしスレッドの属性の変更も可能
  - RCLCPP_EXECUTOR_SINGLE_THREADED
  - RCLCPP_EXECUTOR_MULTI_THREADED

rcl層でのスレッド属性の管理(REP-2017で提案中)

- rclとrcutilsへのPull-Request
  https://github.com/ros2/rcl/pull/1075
  https://github.com/ros2/rcutils/pull/424
- コマンドライン引数or環境変数で属性を設定
  - コマンドライン
    `--thread-attrs-value=[YAMLでの属性設定の文字列]`
    `--thread-attrs-file=[YAMLファイルのPATH]`
  - 環境変数
    `ROS_THREAD_ATTRS_VALUE=[YAMLでの属性設定の文字列]`
    `ROS_THREAD_ATTRS_FILE=[YAMLファイルのPATH]`

```
- priority: 40
  tag: RCLCPP_EXECUTOR_SINGLE_THREADED
  core_affinity: [3]
  scheduling_policy: FIFO
- priority: 10
  tag: RCLCPP_EXECUTOR_MULTI_THREADED
  core_affinity: [4,5]
  scheduling_policy: OTHER
```

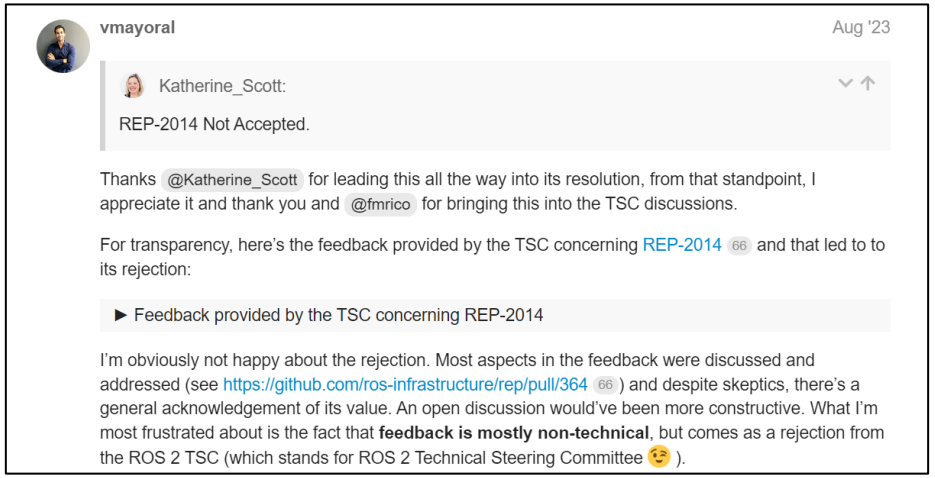
- Client libraryやアプリは、以下のI/Fを経由して属性を取得できる。

```
rcutils_thread_attrs_t
   * rcl_context_get_thread_attrs(const rcl_context_t * context);
```

- 設定できるのは優先度、タグ、スケジューリングポリシ、コアアフィニティ

REP(ROS Enhancement Proposal)とは

- ROSでのIETFにおけるRFCに相当する文章で、コミュニティ内での合意をとるために用いられる。
  https://github.com/ros-infrastructure/rep/
- 文章は以下の3種類
  - Standards Track: ROSの新機能や実装
  - Informational: 新機能を含まないガイドラインや推奨事項
  - Process: ROS開発プロセス自体を改善するための提案
- ROS 2ではStandard Trackでの採択は2件のみ
  - REP-2007: Type Adaptation Feature
  - REP-2009: Type Negotiation Feature
- 採択プロセスが不透明であるとの批判があり昨年Discourseが少し炎上
  ROS 2 TSC Meeting Minutes 2023-08-17



vmayoral Aug '23

> Katherine_Scott:
> REP-2014 Not Accepted.

Thanks @Katherine_Scott for leading this all the way into its resolution, from that standpoint, I appreciate it and thank you and @fmrico for bringing this into the TSC discussions.

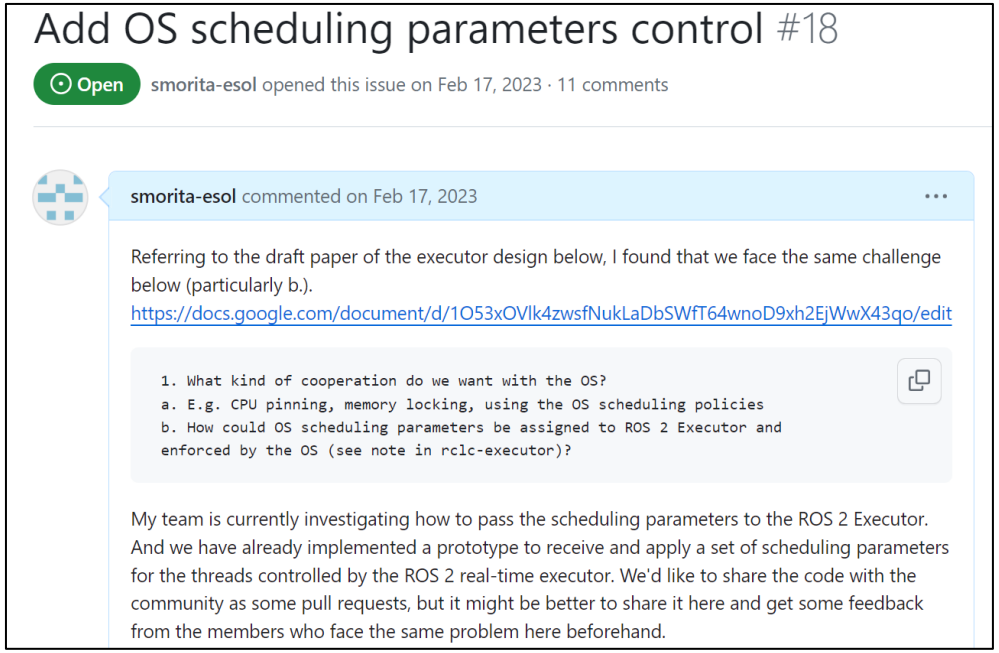
For transparency, here's the feedback provided by the TSC concerning REP-2014 66 and that led to to its rejection:

▶ Feedback provided by the TSC concerning REP-2014

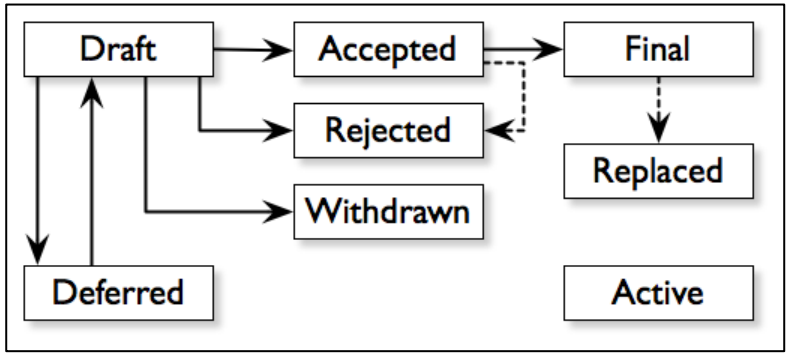
I'm obviously not happy about the rejection. Most aspects in the feedback were discussed and addressed (see https://github.com/ros-infrastructure/rep/pull/364 66) and despite skeptics, there's a general acknowledgement of its value. An open discussion would've been more constructive. What I'm most frustrated about is the fact that **feedback is mostly non-technical**, but comes as a rejection from the ROS 2 TSC (which stands for ROS 2 Technical Steering Committee 😉).

REPを提案するためのあれこれ

- まずはDiscourseや適切なWGで提案してみる。
  - ここでダメならREPとしても受けて付けられない
  - 英語は片言でもなんとかなる(と思う)。
- REPの作成ガイドは書式ぐらいしか参考にならない。
  https://github.com/ros-infrastructure/rep/blob/master/rep-0001.rst
  - REPの作成フローはまともに機能していない。
  - REPの番号は空いているのを勝手に使ってPull-Requestを挙げる。
- TSCに乗り込んで(呼ばれて？)主張する覚悟は必要
- 時間は年単位でかかりそう。



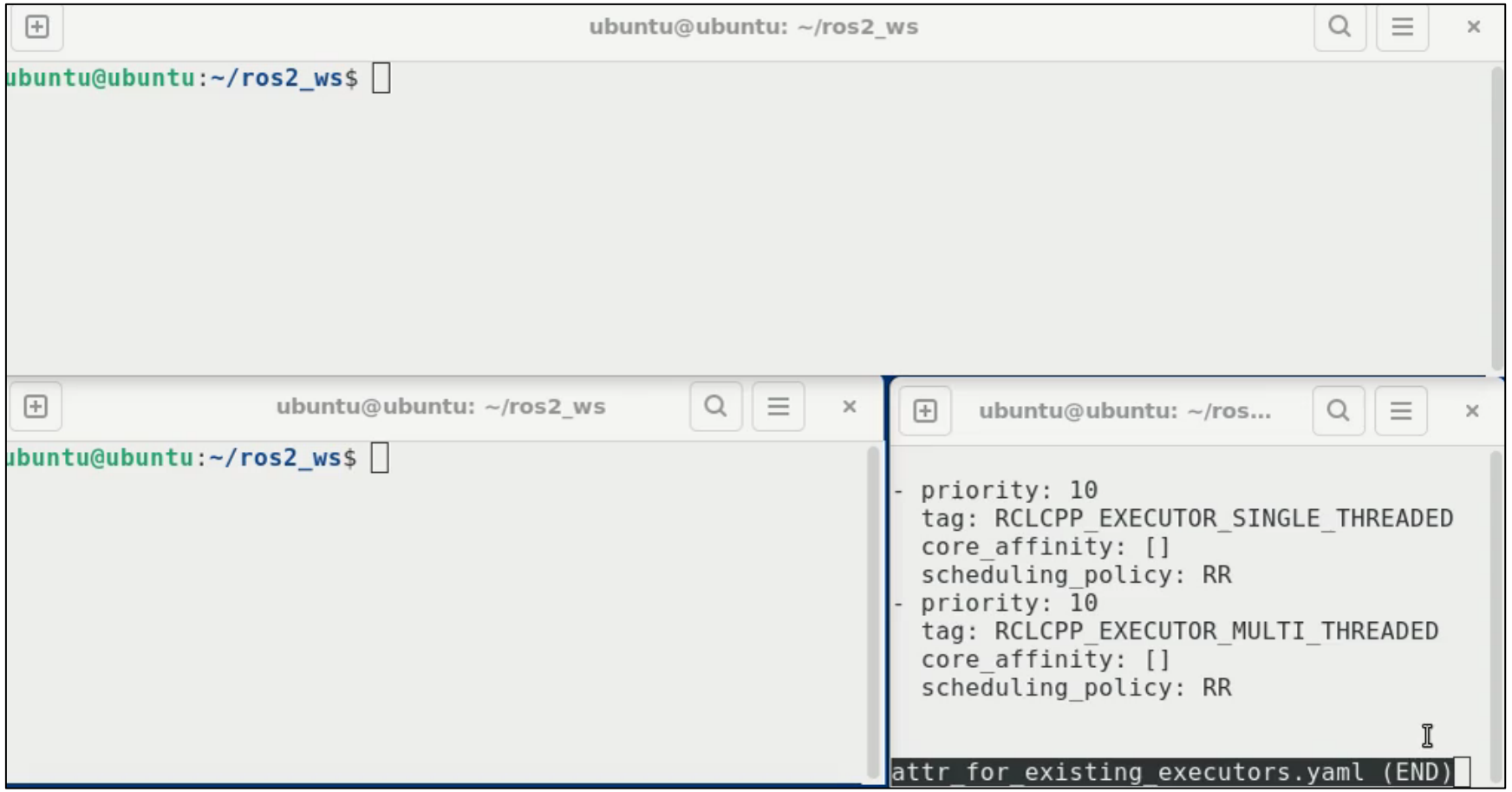
Real-time WGでIssueを挙げる
https://github.com/ros-realtime/ros-realtime.github.io/issues/18



おそらく機能していないREP作成フロー

ちょっとしたデモ(動画: Link)

■ 予約済みのスレッド属性のタグを使った既存アプリのスレッド属性の調整



- 開発済みのアプリをそのままに、スレッド優先度を変更できる。
- スレッド属性の調整はファイルやコマンドライン引数の編集のみでOK

まとめ

- ROS 2のReal-time対応はExecutorを中心に進みつつある。
  → ROS 2でReal-timeする土壌は整いつつある。
- プラットフォームとしてのROS 2のReal-timeのサポートはまだまだ貧弱
  → ROS 2に任せることができるものは任せたい。
- RTWGの活動として、スレッド属性設定の入り口であるrcl1の拡張を提案した。
  → RTWGでは簡単なデモアプリが用意できないかを検討中
- 今後は、言語バインディング側の拡張の検討を進めるべきだが、rcl1以上に多方面の意見・アイデアの集約が必要である。
  → 次のREP(2018?)の策定を一緒にやっていただける方おられませんか？

Challenge with Passion

REP-2017に対するFAQ

- POSIXに寄りすぎていてWindowsでは動かないのでは？
  - 現状、rcpputil::threadのWindows向けの実装はただのstd::threadとして動作するののので動かないわけではない。
  - YAMLのパラメタセットはPOSIXベースです。ただ、Windows向けにrcpputils::threadを用意する場合は、各パラメタのいくつかはWindowsの仕様に射影できると考えてる。
- ROS 2の枠組みではなく、汎用のシステム全体のスレッド属性制御フレームワークを作ればいいのでは？
  - 仮にシステム全体のスレッド属性を取り扱う便利なフレームワークがあったとしても、各スレッド属性とExecutorの紐づけをするためのREP-2017のような枠組みは必要です。
  - 総花的なスレッド属性制御フレームワークを作成するためには、技術分野を横断する知見の集約が必要であり難しい。
  - 一足飛びに汎用性を持たせるのではなく、ROS 2のExecutorに用途を絞って抽象化し、徐々に適用範囲を広げるアプローチを取りたい。