ROS-Industrial Conference 2020

Advanced Execution Management with ROS 2

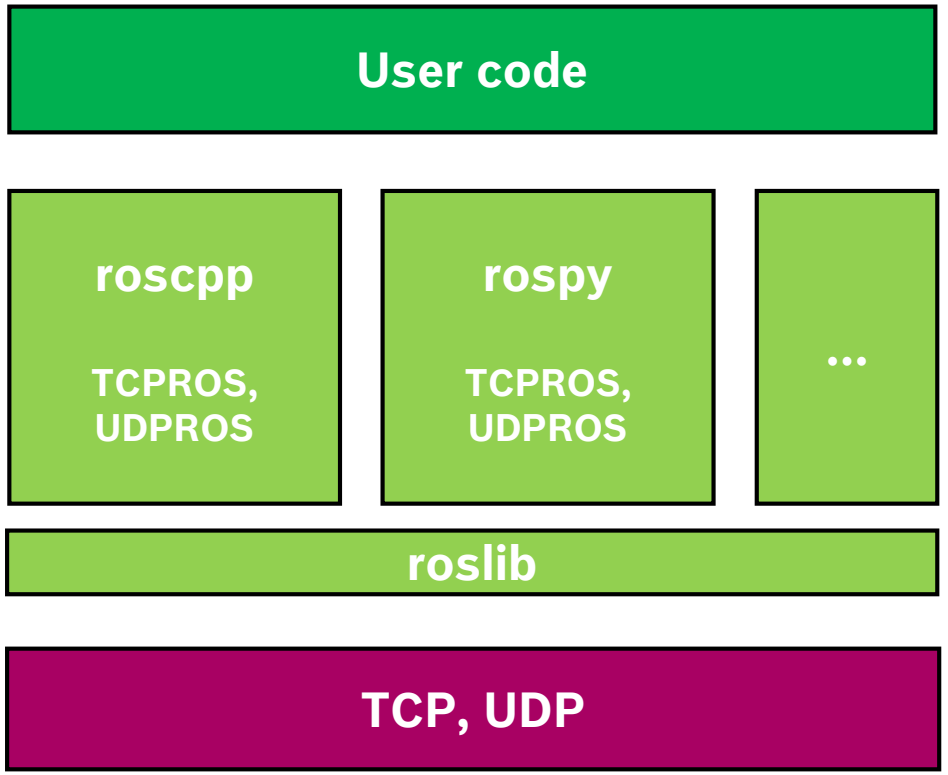
Dr. Ralph Lange
Bosch Corporate Research

BOSCH

```
int main(int argc, char* argv[])
{
  ros::init(argc, argv, "my_node");
  ros::NodeHandle nh;
  // Init some stuff
  ros::spin();
  return 0;
}
```

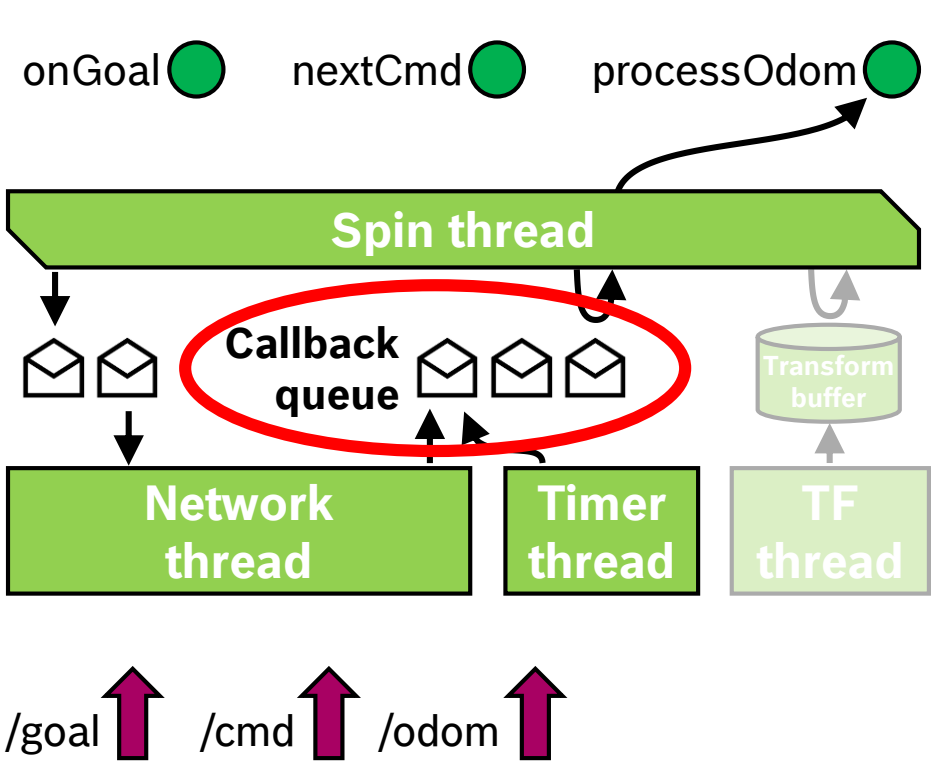
BOSCH

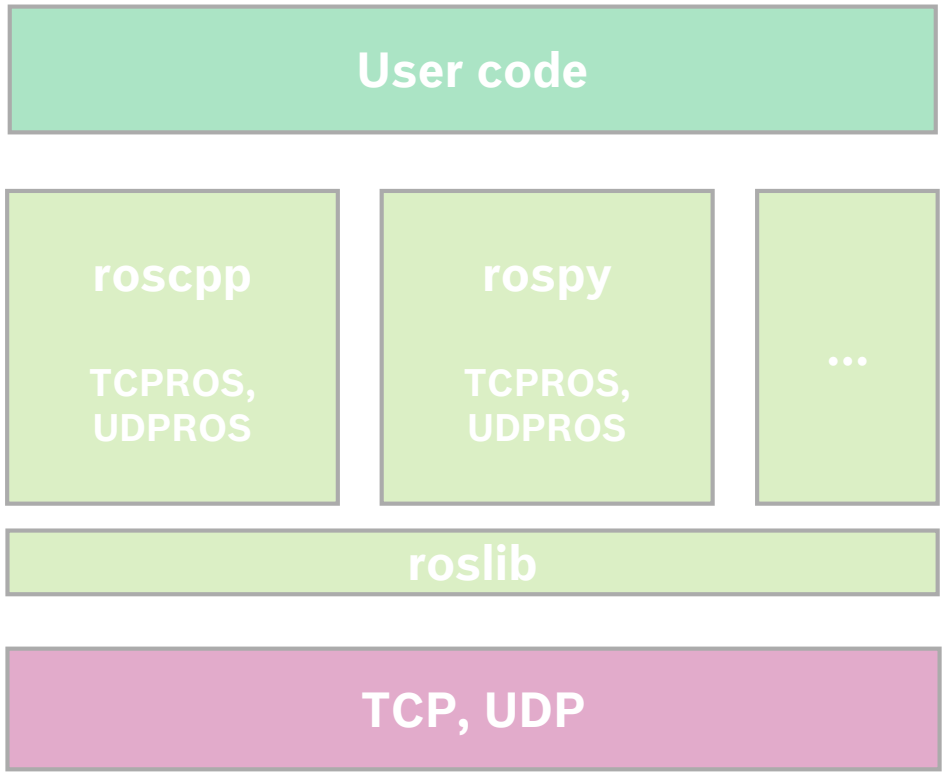
Execution Management in ROS 1



BOSCH

Execution Management in ROS 1



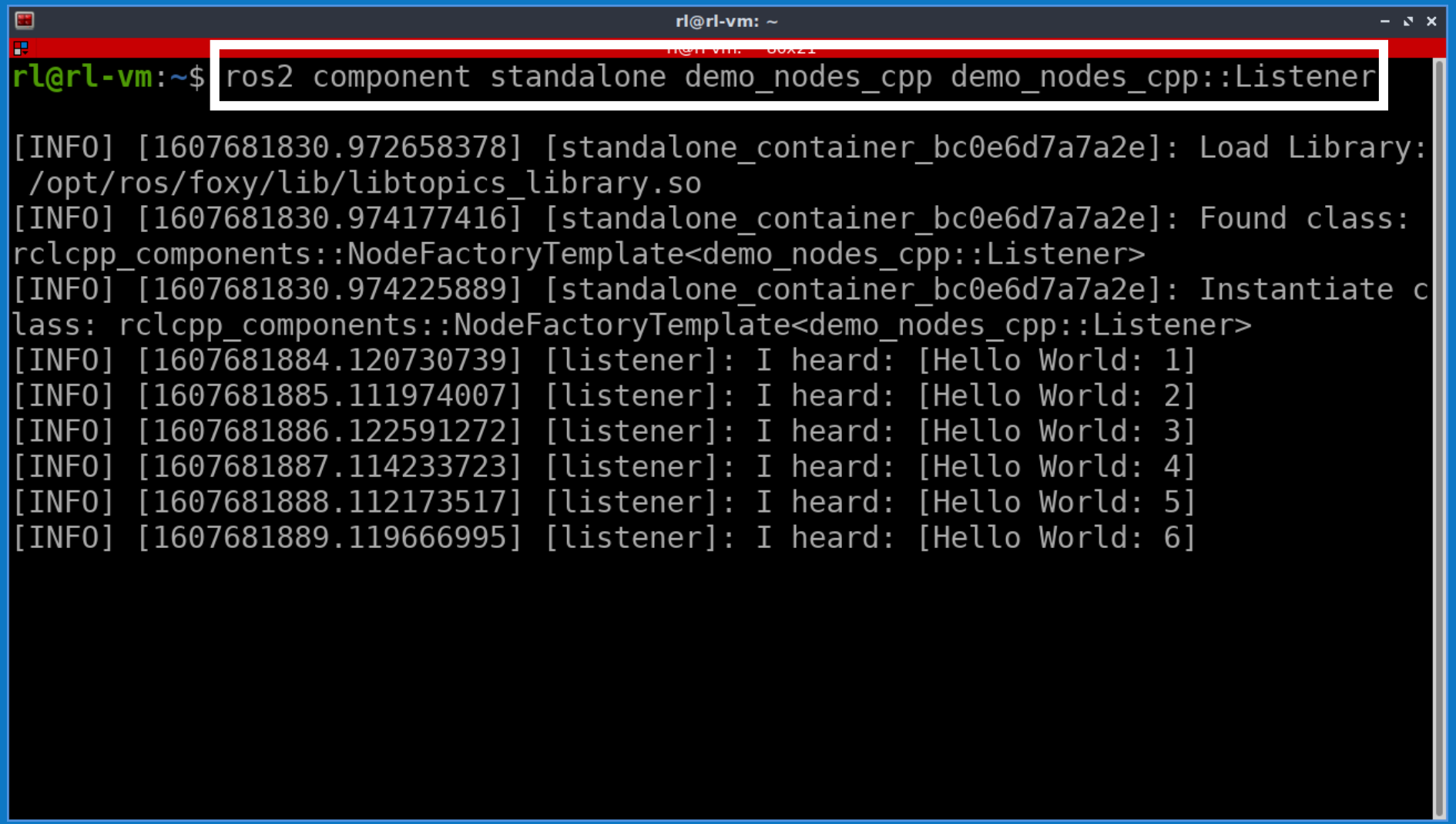


Ingo Lütkebohle: “Determinism in ROS”, https://vimeo.com/236186712

BOSCH

```
int main(int argc, char* argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::Node::SharedPtr node = ...
  rclcpp::spin(node);
  return 0;
}
```

BOSCH


rl@rl-vm: ~
rl@rl-vm:~$ ros2 component standalone demo_nodes_cpp demo_nodes_cpp::Listener
[INFO] [1607681830.972658378] [standalone_container_bc0e6d7a7a2e]: Load Library:
/opt/ros/foxy/lib/libtopics_library.so
[INFO] [1607681830.974177416] [standalone_container_bc0e6d7a7a2e]: Found class:
rclcpp_components::NodeFactoryTemplate<demo_nodes_cpp::Listener>
[INFO] [1607681830.974225889] [standalone_container_bc0e6d7a7a2e]: Instantiate c
lass: rclcpp_components::NodeFactoryTemplate<demo_nodes_cpp::Listener>
[INFO] [1607681884.120730739] [listener]: I heard: [Hello World: 1]
[INFO] [1607681885.111974007] [listener]: I heard: [Hello World: 2]
[INFO] [1607681886.122591272] [listener]: I heard: [Hello World: 3]
[INFO] [1607681887.114233723] [listener]: I heard: [Hello World: 4]
[INFO] [1607681888.112173517] [listener]: I heard: [Hello World: 5]
[INFO] [1607681889.119666995] [listener]: I heard: [Hello World: 6]

BOSCH

Executor

```
...
rclcpp::executors::SingleThreadedExecutor executor;
executor.add_node(node);
executor.spin();
...
```

BOSCH

Agenda

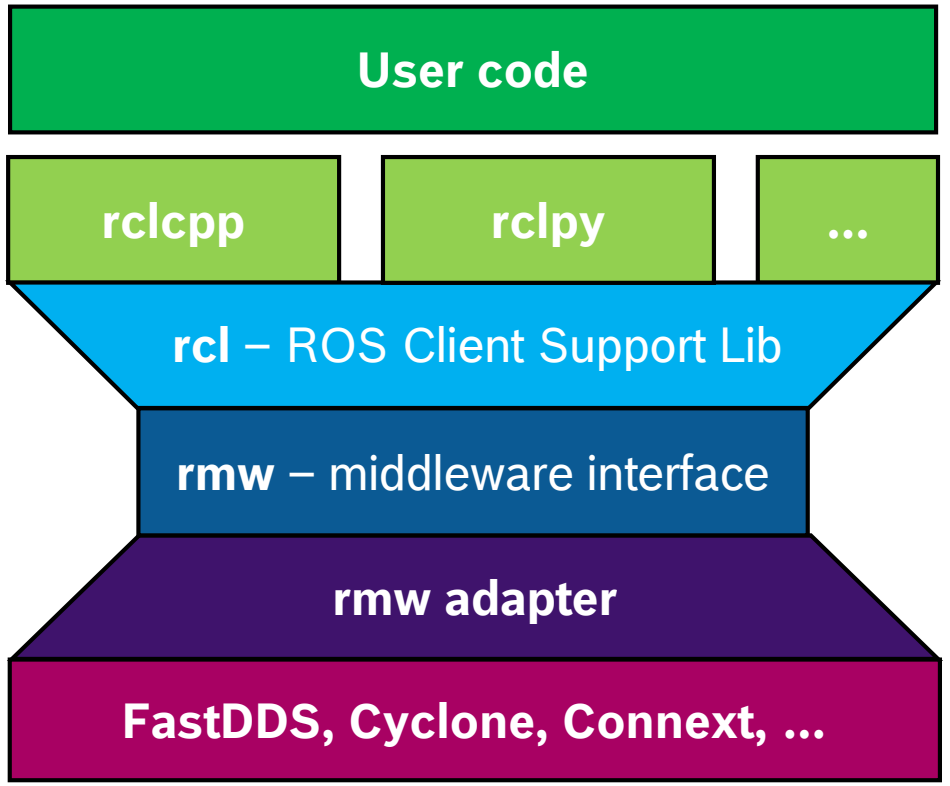
- Objectives behind Executor design
- Default scheduling semantics – and its issues

- Static Executor
- Callback-group-level Executor

- Determinism – and particularly FIFO ordering

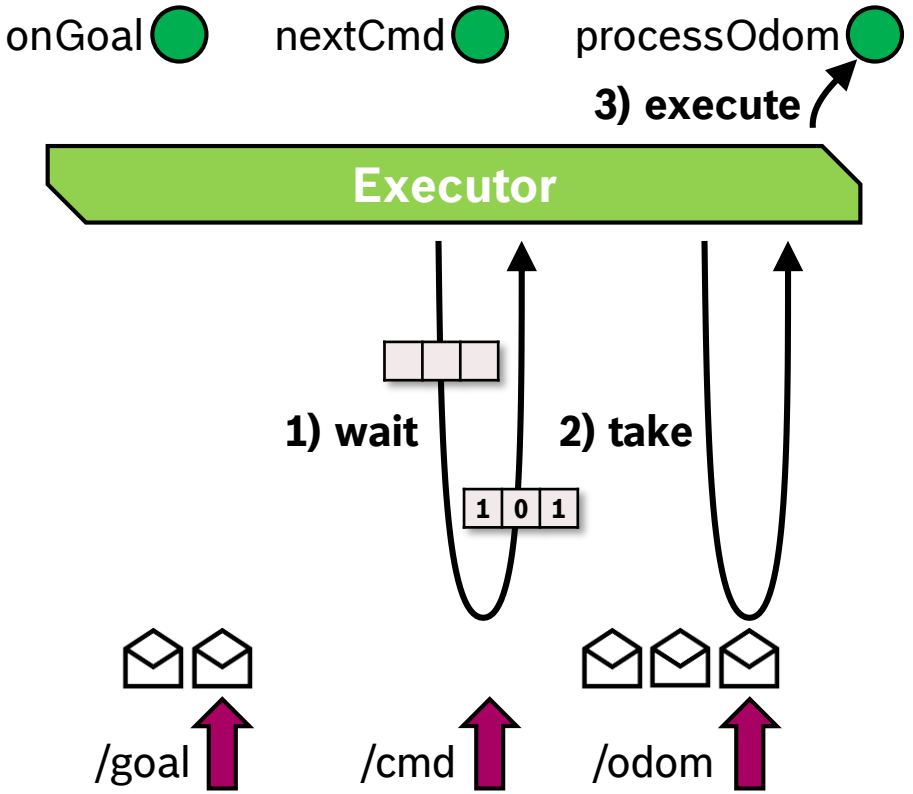
- rclc Executor (micro-ROS)

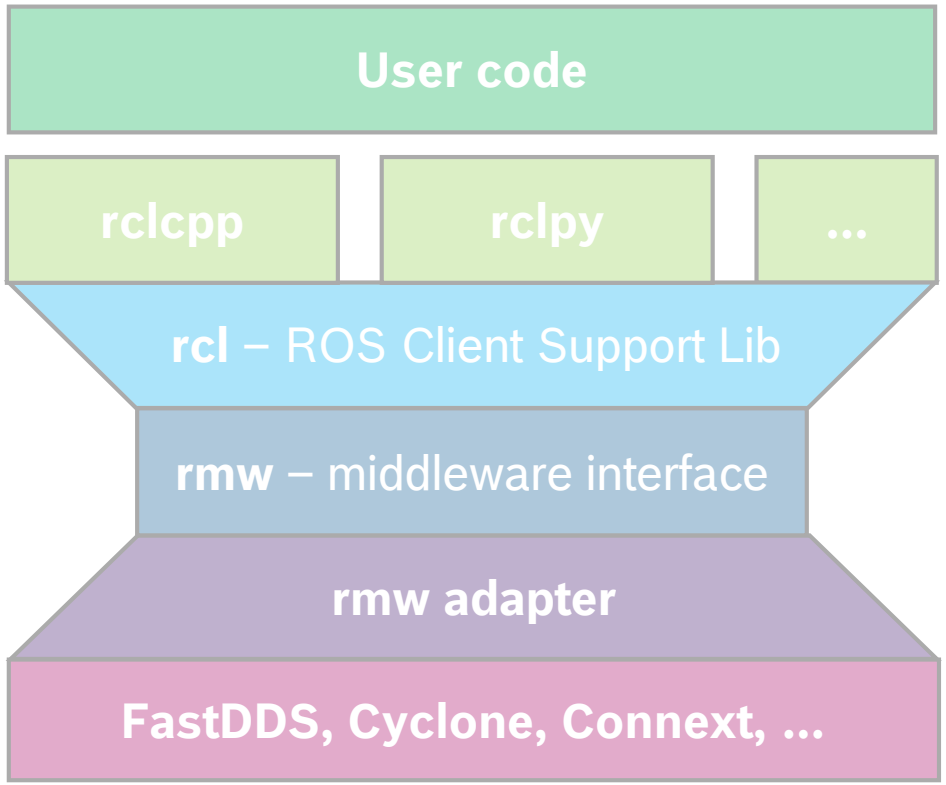
Executor Design



Ralph Lange (CR/AEE1) | 2020-12-16

Executor Design



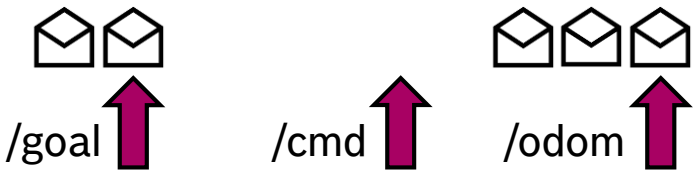


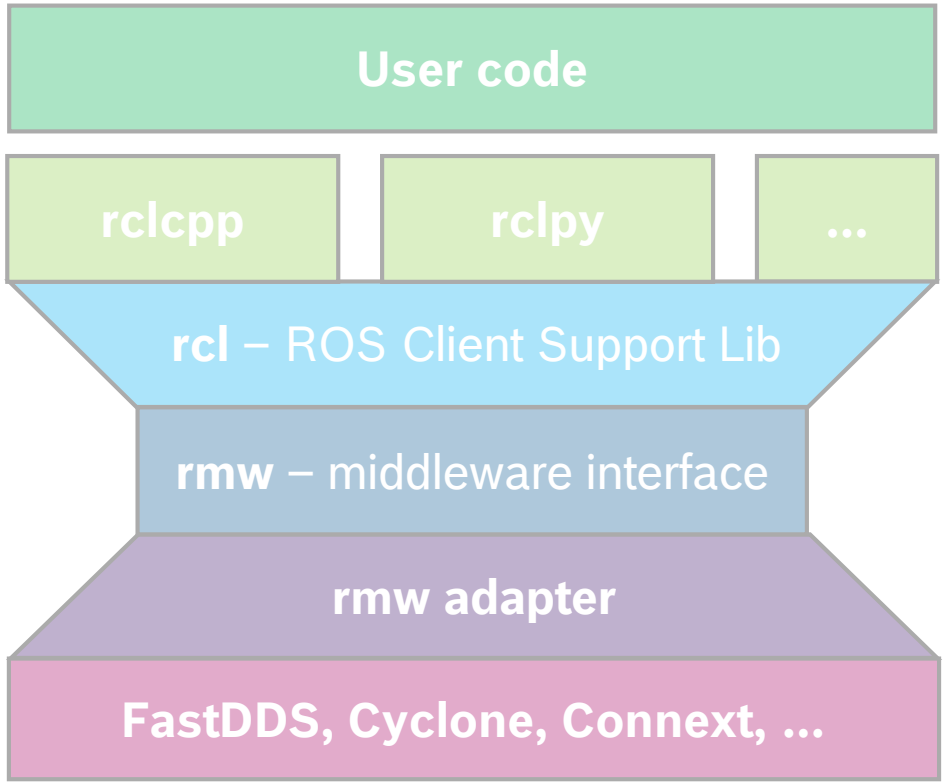
Executor Design

Design objectives

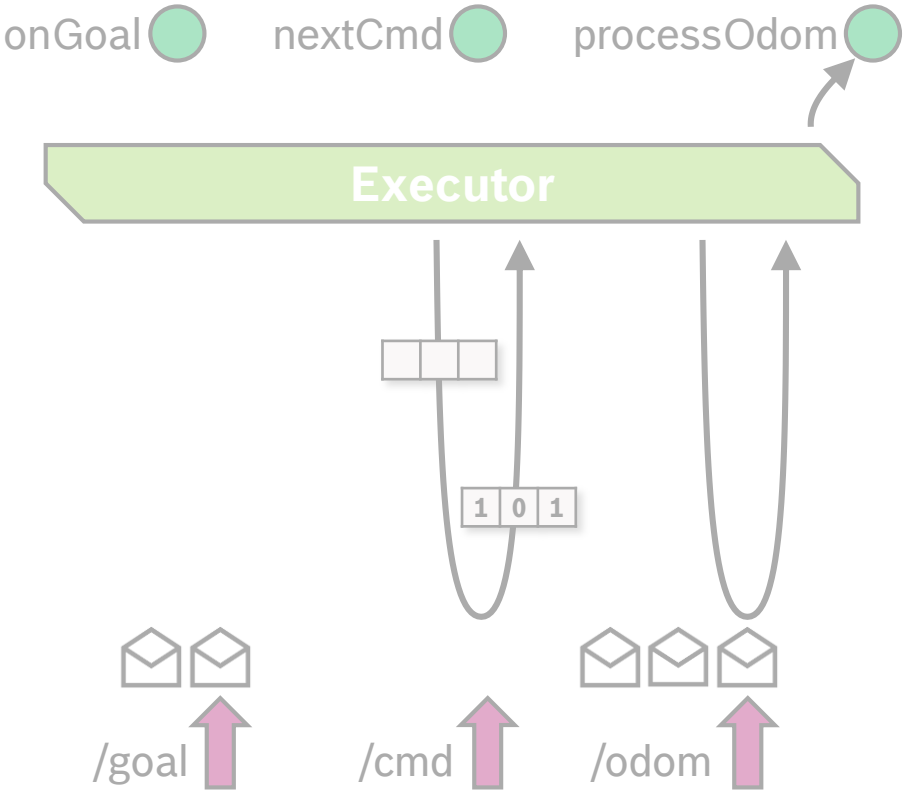
- Avoid additional queue in client library
- Utilize DDS QoS mechanisms
  - Lifespan, history, priorities, …

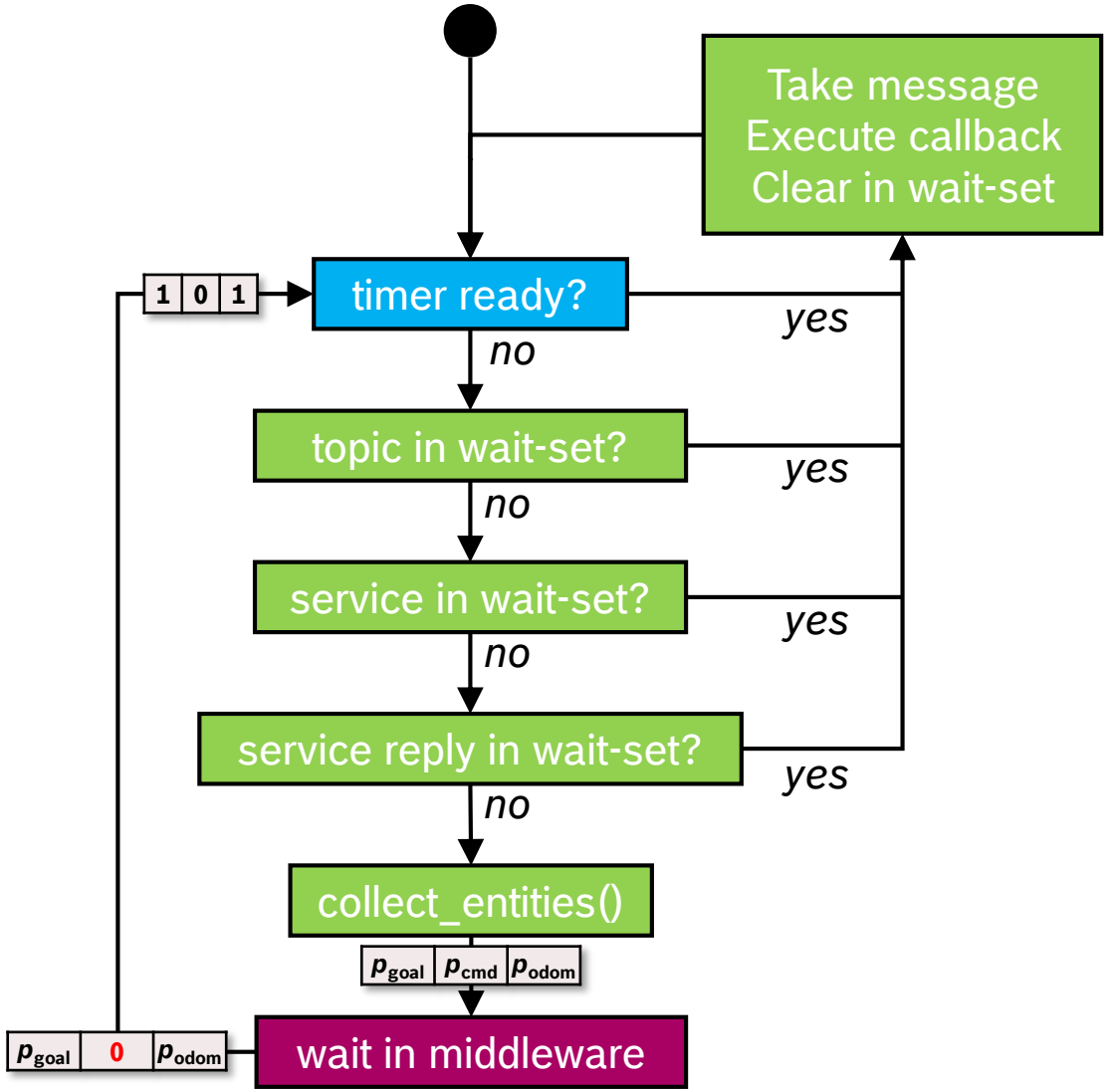
Decision on processing order is distributed to middleware and client lib





Scheduling Semantics



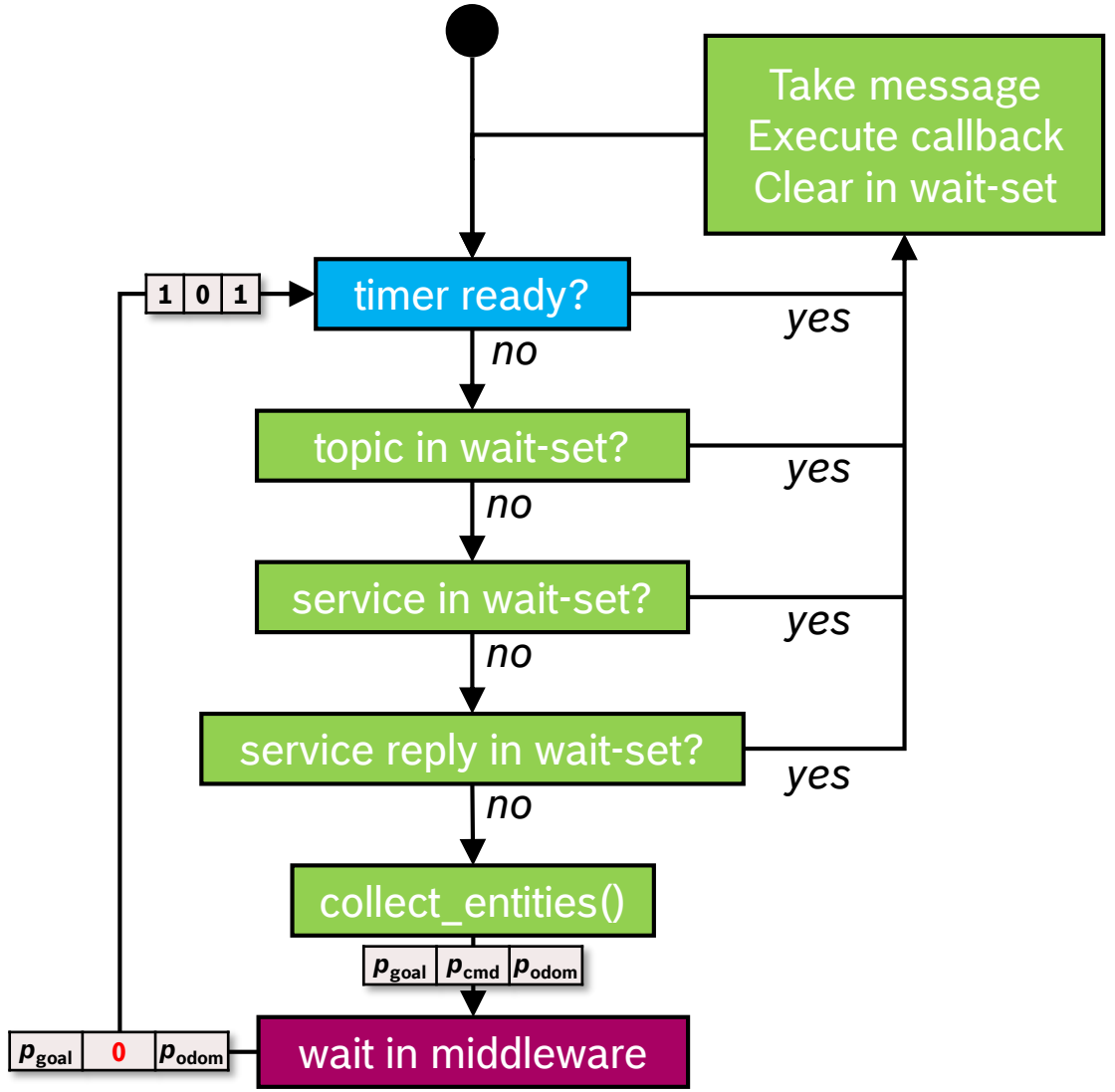


Ralph Lange (CR/AEE1) | 2020-12-16

BOSCH

Scheduling Semantics

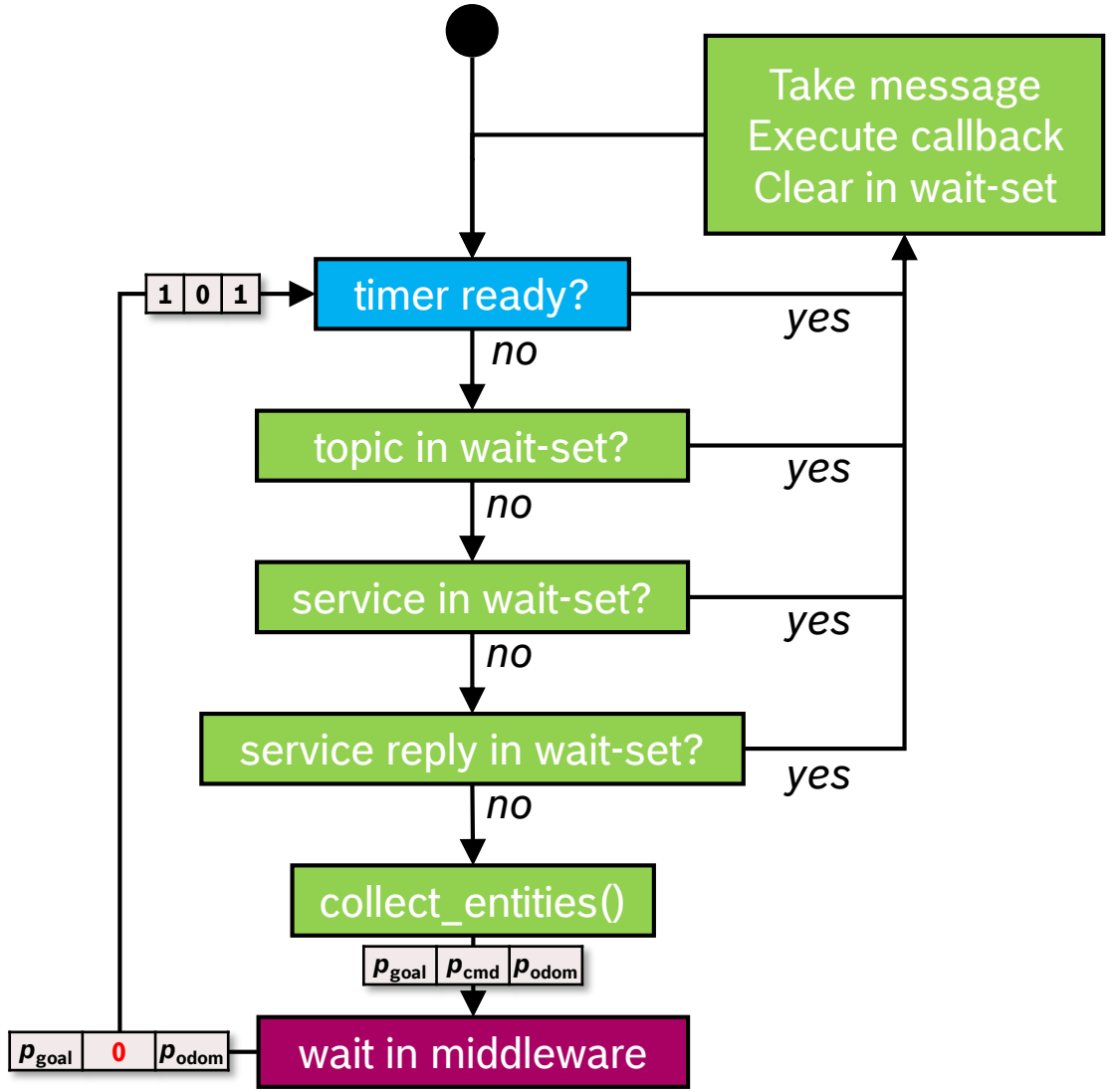
Non-preemptive priority + round-robin



Ralph Lange (CR/AEE1) | 2020-12-16

Scheduling Semantics

Non-preemptive priority + round-robin



No FIFO processing in case of congestions!

D. Casini, T. Blass, I. Lütkebohle, and B. Brandenburg: “Response-Time Analysis of ROS 2 Processing Chains under Reservation-Based Scheduling”, *Proc. of 31st ECRTS 2019*, Stuttgart, Germany, July 2019.

Discussion

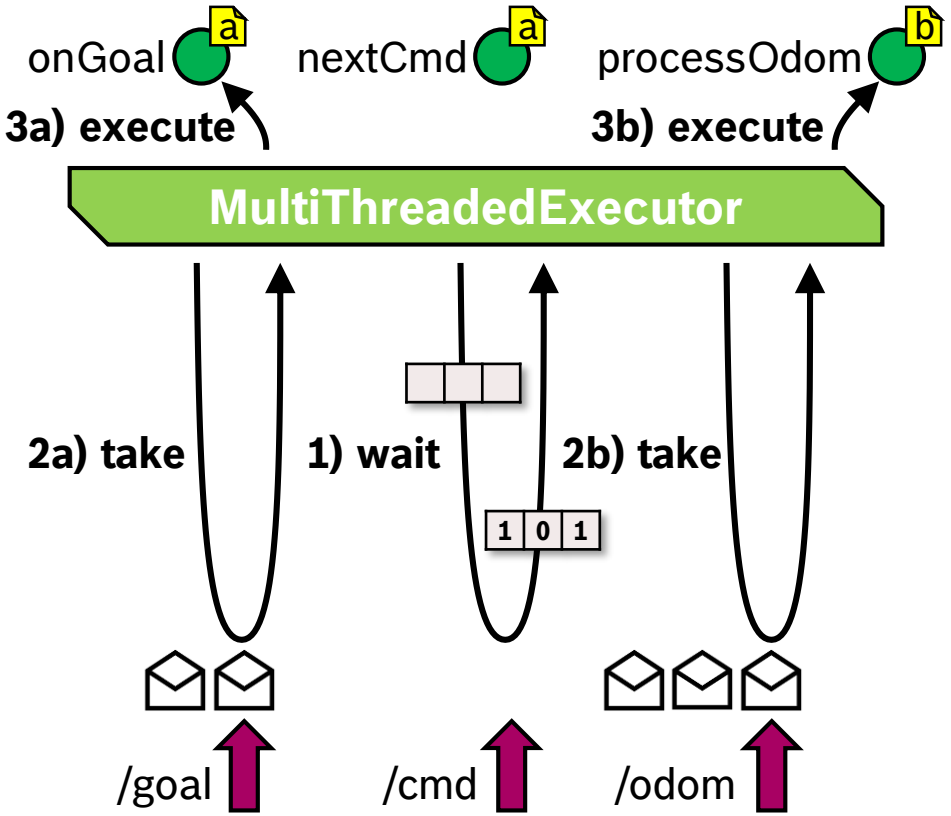
Requirements

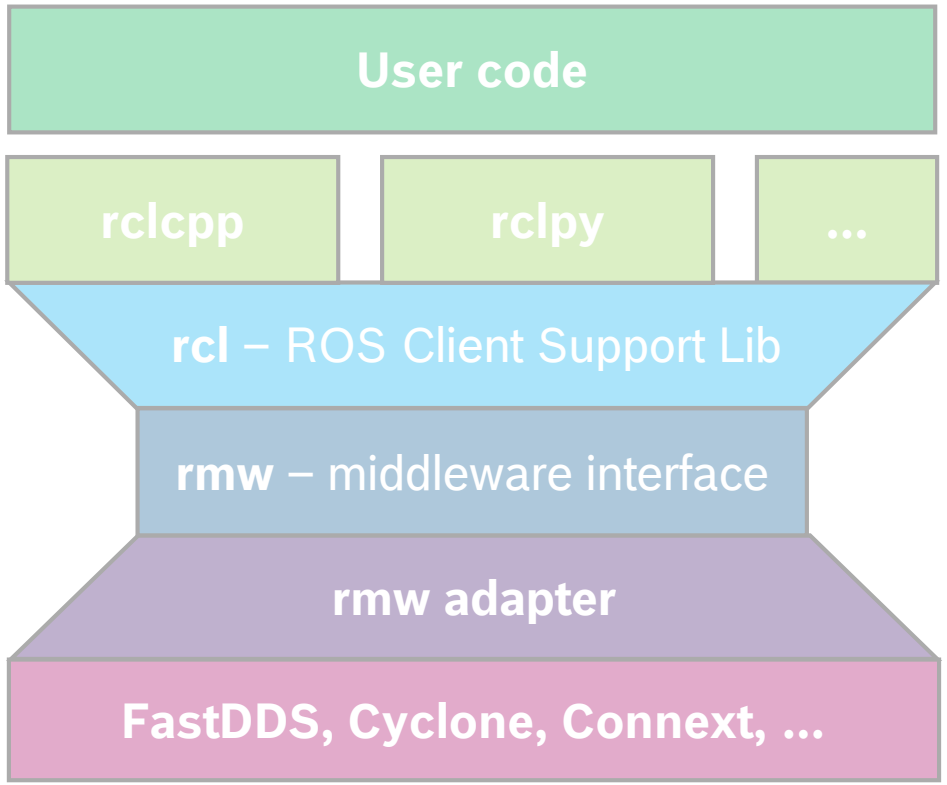
- End-to-end latency guarantees
- Support for mixed real-time criticality
- Parallelization
- Determinism

On-going works

- Runtime overhead by layered design
  - Costly wait-set operations
- Mapping to OS scheduling mechanisms
- FIFO ordering (by message timestamps)

Multi-Threaded Executor





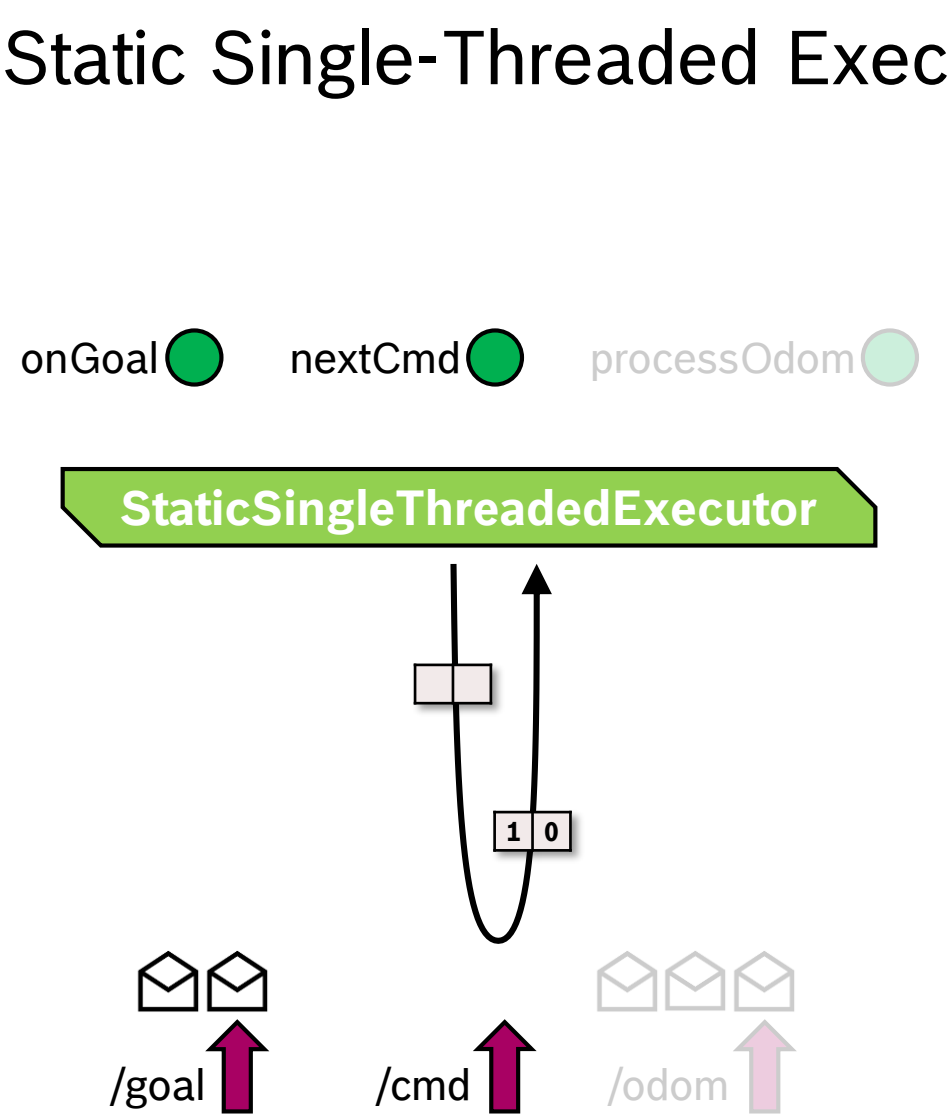
```
...

auto secondGroup = create_callback_group(type);

...
```

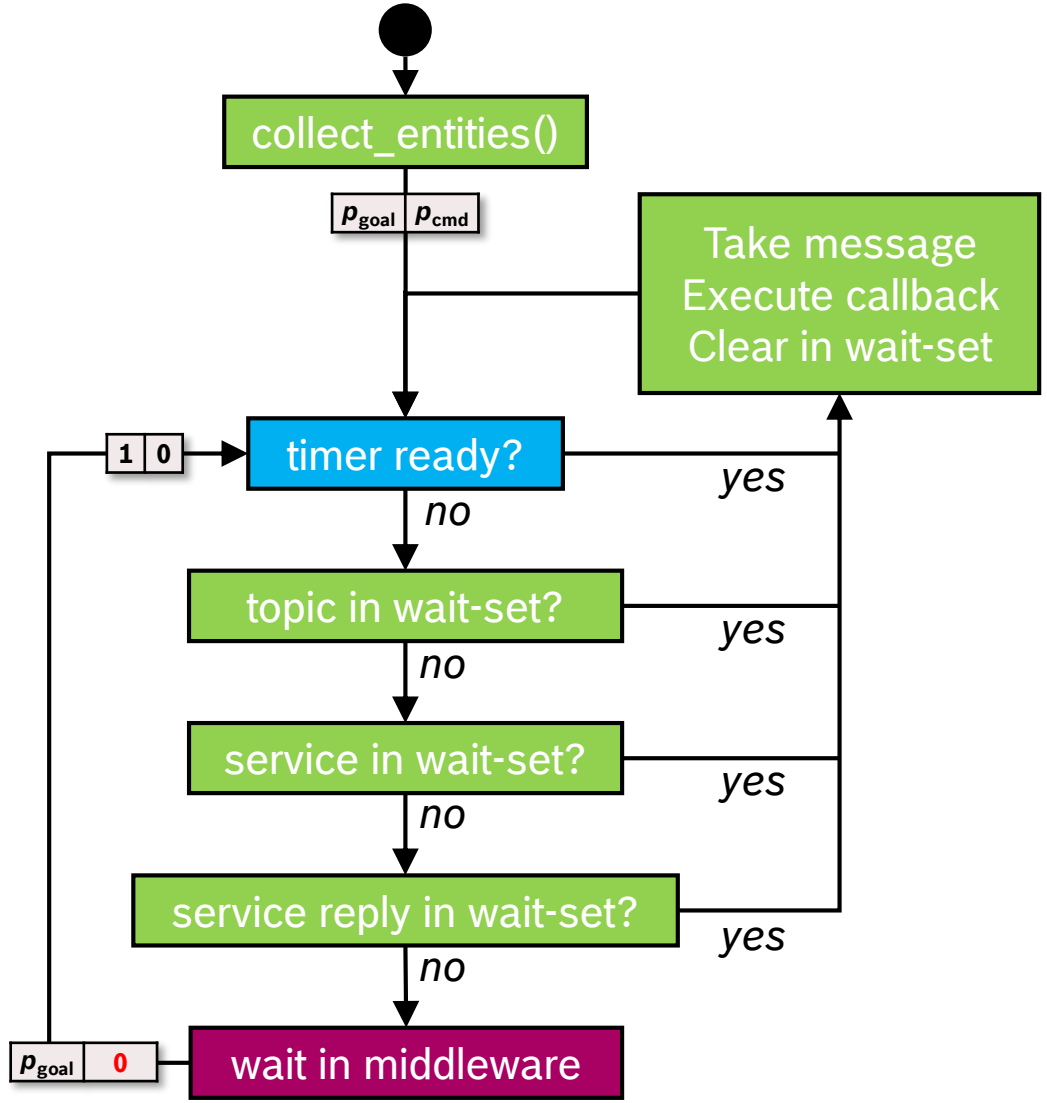
where type is `rclcpp::CallbackGroupType::MutuallyExclusive`
or `rclcpp::CallbackGroupType::Reentrant`

```
...

rclcpp::SubscriptionOptionsWithAllocator<..> options;
options.callback_group = secondGroup;

mySub = create_subscription<..>("/odom",
            rclcpp::SensorDataQoS(), processOdom, options);

...
```

```
...

myTimer = create_wall_timer(100ms, myCallback, secondGroup);

...
```

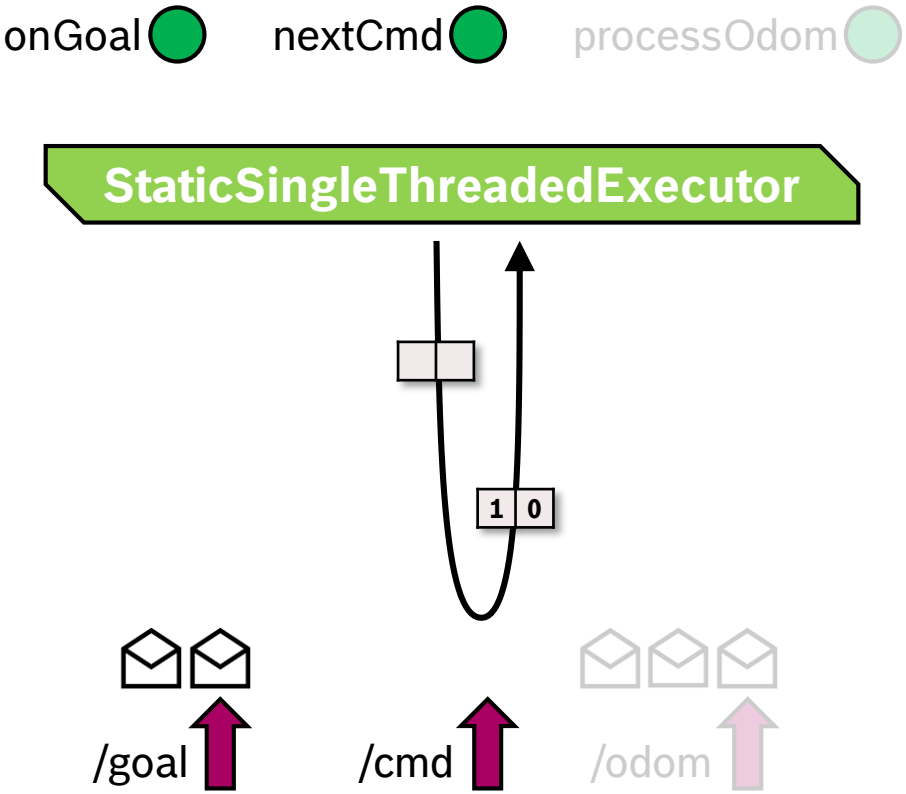
Static Single-Threaded Executor





Ralph Lange (CR/AEE1) | 2020-12-16

Static Single-Threaded Executor



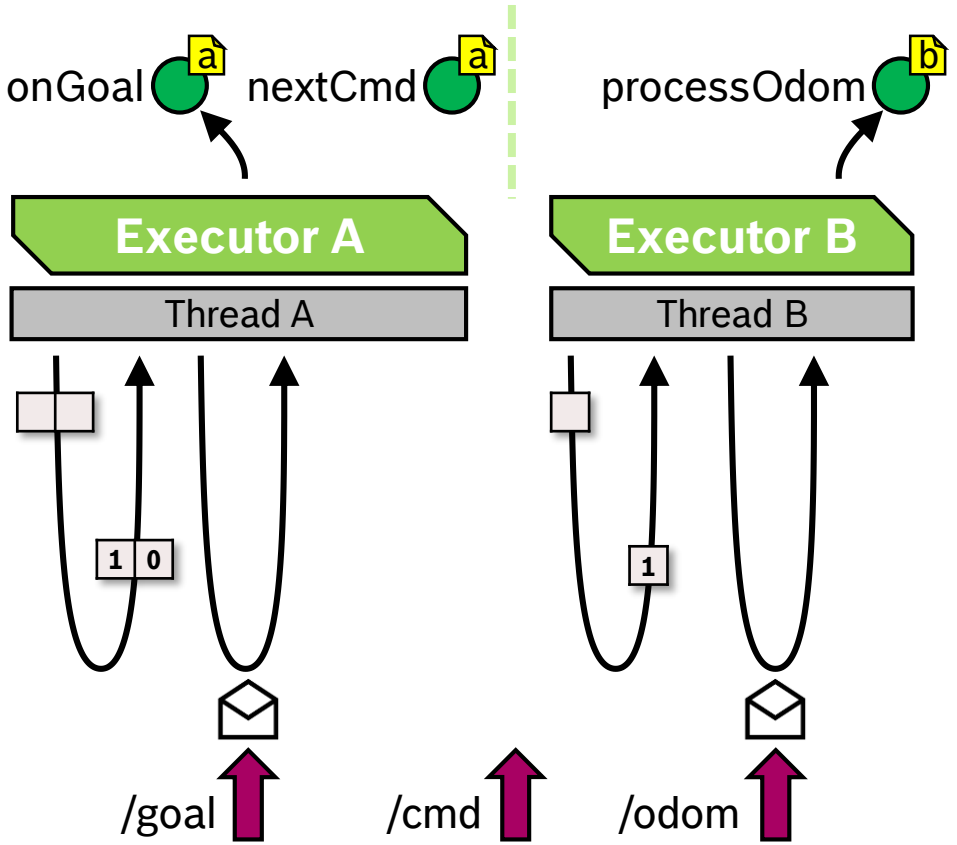

```
StaticSingleThreadedExecutor executor;
executor.add_node(node);


void nextCmd(Cmd msg)
{
  if (msg == "activate_process_odom")
  {
    processOdomSub_ = create_subscription(...);
  }
}
```


Ralph Lange (CR/AEE1) | 2020-12-16

Callback-group-level Executor

is NOT another Executor

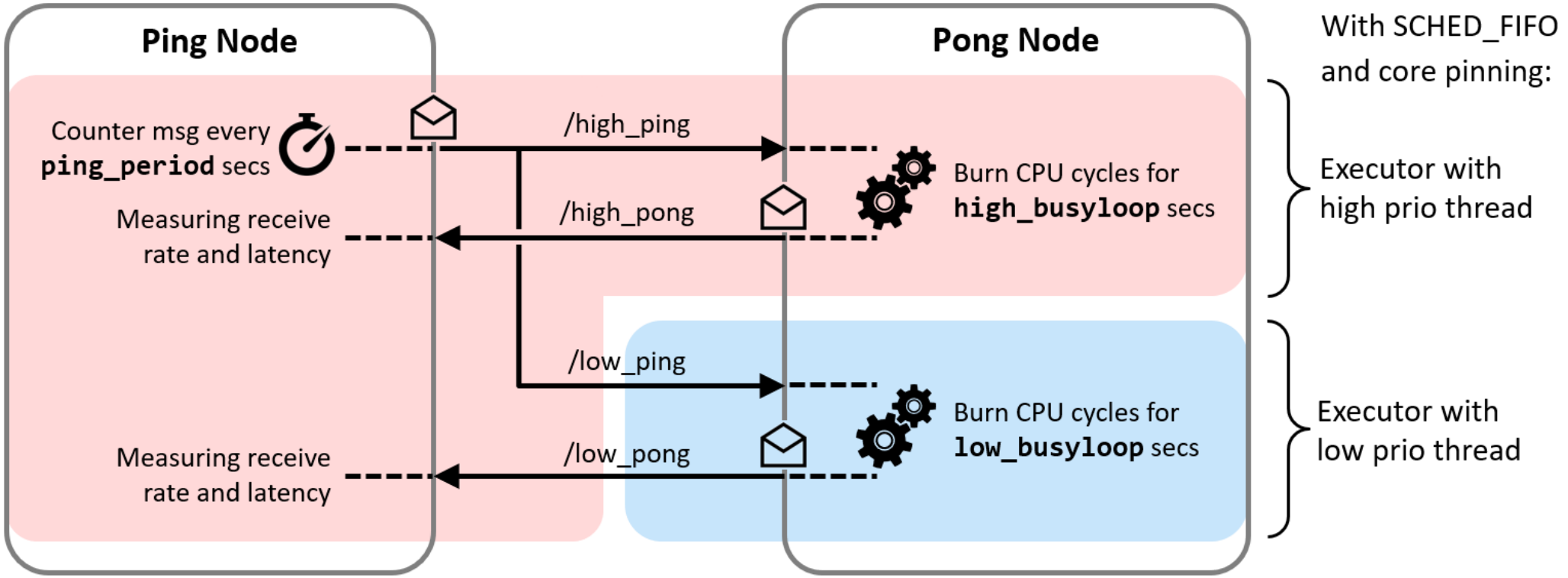
Callback-group-level Executor



Support mixed real-time criticality in a node

- Refines interface of Executor to callback groups
- Prototype presented by me at ROSCon 2018
- Recently brought mainline by Pedro Pena and William Woodall (many thanks!)
- Implemented for all Executors in rclcpp now
- Available in Rolling release

The cbg_executor_demo Package

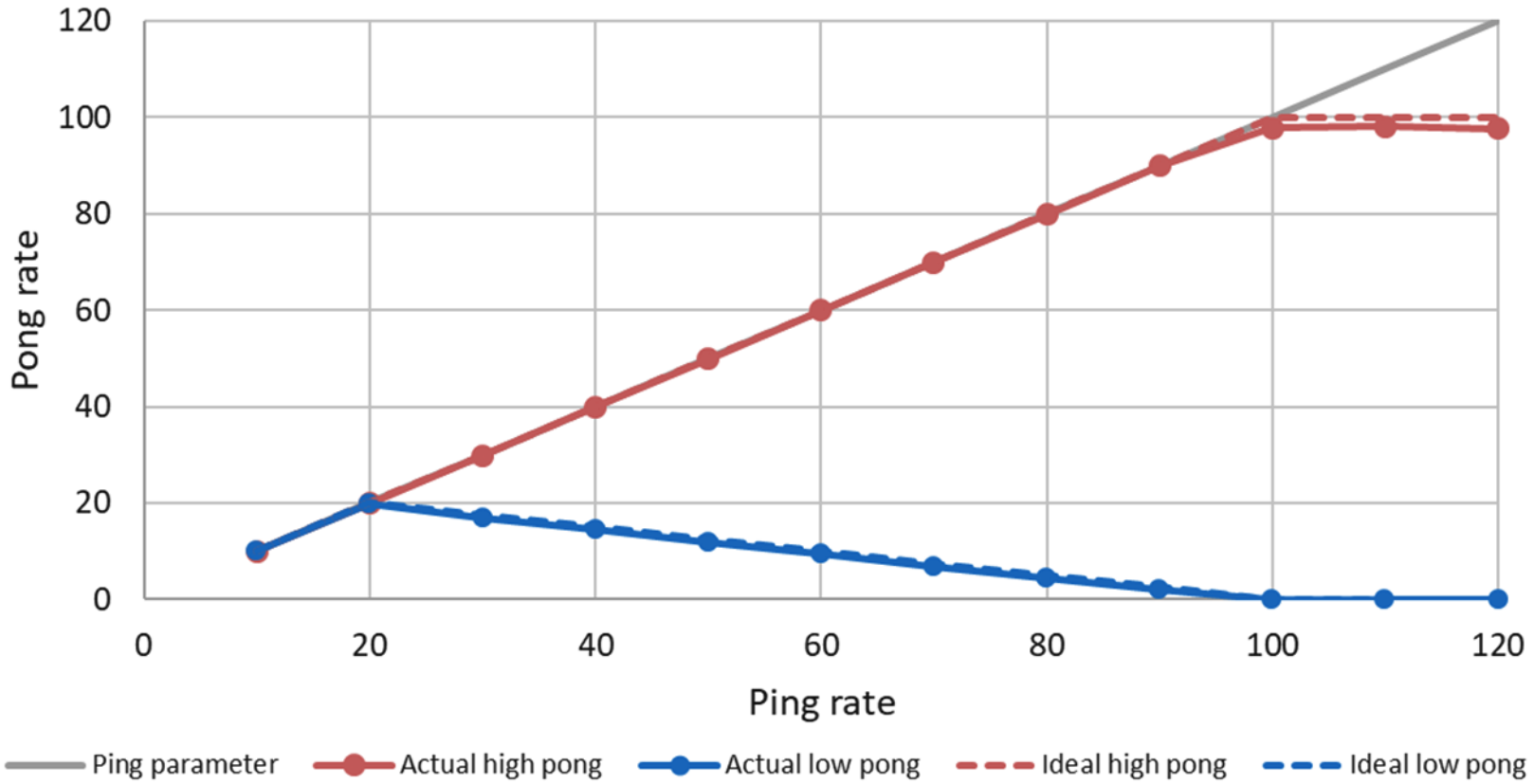


Source code at https://github.com/boschresearch/ros2_demos

Ralph Lange (CR/AEE1) | 2020-12-16

BOSCH

The cbg_executor_demo Package




```
high_busyloop = 0.01 s

low_busyloop = 0.04 s
```

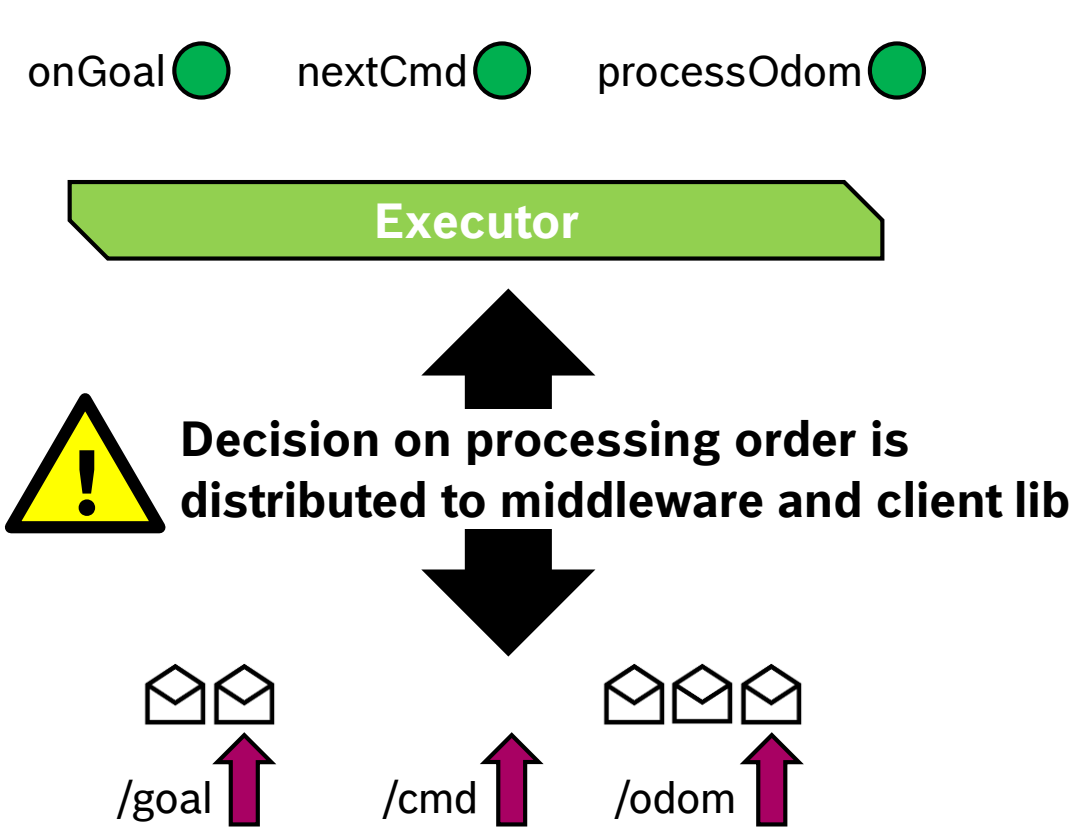
Source code at https://github.com/boschresearch/ros2_demos

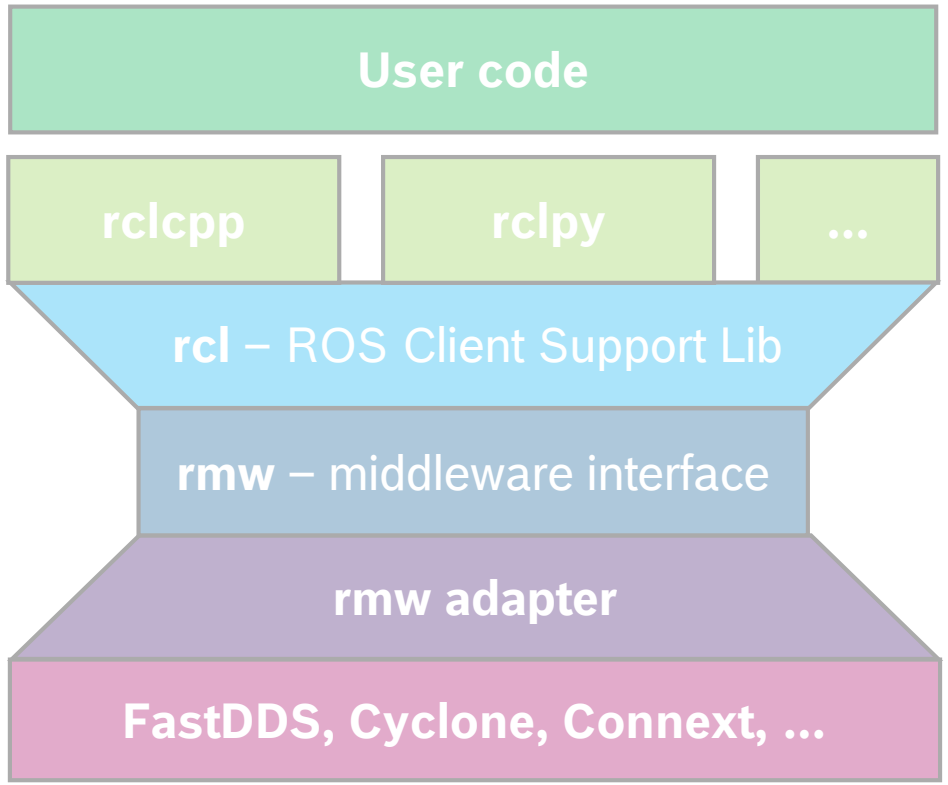
Ralph Lange (CR/AEE1) | 2020-12-16

BOSCH

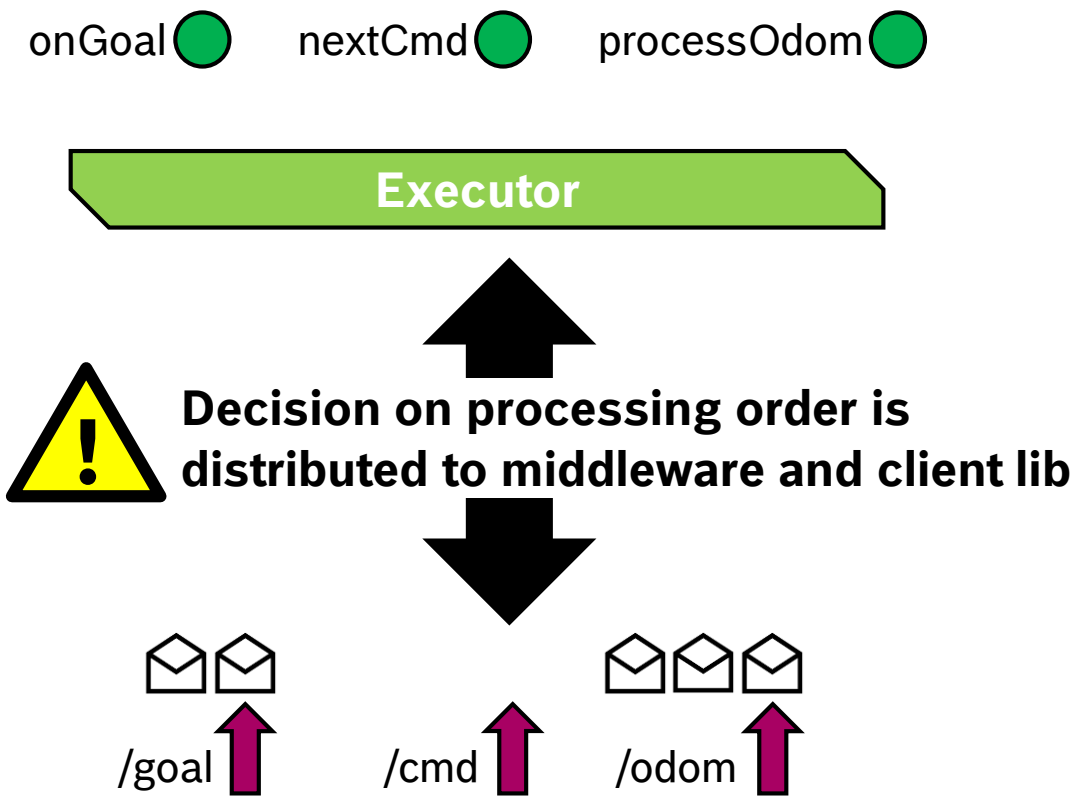
... but no solution for determinism
or at least FIFO ordering

Design Revisited





Design Revisited



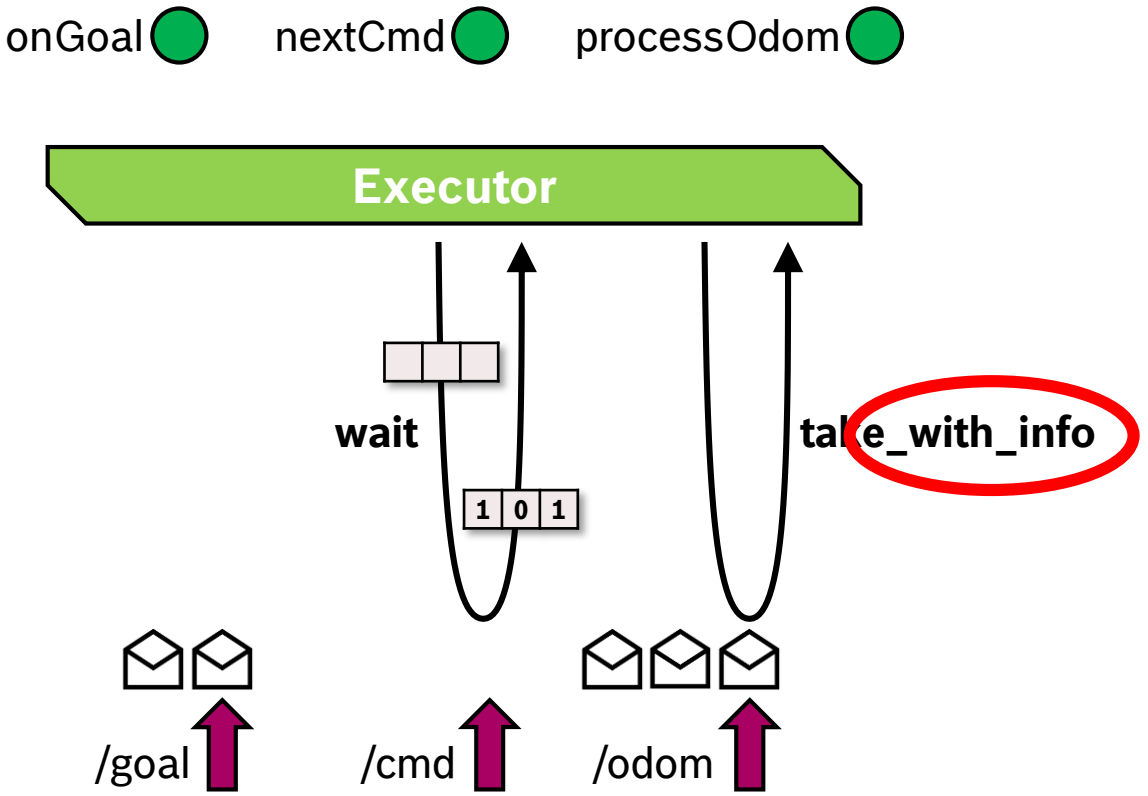
Ideas:

1. Decide completely in middleware
   - Lack of application knowledge
2. Additional queue in client library
   - Thwarts middleware QoS
3. Comprehensive view on middleware
   - Expensive synchronization

Many subtle technical issues:

- Memory management
- Integration of timers
- Access to DDS metadata

Message Info (since Foxy)



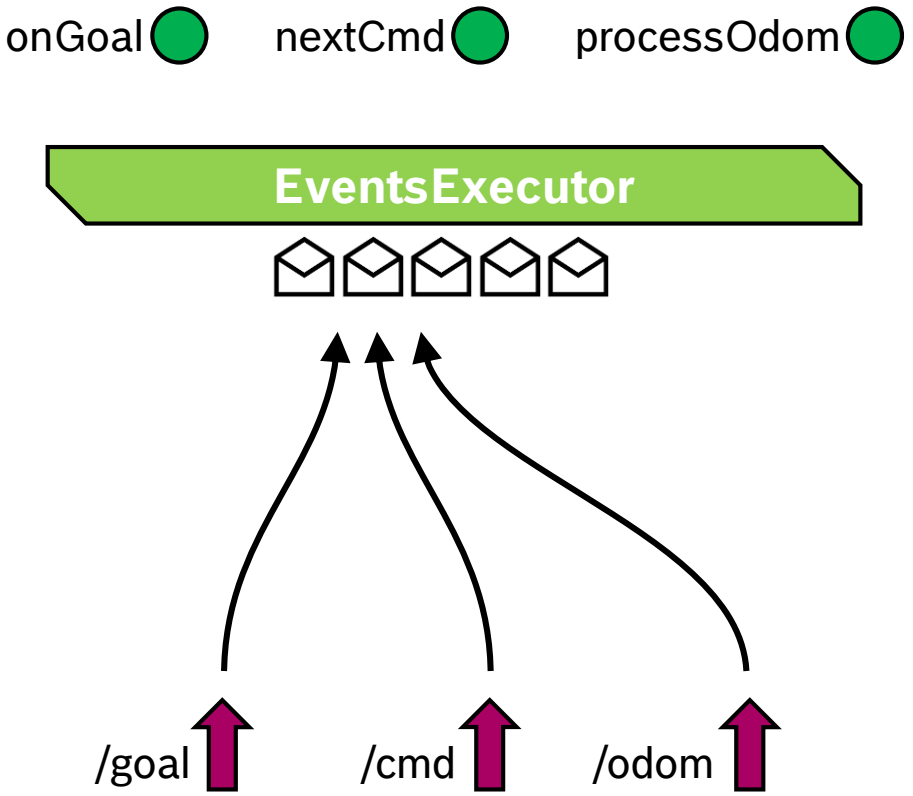
Ideas:

1. Decide completely in middleware
   - Lack of application knowledge
2. Additional queue in client library
   - Thwarts middleware QoS
3. Comprehensive view on middleware
   - Expensive synchronization

Many subtle technical issues:

- Memory management
- Integration of timers
- Access to DDS metadata

EventsExecutor (Proof of Concept)



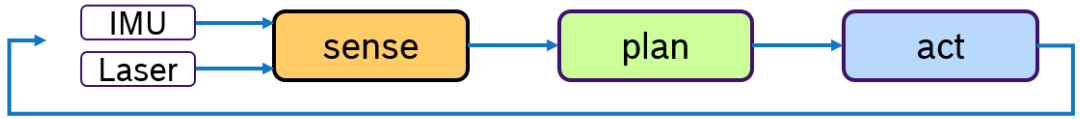
Thread at https://discourse.ros.org/t/ros2-middleware-change-proposal/

- Improved performance
- FIFO ordering
- Possible to use DDS listeners
- Event queue or work queue?

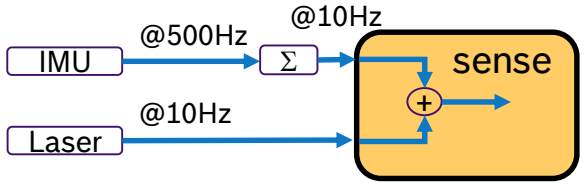
rclc Executor for micro-ROS

Typical Execution Patterns

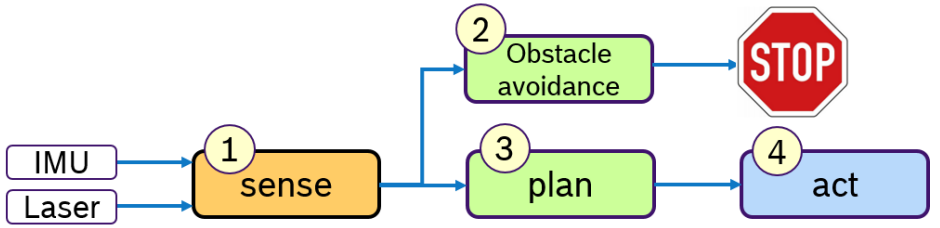
- Control loops



- Data fusion



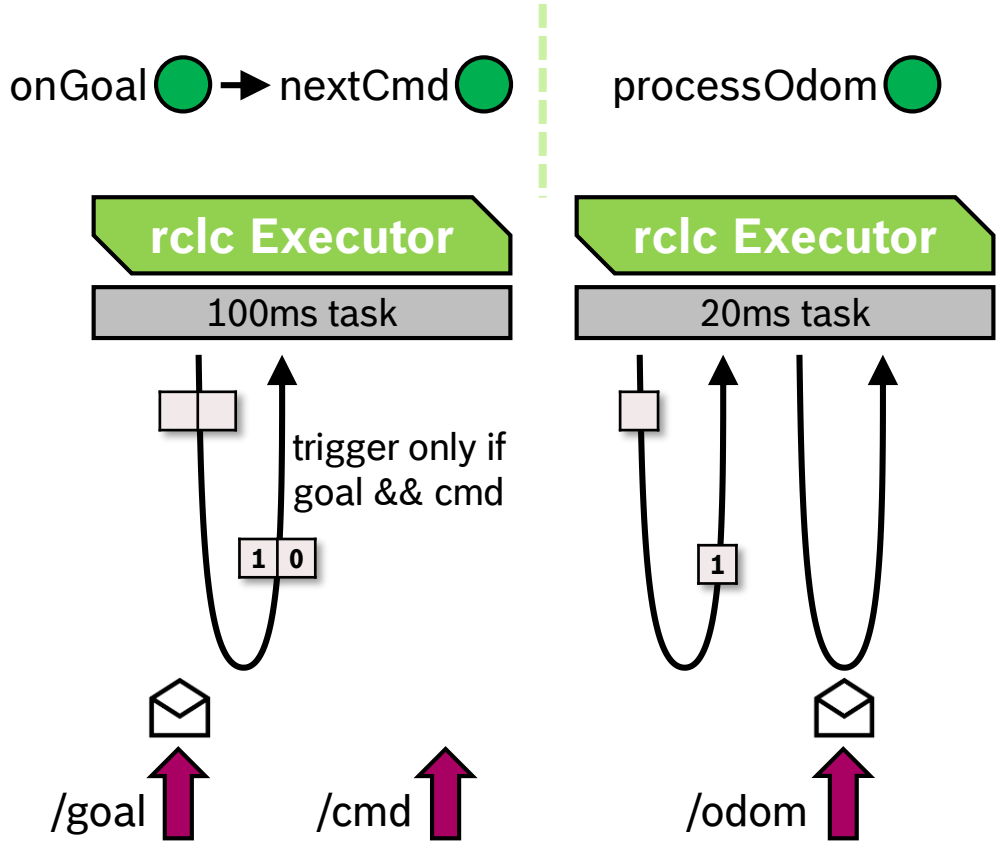
- Prioritized paths



Ralph Lange (CR/AEE1) | 2020-12-16

Key Concepts of rclc Executor

- Individual registration of each callback
  - Not uncommon in deeply embedded software
- User-defined processing sequence
- Custom trigger conditions
- Optional: LET semantics



Source code at https://github.com/ros2/rclc/

Conclusions on Execution Management in ROS 2

- Very different semantics compared to ROS 1
  - No FIFO ordering in case of congestions
- Decision on processing order is distributed to middleware and client library
  - Key questions: Determinism? Integration with middleware QoS?
  - On-going discussion – join middleware and real-time working group
- Several new concepts available in Foxy and Rolling

Looking forward to your questions!

Dr. Ralph Lange
Bosch Corporate Research

ralph.lange@de.bosch.com
github.com/ralph-lange

BOSCH