Content Filtering in ROS 2

ROSCon 2022

Speaker:
- Eduardo Ponz Segrelles, Project Manager @ eProsima
eduardoponz@eprosima.com

October 21st, 2022
Agenda

01 Motivation
- What is the problem we are trying to solve
- How is now tackled

02 Content Filtering
- Reader & Writer side
- Fast DDS optimizations
- Implementation in ROS 2 when middleware does not offer support

03 Usage in ROS 2
- Available APIs and how to use them
- Writer side filtering in ROS 2 with Fast DDS

04 Bandwidth savings
- Benchmark results
- Comparison with current “solutions”
Motivation

Subscribe to a subset of the topic data

Filter by data source
With tagged data, a subscriber may only want data from specific sources

Topic

Alarms
A monitoring application may fire an alarm if a certain field within a topic samples goes above/below a threshold

Load balancing
If computation is expensive on the receiver side, several subscribers may be set to process data in turns
Motivation

Current solutions

Application level filtering

1. Bandwidth utilization
2. Subscriber-side computation
3. Memory and latency overhead
4. CPU usage
5. Maintainability

Splitting the topic

1. Resource utilization
2. Logic management
3. Possible duplicities
4. Subscriber side business logic in the publisher
5. Maintainability
Motivation

Real ROS 2 problem

* Borrowed from ROS 2 CFT design by Mr. Tomoya Fujita
Content Filtered Topics

A Topic extension to let ROS 2 filter messages based on an SQL-like expression
Main features

**Content Filtered Topic main features**

- Configured at subscriber side
- SQL-like syntax for expression with parameters
  - Expression: ‘node = %0’
  - Parameters: ['my_node']
- Each subscriber may set a different filter
- Possibility of filter on the publication side
- Dynamically configurable
- DDS native support
ROS 2 filtering with Fast DDS

How does CFT work
ROS 2 filtering with Fast DDS

How does CFT work

Publisher application

rclcpp  rclpy  rcljava

rcl
rmw
rmw_fastrtps

DataWriter History
FAST DDS
How does CFT work

Filtering before serialization
ROS 2 filtering with Fast DDS

How does CFT work

Publisher application
- rclcpp
- rclpy
- rcljava

Transport layer
- rcl
- rmw
- rmw_fastrtps
- DataWriter History
- FAST DDS
ROS 2 filtering with Fast DDS

How does CFT work

Publisher application
- rclcpp
- rclpy
- rcljava
- rcl
- rmw
- rmw_fastrtps
- DataWriter History
  - FAST DDS

Subscriber application
- rclcpp
- rclpy
- rcljava
- rcl
- rmw
- rmw_fastrtps
- DataWriter History
  - FAST DDS

Filtering before deserialization
ROS 2 filtering with Fast DDS

How does CFT work

Publisher application

- rclcpp
- rclpy
- rcljava
- rcl
- rmw
- rmw_fastrtps

DataWriter History

FAST DDS

Subscriber application

- rclcpp
- rclpy
- rcljava
- rcl
- rmw
- rmw_fastrtps

DataWriter History

FAST DDS

Transport layer
# ROS 2 filtering without middleware support

## How does CFT work

<table>
<thead>
<tr>
<th>Publisher application</th>
</tr>
</thead>
<tbody>
<tr>
<td>rclcpp</td>
</tr>
<tr>
<td>rcl</td>
</tr>
</tbody>
</table>

**Middleware implementation**
ROS 2 filtering without middleware support

How does CFT work

Publisher application

rclcpp  rclpy  rcljava

rcl
rmw
rmw_connector

Middleware implementation

Transport layer
ROS 2 filtering without middleware support

How does CFT work

Filtering after deserialization
ROS 2 filtering without middleware support

How does CFT work

Publisher application

rclcpp  rclpy  rcljava

rcl
rmw
rmw_connector

Middleware implementation

Transport layer
Usage in ROS 2

How to configure content filtering in ROS 2
ROS 2 APIs for Content filtering

Code snippet within the context of a constructor of a class inheriting from Node

```cpp
rclcpp::QoS qos(10);
rmw_time_t lease_duration({0, 0});
qos.liveliness(RMW_QOS_POLICY_LIVELINESS_AUTOMATIC);
qos.liveliness_lease_duration(lease_duration);

rclcpp::SubscriptionOptions sub_options;
sub_options.content_filter_options.filter_expression = "data = %0";
sub_options.content_filter_options.expression_parameters = {"'Hello!'"};

subscription_ = this->create_subscription<std_msgs::msg::String>(
    "topic",
    qos,
    std::bind(&MinimalContentFilteringSubscriber::topic_callback, this, _1),
    sub_options);

if (!subscription_->is_cft_enabled()) {
    RCLCPP_WARN(this->get_logger(), "Content filter is not enabled since it's not supported");
} else {
    RCLCPP_INFO(this->get_logger(), "Subscribed to topic with content filtering");
}
```
Bandwidth savings

Only sending what you need
Benchmark of different filtering strategies

Writer-side filtering savings using Fast DDS

- 1 Reliable pub
- 10 Reliable subs each
  filtering 9 out of every
  10 samples.
- ~60 KiB data samples
- Pub rate: 1 Hz
- Duration: 60 s