Quality of Service Policies for ROS2 Communications

ROSCon 2019
Macau 2019–11–01

AWS Robotics
Emerson Knapp eknapp@amazon.com
Github: emersonknapp
AWS Robotics

Contributing actively to ROS2 to help accelerate progress in the robotics community

[GitHub](https://github.com/aws-robotics)

[GitHub](https://github.com/orgs/ros2/teams/aws-robotics)

Members of ROS2 working groups – Security, Tooling

Agenda

- Introduction to Quality of Service
- QoS Policies available in ROS2
- Using QoS in ROS2
Introduction to Quality of Service
What is Quality of Service (QoS)?

Advanced behavior of publish and subscribe (pub/sub) communications

QoS “type” = “Policy”
Collection of Policies = “Profile”
Exposed QoS policies are pulled from DDS specification

Formalized as ROS2 native concept

- Non-DDS RMWs can choose to provide QoS

Publisher/subscription

<table>
<thead>
<tr>
<th>Topic</th>
<th>Message type</th>
<th>QoS Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmw (DDS)</td>
<td>&quot;QoS for Free!&quot;</td>
<td></td>
</tr>
<tr>
<td>rmw (non-DDS)</td>
<td>&quot;QoS implemented&quot;</td>
<td></td>
</tr>
<tr>
<td>rmw_simple</td>
<td>&quot;QoS not supported&quot;</td>
<td></td>
</tr>
</tbody>
</table>

RMW API
QoS profile matching

Publishers offer profile
Subscriptions request
Policies have compatibility rules
Subscriptions get “Actual QoS”
QoS policies in ROS2
QoS policy: History

“How many messages to keep locally?”

Compare ROS1 “queue_size”

Legal values
- KEEP_ALL
- KEEP_LAST + depth

Example:
- Image processing queue

Compatibility
N/A—does not apply to matching
QoS policy: History

Publisher

Message
Message
Message
Too old

History

Message
Message
Message

Message

Subscription

“missed”
QoS policy: **Durability**

“Should Publishers provide old messages?”

Compare ROS1 “latching”

**Legal values**
- VOLATILE: Late joining subscriptions receive nothing
- TRANSIENT_LOCAL: Publishers provide old messages

**Compatibility**
- TRANSIENT_LOCAL > VOLATILE

**Example:**
- Latest mission state machine
QoS policy: **Durability**

**Publisher**
- History: 2
- Durability: TRANSIENT_LOCAL

- msg1
  - time=1

- msg2
  - time=5

**Subscription**
- History: 1
- Durability: TRANSIENT_LOCAL
- Joined at time: 6

msg1
- time=1

msg2
- time=5
QoS policy: Reliability

“Do messages have to be delivered/received?”

Legal values
• BEST EFFORT: No delivery guarantee
• RELIABLE: Guaranteed delivery*

Compatibility
• RELIABLE > BEST EFFORT

Example:
• Visualizer for humans doesn’t need retry
• Safety critical update must get through
QoS policy: **Reliability**

- **Diagnostic Publisher**
  - BEST_EFFORT
  - RELIABLE

- **Distance Sensor**
  - RELIABLE

- **Safety Critical System**

- **Cellular Network**

- **Web Dashboard**
QoS policy: **Lifespan**

“How long before an un-sent message is not useful anymore?”

**Legal values**
- Duration: duration

**Compatibility**
- Offered duration \(\geq\) Requested duration

**Example:**
Estimated pose when moving.
QoS policy: **Lifespan**

- **Publisher**
- **Message**
- **Message**
- **Message**
- **Message**
- **History**

**Lifespan expired**
New concept - event callbacks

Some QoS Policies generate “events”

Subscriptions already had “message callback”—now add “QoS Event callback”

Publishers get them too!
QoS policy: Deadline

“How often must I send messages?” (minimum frequency)

Legal values
• Period: duration

Compatibility
• Offered period <= Requested period

Callbacks
• Publisher — OfferedDeadlineMissed
• Subscription — RequestedDeadlineMissed

Example:
/cmd_vel safety watchdog!
QoS policy: **Deadline**

- **OfferedDeadlineMissed callback**
  - Message time = 4
  - Message time = 2
  - Message time = 1

- **Publisher Deadline = 1**

- **RequestedDeadlineMissed callback**
  - Message time = 4
  - Message time = 2
  - Message time = 1

- **Subscription Deadline = 1**

Time = 3
QoS policy: **Liveliness**

“What type of heartbeat do I need to give to prove I’m not dead?”

**Legal values**
- Kind: AUTOMATIC, MANUAL_BY_TOPIC, MANUAL_BY_NODE
- Lease Duration: duration

**Compatibility**
- MANUAL_BY_TOPIC > MANUAL_BY_NODE > AUTOMATIC
- Offered lease duration <= Requested lease Duration

**Callbacks**
- Publisher—LivelinessLost
- Subscription—LivelinessChanged
QoS policy: **Liveliness** (AUTOMATIC)

Publisher
Liveliness: AUTOMATIC

rmw implementation

Heartbeat

Subscription
Liveliness: AUTOMATIC

rmw implementation

LivelinessChanged callback

alive + 1

CRASH!

Publisher
Liveliness: Automatic

rmw implementation

Subscription
Liveliness: AUTOMATIC

rmw implementation

LivelinessChanged callback

alive - 1
QoS policy: **Liveliness** (MANUAL*)

- Publisher: Liveliness: MANUAL_BY_NODE
  - LivelinessLost callback
  - Assert liveliness
  - Assert liveliness
  - Assert liveliness
  - Assert liveliness

- Subscription: Liveliness: MANUAL_BY_NODE
  - LivelinessChanged callback
  - LivelinessLost callback
  - Assert liveliness
  - Assert liveliness
  - Assert liveliness
  - Assert liveliness
  - alive - 1
  - alive + 1
Using QoS in ROS2
Using QoS in ROS2 code

New arguments to create_publisher/create_subscription
- QoSProfile structure
- QoSEventCallbacks (deadline, liveliness)

Pre-defined “preset profiles” available

When in doubt, just History
CLI usage

ros2 topic pub -h
ros2 topic echo -h

Takes most QoS policies

Work ongoing to add to `ros2 topic info`
Thank you!