Quality of Service Policies for ROS2 Communications

ROScon 2019
Macau 2019-11-0[illegible]

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

AWS Robotics

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

https://github.com/aws-robotics

https://github.com/orgs/ros2/teams/aws-robotics

Members of ROS2 working groups – Security, Tooling

AWS RoboMaker https://aws.amazon.com/robomaker

Agenda

Introduction to Quality of Service

QoS Policies available in ROS2

Using QoS in ROS2

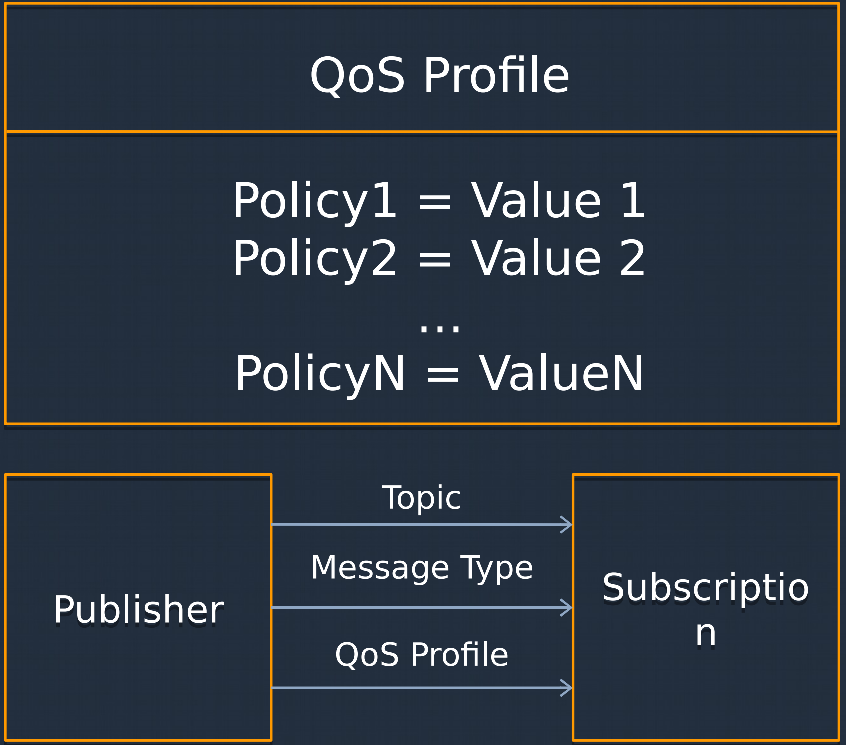
Introduction to Quality of Service

What is Quality of Service (QoS)?

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

QoS “type” = “Policy”

Collection of Policies = “Profile”

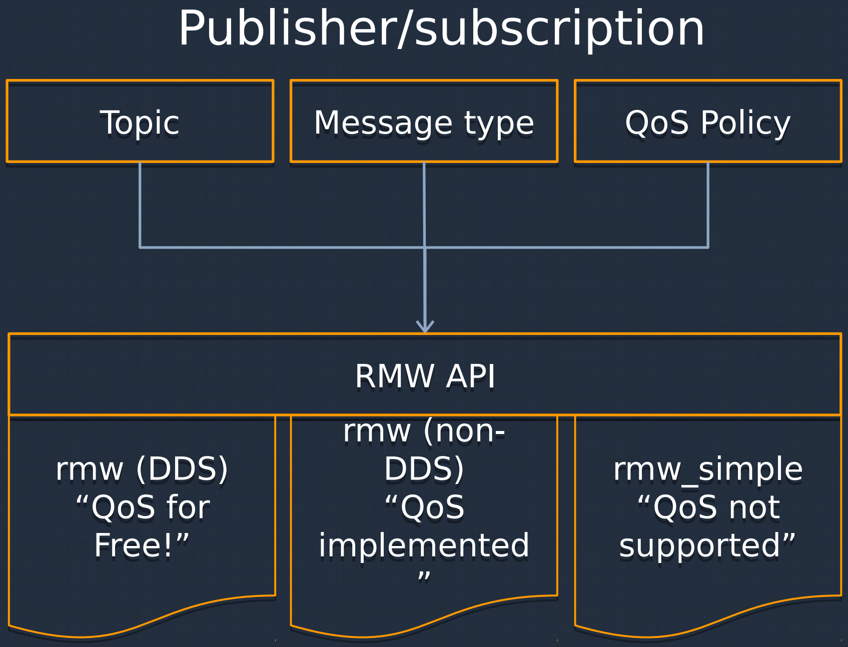


Relationship to DDS QoS

Exposed QoS policies are pulled from DDS specification

Formalized as ROS2 native concept

- Non-DDS RMWs can choose to provide QoS


Publisher/subscription
Topic
Message type
QoS Policy
RMW API
rmw (DDS) “QoS for Free!”
rmw (non-DDS) “QoS implemented”
rmw_simple “QoS not supported”

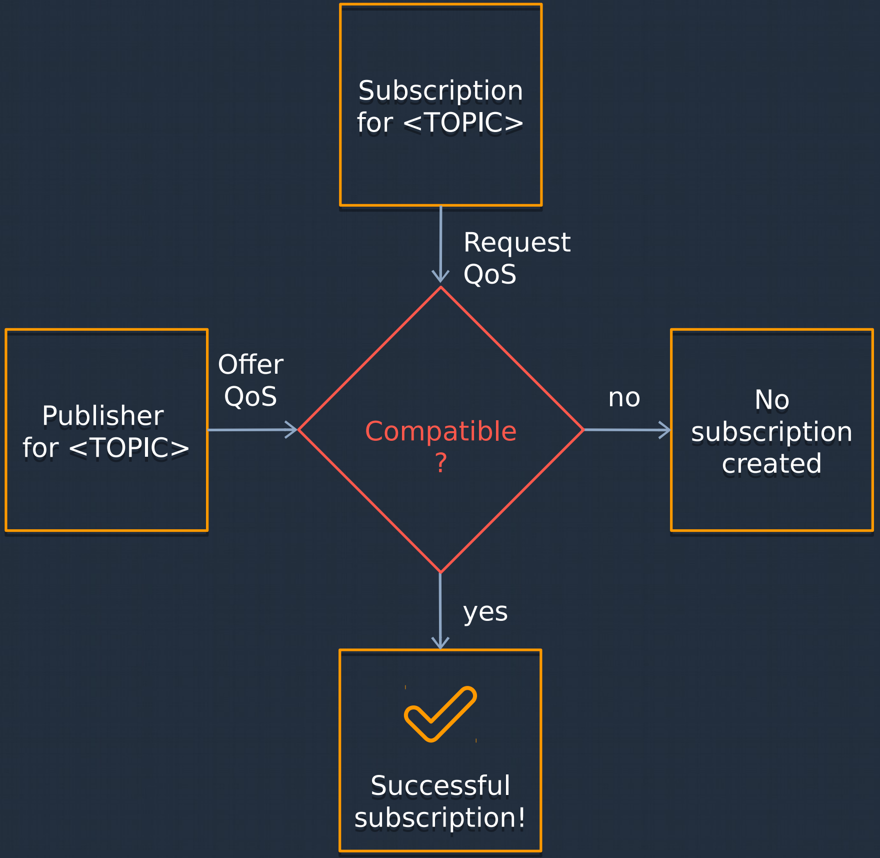
QoS profile matching

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

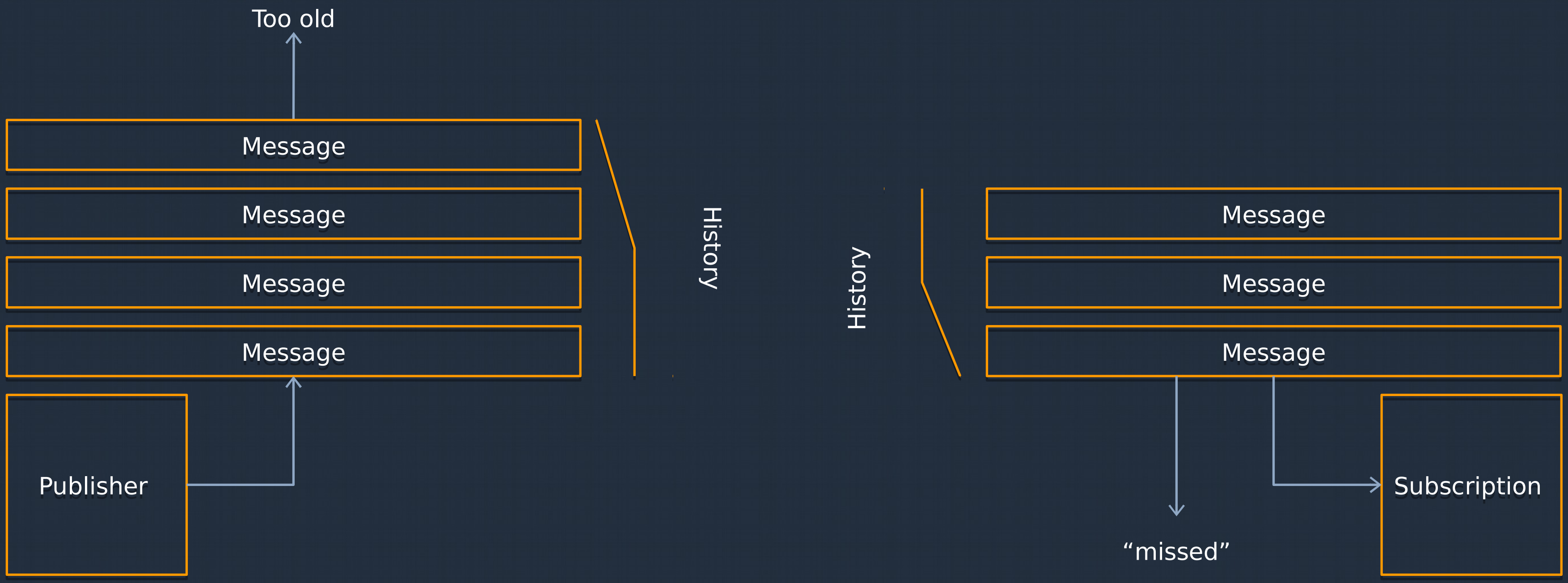
Compatibility

N/A—does not apply to matching

Example:

- Image processing queue

QoS policy: History



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

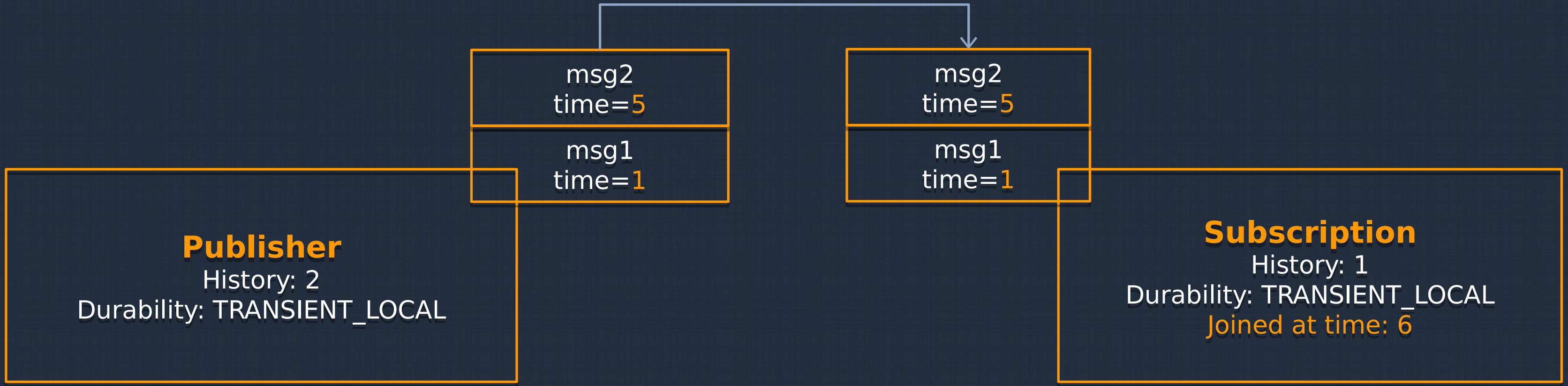
Compatibility

- TRANSIENT_LOCAL > VOLATILE

Example:

- Latest mission state machine

QoS policy: Durability



QoS policy: Reliability

“Do messages have to be delivered/received?”

Legal values

- BEST_EFFORT: No delivery guarantee
- RELIABLE: Guaranteed delivery*

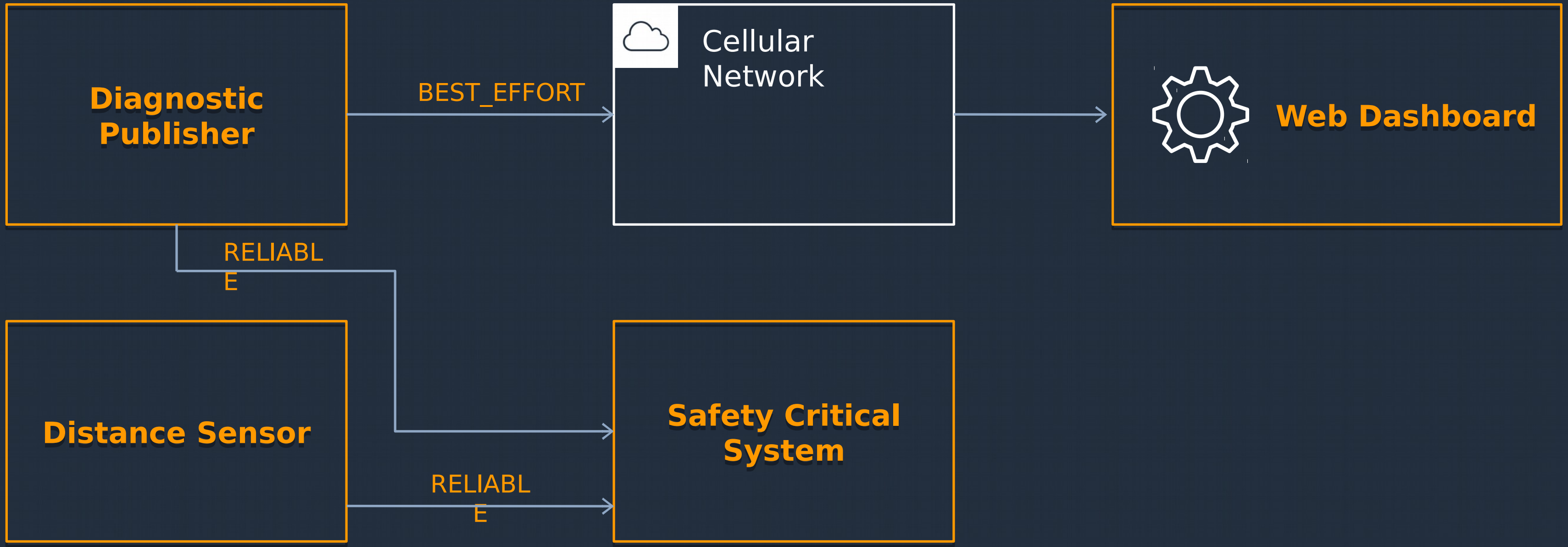
Compatibility

- RELIABLE > BEST_EFFORT

Example:

- Visualizer for humans doesn’t need retry
- Safety critical update must get through

QoS policy: Reliability



QoS policy: Lifespan

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

Legal values

- Duration: duration

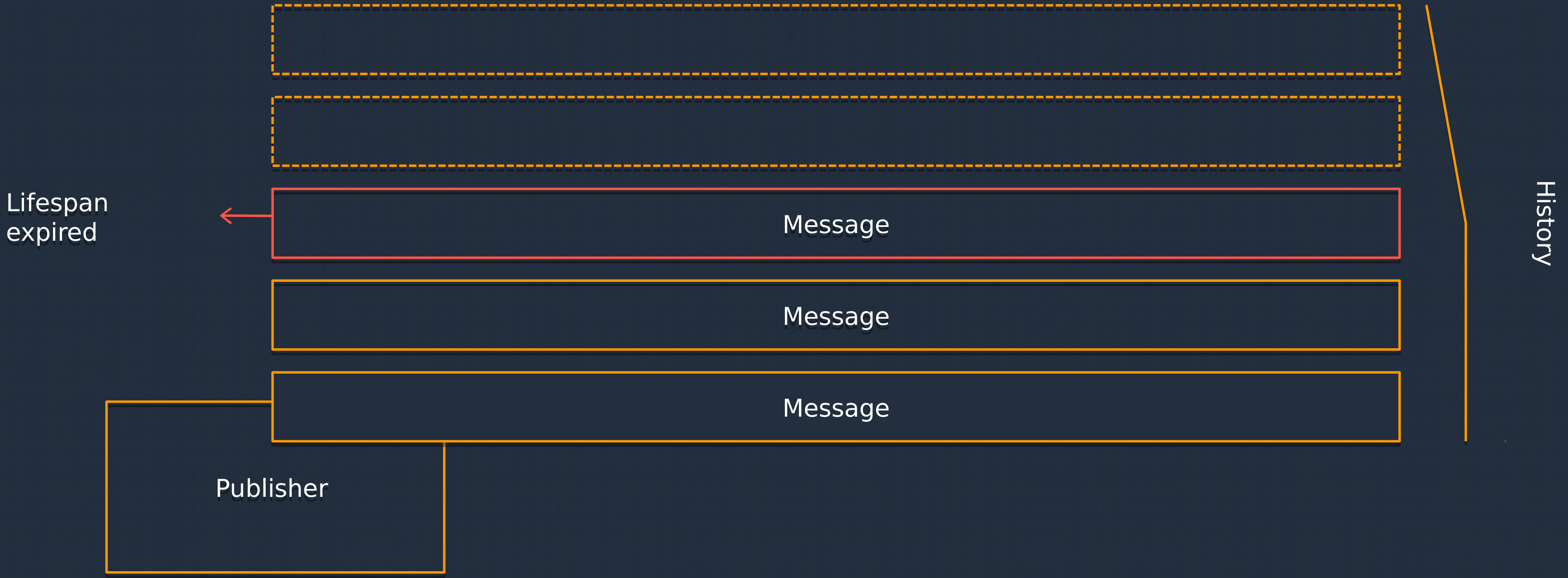
Compatibility

- Offered duration >= Requested duration

Example:

Estimated pose when moving.

QoS policy: Lifespan



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

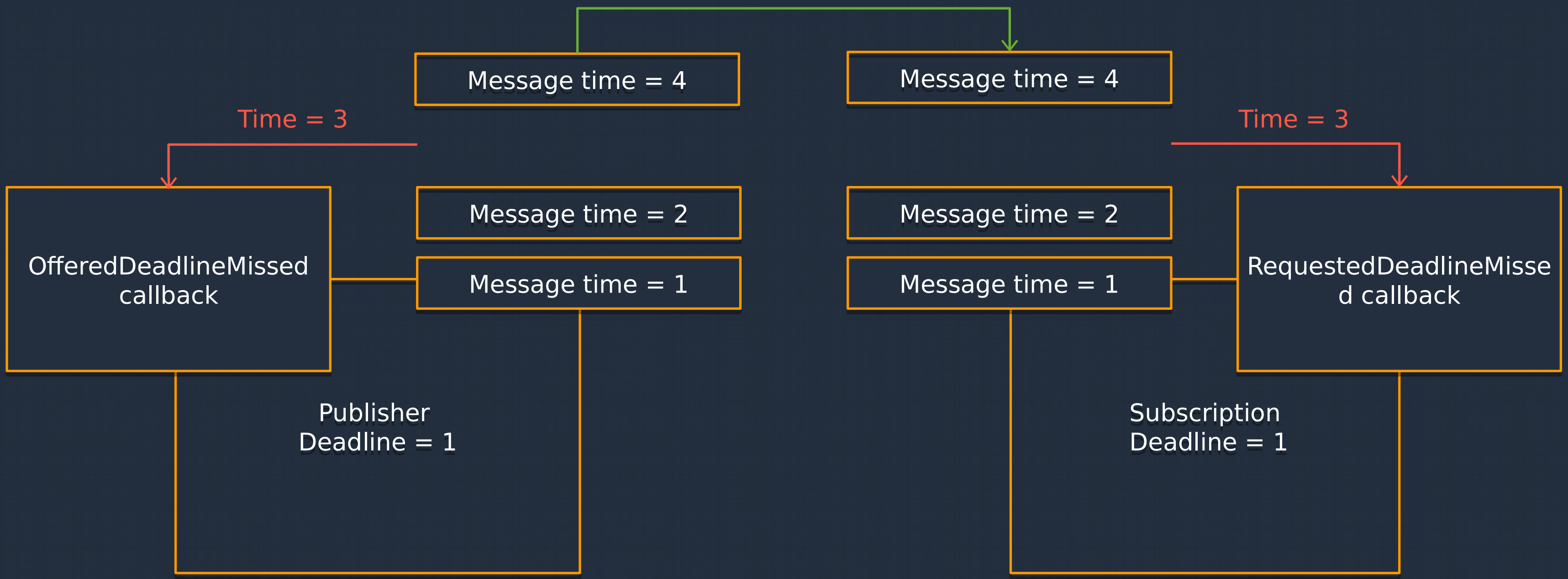
Callbacks

- Publisher — OfferedDeadlineMissed
- Subscription — RequestedDeadlineMissed

Example:

/cmd_vel safety watchdog!

QoS policy: Deadline



QoS policy: Liveliness

"What type of heartbeat to I need to give to prove I'm not dead?"

Legal values

- Kind: AUTOMATIC, MANUAL_BY_TOPIC, MANUAL_BY_NODE
- Lease Duration: duration

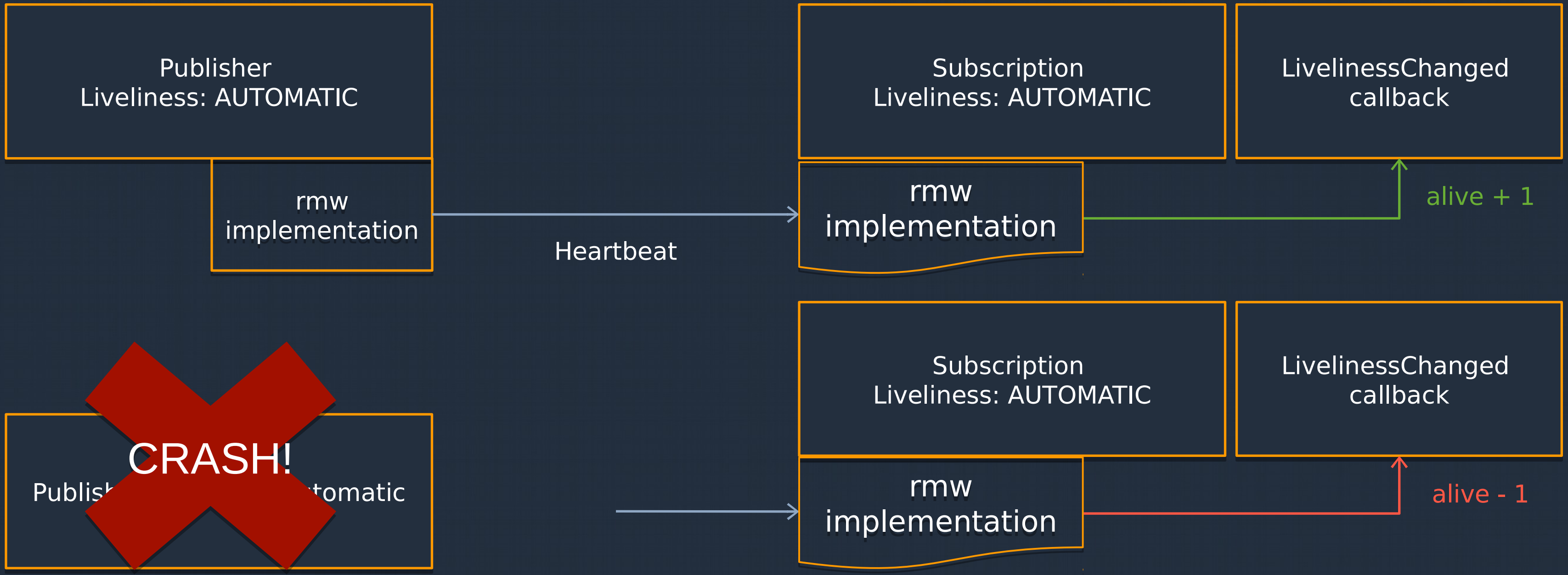
Compatibility

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

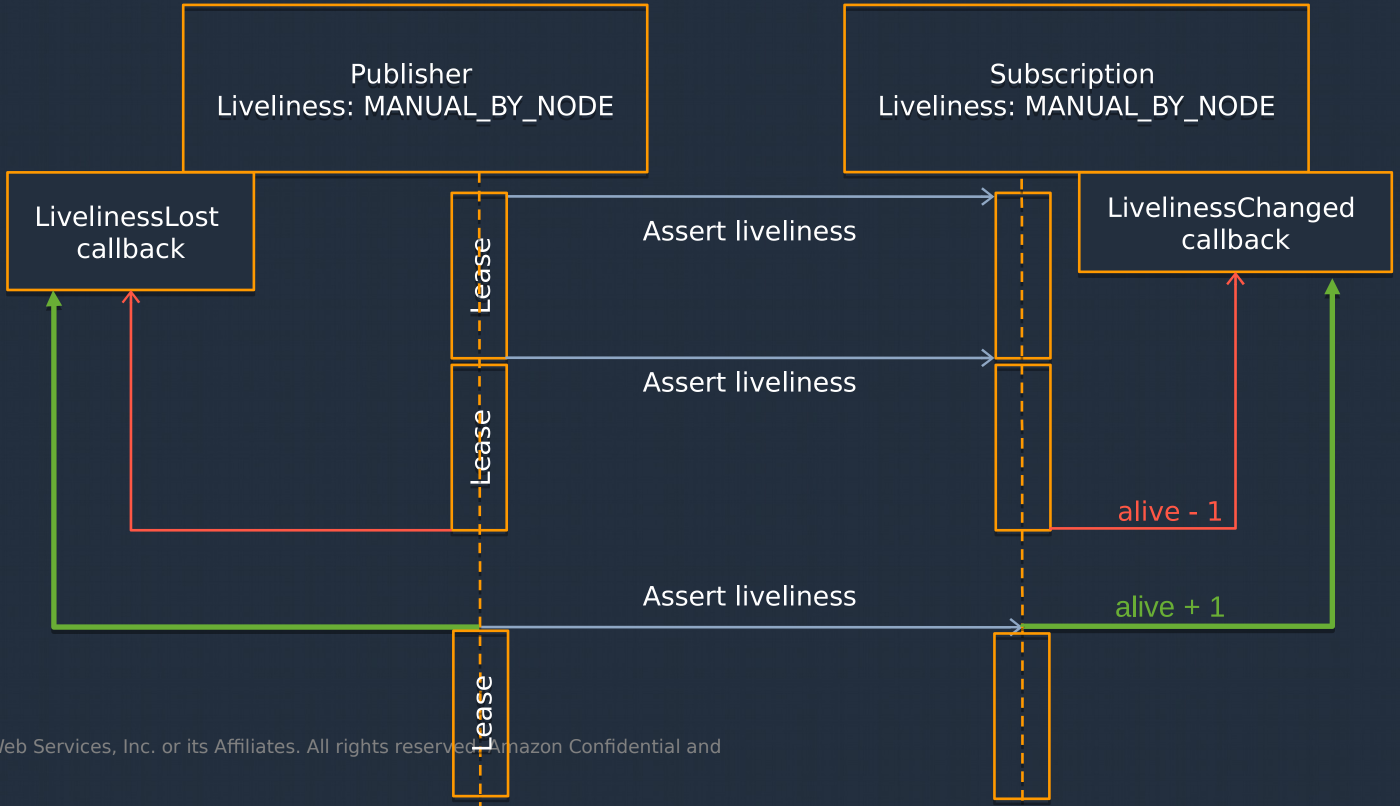
Callbacks

- Publisher—LivelinessLost
- Subscription—LivelinessChanged

QoS policy: Liveliness (AUTOMATIC)



QoS policy: Liveliness (MANUAL*)



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!