RoboFuzz: Fuzzing Robotic Systems over Robot Operating System (ROS) for Finding Correctness Bugs

ESEC/FSE 2022

Seulbae Kim, Taesoo Kim

Georgia Tech College of Computing
School of Cybersecurity and Privacy

SSLab
@GeorgiaTech

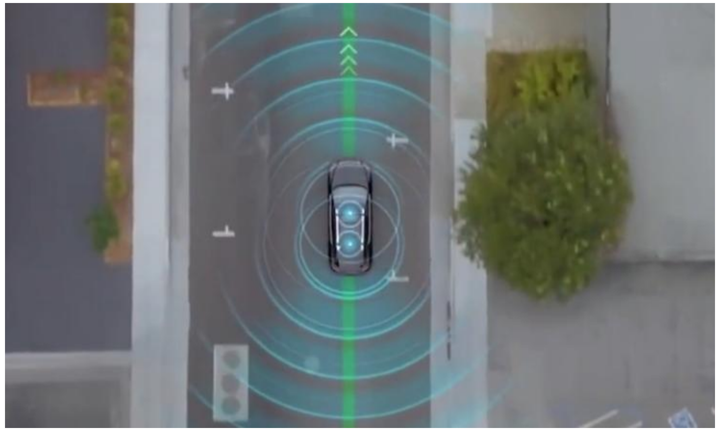
Presented by: **Wonyoung Kim**

Introduction

Agriculture

Service robotics

Planetary Exploration



Autonomous Vehicles

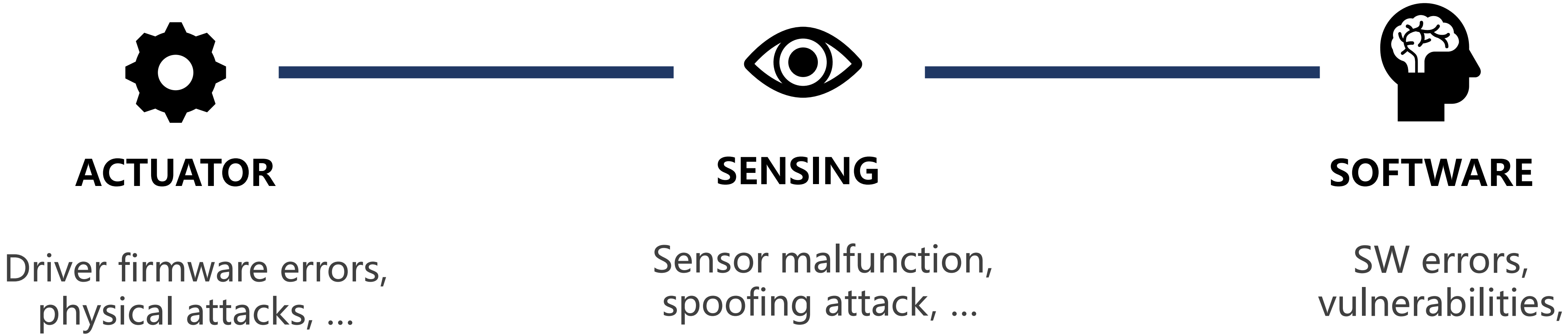
Drones

Factory Logistics

https://www.ros.org

Introduction

- Robotic System is one type of Cyber-Physical Systems
- Motors, sensors and software must work together seamlessly



ACTUATOR

Driver firmware errors, physical attacks, …

SENSING

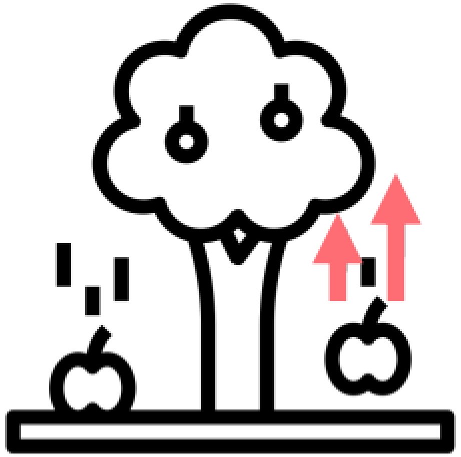
Sensor malfunction, spoofing attack, …

SOFTWARE

SW errors, vulnerabilities,

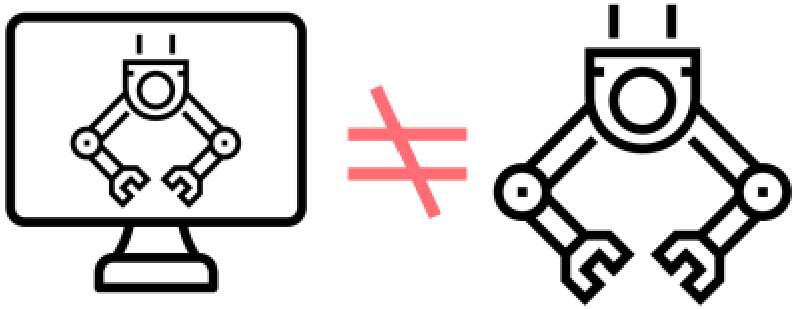
Which types of bugs are we looking for?

- A new class of bugs in robotic systems: **Correctness bugs**



Violation of physical laws

Violation of specification



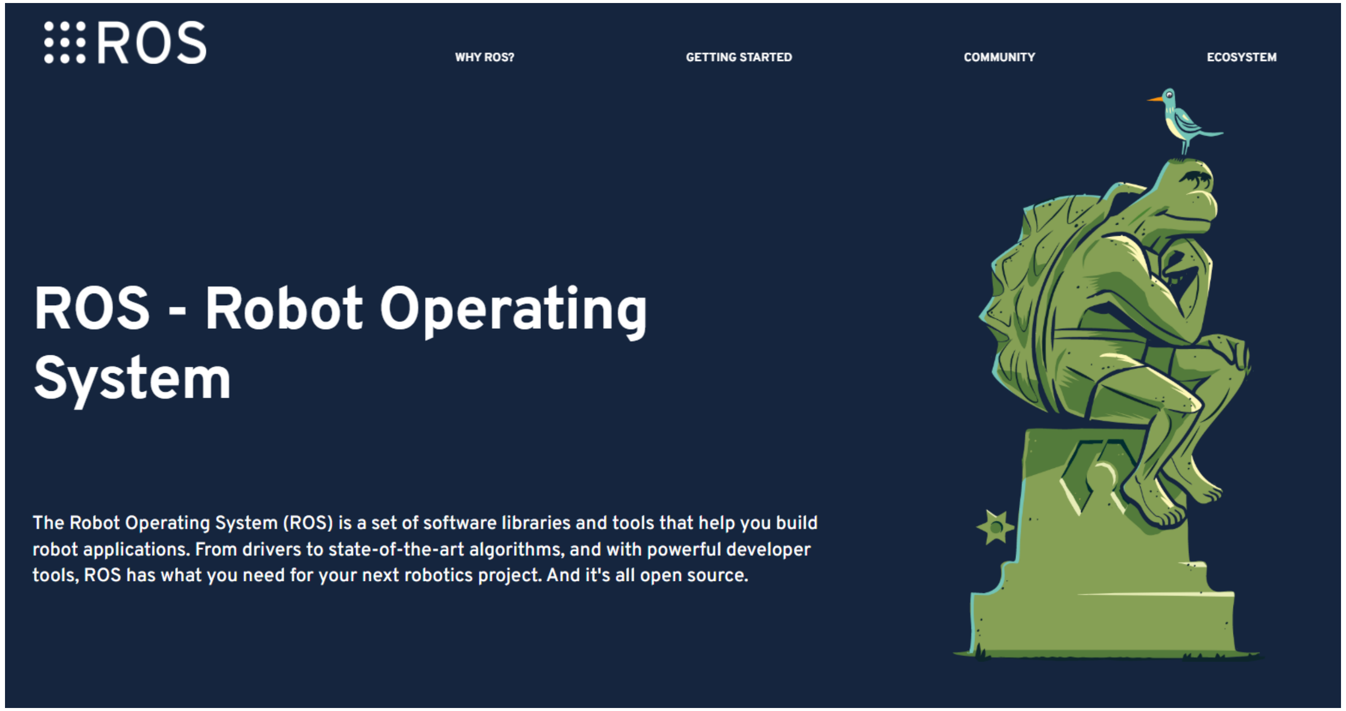
Cyber-physical discrepancy

Introduction

A Compilation of Robots Falling Down at the DARPA Robotics Challenge,
https://www.youtube.com/watch?v=g0TaYhjpOfo

Robot Operating System (ROS)

- ROS is a set of software libraries for robotics applications.



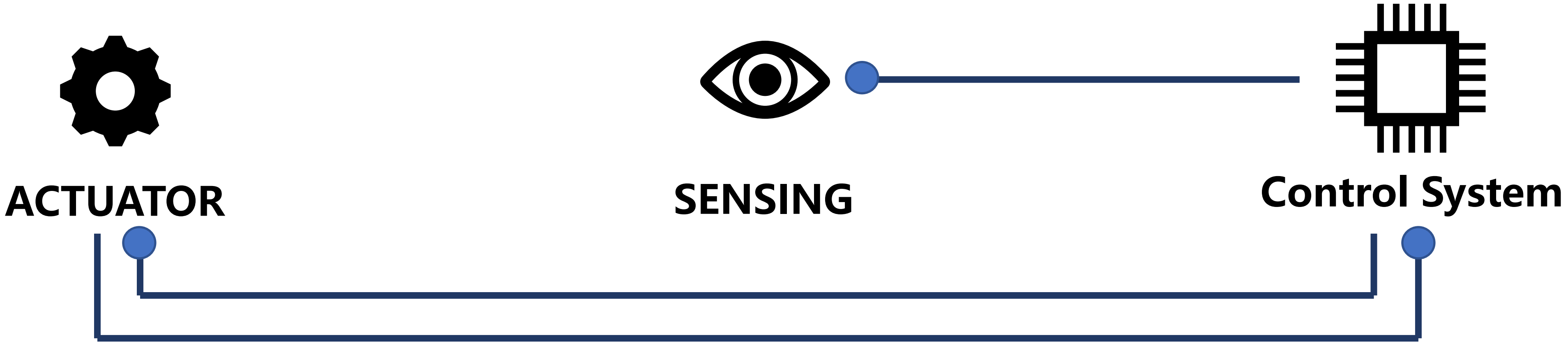
ROS - Robot Operating System

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

https://www.ros.org

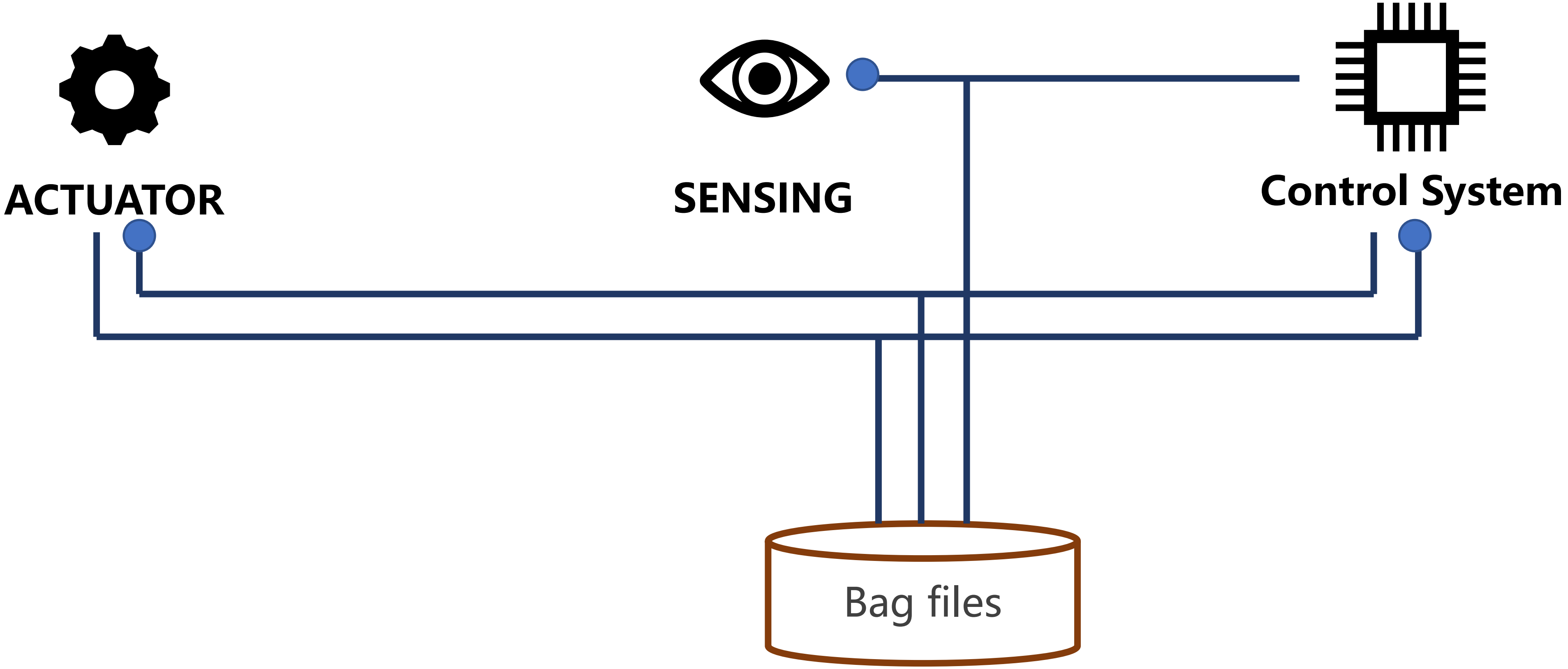
Robot Operating System (ROS)

- ROS simplifies robot development
- ROS facilitates communication between components using messages and topics



Robot Operating System (ROS)

- Each topic and message can be recorded as bag files or logs
- This enables easy testing, training, and validation.



Challenges of testing robotic systems

C1. Robotic systems are heterogeneous

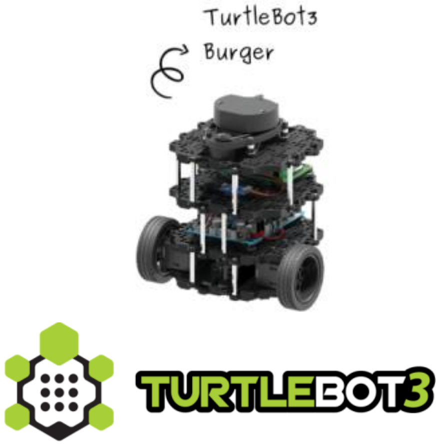
C2. Input space is humongous

C3. Physical processes are noisy

Challenge 1. Heterogeneity

- Enormous diversity
  - Drones, factory robots, surgical robots, autonomous cars, …
- Behavioral variations
  - The same software operating on different hardware
  - The same robot functioning in diverse environments

➔ One methodology may not fit all robots



https://docs.px4.io/v1.14/en/complete_vehicles/px4_vision_kit.html
https://www.turtlebot.com/turtlebot3/
https://picknik.ai/moveit/ros/2019/05/31/moveit2-alpha-release.html
https://wiki.ros.org/turtlesim

Challenge 1. Heterogeneity

- Enormous diversity
  - Drones, factory robots, surgical robots, autonomous cars, …
- Behavioral variations
  - The same software operating on different hardware

Solution: Focus on integral property of robotic systems

MoveIt2

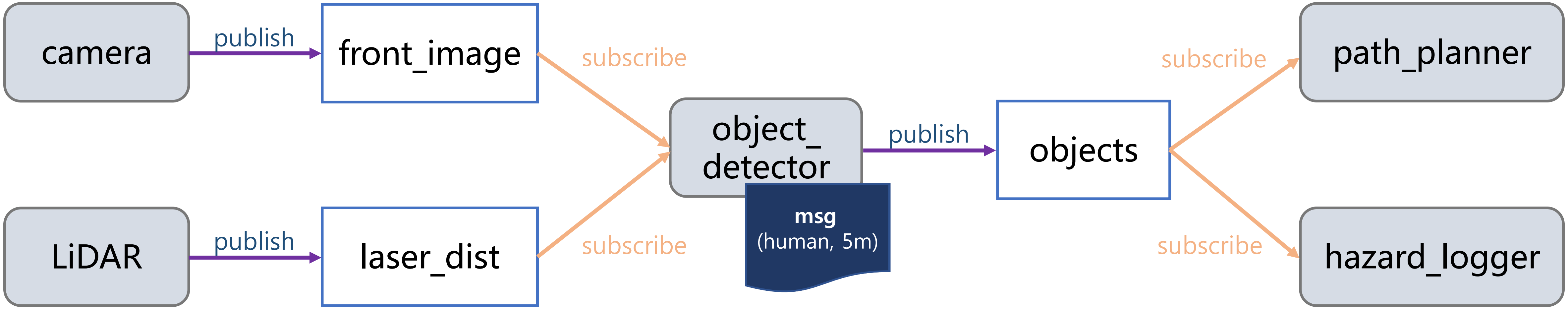
MoveIt 2 alpha

TURTLEBOT3

https://docs.px4.io/v1.14/en/complete_vehicles/px4_vision_kit.html
https://www.turtlebot.com/turtlebot3/
https://picknik.ai/moveit/ros/2019/05/31/moveit2-alpha-release.html
https://wiki.ros.org/turtlesim

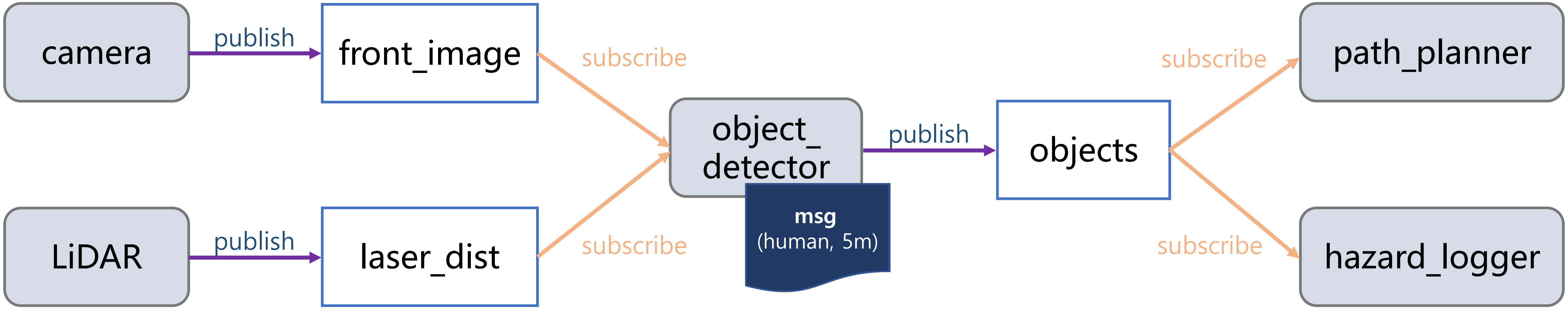
Robot development using ROS

- ROS-based robotic application: node + topic + msg
- Robotic behaviors can be represented by the data flow



Robot development using ROS

- ROS-based robotic application: node + topic + msg
- Robotic behaviors can be represented by the data flow



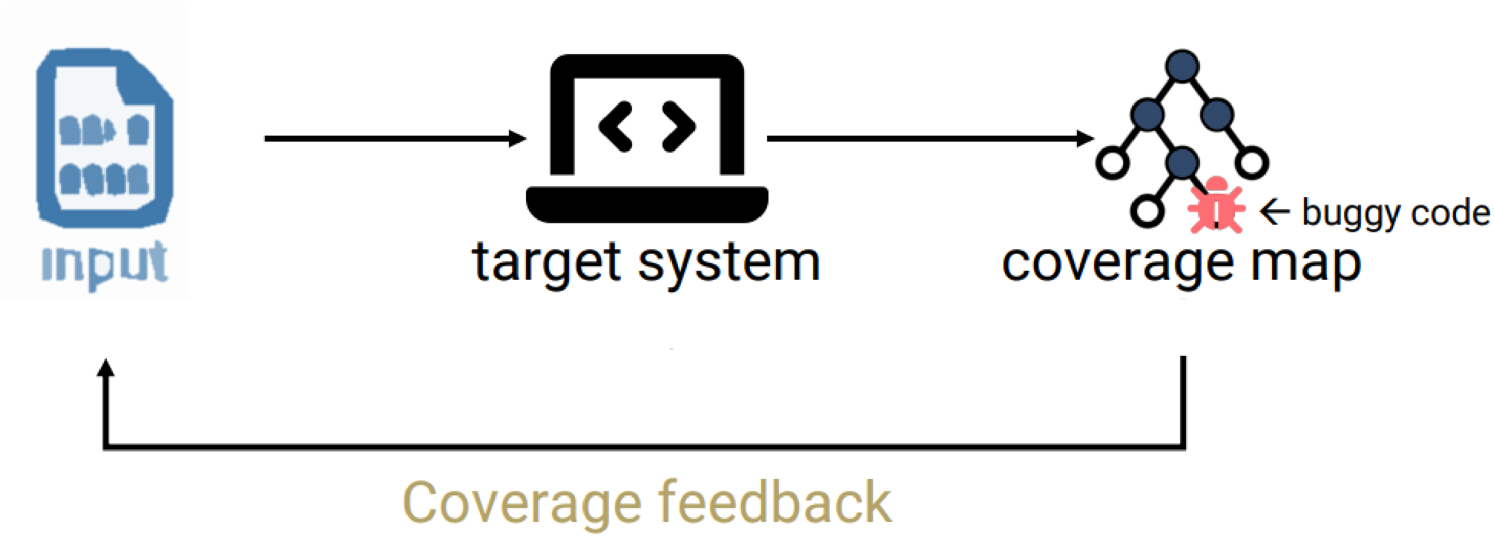
Challenge 2. Huge Input Space

- Robots operate in diverse conditions and environments
- Need to efficiently explore the search space

Solution: Feedback-driven fuzzing to the rescue

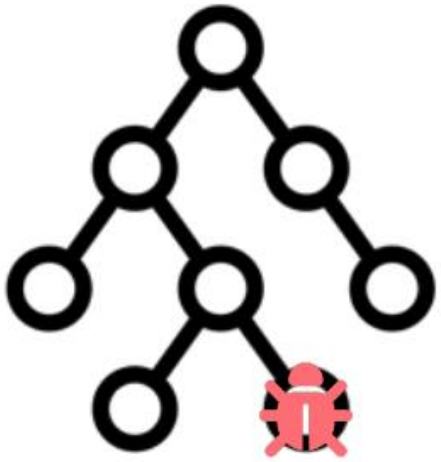
Feedback-driven fuzzing

- Proven in general software applications
- Effective for exploring large spaces

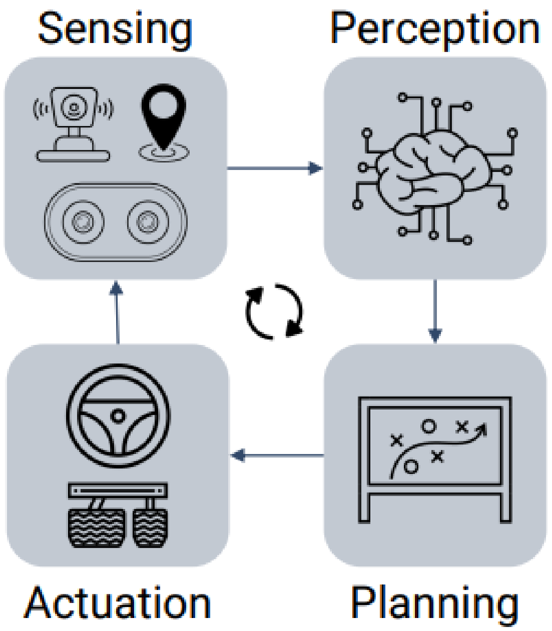


A need for a new feedback mechanism

General software programs



Robotic systems



- Diverse, linear code paths
- More code paths ≃ more bugs found

- Robotic system is distributed system
- Behavior is driven by state changes in a loop, not by code paths

Fundamental questions

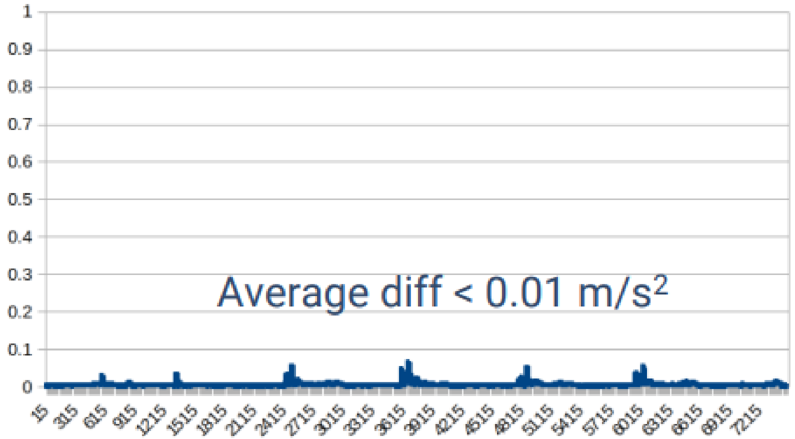
- How do we determine if the robotic system is approaching an undesirable state?
- What indicates that the robotic system is being driven towards buggy states?

Semantic feedback for robotic systems

- Runtime signs may indicate the robotic system transitioning into buggy states.
- E.g., Redundant Sensor Inconsistency Feedback
  - Pixhawk 4 has two Inertial Measurement Units (IMU)

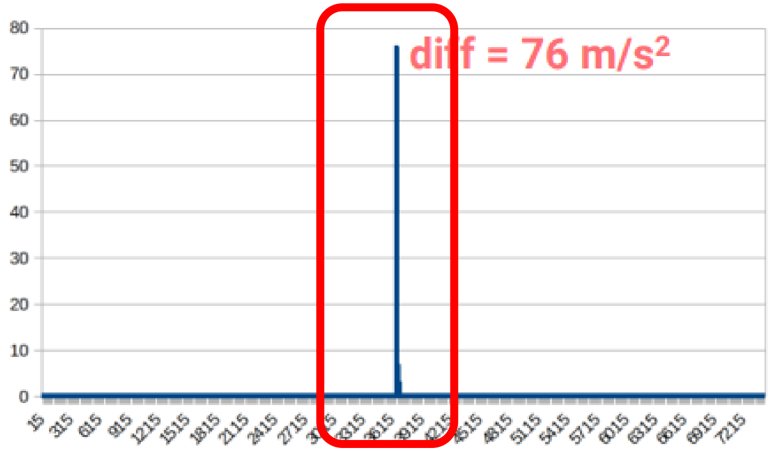
ICM-20689 of TDK

BMI-055 of Bosch



Average diff < 0.01 m/s^2

Diff. of measured acceleration (stable operation)



diff = 76 m/s^2

Diff. of measured acceleration (crashed mission)

https://invensense.tdk.com/products/motion-tracking/6-axis/icm-20689/
https://www.mouser.kr/ProductDetail/Bosch-Sensortec/BMI055?qs=d72FGnIDsgRto2b9Ru3roQ%3D%3D

Challenge 3. Noisy Hardware

- Hardware components interact with real world
  - Sensors and actuators inherently noisy
    - E.g., GPS reports changing values even when stationary

- It is impossible to perfectly model the physical world

Challenge 3. Noisy Hardware

- Hardware components interact with real world
  - Sensors and actuators inherently noisy
    - E.g., GPS reports changing values even when stationary

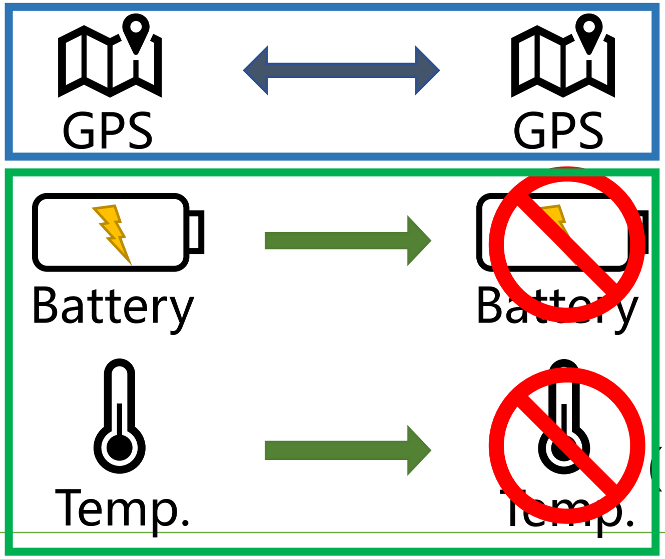
- It is impossible to perfectly model the physical world

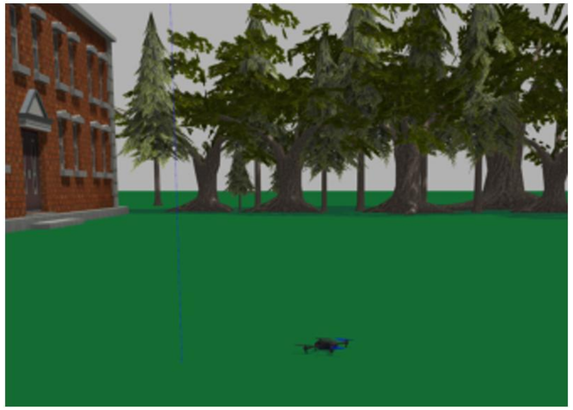
Solution: Simultaneously exexcute in the real world and a simulator

Simultaneously executing a robotic system

- Filling missing states
  - Some states exist only in the physical world
  - E.g., battery consumption, motor temperature
- Detecting divergent States
  - Some states diverge, which is an important execution feedback
  - E.g., location (GPS)

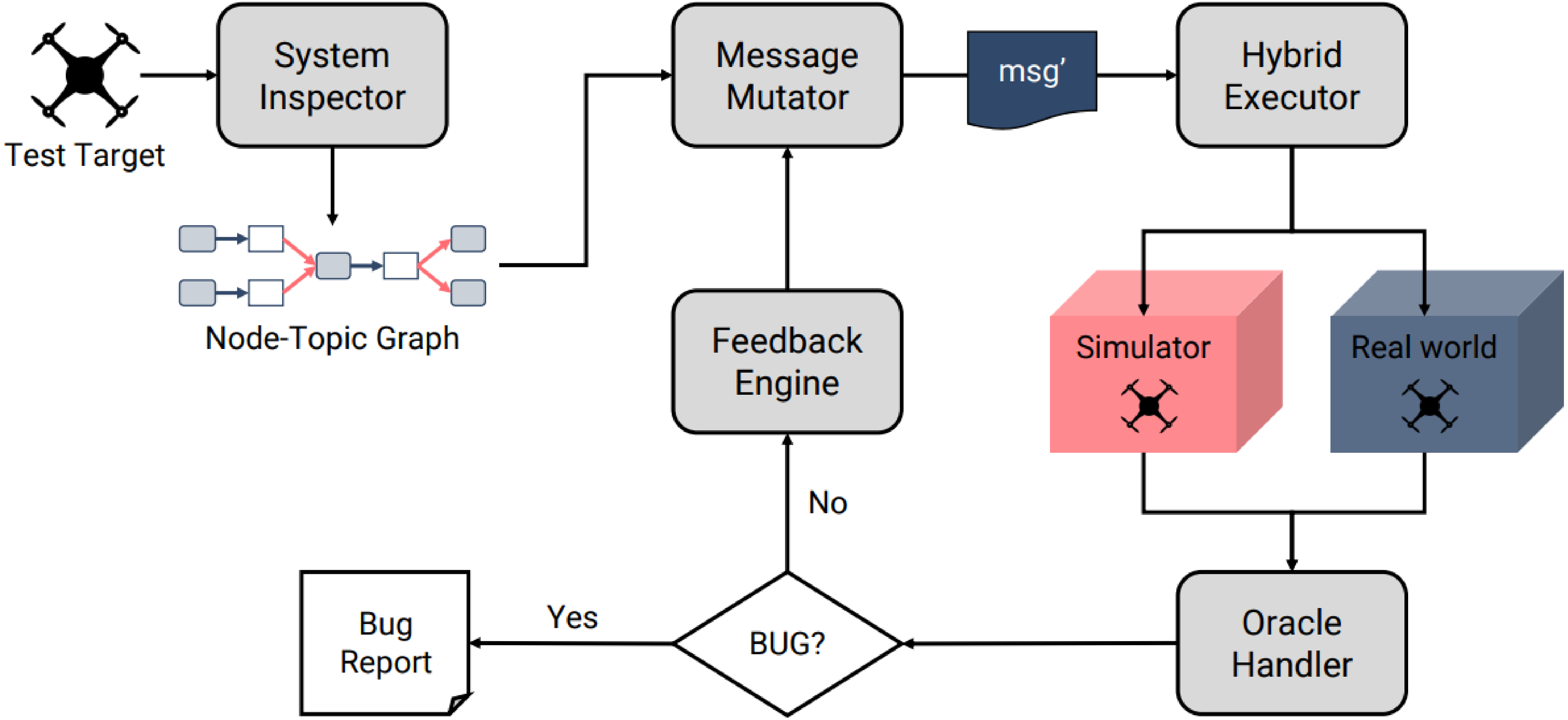
(a) PX4 drone in the real world



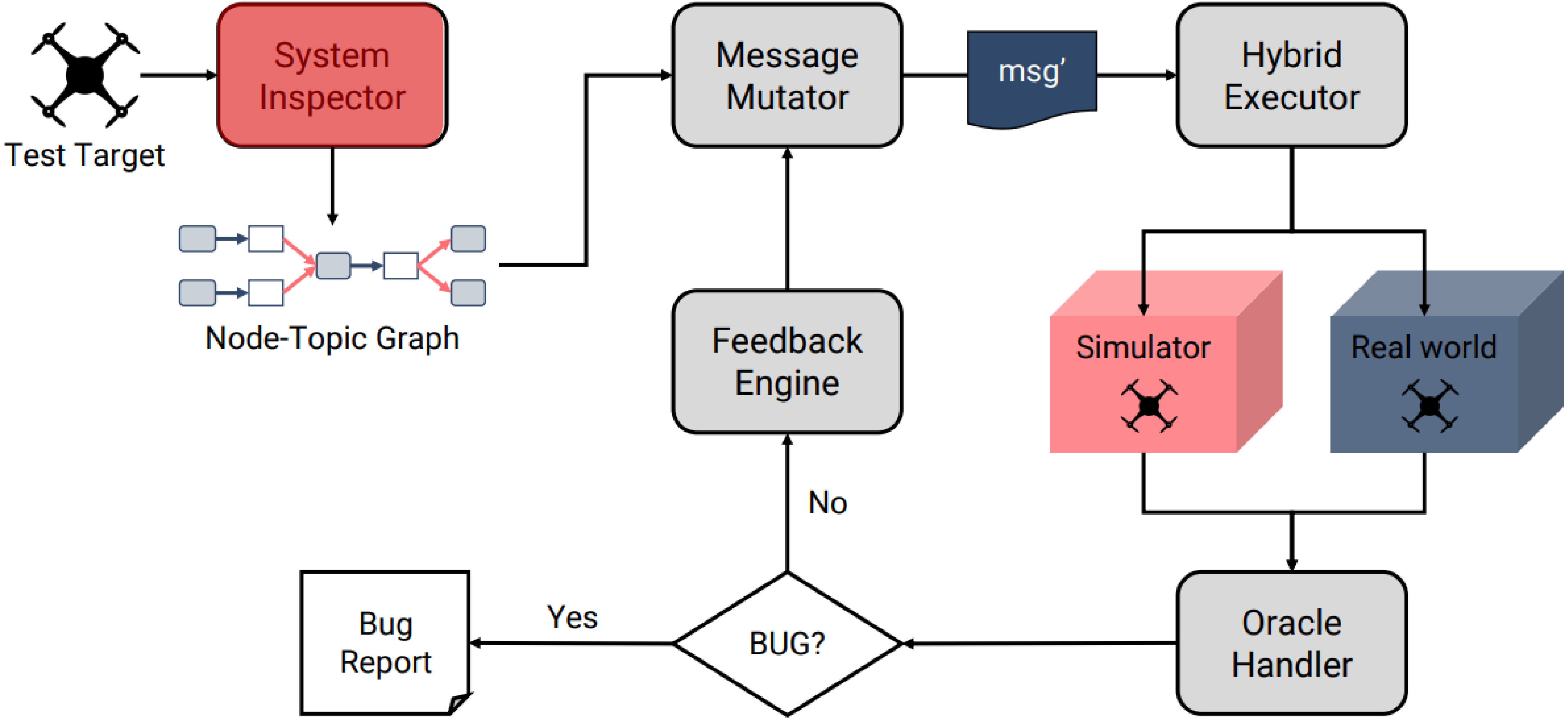


(b) PX4 drone in Gazebo simulator

Overview of RoboFuzz

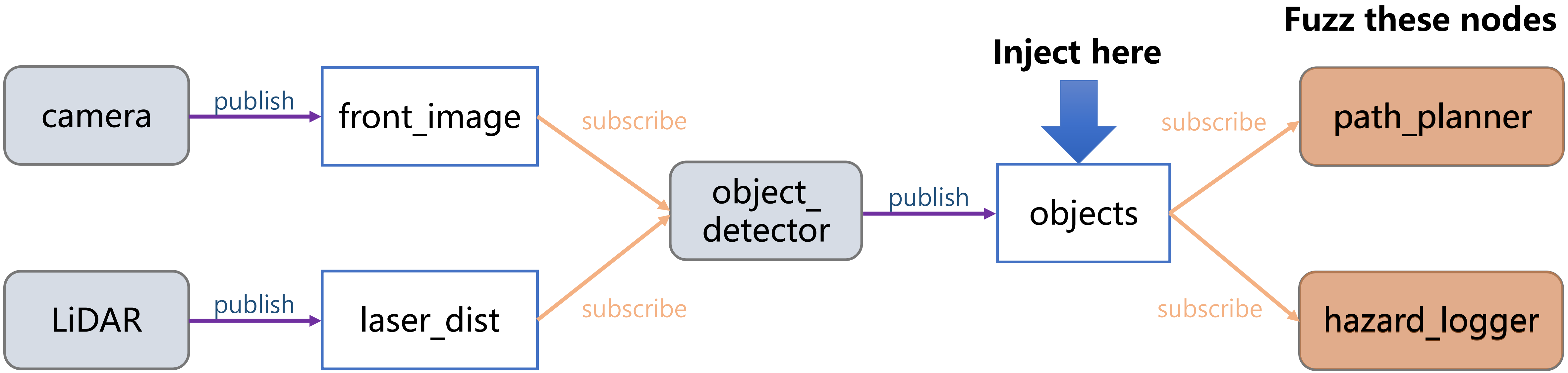


System Inspector



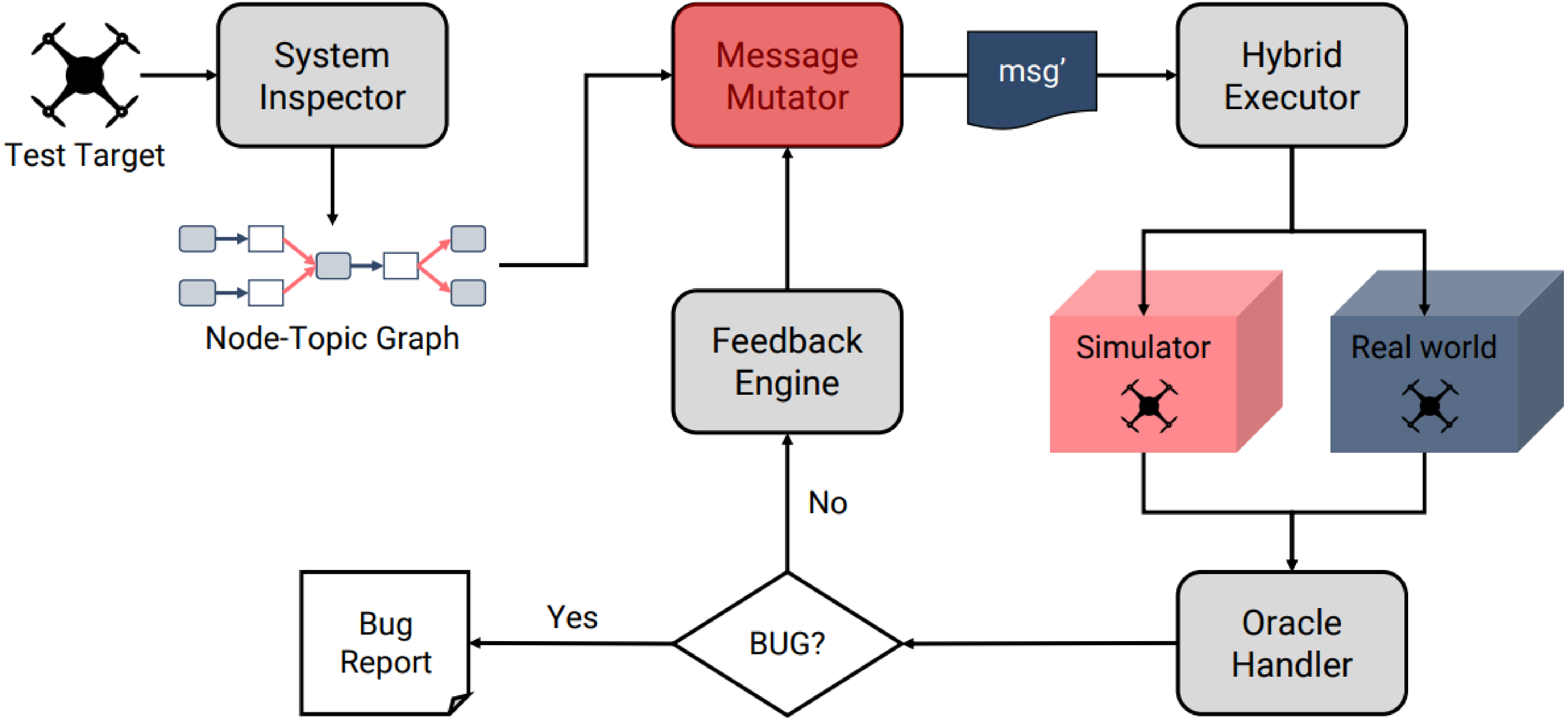
System Inspector

- Generates a node-topic graph
  - Select a topic to inject mutated messages



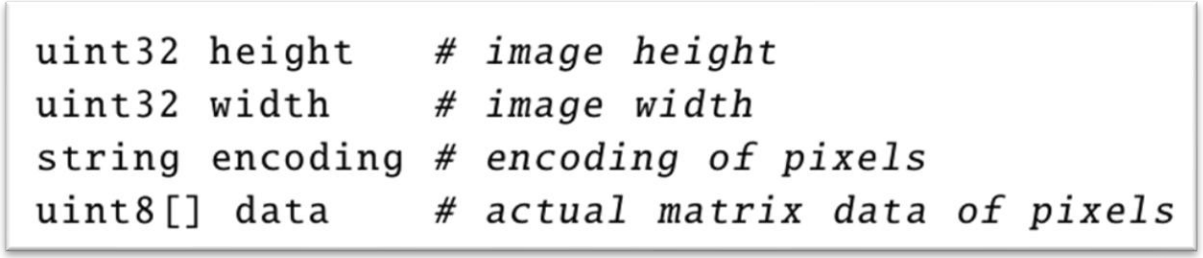
node-topic graph

Message mutator

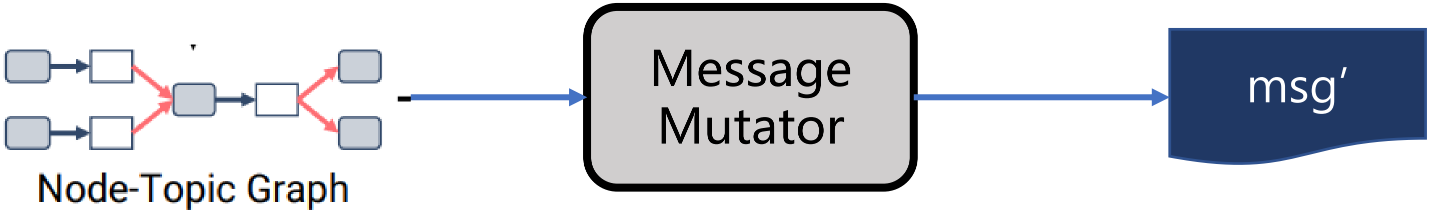


Message mutator

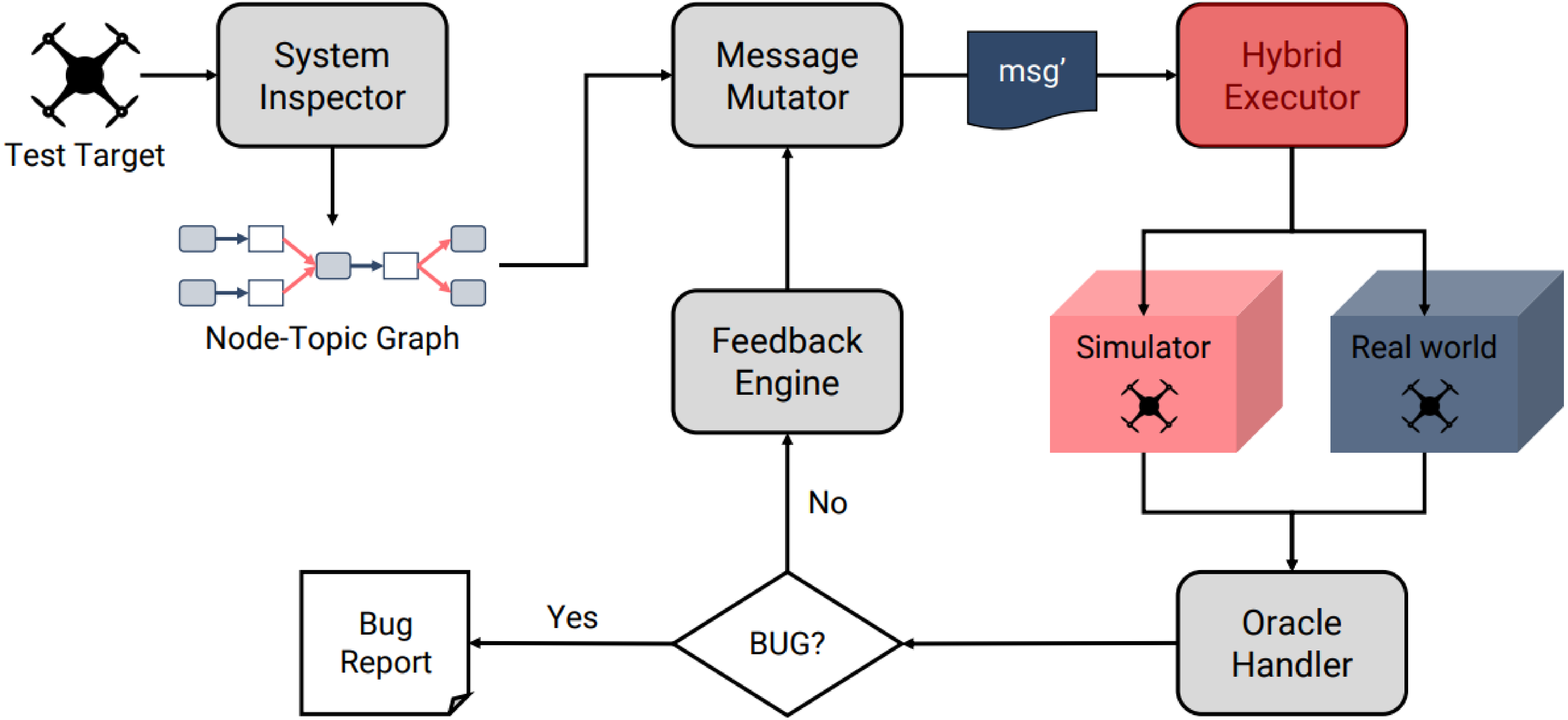
- Structure-aware mutation
  - ROS messages are structured
  - E.g., message definition for image data.



```
uint32 height    # image height
uint32 width     # image width
string encoding # encoding of pixels
uint8[] data     # actual matrix data of pixels
```

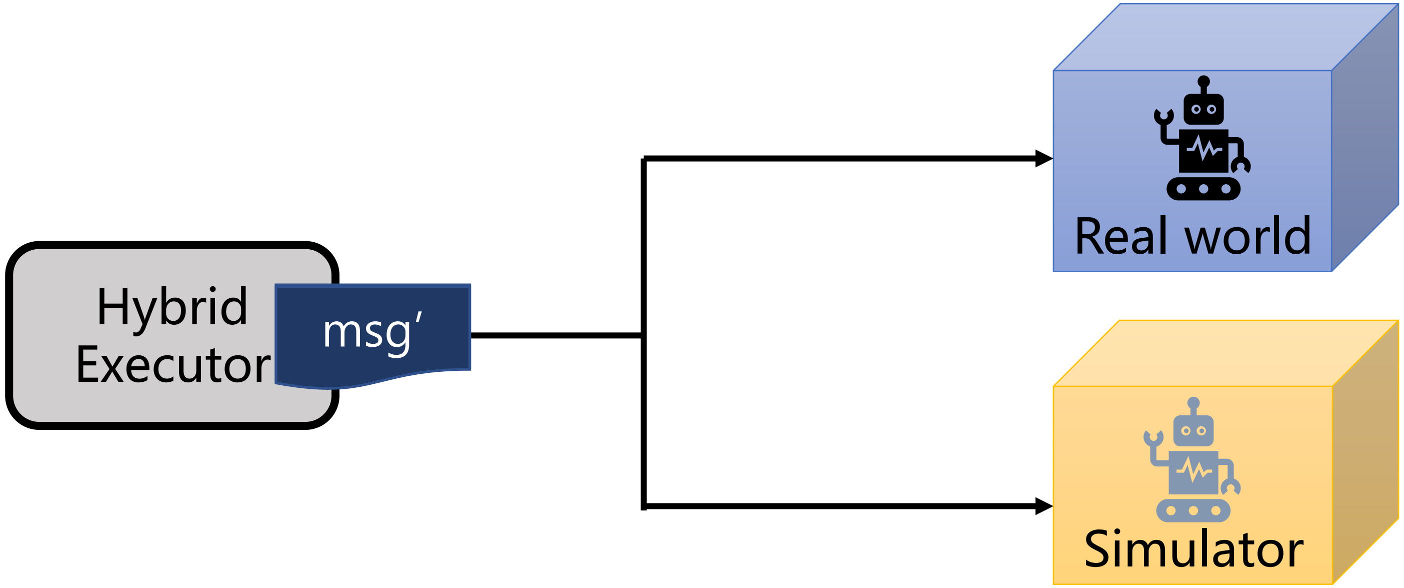


Hybrid executor

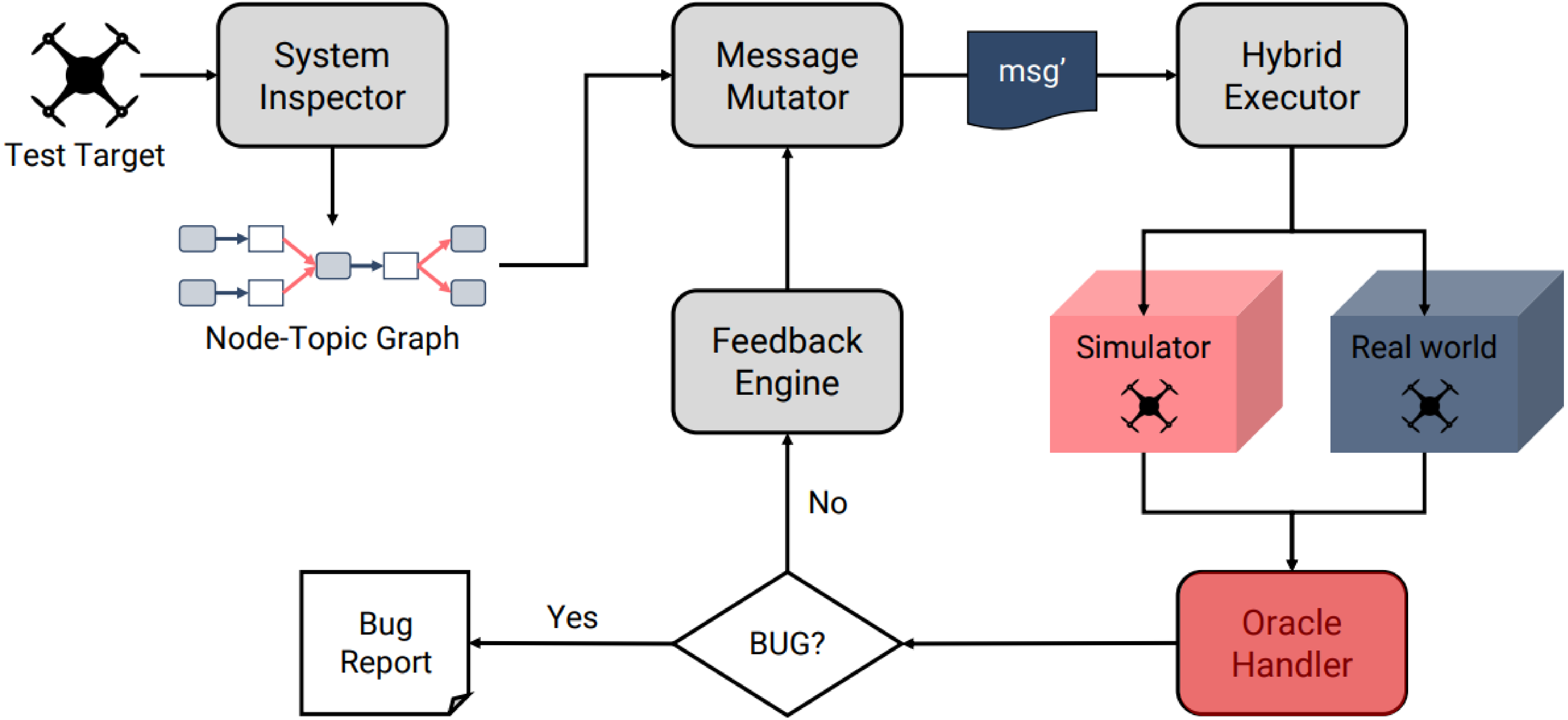


Hybrid executor

- Set up a pair of simulated and physical test beds
  - Robots subscribe to the same topic
  - Publish mutated messages to the topic
  - Both robots receive the message and take corresponding actions

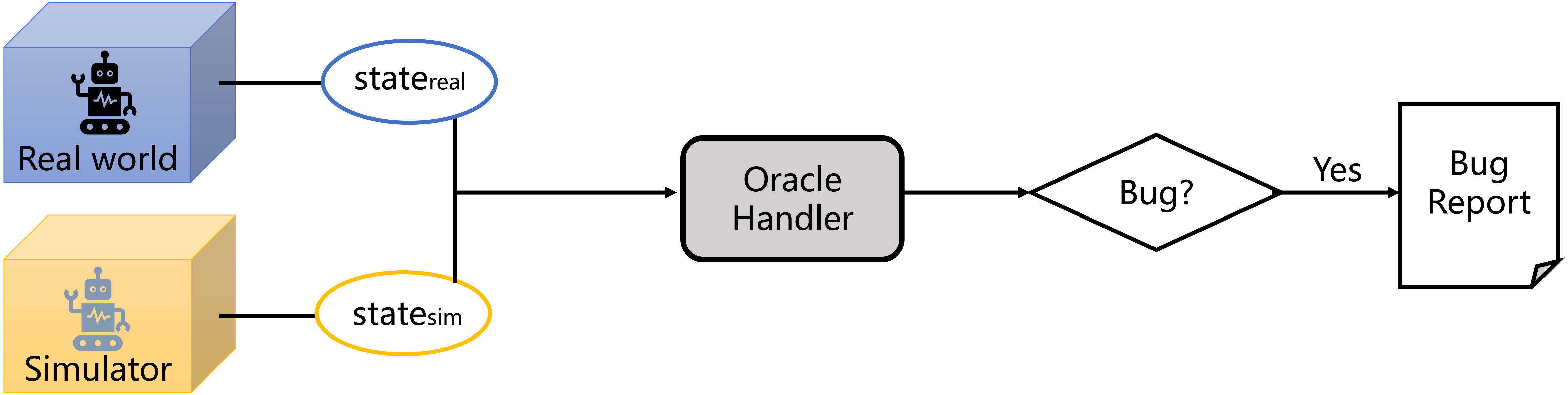


Oracle handler

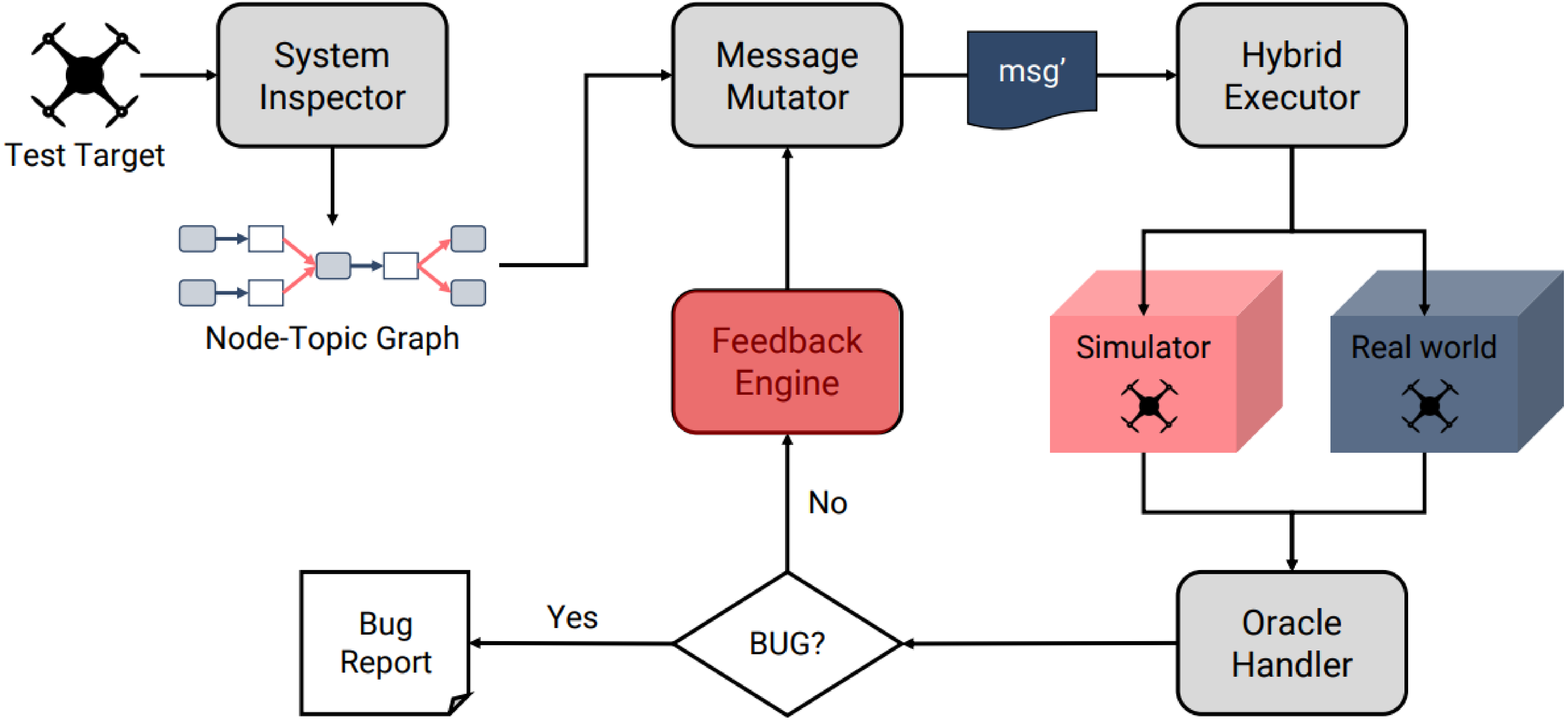


Oracle handler

- Collects and merges states from hybrid execution
- Reports if any violation is found

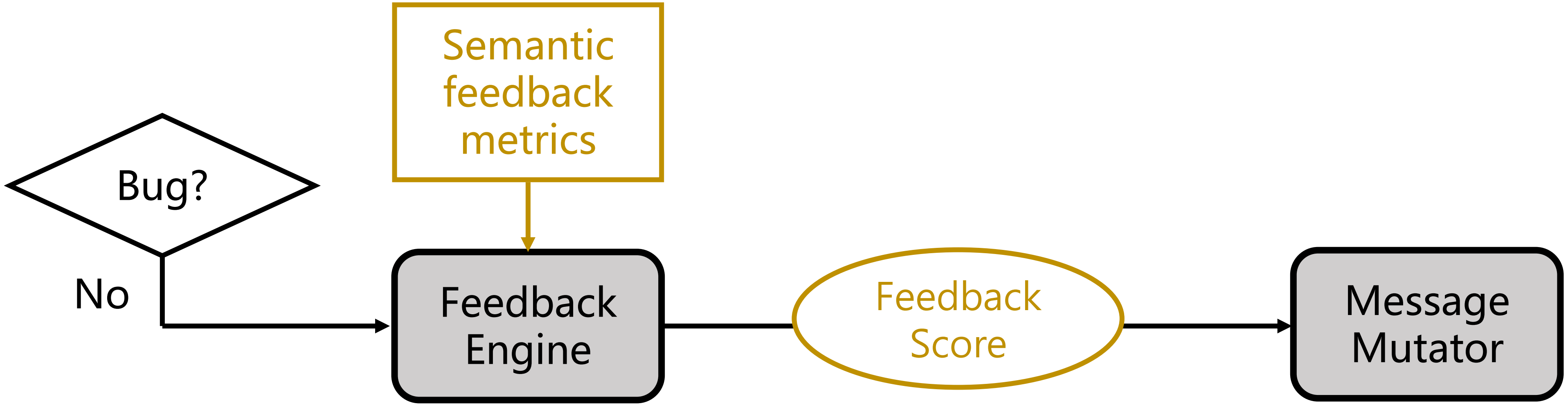


Feedback engine



Feedback engine

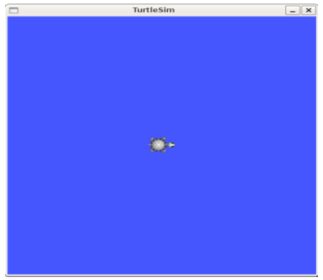
- If no bug is found, calculates the feedback score
  - Using the semantic feedback metrics
    - e.g., redundant sensor inconsistency
  - Users can register custom feedback metrics
- Favorable inputs are enqueued
  - Further mutated in the subsequent fuzzing rounds



Evaluation

- **Environment**
  - Laptop machine running Ubuntu 20.04
  - Intel i7-8850H 2.6Ghz, 16GB RAM, Quadro P2000 Mobile GPU

- **Six fuzzing targets**
  - ROS 2-based robots
    - PX4
    - TurtleBot3
    - MoveIt2
    - Turtlesim
  - ROS 2 Internals
    - Type system (ROSIDL)
    - Client library (rclpy/rclcpp)

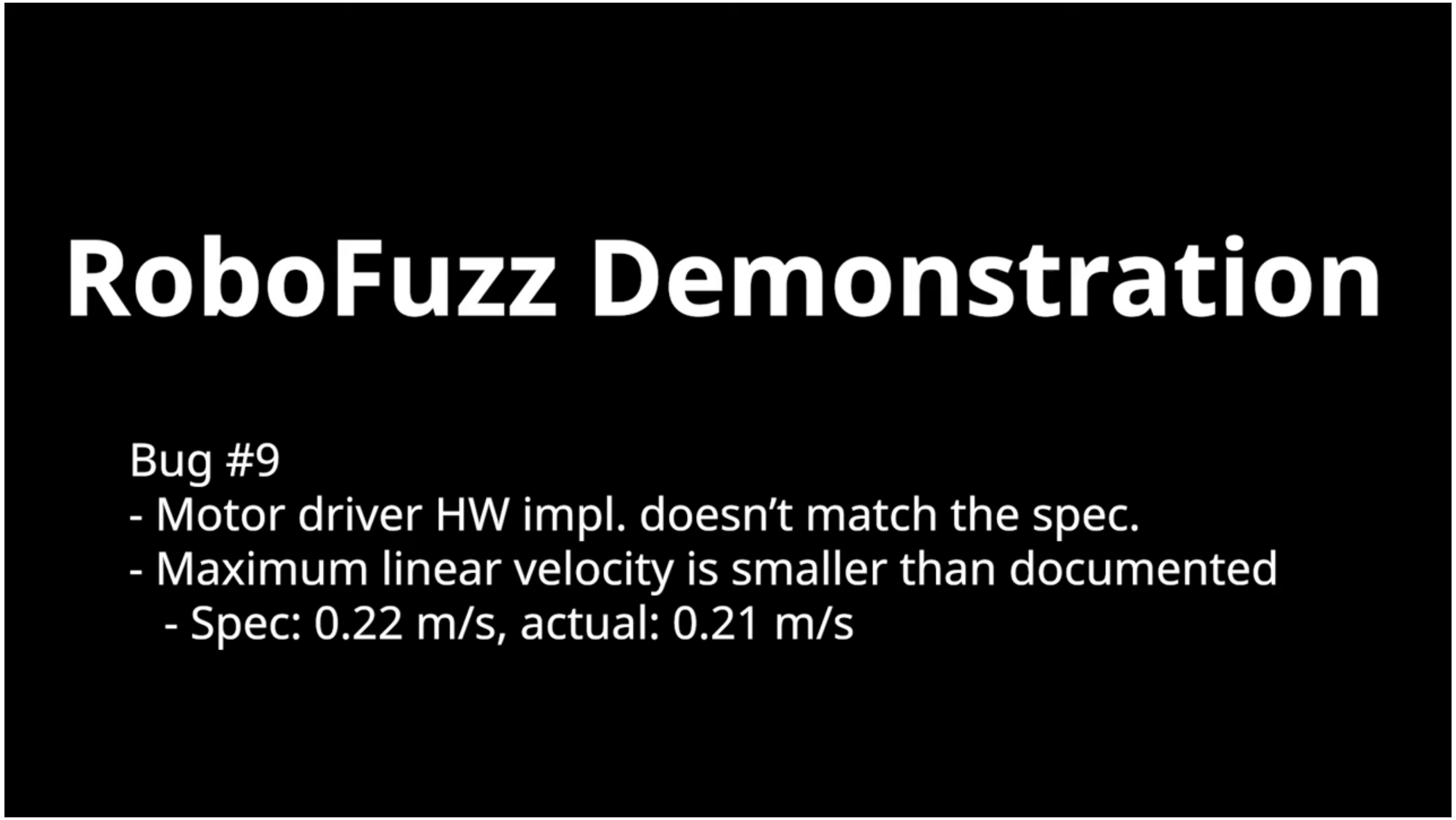


Turtlesim

Overall effectiveness of RoboFuzz

- Found 30 new correctness bugs (25 acknowledged, 6 fixed)
  - ROS 2 Internal layers
    - **8** in ROSIDL
    - **5** in rclpy/rclcpp
  - Applications
    - **8** in PX4 drone
    - **5** in TurtleBot3
    - **2** in MoveIt2
    - **2** in Turtlesim

Demo - TurtleBot3 spec. violation



RoboFuzz Demonstration

Bug #9
- Motor driver HW impl. doesn’t match the spec.
- Maximum linear velocity is smaller than documented
  - Spec: 0.22 m/s, actual: 0.21 m/s

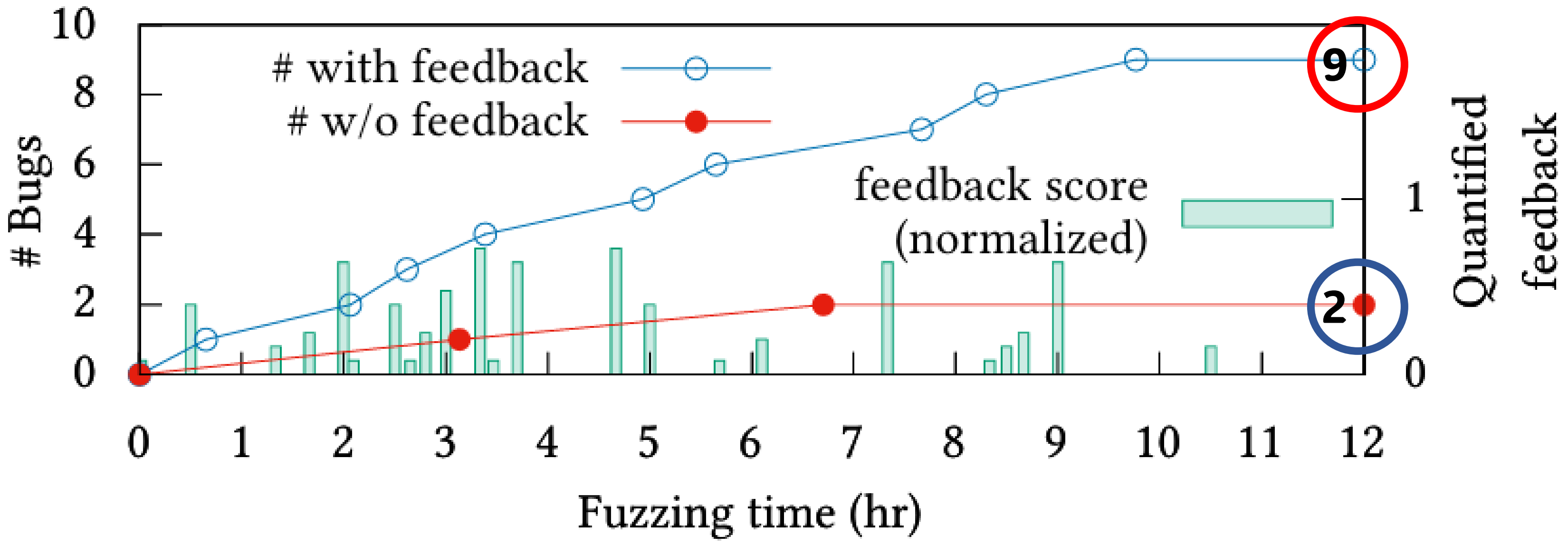
- Bug

Achievable velocity is smaller than documented due to a float handling bug in motor driver

Spec : 0.22 m/s
Actual: **0.21 m/s**

Effectiveness of semantic feedback

- Fuzzing PX4 with and without semantic feedback for 12 hr.
  - 9 bugs **with** feedback
  - 2 bugs **without** feedback



Related works

- PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles
  - H. Kim et al., NDSS, 2021
- DriveFuzz: Discovering Autonomous Driving Bugs through Driving Quality-Guided Fuzzing
  - S. Kim et al., CCS, 2022
- ROCAS: Root Cause Analysis of Autonomous Driving Accidents via Cyber-Physical Co-mutation
  - S. Feng et al., ASE 2024
- Enhancing ROS System Fuzzing through Callback Tracing
  - Y. Shen et al., ISSTA, 2024

Summary

- Targeted correctness bugs in **ROS** and **ROS-based robots**
- Semantic feedbacks are defined and registered to efficiently explore the input space
- Utilized hybrid execution model to collect and compare the states of both cyber and physical robots
- Found 30 new correctness bugs in multiple robotic systems

Thank you

Good questions

- This framework could easily be adapted to ROS 1 or would significant changes need to be made?
- Is RoboFuzz better understood as a quality assurance tool?
- Is the communication between each node in the ROS system secure?
- Is there any way to make this more efficient and accessible for lower budget environments?
- How to address the challenge of oracle generation?

Great Questions

- YoungHyo Kang: The paper focuses on quickly finding bugs through the fuzzer. However, from a developer's perspective, I believe that finding as many bugs as possible, as long as it is not infeasible, would be beneficial. From that perspective, while **the code coverage** measure was inefficient within 12 hours than semantic feedback method, **I think it could ultimately help discover more bugs**. What are your thoughts on this? Additionally, which do you think is more important: time or finding more bugs?

- Pierre Noyer: How well do you think RoboFuzz would **scale** when testing more complex robotic systems, such as ones with many more degrees of freedom or that operate in more complex environments?

- Donghyo Bang: How does RoboFuzz handle **real-time** systems and ensure timing accuracy in fuzzing **time-sensitive** processes?