ROS2 퍼징을 위한 피드백 메커니즘 비교 연구

강지원[1], 오현영[2*]
[1]가천대학교 인공지능학과 학부생
[2*]가천대학교 인공지능학과 교수
{gjwon123, hyoh}@gachon.ac.kr

Comparative Study of Feedback Mechanisms for ROS2 Fuzzing

Giwon Kang[1], Hyunyoung Oh[2]
Dept. of AI, Gachon University

요 약

퍼징(fuzzing)은 자동으로 입력을 생성·투입해 프로그램의 예외나 취약 동작을 찾아내는 검증 기법으로, 전통적으로는 코드 커버리지를 피드백으로 활용한 그레이박스 퍼징이 주류를 이루어 왔다. 본 논문은 ROS2 환경을 대상으로 제안된 여러 퍼저(fuzzer)의 피드백 메커니즘을 정리하고 비교한다. 코드 커버리지, 물리·시맨틱 기반 오라클, 메시지 특성, 콜백 및 라이프사이클 기반 피드백 등 다양한 접근법을 검토하고, 각 방식의 장단점과 한계를 논의한 후, 이를 개선한 새로운 퍼징 도구 개발의 필요성에 대해 논의한다.

1. 서론

ROS는 로봇 응용의 표준 플랫폼이며, ROS2는 실시간성, 보안, 다중 로봇 지원을 강화해 산업과 연구에서 널리 활용되고 있다[1]. 시스템 복잡도가 커질수록 예기치 않은 입력에 대한 강건성 검증이 중요해지고, 퍼징은 자동화된 검증 기법으로 각광받고 있다[2]. 특히 피드백 기반 퍼징(feedback-guided fuzzing)은 입력 실행 결과를 피드백 신호로 활용하여 테스트 효율을 높이는 대표적 방식이다[3]. 그러나 ROS2는 다중 노드, 비동기 콜백, 분산 통신 구조를 갖고 있어, 단순 커버리지 피드백만으로는 특정 유형의 결함을 찾기 어렵다. 본 논문은 기존 ROS2 퍼저 연구에서 제시된 피드백 메커니즘을 비교·정리하고, 향후 연구 과제를 도출하고자 한다.

2. 배경지식

2.1. ROS2 아키텍처

ROS2는 노드 간 통신을 토픽(Topic), 서비스(Service), 액션(Action)으로 처리하며, Executor가 콜백 큐를 관리한다. 분산·비결정적 실행 특성 때문에 검증 과정에서 재현성이 떨어지고, 다양한 관점에서의 피드백 설계가 필요하다[1].

2.2. 피드백 기반 퍼징

피드백 기반 퍼징은 코드 커버리지, 크래시, 상태 변화 등 실행 신호를 활용하여 입력 변이를 가이드한다. AFL과 같은 그레이박스 퍼저는 분기 커버리지를 기준으로 새로운 경로를 탐색하고, AddressSanitizer 등 도구와 결합해 메모리 오류를 탐지한다[3],[5]. ROS2 환경에서는 커버리지 외에도 통신, 상태, 스케줄과 같은 다양한 신호가 고려된다.

3. 사례 분석

3.1. 코드 커버리지 기반

Ros2-fuzz는 특정 노드에 메시지를 주입하여 분기 커버리지를 측정한다[4]. ROZZ는 여러 노드의 커버리지를 통합해 '분산 분기 커버리지'를 적용한다[6]. 커버리지는 구현이 간단하고 일반성이 높지만, 교차 노드 상호작용이나 동시성 오류를 잡기 어렵다.

3.2. 물리, 시맨틱/상태 기반

Phys-Fuzz는 물리적 위험 점수를 계산하고[7], RoboFuzz는 사전 정의된 상태 불변식을 위반하는지를 피드백으로 삼는다[8]. CPS 안전 검증에 유용하

* 교신저자

퍼저 (Fuzzer)	Key Feedback	목적	한계
Ros2-fuzz[4] / ROZZ[6]	분기 커버리지	단일, 다중 노드 코드 경로	교차 상호작용, 동시성 민감성 낮음
Phys-Fuzz[7]	물리적 위험 점수	사이버-물리 시스템 안전성	오라클 설계 비용 높음
RoboFuzz[8]	시맨틱/상태	로봇 동작 불변속성	도메인 종속
ROFER[9]	메시지 특징	분산 통신 상호작용	의미 오류 판정 한계
R2D2[10]	콜백 순서	비결정적/비동기 콜백 실행	재현성, 오버헤드 문제
ROCF[11]	라이프사이클 전이	동시성 버그, 메모리 오류	구현 비용 높음

(표 1) Feedback에 따른 퍼저 비교 정리

지만, 오라클 설계 비용과 도메인 종속성이 크다는 한계가 있다.

3.3. 메시지 기반

ROFER는 메시지 교환의 순서·빈도·값 변화를 피드백으로 사용해 다중 노드 통신 상호작용을 탐색한다[9]. 분산 통신을 직접 자극할 수 있으나, 의미적 결함 판정은 별도의 오라클이 필요하다.

3.4. 콜백·라이프사이클 기반

ROS2 Executor는 콜백을 비결정 순서로 실행해 희소한 경쟁 조건을 발생시킨다. R2D2는 콜백 실행 순서를 추적하고 새로운 순서를 찾는 입력을 우선시하며[10], ROCF는 라이프사이클 전이 순서를 피드백으로 삼아 전이 시점의 오류를 탐지한다[11]. 동시성 버그 탐지에 강점이 있지만, 재현성과 오버헤드 관리가 과제로 남는다.

4. 비교 정리

표1은 주요 퍼저와 피드백의 차이를 정리한 것이다.

5. 결론 및 향후 연구

이본 논문은 ROS2 환경에서 제안된 여러 퍼저의 피드백 메커니즘을 비교·분석했다. 커버리지는 가장 널리 쓰이고 효율적이지만 분산·동시성 결함에는 한계가 있으며, 물리/시맨틱 오라클은 의미적 오류 탐지에 유용하나 비용이 크다. 메시지, 콜백, 라이프사이클 기반 피드백은 ROS2 특성에 더 밀착되어 있으나 재현성과 구현 복잡성이 과제다.

향후 연구에서는 여러 피드백을 조합하는 하이브리드 방식이나, 재현성을 보장할 수 있는 스케줄링 제어 기법이 필요하다. 또한 오라클 설계 비용을 줄이기 위한 자동화·표준화 기법 연구[12-13]가 고도화될 필요가 있다. 이러한 방향은 ROS2 퍼징을 보다 효율적이고 실용적으로 만드는 데 기여할 것이다.

사사문구

이 논문은 2025년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. RS-2024-00337414, SW공급망 운영 환경에서 역공학 한계를 넘어서는 자동화된 마이크로 보안 패치 기술 개발)과 한국산업기술기획평가원의 지원(No. RS-2024-00406121, 자동차보안취약점기반위협분석시스템개발(R&D))을 받아 수행된 연구 결과임.

참고문헌

[1] S. Macenski, et al., "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, no. 66, 2022.

[2] P. M. Sutton, et al., "Fuzzing: Brute force vulnerability discovery," in Proc. USENIX Security, 2007.

[3] M. Zalewski, AFL Documentation Release 2.53b, 2019, https://afl-1.readthedocs.io/_/downloads/en/latest/pdf

[4] I. Lütkebohle, "ros2_fuzz, a simple ROS 2 node fuzzer," GitHub, 2020, https://github.com/iluet/ros2_fuzz

[5] K. Serebryany, et al., "AddressSanitizer: a fast address sanity checker," in Proc. ATC, 2012.

[6] J. Choi, et al., "ROZZ: A Fuzzer for ROS," in Proc. ICRA, 2021.

[7] T. Woodlief, et al., "Fuzzing Mobile Robot Environments for Fast Automated Crash Detection," in Proc. ICRA, 2021.

[8] S. Kim and T. Kim, "RoboFuzz: Fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs," in Proc. ESEC/FSE, 2022.

[9] J.-J. Bai, et al., "Multi-Dimensional and Message-Guided Fuzzing for Robotic Programs in Robot Operating System," in Proc. ASPLOS, 2024.

[10] Y. Shen, et al., "Enhancing ROS System Fuzzing through Callback Tracing," in Proc. ISSTA, 2024.

[11] Si-Miao Gao, et al., "ROCF: Detecting Lifecycle-Related Concurrency Bugs in ROS Programs via Coverage-Guided Fuzzing," IEEE Trans. Inf. Forensics Security, 2025.

[12] E. T. Barr, et al., "The oracle problem in software testing: A survey," IEEE Trans. Softw. Eng., vol. 41, no. 5, 2015.

[13] C. Lemieux, et al., "Do LLMs Generate Test Oracles That Capture the Actual or the Expected Program Behaviour?," in Proc. ICSE, 2024.