O-RAN 테스트 환경 구성을 위한 동시성 기반 SD-RAN 시뮬레이터 설계 및 활용

권혁선, 유현민, 최호성, 김우중*, 홍인기

경희대학교, *마이크로소프트 코퍼레이션

gurtjs0116@khu.ac.kr, yhm1620@khu.ac.kr, to6044@khu.ac.kr, *woojoongkim@microsoft.com, ekhong@khu.ac.kr

Design and Application of a Concurrency-Based SD-RAN Simulator for O-RAN Test Environment

Hyuksun Kwon, Hyunmin Yoo, Hoseong Choi, Woojoong Kim*, Een Kee Hong

Kyunghee University, *Microsoft Corporation

요 약

본 논문은 클라우드 네이티브 전환이 가속화되고 있는 무선 액세스 네트워크 분야에서, O-RAN 표준과 Go 언어의 경량 동시성 모델을 결합한 Software-Defined RAN(SD-RAN) 시뮬레이터의 설계와 활용 방안을 제시한다. Go의 고루틴·채널·컨텍스트를 활용한 시뮬레이터의 아키텍처를 심층 분석하고, 표준 절차에 따라 near-RT RIC과의 연동을 검증한 결과, 시뮬레이터가 확장성(Scalability)·탄력성(Resilience)·관찰가능성(Observability)을 충족함을 확인하였다. 또한 Go 기반 xApp의 핵심 구성 요소와 동작 구조를 도출함으로써, 학계와 산업계가 클라우드 네이티브 형태의 O-RAN 테스트 환경을 구축하고 지능형 RAN 연구·개발을 가속화할 수 있는 실질적 방향성을 제시한다.

Ⅰ. 서 론

Open RAN(O-RAN)은 무선 접속망(RAN)의 구성 요소 간 인터페이스를 표준화하고 기능을 분리함으로써, 다양한 벤더 간 상호운용성과 유연한 네트워크 구성을 가능하게 한다. 이러한 구조적 변화는 Central Unit(CU)와 Distributed Unit(DU)를 포함한 RAN 기능을 소프트웨어화하고, 클라우드 인프라 상에 배치할 수 있는 기반을 제공한다. 이에 따라 클라우드 네이티브 기술이 RAN에 본격적으로 적용되며, RAN Intelligent Controller(RIC)를 활용한 네트워크 지능화에 대한 관심이 높아지고 있다 [1]. 하지만 실제 장비 기반의 검증은 높은 비용과 물리적인 제약이 따르기 때문에, RAN과 RIC을 모두 포함한 지능형 O-RAN 구조를 실험적으로 구현할 수 있는 테스트 환경과 경량화된 시뮬레이터의 필요성이 커지고 있다. 이에 본 논문에서는 오픈소스 O-RAN 프로젝트인 SD-RAN에서 제공하는 RAN 시뮬레이터 구조를 분석하며, 특히 Go 언어의 경량 동시성 모델을 활용한 클라우드 네이티브 환경에도 적합한 구현 방식에 주목한다. 또한, Near-realtime RIC(Near-RT RIC)과의 연동 및 xApp 개발 아키텍처를 분석함으로써, SD-RAN 시뮬레이터의 실질적인 개발 및 활용 방향을 제시한다.

Ⅱ. 본론

A. SD-RAN 프로젝트

SD-RAN은 Open Networking Foundation(ONF)이 주도하는 오픈소스 프로젝트로, Software-Defined RAN의 구현을 목표로 한다. 3GPP의 분산형 RAN 아키텍처와 O-RAN 표준을 기반으로 하며, Open Network Operating System(ONOS)을 활용한 near-RT RIC을 마이크로서비스 구조로 구현하고 있다. 특히 Go언어의 경량 동시성 모델과 gRPC 기반 인터페이스를 채택하여 클라우드 네이티브 환경에서의 유연한 배포와 확장을 지원한다. SD-RAN은 E2 Service Model(E2SM)과 E2 Application Protocol(E2AP) 같은 표준 인터페이스를 제공함으로써, xApp의 개발과 검증에 최적화된 구조를 갖추고 있으며, 쿠버네티스 기반의 배포 방식은 차세대 통신 아키텍처의 요구사항을 충족시킨다. 전체 시스템은 중앙집중형 제어 기능을 담당하는 near-RT RIC과 기지국 CU/DU 기능을 시뮬레이션하는 RAN 시뮬레이터(ransim)로 구성된다.

B. Golang & gRPC

Go 언어(Golang)는 구글에서 개발한 정적 타입의 컴파일 언어로, 간결한 문법과 고루틴(goroutine), 채널(channel), 컨텍스트(context)를 활용한 Communicating Sequential Processes(CSP) 기반의 동시성 처리 모델을 제공한다. 이는 전통적인 스레드 방식보다 코드의 간결성과 실행 성능 면에서 뛰어나며, 마이크로서비스 아키텍처에서의 비동기 처리와 에러 핸들링에 적합하다. Java의 스레드 방식과 비교할 때, Go의 경량 고루틴 기반 방식은 데드락(deadlock)과 레이스 컨디션(race condition) 방지 측면에서 뛰어난 효율성을 제공하며, 이러한 특성은 수천 개의 Virtualized Network Functions(VNF)을 안정적으로 처리하는 데에도 효과적일 것으로 기대된다 [2]. 정적 컴파일, 단일 바이너리 배포, 자동 메모리 관리(garbage collection) 등은 쿠버네티스 환경과의 높은 호환성을 보장한다. 이러한 구조적 이점을 바탕으로 한 Go 기반의 SD-RAN 프로젝트는 ORANalyst 연구에서도 스레드 풀 없이도 안정적인 동시성 성능을 보였으며 [3], 모듈화, 상태 비저장(stateless) 처리, 자동화된 오케스트레이션 등 클라우드 네이티브 RAN 설계 요구와도 잘 부합한다. 또한, Go는 Google Remote Procedure Call(gRPC)를 기본적으로 지원하여 마이크로서비스 간 실시간 인터페이스와 inter-Pod 통신에 최적화되어 있다 [4]. gRPC는 경량 직렬화 포맷인 Protocol Buffers(Protobuf)를 기반으로 하여, 네트워크 효율성과 명확한 데이터 스키마를 제공한다.

C. RAN 시뮬레이터 구조 및 동작 원리

Ransim은 기본적으로 쿠버네티스 환경에서 구동되며, YAML 파일로 정의된 대규모 RAN 토폴로지를 기반으로 단말, E2 Node, E2 Agent 등의 핵심 구성 요소로 구성된다. E2 Node는 O-RAN 구조에서 E2 인터페이스를 통해 near-RT RIC과 연결되는 RAN 구성 요소로, 각 Node는 하나 이상의 셀(Cell)을 관리한다 (그림 1). E2 Agent는 E2 인터페이스를 통한 통신을 담당하는 엔드포인트이다. 네트워크 토폴로지 정보(E2 Node, 셀, 단말, RF 파라미터 등)는 Store 객체에 저장되며, Store의 상태 변화는 채널

을 통해 전파되고 Watcher를 통해 모니터링된다. 전체적인 시뮬레이션 관리는 Manager 객체가 수행한다. 단말 이동성과 핸드오버(Handover)는 Event A3 기반의 측정(Measurement) 컨트롤러와 핸드오버 컨트롤러가 담당하며, Go의 채널과 컨텍스트를 활용한 경량 파이프라인으로 구현되어 효율적으로 병렬적인 시뮬레이션이 가능하다. 이를 통해 ransim은 Go의 경량 동시성 모델을 바탕으로, 대규모 RAN 토폴로지뿐만 아니라 단말 이동성과 핸드오버까지 유연하게 시뮬레이션할 수 있다 [5].

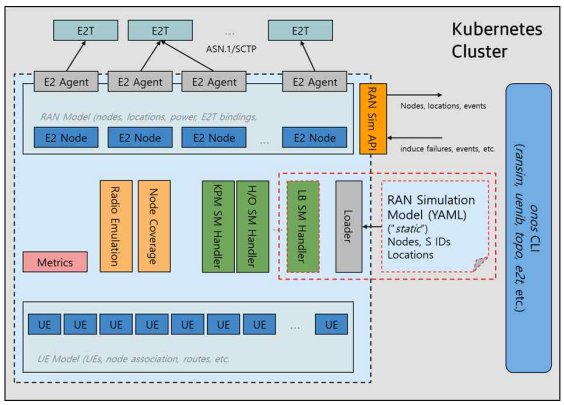

그림 1. RAN 시뮬레이터 구조 [5]

각 E2 Node 별로 생성되는 E2 Agent는 관련 Store 및 Service Model을 관리하며, near-RT RIC의 E2 Termination(E2T)과의 연결을 위한 e2Connection 클라이언트 객체를 생성한다. Ransim은 고루틴 기반의 비동기 처리로 E2 연결 상태를 모니터링하며, 자동 재시도와 에러 핸들링을 통해 효율적인 대규모 E2 Agent - E2T 연결 구성을 가능하게 한다. Ransim은 E2T와의 연결을 위해 SCTP 기반의 스트림(stream)을 생성하고 E2SetupRequest 메시지를 전송하는데, Protobuf로 직렬화된 바이트 스트림이 되고 ASN.1 형식으로 인코딩된 후 전송된다. 또한 e2Connection은 E2SetupResponse, RICSubscriptionRequest, RICControl 등의 메시지를 이벤트 기반으로 수신 및 처리한다. 추가적으로 ransim 내부에는 Northbound Interface(NBI)로 연결된 gRPC 서버가 존재하며 시뮬레이션 관리 및 모니터링 기능이 API 형태로 제공된다 (그림 2).

D. SD-RAN xApp 구조

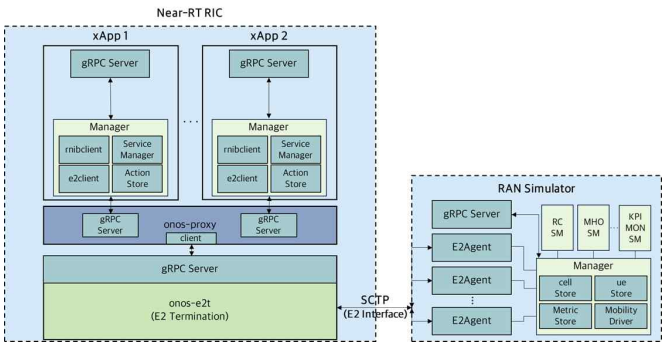

그림 2. Ransim & Near-RT RIC 연동

SD-RAN의 xApp은 ONOS RIC 아키텍처를 기반으로 Helm Chart를 통해 배포되며, YAML 설정을 기반으로 구동된다. 실행 시 Manager가 생성되어 메시지 브로커, 저장소, 서비스 모델, e2client, rnibclient, Subscription Manager, Controller 등을 포함하며, 이 구성요소들이 ransim과의 통신과 제어 알고리즘 실행을 담당한다. E2client는 E2T과 통신하며 RICIndication 수신, RICControl 전송, 구독 관리를 수행하고, rnibclient는 onos-topo와 연동해 네트워크 토폴로지 정보를 수집·제공함으로써 정책 기반 제어에 필요한 핵심 데이터를 지원한다. XApp은 Southbound Interface(SBI)를 통해 ransim과, Northbound Interface(NBI)를 통해 외부 시스템과 통신하며 상태 정보 및 제어 기능을 제공한다. Subscription Manager는 SubscriptionRequest를 통해 ransim과 구독을 맺고, 이후 수신되는 RICIndication 메시지를 Controller가 처리하여 제어 명령으로 변환한다. 예를 들어, onos-kpimon xApp은 KPI 모니터링 전용으로 RICIndication까지를 담당하며 RICControl을 전송하지 않지만, onos-mho xApp는 A3 이벤트 기반 핸드오버 제어까지 포함한다.

결과적으로, xApp 개발의 핵심은 각 애플리케이션의 목적에 맞는 Controller를 구현하고, E2T를 통한 메시지 송수신 흐름을 구성하며, ransim에서 수신되는 RICIndication 메시지를 목적에 맞게 처리하여 제어 알고리즘을 효과적으로 수행하는 데 있다. 이러한 계층적이고 구조화된 xApp 설계는 O-RAN 기반의 동적 네트워크 상태에 유연하게 대응하는 동시에, rnibclient를 통해 수집한 네트워크 구조 정보와 e2client를 통한 실시간 메시지 흐름을 기반으로 정교한 의사결정을 가능하게 하며, 이를 통해 클라우드 네이티브 기반 SD-RAN의 핵심 운영 철학을 실현하는 데 중요한 역할을 한다.

Ⅲ. 결론

본 연구는 O-RAN 표준과 Go 언어의 경량 동시성 모델을 결합한 SD-RAN 시뮬레이터를 설계하고 분석함으로써, 클라우드 네이티브 환경에서도 RAN 기능을 유연하게 검증할 수 있음을 보였다. 또한, Protobuf 기반 ASN.1 포맷을 준수하는 E2 인터페이스를 통해 시뮬레이터와 near-RT RIC 간의 SCTP 통신을 분석하고, xApp과의 표준 호환성을 검증하였다. 결과적으로 전반적인 분석을 통해 시뮬레이터 고도화 및 AI/ML 기반 지능형 xApp 개발에 필요한 핵심 요소를 도출하였으며, 이는 학계와 산업계에서 클라우드 네이티브 기반 O-RAN 연구·개발을 촉진할 수 있는 기반을 마련한다는 점에서 의의가 있다.

ACKNOWLEDGMENT

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-대학ICT연구센터(ITRC)의 지원(IITP-2025-RS-2021-II212046, 50%)과 과학기술정보통신부 및 정보통신기획평가원의 오픈랜 인력양성 프로그램(연세대) 연구 결과로 수행되었음(IITP-2025-RS-2024-00434743, 50%)

참 고 문 헌

[1] H. Liu, J. Zong, et al., "Cloud Native Based Intelligent RAN Architecture Towards 6G Programmable Networking," 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 2022, pp. 623-627

[2] Togashi, N., & Klyuev, V., "Concurrency in Go and Java: Performance analysis," 2014 IEEE International Conference on Information Science and Technology (ICIST), 412 - 417, 2014.

[3] T. Yang et al., "ORANalyst: Systematic Testing Framework for Open RAN Implementations," 33rd USENIX Security Symposium, Philadelphia, PA, 2024.

[4] P. Song, H. Peng and X. Zhang, "A Micro-Service Approach to Cloud Native RAN for 5G and Beyond," in IEEE Access, vol. 11, pp. 130257-130271, 2023.

[5] O. Sunay et al., "SD-RAN: ONF's Software-Defined RAN Platform Consistent with the O-RAN Architecture," Open Networking Foundation, White Paper, Feb. 2020.