https://doi.org/10.7840/kics.2024.49.8.1159

오픈소스를 활용한 Open RAN 테스트베드 구현 및 결과 분석

유 현 민◆, 홍 인 기°

Open RAN Testbed with Open Source: Implementation and Result Analysis

Hyun-Min Yoo◆, Een-Kee Hong°

요 약

차세대 이동통신 네트워크를 효율적으로 구축하고 data-driven 방식으로 운영하기 위한 개방형·지능형 radio access network (RAN)인 open RAN 기술이 본격적으로 주목을 받고 있다. 동시에, 서버 엔터프라이즈 분야에서 큰 성공을 거둔 오픈소스가 이동통신 분야로 확장되면서 open RAN의 생태계에 큰 영향을 미치고 있다. 본 논문은 open RAN 테스트베드를 구현하기 위해 요구되는 하드웨어 및 소프트웨어 기술과 RAN intelligent controller (RIC)에 대해 소개하고, 오픈소스를 활용한 테스트베드 구현 과정에 대해 분석한다. 무선 기술 개발을 중점으로 연구하던 기존 이동통신 연구자들에게 다소 생소한 네트워크 기술과 가상화, 마이크로서비스의 개념을 최대한 이해하기 쉽게 설명하기 위해 노력하였다. 또한, 테스트베드 동작 시 생성되고 송수신되는 개방형 인터페이스 내부의 메시지를 자세히 분석하고, xApp을 실행 결과를 제공하여 O-RAN 표준과 use case 대한 이해를 돕는다.

Key Words : Open RAN, Open Source, Virtualization, RAN Intelligent Controller, Next Generation Mobile Communications

ABSTRACT

Open Radio Access Network (Open RAN) technology, which aims to effectively deploy next-generation mobile communication networks and operate them in a data-driven manner, is in the spotlight. Simultaneously, the open source approach, which has achieved great success in the enterprise server domain, is extending into the mobile communication field and making a significant impact on the ecosystem of open RAN. In this paper, we introduce the hardware and software technologies and RAN intelligent controller (RIC) required to implement an open RAN testbed and provide an analysis of the testbed implementation process using open sources. We have made efforts to explain network technology, virtualization and microservices in a way that is accessible to existing mobile communication researchers who primarily focus on wireless technology. Additionally, we thoroughly explore a detailed analysis of messages within the open interfaces during testbed operation, and provide execution result of xApp, aiming to enhance understanding of O-RAN standards and use cases.

※ 본 논문은 연구재단 4단계 BK21 사업으로부터 지원받은 연구임 (GS-5-JO-NON-20230258)
◆ First Author : Kyunghee University Department of Electronics and Information Covergence, yhm1620@khu.ac.kr, 학생회원
° Corresponding Author : Kyunghee University Department of Electronics and Information Covergence, ekhong@khu.ac.kr, 종신회원
논문번호 : 202404-057-B-RN, Received April 1, 2024; Revised April 19, 2024; Accepted April 22, 2024

Ⅰ. 서 론

차세대 6G 네트워크의 핵심 기술인 open radio access network (open RAN)은 무선 신호 처리 장치(radio unit, RU)와 디지털 신호 처리 장치인 distributed unit (DU)/centralized unit (CU) 사이의 인터페이스인 프런트홀 (fronthaul)을 개방하고 표준화하여, 서로 다른 장비사의 RU와 DU가 연동될 수 있도록 정의하는 개방형 네트워크 기술이다[1]. 개방형 프론트홀이 표준화됨에 따라 이동통신 장비 시장에 RU를 개발하는 다수의 기업이 진입할 수 있게 되어 네트워크 구축 가격이 대폭 인하되는 효과로 이어진다[2]. 또한, 그림 1처럼 고성능 컴퓨팅 장비인 DU/CU (기지국) 장비의 하드웨어와 소프트웨어 사이의 인터페이스를 개방하여 기지국 장비를 programmable한 white box 형태로 구성할 수 있게 되었다. 따라서, 기지국 장비의 기능을 소프트웨어로 구현하고 필요에 맞게 커스터마이징 할 수 있는 '소프트웨어 기반 기지국' 개념이 탄생하게 되었다[3].

기지국을 소프트웨어 형태로 구축하고 유연하게 커스터마이징할 수 있게 됨에 따라, open RAN 표준에는 이들을 data-driven 방식으로 네트워크 상황에 따라 효율적/지능적으로 제어할 수 있게 하는 새로운 요소인 RAN intelligent controller (RIC)이 등장하였다[4]. RIC은 차세대 이동통신 시스템의 복잡한 무선 자원 관리 문제를 해결할 수 있는 인공지능 기반의 알고리즘들과 데이터 베이스 등의 소프트웨어 기술들이 실제로 구현될 수 있는 고성능의 서버 장비이며, 현재 세계 각국에서 관련 기술 선점과 use case 개발을 위해 중점적으로 연구되고 있다.

최근에는 컴퓨팅 기술의 발전으로 범용 CPU 기반의 상용 (commercial-off-the-shelf, COTS) 서버 장비에 소프트웨어 기반 기지국과 RIC을 구현할 수 있게 되었다. 결과적으로, 하드웨어 개발 역량 및 네트워크 설계 관련 지식과 소프트웨어만 있으면 COTS 서버 장비를 활용하여 기지국 장비와 RIC을 활용한 개방형/지능형 open RAN 테스트베드를 구현할 수 있다. 동시에, 기존에 서버 엔터프라이즈와 데이터 센터 분야에서 큰 성공을 거둔 오픈소스 문화가 이동통신 분야에도 접목되고 있다[5]. 예를 들어, 기지국 소프트웨어를 공개하는 오픈소스 프로젝트로는 Open Air Interface (OAI)가 있고, RIC을 개발하는 프로젝트로는 O-RAN Software Community (O-RAN SC) 등이 있다[6-7]. 이렇게 open RAN 분야에 오픈소스 생태계가 체계적으로 구축됨에 따라, COTS 서버 장비만 있으면 추가적인 비용 없이 누구나 open RAN 테스트베드를 구현할 수 있으며, 비용이 제한된 대학 연구실 수준에서도 직접 테스트베드를 구현하여 연구를 진행할 수 있게 되었다[8]. 다만, 이를 위해서는 이동통신 관련 지식에 더해 높은 수준의 하드웨어/소프트웨어 개발 역량과, 특히 가상화 (virtualization) 관련 지식이 필요하여 open RAN 연구를 처음 시작하는 기존 이동통신 연구자들에게 큰 기술적 장벽이 되고 있다[9].

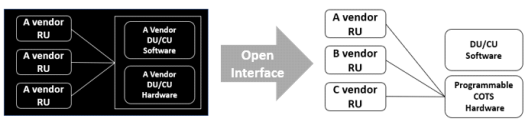

그림 1. 개방형 인터페이스를 통한 기지국 장비의 혁신
Fig. 1. The innovation of base station equipment via open interface

최근 들어 open RAN 기술에 대한 학계의 관심이 더욱 높아짐에 따라, 대학 연구실 수준에서 COTS 서버 장비와 오픈소스 소프트웨어를 활용하여 open RAN 테스트를 구현한 내용을 다룬 논문들이 다수 발행되었다. 참고문헌 [10]에서는 상용 네트워크 장비에 open RAN 표준에서 정의한 RIC과 RAN 간의 개방형/지능형 E2 인터페이스[11] 기능을 구현한 후, FlexRIC 프로젝트의 실시간 RIC과 연동하여 대학 캠퍼스 수준의 open RAN 테스트베드를 구현하였다. 참고문헌 [12]에서는 O-RAN SC의 실시간 RIC을 구축하기 위해 필요한 하드웨어/소프트웨어 지식을 간단히 다루고, 실시간 RIC의 실행 결과를 소개하였다. 참고문헌 [13]에서는 오픈소스 프로젝트에서 제공하는 기지국 소프트웨어와 범용 RU 장비인 universal software radio peripheral (USRP)를 이용하여 RAN을 구축하고 이를 O-RAN SC에서 제공하는 실시간 RIC과 E2 인터페이스로 연결하는 실험실 수준의 테스트베드를 구현하였다. 하지만, 상기 언급한 논문들은 테스트베드 구현을 위해 사용한 오픈소스 프로젝트에 대해 간단히 소개했을 뿐, 전체적인 네트워크 구축 과정과 소프트웨어 개발 과정에 대한 자세한 설명은 포함하지 않고 있다.

본 논문에서는 COTS 서버와 오픈소스 소프트웨어를 활용해 open RAN 테스트베드를 구현하는 과정을 상세히 소개한다. Open RAN 연구를 시작하는 통신 연구자들을 위해 RIC의 이해에 필요한 가상화/소프트웨어 기술, 테스트베드를 구현하기 위해 필요한 하드웨어/네트워크 지식과 이를 활용하여 물리적으로 테스트베드를 구현하는 과정을 설명하는 것을 목표로 한다. 다만, 가상화 기술과 네트워크 분야 모두 그 자체의 역사가 매우 깊고 방대하므로, 테스트베드 구현에 필요한 최소한의 내용만을 효율적으로 전달하고자 한다.

기존 연구에서 많이 다루어진 O-RAN SC의 오픈소스 소프트웨어 대신, 본 논문에서는 표준화 기구 Open Networking Foundation (ONF)에서 운영하는 SD-RAN 프로젝트의 오픈소스 소프트웨어를 사용한다[14]. O-RAN SC의 실시간 RIC은 open RAN 표준 제정 단체인 O-RAN Alliance의 공식 산하 오픈소스 플랫폼이어서 표준이 가장 빠르게 반영된다는 장점이 있지만, 오픈소스 커뮤니티를 통한 관리 및 버그 개선과 문서화 작업이 잘 이루어지지 않아 사용성이 현저히 떨어진다는 단점이 있다[15].

반면, SD-RAN 프로젝트는 활발한 커뮤니티 운영을 통해 개발자의 버그 리포팅에 빠르게 대처해 사용이 편리하고, 문서화가 잘 되어있어 처음 open RAN 연구를 시작하는 개발자가 사용하기에 비교적 용이하다. 다만, SD-RAN 프로젝트 역시 개발자가 가상화 기술과 네트워크 지식을 갖추고 있음을 가정하므로 연구 시작을 위해 넘어야 할 기술적 장벽이 높다. 본 논문은 이러한 격차를 해결하는 것을 목표로 하며, 기술적 이해에 필요한 가상화, 네트워크 지식을 상세히 설명한다. 이후 해당 지식들을 기반으로 SD-RAN 프로젝트에서 제공하는 오픈소스 소프트웨어 (DU/CU, 실시간 RIC)를 활용해 open RAN 테스트베드를 구현하는 과정을 소개한다. 더 나아가, 테스트베드에 활용한 실시간 RIC의 구조와 이를 실행할 때 시스템상에서 발생하는 E2 인터페이스 메시지를 상세히 분석하여 시스템으로 구현된 open RAN 표준에 대한 이해를 돕는다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 테스트베드 구현을 위해 필요한 RIC에 대한 사전 지식을 다룬다. 이 내용을 기반으로 3장에서는 물리적인 테스트베드의 구조 및 구현 과정과 구성 요소에 대해 상세히 설명한다. 4장에서는 테스트베드의 동작 결과를 graphic user interface (GUI) 캡처본과 함께 자세히 설명한다. 5장에서는 논문의 결론과 함께 향후 연구 방향을 소개하며 논문을 마친다.

Ⅱ. RIC과 이동통신 네트워크

2.1 RIC as a Server

기존 이동통신 네트워크는 하나의 기지국이 여러 단말에게 무선으로 서비스를 공급하는 구조로 이루어져 있으며, '무선'이라는 특수성으로 인해 유선 네트워크와는 별도로 표준이 개발되어 왔다. 하지만 다수의 기지국을 지능적으로 제어하는 RIC의 등장으로, 하나의 RIC이 다수의 기지국을 '유선'으로 제어하는 새로운 계층이 생성되었다. 그림 2는 기존의 무선 네트워크와 지능형 유선 네트워크의 구조를 보여준다. RIC과 기지국은 유선 인터넷을 기반으로 형성된 서버-클라이언트 구조로 모델링될 수 있으며, 서버로서의 RIC (RIC as a Server)을 효율적으로 운영하기 위한 최첨단 정보 기술들이 RIC에 적용되기 시작하였다. 이것이 무선을 연구하던 이동통신 연구자들이 RIC의 이해에 어려움을 겪는 근본적인 이유이며, 동시에 open RAN 전문가가 되기 위해서는 이동통신 기술과 정보 기술을 동시에 갖추어야 함을 시사한다.

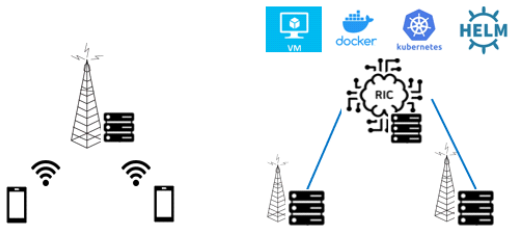

그림 2. 기존 무선 네트워크와 지능형 유선 네트워크
Fig. 2. Traditional wireless networks and intelligent wired networks.

따라서, 해당 절의 이어지는 내용은 오픈소스 RIC의 이해에 필요한 정보 기술의 기초를 최대한 쉽게 설명하며 이 기술들을 테스트베드 구현에 응용하는 방안을 제시한다. 핵심은 RIC이 서버로서의 역할을 수행하고, 서버의 가장 큰 목적은 어떤 기능을 가진 '어플리케이션'을 운용하고 이를 활용하여 '서비스'를 클라이언트에게 제공하는 것이다. 예를 들면, 대표적인 서버 엔터프라이즈인 구글은 GMail이라는 어플리케이션을 운용하며, 클라이언트들은 GMail 어플리케이션을 활용해 메일 전송, 수신, 삭제 등의 서비스를 요청하게 된다. RIC 역시 지능적으로 기지국을 제어하기 위한 어플리케이션들을 포함하고 있으며, 이들은 O-RAN 표준에 맞는 서비스를 제공하게 된다.

서버에서 운용되는 어플리케이션이 다수의 클라이언트로부터 요청받는 서비스를 효율적으로 서비스하기 위해 개발된 소프트웨어 기술이 바로 가상화이다[16]. 가상화는 '물리적인' 하드웨어 (CPU, 메모리, 스토리지)를 다수의 '논리적인' 요소로 분할하는 기술이며, 구글, 아마존 등 대규모 서비스를 운영하는 기업들이 하드웨어 자원을 효율적으로 사용하기 위해 연구되어 왔다. 초기 상용화된 가상화 기술은 하나의 물리적인 하드웨어에 별도의 운영체제를 갖는 다수의 어플리케이션을 운용하였다. 마치 가상의 컴퓨터가 하나의 물리적인 하드웨어에 구축된다고 하여 가상화라는 이름이 붙었으며, 별도의 운영체제를 갖는 어플리케이션을 가상머신

(virtual machine)이라고 한다.

하지만 가상머신은 어플리케이션마다 별도의 무거운 운영체제를 탑재하고 있어 속도가 느리다는 단점이 있다. 이 문제를 해결할 수 있는 기술은 컨테이너(container)로, 가상머신과 달리 별도의 운영체제 없이 어플리케이션 실행에 필요한 최소한의 바이너리 (bin)/라이브러리 (lib) 파일만을 탑재하여 격리한 경량화 패키지이다. 이는 리눅스 계열 운영체제의 자원 격리 기술을 통해 가능하며, 매우 빠른 속도로 어플리케이션을 실행할 수 있어 가상화의 장점을 극대화할 수 있다[17]. 자원 격리 기술을 통해 어플리케이션을 컨테이너로 격리하여 실행하고 관리할 수 있도록 하는 기술은 도커(Docker)라 하며, 사용자는 도커만 설치하면 원하는 어플리케이션들을 컨테이너로 만들 수 있다[18]. 그림 3은 물리적 하드웨어 위에 구현된 가상머신과 컨테이너 기반의 구조를 보여준다.

그림 3에서 확인할 수 있는 가상머신과 컨테이너의 핵심은, 이들이 하나의 물리적 하드웨어 위에 논리적으로 구성되어 있지만, 본질은 하나의 컴퓨터라는 점이다. 따라서 가상머신 혹은 컨테이너로 구현된 하나의 어플리케이션마다 별도의 internet protocol (IP) 주소와 포트 번호를 갖는다. RIC을 하나의 물리적 하드웨어 위에 여러 어플리케이션을 가상머신이나 컨테이너로 구현하게 되면, 각 서비스마다 별도의 IP 주소와 포트 번호를 갖게 되며, 테스트베드를 구현할 경우 DU/CU 장비와 인터넷 프로토콜로 통신할 수 있게 된다.

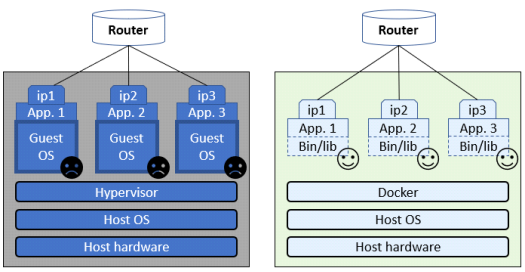

그림 3. 가상머신 기반 구조와 컨테이너 기반 구조의 비교
Fig. 3. Comparison of virtual machine-based and container-based structures.

2.2 Microservice Architecture

최근 가상화 기술과 함께 서버를 더욱 효율적으로 운영하기 위해 마이크로서비스 (microservice) 구조가 접목되고 있으며[19], SD-RAN과 O-RAN SC의 RIC 역시 마이크로서비스 구조를 채택하고 있다. 마이크로서비스 구조는 서버에 존재하는 어플리케이션들을 일체형 (monolithic)으로 구현하지 않고 이들을 모듈화(modularization)하여 다수의 작고 독립적인 서비스들로 나누는 구조이다. 마이크로서비스 구조의 각 어플리케이션들은 독립적으로 배포 (deployment)되고 운영될 수 있어, 서버 내의 특정 어플리케이션의 부하가 증가하면 해당 어플리케이션만 자원을 확장할 수 있어 서버의 전체적인 확장성 (scalability)을 향상시킨다. 마이크로서비스 구조로 형성된 어플리케이션들은 독립적으로 개발, 배포 및 업데이트될 수 있어 새로운 기능을 빠르게 도입할 수 있고, 이에 맞춰 기존 기능을 빠르게 수정할 수 있어 높은 유연성 (flexibility)을 가진다. 또한, 이들은 모두 독립적인 소프트웨어로 구성되기 때문에 필요에 따라 다른 프로그래밍 언어, 프레임워크, 데이터베이스 (database) 등을 사용할 수 있어 다양한 기술을 효과적으로 적용할 수 있다. 이러한 마이크로서비스 구조로 형성된 어플리케이션들을 컨테이너 형태로 구성하면, 각 어플리케이션을 격리된 소프트웨어 패키지로써 가볍게 실행할 수 있어 확장성과 유연성을 극대화할 수 있다.

다수의 어플리케이션들이 컨테이너로 구축되면 하나의 서버에 수많은 컨테이너가 존재하게 되는데, 개별적으로 컨테이너를 관리하는 것은 매우 비효율적이다. 이 문제를 해결하기 위해 다수의 컨테이너를 효율적으로 조직화하고 자동화하여 관리하기 위한 쿠버네티스 (Kubernetes) 플랫폼이 등장하였으며, 이를 컨테이너 오케스트레이션 (orchestration) 기술이라 한다[20]. 쿠버네티스 플랫폼에서는 어플리케이션을 구성하는 한 개 이상의 컨테이너를 '파드 (pod)'라는 단위로 묶어서 관리하며, 외부와 통신하며 이들의 기능을 외부에 공급하는 '서비스 (service)'라는 논리적인 개념과 함께 '클러스터 (cluster)'라는 쿠버네티스 공간에 배포된다[21]. 파드는 서버의 상황에 따라 지속적으로 재생성이 되며, 생성될 때마다 변동되는 내부 IP 주소가 할당되는 휘발적 특성을 가진다. 반면, 서비스는 외부와 통신을 하기 위한 목적으로 정의된 개념이어서 고정된 IP 주소를 가지며, 내부적으로 파드와 연결되어 사용자에게 어플리케이션의 기능을 제공한다. 즉, 사용자는 서비스의 고정 IP 주소를 통해서 내부 IP 주소를 가지는 파드와 연결되어 파드 내부의 컨테이너들이 구성하는 어플리케이션을 이용하게 된다. 다수의 파드와 서비스들을 효율적으로 관리하기 위해, 쿠버네티스는 용도에 따라 이들을 'namespace'라는 논리적인 그룹으로 나누어 관리한다. 그림 4는 사용자에게 서비스를 제공하는 쿠버네티스 클러스터의 전체적인 구조를 나타낸다.

쿠버네티스는 그 자체만으로도 개발 철학과 기술적 내용이 심도 있고 광범위하지만, 테스트베드 구현을 목

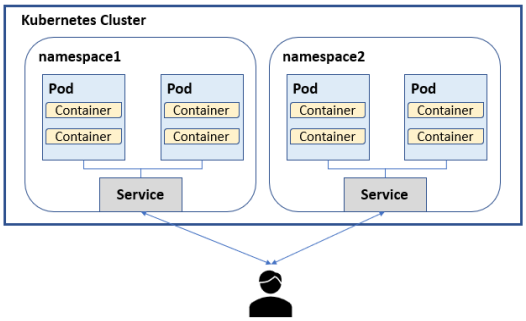

그림 4. 쿠버네티스 클러스터 구조
Fig. 4. Kubernetes cluster architecture.

적으로 하는 본 논문의 범위를 벗어나므로 상기한 내용으로 설명을 마친다. 테스트베드 구현에는 쿠버네티스의 개념에 대한 이해와 약간의 명령어만 숙지하는 것으로 충분하다. SD-RAN 프로젝트를 커스터마이징하여 원하는 지능형 기능을 구현하는 데는 해당 기술들에 대해 조금 더 깊은 이해가 필요하지만, 이를 다룬 내용은 향후 연구에서 다루도록 한다.

Ⅲ. 테스트베드 설계 및 구현

본 장에서는 Ⅱ장에서 다룬 배경지식과 SD-RAN 프로젝트의 오픈소스를 활용하여 테스트베드를 설계하고 구현하는 과정을 다룬다. 그림 5는 구현한 테스트베드의 물리적 구조를 나타낸다.

범용 RU 장비 USRP B210를 통해 무선 신호 처리부를 구성하고 COTS 서버 장비 Intel NUC을 활용하여 SD-RAN의 DU/CU, 실시간 RIC, 코어 네트워크를 구성하였다. SD-RAN의 DU/CU 소프트웨어는 OAI의 소스코드를 커스터마이징하여 O-RAN 표준에서 지정한 E2 인터페이스를 emulate하여 실시간 RIC과 호환될 수 있도록 하였다. DU/CU 장비와 실시간 RIC 장비는 공유기를 통해 유선 (이더넷)으로 연결되어 있다. DU/CU와 실시간 RIC 장비는 각자 고정된 내부 IP (192.168.1.107, 192.168.1.108)를 가지고 있으며, CU와 실시간 RIC은 O-RAN 표준의 E2 인터페이스로 통신한다.

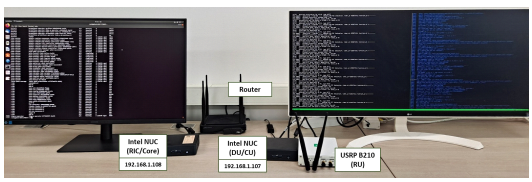

그림 5. SD-RAN 테스트베드 구조
Fig. 5. SD-RAN testbed architecture.

DU/CU 장비와 USRP B210은 USB 3.0 케이블을 통해 물리적으로 연결되어 있다. DU와 CU는 한 대의 Intel NUC 장비에 일체형으로 내장되어 논리적으로 구분되어 동작한다. 코어 네트워크와 실시간 RIC 역시 한 대의 Intel NUC 장비를 공유한다. 정리하면, 물리적 테스트베드 구현을 위해서는 2개의 이더넷 케이블, 1개의 USB 3.0 케이블, 2대의 Intel NUC, 1대의 USRP B210, 공유기 등의 하드웨어가 필요하다. DU/CU와 코어 네트워크, 실시간 RIC의 소프트웨어는 SD-RAN 공식 가이드 [22]에서 다운로드 받을 수 있다.

소프트웨어 설치 후, 코어 네트워크와 실시간 RIC의 기능을 이루는 각 요소들이 파드로 구현되어 쿠버네티스 클러스터상에서 'riab'이라는 namespace에서 동작하고 있음을 확인할 수 있다. 그림 6은 riab namespace에서 동작하는 파드들의 목록을 나타낸다.

실시간 RIC은 'onos'라는 접두어를 가지는 파드들로 이루어져 있으며, 코어 네트워크 기능과 쿠버네티스 관련 dependency들이 riab namespace에서 함께 동작하고 있다. 접두어 onos는 'open network operating system'의 약어로, 가상화 기반의 네트워크 기능 구현을 위한 플랫폼이다[23]. ONF에서 software-defined network (SDN)와 network function virtualization (NFV) 솔루션의 생태계를 넓히기 위해 2014년에 공개한 오픈소스이며[24], 가상화 기반으로 구축된 SD-RAN 프로젝트의 실시간 RIC 역시 onos 플랫폼을 준용하였다. 표 1에 나타낸 것처럼, onos 플랫폼으로 이루어진 파드들은 크게 실시간 RIC 자체의 기능과, 실시간 RIC에서 동작하며 기지국 네트워크를 제어하는 소프트웨어인 'xApp'으로 분류할 수 있다.

onos-e2t (e2 termination)는 E2 인터페이스의 proxy로 작동하며 RIC의 하단에 위치하여 기지국의 연결을

```
yhm2@yhm2:~$ kubectl get pod -n riab
NAME                                READY   STATUS    RESTARTS
cassandra-0                         1/1     Running   0
hss-0                               1/1     Running   0
mme-0                               4/4     Running   0
oai-ue-0                            1/1     Running   0
onos-a1t-68c59fb46-7rtkm            2/2     Running   0
onos-cli-c7d5b54b4-bmbwd            1/1     Running   0
onos-config-5786dbc85c-w4bmm        3/3     Running   0
onos-e2t-5798f554b7-sj57c           2/2     Running   0
onos-kpimon-555c9fdb5c-85q6m        2/2     Running   0
onos-rsm-7b6d84b5fc-gqkzm           2/2     Running   0
onos-topo-6b59c97579-87xfr          2/2     Running   0
onos-uenib-6f65dc66b4-2tn5b         2/2     Running   0
pcrf-0                              1/1     Running   0
sd-ran-consensus-0                  1/1     Running   0
sd-ran-consensus-1                  1/1     Running   0
sd-ran-consensus-2                  1/1     Running   0
spgwc-0                             2/2     Running   0
upf-0                               4/4     Running   0
```

그림 6. riab namespace 내부 파드 목록
Fig. 6. List of pods in the riab namespace.

표 1. SD-RAN 실시간 RIC의 파드 분류
Table 1. Pod classification for SD-RAN near-real time RIC.

RIC	onos-e2t / onos-topo / onos-uenib / onos-cli / onos-config / onos-a1t
xApp	onos-kpimon / onos-rsm

관리한다. 또한, 기지국에서 생성된 모든 메시지를 O-RAN 표준에 정의된 저수준 프로토콜인 E2 application protocol (E2AP)로 정의하여 수신한다. onos-topo는 네트워크의 토폴로지 및 셀 정보를 저장하는 데이터베이스이며, RIC은 이 정보를 활용하여 연결된 기지국의 정보를 확인할 수 있다. onos-uenib은 사용자 단말과 관련된 정보를 포함하는 데이터베이스이며, 현재 네트워크에 존재하는 사용자 단말의 세부 정보 및 서비스하는 기지국들에 대한 정보를 저장한다. onos-config는 RIC 내부의 마이크로서비스 및 xApp을 위한 설정값을 저장한다. onos-cli는 RIC과 사용자가 명령을 통해 상호 작용할 수 있도록 지원하는 인터페이스이다. onos-a1t는 아직 SD-RAN 프로젝트에서 구현되지 않은 비실시간 RIC과의 인터페이스에 해당한다.

SD-RAN의 실시간 RIC에는 default로 두 가지 xApp이 배포되어 있다. onos-kpimon xApp은 key performance indicator (KPI) monitoring을 의미하며, onos-e2t를 통해 기지국으로부터 수집되는 KPI 정보를 RIC에 보고한다. onos-rsm xApp은 역시 onos-e2t를 통해 DU/CU에 구성된 slice의 자원을 관리하는 기능을 담당한다. 그림 7은 상기한 파드들로 구성된 SD-RAN 실시간 RIC의 블록도이다.

실시간 RIC은 onos-e2t를 통해 기지국과 연결되므로, 테스트베드의 DU/CU를 onos-e2t와 연결해 주어야 한다. 이는 DU/CU의 configuration 파일에 onos-e2t의 IP 주소를 입력함으로써 연결이 구현될 수 있다. 단, 3장에서 언급한 대로 onos-e2t 파드가 아닌 고정 IP 주소를 갖는 onos-e2t 파드의 서비스를 통해 연결되어야 한다. onos-e2t 파드의 서비스를 조회하면 그림 8과 같은 결과를 얻을 수 있다.

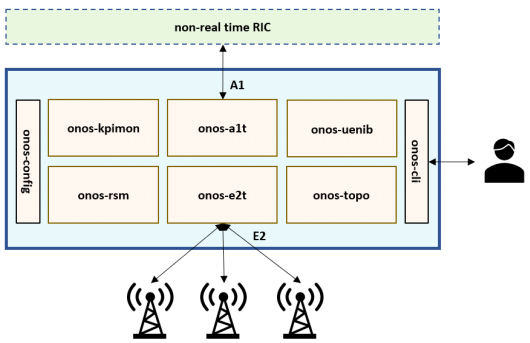

그림 7. SD-RAN 실시간 RIC 블록도
Fig. 7. Block diagram of SD-RAN near-real time RIC.

onos-e2t 파드의 서비스는 onos-e2t, onos-e2t-hs, onos-e2t-external로 구성되어 있다. 이 중 외부와의 연결을 위한 서비스는 onos-e2t-external이며, SCTP 프로토콜을 통해 36401번 포트 (port)로 해당 서비스와 연결될 수 있다. O-RAN 표준에서 기지국-RIC 간의 통신 프로토콜로 SCTP를 지정하므로[25], SD-RAN의 실시간 RIC 역시 SCTP 프로토콜을 통한 E2 인터페이스 통신을 지원한다. onos-e2t-external 서비스는 자체적인 고정된 내부 IP (192.168.85.203)를 갖지만, 해당 서비스가 구동되는 물리적 장비의 IP인 192.168.1.108에 포트번호 36401을 명시하면 장비와 연결될 수 있다. 즉, DU/CU의 configuration 파일에 IP 주소 192.168.1.108과 포트번호 36401을 지정하여 onos-e2t-external 서비스를 통해 실시간 RIC과 통신하는 시스템을 갖출 수 있다.

SD-RAN의 실시간 RIC과 DU/CU 장비의 물리적 연결을 쿠버네티스 관점에서 재구성한 구조를 그림 9에 나타내었다. 실시간 RIC이 구동하는 물리적 장비에 쿠버네티스 클러스터가 설치되어 있고, riab namespace에 onos-e2t 파드가 실행되고 있으며, onos-e2t 서비스를 통해 외부와 연결되고 있다. onos-e2t 서비스들 중 onos-e2t-external이 DU/CU 장비와 물리적으로 연결

그림 8. onos-e2t의 서비스 목록
Fig. 8. List of services of onos-e2t.

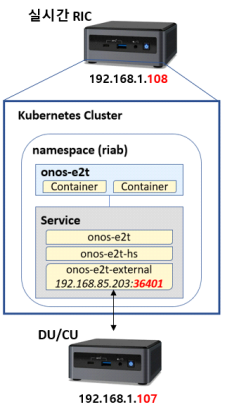

그림 9. 쿠버네티스 관점에서의 테스트베드 구조
Fig. 9. Testbed Structure from Kubernetes' perspective.

되어 E2 인터페이스를 통해 SCTP 프로토콜로 통신하는 구조를 갖추고 있다.

O-RAN 표준에서는 E2 인터페이스로 RIC에 연결된 RU, DU, CU를 포함한 기지국 장비를 'E2 Node'라고 명명한다[26]. 이어지는 장에서는 RIC과 E2 Node의 E2 인터페이스를 통한 연결을 위해 필요한 절차에 대해 다룬 후, 테스트베드를 실행한 결과 발생하는 메시지에 대해 분석한다.

Ⅳ. 테스트베드 실행 결과

E2 인터페이스로 연결된 E2 Node와 실시간 RIC은 E2AP를 통해 연결을 맺고, 프로토콜의 각 절차마다 정보를 교환하기 위한 E2 메시지를 송수신한다[27]. 이 메시지들은 기존 3GPP 표준에 정의된 4단계의 원리(principle)를 준용하고 있으며[28], 각 단계는 표 2에 정리하였다. 이 메시지 프로토콜은 기존 3GPP 표준에서 정의한 기지국 간의 X2, 기지국과 mobility management entity (MME) 간의 S1 등 이동통신 네트워크 구성 요소 간의 인터페이스에도 사용되었으며, O-RAN 표준에서 정의한 실시간 RIC과 기지국 간의 E2 인터페이스에도 그대로 적용되어 기존 표준과의 호환성을 극대화하였다[29].

Message와 IE는 메시지의 중요도를 나타내는 'criticality' 태그를 갖는다. 크게 ignore과 reject으로 나뉜다[28]. 표준 정의상 ignore 태그는 송신 측에서 진행하는 절차가 수신 측에서의 지원 여부와 상관없이 진행될 경우 붙고, 수신 측에서 지원하지 않는 경우 거절되는 절차는 reject 태그가 붙는다. 대체적으로 해당 절차가 새롭게 시작하는 경우 reject을, 사전에 정보 교환이 이루어진 경우는 ignore 값을 갖는다.

SD-RAN 테스트베드의 E2 인터페이스 연결 절차는 그림 10과 같이 진행되며, 절차마다 표 2의 원리에 따른 메시지를 주고받는다. SCTP 연결 이후 E2 Setup 절차를 통해 E2 인터페이스 연결을 맺는다. 실시간 RIC은 E2 Connection 절차를 통해 E2 Node에게 두 기기 간의

표 2. 호환성 보장을 위한 E2 메시지의 4단계 원리
Table 2. Four principles to ensure compatibility of E2 messages.

Level	Principle
1	Protocol
2	Message
3	Information element (IE)
4	Values

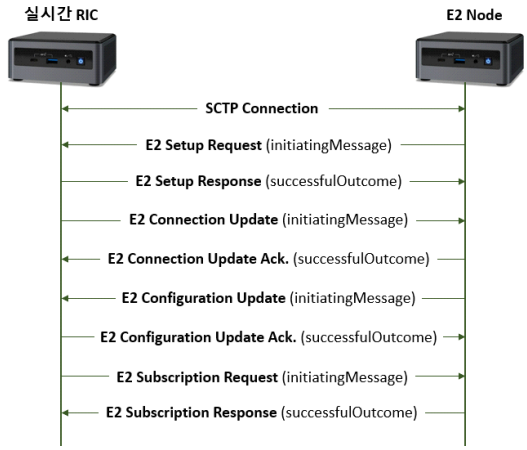

그림 10. E2 인터페이스 연결 절차
Fig. 10. Connection procedure of E2 interface.

연결에 대한 정보를 알리고, E2 Node는 환경 설정에 대한 변경 사항을 E2 Configuration Update 절차를 통해 실시간 RIC에게 전송한다.

그 후 실시간 RIC은 E2 Node에서 제공하는 기능에 대한 정보를 받기 위한 E2 Subscription Request 절차를 거치며 본격적인 지능형 역할을 수행한다. SCTP 연결 이후의 각 절차는 모두 초기 요청과 이에 대한 response 혹은 ack 수신 과정을 거친다. 초기 요청과 ack 과정은 표 2의 4단계의 원리 중 'Message'의 값으로 각각 'initiatingMessage'와 'successfulOutcome' 값을 갖는다.

그림 11은 위 단계 중 테스트베드의 E2 Subscription Request 과정에서 발생하는 E2 메시지를 나타낸다. 상기 언급한 3GPP 표준의 4단계 원리를 모두 준용하는 동시에, XML 언어로 기술되어 있다[30]. E2AP를 protocol data unit (PDU)으로 명시한 후 Message 원리가 시작된다. RIC Subscription Request는 초기 요청에 해당하므로 값은 initiatingMessage를 가지며, 이어지는 E2 연결 절차에 해당하는 코드 (procedure code)와 그것의 criticality를 명시한다. 뒤이은 메시지를 통해서, E2AP 표준에서 RIC Subscription의 코드는 8번으로 정의되어 있으며, criticality로 reject을 갖도록 설정되어 있음을 알 수 있다. 최종적으로, Message는 RIC subscription 절차의 initiatingMessage에 해당하는 RIC Subscription Request를 value로써 반환한다.

이후 'RICsubscriptionRequest-IE' Message를 통해 호출되는 세 개의 E2AP IE가 subscription request 절차를 이루며, 이들에 대한 정리는 표 3과 같다. 각 IE에 대한 설명은 다음과 같다.

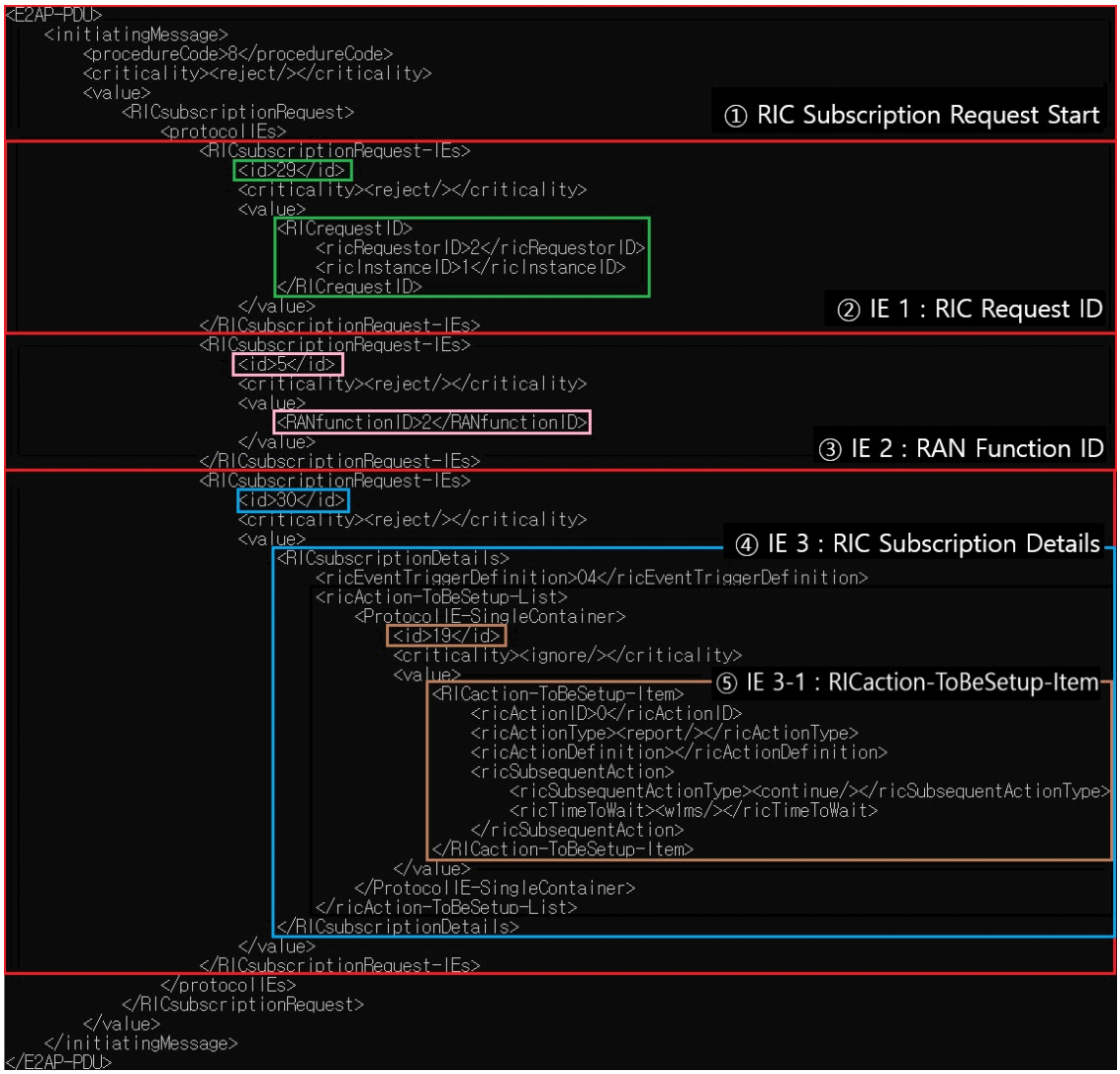

그림 11. E2 subscription request 과정에서 발생하는 E2 메시지
Fig. 11. E2 messages generated from E2 subscription request

1) 'RIC request ID' IE는 실시간 RIC이 E2 Node에게 알리는 자신의 ID (requestor ID, instance ID)가 기록되어 있다.

2) 'RAN function ID' IE는 실시간 RIC이 구독하길 원하는 E2 Node의 function의 ID를 전달한다.

3) 세 번째 IE인 'RIC subscription Details' IE에는 실시간 RIC이 E2 Node에게 전달하는 핵심 내용이 담겨있다. RIC subscription Details IE는 표준 정의상 또 다른 E2AP IE인 'RICaction-ToBeSetup-Item'을 호출하도록 설계되어 있다.

4) 'RICaction-ToBeSetup-Item IE'는 실시간 RIC이 E2 Node에 취할 행동이 상세히 기록되어 있다. 그림 11의 메시지에서는 실시간 RIC이 E2 Node의 상태를 보기 위한 'report' 행동을 취하며, 1ms 단위로 지속 (continue)한다.

표 3. RIC subscription request 과정에서 호출되는 E2AP IE
Table 3. The E2AP IEs called in RIC subscription request procedure.

Index	IEs (id)
1	RIC request ID (29)
2	RAN function ID (5)
3	RIC subscription Details (30)
3-1	RICaction-ToBeSetup-Item (19)

RIC subscription request 이후에는 그림 10에 명시되었듯 RIC subscription response 절차가 이어지며, 해당 절차는 RIC subscription request 절차에 대한 ack에 해당하므로 Message 값으로 'successfulOutcome'을 가짐을 쉽게 이해할 수 있다.

이처럼 E2AP 프로토콜 기반의 E2 인터페이스 연결 과정에서 발생하는 E2 메시지는 3GPP 표준의 4단계 원리를 준용하고 있으며, 각 절차마다 실시간 RIC과 E2 Node는 그림 11의 형태를 갖춘 E2 메시지를 송수신하며 연결을 완료하게 된다.

그림 12는 O-RAN의 대표적인 use case인 KPI monitoring을 위한 onos-kpimon xApp의 실행 결과를 나타낸다. 구현한 테스트베드의 DU/CU 장비 (E2 Node)는 e2:4/e00/2/64라는 Node ID로 인식되고 있으며, Object ID와 Global ID도 함께 할당되는 것을 볼 수 있다. 그 외에 연결된 단말에 대한 radio resource control (RRC) 정보를 확인할 수 있으나, 현재는 구현한 테스트베드에 단말이 연결되지 않아 0으로 표시된다. 향후 연구에서는 단말을 연결한 후 KPI monitoring 등 다양한 O-RAN use case를 실험할 계획이다.

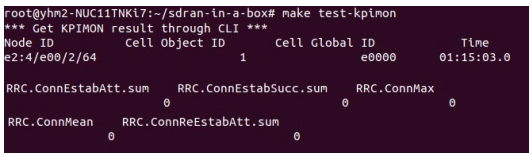

그림 12. onos-kpimon xApp 실행 결과
Fig. 12. Execution result of onos-kpimon xApp.

V. 결 론

본 논문은 SD-RAN 프로젝트의 오픈소스를 활용하여 open RAN 테스트베드를 구현하고 동작 결과를 O-RAN 표준 관점에서 상세히 분석하였다. 테스트베드 구현에 필요한 가상화 기술과 네트워크 기술에 대해 소개하였으며, 장비 간 연결을 위한 쿠버네티스의 기술적 개념을 다루었다. 구현된 테스트베드의 실행을 통해 RIC과 E2 Node 간의 E2 인터페이스 연결 절차에서 발생하는 E2 메시지와 xApp의 실행 결과를 확인할 수 있었고, 이를 상세히 분석하여 O-RAN 표준과 use case에 대한 깊은 이해를 도왔다.

논문에서 구현한 테스트베드는 SD-RAN 프로젝트에서 제공하는 기본적인 기능만을 탑재하고 있다. 향후 연구에서는 커스터마이징을 통하여 연구자가 개발한 알고리즘을 xApp으로 제작하여 테스트베드에서 실행하는 연구가 진행될 수 있을 것이다. 이를 위해서는 xApp 소프트웨어에 대한 심층적인 연구와 높은 수준의 프로그래밍 능력을 요구한다. 향후 본 논문에서 소개한 테스트베드 구현 방법을 기반으로, 더 높은 수준의 구현 연구가 진행되어 open RAN 분야의 생태계 확장에 기여할 수 있기를 기대한다.

References

[1] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, "Toward next generation open radio access networks: What O-RAN can and cannot do!," *IEEE Netw.*, vol. 36, no. 6, pp. 206-213, Jul. 2022. (https://doi.org/10.1109/MNET.108.2100659)

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Commun. Surv. Tuts.*, vol. 25, no. 2, pp. 1376-1411, Jan. 2023. (https://doi.org/10.1109/COMST.2023.3239220)

[3] H. Holma, A. Toskala, and T. Nakamura, *5G technology: 3GPP evolution to 5G-advanced*, 2nd Ed., John Wiley & Sons, 2024.

[4] B. Balasubramanian, et al., "RIC: A RAN intelligent controller platform for AI-enabled cellular networks," *IEEE Internet Comput.*, vol. 25, no. 2, pp. 7-17, Mar. 2021. (https://doi.org/10.1109/MIC.2021.3062487)

[5] S. Sirotkin, *5G radio access network architecture: The dark side of 5G*, John Wiley & Sons, 2020.

[6] *Open Air Interface*, Retrieved 9 Mar. 2024, from https://openairinterface.org/

[7] *O-RAN software community*, Retrieved 9 Mar. 2024, from https://o-ran-sc.org/

[8] M. Hoffmann, et al., "Open RAN xApps design and evaluation: Lessons learnt and identified challenges," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 2, pp. 473-486, Feb. 2024. (https://doi.org/10.1109/JSAC.2023.3336190)

[9] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for next-generation

SD-RANs," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.*, pp. 411-425, New York, USA, Dec. 2021. (https://doi.org/10.1145/3485983.349487)

[10] M. V. Ngo, N.-B.-L. Tran, H.-M. Yoo, et al., "RAN intelligent controller (RIC): From open-source implementation to real-world validation," *ICT Express*, Feb. 2024. (https://doi.org/10.1016/j.icte.2024.02.001)

[11] O-RAN Working Group 1, "*O-RAN Architecture Description-v11.0*," Feb. 2024.

[12] D. H. Kim, H. Nam, S. Baek, and M. S. Kang, "Constructing open RAN test environment," in *KCC 2023*, pp. 1374-1376, Jeju Island, Korea, Jun. 2023.

[13] H. M. Yoo, et al., "A study on the O-RAN software community and RAN intelligent controller implementation," in *KICS Summer Conf. 2023*, pp. 844-845, Jeju Island, Korea, Jun. 2023.

[14] ONF, Retrieved Mar. 4, 2024, from https://opennetworking.org/open-ran/

[15] A. Kak, V. -Q. Pham, et al., "ProSLICE: An open RAN-based approach to programmable RAN slicing," in *Proc. IEEE Glob. Commun. Conf.*, pp. 197-202, Rio de Janeiro, Brazil, Dec. 2022. (https://doi.org/10.1109/GLOBECOM48099.2022.10001497)

[16] J. Bhimani, Z. Yang, M. Leeser, and N. Mi, "Accelerating big data applications using lightweight virtualization framework on enterprise cloud," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, pp. 1-7, Waltham, USA, Sep. 2017. (https://doi.org/10.1109/HPEC.2017.8091086)

[17] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf.* Cloud Eng., pp. 386-393, Tempe, USA, Mar. 2015. (https://doi.org/10.1109/IC2E.2015.74)

[18] Docker, Retrieved Mar. 8, 2024, from https://www.docker.com/

[19] F. Auer, et al., "From monolithic systems to Microservices: An assessment framework," *Inf. Softw. Technol.*, vol. 137, Sep. 2021. (https://doi.org/10.1016/j.infsof.2021.106600)

[20] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81-84, Sep. 2014. (https://doi.org/10.1109/MCC.2014.51)

[21] Kubernetes, *Learn Kubernetes Basics*, Retrieved 9 Mar. 2024, from https://kubernetes.io/docs/tutorials/kubernetes-basics/

[22] *SD-RAN Documentation*, Retrieved 19 Apr. 2024, from https://docs.sd-ran.org/master/index.html

[23] ONF, *onos*, Retrieved 17 Mar. 2024, from https://opennetworking.org/onos/

[24] P. Berde, et al., "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Wkshp. Hot Topics Softw. Defined Netw.*, pp. 1-6, New York, USA, Aug. 2014. (https://doi.org/10.1145/2620728.2620744)

[25] O-RAN Working Group 3, "*Near-RT RIC APIs specification-v1.0*," Mar. 2023.

[26] O-RAN Working Group 3, "*Near-RT RIC Architecture-v4.0*," Mar. 2023.

[27] O-RAN Working Group 3, "*E2 General Aspects and Principles-v4.0*," Mar. 2023.

[28] 3GPP TR 25.921 v7.0.0, "*Guidelines and principles for protocol description and error handling*," Jun. 2007.

[29] O-RAN Working Group 3, "*E2 Application Protocol (E2AP)-v3.0*," Mar. 2023.

[30] ITU-T Recommendation X.680, "*Information technology – Abstract syntax notation one (ASN.1): Specification of basic notation*," Jul. 2002.

유 현 민 (Hyun-Min Yoo)

2021년 2월 : 경희대학교 전자공학과 학사 졸업
2023년 2월 : 경희대학교 전자정보융합공학과 석사 졸업
2023년 3월~현재 : 경희대학교 전자정보융합공학과 박사과정

<관심분야> Mobile Communication, O-RAN, 5G
[ORCID:0000-0001-6385-2655]

홍 인 기 (Een-Kee Hong)

1989년 2월 : 연세대학교 전기공학과 학사 졸업
1991년 2월 : 연세대학교 전기공학과 석사 졸업
1995년 8월 : 연세대학교 전기공학과 박사 졸업
1995년~1999년 : SKT 선임연구원
1999년~현재 : 경희대학교 전자공학과 교수
2012년~현재 : 미래창조과학부 주파수 정책 자문위원
2013년~현재 : 5G 포럼 주파수 위원회 위원장
2014년~현재 : 국무조정실 주파수 심의위원
2018년~현재 : 한국통신학회 이사, 수석부회장, 회장
2018년~현재 : 과기정통부 전파정책 자문위원
2021년~현재 : 위성통신포럼 주파수 위원회 위원장

<관심분야> Mobile Communication, O-RAN, 5G
[ORCID:0000-0001-5086-4008]