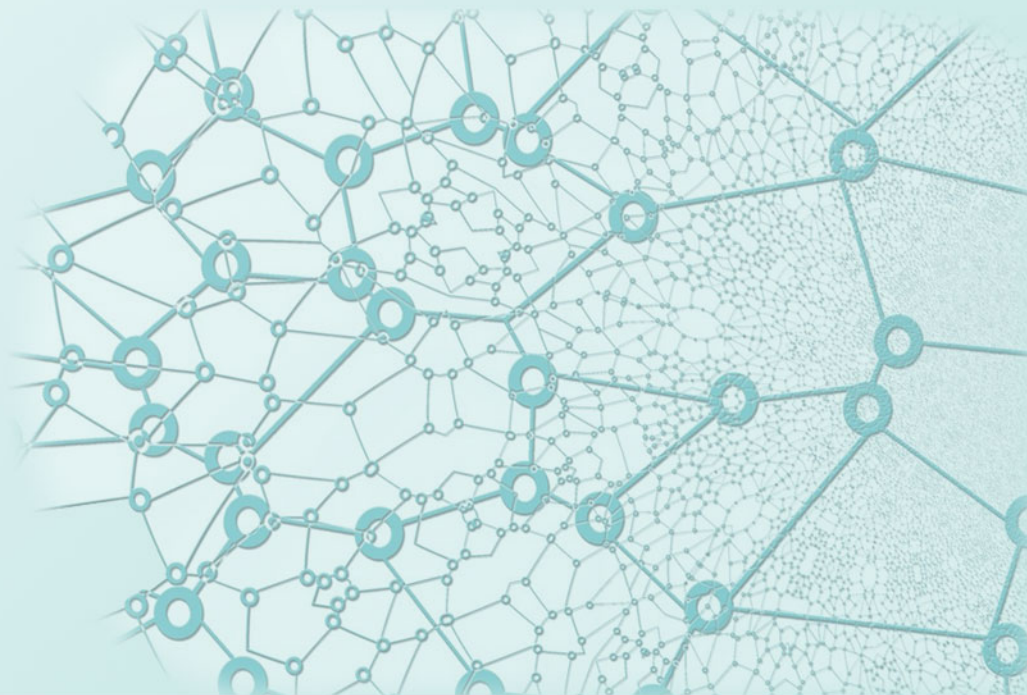


KNOM Review | 27권 2호, 2024년 12월 | ISSN 2287-1543 (Online)

KNOM REVIEW

Network Operations & Management

Vol. 27 No.2 | DEC. 2024



한국통신학회 통신망운용관리연구회

The Committee on Korean Network Operations and Management (KNOM)
Korean Institute of Communication Sciences (KICS)

<http://www.knom.or.kr>

목 차

제27권 제2호, 2024년 12월

자기주권형 신원인증 시스템의 클러스터 노드 구성에 따른 성능 평가 김정헌, 강창훈, 우종수, 홍원기	1
ZSM/ENI 기반의 자율 네트워크 및 서비스 관리 양우석, 이재오	12
머신 러닝을 활용한 스마트 컨트랙트 취약점 탐지 연구 동향 이의찬, 방지원, 최미정	24
K-P2PLab: P2P 네트워크 토폴로지 분석을 위한 테스트베드 및 분석 플랫폼 개발 이성욱, 주홍택	40
네트워크 관점에서의 3D 콘텐츠 표현 방식 분석 유상우, 홍원기	49

자기주권형 신원인증 시스템의 클러스터 노드 구성에 따른 성능 평가

김 정 현*, 강 창 훈*, 우 중 수*, 홍 원 기°

Performance Evaluation of Self-Sovereign Identity Systems Based on Cluster Node Configuration

Jeongheon Kim*, Changhoon Kang*, Jongsoo Woo*, James Won-Ki Hong°

요 약

자기주권형 신원인증 패러다임은 기존의 중앙 기관이 모든 신원정보를 통제하는 방식의 위험성에서 벗어나, 개인이 직접 신원정보를 관리하고, 특정 서비스 이용을 위해 원하는 정보만을 제출할 수 있다는 점에서 보안성이 뛰어나다. 그러나 여전히 사용자 대부분은 중앙 집중적 방식의 높은 사용자의 편의성과 속도에 익숙해져 있으며, 이는 자기주권형 신원인증 시스템의 가장 큰 걸림돌이 되고 있다. 본 논문에서는 자기주권 신원인증 시스템을 구현하기 위한 오픈소스 프로젝트인 A Cloud Agent - Python (ACA-Py)과 이에 특화된 하이퍼레저 인디(Hyperledger Indy) 블록체인을 활용하여 전체적인 시스템을 구성하였다. 또한, ACA-Py를 클러스터 형태로 구성하여, 클러스터 내부의 노드 개수 변화에 따른 실제 자기주권 신원증명 시스템의 자격증명 발급 및 검증 등의 성능을 평가한다. 실험 결과 노드가 1개인 경우 부하 분산 실패로 높은 응답시간과 에러율이 발생한 반면, 3개 또는 5개 노드로 구성된 클러스터는 에러 없이 우수한 성능을 보였다.

Key Words : Self-Sovereign Identity, Performance evaluation, Hyperledger Aries

ABSTRACT

The self-sovereign identity (SSI) paradigm offers high security by allowing individuals to manage their own identity information and selectively disclose specific details required for certain services, rather than relying on a centralized authority to control all identity information. However, most users are still accustomed to the high convenience and speed of centralized methods, which remains a significant obstacle for SSI systems. In this paper, we construct a comprehensive SSI system using A Cloud Agent - Python (ACA-Py), an open-source project designed for SSI implementation, and Hyperledger Indy blockchain. Additionally, we configure ACA-Py in a clustered form and evaluate the performance of credential issuance and verification within the SSI system as the number of nodes in the cluster changes. In our experiment, clusters with a single node experienced high response times and error rates due to load balancing failures, whereas clusters with 3 or 5 nodes showed excellent performance without any errors.

※이 논문은 경찰청이 지원한 '경찰건강 스마트관리(www.kipot.or.kr)'사업의 지원을 받아 수행된 연구결과입니다. [과제명: 경찰관 맞춤형 건강관리 서비스를 위한 지능형 빅데이터 통합플랫폼 개발 / 과제번호: 220222M01]

• First Author : Pohang University of Science and Technology (POSTECH), Department of Computer Science and Engineering, kjheon1118@postech.ac.kr

° Corresponding Author : POSTECH, Department of Computer Science and Engineering, jwkhong@postech.ac.kr

* POSTECH, Department of Computer Science and Engineering, {chkang, woojs}@postech.ac.kr

논문번호 : KNOM2024-02-02, Received November 18, 2024; Revised December 10, 2024; Accepted December 22, 2024

I. 서 론

전통적인 신원정보 관리 방식은

기업 또는 정부 등의 중앙 기관이 전체적인 정보를 저장하며 관리한다. 이러한 중앙 집중적 방식은 데이터 관리가 쉽다는 장점이 있으나, 데이터가 유출될 경우 대규모 피해가 발생한다는 단점이 있다. 또한, 개인정보 보호의 중요성이 주목받는 추세에 의해 자기주권 신원(Self-Sovereign Identity, SSI)이 등장하였다^[1].

자기주권 신원은 개인이 자신의 신원정보를 중앙 기관에 위탁 관리하지 않으며, 개인이 신원정보에 대한 통제권을 보유하는 패러다임을 뜻한다. 중앙 기관이 다량의 개인정보를 보관하지 않기에 데이터 유출의 위험성이 적으며, 개인이 원하는 서비스에만 개인정보 동의를 통해 신원정보를 제공할 수 있다. 이때, 서비스 이용에 필요한 특정 신원정보만을 제공할 수 있을 뿐 아니라, 영지식 증명(Zero-Knowledge Proof)을 활용하여 신원정보 자체를 제공하지 않고도 서비스가 요구하는 조건을 만족한다는 사실을 증명할 수 있다^[2].

이러한 자기주권 신원을 구현하기 위해 탈중앙화 식별자(Decentralized Identifier, DID)가 사용되며, 해외뿐 아니라 국내에서도 DID 기술을 활용한 자기주권 신원 생태계를 넓혀가고 있다^[3]. 마이크로소프트는 W3C가 정의한 DID 표준을 기반으로 하는 Entra Verified ID 서비스를 출시하였다^[4]. 해당 ID를 기반으로 다양한 자격증명을 발급하여 검증할 수 있도록 하는 서비스이다. 국내에서는 SKT가 타 주요 통신사 및 금융기관과 협력하여 DID 기반 모바일 전자증명 서비스인 Initial을 개발하였다^[5]. 신원증명뿐 아니라, 정부의 전자증명서 발급 및 고객센터 서류 제출 및 검증 등의 서비스를 제공하고 있다.

DID는 DID Method를 통해서 생성 및 관리되며, 다양한 서비스들이 독자적으로 DID Method를 구현하여 활용한다^[6]. 글로벌 공개 SSI 네트워크인 Sovrin 네트워크를 운영하는 Sovrin Foundation에서는 did:sov 메소드를 정의하여 사용한다^[7]. 앞에서 소개한 initial 서비스는 did:ssw 메소드를 정의하였다^[8]. 누구나 DID Method를 정의하고 구현할 수 있기에, 현재 DID Method는 200개를 넘어가고 있다^[9].

다양한 DID Method를 활용할 수 있는 오픈소스

프레임워크로는 OpenWallet Foundation에서 관리하는 A Cloud Agent - Python (ACA-Py)가 존재한다^[10]. ACA-Py는 SSI를 DID에 기반한 검증가능한 자격증명(Verifiable Credential, VC)[11]을 통해 구현하고, 자격증명 발급, 검증, 폐기 등의 기능을 제공하는 에이전트이다. Sovrin Network^[12], Findy Agency^[13], IDUnion^[14], IdRamp^[15], Initial^[5] 등 다양한 자기주권형 신원인증 시스템들이 ACA-Py를 기반으로 운영되고 있다.

ACA-Py 소스코드의 동시성 관리에 관한 부분을 분석해보면, 각 기능별로 하나의 ACA-Py 에이전트가 할당되어 시스템이 구성된 경우 시스템이 동시에 처리할 수 있는 요청 건수가 적도록 제약이 걸려있다. 즉, 더 많은 에이전트를 시스템에 추가할수록 소프트웨어적으로 제한되어 있는 동시 처리 작업의 최대 개수를 늘릴 수 있다.

본 연구에서는 ACA-Py를 기반으로 한 자기주권형 신원인증 시스템을 구성하여 자격증명 발급 및 검증과정의 성능을 측정한다. 시스템을 구성하고 있는 ACA-Py는 각 역할 군(발급자, 소유자, 검증자)별로 여러 개의 컨테이너가 배치되어 클러스터를 이루며, 이를 통해 더 많은 수의 요청을 적절히 분산하여 동시에 처리할 수 있도록 하였다. 이때, 클러스터를 이루고 있는 컨테이너의 개수에 따라 전체적인 시스템의 성능 변화를 측정 및 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 자기주권형 신원인증 시스템에 관한 연구 배경과 ACA-Py의 성능 평가에 관련된 연구를 소개한다. 3장에서는 전체적인 자기주권형 신원인증 시스템의 구조 및 ACA-Py 클러스터에 관해서 설명한다. 4장에서는 실험 및 결과에 대해서 분석한다. 5장에서는 결론 및 향후 연구에 관하여 기술한다.

II. 연구 배경 및 관련 연구

2.1. 연구 배경

자기주권형 신원증명 시스템은 개인이 자신의 신원 정보에 대한 주권을 가지고 관리하며, 특정 서비스를 선택하여 신원정보를 제공할 수 있는 특징을 가지고 있다. 이러한 특성으로 인해 자기주권형 시스템은 발급자(Issuer), 소유자(Holder), 검증자(Verifier), 저장소(Registry)가 핵심 요소로 구성된다^[16]. 해당 네 가지 구성 요소의 구조와 상호작용 방식을 통해 신원정보를 중앙 기관의 개입 없이 관

리할 수 있다.

발급자는 자격증명(신분증, 증명서, 자격증, 면허 등)을 발행하는 믿을 수 있는 주체이다. 발급자는 소유자로부터 자격증명 발급에 대한 요청을 받을 수 있으며, 이에 대한 자체적인 유효성 검증을 거친다. 유효성이 확인되면 DID에 기반을 둔 자격증명을 발행하게 되는데, 이때 발급자는 자신의 비밀키를 활용하여 서명을 생성하고, 이를 자격증명에 포함한다. 발급자는 자신을 식별하기 위한 DID 문서와 공개키를 공개적이며 위변조가 어려운 저장소에 등록해 놓음으로써 특정 자격증명이 해당 발급자로부터 발급된 것임을 증명할 수 있다.

소유자는 자격증명을 소유하고 관리하는 개인이나 조직으로, 자신의 자격증명을 전자 지갑 등을 사용하여 안전하게 보관하고, 필요한 경우 검증자에게 부분적으로 제출할 수 있다. 검증자에게 제출하는 경우, 자격증명은 검증가능한 프레젠테이션(Verifiable Presentation)으로 구성되며, 발급자의 서명에 더해 소유자의 비밀키를 활용한 서명까지 포함하여 검증자에게 제출함으로써 특정 프레젠테이션이 해당 소유자로부터 제출된 것임을 증명할 수 있다.

검증자는 소유자가 제출한 자격증명(프레젠테이션)이 신뢰할 수 있는지 확인할 수 있다. 해당 자격증명이 지정된 발급자로부터 발급된 것이 맞는지 발급자 서명의 유효성을 통해 검증할 수 있다. 또한, 특정 자격증명에는 유효기간 등이 설정되어 있거나, 검증자가 제공하는 서비스의 조건을 충족시키지 못하는 경우(e.g. 서비스의 나이 제한이 19세이지만 제출된 자격증명에 기록된 정보는 19세 미만인 경우) 등을 고려하여 서비스 제공 여부를 결정할 수 있다.

저장소는 발급자, 소유자, 검증자가 DID 및 여러 메타데이터를 포함하는 DID 문서를 안전하게 저장하고 관리하는 공간이다. 많은 수의 자기주권형 신원증명 시스템이 저장소를 분산원장을 가지는 블록체인을 채택하고 있으며^[17], 이를 통해 DID 및 여러 데이터의 위변조를 막음으로써 자기주권 신원증명 시스템의 신뢰성을 높일 수 있다.

2.2. 관련 연구

Muhle et al. 은 자기주권 신원증명 시스템을 구축하기 위해서는 전체적으로 어떠한 요소들이 필요한지 설명하였다^[26]. Soltain et al. 은 ACA-Py 및 Hyperledger Indy 블록체인을 사용하는 Sovrin

Network과 다양한 자기주권 신원증명 시스템을 소개하였다^[27]. Jolocom^[28]은 Ethereum을 기반으로 동작하는 오픈소스 자기주권 신원증명 플랫폼으로, 생성된 DID는 IPFS^[29]에 저장된다. 또한 SelfKey^[30]와 LifeID^[31]는 영지식 증명을 사용하여 최소한의 필요한 데이터만 공유하도록 자기주권 신원증명 시스템을 구현하였다.

Siqueira et al. 은 ACA-Py에 기반을 둔 자기주권 신원증명 시스템을 구축하여 실제 성능을 측정하였다^[18]. 이때, 발급자와 검증자는 1개의 ACA-Py 노드로 구성하였고, 소유자는 2개의 노드로 구성하였다. 이를 기반으로 ACA-Py 노드 간의 연결, 자격증명의 발급 및 검증과정에 걸리는 시간을 측정하였으며, throughput과 서버의 CPU 사용량을 측정하였다. 동시에 발급 및 검증 요청을 하는 스레드가 많아질수록 throughput이 증가하였으나, 80회의 동시 요청 횟수가 넘어갈수록 에러율이 증가하는 것을 확인할 수 있었다. 해당 논문을 제외하면 ACA-Py의 실제 성능을 측정하는 연구를 확인할 수 없었다.

본 논문에서는 위에서 소개한 관련 연구에서 더 나아가, ACA-Py를 기반으로 자기주권 신원증명 시스템을 구축하고 이를 실제 운용한다고 가정했을 때, 이 시스템의 성능 개선을 위한 방법의 제안 및 검증을 목표하고 있다. ACA-Py 에이전트의 수를 클러스터 구성으로 늘리는 방법을 통해 시스템의 성능이 실제로 얼마나 향상될 수 있는지 검증한다.

III. 시스템 구조

본 장에서는 ACA-Py를 기반으로 구성한 자기주권형 신원인증 시스템의 전체 구조를 설명하고, ACA-Py를 클러스터 형태로 구성한 이유를 설명한다. 또한, 시스템에서 사용하는 자격증명 발급 과정과 검증과정에 대하여 설명한다.

3.1. ACA-Py 기반 자기주권형 신원인증 시스템 구조

일반적으로 자기주권형 신원인증 시스템은 발급자, 소유자, 검증자가 상호작용하는 구조를 이루며, 본 논문에서 구현한 시스템도 이와 같은 형태를 따른다. 시스템의 전체 구조는 그림 1과 같이, 리버스 프록시(Reverse Proxy), ACA-Py 컨트롤러, ACA-Py 클러스터, 하이퍼레저 인디 블록체인으로 구성된다. 리버스 프록시는 사용자들의 요청을 받아들이며 해당

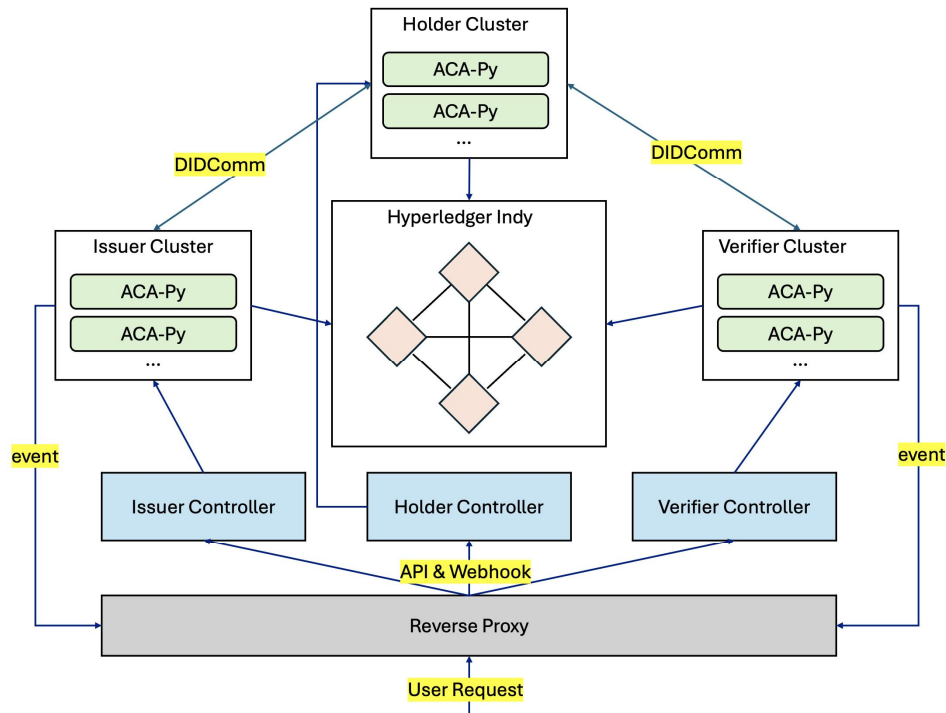


그림 1. 자기주권형 신원증명 시스템 구조
Fig. 1. Self-Sovereign Identity System Architecture

요청을 처리하여야 하는 ACA-Py 컨트롤러에 전달하는 역할을 한다. 또한, 컨트롤러가 ACA-Py 클러스터에 요청을 전달하면, ACA-Py는 작업을 처리한 후 해당 결과를 event 형태로 다시 리버스 프록시에 전달하여 event에 대응되는 컨트롤러에 전달하는 역할을 한다.

컨트롤러는 리버스 프록시로부터 전달된 사용자의 요청을 처리할 뿐 아니라, 자격증명 발급 또는 ACA-Py 간의 connection을 생성하는 과정에 대한 state를 자동으로 감지하여 추후의 과정을 자동으로 진행하며, 이에 관한 결과를 사용자에게 전달해주는 역할을 한다.

ACA-Py는 DID 발급, DID 및 자격증명 관리를 위한 지갑(Wallet) 생성, ACA-Py 간의 연결 생성, 자격증명 발급, 자격증명 검증의 역할을 수행한다. 이때, 발급자나 검증자의 ACA-Py와는 달리 소유자의 ACA-Py는 하나의 ACA-Py에서 여러 개의 지갑을 발급하고 자격증명을 관리할 수 있도록 하는 다중 테넌시(Multi-tenancy) 모드로 동작한다. 기본적으로 하나의 ACA-Py는 하나의 지갑을 생성하여 관리하기에 발급자 또는 검증자는 이에 적합하지만, 다양한 사용자들을 관리하여야 하는 소유자 ACA-Py는 이에 적합하지 않다. 이때, 본 논문에서는 소유자가 따로 지갑을 생성할 수 있는 모바일

어플리케이션을 사용하지 않는 경우를 가정한다. 그림 2와 같이 소유자의 ACA-Py를 다중 테넌시 모드로 동작시켜 하나의 tenant가 하나의 사용자를 관리하도록 설정하였으며, 같은 ACA-Py 내에 존재하는 tenant 간에는 같은 inbound와 outbound transport를 사용하도록 하여, 하나의 ACA-Py가 다수의 사용자를 관리할 수 있도록 구성하였다.

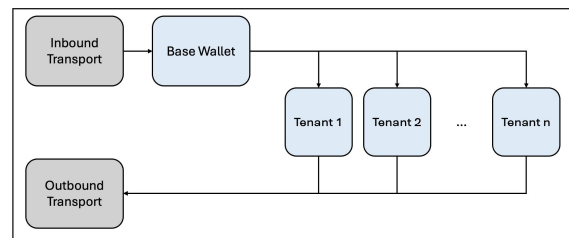


그림 2. 다중 테넌시 ACA-Py 구조
Fig. 2. Multi-tenancy ACA-Py Structure

DID 및 메타데이터의 저장소로는 하이퍼레저 인디(Hyperledger Indy)^[19] 블록체인을 사용하였다. 하이퍼레저 인디는 SSI를 구현하기 위해 설계된 오픈소스 분산원장 프로젝트로, 블록에 대한 합의를 수행하지 않고 트랜잭션 단위로 합의를 수행하여 분산원장에 기록한다는 특징이 있다. 또한, Aries 프로토콜을 통해 DIDComm 프로토콜을 지원하므로, ACA-Py와 상호 운용이 가능하다. 본 시스템에

서는 브리티시 컬럼비아에서 개발한 VON (Verifiable Organizations Network)^[20] 네트워크를 활용하여 4개의 indy 노드로 운영되는 하이퍼레저 인디 블록체인을 구성하였다.

3.2. ACA-Py 클러스터 구성

ACA-Py의 실제적인 성능을 평가했던 논문^[18]에 의하면, 동시에 80명 이상의 사용자가 동시에 자격증명의 발급 및 검증과정을 진행할 시, 점진적으로 에러 발생률이 올라가는 것을 확인할 수 있었다. 이러한 에러가 발생하는 원인으로는 자격증명 발급 및 검증과정은 모두 ACA-Py 간의 상호작용이 있어야 진행될 수 있으나, 각 ACA-Py 하나당 outbound transport를 처리할 수 있는 건이 200건이 한계치이며, 하나의 호스트에서는 오직 50건이 한계치이기에 발생하는 것으로 예상된다^[10].

이를 해결하기 위해서 본 시스템에서는 각 ACA-Py 역할 군마다 여러 개의 ACA-Py 노드로 구성된 클러스터를 구성하였다. 성능 평가를 위하여 클러스터의 구조를 변경하여 테스트를 진행한다. 하나의 클러스터가 가질 수 있는 최대 ACA-Py 노드의 개수는 5개이며, 최소 개수는 1개로 구성하여 차이를 두었다.

3.3. 자격증명 발급 및 검증과정

ACA-Py를 통해 자격증명 발급 및 검증을 위해서는 다양한 과정을 거쳐야 한다. 가장 먼저 소유자 ACA-Py에 연결된 사용자의 지갑을 생성하여야 한다. ACA-Py에서는 현재 askar 타입 지갑과 indy 타입 지갑을 모두 지원하고 있으나, indy 지갑을 사용하고 관리하기 위한 Indy SDK의 사용종료가 예정되어 있기에, 본 시스템에서는 askar 타입 지갑을 사용하였다. 다중 테넌시 모드로 동작하는 ACA-Py에서 지갑을 생성하게 되면, 지갑의 고유 id와 Bearer token이 생성되며, 이는 지갑과 관련된 기능들을 사용하는 데 필요하다.

지갑을 생성한 후, 자격증명을 발급하기 위해서는 발급자 ACA-Py와의 연결을 생성하는 과정이 선행되어야 한다. 본 시스템에서는 ACA-Py 간의 DID 교환을 통해 연결을 생성하는 Aries RFC 0023 (DID Exchange Protocol v1)^[21]을 사용하였다. 기존 ACA-Py 간의 연결을 생성하기 위해서는 Aries RFC 0160 (Connection Protocol)^[22]을 사용하여야 했으나, DID Exchange 프로토콜이 더 효율적이고 높은 보안성을 제공하여 Connection 프로토콜에 대

한 지원이 종료될 예정이다. 자격증명 발급을 담당하는 ACA-Py와 자격증명 검증을 담당하는 ACA-Py 모두 소유자 ACA-Py 내의 테넌트 별로 연결이 필요하다. 그림 3에서는 발급자와 검증자가 자신의 공개 DID를 전달하면, 소유자가 이를 기반으로 DID Exchange 요청을 전송하며, 발급자와 검증자는 이에 대한 반응을 전송하여 연결이 생성되는 과정을 설명하고 있다.

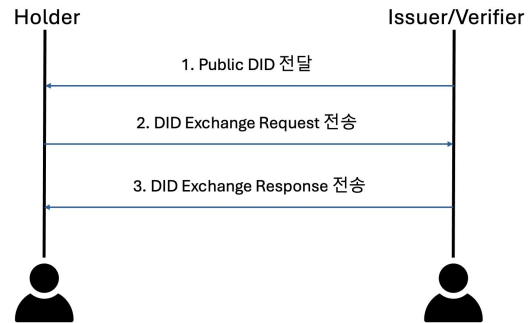


그림 3 DID 교환 과정
Fig. 3. DID Exchange Process

자격증명 발급 과정은 Aries RFC 0453 (Issue Credential Protocol 2.0)^[23]을 통해 진행되며 총 5단계의 과정으로 진행된다. 그림 4를 통해서 확인할 수 있듯이 자격증명 발급 과정은 소유자가 자격증명 발급에 대한 proposal을 발급자에게 전송하여 제안한다. 그 뒤의 offer, request, issue 모두 해당 proposal에 정의된 내용을 기반으로 진행된다. 이처럼 단계별로 발급 과정을 진행함으로써, 각 당사자에게 유연성과 제어권을 제공하여 상호 신뢰와 보안을 강화할 수 있다.

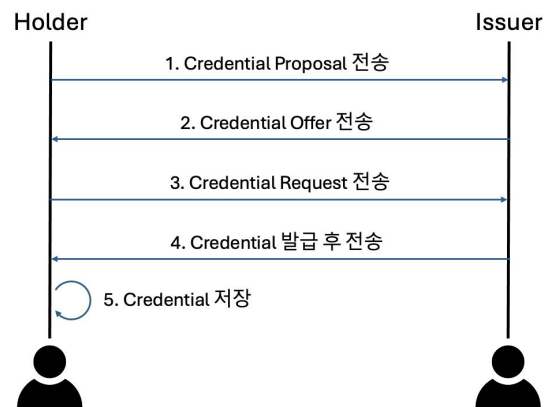


그림 4. 자격증명 발급 과정
Fig. 4. Verifiable Credential Issuance Process

자격증명 검증과정은 Aries RFC 0454 (Present Proof Protocol 2.0)^[24]을 통해 진행되며, 발급 과정과 유사하게 총 5단계의 과정으로 진행된다. 소유자가 자격증명 검증에 대한 proposal을 검증자에게 전

송하여 제안하며, 검증자는 이를 확인 후 해당 자격 증명 검증을 위한 presentation을 요청한다. 소유자가 해당 presentation을 제출하면, 발급자 서명, 소유자 서명 등의 과정을 통해 자격증명의 유효성을 확인한 뒤, 검증 결과를 소유자에게 전달한다.

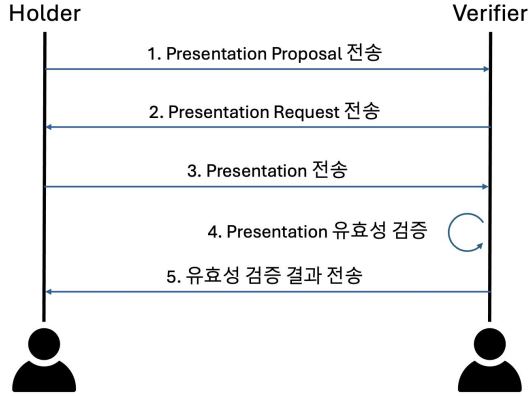


그림 5. 자격증명 검증 과정

Fig. 5. Verifiable Credential Verification Process

IV. 실험 및 분석

4.1. 실험 환경

자격증명 발급을 위해서는 해당 자격증명이 어떤 특성들을 포함할지 미리 정의하여야 하며, 이를 스키마(schema)라고 부른다. 또한, 스키마는 여러 발급자 사이에서 공통으로 사용될 수 있으므로, 해당 스키마에 추가로 발급자의 공개키 및 메타데이터를 포함하는 것을 자격증명 정의(credential definition)라고 한다. 자격증명 정의를 통해 특정 발급자가 특정 스키마에 따라 자격증명을 발급할 때, 해당 자격증명에 대한 발급자 정보 및 검증 방법을 지정할 수 있다. 스키마와 자격증명 정의 모두 하이퍼레저 인디 블록체인에 저장되어 신뢰성을 높일 수 있다. 본 논문에서는 실험을 위해 간단한 구조의 스키마를 사용하며, 해당 스키마의 구조는 표 1에서 확인할 수 있듯이 이름, 이메일 주소, 날짜로 구성된다.

표 1. 자격증명 스키마 구조

Table 1. Verifiable Credential Schema Structure

Schema Attributes	Attribute details
name	사용자의 이름
email	사용자의 이메일 주소
date	자격증명 발급요청 일자

본 연구에서는 ACA-Py 클러스터의 노드 개수를

조절함으로써 성능 변화를 테스트하고 분석하는 것을 목표로 한다. 이를 위해 ACA-Py 클러스터들(발급자, 소유자, 검증자)의 노드 개수를 1개, 3개, 5개로 설정하고, 동시에 자격증명 발급 및 검증을 요청하는 사용자의 수를 점차 늘리는 시나리오를 통해 테스트를 진행한다.

이때, ACA-Py 노드의 환경설정은 같은 클러스터 내부의 ACA-Py는 같은 설정 정보를 가지며, 세부적인 설정 정보는 표 2에서 확인할 수 있다. 각 노드는 비슷한 환경으로 설정되어 있으나, 소유자는 다중 테넌시 모드로 동작한다는 차이점이 있으며, ACA-Py 간의 연결을 생성할 때 공개적인 DID를 통해 신뢰를 확보하여야 하는 발급자와 검증자는 공개적인 초대를 가능하도록 설정하였다. 빠른 작업이 진행될 수 있도록 발급자는 소유자의 여러 단계에 걸친 자격증명 발급을 자동화할 수 있도록 설정하였으며, 반대로 검증자는 소유자의 자격증명 검증을 자동화할 수 있도록 설정하였다.

표 2 ACA-Py 환경 변수

Table 2. ACA-Py Environment Variables

Environment	Issuer	Holder	Verifier
version	1.0.0	1.0.0	1.0.0
db	postgres	postgres	postgres
wallet type	askar	askar	askar
multi-tenancy	false	true	false
did method	sov	sov	sov
public invites	true	false	true
auto accept invites	true	true	true
auto accept credentials	true	false	false
auto accept presentations	false	false	true

본 구현에서는 자기주권형 신원인증 시스템을 구성하는 모든 요소(블록체인, 컨트롤러, ACA-Py, 리버스 프록시)는 하나의 서버 내에서 동작하도록 구현하였다. 서버의 CPU는 AMD EPYC 7443P 2.75GHz 24 Core를 사용하였으며, RAM은 64GB

표 4. 노드 1개 구성 클러스터 실험 결과
Table 4. Node #1 Cluster Evaluation Result

API Request	Cluster - node #1					
	20 users		50 users		100 users	
	response time (ms)	error rate (%)	response time (ms)	error rate (%)	response time (ms)	error rate (%)
Step - Create wallet	100	0	150	0	272	0
01 - Holder create wallet	100 (5-274)	0	150 (4-777)	0	272 (5-1338)	0
Step - Connect issuer/holder	723	0	8596	22.35	24225	78.22
01 - Issuer invitation	9 (3-90)	0	10 (3-204)	0	73 (2-536)	0
02 - Holder receive invitation	714 (117-1295)	0	8586 (105-30272)	22.35	24152 (360-31133)	78.22
Step - Issue credential	1485	0	4215	32.85	6823	98.98
01 - Holder send proposal	314 (65-527)	0	2062 (5-30115)	27.61	4697 (5-30746)	90.1
02 - Holder send request	627 (200-983)	0	1194 (11-10157)	30	1986 (9-30869)	98.98
03 - Holder store credential	544 (108-916)	0	959 (10-2609)	32.85	140 (9-1027)	98.98
Step - Connect verifier/holder	746	0	1638	0	1264	0
01 - Verifier invitation	7 (3-42)	0	8 (3-88)	0	27 (3-544)	0
02 - Holder receive invitation	739 (110-1167)	0	1630 (105-3631)	0	1237 (125-2882)	0
Step - Verify credential	798	0	1279	33.83	149	97.22
01 - Holder send proposal	303 (54-573)	0	562 (4-1692)	25.19	49 (4-616)	66.67
02 - Holder send presentation	495 (156-745)	0	717 (20-2081)	33.83	100 (12-321)	97.22

표 5. 노드 3개 구성 클러스터 실험 결과
Table 5. Node #3 Cluster Evaluation Result

API Request	Cluster - node #3					
	20 users		50 users		100 users	
	response time (ms)	error rate (%)	response time (ms)	error rate (%)	response time (ms)	error rate (%)
Step - Create wallet	38	0	93	0	161	0
01 - Holder create wallet	38 (5-430)	0	93 (4-552)	0	161 (4-1507)	0
Step - Connect issuer/holder	201	0	667	0	1405	0
01 - Issuer invitation	7 (3-76)	0	10 (2-211)	0	15 (2-459)	0
02 - Holder receive invitation	194 (111-423)	0	657 (112-1752)	0	1390 (101-2634)	0
Step - Issue credential	587	0	1435	0	2749	0
01 - Holder send proposal	112 (63-238)	0	288 (65-710)	0	605 (68-1510)	0
02 - Holder send request	288 (207-562)	0	633 (215-1239)	0	1112 (200-1840)	0
03 - Holder store credential	187 (115-329)	0	514 (108-1167)	0	1032 (112-1840)	0
Step - Connect verifier/holder	203	0	703	0	1543	0
01 - Verifier invitation	7 (3-75)	0	7 (3-117)	0	20 (2-458)	0
02 - Holder receive invitation	196 (112-383)	0	696 (115-1560)	0	1523 (117-2520)	0
Step - Verify credential	312	0	765	0	1535	0
01 - Holder send proposal	96 (53-232)	0	279 (55-686)	0	632 (55-1144)	0
02 - Holder send presentation	216 (152-415)	0	486 (149-1018)	0	903 (158-1504)	0

표 6. 노드 5개 구성 클러스터 실험 결과
Table 6. Node #5 Cluster Evaluation Result

API Request	Cluster - node #5					
	20 users		50 users		100 users	
	response time (ms)	error rate (%)	response time (ms)	error rate (%)	response time (ms)	error rate (%)
Step - Create wallet	21	0	52	0	136	0
01 - Holder create wallet	21 (5-71)	0	52 (4-739)	0	136 (4-4621)	0
Step - Connect issuer/holder	156	0	581	0	798	0
01 - Issuer invitation	4 (2-15)	0	245 (2-1036)	0	282 (3-835)	0
02 - Holder receive invitation	152 (99-315)	0	336 (117-985)	0	516 (102-1413)	0
Step - Issue credential	471	0	784	0	1193	0
01 - Holder send proposal	85 (59-197)	0	169 (70-434)	0	260 (57-1479)	0
02 - Holder send request	239 (186-341)	0	364 (205-786)	0	534 (193-1331)	0
03 - Holder store credential	147 (105-255)	0	251 (106-589)	0	399 (119-1245)	0
Step - Connect verifier/holder	160	0	336	0	642	0
01 - Verifier invitation	4 (3-43)	0	21 (2-687)	0	66 (3-488)	0
02 - Holder receive invitation	156 (96-331)	0	315 (109-794)	0	576 (106-1505)	0
Step - Verify credential	253	0	455	0	679	0
01 - Holder send proposal	69 (47-179)	0	156 (51-398)	0	246 (54-985)	0
02 - Holder send presentation	184 (142-299)	0	299 (161-590)	0	433 (160-1116)	0

이며, OS는 Ubuntu 18.04를 사용하였다.

실험은 Postman^[25]의 API들을 연속적으로 호출할 수 있는 Collection 기능, 가상의 동시 접속 사용자 수를 설정할 수 있는 Performance Test 기능을 통해 진행하였다. 1개, 3개, 5개의 노드로 이루어진 클러스터별로, 각각 20명, 50명, 100명의 동시 접속 사용자 수를 설정하여 성능을 테스트하였다.

표 3. 성능 평가 지표

Table 3. Performance Evaluation Metrics

Evaluation Metrics	Details
response time	단계별 요청 처리 시간
throughput	초당 요청 처리 횟수
CPU usage	서버의 CPU 사용률
error rate	요청 에러 확률

이때, 성능을 평가하기 위한 지표는 표 3에서 볼 수 있듯이, 클러스터 내 노드의 변경에 따라 여러 단계로 나누어져 있는 지갑 생성, ACA-Py 간 연결 생성, 자격증명 생성, 자격증명 발급 과정의 단계별 요청 시간을 측정한다. 또한, 초당 요청 처리 횟수를 측정하여 최적의 설정을 확인하고, 전체적인 ACA-Py 클러스터들이 어느 정도의 CPU를 사용하는지 확인한다. 마지막으로, ACA-Py 자체의 한계로 인해 클러스터 내 노드의 개수가 적을 때 많은 요청이 들어오게 된다면 작업을 처리하지 못해 에러가 발생하게 되는데, 어느 정도의 에러가 발생하는지 측정한다.

4.2. 성능 분석

표 4, 5, 6은 각각 클러스터 내의 ACA-Py 노드가 1개, 3개, 5개로 구성되었을 때의 실험 결과를 나타낸다. 지갑 생성, ACA-Py 노드 간의 연결 생성, 자격증명 발급, 자격증명 검증과정 내의 세부 단계들로 나누어 응답시간의 최소, 최대, 평균을 측정함과 동시에, 에러율까지 같이 측정하였다. 전체 단계들 중 자격증명 발급에서 항상 가장 긴 응답시간이 소요되었고, 노드가 1개인 표 4의 경우, 더 많은 노드가 있을 때는 발생하지 않던 에러도 꽤 높은 비율로 발생하는 것을 확인할 수 있다.

응답시간의 경우, 3가지 경우의 클러스터 모두 동시에 요청하는 사용자의 수가 늘어날수록 응답시간도 같이 증가하는 경향을 띠고 있는 것을 알 수 있었다.

그림 6에서 볼 수 있듯이, 오직 노드 1개로만 구성된 클러스터의 경우, 다른 클러스터들보다 훨씬 긴 응답시간이 필요했다. 또한, 노드 3개로 구성된

클러스터보다 노드 5개로 구성된 클러스터가 조금이지만 더욱 빠른 응답시간을 나타내는 것을 확인할 수 있었다. 노드 1개로 구성된 경우가 응답시간이 오래 걸리는 원인으로는 부하를 분산하지 못하는 점뿐 아니라, 표 4에서 에러율을 확인하면 더욱 쉽게 설명할 수 있다.

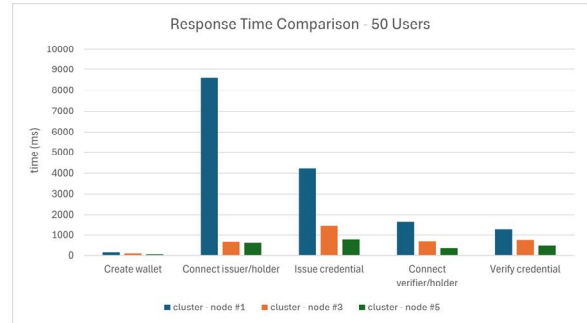


그림 6. 동시 사용자 50명 응답시간 비교

Fig. 6. Response time comparison for 50 users cases

노드 1개 클러스터의 경우 동시 사용자가 늘어날수록 에러율이 급격하게 상승하였다. 100 명이 동시에 요청하였을 때 특정 단계에서 98% 이상이 에러를 발생시킨 것을 확인할 수 있다. 그러나 노드가 3개 또는 5개인 클러스터들은 단 하나의 에러도 발생시키지 않고 요청을 문제없이 처리한 것을 확인할 수 있다. 이는 ACA-Py 노드 1개가 자체가 처리할 수 있는 한계점이 명백히 존재한다는 것을 의미한다.

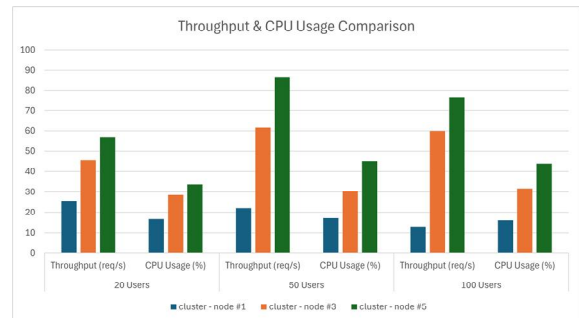


그림 7. 요청 처리량 & CPU 사용량 비교

Fig. 7. Throughput & CPU Usage Comparison

다음은 그림 7에서 클러스터별로 동시 사용자 수가 변화할 때, 전체 throughput과 서버의 CPU 사용량을 측정하였다. 3가지 클러스터 모두 throughput이 높을수록 CPU 사용량이 많기에, throughput과 CPU 사용량은 비슷한 경향을 띠고 있다는 것을 확인할 수 있다. 5개의 노드로 구성된 클러스터의 throughput 및 CPU 사용량이 가장 높았으며, 1개의 노드로 구성된 클러스터는 가장 낮았다. 이는 하나

의 ACA-Py가 처리할 수 있는 요청에 한계가 있다는 것을 뜻한다. 다만 3개, 5개 노드로 구성된 클러스터 모두 100명의 사용자를 동시에 처리할 때보다, 50명의 사용자를 동시에 처리할 때의 성능이 미세하게 더 높은 것을 확인할 수 있었다. 이는 요청의 분산이 적절히 이루어지지 못해 발생한 차이와 더불어, 50명의 사용자가 요청할 때 ACA-Py가 처리할 수 있는 적절한 수의 요청이 들어왔던 것으로 생각된다.

V. 결론 및 향후 연구

자기주권 신원증명의 패러다임은 자신의 신원정보를 직접 관리할 수 있으며, 원하는 정보만을 제공하는 등의 방법을 통해 보안성을 높이는 측면에서 굉장히 혁신적인 방법이다. 그러나, 아직 사용자 대부분이 특정 중앙 기관들이 직접 신원정보를 관리하는 중앙 집중적 방식을 사용하고 있는 이유는 서비스를 사용할 때의 편의성과 속도 때문이다. 많은 기존 서비스들이 구글 또는 네이버를 통한 간편 로그인을 지원하는 등 편의성 및 속도 측면에서 앞서고 있고, 따라서 자기주권 신원증명 방식도 이러한 격차를 줄이기 위한 노력이 중요하다.

본 연구에서는 현재 자기주권 신원증명을 구현하기 위해 주로 사용되는 오픈소스 프로젝트인 ACA-Py 노드의 실제 성능을 평가하였다. ACA-Py와 하이퍼레저 인디 블록체인을 통해 전체적인 시스템을 구현하였으며, ACA-Py의 노드들을 클러스터로 구성하여 클러스터의 구성을 다르게 하여 실제 성능을 평가하였다.

향후 연구에서는 동시 접속 사용자 수가 더 늘어나는 경우 시스템 운영을 위해 어느 정도 수의 노드가 필요한지 측정해볼 계획이다. 추가로, ACA-Py 기준에 사용했던 자격증명 발급 및 검증과정을 변경하여 ACA-Py 간의 연결을 생성하지 않아도 되는 connectionless 방법의 자격증명 발급 및 검증과정을 구현하여 기존 방법과의 성능을 비교해볼 예정이다. 또한, 본 연구에서는 모든 시스템의 구성 요소들이 하나의 서버에 위치하였는데, 발급자, 소유자, 검증자 역할별로 물리적으로 서버를 분리하여 전체적인 성능 변화를 측정해볼 예정이다.

References

[1] Tobin, A., & Reed, D. (2016). The inevitable rise

of self-sovereign identity. The Sovrin Foundation, 29(2016), 18.

- [2] Dieye, Mohameden, et al. "A self-sovereign identity based on zero-knowledge proof and blockchain." I EEE Access 11 (2023): 49445-49455.
- [3] Avellaneda, Oscar, et al. "Decentralized identity: W here did it come from and where is it going?." I EEE Communications Standards Magazine 3.4 (20 19): 10-13.
- [4] Introduction to Microsoft Entra Verified ID. Availa ble at "https://learn.microsoft.com/en-us/entra/verified -id/decentralized-identifier-overview", Accessed at: 2024-11-09
- [5] Initial. Available at "https://www.skntenterprise.com/pr oduct/detail/209", Accessed at: 2024-11-09
- [6] Decentralized Identifiers (DIDs) v1.0, Available at "https://www.w3.org/TR/did-core/", Accessed at: 20 24-11-09
- [7] Sovrin DID Method Specification. Available at "htt ps://sovrin-foundation.github.io/sovrin/spec/did-metho d-spec-template.html", Accessed at: 2024-11-09
- [8] SSW DID Method Specification. Available at "http s://sktston.github.io/ssw-did/did-method-spec.html", Accessed at: 2024-11-09
- [9] DID Method Lists. Available at "https://diddirectory. com/", Accessed at: 2024-11-09
- [10] ACA-Py - A Cloud Agent - Python Github. A vailable at "https://github.com/openwallet-foundation /acapy", Accessed at: 2024-11-09
- [11] Sedlmeir, Johannes, et al. "Digital identities and v erifiable credentials." Business & Information Syst ems Engineering 63.5 (2021): 603-613.
- [12] Sovrin Network Homepage. Available at "https://sov rin.org/overview/", Accessed at: 2024-11-09
- [13] Findy Agency Homepage. Availabe at "https://findy- network.github.io/", Accessed at: 2024-11-09
- [14] IDUnion Homepage. Available at "https://idunion.or g/", Accessed at: 2024-11-09
- [15] IdRamp: Identity Verification Orchestration Hompa ge. Available at "https://idramp.com/", Accessed a t: 2024-11-09
- [16] Verifiable Credentials Data Model v1.1. Available at "https://www.w3.org/TR/vc-data-model/", Access ed at: 2024-11-09
- [17] Ferdous, Md Sadek, Farida Chowdhury, and Madin

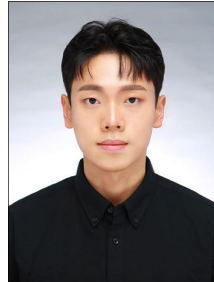
- i O. Alassafi. "In search of self-sovereign identity leveraging blockchain technology." IEEE access 7 (2019): 103059-103079.
- [18] Siqueira, Alexandre, Arlindo F. Da Conceição, and Vladimir Rocha. "Performance Evaluation of Self-Sovereign Identity Use Cases." 2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). IEEE, 2023.
- [19] Bhattacharya, Manas Pratim, Pavol Zavarsky, and Sergey Butakov. "Enhancing the security and privacy of self-sovereign identities on hyperledger indy blockchain." 2020 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2020.
- [20] BC Gov - Von Network Github. Available at "https://github.com/bcgov/von-network", Accessed at: 2024-11-09
- [21] Aries RFC 0023: DID Exchange Protocol 1.0. Available at "https://github.com/hyperledger/aries-rfcs/blob/main/features/0023-did-exchange/README.md", Accessed at: 2024-11-09
- [22] Aries RFC 0160: Connection Protocol. Available at "https://github.com/hyperledger/aries-rfcs/blob/main/features/0160-connection-protocol/README.md", Accessed at: 2024-11-09
- [23] Aries RFC 0453: Issue Credential Protocol 2.0. Available at "https://github.com/hyperledger/aries-rfcs/blob/main/features/0160-connection-protocol/README.md", Accessed at: 2024-11-09
- [24] Aries RFC 0454: Present Proof Protocol 2.0. Available at "https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md", Accessed at: 2024-11-09
- [25] Postman Homepage. Available at "https://www.postman.com/", Accessed at: 2024-11-10
- [26] Mühle, Alexander, et al. "A survey on essential components of a self-sovereign identity." Computer Science Review 30 (2018): 80-86.
- [27] Soltani, Reza, Uyen Trang Nguyen, and Aijun An. "A Survey of Self Sovereign Identity Ecosystem." Security and Communication Networks 2021.1 (2021): 8873429.
- [28] Jolocom. Available at "https://stories.jolocom.com/", Accessed at: 2024-11-14
- [29] Benet, Juan. "Ipfes-content addressed, versioned, p2p file system." arXiv preprint arXiv:1407.3561 (2014).
- [30] SelfKey. Available at "https://selfkey.org/", Accessed at: 2024-11-14
- [31] LifeID. Available at "https://lifeid.io/", Accessed at: 2024-11-14

김 정 현 (Jeongheon Kim)



2020~현재 포항공과대학교 컴퓨터공학과 통합과정
2017 포항공과대학교 IT융합공학
<관심분야> 블록체인 및 암호화폐, 머신러닝, 딥러닝

강 창 훈 (Changhoon Kang)



2019~현재 포항공과대학교 컴퓨터공학과 통합과정
2015 포항공과대학교 컴퓨터공학과 학사
<관심분야> 블록체인 및 암호화폐, 딥러닝

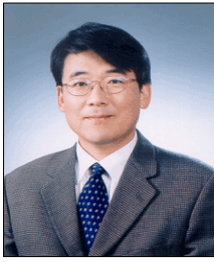
우 종 수 (Jongsoo Woo)



2020~현재 포항공과대학교 컴퓨터공학과 연구교수
2020~현재 포스텍 크립토틀록체인연구센터 공동센터장
2018~현재 DGIST 이사장
2016~2019 포스코교육재단 이사장
2014~2016 산업과학기술연구원 원장
2011~2014 포스코 기술연구원 원장

<관심분야> 암호화폐, 블록체인

홍 원 기 (James Won-Ki Hong)



2020~현재 포스텍 크립토블록
체인연구센터 공동센터장

2012~2014 KT 종합기술원,
원장

2008~2012 포항공과대학교, 정
보전자융합공학부장

2008~2010 포항공과대학교, 컴
퓨터공학과 주임교수

2007~2010 포항공과대학교, 정보통신연구소 연구소
장

2007~2011 포항공과대학교, 정보통신대학원장

1995~현재 포항공과대학교, 컴퓨터공학과 교수

1992~1995 Univ. of Western Ontario, 연구교수

<관심분야> 네트워크 트래픽 모니터링, 네트워크
및 시스템 관리, SDN/NFV, IoT, 무크기반 교육,
블록체인 및 암호화폐, Vmeeting 화상회의시스템

ZSM/ENI 기반의 자율 네트워크 및 서비스 관리

양 우 석*, 이 재 오°

Autonomous Network and Service Management based on ZSM/ENI

Wooseok Yang*, Jaehoh Lee°

요 약

네트워크 및 서비스 운영자는 이질적이고 다양한 네트워크 장치들 간의 복잡한 상호작용 및 동적으로 변화하는 사용자 요구 사항, 비즈니스 목표 및 환경에 적용할 수 있는 지능화된(intelligent) 네트워크 및 서비스 관리에 관심을 가지고 있다. 네트워크 관리는 자체 구성 네트워크(Self Organizing Network: SON) 기반의 동적인 서비스 전개와 연관되어 자율(autonomous) 네트워크 및 서비스 관리의 연구 필요성이 대두되고 있다. 자율 관리는 다양한 AI/ML 기술을 융합 및 적용하고 있다. 따라서 이러한 AI/ML 기술을 적용할 수 있는 표준화된 자율 네트워크 및 서비스 관리 구조와 모델을 연구하는 노력도 진행 중이다.

본 논문에서는 자율 네트워크 및 서비스 관리를 위하여 사용 될 수 있는 관련 AI/ML 기술을 소개한다. 또한 자율 네트워크 및 서비스 관리 표준안을 제안하고 있는 ETSI(European Telecommunications Standards Institute)의 ZSM(Zero-touch Network and Service Management)과 ENI(Experiential Networked Intelligence) 시스템 구조에 관한 연구 동향을 살펴보고자 한다.

Key Words : Autonomous Network and Service Management, Machine Learning, Closed Loop, ZSM, ENI

ABSTRACT

Network and service operators are interested in intelligent network and service management that can be applied to complex interactions between heterogeneous and diverse network devices, dynamically changing user requirements, business goals, and environments. Network management is related to dynamic service deployment based on Self-Organizing Network (SON), and the need for autonomous network and service management is emerging for study. Autonomous management integrates and applies various AI/ML technologies. Therefore, efforts are also underway to study standardized autonomous network and service management structures and models that can use these AI/ML technologies.

In this paper, we introduce some related AI/ML technologies for autonomous network and service management. We also examine research trends on the Zero-touch Network and Service Management (ZSM) and Experiential Networked Intelligence (ENI) system architecture of the European Telecommunications Standards Institute (ETSI), which are proposing standards for autonomous network and service management.

* 이 논문은 2024년도 한국기술교육대학교 교수 교육연구진흥과제 지원에 의하여 연구되었음.

♦ First Author : Korea University of Technology and Education Department of Information Communication, oos13@koreatech.ac.kr

° Corresponding Author : Korea University of Technology and Education Department of Information Communication, jolee@koreatech.ac.kr

I. 서 론

미래 통신 및 네트워크 분야에서 인공지능/기계학습(Artificial Intelligence/Machine Learning: AI/ML) 기술은 다양한 방법으로 융합되어 중요한 역할을 하리라고 본다. 기계 학습은 복잡한 시스템의 인식 및 의사 결정 문제에 대한 솔루션을 제공하는 중요한 방법이다. 네트워크 기반의 컴퓨팅 기술과 관련된 기계 학습 기술은 네트워크 관리, 네트워크 슬라이싱(slicing), 적응성(adaptive) 자원 할당, 이동성 모델링, QoS, 보안 및 에너지 효율성을 자율화하는데 필요한 기술로서 대두되고 있다[1, 10]. 기계 학습을 적용할 수 있는 표준화된 기술의 조사 연구 및 동향은 향후 네트워크 및 네트워크 관리 기술 발전 방향을 예측하는데 도움이 된다.

자동(automation)은 사람의 개입 없이 자율적으로 작업을 실행하는 것을 말한다. 자율은 비즈니스 프로세스, 네트워크 관리, 자원 할당, 라우팅 프로토콜, 자원 최적화 등 다양한 애플리케이션에 적용될 수 있다[1, 2]. 서비스 기반 아키텍처, 인공지능 기반 기술, 프로그래밍 가능한 네트워크 및 소프트웨어 컴퓨팅 전략을 통해 자동화를 달성할 수 있다. 자율화 적용의 주요 목표는 서비스에 대한 확장성, 유연성 및 민첩성을 확보하여 다양한 주요 문제의 해결을 위한 최첨단 솔루션을 달성하는 것이다.

본 논문에서는 통신 및 네트워크 관리 분야에서 적용될 수 있는 기계 학습 방안에 관하여 조사 및 연구함으로써 미래의 네트워크 및 서비스 관리를 자율화할 수 있는 대표적인 표준 구조의 기술 내용 및 동향을 연구하려고 한다. 따라서 관련 적용 가능한 AI 기술 연구로서 기계 학습, 심층 강화 학습(deep reinforcement learning), 연합 학습(federated learning), 전이 학습(transfer learning)을 간략하게 살펴본다. 그리고 대표적인 자율 네트워크 및 서비스 관리 참조 아키텍처 및 시스템인 ETSI의 ZSM(Zero-touch Network and Service Management)과 ENI(Experiential Networked Intelligence)를 기술하려고 한다.

논문 구성은 ZSM 및 ENI 구조에서 적용 가능한 AI/ML 관련 기술로서 기계 학습 개요, 심층 강화 학습, 관리 도메인(management domain) 간의 협력 모델에 대하여 살펴본다. 그리고 대표적인 표준으로서 자율 네트워크 및 서비스 관리 구조인 ZSM 및 ENI에 관하여 기술한다. 마지막으로 ZSM/ENI 기반의 전형적인 자율 관리 시스템을 살펴보고자 한다.

II. 관련 기술 연구

본 절에서는 자율 네트워크 및 서비스 관리 분야에 적용 가능한 대표적인 AI/ML 기술을 살펴본다.

2.1. 기계 학습

기계 학습은 인공지능의 하위 집합으로 경험을 통해 자동으로 개선되는 컴퓨터 알고리즘에 대한 연구이다. 기계 학습은 데이터에서 학습하고 데이터에 대한 예측을 수행할 수 있는 알고리즘의 연구 및 구성을 탐구한다. 더 많은 데이터를 기반으로 기계 학습은 작업과 응답을 변경하여 더 효율적이고 적응 가능한 형태로 확장할 수 있다. 심층 학습은 의사 결정을 달성하기 위해 훈련 데이터에 인공 신경망을 사용한다. 신경망은 여러 계층에서 계산을 통해 미시적 계산을 수행하고 인간과 같은 작업을 처리할 수 있다[1, 3].

기계 학습은 지도(supervised) 학습, 비지도(unsupervised) 학습, 강화 학습으로 나뉜다[4]. 지도 학습 모델에서 알고리즘은 레이블이 지정된 데이터 세트에서 학습하여, 새 데이터에 대한 예상 및 예측된 응답을 생성한다. 지도 학습의 목표는 훈련(training) 데이터를 기반으로 입력과 출력 간의 매핑을 찾는 것이다. 이를 이용하여 훈련 중에는 나타나지 않는 새로운 입력에 대한 정확한 예측을 설정하고 수행한다. 지도 학습은 주로 분류(classification) 및 회귀(regression) 문제에 사용된다.

비지도 학습 모델에서 알고리즘은 레이블이 지정되지 않은 데이터 세트에서 학습하고 자체적으로 기능, 동시 발생 및 기본 패턴을 추출하여 의미를 찾으려고 한다. 훈련 데이터는 어떤 출력 값도 포함하지 않으며 학습자는 환경에서 피드백을 받지 못한다. 이는 주로 클러스터링 및 연관(association) 문제에 사용된다. 비지도 학습 유형은 클러스터링(clustering), 이상 감지(abnormal detection), 연관, 오토 인코더(auto encoder), 잠재 변수 모델(Latent Variable Model), 신경망(Neural Network) 등이 있다[1, 2].

강화 학습은 모델이 일부 작업을 수행하고 반응을 분석하여 환경에서 행동하는 방법을 배우는 일종의 기계 학습이다. 강화 학습은 특정 상황에서 긍정적인 반응을 극대화하기 위해 적절한 조치를 취한다. 강화 모델은 주어진 작업을 수행하기 위해 취해야 할 조치를 결정하므로 경험 자체에서 배워야 한다. 학습자 에이전트는 특정 작업을 실행하는 상태 정보와 함께 누적 보상(reward)의 다른 개념을 최대화하려고 노력한다

다. 학습자는 수행된 결정에 대한 피드백을 받지만 훈련 데이터는 외부 감독자에 의해 레이블이 지정되지 않는다. 결과적으로 학습자는 응답이 옳았거나 잘못되었는지 알 수 없다. 대신 반복적인 경험에서 배우고 자체적으로 최상의 정책(policy)을 발견해야 한다. 따라서 강화 학습은 보상과 처벌(punishment)에 의해 구동되는 적응성 의사 결정 문제를 모델링하기 위한 정성적 및 정량적 프레임워크를 제공한다[1, 3]. 강화 학습은 ZSM 프레임워크 아키텍처 내에서 네트워크 및 서비스 자동화를 위한 유용한 지원 도구가 될 수 있다[22]. 그림 1은 기계 학습 분류의 형태를 보여준다.

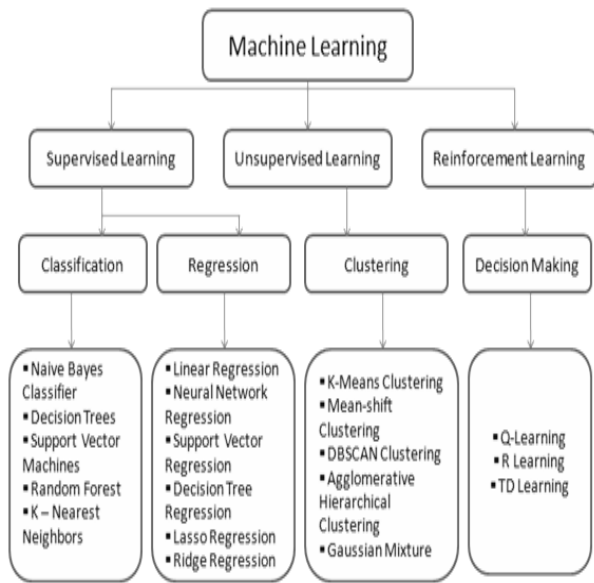


그림 1. 기계 학습 분류
Fig. 1. Classification of Machine Learning

2.2 심층 강화 학습

심층 강화 학습은 심층 학습과 강화 학습을 결합한 기계 학습의 한 분야이다. 복잡한 시스템의 인식 및 의사 결정 문제에 대한 솔루션을 제공하기 위해 심층 학습과 강화 학습의 장점을 모두 가지고 있다[1, 2, 4, 11]. 강화 학습은 다양한 알고리즘들이 존재하며, 이들을 분류하는 방법 또한 다양하다. 환경 모델은 환경의 행동을 모방하거나 보다 일반적으로 환경이 어떻게 행동할 지에 대한 추론을 할 수 있게 해주는 것이다. 예를 들어, 상태(state)와 행동(action)이 주어졌을 때 모델은 결과적인 다음 상태와 다음 보상을 예측할 수 있다. 모델은 계획(planning)에 사용되며, 이는 실제로 경험하기 전에 가능한 미래 상황을 고려하여 행동 과정을 결정하는 모든 방법을 의미한다. 모델과 계

획을 사용하여 강화 학습 문제를 해결하는 방법을 모델 기반(model-based) 방법이라고 하며, 모델 없는(model-free) 방법은 계획이 거의 없는 것으로 간주되어 명시적으로 시행착오(trial-and-error) 기반으로 학습하는 방식이다[4]. 강화 학습의 분류 방법 중 하나는 그림 2와 같다[2].

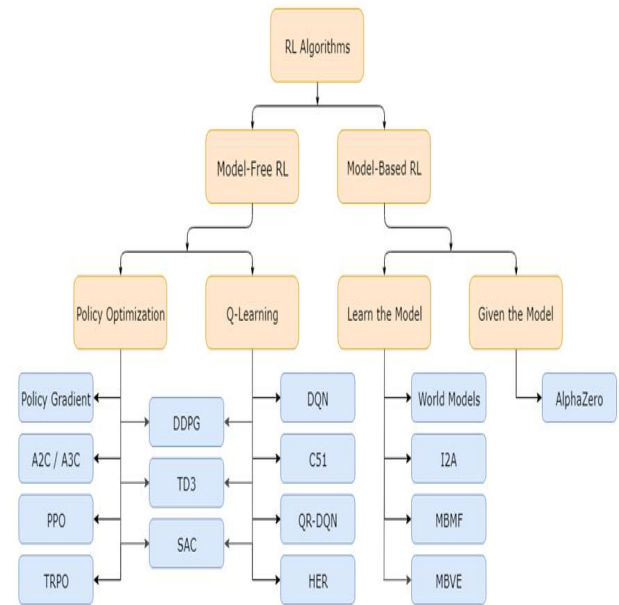


그림 2. 강화 학습 분류
Fig. 2. Classification of Reinforcement Learning

강화 학습은 가치 기반(value-based) 방법과 정책 기반(policy-based) 방법으로 나눌 수 있다. 마찬가지로 심층 강화 학습도 같은 방식으로 나눌 수 있다. 가치 기반 심층 강화 학습은 주어진 행동 또는 주어진 상태가 에이전트에게 얼마나 좋은지를 의미한다. 보상 측면에서 주어진 상태에서 행동들의 선택에 따라서 에이전트의 수익이 증가(또는 감소)할 수 있다. 이를 미리 알면 에이전트가 예상 수익을 최대화하기 위해 어떤 상태에서 어떤 행동을 취해야 할지 결정하는 데 도움이 될 수 있습니다. 가치 기반의 대표적인 알고리즘인 심층 Q-러닝 네트워크(DQN: Deep Q-learning Network)는 정책을 훈련하기 위한 기능 엔지니어링을 피하기 위해 DNN을 사용하여 큰 상태 공간이 있는 상황을 모델링하고 이전 단계에서 학습한 정보를 계속 사용하기 위해 재생 메모리 세트를 사용할 수 있다. DQN은 특히 5G/6G와 같이 동적이며 유연성(flexibility)을 가져야 하는 ad-hoc 네트워크의 최적화 정책을 발견하는데 적용될 수 있다[1, 8, 9].

정책 기반 심층 강화 학습은 에이전트가 주어진 상태에서 주어진 행동을 취할 가능성이 얼마나 되는지,

즉 에이전트가 특정 상태에서 특정 행동을 선택할 확률을 의미한다. 정책 기반의 대표적인 방식으로 AC(Actor-Critic)가 제안되었다[8, 11, 13]. AC는 정책 기반 방법과 가치 기반 방법의 장점을 결합한 알고리즘이다. Actor는 예상 보상을 최대화하기 위해 행동을 선택하는 방법을 학습하는 정책 기반 에이전트이며, 상태를 행동 확률에 매핑하는 신경망에 의해 매개변수화 된다. Critic은 주어진 상태에서 예상되는 미래 보상을 예측하는 가치 함수를 추정하는 방법을 학습하는 가치 기반 에이전트로, 신경망에 의해 매개변수화 된다. TD(Temporal Difference) 에러는 상태 가치 함수의 추정 값과 실제 얻은 보상 간의 차이를 의미한다. 이는 예상했던 보상과 실제 얻은 보상의 차이라고 할 수 있다. 그림 3은 AC 모델을 나타낸다.

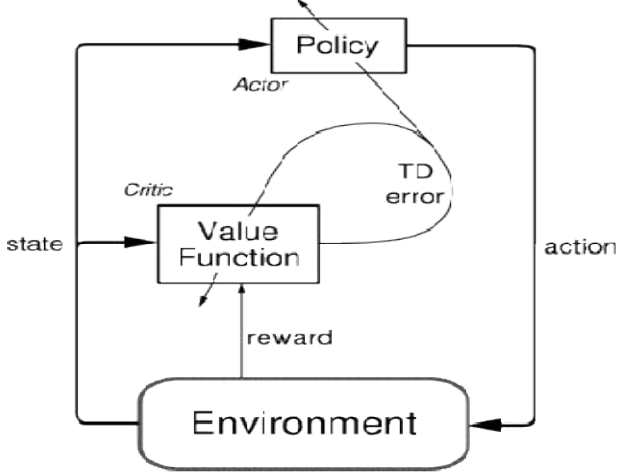


그림 3. Actor-Critic 모델
Fig. 3. Actor-Critic Model

A3C(Asynchronous, Advantage, Actor-Critic)는 단일 신경망으로 대표되는 단일 에이전트가 단일 환경과 상호 작용하는 DQN과 달리 보다 효율적으로 학습한다. A3C에는 전역(global) 네트워크와 각각 고유한 네트워크 매개변수 세트가 있는 여러 작업자 에이전트가 있다. 이것이 단일 에이전트를 사용하는 것보다 더 잘 작동하는 이유(더 많은 작업을 완료하는 속도가 빨라지는 것 이상으로)는 각 에이전트의 경험이 다른 에이전트의 경험과 독립적이기 때문이다. 이러한 방식으로 훈련에 사용할 수 있는 전반적인 경험이 더욱 다양해진다.

2.3 관리 도메인들 사이의 AI/ML 모델

표준 기계 학습 접근 방식을 사용하려면 학습 데이터를 한 기계 또는 데이터 센터에 중앙 집중화해야 한다. 그러나 통신 분야의 데이터는 일반적으로 여러

장소(locale)에 있는 고도로 분산된 네트워크를 특징으로 한다. 따라서 네트워크의 다양한 위치에서 공통 처리 위치로 데이터를 가져오는 것은 전송 자원을 소모하기 때문에 비용이 많이 든다. AI/ML 능력을 가진 관리 도메인들 사이에 공통의 문제를 해결하기 위하여 협력이 필요하다. AI/ML 능력을 가진 도메인 관리 시스템들 사이에 협력이 필요한 경우에 사용될 수 있는 대표적인 모델로서 연합 학습(federated learning)과 전이 학습(transfer learning)이 있다.

연합 학습은 학습 능력을 소유한 클라이언트 역할을 하는 다수의 관리 도메인 시스템 응용이 각각 학습하여, 중앙에 있는 관리 시스템의 전역 모델을 학습하는 기술이다[5]. 따라서 이 접근 방식은 관리 도메인의 지역 데이터를 안전하게 유지해야 할 때 사용된다. 연합 학습은 여러 도메인에서 AI/ML 관리 시스템 응용을 훈련한 다음 중앙 위치에서 집계하여 처리한다[21]. 모델 매개변수 및 훈련 정보와 같은 AI/ML 특정 정보 교환을 지원하면 운영자가 연합 학습을 기반으로 AI/ML 솔루션을 배포하고 활용할 수 있다.

전이 학습은 네트워크의 상이한 장소나 도메인 그리고 다소 유사한 상황에서 주어진 새로운 지역 상황으로 지식을 전송할 수 있게 해주기 때문에 지역 데이터의 중앙 집중화 문제를 해결할 수 있다[8, 9, 10]. 전이 학습 방법을 사용하면 사전 훈련된 AI/ML 관리 시스템 응용을 관리 도메인별 AI/ML 관리 시스템 응용을 얻기 위한 시작점으로 사용할 수 있다. 이 방법은 필요한 훈련 시간과 계산 리소스를 줄여 AI 관리 시스템 응용의 신속한 배포를 용이하게 한다[16, 21]. 따라서 전이 학습은 네트워크 및 서비스 자동화를 위한 핵심 지원 요소로, 특히 효율성을 높이고 계획, 배포 및 최적화 작업을 지원한다[22]. 일반적인 통신 연구 분야에서 전이 학습은 현재 초기 단계이지만 6G에서 훨씬 더 중요한 역할을 할 것이라고 보고 있다.

III. ZSM 및 ENI 기반 네트워크 및 서비스 관리

네트워크 및 서비스 운영자는 동적이며 복잡한 통신 및 네트워크 장치들의 문맥적 변화(contextual changes)의 감지 및 어떤 장치 및 서비스 그룹이 서로 영향을 미치는지 확인하여 서비스 수준 계약(Service Level Agreement: SLA)을 유지할 수 있는 지능화된 네트워크 및 서비스 관리를 요구하고 있다[11].

3.1. ZSM 아키텍처

ETSI에 따르면 ZSM 아키텍처는 네트워크 자율화와 관련된 디지털 서비스에 필요한 확장성, 유연성 및 민첩성을 제공하도록 설계되었다. 또한 아키텍처는 본질적으로 적응력이 있어 자율화된 기능이 새로운 기술에 대한 규정 준수 표준을 충족할 것으로 예상된다 [8, 9]. 목표는 자동화의 가장 높은 추상화를 달성하여 적응성, 인식 및 자동 특성 또는 기능의 조합을 통해 독립적으로 생각하고 행동할 수 있는 능력을 가진 자율 시스템을 개발하는 것이다 [12, 16]. 그림 4는 ZSM 프레임워크 참조 아키텍처이다 [16].

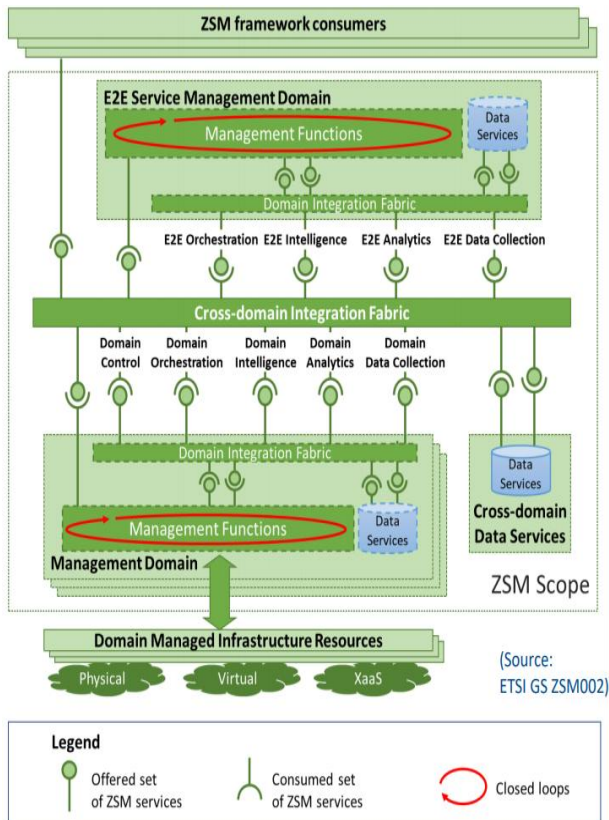


그림 4. ZSM 프레임워크 참조 아키텍처
Fig 4. ZSM Framework Reference Architecture

ZSM 프레임워크는 관리 도메인으로 조직된 분산 관리 및 데이터 서비스로 구성되며, 통합 패브릭 (Integration Fabric)을 통해 합쳐진다. 모든 관리 도메인 및 종단 간(E2E) 서비스 관리 도메인은 서비스 종단의 집합을 노출 및/또는 소비하는 관리 기능 (Management Function)을 통해 일련의 ZSM 서비스 기능을 제공한다 [13, 14, 16, 17, 18]. 통합 패브릭은 관리 서비스의 등록/해제, 검색 및 호출 지원 수단과 관리 기능 간의 통신을 포함하여 관리 도메인 내부 및 외부에서 관리 기능 간의 상호 운용 및 통신을 가

능하게 한다. 제공되는 서비스 패턴은 요청-응답과 게시/구독 모델이 지원된다. 교차 도메인 간 통합 패브릭(Cross-domain Integration Fabric)은 기능 제공 및 종단 교차 도메인 접근을 용이하게 한다. 일부 서비스는 관리 도메인 내에서 로컬로만 제공되고 사용된다. 모든 관리 서비스의 논리 그룹은 관련된 기능을 갖는 서비스들을 포함한다 [18].

데이터 서비스(Data Service)는 관리 도메인 내 혹은 전체에서 관리 서비스 전반에 걸쳐 ZSM 서비스를 사용할 수 있는 인증된 ZSM 프레임워크 소비자 (Framework Consumer)가 공유 관리 데이터 접근 및 지속적으로 일관된 수단을 제공한다. 관리 도메인은 자원에서 제공되는 (다양한) 인터페이스를 통해 관리 대상인 인프라 자원(infrastructure resource)을 관리한다. E2E 서비스 관리 도메인(Service Management Domain)은 다양한 관리 도메인에 걸쳐 있는 종단 간 서비스를 관리하고 오케스트레이션을 사용하여 관리 도메인 간을 조정한다 [16].

3.1.1 관리 서비스(Management Service)

본 절에서는 ZSM 프레임워크에서 정의된 기본 관리 도메인 및 E2E 서비스 관리 도메인에서 정의된 서비스들에 관하여 살펴본다 [16].

3.1.1.1 데이터 수집(Data Collection)

관리 도메인 및 종단 간 관리 도메인에서 서비스 관련 장애(fault), 성능과 같은 데이터 수집을 담당한다.

3.1.1.2 분석(Analytic)

다양한 분석 서비스는 기본 관리 도메인 및 기타 입력에서 들어오는 사건(event), 장애 및 성능과 관련된 정보를 처리하여 집계 및 분석하고 파생한다. 또한 분석 서비스에는 제공된 서비스에 고유한 종단 간 서비스 KPI((Key Performance Indicator) 및 사용자 경험(예: 비디오 정지 또는 프레임 제거)을 추적하는 기능이 있을 수 있다. 이는 기본 관리 도메인의 서비스별 관찰 체계 또는 조사를 기반으로 작동한다. 분석 서비스는 분석 결과를 제공하는 방법과 시기를 사용자가 정의하도록 구성할 수도 있다. 예를 들어 통찰이 생성되는 방식과 사용되는 알고리즘을 모두 구성할 수 있다.

실제 특정 분석 서비스는 일반 분석 서비스와 동일한 기능 세트를 갖지만 생성하는 분석 결과 유형, 수행하는 분석 및 실제 구성이 다르다. 다음은 ZSM에

서 정의한 특정 분석 서비스를 나타낸다.

- E2E 이상 탐지 서비스(E2E anomaly detection service): 이상 탐지 서비스는 기본 관리 도메인의 도메인 데이터 수집 서비스에서 제공하는 정보를 사용하여 E2E 서비스와 관련된 이상 상태를 감지하는 기능을 제공한다. 예를 들면 이러한 조건은 보안 위반/이상, 서비스 수준 사양(Service Level Specification: SLS) 위반이 감지되지 않더라도 E2E 서비스의 국지적 또는 단기적 사용 불가능 또는 서비스의 불일치로 이어지는 결함 조건일 수 있다. 이상 또는 비정상적인 동작이 감지되면 그 결과는 E2E 서비스의 지능 서비스 생산자가 적절한 결정을 내리도록 할 수 있다. 해결책이나 해결 방법을 찾거나 추가 분석을 위해 다른 분석 서비스에서 데이터를 사용할 수 있도록 하거나 도메인 간 통합 패브릭을 통해 ZSM 프레임워크 소비자에게 정보를 노출할 수 있다.
- 배포된 AI 모델 성능 평가 서비스(Deployed AI model performance evaluation service): 배포된 AI 모델 성능 평가 서비스를 통해 AI 모델의 성능 수준을 파악하고 탐지할 수 있다. 성능 저하(AI 모델은 시간이 지남에 따라 성능 저하를 겪을 수 있고 재교육이 필요한 확률 시스템), 갑작스러운 성능 저하, 불규칙한 성능 등을 배포된 AI 모델 평가 서비스에서 사용할 수 있다. 배포된 AI 모델에서 수행할 가장 적절한 작업을 결정한다.
- 근본 원인 분석 서비스(Root cause analysis service): 여러 오류에 대한 정보의 연관성 및 종속성을 식별하고 추가 통찰력 또는 지식을 사용하여 문제의 근본 원인을 결정한다.
- 영향 분석 서비스(Impact analysis service): 장애 또는 기타 문제(예: 성능 문제)가 실제 관리 서비스 성능 또는 가용성에 미치는 영향을 결정한다.
- 성능 분석 서비스(Performance analysis service): 기본 관리 도메인의 데이터 수집 서비스에서 수집한 데이터를 처리하여 높은 수준의 서비스 성능 통찰력을 수행한다. 과잉 밀집(congestion) 및 과부하 상황과 같은 서비스 저하와 관련된 KPI 및 정보이다.
- 보안 분석 서비스(Security analytics service): 보안에 영향을 줄 수 있는 특정 공격 패턴, 위협 및 의심스러운 동작을 식별하기 위해 보안 사건을 처리한다.

E2E 서비스 품질 관리 서비스를 통해 인증된 소비

자는 E2E 서비스의 서비스 수준 목표(Service Level Objective: SLO) 및 SLS를 관리하고 위반 사항을 추적할 수 있다. SLS 및 SLO는 SLA와 관련이 있다 [17]. SLS는 SLO로 구성되며 제공할 SLA에서 적용되는 서비스 품질을 정의하는 기술 매개변수를 제공한다.

3.1.1.3 서비스 오케스트레이션(Orchestration)

E2E 서비스 오케스트레이션은 도메인 간 고객 대면 서비스를 생성/수정/삭제하기 위해 여러 관리 도메인의 카탈로그(catalogue) 기반 E2E 오케스트레이션을 담당한다. 서비스 모델은 서비스의 다양한 부분이 함께 연결되고 관리 도메인에 매핑 되는 방식을 정의한다. E2E 서비스 오케스트레이션 서비스를 통해 인증된 소비자는 E2E 서비스의 생성, 수정 및 종료에 포함하여 네트워크 슬라이스와 같은 E2E 서비스를 인스턴스화하고 유지 관리할 수 있으며 해당 워크플로우를 자동화할 수 있다[12, 14]. 이 서비스는 E2E 서비스 관리 도메인에서 관리 서비스 카탈로그 관리 서비스에 의해 관리되는 E2E 서비스 모델에 적용된다. E2E 서비스 모델은 관리 도메인 및 해당 상호 연결에서 제공하는 서비스 부분 설명에 대한 참조를 포함하여 전체 E2E 서비스를 설명한다. ZSM 프레임워크 소비자는 E2E 서비스 오케스트레이션 서비스를 사용하여 E2E 서비스를 배포할 수 있습니다.

타당성 확인(Feasibility check) 서비스는 특정 매개변수화된 E2E 서비스(예: 네트워크 슬라이스)가 제안된 서비스 수준에서 배포 가능한지 확인하는 데 사용된다. 관리되는 서비스(Managed Service) 카탈로그 관리 서비스는 E2E 서비스 관리 도메인에서 관리하는 사용 가능한 E2E 서비스를 나타내는 서비스 모델의 카탈로그를 관리하고, 이 카탈로그를 ZSM 프레임워크 소비자에게 노출하는 일을 담당한다. E2E 서비스 모델은 지원되는 커버리지 영역, 관련 SLS, 서비스 템플릿 등과 같은 지원 정보를 포함할 수 있으며, 서비스 범주로 그룹화 될 수 있다. 서비스 모델에는 일반적으로 수명 주기 단계를 나타내는 상태가 있다. E2E 테스트 서비스(Testing service)를 사용하면 관련 없는 다른 네트워크 서비스에 영향을 주지 않고 중단 간 서비스 및/또는 그 일부의 능동(active) 또는 수동(passive) 테스트를 실행하여 장애 위치를 찾을 수 있다. ZSM 프레임워크 소비자에 의해 또는 자율화된 절차의 일부로 사용될 수 있다. E2E 서비스 보유 관리 서비스(E2E service inventory management service)는 사용 가능한(인스턴스화 및 활성화) E2E

서비스에 대한 정보를 유지 및 질의할 수 있는 E2E 서비스 관리 도메인 내부의 소비자에게만 제공된다. E2E 서비스 토폴로지 정보 서비스(E2E services topology information service)는 다양한 추상화 수준에서 E2E 서비스 관점의 인프라에 대한 토폴로지 정보를 제공한다.

3.1.1.4 서비스 지능(Intelligence)

E2E 서비스 지능 서비스는 최종 목표 단계인 완전 자율 관리를 통해 다양한 수준의 자동화된 의사 결정 및 인적 감독을 지원하여 E2E 서비스 관리 도메인에서 지능형 폐쇄 루프 자동화를 주도한다. 서비스 지능의 범위는 의사 결정 지원(decision support), 의사 결정(decision making), 행동 계획(action planning) 등이다. 의사 결정 지원 서비스는 AI/ML 및 지식 관리와 같은 기술을 통해 의사 결정을 가능하게 한다. E2E 서비스 관리 의사 결정은 E2E 서비스 데이터 수집, E2E 서비스 분석, E2E 서비스 관리 도메인의 도메인 데이터 서비스, 도메인 간 데이터 서비스 및 적용 가능한 기타 소스의 일부로 정의된 ZSM 서비스에서 제공하는 정보를 기반으로 한다. 후속 작업 계획은 E2E 서비스 오케스트레이션의 일부로 정의된 ZSM 서비스에서 실행할 오케스트레이션/제어 작업을 정의한다.

제공되는 서비스는 AI 모델 관리 서비스(AI Model Management Service), 배포된 AI 모델 평가 서비스(Deployed AI Model Assessment Service), AI 훈련 데이터 관리 서비스(AI Training Data Management Service), E2E 서비스 상태 문제 보고 서비스(E2E Service Health Issue Reporting Service)이다.

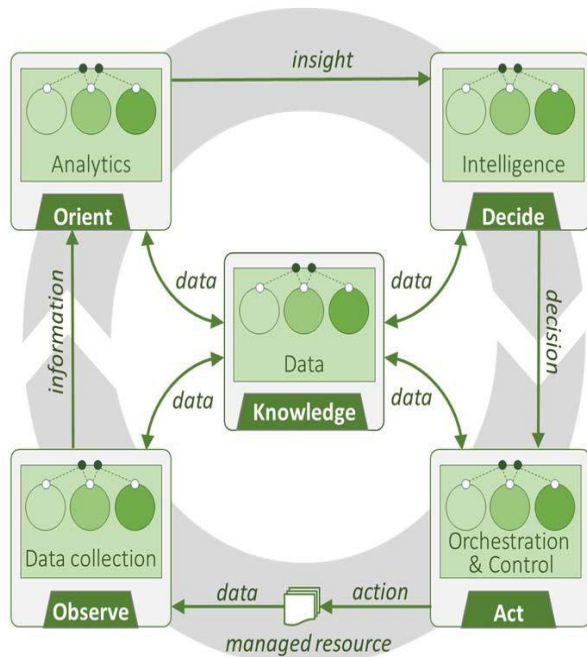
AI 모델 관리 서비스를 통해 AI 모델을 관리할 수 있다. 새로운 입력 데이터를 기반으로 주기적 또는 언제든지 AI 모델을 수정할 수 있다. 배포된 AI 모델의 성능은 현실의 변화로 인해 시간이 지남에 따라 저하될 수 있으며, 그 결과 모델이 현실과 정확하게 일치하지 않아 모델 수정 또는 교체가 필요할 수 있다. 배포된 AI 모델 평가 서비스를 사용하면 사용 가능한 모델 성능 정보를 기반으로 관리 기능에서 배포 및 사용하는 AI 모델에서 실행할 가장 적절한 조치를 결정할 수 있다. 이를 위해 배포된 AI 평가 서비스는 배포된 AI 모델 성능 평가 서비스의 출력을 사용할 수 있다. 배포된 AI 모델 평가 서비스에서 파생된 평가는 재교육, 재구성, 업그레이드, 교체, 일시 중지, 종료와 같이 배포된 AI 모델에서 수행할 가장 적절한 작업을 결정하는 데 사용된다. AI 훈련 데이터 관리 서비스를 통해 AI 모델을 훈련하거나 재훈련하는 데

필요한 훈련 데이터를 관리할 수 있다. 훈련 데이터는 분류, 예측 또는 기타 기계 학습 문제의 입력 특징/속성 및 실측 결과를 포함하는 레이블이 지정된 데이터일 수 있다. 학습 데이터는 학습이 데이터에 의존하는 특정 AI 모델을 의미할 수도 있고, 향후 AI 모델 개발을 위해 기존 AI 모델과 독립적으로 학습 데이터를 축적할 수도 있다. 자동화 시스템에서도 모든 서비스 상태 문제를 자동으로 복구할 수 있는 것은 아니다. 특정 문제의 경우 상위 실체(예: ZSM 프레임워크 소비자)에 보고해야 한다. 이 서비스를 통해 이러한 문제 보고서를 관리할 수 있다. E2E 서비스 상태 문제 보고 서비스는 비정상적인 서비스 상태(예: 결함, 보안 문제 등으로 인한)에 대한 정보를 제공하고 해당 정보를 사용할 수 있는 경우 이를 기본 관리 도메인의 문제와 연관시킬 수 있다. 또한 상태 문제와 심각성을 추적 및 수정하고 제공되는 정보를 구성할 수 있다.

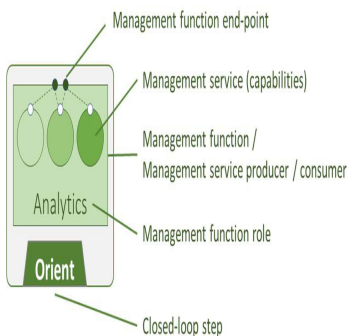
3.1.2 폐쇄 루프(closed-loop) 기반 관리 자율화

폐쇄 루프 기반 관리 체계는 관리 작업의 자율화, 지속적인 최적화, 내부 또는 외부 조건의 변화에 대응하여 관리되는 대상 실체(entity)에 관리 동작 수행을 적용할 수 있으며, 정의된 빌딩 블록으로 구성된다. 관리 기능은 폐쇄 루프의 다른 단계를 달성하기 위해 관리 서비스 능력을 지원한다[19]. 폐쇄 루프 동작은 기본 관리 도메인 및 종단 간 서비스 관리 도메인 수준에서 발생할 수 있으며, 계층적인 관리 도메인 체계를 구성할 수 있다. 다수의 폐쇄 루프는 동시에 실행될 수 있으며 관리 서비스 및 관리 기능의 다양한 하위 집합을 사용하여 수행될 수 있다.

폐쇄 루프의 기본 모델은 OODA(Observe, Orient, Decide, Act)에 근거하고 있으며, MAPE-K(Monitoring, Analysis, Plan, Execution, Knowledge), COMPA(Control, Orchestration, Management, Policy, and Analytics) 등 다양한 유형의 폐쇄 루프를 실현할 수 있다[14]. 폐쇄 루프에는 기계 학습 기술을 적용하여 이점을 얻을 수 있는 여러 단계(step)가 있다. 기계 학습 기반 모니터링, 분석, 의사 결정 및 최적화된 조치(optimized action) 등 모든 단계에서 기계 학습을 통합할 수 있다. 폐쇄 루프는 각 단계에서 필요한 기능의 구성, 그리고 폐쇄 루프의 단계 사이에 의해 필요한 정보 또는 명령의 전송/접근을 제공해야 한다. 관리 기능은 각각의 관리 서비스 기능과 함께 그림 5에 나와 있는 것처럼 폐쇄 루프의 다양한 단계의 기능을 달성한다[16, 19].



Legend:



Different types of data can be written or read to/from the Knowledge step of the closed loop. E.g. collected data (PM), topology, insight / recommendation, decisions, plans, workflows, etc...

The types of data depend on what the closed loop steps need to realize their function, and also what each step need to store as information indicating its activity (e.g. reports, list of actions taken, other metrics...)

그림 5. 아키텍처 빌딩 블록과 폐쇄 루프 단계 사이의 지시적 매핑
Fig 5. Indicative Mapping between Architectural Building Blocks and Closed-loop Steps

폐쇄 루프는 관리 대상 실체인 관리되는 자원 및 해당 운영 문맥(context)에 대한 데이터를 지속적으로 관찰하고 수집한다. 이를 통해 루프의 관리 기능은 이러한 변경 사항을 분석하여, 동작의 진전 상황을 이해하고, 계획 및 결정을 생성한 다음, 관리 대상의 현재 관찰된 상태를 원하는 대상 상태 또는 목표에 맞게 조정하는 작업을 수행할 수 있다.

폐쇄 루프는 동작의 부분으로 다음 기능을 수행한다.

1. 관리 대상 및 기타 관련 데이터 소스에서 데이터를 관찰하고 수집한다.
2. 해당 데이터의 의미와 중요성을 적절한 맥락에서 이해할 수 있도록 이러한 데이터의 처리 방향을 지정한다.

3. 필터링, 상관관계 및 기타 메커니즘을 통해 수집된 데이터를 분석하여 과거, 현재 및 미래 상태의 모델을 정의한다.
4. 기계 학습 알고리즘을 실행하여 패턴을 학습하고 인식하며 관찰된 데이터를 예측한다.
5. 추세 추론, 근본 원인 및 순서 결정, 유사한 프로세스를 기반으로 다양한 조치를 계획한다.
6. 관리 행동의 실행 계획과 시기를 결정한다.
7. 서로 다른 목표 간의 충돌을 감지하고 해결한다.
8. 다른 제어 루프와 협업한다.

3.2 ENI(Experiential Networked Intelligence) 시스템

ETSI ISG(Industry Specification Group) ENI 시스템은 운영자 경험을 향상시키는 혁신적인 정책 기반 모델의 기능 구조이다[1]. 네트워크 자율화 외에도 ENI 시스템은 인간과 기계의 의사 결정을 지원하여 비즈니스 요구 사항을 보다 효율적으로 충족하는 인지 네트워크 관리 기능 아키텍처를 정의한다[6, 7, 17, 22]. ENI는 AI를 활용한 폐쇄 루프 네트워크 운용 및 관리를 가능하게 하는 아키텍처이다. ENI는 지능형 네트워크 운용 및 관리 권장사항(recommendations) 및/또는 명령(commands)을 피 지원 시스템(Assisted System: AS)에 제공하는 실체이다[6, 17]. AS의 운영자와 ENI 간의 신뢰도에 의존하여 권장사항과 정책 명령어 사용을 결정할 것이다. ENI 시스템은 다음 세 가지 모드 중 하나로 AS와 통신하는 외부 실체로 전개 및 배포될 수 있다. (1) AS의 운영자, 요소 관리 시스템(EMS), 네트워크 관리 시스템(NMS) 또는 오케스트레이터와 직접 통신하거나 (2) 애플리케이션 프로그래밍 인터페이스들(APIs)을 사용하거나 (3) 이들의 조합이다.

[7, 22]에서 볼 수 있듯이 AS의 인공 지능 사용 유무에 따라 세 가지 유형이 예상된다. 가장 간단한 (a) 유형의 AS는 AI가 없는 AS이다. ENI와의 상호 작용은 제한적이다. AI 기반 결정을 운용의 일부로 통합할 수 있는 기능(facility)이 없을 수 있기 때문이다. 다음 (b) 유형의 AS는 AI를 사용하지만, 운용 제어 루프의 일부로는 사용하지 않는 AS이다. 따라서 ENI와의 상호 작용은 더 강력하지만 실시간 제어 결정을 포함하지 않는다. 마지막(c)의 AS 유형은 제어 루프에서 AI를 사용하는 유형이다. 따라서 ENI는 AS가 ENI에 권한을 부여한 모든 작업에 대해 AS에 권장사항 및/또는 명령을 제공할 수 있다. AS는 상태, 정책, 명령 및 원격 측정 정보를 전달하기 위해 고유한 기

본 데이터 형식을 사용할 수 있다. API 브로커(broker)의 목적은 서로 다른 다양한 AS의 API와 ENI API 간에 변환 및 중재를 수행한다. API 브로커를 사용한 ENI의 고수준 기능 블록 다이어그램은 그림 6에 나와 있다[7]. 화살표는 12개의 외부 기준점 중 하나를 사용하여 데이터 및 정보의 방향성을 나타낸다.

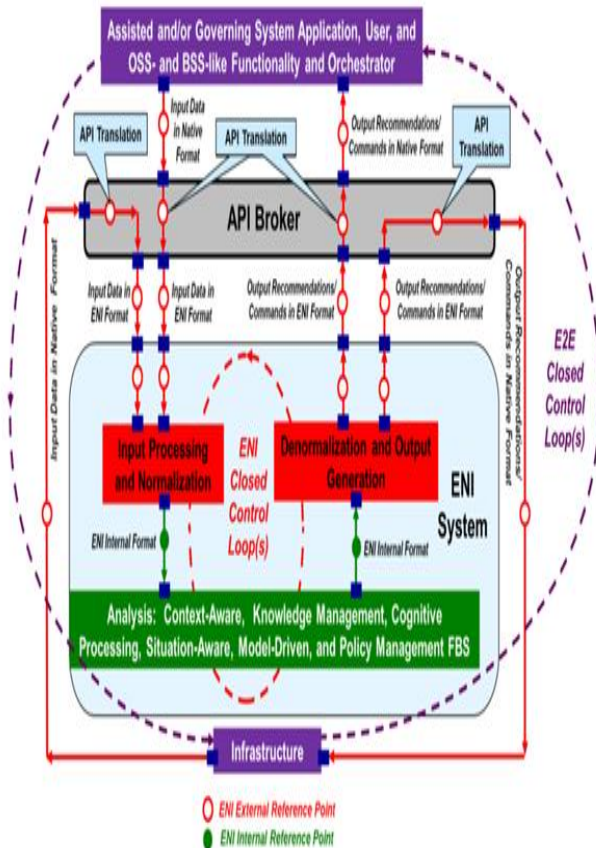


그림 6. API 브로커를 사용한 ENI의 고수준 기능 아키텍처
Fig. 6. High-Level Functional Architecture of ENI when an API Broker is used

ENI 아키텍처는 내부 및 외부의 두 가지 다른 폐쇄 제어 루프(Closed Control Loop)를 제공한다. 폐쇄 제어 루프는 OODA 패러다임을 기반으로 하지만, FOCAL(Founda_{tion}, Observa_{tion}, Compari_{son}, Action, Learn, rEason) 아키텍처와 유사한 개선을 제공할 수 있다[7]. 이를 통해 제어 루프의 각 부분이 AI 기반의 계획 및 학습 메커니즘을 사용하여 인지적 의사 결정을 제공할 수 있다[15]. 내부 제어 루프는 실제로 OODA의 네 가지 기본 프로세스를 병렬로 실행할 수 있는 여러 개의 폐쇄형 제어 루프로 구성된다. 이러한 제어 루프는 ENI 작동 모드와 관계없이 항상 존재한다. 외부 제어 루프는 전체 OODA 프로세스가 동작을 조정하여 원하는 출력 상태를 자

동으로 유지하도록 한다[22, 23].

입력 처리(Input Processing)는 데이터 수집 및 정규화(Normalization) 기능 블록(Functional Block: FB)으로 구성된다. 모든 외부 데이터는 데이터 수집 기능 블록(Data Ingestion Functional Block)으로 모인다. 데이터 수집 FB의 목적은 여러 입력 소스에서 데이터를 수집하고, 이를 다른 ENI 기능 블록에서 추가로 처리하고 분석할 수 있도록 하는 일반적인 데이터 처리 기술이다. 정규화 기능 블록의 목적은 데이터 수집 기능 블록에서 수신한 데이터를 다른 ENI FB가 이해하고 사용할 수 있는 형식으로 처리하고 변환하는 것이다.

분석(Analysis)을 위한 지식 관리 및 처리는 지식 관리(Knowledge Management), 문맥 인식 관리(Context-Aware Management) 및 인지적 처리(Cognitive Processing) 세 가지 기능 블록으로 구성된다. 상황 기반(Situation-based) 정책 생성 기능 블록(Policy Generation Functional Block)은 상황 인식(Situation Aware), 모델 기반(Model-Driven) 엔지니어링 및 정책 관리(Policy Management) 세 가지 FB들로 구성된다. 방향설정(Orient) 부분의 구현은 지식 관리, 문맥 인식, 인지 프레임워크 및 상황 인식 FB로 함께 구성된다.

- 지식 관리 FB는 AI 처리의 기본 요소이며 모델 기반 엔지니어링 및 형식 논리를 통합한다. 모든 ENI FB에서 사용할 수 있는 데이터의 형식적 표현을 정의하여 기계 학습 및 추론을 가능하게 한다. 이는 수집된 데이터와 ENI 시스템 데이터에서 새로운 데이터와 변경된 데이터를 분석, 검증 및 추론한다.
- 문맥 인식 FB는 AS 및 해당 환경에 대한 정보를 수집하여 문맥의 변화에 따라 동작을 조정할 수 있도록 한다(예: 사용자, 장치 혹은 애플리케이션 기록 및 정보, 위치, 동작 특성 등).
- 인지적 처리 FB는 ENI에 대한 일련의 고수준 목표를 만들고 유지 관리하는 역할을 한다. 이러한 목표는 운영자가 학습 및/또는 제공한 SLA, 최적화, 효율성, 보안 등이 될 수 있다.
- 상황 인식 FB는 발생하는 사건과 잠재적 조치가 AS 및 관련 ENI 최종 목표에 어떤 영향을 미칠 수 있는지 평가하기 때문에 중요한 역할을 한다. 따라서 이 FB는 방금 일어난 사건 및 시스템 상태와 ENI에서 권장하거나 추진하는 잠재적 조치를 감안할 때 상황이 왜, 어떻게 전개될 수 있는지 이해할 수 있다.

방향설정 단계는 ENI에 제공된 목표를 기반으로 AS의 현재 상태를 원하는 상태와 비교하고 필요한 경우 선택적 조치 세트를 제안한다.

결정 기능은 모델 기반 엔지니어링 및 정책 관리 FB에서 제공한다. 모델 기반 엔지니어링 FB는 ENI의 AS 구성 내부 표현을 기반으로 제안된 변경 세트를 가져와 적절한 조치 세트로 변환한다. 그런 다음 정책 관리 FB는 이러한 조치를 재사용 가능한 정책으로 변환한다[24].

조치 부분은 역정규화(Denormalization) 및 출력 생성(Output Generation) FB로 구성된다. AS의 특성에 따라 출력은 ENI에서 지원하는 외부 형식에 맞게 적절히 역정규화 된다. 역정규화 기능 블록의 목적은 ENI 시스템의 다른 FB에서 수신한 데이터를 일련의 목표가 된 실체 집합이 이해할 수 있는 형식으로 처리하고 변환하는 것이다. 출력 생성 기능 블록의 목적은 역정규화 기능 블록에서 수신한 데이터를 AS 또는 지정된 실체가 이해할 수 있는 형식으로 변환하는 것이다. 여기에는 적절한 프로토콜 집합 정의, 데이터 인코딩 변경 및 기타 관련 기능이 포함된다. API 브로커는 네이티브 형식으로 AS에 표시되기 전에 출력을 추가로 변환할 수 있다.

산업의 모멘텀과 표준 방식으로 AI 메커니즘을 사용하는 네트워크와의 융합을 촉진하기 위해 ENI는 일련의 외부 참조 지점을 게시한다. 이러한 외부 참조 지점은 ENI 기능의 고수준 의미에 대한 표준 정의를 제공하는 데 유용하며, 수집되고 생성된 정보는 ENI 정보 모델에 저장된다.

IV. ZSM/ENI 기반 자율 관리 시스템

모든 네트워크 기능, 운용 및 애플리케이션을 서비스로 구현함으로써 6G와 같은 미래 차세대 모바일 네트워크 아키텍처는 소프트웨어화되고 지능적으로 구성될 것이다. 따라서 미래 네트워크는 매우 동적이고 복잡하게 구성 및 유지될 것이며, ZSM/ENI 표준 개념을 적용한 네트워크 및 서비스 관리 시스템의 연구 및 동향 분석은 중요한 역할을 담당하리라고 본다. 자율 관리 시스템은 지능화된 네트워크, 서비스의 관리 및 오케스트레이션의 수행을 목표로 하고 있다. 그림 7은 ZSM/ENI 기반 시스템을 통합하여 적용한 전형적인 자율 지향적 네트워크 및 서비스 관리 시스템을 구성한 것으로 전반적인 자율 관리 시스템 체제를 살펴보면 도움이 되리라고 본다.

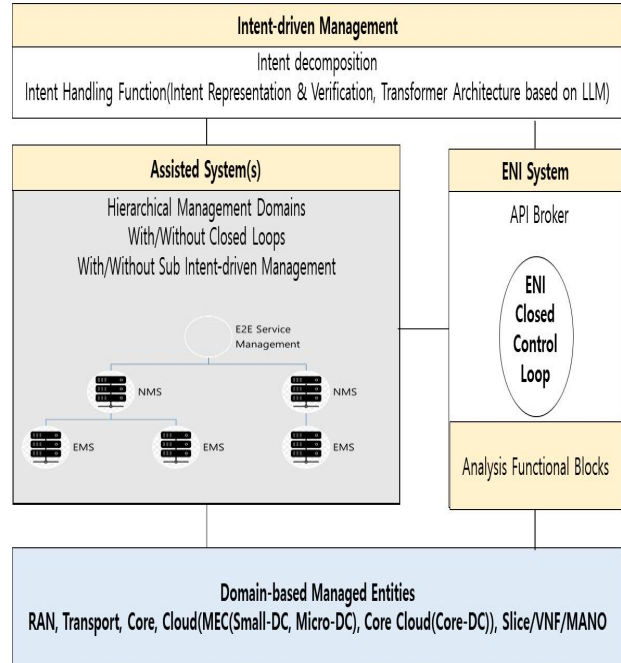


그림 7. ZSM/ENI 기반 통합 관리 시스템들

Fig. 7. Integrated Management Systems based on ZSM/ENI

관리 대상 실체들은 도메인 기반으로 다양하고 이질적인 물리적 및 논리적 가상화된 자원들로 구성된다. 이들은 RAN, 전송, 코어, 에지 클라우드, 코어 클라우드, MANO[14]와 같은 다양한 기술적 도메인에 교차되어 관리되어질 것이다[1, 4, 11, 13, 14, 17]. 관리 대상 실체는 AS의 계층적 관리 시스템들의 최하위단인 EMS 및 ENI 시스템에 의하여 직접 관리된다. 이들은 관리 명령어를 기반으로 제어를 수행하거나 모니터링 대상이 되는 데이터를 제공받아 유지할 것이다. 관리 영역인 FCAPS(Fault, Configuration, Accounting, Performance, and Security)의 일반적인 관리 방법론은 관리 대상 실체의 상태에 대한 필요한 정보를 제공할 수 있는 정교한 데이터 모니터링 및 수집 프레임워크를 필요로 한다. AS들은 SMS, NMS, EMS로 계층적으로 구성되어있으며, 각 관리 시스템은 폐쇄 루프를 지원하거나 지원하지 않을 수 있다. 이는 기존 및 신규 관리 시스템의 구성을 가능하게 한다[6]. 계층적 관리 시스템에서 최상위 단인 E2E 서비스 관리 시스템의 폐쇄 루프는 광범위한 관점을 가지고 있으며, 비즈니스 입력을 도메인별 입력으로 변환하고 분해하는 역할을 한다. 최하위단 관리 도메인의 폐쇄 루프는 협의의 관점을 가지고 있으며 E2E 서비스의 일부를 보장 및 관리하는 역할을 수행한다. ENI 시스템은 AS의 능력에 따라서 API 브로커를 사용하여 표준화된 폐쇄 제어 루프 기능들을 제공할 수 있다.

의도 주도(intent-driven) 관리 시스템들은 ENI 시스템 혹은 기존의 AS를 지원하기 위하여 사용되어질 수 있다. 의도 주도 기반 관리로 입력 정보를 더 간단하고 사용자 친화적으로 표현할 수 있으며, 자동화의 유연성이 더 높아진다. 의도는 자동화를 증진시키고 관리를 더 단순하게 만드는 핵심 요소이다[21, 22, 25]. 의도는 분해(decomposition)가 이루어지고, 처리 기능(Handling Function)을 거쳐 AS 및 ENI에 정책의 형태로 관리 명령어들이 제공된다. 이를 효과적으로 수행하기 위하여 의도 표현 및 검증(Intent Representation and Verification), 대규모 언어 모델(LLM)과 같은 트랜스포머 모델이 사용될 수 있다[14, 15].

네트워크 및 서비스 오케스트레이션을 위한 미래 통신 및 네트워크 성능 목표를 달성하려면 자율화가 필요하다. 따라서 의도 주도 기반 관리, 폐쇄 루프 및 AI/ML과 같은 기술을 구조적으로 일관성 있게 표준화 형태로 관리 시스템을 제공하는 것은 중요할 것이다. 예를 들면 폐쇄 제어 루프는 네트워크 슬라이스 관리 및 오케스트레이션을 수행하기 위한 AI/ML 프레임워크를 제공한다. ENI 폐쇄 제어 루프들은 주로 AI/ML에 의존하며, 자체 관리(self-management), 자체 구성(self-configuration) 및 자체 적응(self-adaptation)을 달성할 수 있다[11]. 이는 AS, 오케스트레이터 및 인프라와의 향상된 협업을 통하여 자동화 수준을 높여 완전 자율화된 폐쇄 루프 제어를 목표로 설계될 것이다.

V. 결 론

6G 시스템과 같은 미래의 차세대 모바일 네트워크는 여러 기술 영역(예: 무선 접근 네트워크, 전송, 코어, 에지 클라우드 및 코어 클라우드)을 걸쳐 다수의 이질적인 네트워크 서비스 및 슬라이스를 동적이며 복잡한 형태로 제공할 것으로 예상된다. 이는 확장성과 지속성의 관점에서 고전적인 중앙 집중식 관리와 오케스트레이션 접근 방법에 대한 중요한 도전으로, 일관성 있고 표준화된 지능형 기반의 네트워크 및 서비스 관리 시스템을 필요로 한다.

이를 위하여 ZSM 및 ENI 그리고 의도 주도 기반 관리 기술의 연구 및 동향을 지속적으로 살펴보는 것은 중요하다. 자율화를 목표로 하는 이러한 기술들은 AI/ML을 기반으로 강화 학습, 연합 학습, 전이 학습을 실제 구현 및 적용하는 많은 연구가 이루어질 것으로 본다.

References

- [1] Miranda McClellan, et al., "Deep Learning at the Mobile Edge: Opportunities for 5G Networks," Applied Sciences, July 2020.
- [2] Weikang Xu, et al., "A Comprehensive Discussion on Deep Reinforcement Learning," IEEE 3rd International Conference on Communications, Information System and Computer Engineering (CISCE 2021), pp. 697-702, 2021.
- [3] Charles Ssengonzi, et al., "A Survey of Deep Reinforcement Learning Application in 5G and beyond Network Slicing and Virtualization," Elsevier Array 14, April 2022.
- [4] 조재형, 이재오, "기계 학습 기반의 스마트 팩토리 설비 장애 관리 시스템," KNOM Review, Vol. 26, No. 1, pp. 31-40, 2023.
- [5] Xiaofei Wang, et al., "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," IEEE Network, Vol. 33, Issue 5, 2019.
- [6] Yue Wang, et al., "Network Management and Orchestration Using Artificial Intelligence: Overview of ETSI ENI," IEEE Communications Standards Magazine, Vol. 2, Issue 4, Dec. 2018.
- [7] Yue Wang et al., "From Design to Practice: ETSI ENI Reference Architecture and Instantiation for Network Management and Orchestration Using Artificial Intelligence," IEEE Communications Standards Magazine, pp. 38-45, Sept. Sep. 2020.
- [8] Xiaoyu Qiu, et al., "Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective," IEEE Transactions on Parallel and Distributed Systems, Vol. 32, No. 5, May 2021.
- [9] Hojjat Baghban, et al., "Edge-AI: IoT Request Service Provisioning in Federated Edge Computing Using Actor-Critic Reinforcement Learning," IEEE Transactions on Engineering Management, Vol. 71, pp. 11519-11528, May 2022.
- [10] Gerald M. Karam, et al., "The Evolution of Networks and Management in a 6G world: An Inventor's View," IEEE Transactions on Network and Service Management, 2022.

[11] Shalli Rani, et al., "Network Slicing for Zero-Touch Networks: A Top-Notch Technology," IEEE Network, Vol. 37, Issue 5, 2023.

[12] Ovidiu Iacoboaia, et al., "From Design to Deployment of Zero Touch Deep Reinforcement Learning WLANs," IEEE Communications Magazine, Vol. 61, Issue: 2, 2023.

[13] Sand L. Correa, et al., "Supporting MANOaaS and Heterogenous MANOaaS Deployment Within the Zero-Touch Network and Service Management Framework," IEEE Communications Standards Magazine, Vol. 8, Issue: 2, 2024.

[14] Jingwen Zang, et al., "Intent-Driven Closed Loop Control and Management Framework for 6G Open RAN," IEEE Internet of Things Journal, Vol. 11, No. 4. Feb. 2024.

[15] Ray Forbes, et al., "Transformers and Large Language Models as Used in ETSI ISG Experiential Networked Intelligence," IEEE ICC 2024 1st Workshop on The Impact of Large Language Models on 6G Networks, pp. 1250-1255, 2024.

[16] ETSI GS ZSM 002 V1.1.1 (2019-08) Zero-touch network and Service Management (ZSM); Reference Architecture.

[17] ETSI GS ZSM 003 V1.1.1 (2021-06) Zero-touch network and Service Management (ZSM); End-to-end management and orchestration of network slicing.

[18] ETSI GS ZSM 008 V1.1.1 (2022-07) Zero-touch network and Service Management (ZSM); Cross-domain E2E service lifecycle management.

[19] ETSI GS ZSM 009-1 V1.1.1 (2021-06) Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers.

[20] ETSI GR ZSM 011 V1.1.1 (2023-02) Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects.

[21] ETSI GS ZSM 012 V1.1.1 (2022-12) Zero-touch network and Service Management (ZSM); Enablers for Artificial Intelligence-based Network and Service Automation.

[22] ETSI GS ENI 005 V3.1.1 (2023-06) Experiential Networked Intelligence (ENI); System Architecture.

[23] ETSI GR ENI 017 V2.1.1 (2021-08) Experiential Networked Intelligence (ENI); Overview of Prominent Control Loop Architectures.

[24] ETSI GS ENI 030 V4.1.1 (2024-03) Experiential Networked Intelligence (ENI); Transformer Architecture for Policy Translation.

[25] ETSI GS ZSM 016 V1.1.1 (2024-10) Zero-touch network and Service Management (ZSM); Intent-driven Closed Loops.

양 우 석 (Wooseok Yang)



2014년 8월: 한국기술교육대학교, 정보통신학과 학사

2016년 8월: 한국기술교육대학교, 전기전자통신공학 석사

2016년 9월~현재: 한국기술교육대학교, 전기전자통신공학 박사과정 수료

<관심분야> 네트워크 및 서비스 관리, NMS,, NFV/SDN, ZSM/ENI

이 재 오 (Jaeoh Lee)



1987년 2월: 광운대학교 전자계산학과 학사

1989년 2월: 광운대학교 전자계산학과 석사

1993년 8월: 광운대학교 전자계산학과 박사

2002~현재: 한국기술교육대학교

전기전자통신공학부, 교수

<관심분야> 차세대 서비스 응용, 서비스 가상화 및 네트워크 슬라이싱, AI/ML 기반 네트워크 및 서비스 관리, ZSM/ENI

머신 러닝을 활용한 스마트 컨트랙트 취약점 탐지 연구 동향

이 의 찬*, 방 지 원*, 최 미 정°

A Survey on Vulnerability Detection in Smart Contract Using Machine Learning Techniques

UI-Chan Lee*, Jiwon Bang*, Mi-Jung Choi°

요 약

최근 블록체인 기술이 다양한 산업(금융, 의료)에서 활발히 활용되며, 블록체인의 핵심 기술 중 하나인 스마트 컨트랙트(Smart Contract) 또한 큰 관심을 받고 있다. 스마트 컨트랙트는 제 3자의 개입 없이 탈중앙화된 환경에서 신뢰성 있는 거래 기능을 제공한다. 하지만 스마트 컨트랙트의 사용이 늘어남에 따라 여러 취약점이 노출될 가능성 역시 증가하고 있다. 실제 스마트 컨트랙트의 여러 보안 취약점으로 인한 금전적인 피해 사례들이 보고되고 있다. 이러한 문제를 방지하기 위해 스마트 컨트랙트 코드 내의 취약점을 사전에 탐지하는 연구가 활발히 진행되고 있다. 최근에는 다양한 분야에서 뛰어난 성능을 보이는 머신 러닝 기법을 도입한 스마트 컨트랙트 취약점 탐지도 활발히 연구되고 있다. 본 논문에서는 스마트 컨트랙트의 주요 취약점들과 머신 러닝 기법을 활용한 스마트 컨트랙트 취약점 탐지 연구를 소개한다. 또한 선정한 연구를 입력 데이터 별로 분류하고 각 연구의 특징을 조사한다.

Key Words : Blockchain, Ethereum, Smart Contract, Vulnerability, Machine Learning, Detection

ABSTRACT

Blockchain technology has been actively utilized in various industries, such as finance and healthcare, with growing interest in smart contracts, one of the core technologies of blockchain. Smart contracts enable reliable transactions in a decentralized environment without the involvement of third parties. However, as the use of smart contracts increases, the potential for various vulnerabilities to be exposed also rises. Indeed, there have been reported cases of financial losses caused by security vulnerabilities in smart contracts. To address these issues, research on detecting vulnerabilities in smart contract code in advance has been actively conducted. Recently, machine learning techniques, which have demonstrated excellent performance in various fields, have been applied to the detection of smart contract vulnerabilities. This paper introduces the key vulnerabilities of smart contracts and explores studies that utilize machine learning techniques for vulnerability detection. Additionally, it categorizes the selected studies based on input data and investigates the characteristics of each study.

※본 연구는 보건복지부의 재원으로 한국보건산업진흥원의 보건의료기술연구개발사업지원에 의하여 이루어진 것임(과제 번호: HI23C0733)

♦ First Author : kangwon National University Department of Computer Science and Engineering, jungkls3@kangwon.ac.kr

° Corresponding Author : Kangwon National University Department of Computer Science and Engineering, IGP. in Medical Bigdata Convergence, and Department of Data Science, mjchoi@kangwon.ac.kr

* kangwon National University IGP. in Medical Bigdata Convergence, and Department of Data Science, jiwonbang@kangwon.ac.kr

논문번호 : KNOM2024-02-03, Received November 18, 2024; Revised December 13, 2024; Accepted December 20, 2024

I. 서 론

블록체인 기술은 2008년 사토시 나카모토의 Bitcoin: A Peer-to-Peer Electronic Cash System[1]을 통해 처음 공개되었다. 블록체인은 중앙 기관 없이도 거래의 투명성과 무결성을 확보할 수 있는 기술로, 금융, 의료, 물류 등 다양한 산업에서 활용되고 있다. 블록체인 기술이 주목받기 시작하면서 스마트 컨트랙트(Smart Contract)에 대한 관심 또한 증가하였다. 스마트 컨트랙트는 2015년 비탈릭 부테린이 이더리움(Ethereum)을 발표하면서, 조건에 따라 자동으로 계약이 실행되는 시스템이 등장했다. 스마트 컨트랙트는 특정 조건이 충족될 때만 계약이 이행되므로, 거래 당사자 간에 신뢰할 수 있는 제 3자의 개입 없이도 안전한 거래가 가능하다.

하지만 스마트 컨트랙트에는 다양한 보안 취약점이 존재하고, 이로 인한 실제 피해 사례들[2, 3]이 꾸준히 보고 되고 있다. 블록체인에 배포된 스마트 컨트랙트는 수정이 불가능한 특징이 있다. 이로 인해, 스마트 컨트랙트 내에 보안 취약점이 발생했을 때 그 피해가 크다.

실제로, 2016년에 설립된 The DAO는 스마트 컨트랙트의 취약점으로 인해 금전적인 피해를 입었다. DAO(Decentralized Autonomous Organization)는 전통적인 조직과 달리 중앙 관리자 없이 스마트 컨트랙트를 통해 규칙과 절차가 자동으로 실행되는 탈중앙화된 자율 조직이다. 익명의 공격자가 The DAO 컨트랙트에 존재하는 재진입(reentrancy) 취약점을 악용하여 대략 360만 ETH를 탈취했으며, 이는 당시 시세로 약 640억원 상당 금액이다[4]. 다른 사건으로 2021년 Cream Finance에서 발생한 해킹 사건이 있다. Cream Finance는 탈중앙화 금융(DeFi) 플랫폼으로, 사용자가 자산을 빌리고 빌려주는 기능을 제공한다. 그러나 스마트 컨트랙트 코드에 존재하는 플래시 론(flash loan) 취약점을 악용하여 약 1600억원의 손실이 발생하였다[5].

이러한 피해를 방지하기 위해 스마트 컨트랙트를 배포하기 전에 취약점을 탐지하는 연구가 활발히 이루어지고 있다. 정적 분석(static analysis), 기호 실행(symbolic execution)과 같은 고전적인 탐지 방법뿐만 아니라 최근에는 머신 러닝을 활용한 방법이 연구되고 있다. 또한 스마트 컨트랙트

트 취약점 탐지 연구에 대한 연구 동향 논문 또한 지속적으로 발표되고 있다[6, 7, 8, 9].

기존의 동향 논문은 주로 머신 러닝 모델에 초점을 맞추어 모델을 중심으로 각 연구의 특징을 제시했다. 하지만 이러한 접근 방식은 머신 러닝 기법에서 핵심적인 요소인 입력 데이터에 대한 정보 전달이 부족하다. 머신 러닝을 활용한 많은 연구가 진행되는 상황에서 입력 데이터에 대한 체계적인 분류와 분석이 필요하다.

본 논문은 스마트 컨트랙트 취약점 탐지 연구에서 입력 데이터 유형의 중요성을 강조하며, 이를 기반으로 연구를 입력 데이터 유형(바이트코드, 오피코드, 소스 코드)별로 분류하고 분석한다. 또한 스마트 컨트랙트에 존재 가능한 주요 취약점을 정리하고, 선정한 연구의 주요 공헌도를 설명한다. 이를 통해 연구자들이 각 데이터 유형별 연구 동향과 특징을 파악하고, 데이터 선택 및 모델 설계 전략을 수립하는 데 가이드를 제공한다.

본 논문의 구성은 다음과 같다. 제 2장에선 블록체인과 스마트 컨트랙트 같은 배경 지식을 소개하고 제 3장에선 스마트 컨트랙트의 주요 취약점들에 대해 소개한다. 제 4장에선 머신 러닝을 활용한 취약점 탐지 연구를 분류하고 특징을 설명한다. 마지막 제 5장에서는 연구 결과를 요약하고 향후 연구에 대해 논의한다.

II. 배경

2.1 블록체인

블록체인은 데이터를 중앙 서버가 아닌 여러 노드에 분산 저장하는 방식으로 탈중앙성, 보안성, 투명성을 제공하는 분산 원장(distributed ledger) 기술이다. 블록체인의 각 노드는 블록이라고 불리는 데이터를 저장하고, 블록들은 체인 형태로 연결되어 불변성을 보장한다. 블록의 구조는 거래 기록, 이전 블록의 해시값, 현재 블록의 해시값 등의 정보를 포함하고 있으며, 블록체인마다 구조는 다를 수 있다.

블록체인 기술은 익명의 개발자, 사토시 나카모토에 의해 처음 공개되었다. 전자화폐 시스템에서 발생하는 이중 지출(double-spending) 문제를 작업 증명(proof of work)이란 합의 알고리즘을 통해 해결하였다. 작업 증명 방식은 SHA 해시함수를 활용하여 특정 해시값을 찾는 계산 작업을 통

해 새로운 블록을 생성한다. 이 과정에서 거래 내역을 조작하기 위해선 많은 연산 자원이 필요하므로 이중 지불 문제를 해결할 수 있다. 합의 알고리즘을 통해 네트워크 내의 모든 노드가 동일한 거래 내역을 공유하고, 새로운 거래를 추가할 때 이를 검증하여 네트워크의 무결성을 유지할 수 있다.

블록체인의 특징은 다음과 같다. 블록체인은 중앙 관리자 없이도 네트워크의 모든 참여자가 동일한 원장을 유지한다. 이를 통해 중개자 없이도 신뢰할 수 있는 탈중앙화된 시스템을 구축할 수 있다. 블록체인 네트워크에 참여하는 누구나 원장의 내용을 열람할 수 있으므로 거래의 투명성을 보장한다. 한번 블록체인에 기록된 데이터는 변경할 수 없다. 모든 거래는 이전 블록과 연결되어 중간에 거래 내용을 수정하려면 전체 체인을 변경해야 하는데 이는 사실상 불가능하다.

2.2 스마트 컨트랙트

스마트 컨트랙트는 1994년 닉 제보(Nick Szabo)가 최초로 제안한 개념이다[10]. 스마트 컨트랙트는 계약 실행에 필요한 조건을 명시하고, 해당 조건이 만족되면 계약이 자동으로 실행되는 프로그램이다. 제 3자의 개입 없이도 신뢰할 수 있는 계약 이행이 가능하도록 설계되었다. 스마트 컨트랙트는 데이터 위변조 문제와 기술적 한계로 인해 실제 구현이 어려웠으나, 2015년 비탈릭 부테린은 스마트 컨트랙트를 블록체인에 도입한 이더리움(Ethereum)을 발표하면서 기술적 구현이 가능해졌다.

스마트 컨트랙트는 블록체인에 배포된 이후 수정이 불가능하다는 특징이 있다. 이러한 불변성은 스마트 컨트랙트의 신뢰성을 보장하는 중요한 요소지만, 동시에 치명적인 보안 취약점이 될 수 있다. 스마트 컨트랙트 코드를 배포한 후 발견되는 오류나 보안 문제는 해결하기 어렵기 때문에, 스마트 컨트랙트의 보안 검증은 매우 중요하다.

스마트 컨트랙트는 실행 과정에서 크게 3가지 형태를 거친다. 그림 1은 스마트 컨트랙트가 실행되는 과정에서 형태의 변화를 나타낸다. 소스 코드(source code)는 사람이 이해할 수 있는 형태로, 솔리디티(Solidity)[11], 바이퍼(Vyper)[12]와 같은 다양한 스마트 컨트랙트 전용 프로그래밍 언어로 작성된다. 개발자는 이를 통해 함수, 상태 변수 등을 설정하여 스마트 컨트랙트의 로직을 담을 수 있다. 바이트코드(bytecode)는 고수준

언어로 작성된 스마트 컨트랙트 소스 코드를 컴파일러가 변환한 결과물이다. 바이트코드는 16진수 형태로 나타나며, 이더리움의 독립적인 실행 환경인 이더리움 가상 머신(Ethereum Virtual Machine, EVM)을 통해 실행된다. 오퍼코드(opcode)는 컴파일된 바이트코드를 구성하는 기본 명령어 단위로, EVM에서 이해하고 실행할 수 있는 최소 실행 명령어이다. 오퍼코드에는 여러 명령어(ADD, MUL, PUSH)가 포함된다. 결론적으로 솔리디티와 같은 고수준 언어로 작성된 스마트 컨트랙트는 컴파일 과정을 거쳐 바이트코드로 변환되며, 이 바이트코드는 오퍼코드로 구성되어 EVM에서 실행된다.

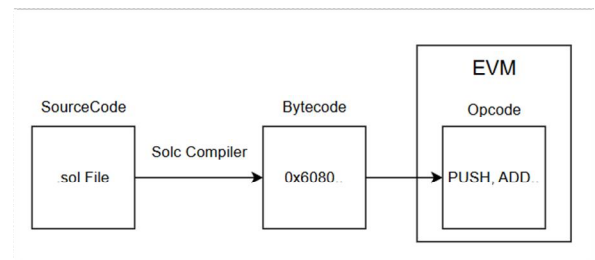


그림 1. 스마트 컨트랙트 실행 과정

Fig. 1. The process of executing a smart contract

Ⅲ. 스마트 컨트랙트 취약점

스마트 컨트랙트는 블록체인 환경에서 자동으로 실행되는 코드로, 한 번 배포되면 수정이 어려워 코드 작성 과정에서 보안이 매우 중요하다. 그러나 스마트 컨트랙트 코드 작성 시 개발자의 실수, 코드 내 버그, 잘못된 API 호출 등으로 인해 다양한 보안 취약점이 발생할 수 있다. 본 논문에서는 영국의 NCC Group에서 공개한 DASP(Decentralized Application Security Project)[13]의 9가지 주요 취약점을 기반으로 스마트 컨트랙트 취약점을 설명한다:

- RE(Reentrancy) : The DAO 사건에서 발생한 주요 취약점인 재진입 공격은 외부 컨트랙트를 호출할 때 원래 컨트랙트로 반복해서 재진입할 수 있는 구조에서 발생한다. 이로 인해 자금의 중복 인출 같은 문제가 발생할 수 있다.
- AC(Access Control) : AC는 모든 프로그램에서 흔하게 발생하는 취약점으로 권한 제어가 제대로 설정되지 않는 것을 의미한다. 이에 악의적인 사용자가 스마트 컨트랙트의 중요한

기능에 접근할 수 있게 되어 자산의 탈취, 관리 기능의 조작 등 다양한 보안 문제가 발생할 수 있다.

- AE(Arithmetic Error) : 정수 오버플로(overflow) 및 언더플로(underflow)로도 알려진 취약점으로, 계산이 예상 범위를 벗어날 때 발생하여 잘못된 결과를 초래할 수 있다.
- UE(Unhandled Exception) : 저수준(low level) 함수 호출(call, send) 후 return 값에 대한 처리를 하지 않으면 저수준 함수 실행이 실패해도 계약이 계속 진행될 수 있다. 이에 자금 손실 또는 의도하지 않은 문제가 발생할 수 있다.
- DoS(Denial of Service) : 스마트 컨트랙트에 존재하는 특정 기능의 과도한 실행을 유도하거나, 자원을 독점해 스마트 컨트랙트가 정상적으로 작동하지 못하게 할 수 있다. 이로 인해 자원 낭비와 기능의 비정상적 동작 등의 문제를 초래할 수 있다.
- BR(Bad Randomness) : 스마트 컨트랙트에서 난수를 예측 가능한 방식으로 생성하면 공격자가 결과를 조작할 수 있다. 난수 생성에 문제가 있으면 무작위성이 중요한 서비스(게임, 복권 등)에서 부정확한 결과가 발생할 수 있다.
- FR(Front-Running) : 공격자가 네트워크상의 트랜잭션을 미리 감지하고, 더 높은 가스비를 지불하여 자신의 트랜잭션을 먼저 처리되도록 함으로써 이득을 취할 수 있다. race condition, transaction ordering dependence로도 알려져 있다. 이를 통해 다른 사용자의 거래를 방해하거나 유리한 조건을 선점하여 부당한 이익을 취할 수 있다.
- TD(Timestamp Dependence) : 블록 타임스탬프 취약점은 스마트 컨트랙트에서 블록의 타임스탬프를 사용하여 특정 조건을 제어하는 경우 발생한다. 악의적인 채굴자가 타임스탬프를 조작할 수 있다. 타임스탬프 조작으로 인해 시간에 따라 실행 조건이 달라질 수 있는 스마트 컨트랙트가 영향을 받을 수 있다.
- SA(Short Address Attack) : 해당 취약점은 트랜잭션 데이터의 길이가 올바르게 확인되지 않을 때 발생한다. 악의적인 공격자가 의도적으로 일부분을 잘라낸 주소를 제공할 경우, 스마트 컨트랙트가 예상하지 못한 방식으로 입력을 해석하게 만들 수 있다.

IV. 머신 러닝을 활용한 취약점 탐지 연구

머신 러닝이 다양한 분야에서 뛰어난 성능을 보여, 스마트 컨트랙트 취약점 탐지에도 이를 활용한 방법이 연구되고 있다. 본 논문에서는 최신 연구 동향 분석을 위해 2021년 이후에 발표된 머신 러닝을 활용한 연구를 선정했다. 머신 러닝을 활용한 연구에서 주로 사용되는 입력 데이터는 바이트코드, 오피코드, 소스 코드이다. 입력 데이터 유형별 연구 동향을 분석하기 위해, 이를 분류기준으로 선정했다. 선정된 논문을 주요 입력 데이터를 기준으로 분류하고, 각 연구의 특징을 소개했다. 또한 주요 공헌도와 탐지 성능을 표로 정리하여 명시했다. 표의 Model 필드는 연구에서 사용된 모델을 나타낸다. 여러 모델이 연구된 경우 "Many"로 표시했다. Vulnerabilities 필드는 해당 연구에서 탐지한 주요 취약점 목록을, Contribution 필드는 연구의 핵심 공헌도를 나타낸다. 성능 지표의 경우 F1-score와 정밀도(precision)의 평균 값을 명시했다.

4.1 바이트코드 기반 연구

바이트코드는 스마트 컨트랙트 소스 코드가 컴파일 과정을 통해 변환된 형태를 의미한다. 바이트코드 기반 연구의 가장 큰 특징은 소스 코드가 공개되지 않은 경우에도 취약점을 탐지할 수 있다는 점이다. 표 1은 본 논문에서 조사한 바이트코드 기반 연구들을 연도별로 오름차순으로 정리하고, 각 연구의 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

ESCORT[14]는 DNN(Deep Neural Network)과 전이 학습(transfer learning)을 활용한 프레임워크이다. ESCORT는 특징 추출기를 통해 컨트랙트의 의미론적 특징을 추출하고, 다중 출력(multi output) 신경망 구조를 채택하여 각 취약점을 탐지한다. 또한 새로운 취약점 유형에 대한 경량 전이 학습을 지원하여 최소한의 수정으로 확장이 가능하다. 또한 앞서 소개한 주요 취약점 이외에도 CD(Callstack Depth)와 AV(Assert Violation)을 탐지한다. CD는 EVM의 호출 스택 깊이 제한을 악용하여 컨트랙트가 추가 호출을 처리하지 못하도록 방해할 수 있다. AV는 assert문이 false가 되도록 유도하여,

컨트랙트를 오류 상태로 만들 수 있는 취약점이다. 저자들은 스마트 컨트랙트를 수집하고 기존 도구를 통한 라벨링으로 데이터셋을 구축했다.

VSCL[15]은 DNN과 메트릭 학습(metric learning)을 결합한 프레임워크이다. VSCL은 스마트 컨트랙트의 바이트코드를 CFG(Control Flow Graph)로 변환하여 실행 경로와 흐름을 파악한다. 각 CFG의 노드는 컨트랙트의 실행 가능한 경로를 나타내며 CFG 시퀀스는 n-그램과 TF-IDF(Term Frequency-Inverse Document Frequency)를 통해 특징 벡터로 변환된다. 저자들은 메트릭 학습을 통해 클래스 내 분산을 줄이고, 클래스 간 분리를 향상 시켜 성능을 높였다. 또한 3가지 도구(Oyente, Mythril, Vandal) 중 2가지 이상이 탐지한 경우를 취약점으로 고려한 데이터셋과 3가지 도구 중 하나라도 탐지한 경우를 취약점으로 고려한 데이터셋, 총 두 가지를 사용했다. RE, UE 같은 주요 취약점 이외에도 CD와 같은 취약점을 탐지하며, 다각도로 성능 평가를 진행했다.

J. Huang et al.[16]의 저자들은 바이트코드 매칭을 위해 그래프 임베딩(graph embedding)을 도입했다. 그래프를 수치화된 벡터로 변환해 유사성을 측정하고 취약점을 탐지했다. 이 과정에서 다양한 컴파일러 버전으로 인한 바이트코드의 불일치와 불필요한 코드로 인한 간섭 문제를 해결하기 위해, 정규화 및 슬라이싱 기법을 제안했다. 스마트 컨트랙트의 데이터 및 제어 흐름을 추적하여, 취약점과 관련 없는 코드를 제거함으로써 탐지 정확도의 향상을 보였다. 성능 평가를 위해 Mist[17], Etherscan[18]에서 컨트랙트를 수집하고, 데이터셋을 구성했다.

Attention-based[19]의 저자들은 CNN(Convolutional Neural Network)과 셀프 어텐션(self-attention) 메커니즘을 결합한 모델인 ABCNN(Attention-Based CNN)을 제안했다. 오피코드의 특징을 추출하고 셀프 어텐션 기반 임베딩을 통해 스마트 컨트랙트 명령어 간의 연관성을 파악한다. 또한 전처리 과정을 통해 취약점을 나타내는 패턴을 효과적으로 검출하도록 설계했다. ABCNN은 기존의 정적 분석 도구인 Oyente와 Mythril 대비 높은 성능을 보였다. 성능 평가를 위해 Etherscan에서 컨트랙트를 수집하여 데이터셋을 구성했다.

SmartDagger[20]는 두 개 이상의 스마트

컨트랙트 간의 호출 경로에서 발생하는 크로스-컨트랙트 취약점 탐지를 위한 도구이다. 서로 다른 스마트 컨트랙트 간의 상호작용 중 나타나는 문맥 정보, 제어 흐름 의존성, 데이터 흐름 의존성 등을 분석한다. SmartDagger는 바이트코드에서 호출 경로 내의 누락된 컨트랙트 속성을 복원하기 위해 NMT(Neural Machine Translation)를 사용한다. NMT를 통해 스마트 컨트랙트의 의미론적 정보를 되찾고 분석을 수행한다. 또한 크로스-컨트랙트 취약점과 관련된 함수만을 선택적으로 분석하고, 이전의 데이터 흐름 결과를 재사용하여 성능향상을 보였다. 저자들은 성능 평가를 위해 Etherscan에서 컨트랙트 수집하여 데이터셋을 구성했다.

S. J. Qin et al.[21]는 중요 경로 조합(Critical Combination Path, CCP)과 딥러닝을 활용한 취약점 탐지 방법을 제안했다. 저자들은 CFG(Control Flow Graph)와 상태 변수 종속성 그래프를 생성하고, 취약점과 관련된 경로만을 필터링하여 CCP를 구성했다. 또한, 오피코드에서 불필요한 연산자를 제거함으로써 데이터의 노이즈를 줄였다. 정규화된 CCP를 SimHash를 사용해 벡터화하고, 이를 그레이스케일 이미지로 변환하여 ResNet50의 입력으로 활용했다. 성능 평가를 위해 여러 논문과 Awesome-Buggy-ERC20-Tokens[22]에서 컨트랙트를 수집하고 데이터셋을 구성했다.

P. Qian et al.[23]의 저자들은 교사(teacher)-학생(student) 네트워크로 구성된 크로스-모달리티(cross-modality) 상호 학습 프레임워크를 제안했다. 교사 네트워크는 소스 코드와 바이트코드 모두를 사용하여 취약점 정보를 학습하고, 학생 네트워크는 바이트코드 정보만을 학습한다. 학습 중 두 네트워크가 서로 정보를 교환하여, 학생 네트워크가 바이트코드만으로 소스 코드에서 파생된 정보(제어 흐름, 데이터 흐름)를 예측하도록 훈련한다. 이를 통해 탐지 단계에서 바이트코드만으로 RE, TD, AE, DC(DelegateCall)와 같은 여러 취약점을 탐지한다. DC는 호출자의 실행 환경에서 피호출자의 코드를 실행하므로, 환경 차이에 의해 의도치 않은 결과가 발생 가능한 취약점이다. 저자들은 성능 평가를 위해 여러 출처(Github, Ethereum 네트워크)에서 컨트랙트를 수집하고, 데이터셋을 구성했다.

VulHunter[24]는 다중 인스턴스 학습

표 1. 바이트코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 1. Research on Bytecode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
ESCORT[14]	DNN	RE, DoS, FR, AC, Callstack Depth, Assert Violation	- 경량 전이 학습을 지원하는 최초의 DNN 기반 프레임워크 - 최소한의 수정과 재학습으로 확장 가능 - 성능 평가: 정밀도-97%, F1-score-95%
VSCL[15]	CNN	RE, UE, FR, TD, AC, Callstack Depth	- CFG 기반 특징 벡터화와 Metric Learning을 적용한 DNN 모델을 개발 - 성능 평가: 정밀도-87%, F1-score-76%
J. Huang et al.[16]	Graph Embedding	RE, AE, BR, AE, UE	- 바이트코드 매칭을 위해 그래프 임베딩 기법을 최초로 도입 - 성능 평가: 정밀도-91.95%
Attention-based[19]	Hybrid Model	RE, AE, TD	- CNN과 self-attention을 결합한 모델을 제안 - 성능 평가: 정밀도-83%, F1-score-88%
SmartDagger[20]	Hybrid Model	RE, TD, DoS, AC	- 크로스-컨트랙트 취약점 탐지를 위한 최초의 도구 - 바이트코드에서 누락된 컨트랙트 속성을 복원, 분석 - 성능 평가: 정밀도-88%
S. J. Qin et al.[21]	CNN	RE, AC, AE, TD	- CCP 기법을 사용한 코드 필터링을 통해 노이즈를 감소 - 정규화된 CCP를 SimHash를 사용해 벡터화한 후 분류 - 성능 평가: 정밀도-81.20%, F1-score-82.89%
P. Qian et al.[23]	Hybrid Model	RE, TD, AE, Delegatecall	- 최초로 상호 학습을 활용한 연구 - 두 개의 네트워크가 상호 학습을 통해 취약점을 탐지하는 새로운 접근 방식을 제안 - 성능 평가: 정밀도 87%, F1-score-84%
VulHunter[24]	Hybrid Model	RE, TD, AE 외 26가지 취약점	- 다중 인스턴스 학습 기반 프레임워크 - 분류 결과와 함께 결합 위치와 해당 인스턴스 식별 - 성능 평가: F1-score-86%
COBRA[25]	Hybrid Model	RE, AE, UE, FR, TD	- 시맨틱 컨텍스트에서 함수 입력 매개변수를 추출하기 위해 seq2seq 구조를 활용 - 성능 평가: F1-score-95%
L. Prifti et al.[27]	CNN, RNN	RE, AE, UE, AC, Other	- 여러 모델(ResNet, Inception, LSTM)과 RNN, CNN 모델의 성능을 비교 - 4가지 취약점을 제외한 기타 취약점을 Other로 표시 - 성능 평가: Micro F1-score-89%

(multi-instance learning) 기반 프레임워크이다. 저자들은 스마트 컨트랙트의 실행 흐름을 분석하기 위해 CFG를 구축하고, 각 경로를 개별 실행 인스턴스 단위로 나눴다. 이를 Bi-LSTM(Long Short-Term Memory)을 활용한 다중 인스턴스 학습을 통해 바이트코드 시퀀스의 앞뒤 맥락을 모두 학습하도록 했다. 또한 셀프 어텐션(self-attention) 메커니즘을 도입하여 중요한 부분에 높은 가중치를 부여함으로써, 모델이 취약한 경로를 집중적으로 학습하도록 설계했다. 저자들은 EtherScan, GitHub 및 다른 공개 데이터셋을 활용하여 데이터셋을 구성했다. 이를 통해 여러 취약점(RE, TD, Tx-origin, Send-transfer)을 탐지하고 VulHunter의 성능을 평가했다.

COBRA[25]의 저자들은 의미적 맥락과 함수 인터페이스 정보를 결합한 SRIF(Signatures Reverse Inference from Functions) 모델

을 제안했다. 시퀀스-투-시퀀스(seq2seq)구조를 사용한 인코더(encoder) 모델을 채택하여 스마트 컨트랙트의 의미적 맥락을 파악하고, SRIF를 통해 의미적 맥락과 함수의 고유한 식별자 역할을 하는 함수 시그니처 간의 매핑을 수행한다. 이를 통해 ABI(Application Binary Interface)가 없는 경우에도 사용된 함수 정보를 복원하여 컨트랙트 간 상호작용에 포함된 취약점을 탐지 가능하도록 설계했다. XBlock_Eth dataset[26]을 활용해 데이터셋을 구성했다.

L. Prifti et al.[27]의 저자들은 CNN과 RNN(Recurrent Neural Network) 모델을 사용하여 스마트 컨트랙트 내 취약점을 탐지했다. 입력 데이터를 쌍(pair) 단위로 나누어 EVM 바이트코드의 특징을 더 잘 파악하도록 설계했다. 또한 RNN 모델은 약 80,000개의 파라미터를, CNN 모델은 약 36,000개의 파라미터를 사용하여 상대적으로 단순한 구조로 설계했다. 다른 논문

에서 사용된 데이터셋[28]을 활용해 다른 모델들(ResNet, Inception 등)과 비교하며 성능을 평가했다.

4.2 오피코드 기반 연구

오피코드는 바이트코드가 EVM이 이해할 수 있는 명령어 형태로 변환된 것을 의미한다. 오피코드는 명령어의 시퀀스 정보를 포함하여, 이를 활용한 취약점 탐지가 주로 연구된다. 표 2에서는 표 1과 동일한 방식으로 연구를 정리하였으며, 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

TokenCheck[29]는 LSTM모델을 활용한 도구이다. LSTM은 오피코드의 시퀀스(sequence) 데이터를 학습하여 복잡한 ERC-20 토큰 컨트랙트에 특화된 취약점 탐지를 보였다. 성능 최적화를 위해 EVM 오피코드를 전 처리하여 데이터의 길이를 줄였다. 저자들은 Ethereum Public Dataset[30]을 통해 컨트랙트를 수집하고 데이터셋을 구성했다. 또한 취약점을 Suicidal, Produgal, Greedy로 분류하고 탐지했다. Suicidal은 self-destruct 명령어가 보호되지 않아 강제 종료될 수 있는 경우를 말하며, Produgal은 임의의 주소로 이더(Ether)를 보낼 수 있는 취약점이다. Greedy는 이더를 수신할 수 있으나 반환하지 못하는 상황을 말한다.

J. Su et al.[31]의 저자들은 강화학습(reinforcement learning)과 퍼징(fuzzing) 기법을 결합한 RLF(Reinforcement Learning Fuzzer)를 제안했다. RLF는 스마트 컨트랙트 퍼징을 마르코프 결정 프로세스(markov decision process)로 모델링하여 트랜잭션 시퀀스를 생성하는 강화 학습 프레임워크이다. RLF는 DQN(Deep Q-Network)을 사용하여 트랜잭션 시퀀스를 생성하는 과정에서 보상을 최대화하도록 학습한다. 저자들은 취약성 발견에 중점을 둔 보상 외에도 코드 커버리지(coverage)에 관한 보상을 추가하여 복잡한 취약점을 탐지할 수 있도록 유도했다. 또한 XBlock[32]과 VeriSmart[33]에서 컨트랙트를 수집하여 데이터셋을 구성했다. 이를 통해 UE, AC, DC 같은 주요 취약점과 EL(Ether Leaking), EF(Ether Freezing) 등 여러 취약점을 탐지했다. EF는 이더를 수신 가능하지만 출금할 수 없는 구조를 가질 경우 이더가 잠길 수

있다. EL은 잘못된 접근 제어, 설계 오류 등으로 이더가 탈취 가능한 취약점이다.

J. Huang et al.[34]의 저자들은 다중 작업 학습(Multi-Task Learning) 기반 모델을 제안했다. 다중 학습 모델은 하위 레이어와 작업 레이어로 구성된다. 하위 레이어는 입력 데이터를 처리하고, 어텐션 메커니즘을 적용하여 의미 정보를 학습한다. 작업별 레이어는 CNN을 사용하여 분류 네트워크를 구축하고, 하위 레이어에서 학습된 특징 벡터를 사용해 각 작업의 목표를 달성한다. 다중 작업 모델을 통해 여러 작업을 동시에 수행할 수 있어 단일 작업 모델에 비해 시간, 계산 비용을 절감할 수 있다. 저자들은 XBlock에서 컨트랙트를 수집하여 데이터셋을 구성하고 RE, AE, UA(Unknown Address) 취약점을 탐지했다. UA는 컨트랙트 주소로 알려지지 않은 값이 들어올 경우, 예기치 않은 결과가 발생할 수 있다.

S. Gopali et al.[35]의 저자들은 LSTM 및 TCN(Temporal Convolutional Network) 모델을 이용한 기법을 제안했다. TCN을 통해 컨트랙트 간의 종속 관계를 처리하도록 설계했다. 또한 추가적인 컨트랙트를 수집하고 오버 샘플링과 언더샘플링을 적용하여, 기존 연구들에서 사용하던 데이터셋의 불균형 문제를 개선했다. 저자들은 취약점을 Suicidal, Produgal, Greedy로 분류하고 Ethereum Public Dataset을 활용한 데이터셋으로 LSTM과 TCN의 성능을 비교했다.

Z. Wu et al.[36]의 저자들은 딥러닝 기반 방법을 제안했다. unigram과 bigram을 사용하여 스마트 컨트랙트의 오피코드를 분석하고, 특징 벡터를 생성했다. 또한 TF-IDF 방식을 적용하여 드물게 발생하는 잠재적 악성 특징을 강조했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집하고, SMOTE(Synthetic Minority Oversampling Technique) 기법을 사용해 데이터 불균형 문제를 해결했다. 이를 통해 다양한 모델(CNN, LSTM, CNN-BiLSTM, ResNets)의 성능을 평가했다.

W2V-SA[37]는 여러 모델을 결합한 하이브리드 모델이다. W2V-SA는 CNN과 셀프 어텐션을 결합하여 스마트 컨트랙트의 복잡한 관계를 파악하고 취약점을 탐지한다. 워드 임베딩 레이어에선 스마트 컨트랙트를 word2vec 방식으로 벡터화하고, 이를 학습한다. CNN 레이어는 입력 데이터에

표 2. 오피코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 2. Research on Opcode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
TokenCheck[29]	LSTM	Suicidal, Prodigal, Greedy	- LSTM 활용해 보다 복잡한 ERC-20 토큰의 취약점 탐지 방법을 제안 - 취약점을 보다 큰 3가지 범주로 분류 - 성능 평가: 정밀도-93%, F1-score-93%
J. Su et al.[31]	DNN	UE, AC, TD Ether Leaking, Delegatecall, Ether Freezing	- 강화학습과 DQN을 결합하여 새로운 퍼징 기법인 RLF를 제안 - 성능 평가: 정밀도-98.6%
J. Huang et al.[34]	Hybrid Model	RE, AE, Unknown Address	- 다중 작업 학습과 CNN 구조를 도입한 모델을 제안 - 성능 평가: 정밀도-75.55%, F1-score-79.28%
S. Gopali et al.[35]	TCN	Suicidal, Prodigal, Greedy	- 최초로 TCN을 사용한 취약점 탐지 방법을 제안 - 성능 평가: 정밀도-95.95%, F1-score-94.83%
Z. Wu et al.[36]	Many	RE, AE, FR, TD, UE	- ResNet, LSTM, KNN 등 다양한 모델의 성능을 분석 - 성능 평가: ResNets, Macro F1-score-82%
W2V-SA[37]	Hybrid Model	RE, AE, TD, UE, Assert Violation	- 서로 다른 뉴럴 네트워크 모델과 워드 임베딩 방식을 결합한 모델 제안 - 성능 평가: 정밀도-94.85%, F1-score-94%
J. Zhu et al.[38]	LSTM	RE, AC, TD, UE 외 3 가지 취약점	- 오피코드 시퀀스를 활용한 LSTM 기반 모델을 개발 - 시퀀스 길이와 취약성 유형을 조정한 데이터셋을 생성 - Missing the Transfer Event와 같은 취약점을 추가로 탐지 - 성능 평가: 정밀도-80%, F1-score-79.74%
L. Colin et al.[40]	MLP	UE, FR, TD, AC, AE	- 데이터 처리와 관련된 버그 삽입, 전처리 알고리즘 개발 - MLP와 LSTM을 기반으로 하는 다중 입력 모델과 프레임워크를 설계 - 성능 평가를 위한 MythSlith라는 새로운 도구 개발 - 성능 평가: 정밀도-91.28%, F1-score-91.27%
Y. Boxin[42]	Many	RE, UE, TD, AC, AE, Lack of Standard Events	- 동적 실행 중에 발생하는 오피코드 시퀀스를 통한 탐지 방식을 제안 - N-gram 모델과 가중치 케널링 메커니즘을 결합 - 성능 평가: SVM, F1-score-92.55%
X. Feng et al.[43]	XGBoost	RE, TD, AE, FR, Callstack Depth	- SHAP을 사용해 특징이 모델에 미치는 영향을 해석 - 해석 결과를 활용한 적응형 취약점 탐지 방식 제안 - 성능 평가: F1-score-93%

서 주요 패턴이나 특징을 찾아내고, 셀프 어텐션 메커니즘을 활용해 더 중요한 정보에 집중한다. 저자들은 이더리움 공식 페이지에서 컨트랙트를 수집하여 데이터셋을 구성하고, 여러 취약점(RE, AE, AV)을 취약점을 탐지했다.

J. Zhu et al.[38]의 저자들은 오피코드 시퀀스를 기반으로 취약점을 탐지하는 LSTM 기반 모델을 제안했다. 이를 통해 소스 코드 접근이 어려운 배포된 컨트랙트의 경우에도 오피코드를 통한 취약성 탐지가 가능하도록 설계했다. 저자들은 시퀀스 길이에 집중하여 길이를 1000, 2000, 3000으로 조정하며, 정밀도의 변화를 분석했다. 성능 평가를 위해 이더리움 메인넷에서 오피코드 시퀀스를 수집하고, SODA[39] 플러그인을 통해 라벨링하여 데이터셋을 구축했다. 이를 통해 주요 취약점 이외에 MTE(Missing the Transfer Event), SCB(Strict Check for Balance)를 포함한 7가지 취약점을 탐지했다. MTE는 전

송 후 기록을 관리하는 이벤트를 남기지 않는 경우, SCB는 이더를 전송하기 전 잔액에 대한 검증이 엄격하게 수행되지 않을 경우 발생 가능하다.

L. S. H. Colin et al.[40]의 저자들은 MLP을 포함한 여러 모델(SVM, RF, XCG)을 활용한 취약점 탐지를 수행했다. 오피코드와 CFG를 활용해 실행 패턴과 흐름 정보를 학습하도록 설계했다. 저자들은 Smart Contract Sanctuary[41]에서 컨트랙트를 수집했다. 수집한 데이터에 전처리 방법과 버그 삽입 알고리즘을 적용해 균형 잡힌 데이터셋을 구성했다. 또한 고전적인 탐지 도구(Mythril, Slither)를 결합한 MythSlith를 개발하고, 이를 성능 평가 지표로 활용했다.

Y. Boxin[42]의 저자들은 스마트 컨트랙트의 동적 실행 중 발생하는 오피코드 시퀀스를 분석하여 취약점을 탐지하는 방식을 제안했다. 오피코드 시퀀스를 N-gram을 통해 분석하여, 순차적 특징

을 효과적으로 추출했다. 또한 TF-IDF와 유사한 방식으로 가중치 페널티 메커니즘을 적용해 코드별 중요도를 설정했다. 저자들은 컨트랙트의 실행 상태를 모니터링하여 시퀀스 데이터를 수집하고, 데이터셋을 구성했다. 이를 통해 여러 주요 취약점(RE, UE, TD 등)과 LSE(Lack of Standard Events)를 탐지하고, 여러 모델(SVM, KNN, LR)의 성능을 비교했다. LSE는 주요 상태 변경을 기록하지 않거나 표준 이벤트를 따르지 않을 경우, 투명성, 신뢰성 저하를 가져올 수 있다.

X. Feng et al.[43]의 저자들은 기존의 탐지 연구들과 달리 설명 가능한 스마트 컨트랙트 취약점 탐지 모델을 위해 SHAP(Shapley Additive ExPlanations) 값을 기반으로 한 방식을 제안했다. SHAP 값을 활용해 중요한 특징만을 선택하여 학습에 활용했다. 이러한 적응형 최적화를 통해 모델의 해석 가능성을 높이고, 효율적인 학습을 가능하게 했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집해 데이터셋을 구성하고 여러 취약점(RE, TD, CD 등)을 탐지했다.

4.3 소스 코드 기반 연구

소스 코드는 사람이 이해할 수 있는 고수준 프로그래밍 언어로 작성된 코드로, 주로 솔리디티와 같은 언어를 통해 작성된다. 소스 코드는 스마트 컨트랙트의 문맥과 의미 정보를 모두 포함하고 있어, 이를 활용한 연구가 활발히 이루어지고 있다. 표 3은 표 1, 2와 같은 방식으로 연구를 정리하였으며, 각 연구의 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

Eth2Vec[44]의 저자들은 NLP(Natural Language Processing)을 활용한 기법을 제안했다. EVM Extractor 모듈을 개발해 소스 코드로부터 바이트코드의 구문적, 의미적 정보를 JSON 형태로 변환했다. Eth2Vec은 이를 PV-DM 모델의 입력으로 활용하여, 바이트코드를 구조적으로 분석하고 코드의 의미론적 유사성을 평가한다. 유사성 학습을 통해 소스 코드가 변경되더라도 취약점을 탐지할 수 있다. 저자들은 Etherscan에서 수집한 스마트 컨트랙트를 통해 데이터셋을 구성하고 취약점을 탐지했다. ERC-20 Transfer는 LSE와 유사한 취약점으로, 여러 이벤트 중 transfer 이벤트에 대한 처리를 준수하지 않은 경우이다.

Z. Liu et al.[45]의 저자들은 GNN을 활용한 새로운 접근 방식을 제안했다. 소스 코드를 그래프 형식으로 변환하고 노드 정규화 단계를 추가하여 중요 노드를 강조했다. 특히 새로운 모델인 TMPN(Temporal Message Propagation Network)을 도입해 스마트 컨트랙트 코드의 복잡한 제어 및 데이터 흐름을 학습하고 탐지를 수행했다. 또한 Etherscan과 VNT Chain[46]에서 컨트랙트를 수집하여 데이터셋을 구성하고, 이를 통해 RE, TD, Infinite Loop(IL) 취약점을 탐지했다. IL은 컨트랙트 코드 내에 반복문이 포함되어 있고 종료 조건이 명시되지 않은 경우로 DoS와 같은 문제를 일으킬 수 있다.

DeeSCVHunter[47]는 DNN 기반 프레임워크이다. 저자들은 취약성 후보 슬라이스(Vulnerability Candidate Slice, VCS)라는 새로운 개념을 도입했다. VCS를 통해 코드의 구문적, 의미적 정보를 더 잘 이해하도록 하여 모델 성능을 향상 시켰다. VCS는 구문 및 제어 흐름을 추적하여 중요한 취약성 정보를 효과적으로 학습할 수 있다. 또한 프레임워크 및 사용된 데이터셋을 공개하여, 다른 연구자들이 취약성 탐지 시스템을 구축하고 평가할 수 있도록 했다.

Peculiar[48]는 사전 학습(pre-training)된 GraphCodeBERT를 기반으로 하는 프레임워크이다. Peculiar는 기존의 방식에서 사용하는 복잡한 그래프 구조 대신 중요한 데이터 흐름(crucial data flow)만을 고려하도록 설계됐다. 저자들은 그래프 마스크 어텐션(graph masked attention) 기법을 도입하여 토큰 간 관계를 모델링하고 코드 내 변수들 간의 의존 관계를 효과적으로 학습하도록 설계했다. 또한 SmartBugs Dataset[49]을 활용해 성능을 평가했다. 평가 결과 사전 학습된 모델을 통해 높은 일반화 성능을 보였다.

CodeNet[50]은 CNN을 활용한 탐지 기법을 제안했다. 저자들은 기존 CNN 아키텍처와 달리, 스트라이드를 사용하지 않는 방식을 통해 스마트 컨트랙트 코드의 문맥과 의미를 유지하며 취약점을 탐지하도록 설계했다. 또한 분류 마지막 단계에서 GAP(Global Average Pooling)이 아닌 GMP(Global Max Pooling)을 활용해 이미지의 중요한 특징이 보존되도록 했다. 스트라이드를 사용하지 않아 발생하는 메모리 문제를 Depthwise Separable Convolution을 적

표 3. 소스 코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 3. Research on Sourcecode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
Eth2Vec[44]	NLP	RE, AE, TD, DoS, AC, ERC-20 Transfer	- 함수의 문맥을 학습, 코드의 변화에도 유연한 탐지를 제공 - 구문 정보를 신경망에 입력하는 모듈 개발 - 성능 평가: 정밀도-77%, F1-score-57.5%
Z. Liu et al.[45]	GNN	RE, TD, Infinite Loop	- 코드를 그래프 형식으로 변환, GNN을 통해 취약점을 탐지하는 방법을 제안 - 성능 평가: F1-score-85.43%
DeeSCVHunter[47]	DNN	RE, TD	- 최초의 딥러닝 모델 기반 탐지 프레임워크 - 취약성 후보 슬라이스 개념을 도입, 학습과정에서 구문 및 의미적 정보 이해능력 향상 - 성능 평가: Bi-LSTM, F1-score-86.87%
Peculiar[48]	LLM	RE	- 데이터 흐름 정보에 기반하여 컨트랙트를 표현하는 새로운 접근 방식을 제안 - 탐지에 사전 학습 기법을 활용한 최초의 접근 - 성능 평가: F1-score-92.10%
CodeNet[50]	CNN	RE, UE, TD, AC	- 기존 CNN과 달리 Stride 없는 CNN 설계 - 데이터 전처리 과정에서 코드의 지역성을 유지 - 성능 평가: F1-score-98.75%
SPCBIG-EC[51]	Hybrid Model	RE, TD, AE, Infinite Loop, Callstack Depth	- CNN과 RNN을 결합한 모델 SPCBIG-EC 제안 - CNN 구조 개선한 SPCNN를 제안, 코드 시퀀스 정보를 유지하면서 특징을 추출 - 성능 평가: 정밀도-89.09%, F1-score-90.13%
xFuzz[52]	Ensemble learning	RE, AC, Delegatecall	- 세 가지 크로스-컨트랙트 취약점을 체계적으로 정의 - 탐색 공간을 줄이기 위해 ML 모델을 사용 - 성능 평가: 정밀도-95.5%
Gvd-net[53]	Graph Embedding	AE, AC, Asset Frozen	- 변수와 함수 호출 관계를 바탕으로 비유클리드 그래프를 생성, 복잡한 관계를 반영 - Node2Vec 알고리즘을 사용해 노드 간 관계를 벡터로 임베딩 - 성능 평가: 정확도-90.5%
PSCVFinder[54]	LLM	RE, TD	- 최초로 Prompt-Tuning을 활용한 모델 제안 - CSCV라는 코드 슬라이싱 기법을 제안, 정보를 유지하면서 컨트랙트 코드를 간소화 - 성능 평가:F1-score-94%
VulDet[56]	GAT	RE, TD	- 스마트 컨트랙트 그래프와 도메인 지식을 결합 - GAT를 통해 특정 노드에 가중치 부여, 탐지 정확도를 향상 - 성능 평가: 정밀도-74.82%, F1-score-78.69%
J. Li et al.[58]	Hybrid Model	RE, TD, AE, AC	- 워드 임베딩 방법과 딥러닝 기술을 결합한 방식 제안 - 실행 중 중요한 정보를 수집하여 동적 특징을 추출, 정적 특징과 결합하여 탐지에 활용 - 성능 평가: 정밀도-83.61%, F1-score-84.98%
GPTScan[60]	LLM	FR, AC 외 8가지	- GPT와 정적 분석을 결합한 탐지 도구 제안 - Slippage, Risky First Deposit 등 다양한 취약점을 탐지 - 각 취약점 유형을 시나리오와 속성으로 세분화하여 정의, 탐지 성능을 향상 - 성능 평가: F1-score-67.8%
LLM4Fuzz[63]	LLM	RE, FR, Logical Error, Price Manipulation, Invariant Violation	- LLM 활용한 퍼징 우선순위 스케줄링 도입 - 중요한 코드 영역을 자동으로 식별 및 테스트 - 성능 평가: ITYFuzz보다 넓은 커버리지를 보임
Y. Chen et al.[64]	Transformer	RE, TD, AE	- 대조 학습을 적용한 새로운 방식을 제안 - 상관 관계 레이블을 생성하여 훈련에 사용 - 성능 평가: F1-score-94.52%
SCVHunter[65]	GNN	RE, FR, TD, DoS	- SG와 GNN을 활용한 확장 가능한 프레임워크 - 사용자가 CSG의 핵심 노드를 지정 가능 - 성능 평가: F1-score-85.65%
VulnHunt-GPT[67]	LLM	RE, AC, AE, UE, DoS, BR, FR, TD, SA	- GPT 모델을 활용해 자연어 처리 방식으로 분석하는 접근 방식 도입 - GPT 모델을 사용해 높은 확장성을 제공 - 성능 평가: 유의미한 탐지 수, 실행 시간을 보임

용하여 해결했다. 성능 평가를 위해 SmartBugs 데이터셋을 활용했다.

SPCBIG-EC[51]의 저자들은 새로운 하이브리드 모델인 SPCBIG-EC(Serial-Parallel Convolutional Bidirectional Gated Recurrent Network Model incorporating Ensemble Classifiers)를 제안했다. 이 모델은 CNN과 RNN을 결합한 구조로, 다중 변수가 결합된 특징을 효과적으로 추출할 수 있다. 또한 기존 CNN 구조를 개선한 SPCNN (Serial-Parallel CNN)을 제안하여 코드 시퀀스의 위치 정보와 구조적 정보를 효과적으로 유지하면서 특징 추출이 가능하도록 했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집하여 데이터셋을 구성하고, 여러 취약점(RE, TD, IL, CD)을 탐지했다.

xFuzz[52]는 기존 도구들이 거의 다루지 않는 크로스-컨트랙트 취약점을 탐지하기 위한 도구이다. 세 개 이상의 컨트랙트가 상호작용할 때 발생하는 취약점을 찾아내는 데 특화되어 있다. xFuzz는 머신 러닝 기법을 통해 효율적으로 경로를 필터링하고 우선순위를 부여한다. 잠재적 취약점이 있는 경로를 우선으로 테스트하여 취약점을 효과적으로 탐지할 수 있도록 설계했다. 저자들은 다양한 소스(Etherscan, SmartBugs 등)를 활용해 데이터셋을 구성하고, 여러 취약점(RE, AC, DC)을 탐지했다.

Gvd-net[53]은 스마트 컨트랙트 취약점 탐지에 특화된 그래프 임베딩 기반 모델이다. 기존의 선형 구조 대신 스마트 컨트랙트 코드의 변수 및 함수 호출 관계를 기반으로 비유클리드(non-euclidean) 그래프를 구축했다. 이를 통해 코드와 의미구조 간의 복잡한 관계를 더 잘 표현하고 탐지 성능을 향상 시켰다. node2vec을 사용해 노드로부터 벡터를 생성하고, 유사도 분석을 통해 여러 취약점을 탐지한다. Asset Frozen은 EF과와 같은 취약점으로 이더를 수신할 수 있지만, 외부로 송금할 수 없는 구조로 작성된 경우이다. 저자들은 SmartBugs 데이터셋을 활용해 모델의 성능을 평가했다.

PSCVFinder[54]는 스마트 컨트랙트 취약점 탐지에 프롬프트 튜닝(prompt tuning)을 처음으로 적용한 모델이다. 저자들은 사전 학습된 LLM 모델인 CodeT5를 사용하여 취약점을 탐지했다. CodeT5는 코드에서 중요한 식별자와 토큰

을 이해하는 데 강점을 가진 모델이다. 저자들은 성능 향상을 위해 스마트 컨트랙트 코드 슬라이싱 기법을 제안했다. 취약점 탐지와 관련 없는 불필요한 정보를 제거해 모델 성능을 개선했다. 또한 SmartBugs와 다른 연구[55]에서 사용한 데이터셋을 활용해 성능 평가를 진행했다.

VulDet[56]은 GAN(Graph Attention Network)을 사용한 새로운 모델이다. 스마트 컨트랙트의 소스 코드를 그래프로 변환하고, 모든 변수와 함수 호출을 노드로 처리한다. 저자들은 오픈 소스 데이터셋[57]을 사용해 성능을 평가했다. 그 결과 GAN을 통해 취약점과 밀접하게 연관된 노드에 추가적인 중요도를 부여해, 기존 딥러닝 기반 방식보다 우수한 성능을 보였다.

J. Li et al.[58]의 저자들 스마트 컨트랙트 정적 및 동적 특징을 결합하는 특징 결합(feature fusion) 기반 탐지 방식을 제안했다. 스마트 컨트랙트 오피코드로부터 CFG를 생성하고, 이를 GCN(Graph Convolutional Network)을 통해 학습 시켰다. GCN을 통해 스마트 컨트랙트의 구조적 관계를 학습하고, Bi-LSTM을 통해 시간적 순서를 반영하여 취약점을 예측하는 구조를 설계했다. 저자들은 SmartBugs와 SolidFI[59] 데이터셋을 활용했다. 다양한 모델(RNN, GRU, LSTM)과 비교한 결과, 저자들이 사용한 Bi-LSTM이 가장 높은 성능을 보였다.

GPTScan[60]은 GPT와 정적 분석을 결합한 탐지 도구이다. GTPScan은 취약점 유형을 구체적인 시나리오와 속성으로 분할하고, GPT를 활용해 후보 취약점을 선정한다. 저자들은 GPT 모델의 응답이 불확실성을 띠지 않도록, 동일 질문을 여러 번 묻고 가장 많이 일치한 응답을 채택하는 방식을 사용했다. GTPScan은 최종적으로 분류한 취약점을 정적 분석을 통해 검증한다. 저자들은 Web3Bugs[61], DefiHacks[62] 데이터셋을 활용했다. 이를 통해 Slippage, WCO(Wrong Checkpoint Order)와 같은 10가지 취약점을 탐지했다. Slippage는 자산 스왑(swap) 또는 유동성(liquidity) 추가 시 가격 변동으로 인해 예기치 못한 결과가 발생할 수 있다. WCO는 상태 업데이트와 실제 계산 순서가 올바르지 않을 경우 동기화 문제가 발생할 수 있다.

LLM4Fuzz[63]는 퍼징 스케줄링에 머신 러닝 기법을 도입한 도구이다. LLM4Fuzz는 LLM을 활용한 우선순위 스케줄링을 도입하여,

LLM이 생성한 지표를 바탕으로 테스트를 수행한다. 스케줄링을 통해 복잡도가 높은 함수나 코드 영역을 중점적으로 테스트하도록 설계됐다. 이러한 방식으로 기존 퍼징 도구보다 더 효율적이고 넓은 커버리지를 달성했다. LLM4FUZZ는 탐지 가능한 취약점 목록을 구체적으로 나열하지 않았지만, 논문에서 제시한 사례, 설계 원리를 통해 LE, PM(Price Manipulation), IV(Invariant Violation)와 같은 여러 취약점을 탐지한다. PM은 AMM(Automated Market Maker)에서 유동성 풀(liquidity pool)을 통해 가격을 조작할 수 있다. IV는 컨트랙트 내에서 불변 값이 변경 가능하도록 설정된 경우 토큰의 총량이 변경되는 등 여러 문제가 발생 가능하다.

Y. Chen et al.[64] 저자들은 대조 학습(Contrastive Learning)을 적용한 새로운 모델을 제안했다. 기존의 취약점 탐지 연구에서 잘 고려되지 않던 컨트랙트 간의 상관 관계를 모델링하여 활용했다. 이를 통해 컨트랙트 간의 유사성과 다른 유형 간의 차이를 파악하여 취약성 탐지 성능을 향상 시켰다. 대조 학습 단계에서 컨텍스트 증강(contextual augmentation)을 위해 MLM(Masked Language Model)을 사용했다. MLM은 입력 데이터의 일부 토큰을 무작위로 마스킹하고 주변 문맥을 바탕으로 예측하게 하여 모델이 코드의 구조적, 의미적 특징을 더 잘 이해하도록 한다. 성능 평가를 위해 이전의 연구[23]에서 사용된 데이터셋을 활용했다.

SCVHunter[65]는 스마트 컨트랙트 소스 코드를 CSG(Contract Semantic Graph)로 변환하고, GNN을 활용하여 취약점을 탐지하도록 설계됐다. 이 과정에서 사용자는 간단한 규칙 입력만으로 취약점과 관련된 핵심 노드를 지정할 수 있다. CSG는 핵심 노드, 호출 노드 등으로 구성되어 스마트 컨트랙트의 구조적 의미를 효과적으로 표현한다. SCVHunter는 노드 레벨과 경로 레벨에서 어텐션 메커니즘을 적용하여, 노드 이웃의 중요도를 학습하고 취약점 관련 노드에 더 집중하도록 한다. 저자들은 오픈소스 데이터셋[66]을 활용해 성능을 평가했다.

VulnHunt-GPT[67]의 저자들은 GPT 모델을 활용한 방식을 제안했다. VulnHunt-GPT는 기존의 고전적인 도구와 달리, LLM을 활용하여 취약점 분석을 수행한다. 프롬프트 엔지니어링을 활용하여 GPT 모델의 응답을 조정했다. 주요 취

약점 정보를 사전에 제공해 정확도를 높였다. VulnHunt-GPT는 Python 스크립트를 통해 스마트 컨트랙트 코드를 전처리하고 OpenAI API를 통해 통신한다. 성능 평가 결과 기존 탐지 도구보다 빠른 실행 시간을 보였다. 저자들은 성능 평가를 위해 SmartBugs 데이터셋을 활용했다.

V. 결 론

본 논문은 머신 러닝을 활용한 스마트 컨트랙트 취약점 탐지 연구들에 대해 조사했으며, 스마트 컨트랙트 내에 존재 가능한 대표적인 취약점 9가지에 대해 설명했다. 최근 다양한 분야에서 뛰어난 성능을 보인 머신 러닝 기법을 스마트 컨트랙트 취약점 탐지에 적용한 연구가 진행되고 있다. 본 논문에선 최신 동향 분석을 위해 2021년 이후 발표된 연구를 선정하였다. 머신 러닝을 활용한 연구에서 주로 사용되는 입력 데이터 유형인 바이트코드, 오피코드, 소스 코드를 기준으로 각 연구들을 분류하고, 각 연구의 주요 기여와 특징을 제시하였다.

바이트코드 기반 연구는 소스 코드가 비공개된 경우에도 취약점 탐지가 가능하다는 강점을 가진다. 오피코드 기반 연구는 명령어를 여러 형태(그래프, 이미지)로 변환하거나 명령어 시퀀스를 활용하는 등 다양한 방식으로 연구가 이루어진다. 소스 코드 기반 연구는 자연어 처리와 유사한 방식으로 소스 코드를 분석하여 취약점을 탐지하며, 특히 LLM을 활용한 연구가 활발히 진행되고 있다. 본 논문은 이러한 입력 데이터별 연구 동향과 주요 특징을 조사하여 제시하였다.

향후 연구로는 머신 러닝 기법을 활용한 스마트 컨트랙트 취약점 탐지 모델을 발전시키고자 한다. 스마트 컨트랙트 내에서 발생할 수 있는 새로운 유형의 취약점이 지속적으로 등장하고 있어, 해결하기 위한 탐지 모델 연구가 필수적이다. 기존 연구가 주로 알려진 취약점 탐지에 집중된 한계가 있는 만큼, 새로운 유형의 취약점을 효율적으로 탐지할 수 있는 모델 개발을 목표로 연구를 진행할 계획이다.

또한 현재 연구들에서 사용되는 데이터셋들이 대부분 표준화되지 않아 비교 및 재현성 확보에 어려움이 있다. 이러한 문제를 해결하기 위해 스마트 컨트랙트 데이터셋의 표준화 방안을 함께 연구할 계획이다. 표준화된 데이터셋은 다양한 연구 간 성

능 비교를 용이하게 하고, 연구 결과의 신뢰성을 높이며, 스마트 컨트랙트 취약점 탐지 기술의 발전을 촉진할 수 있을 것으로 기대된다.

References

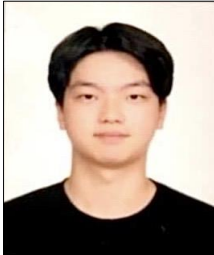
- [1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System(2009), Retrieved Nov. 14, 2024, from <https://bitcoin.org/bitcoin.pdf>.
- [2] The Parity Wallet Hack Explained, Retrieved Nov. 14, 2024, from <https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7>
- [3] Top 17 Smart Contract Hacks in 2021-2022, Retrieved Nov. 14, 2024, from <https://4irelabs.com/articles/top-17-smart-contract-hacks/>
- [4] M. I. Mehar et al., "Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack," Journal of Cases on Information Technology, Vol. 21, No. 1, pp. 19 - 32, Feb. 2019.
- [5] Cream Finance Exploit, Retrieved Nov. 14, 2024, from <https://www.immunebytes.com/blog/cream-finance-exploit-oct-27-2021-detailed-analysis>.
- [6] R. Kiani and V. S. Sheng, "Ethereum smart contract vulnerability detection and machine learning-driven solutions: a systematic literature review," Electronics, Vol. 13, No. 12, pp. 1 - 23, Jun. 2024.
- [7] C. D. Baets et al., "Vulnerability detection in smart contracts: a comprehensive survey," arXiv preprint arXiv:2407.07922, pp. 1 - 46, Jul. 2024.
- [8] Z. A. Khan and A. S. Namin, "A Survey of Vulnerability Detection Techniques by Smart Contract Tools," IEEE Access, Vol. 12, pp. 70870 - 70910, May, 2024
- [9] T. Jiao et al., "A Survey of Ethereum Smart Contract Security: Attacks and Detection," Distributed Ledger Technologies: Research and Practice, Vol. 3, No. 3, pp. 1 - 28, Sep. 2024.
- [10] N. Szabo, "Formalizing and securing relationships on public networks," First Monday, Vol. 2, No. 9, Sep. 1997.
- [11] Solidity, Retrieved Nov. 14, 2024, from <https://docs.soliditylang.org/en/lates>
- [12] V. Buterin, Vyper Documentation(2018), Retrieved Nov. 14, 2024, from <https://readthedocs.org/projects/vyper/downloads/pdf/v0.1.0-beta.8/>
- [13] DASP - TOP 10, Retrieved Nov. 14, 2024, from <http://dasp.co/>
- [14] O. Lutz et al., "Escort: ethereum smart contracts vulnerability detection using deep neural network and transfer learning," arXiv preprint arXiv:2103.12607, pp. 1 - 17, Mar. 2021.
- [15] F. Mi et al., "VSCL: automating vulnerability detection in smart contracts with deep learning," in Proc. of the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, pp. 1 - 9, May 2021.
- [16] J. Huang et al., "Hunting vulnerable smart contracts via graph embedding based bytecode matching," IEEE Transactions on Information Forensics and Security, Vol. 16, pp. 2144 - 2156, Jan. 2021.
- [17] Mist, Retrieved Dec. 9, 2024, from <https://github.com/ethereum/mist>
- [18] Etherscan, Retrieved Dec. 9, 2024, from <https://etherscan.io/>
- [19] Y. Sun and G. Lize, "Attention-based machine learning model for smart contract vulnerability detection," Journal of physics: conference series, Vol. 1820. No. 1, pp. 1 - 6, 2021.
- [20] Z. Liao et al., "SmartDagger: a bytecode-based static analysis approach for detecting cross-contract vulnerability," in Proc. of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), South Korea, pp. 752 - 764, Jul. 2022.
- [21] S. J. Qin et al., "Smart contract vulnerability detection based on critical combination path and deep learning," in Proc. of the 2022 12th International Conference on Communication and Network Security, Beijing, China, pp. 219 - 226, Dec. 2022.
- [22] Awesome-Buggy-Erc20-Tokens, Retrieved Dec. 9, 2024, from <https://github.com/sec-bit/awesome-buggy-erc20-tokens>
- [23] P. Qian et al., "Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode," in Proc. of the ACM Web Conference 2023, Austin, USA, pp. 2220 - 2229, Apr. 2023.
- [24] Z. Li et al., "VulHunter: Hunting Vulnerable Smart Contracts at EVM bytecode-level via Multiple Instance Learning," IEEE Transactions on Software Engineering, Vol. 49, No. 11, pp. 4886 - 4916, Nov. 2023.
- [25] W. Li et al., "COBRA: Interaction-Aware Bytecode-L

- level Vulnerability Detector for Smart Contracts,” in Proc. of the 39th IEEE/ACM International Conference on Automated Software Engineering, Sacramento, USA, pp. 1358 - 1369, Oct. 2024.
- [26] P. Zheng et al., “Xblock-eth: Extracting and exploring blockchain data from ethereum,” IEEE Open Journal of the Computer Society, Vol. 1, pp. 95 - 106, May 2020.
- [27] L. Prifti et al., “Smart Contract Vulnerability Detection Using Deep Learning Algorithms on EVM bytecode,” in Proc. of the 2024 13th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, pp. 1 - 7, Jun. 2024.
- [28] M. Rossini et al., “Smart contracts vulnerability classification through deep learning,” in Proc. of the 20th ACM Conference on Embedded Networked Sensor Systems, Boston, USA, pp. 1229 - 1230, Nov. 2022.
- [29] S. Goswami et al., “TokenCheck: towards deep learning based security vulnerability detection in ERC-20 tokens,” in Proc. of the 2021 IEEE Region 10 Symposium (TENSYP), Jeju, Korea, pp. 1 - 8, Aug. 2021.
- [30] Ethereum Public Dataset, Retrieved Dec. 9, 2024, from https://bigquery.cloud.google.com/dataset/bigquery-publicdata:ethereum_blockchain
- [31] J. Su et al., “Effectively generating vulnerable transaction sequences in smart contracts with reinforcement learning-guided fuzzing,” in Proc. of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, USA, pp. 1 - 12, Oct. 2022.
- [32] XBlock, Retrieved Dec. 9, 2024, from <https://xblock.pro/#/>
- [33] VeriSmart, Retrieved Dec. 9, 2024, from <https://github.com/kupl/VeriSmart-benchmarks>
- [34] J. Huang et al., “Smart contract vulnerability detection model based on multi-task learning,” Sensors, Vol. 22, No. 5, pp. 1 - 24, Feb. 2022.
- [35] S. Gopali et al., “Vulnerability detection in smart contracts using deep learning,” in Proc. of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, USA, pp. 1249 - 1255, Jun. 2022.
- [36] Z. Wu et al., “Detecting Vulnerabilities in Ethereum Smart Contracts with Deep Learning,” in Proc. of the 2022 4th International Conference on Data Intelligence and Security (ICDIS), Shenzhen, China, pp. 55 - 60, Aug. 2022.
- [37] Z. Xu et al., W2V-SA: A Deep Neural Network-based Approach to Smart Contract Vulnerability Detection(2023), Retrieved Nov. 14, 2024, from <https://ceur-ws.org/Vol-3387/paper19.pdf>
- [38] J. Zhu et al., “Opcode Sequences-Based Smart Contract Vulnerabilities Detection Using Deep Learning,” in Proc. of the 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, United Kingdom, pp. 284 - 291, Nov. 2023.
- [39] T. Chen et al., “SODA: A Generic Online Detection Framework for Smart Contracts,” in Proc. of Network and Distributed Systems Security (NDSS), San Diego, USA, pp. 1 - 17, Feb. 2020.
- [40] L. S. H. Colin et al., “An Integrated Smart Contract Vulnerability Detection Tool Using Multi-layer Perceptron on Real-time Solidity Smart Contracts,” IEEE Access, Vol. 12, pp. 23549 - 23567, Feb. 2024.
- [41] Smart Contract Sanctuary, Retrieved Dec. 9, 2024, from <https://github.com/tintinweb/smart-contract-sanctuary>
- [42] Y. Boxin, “Research on Dynamic Detection of Vulnerabilities in Smart Contracts Based on Machine Learning,” in Proc. of the 2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, pp. 219 - 223, Feb. 2024.
- [43] X. Feng et al., “An Interpretable Model for Large-Scale Smart Contract Vulnerability Detection,” Blockchain: Research and Applications, Vol. 5, No. 3, pp. 1 - 10, Sep. 2024.
- [44] N. Ashizawa et al., “Eth2Vec: Learning Contract-Wide Code Representations for Vulnerability Detection on Ethereum Smart Contracts,” in Proc. of the 3rd ACM international symposium on blockchain and secure critical infrastructure, Hong Kong, pp. 47 - 59, Jun. 2021.
- [45] Z. Liu et al., “Combining graph neural networks with expert knowledge for smart contract vulnerability detection,” IEEE Transactions on Knowledge and Data Engineering, Vol. 35, No. 2, pp. 1296 - 1310, Jul. 2021.

- [46] VNT Chain, Retrieved Dec. 9, 2024, from <https://github.com/vntchain/go-vnt>
- [47] X. Yu et al., "Deescvhunter: A deep learning-based framework for smart contract vulnerability detection," in Proc. of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, pp. 1 - 8, Jul. 2021.
- [48] H. Wu et al., "Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques," in Proc. of the 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), Wuhan, China, pp. 378 - 389, Oct. 2021.
- [49] J. F. Ferreira et al., "Smartbugs: A framework to analyze solidity smart contract," in Proc. of the 35th IEEE/ACM international conference on automated software engineering, New York, pp. 1349 - 1352, Dec. 2020.
- [50] S. J. Hwang et al., "CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection," IEEE Access, Vol. 10, pp. 32595 - 32607, Mar. 2022.
- [51] L. Zhang et al., "SPCBIG-EC: a robust serial hybrid model for smart contract vulnerability detection," Sensors, Vol. 22, No. 12, pp. 1 - 32, Jun. 2022.
- [52] Y. Xue et al., "xfuzz: Machine learning guided cross-contract fuzzing," IEEE Transactions on Dependable and Secure Computing, Vol. 21, No. 2, pp. 515 - 529, Jun. 2022.
- [53] Z. Wang et al., "Gvd-net: Graph embedding-based machine learning model for smart contract vulnerability detection," in Proc. of the 2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMINT), Xi'an, China, pp. 99 - 103, Sep. 2022.
- [54] L. Yu et al., "Pscvfinder: a prompt-tuning based framework for smart contract vulnerability detection," in Proc. of the 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), Florence, Italy, pp. 556 - 567, Oct. 2023.
- [55] Z. Liu et al., "Smart contract vulnerability detection: from pure neural network to interpretable graph feature and expert pattern fusion," arXiv preprint arXiv:2106.09282, pp. 1 - 9, Jun. 2021.
- [56] F. Luo and R. Luo, "VulDet: Smart Contract Vulnerability Detection Based on Graph Attention Networks," in Proc. of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Zibo, China, pp. 88 - 92, Jun. 2021.
- [57] Y. Zhuang et al., "Smart contract vulnerability detection using graph neural networks," in Proc. of the 2023 2nd International Conference on Artificial Intelligence and Blockchain Technology (AIBT), Yokohama, Japan, pp. 3283 - 3290, Jan. 2021.
- [58] J. Li et al., "A Smart Contract Vulnerability Detection Method Based on Multimodal Feature Fusion and Deep Learning," Mathematics, Vol. 11, No. 23, pp. 1 - 23, Nov. 2023.
- [59] C. S. Yashavant et al., "Scrawld: A dataset of real world ethereum smart contracts labelled with vulnerabilities," arXiv preprint arXiv:2202.11409, pp. 1 - 5, Feb. 2022.
- [60] Y. Sun et al., "Gptscan: Detecting logic vulnerabilities in smart contracts by combining gpt with program analysis," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp.1 - 13, Apr. 2024.
- [61] Web3Bugs, Retrieved Dec. 9, 2024, from <https://github.com/ZhangZhaoSJTU/Web3Bugs>
- [62] DeFi Hacks, Retrieved Dec. 9, 2024, from <https://woted-meter-1d8.notion.site/0e85e02c5ed34df3855ea9f3ca40f53b?v=22e5e2c506ef4caeb40b4f78e23517ee>
- [63] C. Shou et al., "Llm4fuzz: Guided fuzzing of smart contracts with large language models," arXiv preprint arXiv:2401.11108, pp. 1 - 16, Jan. 2024.
- [64] Y. Chen et al., "Improving Smart Contract Security with Contrastive Learning-based Vulnerability Detection," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp. 1 - 11, Apr. 2024.
- [65] F. Luo et al., "Scvhunter: Smart contract vulnerability detection based on heterogeneous graph attention network," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp. 1 - 13, Apr. 2024.
- [66] T. Durieux et al., "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in Proc. of the the ACM/IEEE 42nd International conference on software engineering, Seoul, Korea, pp. 530 - 541, Jun. 2020.

- [67] B. Boi et al., “VulnHunt-GPT: a Smart Contract vulnerabilities detector based on OpenAI chatGPT,” in Proc. of the 39th ACM/SIGAPP Symposium on Applied Computing, Avila, Spain, pp. 1517 - 1524, Apr. 2024.

이 의 찬 (Ui-Chan Lee)



2019년 3월~현재 : 강원대학교
컴퓨터정보통신공학 학사

<관심분야> 네트워크 관리, 블록체인, 정보보안

방 지 원 (Jiwon Bang)



2018년 2월 : 강원대학교 컴퓨터과학과 졸업
2020년 2월 : 강원대학교 컴퓨터과학과 석사
2020년 3월~현재 : 강원대학교 빅데이터메디컬융합학과 박사

<관심분야> 네트워크 관리, 블록체인, 정보보안

최 미 정 (Mi-Jung Choi)



1998 : 이화여자대학교 공학사
2000 : 포항공과대학교 공학석사
2004 : 포항공과대학교 공학박사
2004 ~ 2005 : 프랑스 INRIA 연구소 박사후 연구원

2005 ~ 2006 : 캐나다 워털루대학 박사후 연구원
2006 ~ 2008 : 포항공과대학 컴퓨터공학과 연구 조교수
2008 ~ 현재 : 강원대학교 컴퓨터학부 컴퓨터과학전공 교수
<관심분야> 네트워크 관리, 블록체인, 정보보안, 악성코드 유포

K-P2PLab: P2P 네트워크 토폴로지 분석을 위한 테스트베드 및 분석 플랫폼 개발

이 성 욱*, 주 흥 택°

K-P2PLab: Development Testbed and Analysis Platform for P2P Network Topology Analysis

Sung-wook Lee*, Hong-taek Ju°

요 약

본 논문은 다수의 노드가 연결된 P2P 네트워크를 생성하고 실행하는 P2P 네트워크 테스트베드를 만들고 만들어진 P2P 네트워크의 토폴로지 분석이 가능한 플랫폼 개발 결과를 제시한다. P2P 네트워크의 토폴로지는 노드들이 연결되어 있는 모양으로 P2P 네트워크의 성능에 영향을 미치는 중요한 요소이나 모든 노드의 연결 정보를 수집해야 토폴로지를 알 수 있어 실제 네트워크에서는 파악하기 매우 어렵다. 따라서 테스트베드를 구축하여 실시간으로 토폴로지의 변화나 특성을 분석하는 플랫폼이 필요하다. 우리는 공개 소프트웨어 중에서 높은 기술 수준과 많이 활용되고 있는 libp2p와 가상화 기술인 도커(Docker)를 활용하여 P2P 네트워크를 만들고 실행하며 각 노드의 연결 정보를 실시간으로 수집하여 토폴로지를 분석할 수 있는 플랫폼을 개발하였다. 개발한 결과를 검증하기 위하여 P2P 네트워크를 만들고 토폴로지를 분석한 결과를 제시하였다.

Key Words : P2P, Blockchain, Topology

ABSTRACT

This paper creates a P2P network testbed that creates and executes a P2P network in which a number of nodes are connected and presents the results of the development of a platform that can analyze the topology of the created P2P network. The topology of a P2P network is an important factor that affects the performance of the P2P network in the form of connecting nodes, but it is very difficult to grasp in a real network because the topology can be known only when all nodes' connection information is collected. Therefore, a platform that analyzes topology changes or characteristics in real-time by building a testbed is needed. We developed a platform that can analyze the topology by creating and executing a P2P network by using a high technology level among public software, the widely used libp2p, and the virtualization technology Docker, and collecting connection information of each node in real time. To verify the developed results, a P2P network was created and the results of topology analysis were presented.

※ 이 논문은 2023년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 이공분야기초연구사업임. (No. NRF-2023R1A2C2006045).

♦ First Author : Keimyung University Department of Computer Engineering, limos1479@gmail.com

° Corresponding Author : Keimyung University Department of Computer Engineering, juht@kmu.ac.kr

논문번호 : KNOM2024-02-08, Received December 6, 2024; Revised December 17, 2024; Accepted December 24, 2024

1. 서론

P2P 구조는 전통적인 클라이언트-서버 구조와 대비되는 네트워크 구조로, P2P 구조에서는 네트워크의 모든 참여자가 동등한 역할과 기능을 수행하는 피어가 된다. 네트워크를 주도하는 별도의 서버 없이 피어 간에 자율적으로 데이터를 주고받는 형태로 네트워크를 구축하게 되고, 이러한 P2P는 블록체인의 기술의 기반이 된다.

블록체인은 Satoshi Nakamoto가 제안한 분산형 디지털 원장 기술로, 신뢰 가능한 데이터 기록 및 관리를 구현하는 탈중앙화 된 네트워크이다. 블록체인은 블록이라는 형태로 데이터를 저장하는데, 이는 암호학적 해시를 통해 이전 블록과 연결되어 P2P 네트워크상에 연속적이고 변경 불가능하게 저장된다. 블록체인의 분산형 디지털 원장을 구현하기 위해 체인에 기록될 모든 데이터는 네트워크에 속한 다른 참여자에게 전파된다. 블록체인은 그 종류에 따라 다양한 종류의 데이터를 네트워크상에서 수집하여 블록에 작성하고, 이것이 네트워크에 다시 전파되어 PoW, PoS 등의 합의 알고리즘을 통해 체인에 영구적으로 연결된다. 블록체인은 P2P 네트워크를 통해 체인에 기록될 블록을 공유하고 합의과정을 수행한다. 모든 참여자 즉, 노드(=피어, P2P 수준에서는 피어, 호스트 수준에서는 노드라는 단어를 사용)는 데이터(블록과 트랜잭션)를 P2P 네트워크를 통해 전파되어 공유되고 모두가 검증하여 올바른 데이터임을 보장받아 탈중앙화된 구조를 구현하고 있다[1-2].

블록체인은 P2P 네트워크를 활용한 응용으로 활용가치가 높아져서 현재 많은 노드가 참여하여 규모가 매우 커져서 수만 개의 노드가 P2P 네트워크에 참여하고 있다. 블록체인 P2P 네트워크가 커짐에 따라 여러가지 문제가 발생하고 있는데, 대표적인 잘 알려진 문제가 바로 블록체인의 3도논법(Trilemma Problem) 문제이다[3]. 이것은 탈중앙화, 보안, 확장성을 모두 만족하는 해결방법이 존재하지 않는다는 문제이다. 블록체인 네트워크의 규모가 커짐에 따라 전체 노드가 블록과 트랜잭션을 공유하고 합의 시간이 오래 걸려 처리하는 초당 처리하는 트랜잭션의 수가 감소하여 확장성이 떨어지고 확장성을 높이기 위해서 특정 노드가 중요한 역할을 수행하면 탈중앙성이 떨어지게 된다. 합의 과정을 단순화시켜서 보안을 감소시키면 탈중앙화와 확장성을

증가시킬 수는 있다.

블록체인의 3도논법 문제를 해결할 수 있는 방안 중 하나는 블록체인 P2P 네트워크의 성능을 높이는 것이다. P2P 성능이 높아지면 블록이나 트랜잭션 전파가 빨라지고 합의 과정이 짧은 시간에 수행되어 초당 처리 트랜잭션 수를 증가시킬 수 있다. 따라서 현재의 보안과 탈중앙성을 유지하면서 확장성을 증가시킬 수 있는 방법이다[4].

블록체인 상의 트랜잭션 및 블록 전파는 브로드캐스팅 형태의 통신방법으로 이루어지고 있으며, 따라서 브로드캐스팅의 성능을 개선함으로써 전파 지연을 감소시킬 수 있다. 브로드캐스팅의 성능을 개선하기 위해서는 다양한 전파 방법, 매개 변수 및 알고리즘의 수정과 실험을 통해 결과를 도출하고, 이를 분석하여 각 실험의 브로드캐스팅 성능을 측정해야 한다. 그러나 전파 지연과 같은 지표를 실제 동작 중인 블록체인 네트워크에서 측정하기는 이전 연구를 통해 매우 어렵다는 것을 이미 알고 있다[5-7]. 그러므로 브로드캐스팅을 실험하고 성능을 측정하기 위한 실제 P2P 네트워크와 같은 방식으로 동작하는 실험 환경 구축이 필요하다.

우리는 이더리움 등에 사용되는 대표적인 P2P 라이브러리인 Go-libp2p[8]와 Docker[9] Swarm 네트워크를 사용해 외부와 분리된 독립된 가상의 P2P 네트워크를 구축하는 P2P 테스트베드, K-P2PLab을 개발하였다. K-P2PLab에서는 다수의 P2P 노드를 만들고 서로 연결하며 메시지를 브로드캐스팅하게 할 수 있다. 각 노드는 도커 네트워크상에서 각 컨테이너마다 하나의 피어로 실행되며 실험 목적에 따라 다양한 형태의 P2P 네트워크를 구축할 수 있다. 또한 구축된 P2P 네트워크에서 다양한 형태의 브로드캐스팅 메시지 전달이 가능하도록 하였다.

K-P2PLab은 Prometheus[10] 및 Grafana[11]를 활용하여 P2P네트워크를 실시간으로 모니터링하고, 파일 형태의 로그 수집해서 P2P 네트워크의 토폴로지, 브로드캐스팅 메시지 전달 과정 및 성능을 분석할 수 있다. 로그 데이터 수집 및 분석도 도커 기술을 활용하여 네트워크의 크기 및 성능 모니터링 데이터의 수집량에 따라서 탄력적으로 모니터링 및 분석 성능을 조절할 수 있게 하였다. 또한 도커를 사용하여 클러스터링 된 서버 내에서 실험 환경의 조절을 용이하게 하였다. 또한 이후에 추가될 네트워크 트래픽 지연의 구현에서도 도커를 사용하면 간단하게 해당 기능을 구현할 수 있다.

위 방법은 실제 네트워크를 간단하게 모방하고

실험할 수 있는 이점이 존재하나, 현재로서는 현실에서의 네트워크 지연이나 여러 네트워크 공격 등을 완벽히 구현하지 못한다는 한계점 등이 있다. 그럼에도 P2P 가상 실험 환경의 구현에 있어, 실험 환경 속 각 피어들이 모두 현실의 P2P 네트워크상의 피어와 같은 모듈을 사용하여 동일한 방식으로 동작하기에, 위 실험 환경에서 브로드캐스팅의 성능과 네트워크의 메시지 라우팅 경로 등을 측정하여 실제 네트워크를 추정 가능하게 한다. 이를 통해 P2P 네트워크의 전파 지연을 개선하기 위해 어떤 방법이 가능할지 소규모 네트워크에서 빠르고 안전하게 미리 적용해보고 테스트해 볼 수 있다는 가치를 지닌다. 이렇게 실험된 아이디어가 실제 환경에 적용되면, 우리의 테스트베드가 정확하게 동작한다는 가정하에 테스트베드 상의 성능 개선과 유사한 성능 개선을 보일 가능성이 높고, 이는 실제 P2P 네트워크 전체를 수정하는 리스크의 발생 횟수를 줄이는 데에 기여할 수 있다.

우리가 개발한 실험 환경에서 P2P 네트워크가 올바르게 작동하고 이 P2P 네트워크의 성능을 충분하고 정확하게 모니터링하고 분석할 수 있음을 P2P 네트워크 토폴로지 분석과 노드의 차수 분석을 시각화하여 검증하였다. 그리고 우리는 K-P2PLab을 통해 현재 libp2p의 연결 방식의 복잡성과 개선의 필요성을 발견할 수 있었다.

2장에서는 우리와 비슷한 목적을 가지고 플랫폼

을 개발한 관련된 연구에 대해서 언급할 것이다. 3장에서는 K-P2PLab을 개발하기 위해 겪은 시행 착오와 서버 단위의 네트워크 관점의 아키텍처에 대해서 소개한다. 이어지는 4장에서는 K-P2PLab의 노드 단위 애플리케이션과 컨테이너의 구조에 대해서 소개하고, 5장에서는 우리가 개발한 K-P2PLab을 검증하고, 직접 활용하여 알게 된 점에 대해서 기술한다. 마지막으로 6장은 결론과 향후 연구에 대해서 언급하며 본 논문을 마친다.

II. 관련 연구

이전에도 Whiteblock[12], Testground[13]와 P2P 네트워크와 성능을 모니터링하기 위한 도구가 있었다. 기존의 도구들 역시 P2P 네트워크의 테스트 환경 생성 및 해당 환경 속 각 피어의 로그를 수집하는 것이 가능하였다. Whiteblock은 외부에서 복제한 별도의 libp2p 프로젝트를 사용하여 Gossipsub의 전파 성능을 90개 내외의 노드를 가진 단일 호스트에서 실험하는 내용이다. 전체 프로젝트가 하나의 도커 컨테이너 내에서 여러 개의 P2P 어플리케이션 프로세스가 통신하는 방법이라 실제 물리적 서버의 성능에 의해 노드의 수를 증가시키는 데에 한계가 분명하다. Testground는 이보다는 이후의 프로젝트이며, 설정 파일을 사용하여 여러 개의 도커 컨테이너를 생성하여 실행할 수 있다. 여러 개의 도커 컨테이너에 각 하나씩 P2P 어플리케이션이 생성되어

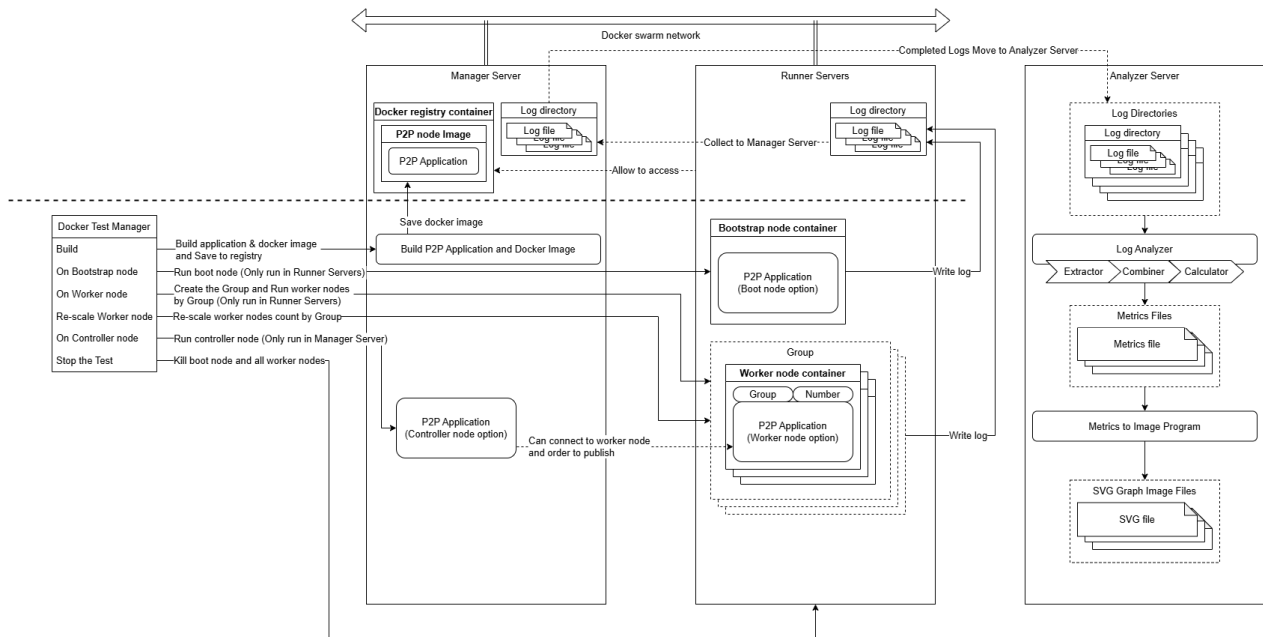


그림 1. K-P2PLab 구조도
Fig. 1. Architecture of K-P2PLab

P2P 네트워크를 형성하고 로그를 생성하고 전체 테스트베드 관리자가 이를 수집하고 분석하는 우리의 K-P2PLab과 유사한 구조를 가지고 있다. 하지만 local:docker 설정을 통한 단일 호스트의 배포만이 남아있고, 스웜 네트워크를 활용하여 클러스터링 여러 개의 된 서버를 활용하는 swarm:docker 설정은 우리가 확인한 바로는 과거에 지원이 종료된 상황이라 현재 상황으로는 제대로 사용하기 어려웠다.

Testground와 Whiteblock은 본 연구의 목적과 같은 목적으로 만들어진 도구들이고 사용한 기술도 동일하다. 도구의 사용법에 대한 설명이 잘 갖추어져 있고 공개 프로젝트로서 소스코드도 공개되어 있다. 그러나 이 두 도구는 기술적인 세부 내용을 자세히 기술한 기술 문서가 부족하다. 또한 Testground와 Whiteblock은 주요 핵심 소스 코드가 2년전 이후 더 이상 개선되거나 변경되지 않고 있고 이 두 도구를 개발한 세부 기술적인 내용이 논문으로 발표되지도 않았다.

우리도 초기에는 이미 구현된 Testground의 기술을 사용하여 P2P 네트워크 테스트 환경의 개발을 시도하였으나, 이를 위해 참고할만한 자료나 기술적 구현에 관한 내용을 쉽게 찾을 수 없었고 최신 리눅스 시스템에서 사용하기에 부적절했다. 또한 스웜 네트워크를 사용하기 위해 프로젝트를 수정하는 데에도 어려움이 있었고, 기술적 구현에 관한 내용이 연구의 성과가 우리가 아는 한 논문이나 백서의 형태로 정리되어 발표되지 않았으며, 학술적 논의가 부족하여 기존의 도구를 최신화하고 향후 연구를

위한 개선에 어려움이 있었다. 또한 개발자들의 연락처나 자문을 구하는 데에도 어려움이 있었고, 문제에 대한 응답을 얻을 수 없었다.

따라서 우리는 P2P 네트워크와 브로드캐스팅 성능 분석이 가능한 플랫폼의 제공뿐만 아니라, 최종적으로는 우리 이후에 있을 P2P 네트워크에 대한 향후 연구까지 고려하는 것을 목표로 한다. 그렇기에 우리의 K-P2PLab의 사용 방법과 아키텍처에 대한 설명, 실험의 목적과 실제 실험 환경에 따른 목적에 따라 프로그램의 수정을 도울 방법 등을 명세화하고 본 논문에서도 이에 대한 기본적인 토대를 다룰 예정이다. 그리고 여기서 더 나아가 기존의 위 연구에서 개발된 도구와 우리의 K-P2PLab을 차별화하기 위해 기존과 다른 접근 방법을 채택하였다. 우리의 도구가 기존의 도구와 가지는 차이는 한마디로 “하드웨어 효율성”으로 스웜 네트워크를 손쉽게 사용할 수 있도록 지원하여 실험자의 하드웨어 상황에 따라 최대 효율로 실험을 가능하게 하는 것이었다. 이에 대해서는 3장에서 다룰 예정이다.

III. 설계 및 구현

K-P2PLab의 최종 목표는 여러가지 브로드캐스팅 알고리즘 개선을 위한 아이디어를 실험할 수 있는 환경 개발이다. 이를 위해, K-P2PLab은 가상의 P2P 환경을 구축하고 해당 환경에서 전파 성능을 실험한 뒤, 데이터를 수집하여 P2P 네트워크를 분석하고 모니터링하는 것을 기본 목표로 하여 설계되었

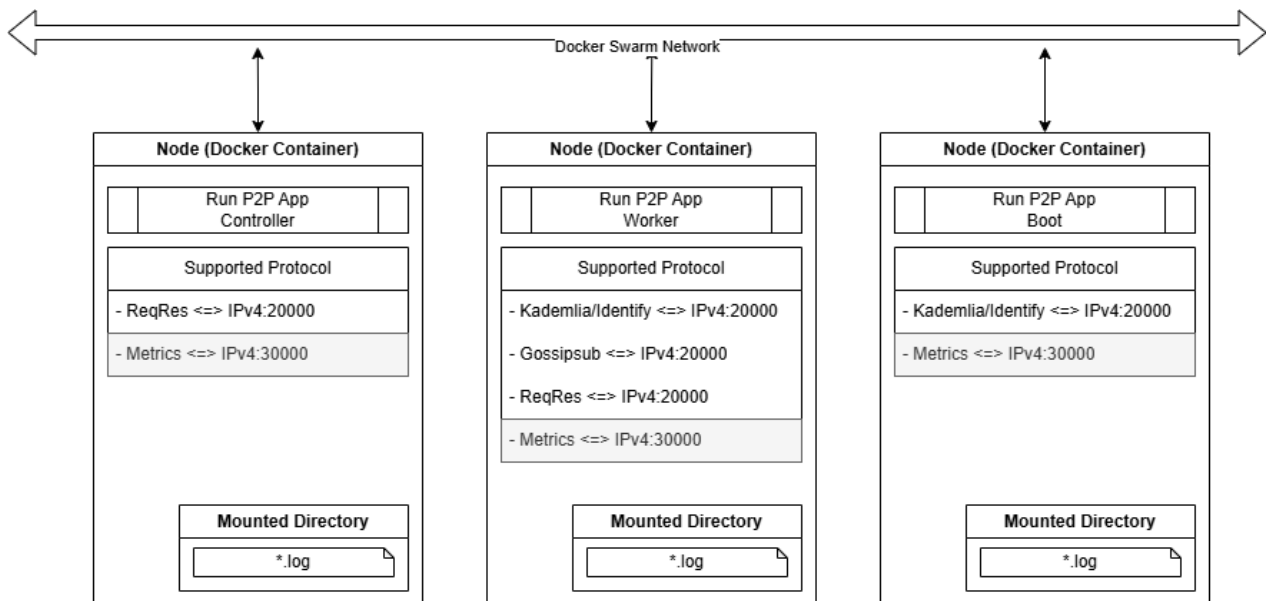


그림 2. K-P2PLab의 노드 단위 프로토콜
Fig. 2. Node-by-Node Protocol for K-P2PLab

다.

기술적으로 노드를 완전히 별도의 호스트로 분리하는 것이 가장 좋으나, 물리적 한계 및 관리의 편리함을 위해 도커의 컨테이너 기술을 사용하여 가상화 된 호스트 즉, 도커 컨테이너 상에 모든 노드를 분리하였다. 또한 P2P 피어는 현재 이더리움 네트워크 등에서 사용되고 있는 가장 대표적인 P2P 라이브러리인 libp2p를 사용하였으며[14], 이더리움의 구현과 동일하게 해당 라이브러리의 Go 언어 버전을 사용하여 P2P 어플리케이션을 구현하였다.

그림 1은 K-P2PLab의 전체적인 구조를 보여준다. 실험 환경은 전체 실험을 관리하는 하나의 Manager 서버와 확장 가능한 한 개 이상의 Runner 서버로 구축되며, 양 서버에서 Boot, Worker, Controller 3가지 형태의 피어가 동작하며 가상의 P2P 환경이 형성된다. 매니저 서버는 도커 네트워크 전체의 클러스터링 매니저 서버이며, 동시에 도커 이미지 레지스트리를 관리하고 러너 서버로부터 로그 파일을 수집한다. 그 외 Prometheus를 비롯한 필요에 따른 기타 분석 도구의 컨테이너가 동작되기도 한다. 초기 설정이 완료된 이후 매니저 서버에서 Build를 실행하게 되면, P2P 어플리케이션 도커 이미지가 빌드되어 미리 실행해둔 로컬 도커 레지스트리에 저장된다. 이 이미지는 이후 부트 노드나 워커 노드를 실행할 때 사용되며, 실행된 각 노드는 도커 스웸 네트워크를 통해 러너 서버에서 작동하게 된다. 컨트롤러 노드는 매니저 서버에서 작동하

여 워커 노드를 선택하여 퍼블리시 명령을 내릴 수 있도록 구성되어 있으며, 매니저 서버에서 실행된다. 그 외에도 워커 노드는 그룹이라는 개념과 각 그룹 안에서 노드의 번호를 부여받는데, 그룹 단위의 노드 개수 조정과 그룹별로 실험자가 원하는 행동 양식에 맞게 움직이도록 설계할 수 있다.

각 노드는 동작하며 자신의 행동에 대한 로그를 남기고 이는 매니저 서버로 수집된다. 매니저 서버에 수집된 로그는 별도의 분석 서버에서 K-P2PLab Analyzer를 통해 Metrics의 형태로 가공되며, 여기에는 시간별 P2P 네트워크의 다양한 토폴로지 정보를 알아보기 쉽게 가공하고 표현해주는 뷰어까지 포함되어 있다.

기존에 위에서 언급한 Whiteblock 및 Testground 두 도구를 비롯한 대부분의 P2P 실험 환경 생성 도구는 로그의 수집이나 노드에 브로드캐스팅 명령을 전달하는 것을 구현하기 위해 REST API를 주로 사용하였다. 이는 K-P2PLab을 구현하는 데에도 영향을 주었고, 초기에 K-P2PLab 역시 REST API를 사용한 위와 유사한 방식으로 구현되었다. 그러나, 구현의 경험상 REST API의 사용은 피어의 리소스 사용량을 늘리고 한 서버 당 노드의 최대 숫자를 감소시키는 영향이 있었다. 우리는 동일한 소형 실험 PC에서 REST API를 사용할 때와 그렇지 않을 때 같은 형태로 배포할 수 있는 최대 피어의 개수가 크게 변화하는 것을 볼 수 있었다. 실험적으로 REST API의 사용 시 최대 30개의 피어를 별도의

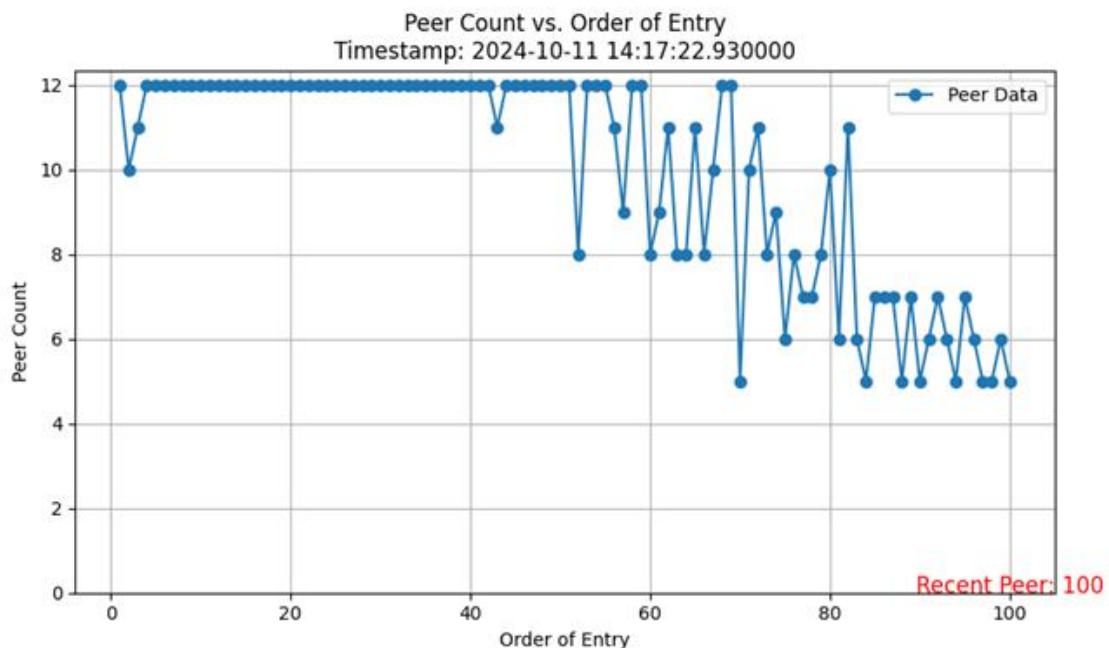


그림 3. 노드의 참여에 따른 각 노드의 차수 변화

Fig. 3. Changes in the degree of each node due to node participation

컨테이너로 가능한 위 PC에서는 REST API를 제거하자 최대 50개까지 피어를 별도의 컨테이너로 동작시킬 수 있었다. 이는 REST API를 사용하지 않는 구현이 소규모 실험자나 열악한 실험 환경에서 더 우세함을 나타내며, 이러한 환경에서의 실험에 도움이 됨을 알 수 있다. 더 많은 사람이 작은 컴퓨터로 실험이 가능해짐에 따라 더 많은 아이디어가 우리의 플랫폼에서 테스트 될 것이고, 이는 P2P 네트워크의 전파 성능을 개선하기 위한 연구를 가속하는데 기여한다.

REST API를 사용하지 않는 대신, 노드에게 전파 명령을 전달할 방법으로 libp2p 자체의 프로토콜을 사용하였다. Controller라는 libp2p 프로토콜 상에서 명령을 전달하는 별도의 피어를 두고, 해당 피어의 위치를 피어들이 사용하는 DHT[15] 네트워크와 완전히 분리시켜 각 피어가 실제 물리적 호스트에 가하는 하드웨어의 부담을 줄이면서도 네트워크에 관여하지 않는 구조를 완성하였다. 또한 각 피어가 로그를 작성함에 있어 네트워크상으로 로그를 전송하는 것이 아닌 로컬 파일로 로그를 작성하고, 이를

매니저 서버가 수집하는 방식을 채택함으로써 P2P 네트워크가 실험 외의 부가적인 환경에 의해 받는 영향을 최소화하였다.

IV. K-P2PLab 기술 사양

Go-libp2p로 개발된 각각의 실행중인 P2P 어플리케이션은 도커 컨테이너 하나당 어플리케이션 하나씩 배포되며, 각 도커 컨테이너는 도커 네트워크 위에서 고유한 IPv4 주소를 부여받고, 20000번 포트를 사용해 libp2p 통신을 진행한다. 이를 위해 Go-libp2p의 구현 기술을 적극 활용하였으며, 그 예시로 피어 검색 기술로 네트워크의 확장성이 높은 Kademlia[16]를 선택하였다. 메시지 라우팅 기술 역시 libp2p의 Gossipsub[17]이라 하는 Protocol Labs에서 개발한 기존의 Pubsub[18]에서 한층 더 발전된 기술을 적극 활용하였다. 각 피어들은 또한 30000번 포트로 자신의 Metrics 정보를 주기적으로 내보내어 이를 Prometheus와 Grafana가 수집하고 분석할 수 있다. 배포 가능한 노드의 최대 숫자는

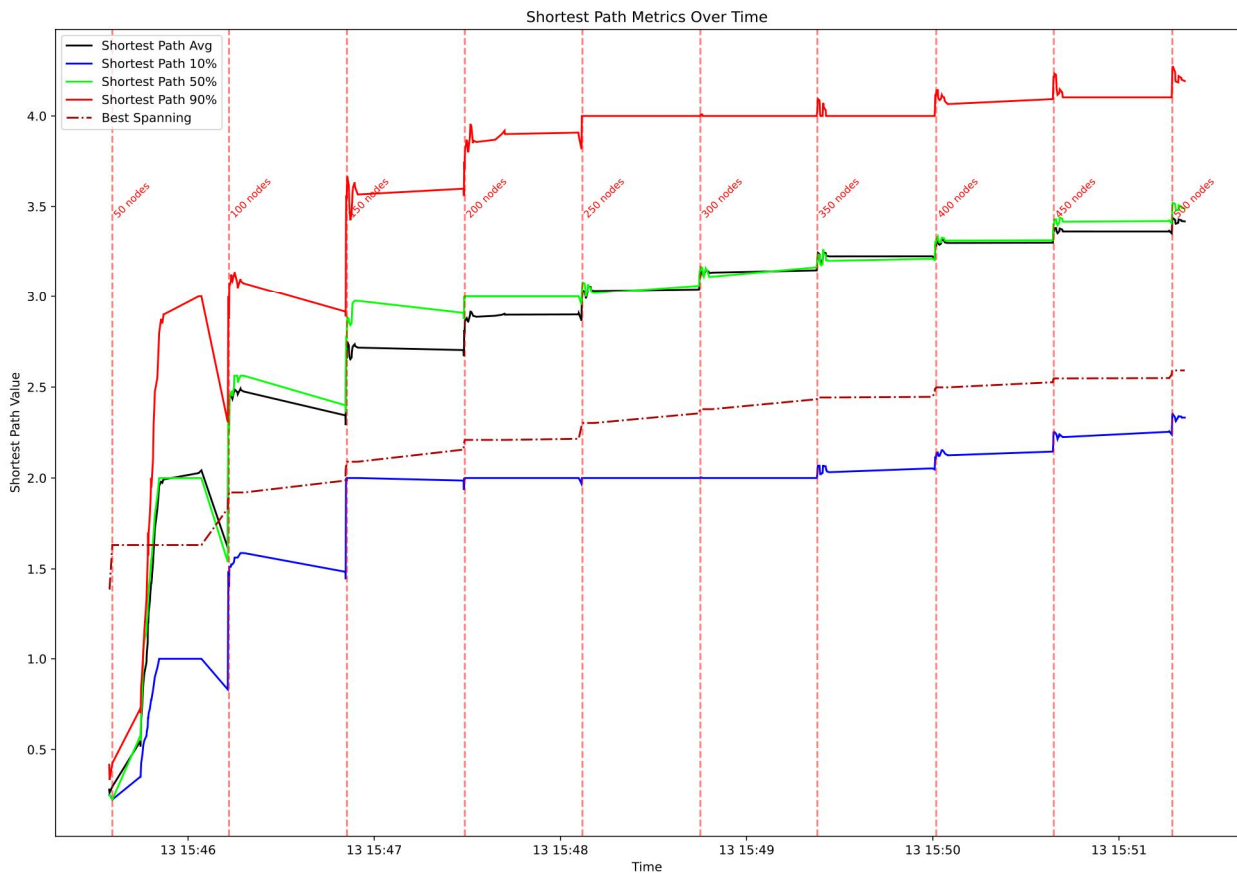


그림 4. 노드의 진입에 따른 각 노드가 갖는 Shortest Path 값 4가지의 Trimmed Average의 변화와 해당 시점에서 가장 이상적인 직경(Diameter)을 나타낸 그림

Fig. 4. Four Shortest Path value's Trimmed Average and the most ideal Diameter at that point due to participation of node

IPv4 전체로 이는 사실상 노드 배포의 한계가 존재하지 않으며, 러너 서버의 증축을 통해 간단히 실험 환경을 확장 가능하며, 서버 환경을 최대로 활용하여 실험을 진행할 수 있다.

피어는 노드가 실행될 때 역할을 부여받는데, 이는 총 3가지로 Boot, Worker, Controller가 있다. 부트 피어는 Kademia를 위한 부트스트랩 노드로, 다른 피어가 네트워크에 진입하기 위한 진입점의 역할을 한다. 부트 피어는 Gossipsub이 구현되어 있지 않으며, 따라서 메시지 전파에는 관여하지 않는다. 하지만 Metrics은 열려 있으므로 필요에 따라 정보를 수집할 수는 있다. 워커 피어는 Gossipsub이 구현된 노드로 메시지 전파에 기여하고 실제 네트워크를 구성하는 노드이다. Kademia를 사용하며 부트 피어를 통해 네트워크에 참여하고, Gossipsub을 통해 메시지를 전파한다. 컨트롤러 피어는 워커 피어 중 하나를 선택해 그 노드에게 퍼블리싱 명령을 전달할 수 있게 만들어진 피어이며, 명령을 전달 받은 워커 피어는 즉시 퍼블리싱을 진행한다. 현재는 임의의 1KiB짜리 패킷을 퍼블리싱 하도록 설계되어 있다. 또한 컨트롤러 피어는 Gossipsub은 물론 Kademia 네트워크에도 전혀 참여하지 않는다. 이는 실행 중인 컨트롤러 노드가 실험 중인 P2P 네트워크와 완전히 분리 되어있음을 의미하며, 오로지 기본적인 단일 다이얼을 통해 특정 워커 피어에게만 명령을 전달한다.

각 피어가 동작하며 발생한 동작은 로그에 기록되며, 로그는 노드 내부의 /libp2p 디렉토리에 작성된다. 이 디렉토리는 각 러너 서버마다 존재하는 /log라는 디렉토리와 마운트 되어있으며, 각 러너 서버의 /log라는 디렉토리는 다시 매니저 서버의 /log-nfs라는 디렉토리에 NFS로 공유 되어있다. 이를 통해 피어가 작성한 모든 로그 파일이 매니저 서버의 한 디렉토리에 모이는 것처럼 동작한다. 로그의 출력 수준 및 필터링은 미리 설정 파일을 통해 설정 가능하며, 그 외에도 다양한 파라미터를 미리 설정 가능하다.

또한 서버를 처음 구축하고 클러스터링 하는 방법과, 실험을 진행하는 방법이 별도의 마스터 프로그램으로 구현되어 있어 편리하면서도 세밀하게 조작이 가능하다. 앞서 말한 마스터 프로그램은 K-P2PLab을 효율적으로 사용하기 위한 CLI의 형태로 제공된다.

마지막으로 모든 피어는 네트워크상에 Metrics를 전파하도록 설계되었는데, 이를 통해 다른 외부 도구를 활용하여 네트워크를 모니터링 및 분석 가능하게 하였다.

V. K-P2PLab 검증 및 활용

우리는 앞서 설명한 우리가 개발한 도구들을 활용하여 직접 로그와 Metrics 데이터를 수집하였고, 이를 Go-libp2p의 구현 명세와 대조하여 정상적으로 작동하는 것을 확인하였다.

이 과정에서 NetworkX 기반의 분석 및 출력 도구로 우리가 앞으로 연구하고자 하는 목표를 조사해보았다. 그중 하나는 TCP 연결을 시각화한 P2P 네트워크 그래프를 출력하였을 때, 기본적인 libp2p 구현이 매우 조밀하고 불필요한 TCP 커넥션을 유지하는 것을 확인하였고, 이는 네트워크 성능에 부담을 줄 우려가 존재한다고 판단하였다. 또 다른 하나는 위 도구로 Gossipsub의 메시 네트워크를 출력함으로써 각 노드의 연결 차수가 최대 차수 D_{High} 에 근접하게 유지되는 것을 확인하였고, 피어가 최대 수치의 메시 연결을 가지면 이후의 Graft 요청을 즉시 Prune 메시지를 반환하여 거절하는 메커니즘이 포함되어 있음을 검증하였다.

그림 3은 100개까지 차례로 노드가 참여만 하고 이탈이 존재하지 않는 환경에서 노드의 진입 순서 (Order of Entry, 가로축)와 노드의 차수(Peer Count, 세로축)에 관한 그래프이다. 기존 노드들이 하나도 이탈하지 않는 극단적인 네트워크에서는 새로운 노드가 참여하여도 앞서 설명한 메커니즘으로 인해 새롭게 참여한 노드들이 기존 노드들에 비해 낮은 이상적인 차수 값 D 에 근접한 메시 연결을 가지는 것을 알아내었다. 현재 libp2p는 위 두 문제에 대해 크게 논의되지 않고 있으나, 이에 대한 자세한 연구 또한 필요할 것으로 생각된다. 100번째 피어가 들어온 시점에서 약 50번째로 들어온 피어를 기준으로 전과 후의 피어들의 전반적인 차수가 차이가 나는 것을 볼 수 있다.

그림 4는 네트워크에 노드가 참여함에 따른 P2P 네트워크의 Shortest Path 값 4가지(한 노드로부터 다른 모든 노드까지 가기 위해 거쳐야 하는 최단 노드의 수를 리스트로 만든 것을 L이라고 할 때, 정렬된 L의 각 10%, 50%, 90%째 값과 L 전체의 평균)를 노드 별로 구한 값들의 절단 평균(Trimmed

Average)을 나타낸 그래프이다. 이때 사용된 Gossipsub 파라미터 D_{Low} , D , D_{High} 는 각각 4, 6, 12인 Go-libp2p Gossipsub의 기본 설정 값이다. 또한 각 그래프 상의 점선은 네트워크의 직경이 최단 거리로 구성될 수 있는 스페닝트리 형태를 구성할 때의 직경으로, 해당 점선과 그래프 상의 다른 직선을 비교하여 현재 네트워크의 파라미터에 따른 효율을 유추할 수 있다. 절단 평균을 사용한 이유는 네트워크상에 노드가 급격히 늘어날 때 발생하는 노이즈와 같은 값을 제거하여 그래프가 시각적으로 난잡해지는 것을 막기 위함이다.

본 실험에서 절단 평균은 각 극단 값을 10%씩 제거하였고, 각 노드가 30초의 간격을 두고 50~500까지 50개씩 참여하였다. 이때 순간적으로 노드가 증가함에 따라 직경값에 심한 노이즈를 보이는 각 네트워크 확장 시간(그래프 상에서 붉은 세로 점선)을 제외하면 전체적으로 로그 스케일에 가까운 증가 추세를 보임을 알 수 있다. 또한 각 10% 값의 절단 평균을 나타내는 파란 선과 50% 값의 절단 평균을 나타내는 초록 선, 90% 값의 절단 평균을 나타내는 빨간 선이 올바르게 배치되어 있다는 사실을 통해 해당 그래프가 네트워크를 올바르게 해석하고 있음을 보이고 있다.

VI. 결론 및 향후 연구

우리는 K-P2PLab이라는 P2P 가상 실험 환경 구축 및 분석을 위한 도구를 개발하였다. 이 도구로 토폴로지 측정 및 전파 성능을 측정하여 P2P 네트워크의 성능 데이터를 충분히 수집하고 분석할 수 있음을 확인하였다. 우리는 도커 기술과 데이터 모니터링 및 네트워크 분석 기술을 활용하였다.

현재 K-P2PLab은 노드 간 연결에 대한 통신 성능을 조절하는 기능이 구현되어 있지 않다. 실제 P2P 네트워크와 동일한 환경에서 실험을 진행하기 위해서는 통신성능을 조절하는 기능이 필요하며 Linux의 TC(Traffic Control) 기능을 활용하여 노드 간 통신성능(전파 시간, 패킷 손실 등)을 조절할 수 있게 발전시킬 계획이다. 이후에는 K-P2PLab을 활용하여 여러 브로드캐스팅에 관련된 성능 실험하고 성능을 높일 수 있는 다양한 방법을 고안할 것이며, 다른 연구자들이 브로드캐스팅 성능 개선을 위한 아이디어를 실험하고 분석하기 위한 도구로 우리의 플랫폼이 사용될 수 있을 것이다. 또한 위에서 언급한 TCP 커넥션 및 Gossipsub 메시 연결 문제에 대

해, TCP 커넥션을 평가하고 조절하여 최대 개수를 제한하는 알고리즘 혹은 프로토콜의 개발에 관한 연구와 주기적 메시 연결 정리를 통해 정적인 P2P 네트워크상의 균등한 차수를 갖게 하는 프로토콜 도입 등의 방법을 통해 해결하는 것도 주요한 향후 연구 주제로 다루어질 수 있다.

References

- [1] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," 2008.
- [2] Buterin, V., "Ethereum white paper," GitHub repository, vol. 1, pp. 22-23, 2013.
- [3] Werth, J., Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N., and Pahl, C., "A review of blockchain platforms based on the scalability, security and decentralization trilemma," in Proc. ICEIS (1), 2023, pp. 146-155.
- [4] Saad, M., Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D., and Mohaisen, D., "Exploring the attack surface of blockchain: A comprehensive survey," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1977-2008, 2020.
- [5] 김정연, 주홍택, "코스모스에서 블록체인 간 메시지 전달 시간 측정 및 분석," 한국통신학회논문지, vol. 47, no. 10, pp. 1556-1564, 2022, doi: 10.7840/kics.2022.47.10.1556.
- [6] 김애리, 주홍택, "비트코인 네트워크에서 압축 블록 전달 방식의 전송 지연 분석," 한국통신학회논문지, vol. 47, no. 6, pp. 826-835, 2022, doi: 10.7840/kics.2022.47.6.826.
- [7] 맹수훈, 주홍택, "이더리움 네트워크에서 액티브 프루빙을 이용한 정확도 높은 토폴로지 탐색 방법," 한국통신학회논문지, vol. 47, no. 3, pp. 453-460, 2022, doi: 10.7840/kics.2022.47.3.453.
- [8] libp2p, "libp2p/go-libp2p," GitHub. Available: <https://github.com/libp2p/go-libp2p>. [Accessed: Nov. 20, 2024].
- [9] Docker, "Docker: Accelerate how you build, share, and run modern applications," Available: <https://www.docker.com/>. [Accessed: Nov. 20, 2024].

- [10] Prometheus, "Prometheus: Monitoring system & time series database," Available: <https://prometheus.io/>. [Accessed: Nov. 20, 2024].
- [11] Grafana Labs, "Grafana: The open observability platform," Available: <https://grafana.com/>. [Accessed: Nov. 20, 2024].
- [12] Whiteblock, "Whiteblock GitHub repositories," GitHub. Available: <https://github.com/whiteblock>. [Accessed: Nov. 20, 2024].
- [13] Testground, "Testground GitHub repositories," GitHub. Available: <https://github.com/testground>. [Accessed: Nov. 20, 2024].
- [14] libp2p, "libp2p and Ethereum," Available: <https://blog.libp2p.io/libp2p-and-ethereum/>. [Accessed: Nov. 20, 2024].
- [15] Dabek, F. F. E., "A distributed hash table," Doctoral dissertation, Massachusetts Institute of Technology, 2005.
- [16] Maymounkov, P., and Mazieres, D., "Kademlia: A peer-to-peer information system based on the xor metric," in Proc. Int. Workshop Peer-to-Peer Syst., Springer Berlin Heidelberg, 2002, pp. 53-65.
- [17] Vyzovitis, D., and Psaras, Y., "Gossipsub: A secure pubsub protocol for unstructured, decentralised p2p overlays," Proc. Protocol Labs TechRep, PL-TechRep-gossipsub-v0., 2019.
- [18] Banerjee, T., and Sahni, S., "Pubsub: An efficient publish/subscribe system," IEEE Trans. Comput., vol. 64, no. 4, pp. 1119-1132, 2014.

이 성 욱 (Sung-wook Lee)



2021년 3월~현재: 계명대학교
컴퓨터공학과 학사 과정

<관심분야> 컴퓨터 공학, 네트워크, 블록체인
[ORCID: 0009-0002-5796-6474]

주 흥 택 (Hong-taek Ju)



1989년 2월: KAIST 컴퓨터공
학과 졸업
1991년 2월: POSTECH 컴퓨
터공학과 석사
2002년 2월: POSTECH 컴퓨
터공학과 박사
2002년 9월~현재: 계명대학교

컴퓨터공학과 교수
<관심분야> 컴퓨터 공학, 네트워크, 블록체인
[ORCID: 0000-0002-8434-485X]

네트워크 관점에서의 3D 콘텐츠 표현 방식 분석

유 상 우*, 홍 원 기^o

Analysis on 3D Content Representations in Network Perspective

Sangwoo Ryu*, James Won-Ki Hong^o

요 약

확장현실(XR) 기술의 발전으로 3D 콘텐츠의 중요성이 급격히 증가하고 있다. 이에 따라 3D 콘텐츠를 효율적으로 표현하고 전송하는 기술의 필요성이 대두되고 있다. 3D 콘텐츠는 다양한 방식으로 표현이 가능하며, 각 표현 방식마다 용량, 품질 등 다양한 부분에서 차이가 존재한다. 본 연구에서는 포인트 클라우드(Point Cloud), 메쉬(Mesh), NeRF(Neural Radiance Field), 3DGS(3D Gaussian Splatting), 총 네 가지의 주요 3D 콘텐츠 표현 방식을 대상으로, 동일한 콘텐츠를 각 표현 방식으로 제작하여 데이터 크기, 품질, 데이터 손실의 영향 등 네트워크 성능 특성을 분석하고, 서비스 및 네트워크 품질에 영향을 주는 추가 고려 요소들과 관련 연구 및 기술들을 제시한다. 이를 통해 다양한 형태의 3D 콘텐츠에 대한 이해를 제공하고, 더 나은 서비스 및 네트워크 품질 관리를 돕는 것을 목표로 한다.

Key Words : 3D Content Representations, Volumetric Video Streaming, Quality of Service

ABSTRACT

The rapid advancement of Extended Reality (XR) technology has significantly increased the importance of 3D content. Accordingly, there is a growing need for technologies that can efficiently represent and transmit 3D content. 3D content can be represented in various ways, with each representation method differing in aspects such as data size and quality. This study focuses on four major 3D content representation methods: Point Cloud, Mesh, Neural Radiance Field (NeRF), and 3D Gaussian Splatting (3DGS). In this research, we analyze the network performance characteristics of these four representation methods by creating the same content using each method and evaluating factors such as data size, quality, and the impact of data loss. Additionally, we present additional considerations that affect service and network quality, along with related research and technologies. The aim of this study is to provide a comprehensive understanding of various 3D content representation and to assist in improving service and network quality management.

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음 (IITP-2024-RS-2024-00437866)

◆ First Author : Pohang University of Science and Technology (POSTECH), Graduate School of Artificial Intelligence, rswoo@postech.ac.kr

^o Corresponding Author : POSTECH, Department of Computer Science and Engineering, jwkhong@postech.ac.kr

논문번호 : KNOM2024-02-05, Received November 18, 2024; Revised December 13, 2024; Accepted December 22, 2024

I. 서 론

최근 가상 현실(Virtual Reality, VR), 증강 현실(Augmented Reality, AR), 혼합 현실(Mixed Reality, MR)과 함께, 이를 모두 포함하는 확장현실(Extended Reality) 기술의 발전으로, 3D 콘텐츠의 중요성이 급격히 증가하고 있다. P&S 인텔리전스는 XR 시장 규모를 2022년 383억 달러에서 2030년 3,948 달러로 38.5% 성장을 전망했으며 [1], IDC는 AR/VR 기기 시장이 2023년부터 2027년까지 연평균 32.6% 성장할 것으로 전망했다 [2]. 이러한 확장 현실 공간에서는 사용자에게 다양한 형태의 3D 콘텐츠를 제공하며, 엔터테인먼트, 교육, 의료, 미디어 등 다양한 산업 분야에서의 활용이 확대되면서, 이를 효율적으로 표현하고 전송하는 기술의 필요성이 중요해졌다.

이러한 맥락에서 주목받고 있는 기술 중 하나가 볼류메트릭 영상(Volumetric Video)이다. 볼류메트릭 영상은 3차원 데이터로 구성된 영상으로, 확장현실 환경에서 3D 콘텐츠를 보여주기 위해 사용되고, 실제 세상의 물체, 사람, 움직임 등을 가상 세계로 가져와 사실적으로 보여준다. 하지만 볼류메트릭 영상과 같은 3D 콘텐츠는 일반적으로 3차원 공간에서의 위치, 색 등 대용량의 데이터를 가지고 있어, 사용자 및 서비스 제공자 모두에게 전송과 처리에 부담을 준다. 특히 실시간 스트리밍 및 대규모 사용자 환경에서의 서비스를 위해서는 이러한 전송 및 처리 과정을 효율적으로 수행할 수 있도록 하는 연구가 필수적이다.

기존에 3D 콘텐츠를 포인트 클라우드(Point Cloud) 혹은 메쉬(Mesh) 형태로 표현한 것에 더해, 최근에는 NeRF(Neural Radiance Field), 3DGS(3D Gaussian Splatting) 등 새로운 고품질 3D 데이터 표현 방식이 제안되고 있다. 이러한 콘텐츠 표현 방식의 다양화는 표현 방식 간 서로 다른 특성으로 인해 서비스 제공자가 고려해야 하는 요소를 더욱 복잡하게 만들고 있다. 대표적으로 각 표현 방식 별로 제작 과정, 품질, 렌더링 방식 및 속도, 데이터 크기 등이 다르며, 이에 따라 사용 시 나리오, 목표 플랫폼, 주로 사용될 네트워크 환경 등을 함께 고려해야 한다. 특히 네트워크 전송 측면에서의 고려사항들은 원활한 전송을 통한 사용자 경험 최적화, 컴퓨팅 자원 최적화를 통한 운영 비용 절감, 확장성 확보 등 전체적인 서비스 품질 향상과

직접적인 관련이 있기 때문에 서비스 제공자가 중요하게 고려해야 할 요소가 된다.

따라서 서비스 제공자가 더 나은 서비스를 제공하기 위해서는 네트워크의 관점에서 3D 콘텐츠 표현 방식들을 분석 및 이해하고, 적절한 표현 방식을 선택하는 것이 중요하다. 본 연구는 네트워크 관점에서 다양한 3D 콘텐츠 표현 방식을 분석하고, 각 방식의 장단점을 비교하는 것을 목적으로 한다. 특히 같은 콘텐츠를 포인트 클라우드, 메쉬, NeRF, 3DGS, 총 네 가지 표현 방식으로 제작하여 데이터 크기, 품질, 데이터 손실의 영향 등 고려 요소들을 분석하고 추가 고려 요소들 및 관련 연구, 기술들을 제시하여 3D 콘텐츠 표현 방식에 대한 이해를 돕고자 한다. 본 연구의 기여는 다음과 같다.

- 1) 같은 콘텐츠에 대한 서로 다른 3D 표현 방식의 네트워크 성능 특성에 대한 분석 결과를 제시
- 2) 각 3D 표현 방식의 고려 요소들에 대한 관련 연구 및 기술들을 제시

II. 배경 및 관련 연구

본 장에서는 본 논문에서 사용할 3D 데이터 표현 방식 네 가지(포인트 클라우드, 메쉬, NeRF, 3DGS)를 설명하고, 해당 표현 방식들의 대략적인 비교를 담고 있는 연구들을 소개한다.

1) 포인트 클라우드(Point Cloud)

포인트 클라우드는 3D 물체를 점들의 집합으로 구성한 데이터 구조이다. 각 점은 X, Y, Z 좌표와 추가적으로 색상 정보(R, G, B), 법선(Normal) 벡터 정보를 가질 수 있으며, 전체 데이터의 크기는 각 점을 이루는 정보들의 데이터 유형(실수, 8비트 정수 등)과 점의 개수로 결정된다.

2) 메쉬(Mesh)

메쉬는 3D 물체를 점들과 점들을 이어 모서리, 다각형 면을 구성한 데이터 구조이다. 메쉬는 각 점의 X, Y, Z 좌표와 선 혹은 면을 이루기 위한 점들의 연결 정보, 추가적으로 점 혹은 면의 색상, 법선 벡터 정보를 담을 수 있으며, 전체 데이터의 크기는 각 점, 선 면을 이루는 정보들의 데이터 유형과 점, 선 면의 개수로 결정된다.

3) NeRF(Neural Radiance Field)

NeRF는 2020년 3D 복원을 위해 제안된 데이터

표현 방식으로 [3], MLP(Multi Layer Perceptron, 다중 퍼셉트론)와 같은 신경망을 이용해 시선의 위치 및 방향에 따른 색과 밀도를 표현하는 구조이다. 기존 포인트 클라우드, 메쉬 등의 표현 방식들과 다르게 3D 공간 상에 데이터가 명시적(Explicit)으로 표현되지 않고, 암시적(Implicit)으로 표현되며, 데이터 크기는 구현 방식에 따라 다양한 요소(신경망 크기, 해시맵 크기 등)에 의해 결정된다.

4) 3DGS (3D Gaussian Splatting)

3DGS는 2023년 제안된 표현 방식으로 [4], 3D 물체를 3D 가우시안(Gaussian) 함수의 집합으로 구성한 데이터 구조이다. 각 가우시안은 X, Y, Z 좌표와 크기, 회전, 투명도 정보, 구면 조화 함수(Spherical Harmonics) 형태의 색 정보 등과 같은 속성을 가진다. 전체 데이터의 크기는 가우시안을 구성하는 속성들의 개수 및 데이터 유형, 그리고 가우시안의 총 개수로 결정된다.

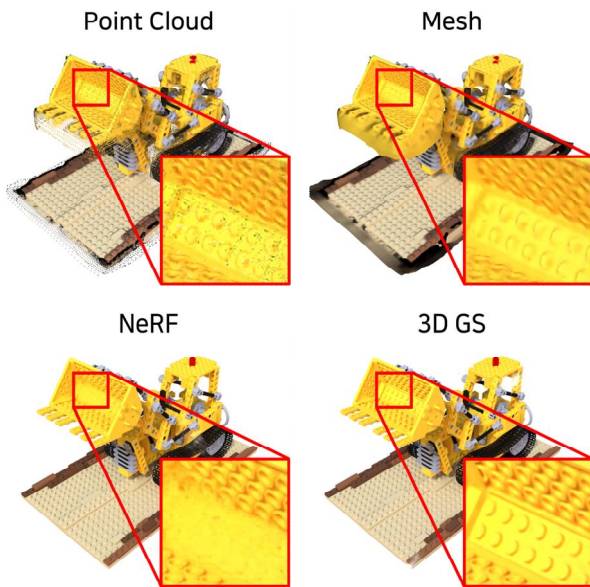


그림 1. 각 표현에 따른 렌더링 예시
Fig. 1. Rendering example of each 3D representation

그림 1은 앞서 설명한 네 가지 표현 방식에 대해 실제로 렌더링을 한 예시를 보여주고, 각 표현 방식의 시각적 특징을 확인할 수 있다. 포인트 클라우드의 경우 일부 빈 공간에 대한 노이즈를 확인할 수 있으며, 메쉬의 경우 적은 수의 면으로 인해 디테일이 뭉개진 모습을 볼 수 있다. NeRF와 3DGS는 학습 기반 방식으로, 전체적으로 높은 품질을 보여주지만, 학습이 완전히 되지 못한 경우에는 일부분이

제대로 표현되지 못한 모습도 확인할 수 있다.

이러한 다양한 3D 데이터 표현 방식에 대해 대략적인 비교를 한 사례가 일부 존재한다. 먼저 논문 [5]에서는 볼류메트릭 영상의 캡처부터 사용자에게 보여주는 부분까지 볼류메트릭 영상 서비스를 구성하는 모든 부분에 대해 관련 연구를 소개한다. 해당 논문에서는 데이터 표현 방식과 관련해서 포인트 클라우드, 메쉬, 복셀 (Voxel) 등 6종류의 표현 방식에 대해 크기, 품질, 자원 사용량 등을 높음, 중간, 낮음 등으로 제시하였다. 논문 [6]은 네트워크 관점에서 볼류메트릭 영상 서비스에 대한 개요, 구조, 예시 설계 등을 제시하였다. 데이터 표현 방식과 관련해서는 포인트 클라우드, 메쉬, V-PCC [31] 등 5종류의 표현 방식에 대해 크기를 높음, 중간, 낮음 등으로 제시하고 장단점을 간단하게 서술하였다. 논문 [7]은 볼류메트릭 영상 스트리밍에서 3DGS 사용을 위한 방법에 대한 연구이지만, 포인트 클라우드, 메쉬, NeRF, 3DGS, 총 4종류의 표현 방식에 대해 렌더링 품질, 속도, 접근성, 크기를 높음, 중간, 낮음 등으로 간단히 비교하는 부분을 포함하였다. 논문 [8]은 3D 복원을 위한 방식 3가지를 비교하는 연구로, 기존에 포인트 클라우드 혹은 메쉬를 만들기 위하여 사용하는 COLMAP[10, 11, 12] 방식과 NeRF, 3DGS 두 가지 새로운 방식을 비교해 유용성을 평가하였다. 논문 [9]는 볼류메트릭 영상 스트리밍에서 NeRF와 같은 신경망 기반 표현 방식들을 사용하기 위해 고려해야 하는 사항들과 이를 위한 사전 실험 결과들을 제시하였다. 논문 [10]은 빠르고 다양하게 제안되는 NeRF 및 3DGS 기반 3D 복원 방식 기술들을 테스트하기 위한 통합 프레임워크를 제안하며, 여러 데이터셋에 대해 시각 품질, 훈련 시간, 메모리 사용량 등에 대한 정량적인 수치를 제공한다.

이와 같이 기존 연구들이 최종 렌더링 품질을 주로 평가하였다면, 본 연구에서는 네트워크 전송 측면을 함께 고려하여 각 표현 방식에 대한 분석 결과와 관련 연구, 기술을 제시하는 것을 목표로 한다.

III. 방법

이 장에서는 한 콘텐츠에 대해 여러 표현 방식의 모델을 여러 크기/품질로 제작하기 위한 방법과, 이를 분석하기 위한 방법을 서술한다.

1. 데이터 준비

한 콘텐츠에 대해 여러 표현 방식으로 모델을 제작하기 위하여, 같은 멀티 뷰 데이터로부터 3D 복원을 수행하는 방법들을 사용하였다. 이를 위해 NeRF [3] 논문에서 함께 제안한 NeRF Synthetic 혹은 Blender 데이터셋의 Lego 씬의 훈련 데이터셋을 이용하였다. Lego 씬은 100장의 훈련 이미지와 200장의 테스트 이미지를 포함하고 있어, 3D 모델 복원을 위해 훈련 이미지만 사용하였다.

포인트 클라우드와 메쉬를 제작하기 위해서는 COLMAP [10, 11, 12]을 사용하였다. COLMAP은 멀티 뷰 이미지로부터 포인트 클라우드 혹은 메쉬를 복원하는 대표적인 방법 중 하나이다. 100장의 훈련 이미지와 COLMAP을 이용해 고품질의 메쉬를 복원하였으며, 해당 메쉬는 Open3D [14] 라이브러리를 이용해 다운샘플링을 수행하여 저화질의 메쉬를 제작하였다.

포인트 클라우드는 고품질 메쉬에서 꼭짓점(Vertex)만 추출하여 복원했으며, Open3D 라이브러리를 이용해 점의 개수를 줄이는 다운샘플링(Downsampling)을 수행하여 저화질의 포인트 클라우드를 제작하였다. 각 품질에 따른 포인트 클라우드와 메쉬의 점, 면 개수는 유사한 데이터 크기 범위를 위해 실험적으로 조정하였다.

표 1, 표 2는 각각 제작된 포인트 클라우드와 메쉬의 점, 면 개수와 데이터 크기를 보여준다. 해당 표의 데이터 크기는 압축이 적용되지 않은 미가공(Raw) 상태의 PLY 파일 데이터 크기이다. Gbps@30Hz는 영상 스트리밍에 사용하는 것을 가정하여, 초당 30프레임의 속도로 해당 파일을 전송하는 경우 사용되는 대역폭이다.

표 1. 여러 품질로 제작된 포인트 클라우드 정보
Table 1. Point cloud information with different quality

Level	# of Points	Size (MB)	Gbps @30Hz
0	102,224	1.462	0.343
1	302,003	4.320	1.013
2	599,129	8.571	2.009
3	885,307	12.664	2.968

표 2. 여러 품질로 제작된 메쉬 정보
Table 2. Mesh information with different quality

Level	# of Points	# of Faces	Size (MB)	Gbps @30Hz
0	37,395	75,720	1.474	0.345
1	110,534	221,918	4.332	1.015
2	222,568	445,842	8.711	2.042
3	322,231	644,697	12.602	2.954

NeRF와 3DGS 모델을 제작하기 위해서는 Nerfstudio [15]를 사용하였다. Nerfstudio는 NeRF 및 3DGS 기반 방법들을 수행하고 테스트할 수 있는 API를 제공하는 프레임워크이다. NeRF 모델을 만들기 위해서는 Nerfstudio에서 제공하는 nerfacto를 사용하였다. nerfacto는 좌표를 해시맵을 이용해 효율적으로 표현하는 해시 인코딩 등 Instant-NGP [16]와 같은 이전 연구들에서 효율적인 학습을 위해 제안되었던 기술들을 모아서 제작한 방식이다. nerfacto는 충분히 작은 신경망(MLP)을 사용하기 때문에 신경망의 크기는 용량에 큰 영향을 주지 않았고, 학습된 모델의 전체 용량은 해시 맵 크기에 영향을 받는 것을 실험적으로 확인하였다. 따라서 NeRF 모델의 크기를 조절하기 위해 해시맵 크기를 결정하는 파라미터 (log2-hashmap-size)를 조정하였다. 해당 파라미터는 기본 모델에서 19로 설정되어 있었고, 일정 값 이하에서는 용량의 변화가 크게 없어 9부터 19까지 단계마다 2 만큼 늘려가며 총 6개의 값을 사용하였다.

3DGS 모델을 만들기 위해서는 Nerfstudio에서 제공하는 splatfacto를 사용하였다. splatfacto는 nerfacto와 비슷하게 여러 연구들에서 제시된 기술들을 모아서 제작한 방식이다. splatfacto 역시 용량에 영향을 주는 파라미터를 관련 문서와 함께 실험적으로 결정했으며, 3DGS에서 가우시안의 색을 표현하기 위한 구면 조화 함수의 차수를 결정하는 파라미터 (sh-degree)와 훈련 과정에서 가우시안의 투명도 임계값 파라미터 (cull-alpha-thresh)를 조정하였다. 이 중 sh-degree 파라미터는 용량에 가장 큰 영향을 주었고, cull-alpha-thresh 파라미터는 일부 영향을 주었다. 두 파라미터를 조정하여 총 6개의 모델을 제작하였다.

표 3, 표 4는 각각 제작된 NeRF와 3DGS 모델의 파라미터 정보와 데이터 크기를 보여준다. NeRF 모델은 신경망 기반 모델로 표준으로 정해진 파일 형식이 없기 때문에, 데이터 크기로 학습된 모델 파

일의 크기를 사용하였다. 3DGS는 NeRF와 다르게 명시적인 표현 방식이기 때문에 포인트 클라우드, 메쉬와 비슷한 형태의 PLY 파일로 저장할 수 있어 해당 파일의 크기를 사용하였다. 해당 표의 데이터 크기는 압축이 적용되지 않은 미가공(Raw) 상태의 모델 혹은 PLY 파일 데이터 크기이다.

표 3. 여러 품질로 제작된 NeRF 모델 정보
Table 3. NeRF information with different quality

Level	log2-hash map-size	Size (MB)	Gbps @30Hz
0	9	28.383	6.652
1	11	28.945	6.784
2	13	31.101	7.289
3	15	38.999	9.140
4	17	67.069	15.719
5	19	167.770	39.321

표 4. 여러 품질로 제작된 3DGS 정보
Table 4. 3DGS information with different quality

Level	sh- degree	cull-alpha -thresh	Size (MB)	Gbps @30Hz
0	1	0.4	4.441	1.041
1	1	0.1	5.265	1.234
2	2	0.4	7.106	1.665
3	2	0.1	8.071	1.892
4	3	0.4	10.683	2.504
5	3	0.1	12.333	2.891

2. 시각 품질 분석

제작된 모델의 시각 품질을 평가하기 위해 PSNR, SSIM [17], LPIPS [18]의 총 세 가지 측정 지표를 계산하였다. 세 지표 모두 이미지의 유사도를 평가하기 위한 지표들로, PSNR, SSIM은 화질의 손실량을 평가하기 위해 주로 사용하며, LPIPS는 앞의 두 지표보다 인간 지각적인 면을 더욱 고려한 지표이다. PSNR, SSIM은 높을수록 좋은 수치이며, LPIPS는 낮을수록 좋은 수치이다. 포인트 클라우드와 메쉬는 테스트 이미지와 같은 시점에서 렌더링을 하기 위하여 데이터셋에서 제공하는 정보 및 변환 행렬을 이용해 카메라의 내부 변수(Intrinsic)와 위치를 계산하였으며, Open3D를 이용해 3D 모델을 불러오고 렌더링하여 이미지로 저장하였다. 저장된 이미지들은 테스트 이미지들과의 정해진 품질 지표를 계산하였고, 각 표현 방식 및 품질마다 평균 및 표준 편차를 계산하였다. NeRF와

3DGS 모델은 Nerfstudio에서 제공하는 평가 툴을 활용하였다. 네 가지 표현 방식 품질 지표 계산을 위해 같은 라이브러리 [19] 및 함수가 사용되었다.

3. 패킷 손실 분석

제작된 모델이 패킷 손실이 발생하는 경우 품질에 주는 영향을 평가하기 위해 두 서버로 이루어진 간단한 콘텐츠 전송 시스템을 구현하였다. 전송 서버와 수신 서버 모두 Python 소켓을 이용해 구현하였으며, 패킷 손실의 영향을 그대로 두기 위해 재전송을 보장하지 않는 UDP 프로토콜을 이용하였다. 3D 모델 파일은 정해진 패킷 크기 (2000 byte)로 분리하여 전송하였다. 패킷 손실은 전송 서버에서 리눅스 tc 명령어를 이용하여 0.1%, 1%, 10%의 환경을 만들고, 수신한 모델은 III-2장 시각 품질 분석에서와 같은 방식으로 시각 품질을 평가하였다.

IV. 분석 결과 및 고려 요소

이 장에서는 본 논문에서 제안하는 분석의 결과와 3D 표현 방식 사용을 위해 고려해야 할 요소들을 서술한다.

1. 데이터 크기(대역폭)-품질 분석

먼저 포인트 클라우드는, 렌더링 환경에서 포인트의 크기 설정에 따라 품질이 다르게 나타났다. 그림 2는 포인트 크기에 따른 용량-품질 (PSNR) 그래프이다. 포인트 클라우드의 용량은 일반적으로 총 포인트의 수 혹은 포인트의 밀도와 비례하는데, 포인트의 크기가 작을 때는 포인트의 밀도가 낮기 때문에 빈 공간이 많이 생겨 품질이 낮게 측정되었다.

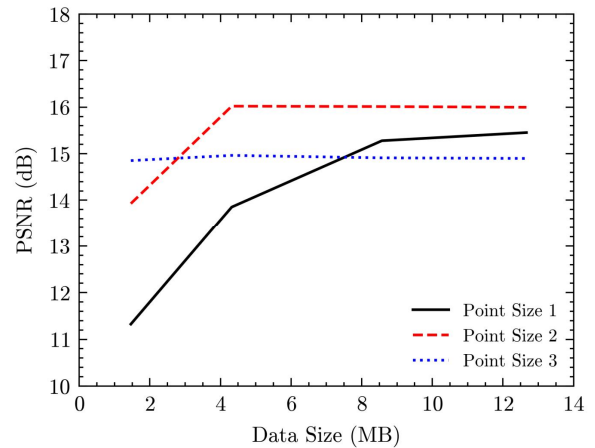


그림 2. 포인트 클라우드에서 포인트 크기에 따른 PSNR
Fig. 2. PSNR according to point size in point cloud

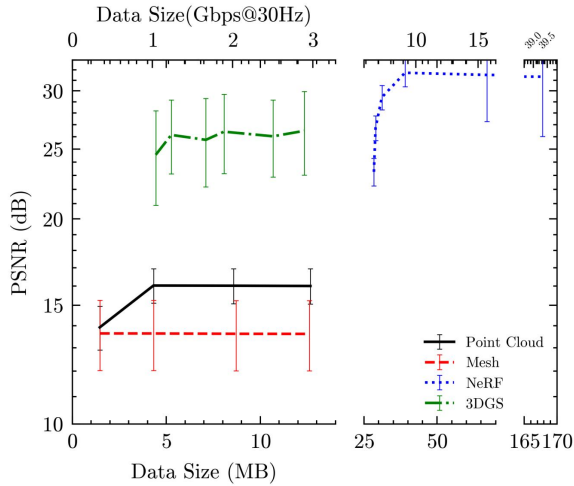


그림 3. 데이터 크기에 따른 시각 품질 (PSNR)
Fig. 3. PSNR according to the data size

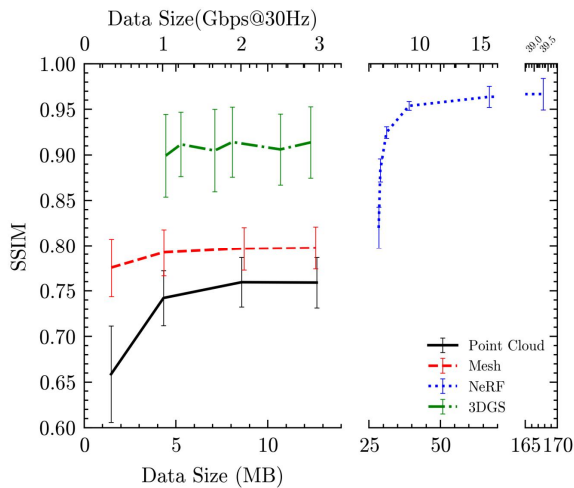


그림 4. 데이터 크기에 따른 시각 품질 (SSIM)
Fig. 4. SSIM according to the data size

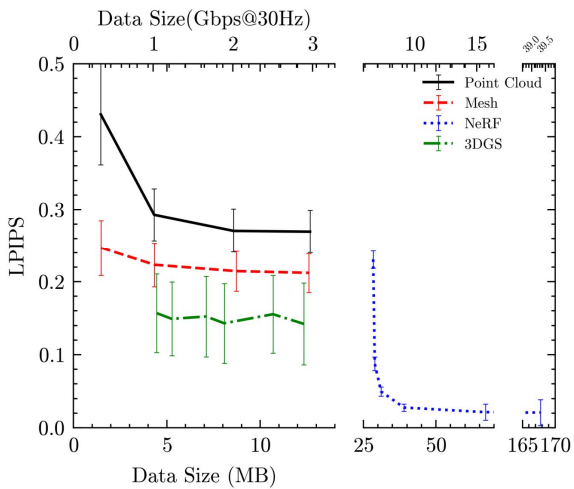


그림 5. 데이터 크기에 따른 시각 품질 (LPIPS)
Fig. 5. LPIPS according to the data size

반면에 포인트의 크기가 크면 이러한 빈 공간을 채워줄 수 있기 때문에 용량(포인트 수/밀도)이 늘어나더라도 품질 차이가 적었다. 이는 포인트 클라우드의 렌더링 방법 또한 품질에 영향을 미칠 수 있다는 것을 보여준다. 다른 3D 표현 방식과의 비교를 위해서는 평가한 세 개의 포인트 크기 중 가장 높은 PSNR을 보인 포인트 크기 2 설정을 사용하였다.

그림 3은 포인트 클라우드, 메쉬, NeRF, 3DGS, 총 네 가지 표현 방식에 대한 데이터 크기-품질 (PSNR) 그래프이다. 그림 4, 그림 5는 각각 데이터 크기-품질(SSIM), 데이터 크기-품질(LPIPS) 그래프이다. 포인트 클라우드의 경우 데이터 크기가 늘어남에 따라 품질 또한 상승했지만, 일정 수준 이상에서는 거의 변화가 없는 모습을 보였다. 이는 콘텐츠에 따라, 혹은 사용자 위치와 콘텐츠 사이의 거리에 따라 적절한 포인트 밀도나 개수를 설정하면 유사한 시각적 품질을 얻을 수 있다 것을 의미한다. 메쉬의 경우 포인트 및 면의 수가 늘어날 때 SSIM과 LPIPS가 소폭 개선되었지만, PSNR는 거의 변화를 보이지 않았다. 이는 메쉬를 이용하는 경우 낮은 용량의 모델을 이용하더라도 품질에서 큰 손해를 보지 않을 수 있다는 것을 의미한다. NeRF는 가장 높은 품질을 보여주었지만, 전체적으로 데이터 크기가 매우 크게 나타났다. 또한 일정 용량 이상에서는 PSNR은 거의 변화가 없었고, SSIM과 LPIPS만 소폭 개선되었다. 3DGS는 모든 크기에 대해 거의 비슷한 품질 수치를 보여주었다. 포인트 클라우드와 메쉬보다는 많이 증가하는 수치를 보여주었고, NeRF보다는 낮은 수치들을 보여주었다. 용량은 NeRF보다는 훨씬 적은 크기로, 용량 대비 품질에서는 가장 좋은 모습을 보여주었다.

2. 패킷 손실의 영향

표 5는 각 표현 방식 별로 0.1%, 1%, 10% 패킷 손실이 일어나는 환경에서 시각 품질 (PSNR) 변화를 보여준다. 포인트 클라우드의 경우에는 포인트 간 연결이 없고, 각 포인트의 위치 및 색 정보만 불러올 수 있다면 렌더링을 할 수 있기 때문에, 손실률이 증가함에 따라 품질이 감소하지만, 모델을 로드할 수 있었다. 손실률에 따라서 일부 포인트들이 잘못된 위치에 렌더링이 되거나 잘못된 색을 가지는 것을 확인할 수 있다. 다만 저장 형식에 대한 정보를 담고 있는 헤더가 손실될 경우에는 로드가 불가능할 것으로 보인다. 메쉬의 경우에는 포인트

정보와 함께 면을 이루기 위한 포인트 간 연결 정보를 함께 담고 있어 해당 부분이 손상되는 경우 모델을 로드할 수 없었다. 포인트 클라우드와 마찬가지로 헤더가 손상될 경우에는 로드가 불가능할 것으로 보인다. NeRF 모델은 로드가 가능한 경우와 불가능한 경우가 모두 있었는데, 역시 헤더 등 체크포인트 내 중요 정보가 손상되지 않는다면 대부분 신경망의 가중치 혹은 해시맵과 관련된 자료일 것으로 예상되어 로드가 가능할 것으로 보인다. 본 실험에서는 손실률 0.1%에서 NeRF 모델 로드에 성공했지만, 항상 로드에 성공할 것이라는 보장을 할 수 없다는 것을 의미한다. 3DGS의 경우에는 포인트 클라우드와 유사하게 각 가우시안에 대한 정보를 담고 있고, 메쉬처럼 면을 구성하기 위한 연결 정보를 담고 있지는 않기 때문에 헤더를 제외한 부분이 손상되더라도 렌더링이 가능했다. 하지만 일부 가우시안이 잘못된 위치, 크기, 색으로 렌더링이 되어 전체 콘텐츠를 방해하며 크게 시각 품질을 떨어뜨리는 모습을 확인할 수 있다. SSIM 및 LPIPS 지표에서도 손실이 늘어남에 따라 품질이 떨어지는 유사한 양상을 보였다. 표 6은 손실률에 따른 SSIM의 변화를 보여주며, PSNR과 마찬가지로 손실이 발생할 경우 SSIM 값이 감소하여 시각 품질이 떨어졌음을 확인할 수 있다. 표 7은 손실률에 따른 LPIPS의 변화를 보여주며, 손실이 발생할 경우 값이 증가해 시각 품질이 떨어졌음을 확인할 수 있다. 손실이 일어날 수 있는 환경에서 안정적으로 3D 콘텐츠를 로드하기 위해서는 적절한 우선순위 설정, 패킷화, 재전송, 대응 방식을 고려할 필요가 있다.

표 5. 손실률에 따른 PSNR 변화

Table 5. PSNR difference according to the loss rate

Loss Rate	Point Cloud	Mesh	NeRF	3DGS
0.1%	-0.000	Load Failed	-0.075	-0.08
1%	-0.035	Load Failed	Load Failed	-17.041
10%	-0.228	Load Failed	Load Failed	-19.132

표 6. 손실률에 따른 SSIM 변화

Table 6. SSIM difference according to the loss rate

Loss Rate	Point Cloud	Mesh	NeRF	3DGS
0.1%	-0.000	Load Failed	-0.001	-0.000
1%	-0.005	Load Failed	Load Failed	-0.151
10%	-0.031	Load Failed	Load Failed	-0.223

표 7. 손실률에 따른 LPIPS 변화

Table 7. LPIPS difference according to the loss rate

Loss Rate	Point Cloud	Mesh	NeRF	3DGS
0.1%	0.000	Load Failed	0.000	0.000
1%	0.013	Load Failed	Load Failed	0.256
10%	0.031	Load Failed	Load Failed	0.335

3. 접근성

본 항목에서는 각 표현 방식을 사용하는데 있어 접근 용이성에 대해 서술한다. 기존 표현 방식인 포인트 클라우드와 메쉬의 경우, PC, 모바일, 웹 등 대부분의 플랫폼에서 지원할 정도로 접근성이 높다. Blender [20], Maya [21] 등 여러 3D 소프트웨어와 Unity [22], Unreal Engine [23]과 같은 게임 엔진에서 대부분 로드가 가능하며, 웹 환경에서도 Three.js [24] 등의 라이브러리를 이용해 WebGL 기반 렌더링이 가능하다. 반면, NeRF의 경우 신경망 기반 표현 방식으로 기존 표현 방식들과 렌더링 방식이 크게 다르고, 렌더링을 위해 신경망 구조와 가중치를 모두 알아야 하기 때문에 구현 방식이나 사용 기술에 따라라도 다르다는 점이 접근성을 낮게 만든다. NeRF 기술과 관련된 연구는 활발하게 이루어지고 있어 관련하여 공개된 코드들과 툴들이 많이 존재하지만, 설치 및 사용이 복잡하여 일반 사용자에게 서비스로 제공하기 위해서는 이런 문제들이 해결되어야 한다. 또한 여러 형태의 신경망, 기술을 읽을 수 있는 표준 표현 방식 혹은 API가 정의될 필요가 있다. 일부 툴 [15]에서는 NeRF로 훈련된 모델을 기반으로 하여 고품질의 포인트 클라우드, 메쉬의 형태로 내보낼 수 있도록 지원을 하고 있으며, NeRF 모델의 대체제로 이용이 가능하다.

표 8. 각 표현 방식 별 장단점 비교

Table 8. Advantages and Disadvantages of each 3D representations

Type	Advantages and Disadvantages
Point Cloud	+ Simple, easy to use. Supported by many existing libraries and frameworks. Available even if loss occurs. Easy to edit (Beneficial for adaptive streaming). - Low visual quality when there are few points or low density. High visual quality requires high bandwidth.
Mesh	+ Simple, easy to use. Supported by many existing libraries and frameworks. Low bandwidth. Relatively easy to edit. - Low visual quality. Unavailable if loss occurs.
NeRF	+ High visual quality (photorealistic). A lot of research is currently ongoing (Disadvantages are likely to be resolved in the future). - Requires training. High bandwidth. Unavailable if loss occurs. Low FPS. Limited or no support by existing libraries and frameworks. Hard to edit.
3DGS	+ High visual quality (photorealistic) with lower bandwidth. High FPS. A lot of research is currently ongoing. Editable. - Requires training. Available even if loss occurs, but quality can be greatly reduced. Limited or no support by existing libraries and frameworks.

3DGS의 경우에는 명시적인 표현 방식으로 NeRF보다는 접근성이 높으며, Blender, Unity와 같은 3D 프로그램을 위한 플러그인 [25], 웹에서 로드하기 위한 뷰어 [26, 27] 등이 존재하지만, 기존 방식과 다른 형태의 렌더링 방식을 요구하기 때문에 기존 표현 방식들에 비해서는 접근성이 떨어진다.

4. 압축 알고리즘

본 항목에서는 높은 대역폭 및 저장 공간을 요구하는 3D 콘텐츠의 미가공 (Raw) 데이터를 더욱 효율적으로 전송하기 위한 압축 알고리즘의 측면에서 각 표현 방식을 비교한다. 압축 과정은 3D 콘텐츠의 용량을 크게 줄이는 과정 중 하나로, 많게는 대역폭 사용량을 기존 대비 1% 이하로 줄일 수 있다. 포인트 클라우드와 메쉬는 오랜 기간 연구되어 압축 알고리즘이 다양하게 존재한다. 포인트 클라우드의 경우에는 Draco [28], PCL [29]에서 지원하는 옥트리 (Octree) 기반 압축 방식, GROOT [30]와 같이 개선된 트리 기반 방식, MPEG의 PCC 표준이 존재한다 (V-PCC [31], G-PCC [32]). 메쉬 역시 Draco와 같은 라이브러리를 사용하여 압축이 가능하며, 특히 glTF [33] 포맷은 3D 모델 및 장면을 효율적으로 전송하기 위한 포맷으로, 기하 정보 뿐만 아니라 변형이 쉽다는 메쉬의 장점을 살려 애니메이션 등 움직임 정보까지 담을 수 있도록 설계되었다. NeRF의 경우 NeRF 기술에 대한 연구와 함께 압축을 위한 연구 [34, 35], 혹은 작은 모델로 효율적으로 학습하기 위한 연구 [36]도 다양하게 이

루어지고 있다. 3DGS의 경우 2023년에 제안된 기술로, 압축 방식에 관한 연구는 많지 않지만, 기존 포인트 클라우드, 메쉬를 저장하기 위해 사용한 파일과 구조 및 형식 면에서 유사하기 때문에 이를 응용할 수 있을 것으로 예상된다.

5. 기타 고려 요소

본 항목에서는 각 표현 방식을 사용하기 위해 추가로 고려해야 하는 요소들에 대한 설명을 작성한다.

먼저 렌더링 속도이다. 기본적으로 렌더링 속도는 GPU 성능에 의존하는데, 기존 표현 방식인 포인트 클라우드와 메쉬는 데이터 크기에 따라 다르지만, 일반적으로 빠른 렌더링 속도를 보여주며, 3DGS 역시 빠른 렌더링 속도를 특징으로 하고 있다. 그에 반해 대부분의 NeRF 방식들은 상대적으로 느린 렌더링 속도를 가지고 있다. 다만 이러한 한계를 극복하기 위해 FastNeRF [37] 등 빠른 렌더링 속도를 위한 연구도 활발히 이루어지고 있다.

다음은 사전 처리 시간이다. NeRF와 3DGS 기반 방식들은 3D 콘텐츠를 만들기 위해 모델 학습을 요구하며, 이에 따라 실시간 스트리밍, 화상회의 등 일부 어플리케이션에서는 적합하지 않을 수 있다. 다만 이 항목 역시 Instant-NGP [16] 등 적은 학습으로도 3D 복원을 가능하게 하는 연구가 많이 제안되고 있다.

다양한 네트워크 상황에 적응하기 위하여 2D 영상에서는 MDC [38], SVC [39] 등과 같은 방식으

로 다중 품질로의 변환을 통해 적응형 스트리밍을 구현할 수 있었다. 포인트 클라우드, 메쉬, 3DGS와 같은 명시적인 표현 방식은 공간상에서 빠르게 샘플링을 통해 품질 조정이 가능한데 반해, NeRF와 같은 암시적인 표현 방식은 이러한 샘플링이 어렵다. NeRF를 스트리밍에 이용하기 위해서 2D 영상에 적용한 방법론을 적용하는 연구들이 진행되고 있다 [40, 41].

마지막으로는 사용자 시선에 따른 처리이다. ViVo [42], GROOT [30] 등 여러 기존 분류메트릭 영상 스트리밍 관련 연구들이 사용자 시선 내의 콘텐츠만 전송하여 대역폭을 절약한다. 3D 공간에서 유저의 위치와 3D 데이터의 위치를 알기 때문에 가능한 것으로, 포인트 클라우드, 메쉬, 3DGS와 같이 명시적인 표현 방식에서 빠르게 가능하지만, NeRF와 같은 암시적인 표현 방식에서는 모호한 부분이 존재한다. 한 콘텐츠에 대해 하나의 신경망이 아닌 위치에 따라 여러 개의 신경망을 활용하는 연구 [43]를 활용할 수 있다.

표 8은 본 장에서 설명한 네트워크 측면에서의 여러 고려 요소들에 대해 각 방식 별로 장단점인 부분들을 정리한 표이다.

V. 한계점

본 연구에서는 네 가지 3D 콘텐츠 표현 방식에 대한 분석 및 관련 연구, 기술들을 제시하였다. 하지만 몇 가지 한계점이 존재하는데, 분석을 위해 같은 이미지 셋에 대한 3D 모델을 구축하였지만, NeRF와 3DGS의 경우 품질과 용량에 영향을 줄 수 있는 요소가 너무 많다. NeRF 혹은 3DGS를 기반으로 하는 각 방식마다 사용하는 신경망의 크기, 데이터 처리 및 저장 방식 등이 모두 다르며, 학습을 위한 파라미터 세팅에 따라서도 용량이 크게 달라질 수 있다. 본 연구에서는 NeRF와 3DGS 각각 한 가지 방식만을 사용하여 평가하였고, 파라미터 결정을 위해서도 다른 요소들보다 용량에 주는 파라미터들을 주로 고려하였다. 이에 따라 다른 세팅으로 3D 모델을 제작하는 경우에는 용량-품질 간 관계가 본 연구에서의 결과와 크게 달라질 수 있다. 또한 본 연구에서는 간단한 콘텐츠 중 하나인 lego 씬을 대상으로 연구를 수행하였지만, 더욱 크고 복잡하거나, 실제 환경을 보여주는 콘텐츠의 경우에는 다른 양상을 보일 수 있다. 이러한 다양한 장면들과 방식을 고려하기 위해서는

Nerfbaselines[10]처럼 다양한 모델과 데이터에 대한 단일화된 API를 만들어 제공하는 방식이 가능하다.

VI. 결 론

본 연구에서는 네트워크 측면에서 3D 콘텐츠 표현 방식에 대한 이해를 돕기 위한 분석 및 관련 연구를 제시하였다. 같은 3D 콘텐츠에 대해 포인트 클라우드, 메쉬, NeRF, 3DGS, 총 네 가지 표현 방식으로 제작한 후, 서비스 품질에 영향을 주는 대역폭, 패킷 손실 등의 측면에서 분석을 수행하고, 압축 알고리즘, 적응형 스트리밍 등 서비스 및 네트워크 품질 관리 측면의 몇 가지 고려 요소들에 대한 관련 연구와 이해를 제시하였다. 향후 연구로는 여러 3D 콘텐츠에 대해 전송과 더불어 실제로 렌더링을 수행하는 시스템을 구축하여 더욱 종합적인 비교 및 사용자 평가를 수행할 것이다.

References

- [1] Prescient & Strategic Intelligence, Extended Reality Market Research Report (2022), Retrieved Nov., 14, 2024, from <https://www.psmarketresearch.com/market-analysis/extended-reality-xr-market-insights>.
- [2] International Data Corporation, Slower Growth for AR/VR Headset Shipments in 2023 but Strong Growth Forecast Through 2027, According to IDC (2023), Retrieved Nov., 14, 2024, from <https://www.idc.com/getdoc.jsp?containerId=prUS50511523>
- [3] Mildenhall, Ben, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." in European Conference on Computer Vision (pp. 405 - 421). 2020.
- [4] Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". ACM Transactions on Graphics 42, no.4 (2023).
- [5] Yili Jin, et al. "From Capture to Display: A Survey on Volumetric Video." arXiv preprint arXiv:2309.05658 (2023).
- [6] Yili Jin, et al. "A Networking Perspective of Volumetric Video Service: Architecture, Opportunities and Case Study". IEEE Network (2024): 1-1.
- [7] Bangya Liu, and Suman Banerjee. "SwinGS: Sliding

- Window Gaussian Splatting for Volumetric Video Streaming with Arbitrary Length." arXiv preprint arXiv:2409.07759 (2024).
- [8] Sambugaro, Zeno, et al. "3D reconstruction methods in industrial settings: a comparative study for COLMAP, NeRF and 3D Gaussian Splatting." in *ITALIA 2024* (2024).
- [9] Liu, Kaiyan, et al. "Toward next-generation volumetric video streaming with neural-based content representations." In *Proceedings of the 1st ACM Workshop on Mobile Immersive Computing, Networking, and Systems*, pp. 199-207. 2023.
- [10] Kulhanek, Jonas, and Torsten Sattler. "NerfBaseline: Consistent and Reproducible Evaluation of Novel View Synthesis Methods." arXiv preprint arXiv:2406.17345 (2024).
- [11] COLMAP, Retrieved Nov., 14, 2024, <https://github.com/colmap/colmap>.
- [12] Schonberger, Johannes L., and Jan-Michael Frahm. "Structure-from-motion revisited." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104-4113. 2016.
- [13] Schönberger, Johannes L., et al. "Pixelwise view selection for unstructured multi-view stereo." In *Computer Vision - ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III* 14, pp. 501-518. Springer International Publishing, 2016.
- [14] Zhou, Qian-Yi, et al. "Open3D: A modern library for 3D data processing." arXiv preprint arXiv:1801.09847 (2018).
- [15] Tancik, Matthew, et al. "Nerfstudio: A modular framework for neural radiance field development." . In *ACM SIGGRAPH 2023 Conference Proceedings* (pp. 1 - 12).2023.
- [16] Muller, Thomas, et al. "Instant neural graphics primitives with a multiresolution hash encoding". *ACM transactions on graphics (TOG)* 41, no.4 (2022): 1 - 15.
- [17] Brunet, Dominique, et al. "On the mathematical properties of the structural similarity index". *IEEE Transactions on Image Processing* 21, no.4 (2011): 1488 - 1499.
- [18] Zhang, Richard, et al. "The unreasonable effectiveness of deep features as a perceptual metric." . In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 586 - 595).2018.
- [19] torchmetrics, Retrieved Nov., 14, 2024, <https://github.com/Lightning-AI/torchmetrics>.
- [20] Blender, Retrieved Nov., 14, 2024, <https://www.blender.org>.
- [21] Maya, Retrieved Nov., 14, 2024, <http://www.autodesk.co.kr/products/maya/overview>.
- [22] Unity, Retrieved Nov., 14, 2024, <https://unity.com>.
- [23] Unreal Engine, Retrieved Nov., 14, 2024, <https://www.unrealengine.com>.
- [24] Three.js, Retrieved Nov., 14, 2024, <https://threejs.org>.
- [25] UnityGaussianSplatting, Retrieved Nov., 14, 2024, <https://github.com/aras-p/UnityGaussianSplatting>.
- [26] Polycam Gaussian Splatting Viewer, Retrieved Nov., 14, 2024, <https://poly.cam/tools/gaussian-splatting>.
- [27] 3D Gaussian Splatting for Three.js, Retrieved Nov., 14, 2024, <https://github.com/mkkellogg/GaussianSplats3D>.
- [28] Draco 3D Data Compression, Retrieved Nov., 14, 2024, <https://google.github.io/draco>.
- [29] Radu Bogdan Rusu, and Steve Cousins. "3D is here: Point Cloud Library (PCL)." . In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011.
- [30] Lee, Kyungjin, et al. "GROOT: a real-time streaming system of high-fidelity volumetric videos." . In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (pp. 1 - 14).2020.
- [31] MPEG V-PCC, Retrieved Nov., 14, 2024, <https://www.iso.org/standard/83535.html>.
- [32] MPEG G-PCC, Retrieved Nov., 14, 2024, <https://www.iso.org/standard/78990.html>.
- [33] glTF 2.0 Specification, Retrieved Nov., 14, 2024, <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>.
- [34] Girish, Sharath, et al. "Shacira: Scalable hash-grid compression for implicit neural representations." . In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 17513 - 17524).2023.

- [35] Li, Lingzhi, et al. "Compressing volumetric radiance fields to 1 mb." . In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4222 - 4231).2023.
- [36] Barron, Jonathan T, et al. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields." . In Proceedings of the IEEE/CVF international conference on computer vision (pp. 5855 - 5864).2021.
- [37] Garbin, Stephan J, et al. "Fastnerf: High-fidelity neural rendering at 200fps." . In Proceedings of the IEEE/CVF international conference on computer vision (pp. 14346 - 14355).2021.
- [38] Goyal, Vivek K. "Multiple description coding: Compression meets the network". IEEE Signal processing magazine 18, no.5 (2001): 74 - 93.
- [39] Schwarz, Heiko, et al. "Overview of the scalable video coding extension of the H. 264/AVC standard," in IEEE Transactions on circuits and systems for video technology, vol. 17, no. 9, pp. 1103-1120, 2007.
- [40] Cho, Junwoo, et al. "Streamable neural fields." . In European Conference on Computer Vision (pp. 595 - 612).2022.
- [41] Yang, Guo-Wei, et al. "Recursive-nerf: An efficient and dynamically growing nerf". IEEE Transactions on Visualization and Computer Graphics 29, no.12 (2022): 5124 - 5136.
- [42] Han, Bo, et al. "ViVo: Visibility-aware mobile volumetric video streaming." . In Proceedings of the 26th annual international conference on mobile computing and networking (pp. 1 - 13).2020.
- [43] Reiser, Christian, et al. "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps." . In Proceedings of the IEEE/CVF international conference on computer vision (pp. 14335 - 14345).2021.

유 상 우 (Sangwoo Ryu)



2020 포항공과대학교, 컴퓨터공학과 학사
 2020 ~ 포항공과대학교, 인공지능대학원 석/박사 통합과정
 <관심분야> 2D/3D 비디오 스트리밍, 영상회의, 인공지능

홍 원 기 (James Won-Ki Hong)



2020~현재 포스텍 크립토펙트체인연구센터 공동센터장
 2019-2023 포스텍 교육혁신센터장
 2012~2014 KT 종합기술원 원장
 2008~2012 포스텍 정보전자융합공학부장

합공학부장

2008~2010 포스텍 컴퓨터공학과 주임교수
 2007~2010 포스텍 정보통신연구소 연구소장
 2007~2011 포스텍 정보통신대학원장
 1995~현재 포스텍 컴퓨터공학과 교수

1992~1995 Univ. of Western Ontario, 연구교수

<관심분야> AI 기반 네트워크관리, SDN/NFV, 무크기반 교육, 블록체인 및 암호화폐, Vmeeting 화상회의시스템

한국통신학회 통신망운용관리연구회

KNOM Review Vol.27/ No.2

ISSN : 2287-1543(Online)

2024년 12월 31일 인쇄

2024년 12월 31일 발행

발행인/ 석승준

편집인/ 김명섭 편집위원장

발행처/ 한국통신학회 통신망운용관리연구회
서울시 서초구 서초동 1330-8 현대기림오피스텔 1504동 6호
전화 : 02-3453-5555
홈페이지 : www.knom.or.kr/knom-review/