www.knom.or.kr

ISSN : 2586-0232

2021
통신망운용관리 학술대회 논문집

Proceeding of KNOM Conference 2021

일시 : 2021년 4월 29일(목) ~ 4월 30일(금)
장소 : Vmeeting/포항공대 (온라인/오프라인 동시)
주관 : 한국통신학회, 통신망운용관리연구회 (KICS KNOM)
주최 : 한국과학기술정보연구원 (KISTI)



한국통신학회 통신망운용관리연구회

초대의 말씀

한국통신학회 통신망운용관리연구회 (KNOM)는 2021년 통신망 운용관리 학술대회 (KNOM Conference 2021)를 통하여 통신망 운용관리 기술의 최신 연구 개발 현황을 국내 관련분야 학자, 연구원, 네트워크 관리자, 및 실무 담당자들에게 소개하고, 활발한 토론을 할 수 있는 장을 마련하고자 합니다.

네트워크와 컴퓨팅 기술은 최근 급속하게 발전하고 변화하고 있습니다. 날로 고속화되는 무선기술과 차세대 인터넷에 대한 지대한 관심은 이러한 네트워크 기술 발전을 대변하고 있습니다. 또한 클라우드 컴퓨팅, 모바일 컴퓨팅 등 향후 컴퓨터의 사용 개념을 혁신할 새로운 컴퓨팅 기술이 각광을 받고 있습니다. 또한 영상 데이터의 급속한 확산은 유무선을 포함한 모든 네트워크에서 상상하기 어려운 새로운 데이터 폭주로 이어지고 있으며 언제 어디서나 동영상을 볼 수 있는 컴퓨팅 체계가 마련되고 있습니다. 빠른 통신기술의 발전과 보급은 무선 Data Explosion이라는 새로운 문제를 야기하고 있고 이를 해결하는 것이 통신 사업자의 가장 큰 이슈가 되어, 지난 10여 년간 통신망 운용관리 분야의 주요 연구주제였던 End-to-End 네트워크 관리는 유무선 통합 네트워크 환경에서 네트워크와 서버 및 단말을 포함해 관리해야 하는 현실적인 문제로 대두되었습니다. 또한 클라우드 컴퓨팅의 보편화는 네트워크 구조와 트래픽의 흐름을 근본적으로 바꾸어가고 있으며, 네트워크와 서비스에 대한 보안침해도 급격히 증가하여 통신망 운용관리 분야의 연구와 개발 범위 또한 급속히 넓어지고 그 중요성이 더욱 강조되고 있습니다.

이러한 추세를 반영하여 KNOM Conference 2021에서는 통신망 전반에 대한 모델링, 설계, 서비스 제공, 운용 관리 및 보안 기술 분야의 최신 연구 개발 결과에 대한 유익한 정보제공과 토론의 장을 제공할 계획입니다. 본 학술대회에 통신망 운용관리와 관련된 연구소, 학계, 통신서비스 사업자, 통신기기 제조업체 등에서 많은 분들이 참석하여 실제적인 기술을 습득하고 토론할 수 있는 좋은 기회가 될 수 있기를 바랍니다.

2021. 02. 04.

2021 통신망운용관리학술대회 운영위원장 석우진
한국통신학회 통신망운용관리연구회 위원장 석우진

운영위원회

운영위원장	석우진(KISTI)	**홍보 및 출판**	조부승(KISTI)
학술	김경백(전남대)	**등록 및 예산**	최미정(강원대)
자문	홍충선(경희대), 최덕재(전남대), 송왕철(제주대), 김영탁(영남대), 유재형(포항공대), 주홍택(계명대)		

프로그램

일시: 2021년 4월 29일(목)
발표장소: 포항공과대학교 및 KNOM 화상회의 시스템
§ **On-line**: https://kreo.vmeeting.io/room/knom_plenary
§ **Off-line**: 포스코 국제관 2층 중회의실 204-A호

시간	주요 행사	발표자
12:30 - 13:00	코로나 대응지침에 의한 사전 준비	
13:00 - 14:00	등록	
	튜토리얼	
14:00 - 14:50	Klaytn API Service (KAS)를 이용한 블록체인 앱 개발	사상민 (그라운드X)
14:50 - 15:00	Coffee Break	
	개회식/기조강연	
15:00 - 15:20	개회사	KNOM2021 학술대회 위원장 석우진 센터장(KISTI)
15:20 - 16:00	기조강연: Artificially Intelligent Cities	KNOM 위원장 Charlie Catlett(ANL)
16:00 - 16:10	Coffee Break	
16:10 - 16:50	기조강연: P4 네트워크 프로그래밍 및 NPB 제품 개발	정지모 ((주)플랜티넷)
	Wrap-Up	
16:50 - 18:00	KNOM OC Wrap-Up meeting	OC

프로그램

일시: 2021년 4월 30일(금)
발표장소: 포항공과대학교 및 KNOM 화상회의 시스템
§ Track Ⅰ: https://kreo.vmeeting.io/room/knom_track1
§ Track Ⅱ: https://kreo.vmeeting.io/room/knom_track2

시간	내용	
09:00 - 09:30	등록	
	Track Ⅰ	Track Ⅱ
09:30 - 10:40	**TS1.** AI기반 네트워크 관리 (좌장: 최태상)	**TS2.** 클라우드 관리 (좌장: 김윤희)
10:40 - 10:50	Coffee Break	
10:50 - 12:00	**TS3.** AI기반 네트워크 서비스 (좌장: 김우태)	**TS4.** 네트워크 응용 (좌장: 석승준)
12:00 - 13:30	Lunch	
13:30 - 14:40	**TS5.** 유무선 네트워크 관리 (좌장: 고한얼)	**TS6.** 블록체인 관리 (좌장: 박세진)
14:40 - 14:50	Coffee Break	
14:50 - 16:00	**TS7.** 네트워크 보안 1 (좌장: 최미정)	**TS8.** 블록체인 응용 (좌장: 박세진)
16:00 - 16:10	Coffee Break	
16:10 - 17:20	**TS9.** 네트워크 보안 2 (좌장: 김명섭)	**TS10.** 가상 네트워크 관리 기술 (좌장: 주홍택)
17:20 - 17:50	우수논문시상 및 폐회식	
16:50 - 18:00	Wrap-Up meeting	

논문 목차

Technical Session 1. AI기반 네트워크 관리

TS1 - 1	**그래프 신경망을 이용한 최적 VNF 관리 시스템** 김희곤, 유재형, 홍원기 (포항공과대학교, POSTECH)	1 - 4
TS1 - 2	**인공지능 기반 네트워크 정책 도구 학습을 위한 KOREN Data추출 및 RouteNet 적용** 조현준, 정혜선, 김경백 (전남대학교, Chonnam National Univ.)	5 - 8
TS1 - 3	**머신러닝을 이용한 서버 장애 예측 기반 VNF Live Migration** 정세연, 유재형, 홍원기 (포항공과대학교, POSTECH)	9 - 12
TS1 - 4	**Deep Q-Networks 기반 Service Function Chaining 구성 연구** 이도영, 유재형, 홍원기 (포항공과대학교, POSTECH)	13 - 16

Technical Session 2. 클라우드 관리

TS2 - 1	**로그 및 자원 분석을 통한 VNF 고장 예측에 관한 연구** 남석현, 홍지범, 유재형, 홍원기 (포항공과대학교, POSTECH)	17 - 20
TS2 - 2	**그리드 컴퓨팅 시스템의 보안 위험성 탐구** 황선홍, 엄익채 (전남대학교, Chonnam National Univ.)	21 - 23
TS2 - 3	**페이로드 시그니처를 이용한 Adobe 소프트웨어 사용자 행위 탐지** 이민성, 최정우, 권윤주, 박지태 (고려대학교, Korea Univ.)	24 - 25
TS2 - 4	**SNI 정보를 활용한 Office 365 사용자 행위 탐지** 최정우, 박지태, 이민성, 권윤주 (고려대학교, Korea Univ.)	26 - 27

Technical Session 3. AI기반 네트워크 서비스

TS3 - 1	**웹 기반 영상회의에서 딥 러닝 모델 적용을 위한 방법 연구** 유상우, 고경찬, 홍원기 (포항공과대학교, POSTECH)	28 - 31
TS3 - 2	**기계학습을 이용한 Office365 서비스 분류 알고리즘 성능 평가** 권윤주, 최정우, 이민성, 박지태 (고려대학교, Korea Univ.)	32 - 33
TS3 - 3	**멀티 스텝 트래픽 예측을 위한 Bi-LSTM 인코더 디코더 모델** 염성웅, 최철웅, Hong-Nam Quach, 김경백 (전남대학교, Chonnam National Univ.)	34 - 37
TS3 - 4	**뉴스 데이터를 이용한 머신러닝 기반 비트코인 가격 등락 예측** 강민규, 김보선, 신희종 (고려대학교, Korea Univ.)	38 - 39

논문 목차

Technical Session 4. 네트워크 응용

TS4 - 1	**IoT System Architecture for Battery Digital Twin** Ardiansyah, Yeonghyeon Kim, Deokjai Choi (전남대학교, Chonnam National Univ.)	40 - 43
TS4 - 2	**군집 연합학습을 이용한 태양광발전량 예측 연구** 유은근, 고한얼 (고려대학교, Korea Univ.)	44 - 46
TS4 - 3	**스마트 공장을 위한 IEEE 802.11n 기반의 무선 네트워크 설계 및 시뮬레이션** 김민철, 김영탁 (영남대학교, Yeungnam Univ.)	47 - 50
TS4 - 4	**Edge Computing Assisted Deep Reinforcement Learning-based Approach for Smart Grid Stability Detection** Luyao Zou, Choong Seon Hong (경희대학교, Kyung Hee Univ.)	51 - 54

Technical Session 5. 유무선 네트워크 관리

TS5 - 1	**Evaluation of BBRv2 and PCC Vivace algorithms over KREONet** Tergel Munkhbat, Busueng Cho (KISTI)	55 - 58
TS5 - 2	**Optimal Resource Allocation and Association in Space-Aerial Assisted Networks** Nway Nway Ei, Choong Seon Hong (경희대학교, Kyung Hee Univ.)	59 - 61
TS5 - 3	**Sensing based Candidating for Resource Allocation in C-V2X Sidelink Mode 4** Moin Ali, Young-Tak Kim (영남대학교, Yeungnam Univ.)	62 - 65

Technical Session 6. 블록체인 관리

TS6 - 1	**블록체인 기반 CBDC 아키텍처 및 트랜잭션 키 관리 시스템 설계** 한정수†, 김정헌†, 허강욱*, 전윤서*, 우종수†, 홍원기† (포항공과대학교†, POSTECH†; 하나금융*, Hana Financial Group*)	66 - 69
TS6 - 2	**비트코인 노드의 압축 블록 전달 지연 원인 분석** 김애리, 주홍택 (계명대학교, Keimyung Univ.)	70 - 80
TS6 - 3	**Gossip 기반 P2P 라우팅을 통한 블록체인 성능 개선 연구** 최원석, 강창훈, 홍원기 (포항공과대학교, POSTECH)	81 - 84
TS6 - 4	**이더리움 노드 탐색 프로토콜 분석을 위한 와이어샤크 해석기 개발** 김정연, 주홍택 (계명대학교, Keimyung Univ.)	85 - 87

논문 목차

Technical Session 7. 네트워크 보안 1

TS7 - 1	**네트워크 침입 탐지를 위한 Stacked Denoising Autoencoder-Deep CNN 모델** 이종화, 김종욱, 최미정 (강원대학교, Kangwon Univ.)	88 - 91
TS7 - 2	**네트워크 침입 탐지 성능 향상을 위한 최적화 연구** 이지원, 김형태, 김경백 (전남대학교, Chonnam National Univ.)	92 - 94
TS7 - 3	**STIX 기반 사이버 위협 인텔리전스를 활용한 산업제어시스템 네트워크 위협 행위 분석 방법** 신동혁, 엄익채 (전남대학교, Chonnam National Univ.)	95 - 97
TS7 - 4	**Opcode 디스어셈블러와 기계 학습 기반의 트랜잭션 패턴 분석을 이용한 이더리움 스캠 컨트랙트 및 악성 유저 탐지 시스템 설계** 이채현, 고경찬, 우종수, 홍원기 (포항공과대학교, POSTECH)	98 - 101

Technical Session 8. 블록체인 응용

TS8 - 1	**비트코인과 이더리움의 구조적 차이에 따른 지갑 주소 클러스터링 방법 비교 및 분석** 신혜영, 주홍택 (계명대학교, Keimyung Univ.)	102 - 108
TS8 - 2	**이더리움 네트워크에서 액티브 프루빙을 이용한 정확도 높은 토폴로지 탐색 방법** 맹수훈, 주홍택 (계명대학교, Keimyung Univ.)	109 - 112
TS8 - 3	**사물인터넷(IoT)에서의 블록체인 활용 관련 연구동향 분석** 강창훈, 최원석, 우종수, 홍원기 (포항공과대학교, POSTECH)	113 - 116
TS8 - 4	**차량 블랙박스 영상 데이터의 위변조 및 프라이버시 문제 해결을 위한 경량화 블록체인** 나동준, 박세진 (계명대학교, Keimyung Univ.)	117 - 120

Technical Session 9. 네트워크 보안 2

TS9 - 1	**가상 네트워크 관리를 위한 기계학습 기반 이상 탐지 시스템 설계** 홍지범, 남석현, 유재형, 홍원기 (포항공과대학교, POSTECH)	121 - 123
TS9 - 2	**Unsupervised learning for anomaly detection in virtual network environment** Chungjun Lee†, Jibeom Hong*, Heeyoul Choi† (한동대학교†, Handong Global University†; 포항공과대학교*, POSTECH*)	124 - 126
TS9 - 3	**자연어 처리를 이용한 네트워크 트래픽 이상 탐지 기법** 김가영, 엄익채 (전남대학교, Chonnam National Univ.)	127 - 130

논문 목차

Technical Session 10. 가상 네트워크 관리 기술

TS10 - 1	Data Provisioning Through Data Models for Machine Learning Analysis Waleed Akbar, Afaq Muhammad, Talha Ahmed Khan, Wang-Cheol Song (제주대학교, Jeju National Univ.)	131 - 133
TS10 - 2	Location-aware Multi-Path Network Provisioning Hong-Nam Quach, Sungwoong Yeom, Kyungbaek Kim (전남대학교, Chonnam National Univ.)	134 - 137
TS10 - 3	A distributed, scalable, high-performance, and real-time data collection and processing architecture for a data-intensive science network Mazahir Hussain, Buseung Cho (KISTI)	138 - 141

그래프 신경망을 이용한 최적 VNF 관리 시스템

김희곤, 유재형, 홍원기
포항공과대학교 컴퓨터공학과,

{sinjint, jihyoo78, jwkhong}@postech.ac.kr

The VNF Management System using Graph Neural Network

Hee Gon Kim, Jae Hyung Yoo, James Won Ki Hong
Department of Computer Science and Engineering, POSTECH

요 약

가상 네트워크 환경은 유연한 네트워크 관리를 가능하게 하여 네트워크의 확장성을 증가시키고 유연한 변화가 가능하게 하였다. 본 논문은 가상 네트워크 환경에서 그래프 신경망 모델을 이용하여 최적의 Virtual Network Function (VNF) 관리 정책을 생성하는 모델을 제안하였다. 제안한 모델을 통하여 네트워크 관리자는 네트워크의 운영비용을 최소화하면서 사용자가 요구하는 네트워크 서비스를 만족하도록 VNF 를 적절하게 배치할 수 있다. 본 논문은 다양한 네트워크 환경에서 실험을 진행하였으며, 개발한 모델은 VNF 의 종류별로 최적의 설치 위치 및 필요 인스턴스 개수를 생성한다.

Ⅰ. 서 론

Software-Defined Networking (SDN) 과 Network Function Virtualization (NFV) 는 OPEX 와 CAPEX 를 감소시키며 네트워크를 동적이고 유연하게 만들어 준다. SDN 컨트롤러와 NFV 관리 시스템은 네트워크와 NFV 의 전반적인 뷰를 제공하며 네트워크의 Orchestration 을 가능하게 한다. 하지만 SDN/NFV 는 어디까지나 관리 기능을 제공할 뿐, 자체적인 네트워크 최적화 모델을 제공하고 있지는 않다.

기존의 네트워크 관리는 네트워크 관리자의 오랜 경험에 맡기거나 Integer Linear Programming 혹은 휴리스틱 알고리즘을 이용하여 최적화를 시도하였다. 하지만 현재 네트워크 관리 모델은 새로운 국면을 맞이하고 있다. SDN/NFV 의 도입으로 네트워크는 보다 유연한 관리가 가능해지게 되었으며, 동시에 거대하고 다양해진 차세대 네트워크의 등장으로 네트워크 관리의 복잡도는 기하급수적으로 증가되었다. 또한 5G 네트워크 등장은 네트워크 관리가 실시간으로 이루어져 네트워크의 변화에 대응하기를 요구하고 있으며 이러한 변화에 대해 기존의 방법들은 요구조건을 맞추고 있지 못한다. 네트워크 관리자가 현재의 복잡한 네트워크의 모든 요소를 고려하여 경험적으로 관리를 하는 것은 사실상 불가능에 가까우며, Integer Linear Programming 은 계산 시간에 따른 비용으로 네트워크 변화에 대응하기 어려우며 휴리스틱 방법은 관리 결과에 대한 무수한 경험 없이 성능을 보장할 수 없으며 동적으로 변화하는 네트워크에 대응하는 능력이 떨어진다[1].

최근 네트워크 관리에 기계 학습(Machine Learning)을 적용하려는 시도가 많이 이루어지고 있다[2]. 기계 학습을 이용하여 효율적인 네트워크의 자원 사용 및 서비스 품질 향상을 바라보고 있으며 자동화된 네트워크 관리의 구축 등 차세대 네트워크의 등장을 기대하고 있다. 하지만 여태까지 생명, 화학 등 많은 분야에서 기계 학습을 사용한 것과 달리 네트워크 분야에서 기계 학습의 적용은 불모지와 다름이 없었는데, 이는 네트워크 데이터와 네트워크 관리의 특이성 때문이라고 말할 수 있다. 복잡하고 다양한 네트워크 데이터의 학습은 기계 학습의 수행에 있어 하나의 큰 장애물로 존재하며 기계 학습이 수행해야할 정교한 네트워크 관리 기능도 또다른 장애물로 존재하였다. 네트워크 학습에 대한 어려움으로 현재 기계 학습을 이용한 네트워크 관리 연구는 비교적 간단한 네트워크 데이터를 사용하여 간단한 네트워크 관리 기능을 제공하고 있다[3].

본 연구는 기존의 연구에서 네트워크 데이터의 학습 및 관리 기능의 수행에 어려움을 겪는 이유를 학습 데이터를 이해할 수 있도록 알맞게 설계된 기계 학습 모델의 부재로 보고 새로운 학습 모델을 제안하였다. 본 논문에서는 네트워크 데이터 별로 Graph Neural Network(그래프 신경망 모델) 와 Feed Forward Neural Network 를 사용한 모델을 제안하였다. 제안한 모델은 그래프 신경망 모델을 이용하여 네트워크 토폴로지 구조를 학습하고, Feed Forward Neural Network 로 유저의 서비스를 학습하여 동적이고 변화하는 네트워크 서비스에 대해 고려를 하여 최적의 VNF 의 배치 정책을 생성하였다.

Ⅱ. 본론

그림 1 은 학습 모델은 개념도이다. 제안한 모델은 학습 단계 (Training Step)와 평가 단계(Test Step)로 최적의 VNF 배치에 대해 학습하고 평가를 받는다. 모델이 학습을 하기 위해서는 학습에 사용되는 데이터 셋이 필요하다. 데이터 셋은 특성 데이터(Feature Data)와 라벨링 데이터 (Labeling Data)로 구성이

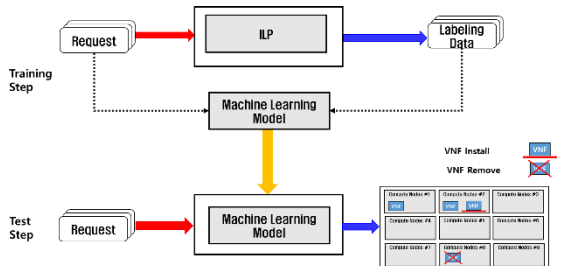


그림 1 학습 모델 개념도

되며 특성 데이터로 네트워크 토폴로지 및 자원 정보와 사용자의 서비스 요구 데이터를 사용하였다. 라벨링 데이터는 VNF 의 배치 정책이 되며 해당 데이터는 특수한 ILP[4]식을 이용하여 생성하였다. 자세한 특성 데이터와 라벨링 데이터의 내용은 아래와 같다.

- 특성 데이터
  - ▶ 서비스 정보
    - 트래픽의 출발지, 도착지 노드
    - 트래픽의 요구 대역폭
    - 트래픽의 최대 허용 지연 시간
    - 서비스 타입
  - ▶ 네트워크 정보
    - 토폴로지 구조
    - 노드의 CPU Core 의 수
    - 노드의 최대 허용 대역폭
    - 링크의 최대 허용 대역폭
    - 링크의 지연시간
    - 노드의 Idle, Peak 전력소모
- 라벨링 데이터
  - 최적의 VNF 배치 위치 및 인스턴스 수

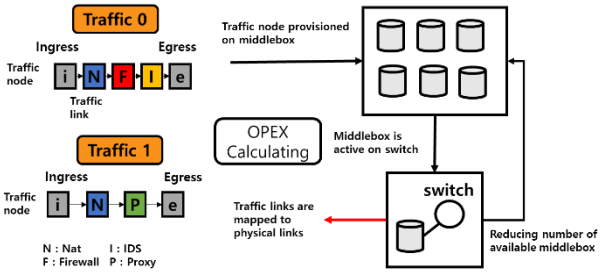


그림 2 Integer Linear Programming (ILP) 모델 개념도

그림 2 는 본 논문에서 사용한 ILP 모델의 개념도를 나타낸 것이다. ILP 는 서비스 정보와 네트워크 정보를 입력으로 받아 최적의 VNF 배치 위치 및 인스턴스 수를 생성한다. 그림 2 에서는 두개의 트래픽(Traffic1, Traffic 2)의 서비스 타입을 일련의 VNF 로 정의하였다. 서비스를 구성하는 VNF 들은 Traffic node 로 정의되며 OPEX 를 최소화하도록 특정 Switch 에 설치된다. 이때, 초기에는 Traffic Node 의 수에 대해 특별한 제한을 두지 않지만 계속해서 이진 탐색으로 가장 적은 수의 Traffic Node 가 Switch 에 설치될 수 있도록 정답을 찾는다. ILP 가 계산을 모든 계산을 완료하게 되면 가장 적은 수의 VNF 의 인스턴스를 가장 OPEX 가 최소화되도록 특정 Switch 에 설치하게 된다. 또한 이때, ILP 는 하나의 서비스에 대해서만 최적화를 하는 것이 아닌, 모든 서비스에 대한 최적의 정답을 생성한다.

ILP 식이 고려하는 네트워크 OPEX 는 VNF 설치 비용, 서버의 에너지 사용 비용(Idle & Peak), 패킷 전송 비용, SLA 위반 비용, 자원 단편화 (Resource Fragmentation Cost)로 총 5 가지의 비용을 최소화한다. ILP 의 계산 시간은 네트워크의 복잡도에 따라 계산 시간이 비례하게 증가하며, 동시에 고려하는 서비스의 수에 따라 지수적으로 시간이 증가한다. ILP 의 계산 시간은 네트워크 관리에 있어 단점이 되지만 현재 제안하는 모델은 ILP 가 계산한 데이터를 오프라인 데이터로 학습하여 사용하기 때문에 해당 문제가 해결되었다.

특성 데이터에 대한 자세한 시나리오는 다음과 같이 구성하였다. 우리는 다양한 환경에서의 실험을 위해 총 2 개의 서비스 데이터 셋 A, 서비스 데이터 셋 B 를 구성하였다. A 데이터 셋의 서비스 데이터는 33Mbps 에서 38Mbps 의 대역폭을 가지며 B 데이터 셋의 서비스 데이터는 330Mbps 에서 380Mbps 의 대역폭을 가진다. 두 데이터 셋은 대역폭을 제외하고는 모든 설정을 공유한다. 네트워크가 한번에 고려해야하는 서비스의 수를 서비스 복잡도 $n(\pi)$ 라고 했을 때, π 의 비율은 1:2:3:4=0.14:0.33:0.36:0.17 이다. 최대 허용 지연시간은 700ms 에서 750ms 사이의 값을 무작위 하게 배정받으며 서비스 타입과 타입의 비율은 표 1 와 같이 정의되었다.

표 1 서비스 카탈로그

서비스 ID	서비스 타입	비율
1	NAT-Firewall-IDS	0.3
2	NAT-Proxy	0.4
3	NAT-WANO	0.3

본 논문에서는 총 5 개의 VNF 를 고려하였으며 각 VNF 는 표 2 와 같이 정의되었다. 우리는 모든 노드가 동일한 16 개의 CPU Core 를 가진다고 가정하였으며 가용 CPU Core 수를 다르게 정의하는 것으로 VNF 의 설치에 제약을 주었다. 서버의 Idle 에너지는 80.5W 이며 Peak Energy 는 2735W 로 정의하였다.

표 2 VNF 카탈로그 [5]

VNF Type	Required CPU core	Processing Capacity	Processing Delay
Firewall	2	900 Mbps	45ms
Proxy	2	900 Mbps	40ms
IDS	4	600 Mbps	1ms
NAT	1	900 Mbps	10ms
WANO	2	400 Mbps	5ms

우리는 다양한 네트워크 토폴로지를 고려하기 위해서 총 4 개의 네트워크 토폴로지를 학습하였다. 각각의 토폴로지는 A, B, C, D 의 이름을 부여 받았으며 그림 3 에 제시되었다. 토폴로지 A 는 실험에서 가장 기본이 되는 토폴로지로 Internet2 토폴로지의 구성을 가져온 것이다. 해당 토폴로지는 15 개의 노드와 15 개의 링크로 구성이 되었다. 토폴로지 B 는 토폴로지 A 를 압축한 구조로 총 8 개의 노드와 10 개의 링크로 구성이 되어있다. 토폴로지 C 는 데이터 센터 네트워크 토폴로지로 토폴로지 A 와의 비교를 위해 동일한 15 개의 노드로 구성이 되었으며 링크의 수는 2 개 더 많은 17 개로 구성되었다. 토폴로지 D 는 Mesh 토폴로지로 네트워크 복잡도에 따른 모델의 성능을 평가하기 위해 토폴로지 B 와 같이 8 개의 노드로 구성이 되었으며 링크의 수는 17 개이다.

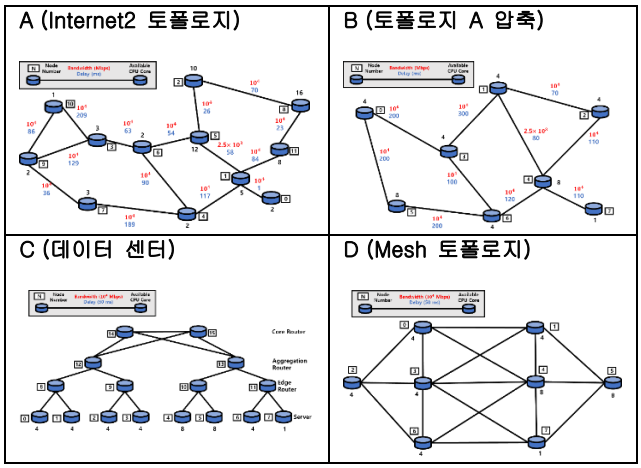


그림 2 토폴로지 A, B, C, D

본 논문에서는 GNN 모델 중 하나인 Edge-conditioned Filters in Convolutional Neural Network(ECGNN)[6]을 사용하여 네트워크 데이터를 학습하였다. 네트워크 데이터를 기계 학습 모델에 학습시킬 때, 가장 고려해야할 부분은 네트워크 구조에 대한 모델의 이해도 향상이다. FNN 과 같은 보통의 모델로 그래프 데이터를 학습 시킬 경우, 노드와 링크의 특성 데이터는 학습시킬 수 있지만 노드와 노드 간의 연결 관계는 학습 시키기 어렵다. 이는 노드와 노드 간의 연결 관계를 데이터로 표현하기 어렵기때문인데, 기계 학습이 이러한 데이터를 이해하기 위해서는 그래프를 Adjacency 매트릭스로 표현을 한 뒤, 매트릭스를 1 차원으로 펼쳐 준비를 해야한다. 이렇게 표현된 데이터는 굉장히 Sparse 한 구조를 가져 메모리적으로 비효율적이며 학습적인 측면에서도 제대로 된 학습결과를 얻기 어렵다. 하지만 ECGNN 과 같이 그래프 신경망 네트워크를 사용할 경우 노드의 특성 데이터에 전체 그래프 구조가 이미 학습되어 표현되어 있기 때문에, 노드와 링크의 특성데이터만으로 기계 학습 모델이 온전히 네트워크 구조를 이해할 수 있게 된다. 즉, GNN 을 이용하여 그래프 데이터를 기계 학습이 쉽게 이해할 수 있는 데이터로 표현하는 과정을 한번 거치고 추가로 네트워크를 최적화하는 학습 모델을 구성하는 것으로 그래프에 대한 모델의 이해도 향상은 물론 최적의 네트워크 관리 모델을 기계 학습을 통해 구현할 수 있게 되었다.

그림 4는 논문에서 제안한 모델로 GNN과 FNN 을 이용하여 네트워크 정보 및 서비스 정보를 학습하고 최적의 VNF 배치 결과를 생성하는 모습을 표현하고 있다. 서비스 데이터와 네트워크 데이터는 페어로 이루어져 있으며 유저의 요구 서비스가 변경되는 시점마다 서비스 데이터와 네트워크 데이터가 생성된다. 서비스 데이터는 유저가 요구하는 여러 서비스 정보가 존재하며 각 서비스는 FNN 모델을 통해 학습하게 되며, FNN 은 학습 파라미터인 Weight 을 공유한다. 네트워크 데이터는 노드, 링크, 토폴로지 구조로 분리된 다음 그래프 형태의 데이터로 변환이 되어 GNN 에 모델에 입력된다. 이후 FNN 과 GNN 모델은 합쳐지고 2 개의 추가 FNN 모델에서 학습을 진행하고 Fully Connected Layer (FCN)을 지나 결과값을 생성한다. 그림 5 는 모델이 생성한 라벨링 데이터를 표현한 것이다. 모델은 서비스 데이터 및 네트워크 데이터를 입력으로 받아 모든 서버들이 특정한 타입의 VNF 를 몇 개 설치해야 하는 지에 대한 정보를 제공한다.

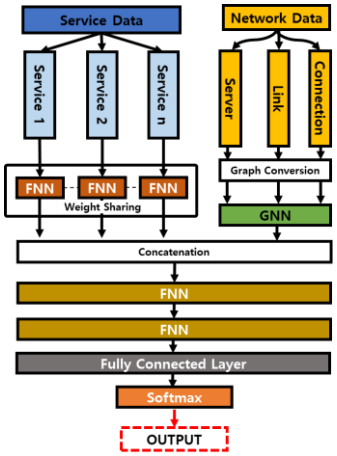


그림 4 GNN-FNN VNF Deployment 모델

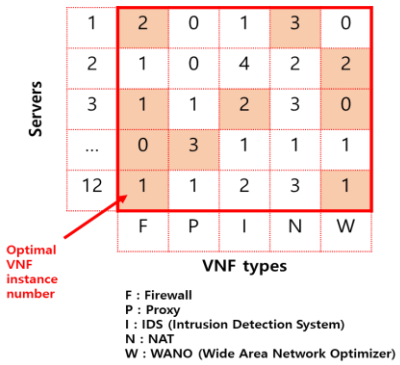


그림 5 라벨링 데이터

III. 실험

본 논문에서는 네 가지의 서로 다른 실험을 진행하였다. 첫번째 실험은 학습 데이터 수 및 서비스 복잡도에 따른 모델의 학습 성능 평가이다. 우리는 모델을 구현하는 도중 학습에 사용하는 데이터의 수 및 네트워크에 요구되는 서비스의 수에 따라서 모델의 성능이 급격하게 변화하는 것을 확인하였다. 표 3 은 토폴로지 A 와 서비스 데이터 셋 A 를 사용한 실험에 대한 결과이다.

표 3 학습 데이터 수 및 서비스 복잡도에 따른 성능 평가

서비스 복잡도	학습 데이터 수			
	1 주	2 주	5 주	10 주
Π=1	100%			
Π≤2	95.0%	98.3%	98.8%	100%
Π≤3	90.9%	91.2%	91.8%	92.3%
Π≤4	88.7%	88.6%	88.6%	88.6%

학습 데이터 수가 증가할수록 모델의 성능이 증가하였으며 서비스 복잡도가 증가할수록 학습 데이터의 성능이 감소하였다. 학습 데이터의 수와 모델의 성능은 자명한 결과이므로 서술은 넘어가겠다. 서비스 복잡도와 모델의 성능에 대한 관계는 다음과 같다. 서비스 복잡도가 증가할수록 모델은 더 많은 데이터를 입력 받기 때문에 모델의 복잡도는 증가하게 된다. 특히 입력 데이터의 선형적인 증가는 모델의 복잡도는 지수만큼 증가시키게 되는데, 이러한 결과로 1 주에서 10 주로 선형적인 학습 데이터 수의 증가는 서비스

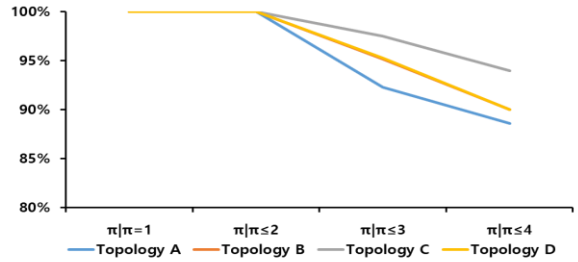


그림 6 토폴로지와 서비스 복잡도에 따른 모델의 성능

복잡도가 2 이하인 경우에서는 5% 증가라는 유의미한 결과를 만들었지만 서비스 복잡도가 4 이하인 경우에서는 유의미한 결과를 만들지 못하였다.

두 번째 실험은 토폴로지와 서비스 복잡도에 따른 모델의 성능 비교 실험이다. 우리는 서비스 데이터 셋 A 를 이용하여 앞서 정의한 4 개의 네트워크 토폴로지에서 실험하였다. 실험 결과는 그림 6 으로 표현되며 우리는 서비스 복잡도가 높을 경우, 네트워크 토폴로지 구조에 따라 모델의 성능이 다를 수 있음을 파악하였다. 서비스 복잡도가 1 일때, 모델은 토폴로지와 관계 없이 100%의 성능을 보였지만 서비스 복잡도가 증가할수록 토폴로지의 성능 차이가 발생하였다. 서비스 복잡도가 4 이하일 때, 토폴로지 A 네트워크가 성능이 가장 많이 감소하였고 B 와 D 가 같은 성능의 감소를 보였으며 C 가 성능 저하가 가장 적었다. 흥미롭게도 토폴로지 C 는 데이터 센터 네트워크 구조를 가지고 있으며 A 는 Mesh 네트워크 구조를 나타내고 있다.

세 번째 실험은 네트워크의 복잡도에 따른 노드(서버) 별 VNF 배치 정확도 실험이다. 우리는 토폴로지의 각 노드가 연결하고 있는 이웃 노드의 수와 VNF 의 배치 정확도의 관계성을 파악하고자 노력하였다. 노드 별 VNF 배치 정확도는 표 4 와 같이 나타났다. 실험 결과를 보면 흥미롭게도 연결된 이웃 노드의 수가 많은 노드일수록 VNF 배치 정확도가 떨어지는 것으로 나타났다. 이는 실험 2 에서 Mesh 네트워크 구조 토폴로지 A 가 데이터센터 네트워크 토폴로지 C 보다 성능이 낮은 것과 일맥상통하는 부분이 있다. 노드들의 연결이 더 많아질수록 네트워크는 Mesh 네트워크 형태가 되며 노드들의 VNF 배치 정확도는 감소하여 전체 네트워크에 대한 모델의 성능이 저하되었다.

표 4 토폴로지의 노드 별 VNF 배치 정확도 (이웃 노드의 수)

노드 번호	토폴로지			
	A	B	C	D
0	91% (1)	93% (1)	98% (2)	91% (4)
1	83% (4)	94% (1)	97% (3)	95% (4)
2	91% (2)	96% (1)	98% (2)	97% (3)
3	90% (3)	95% (1)	95% (3)	94% (6)
4	89% (3)	95% (1)	98% (4)	95% (6)
5	83% (3)	95% (1)	97% (2)	97% (3)
6	89% (3)	96% (1)	96% (3)	93% (4)
7	94% (2)	99% (1)	100% (1)	99% (1)
8	95% (2)			
9	95% (3)			
10	99% (2)			
11	90% (2)			

IV. 결론

본 논문은 기계 학습을 이용한 네트워크 관리 모델을 제시하고 있다. 해당 모델은 GNN 과 FNN 을 사용하여 네트워크 데이터와 서비스 데이터를 학습하고 최적의 위치에 적절한 타입의 VNF 를 알맞은 수만큼 설치하여, VNF 설치 비용, 서버의 에너지 사용 비용, 패킷 전송 비용, SLA 위반 비용, 자원 단편화 등의 네트워크 관리 비용을 감소하였다. 본 논문에서는 동적으로 변화하는 네트워크 환경을 고려하였으며 4 개의 서로 다른 네트워크 토폴로지를 사용하여 다양한 환경에서 실험을 진행하고 모델의 성능을 평가하였다. 향후 연구로 동시에 고려할 수 있는 서비스의 수를 증가시켜 실제 네트워크 환경에서 바로 적용이 가능한 모델을 만드는 것을 계획하고 있다.

ACKNOWLEDGMENT

본 연구는 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발, IITP-2021-2017-0-01633* 대학 ICT 연구센터지원사업)

참 고 문 헌

[1] Lange, Stanislav, et al. "Predicting VNF Deployment Decisions under Dynamically Changing Network Conditions." *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE, 2019.

[2] Cheng, Yang, et al. "Bridging machine learning and computer network research: a survey." CCF Transactions on Networking 1.1 (2019): 1-15.

[3] Boutaba, Raouf, et al. "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities." Journal of Internet Services and Applications 9.1 (2018): 1-99.

[4] Bari, Md Faizul, et al. "On orchestrating virtual network functions." 2015 11th International Conference on Network and Service Management (CNSM). IEEE, 2015.

[5] Lange, Stanislav, et al. "Predicting vnf deployment decisions under dynamically changing network conditions." 2019 15th International Conference on Network and Service Management (CNSM). IEEE, 2019.

[6] Gori, Marco, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains." *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*. Vol. 2. IEEE, 2005.

인공지능 기반 네트워크 정책 도구 학습을 위한 KOREN Data추출 및 RouteNet 적용

조현준, 정혜선, 김경백

전남대학교

limdugmin2@gmail.com, albaneo0724@gmail.com, kyungbaekkim@jnu.ac.kr

Applying KOREN Data extraction and RouteNet for training artificial intelligence-based network policy tools

Hyeonjun Jo, Hyeseon Jeong, Kyungbeak Kim

Chonnam Univ

요 약

정부가 비대면 산업을 육성하며 사회간접자본을 디지털화하는 '디지털 뉴딜' 정책을 발표하면서 5G 융복합 사업에 대한 수요와 산업 현장의 스마트화를 시도하는 기업이 늘어나고 있다. 스마트화가 진행될수록 사용자 및 서비스의 급격한 증가로 인한 네트워크 슬라이스 정책 결정의 복잡도는 점점 더 상승하고 있다. 이를 효율적으로 해결하기 위해 인공지능 기반의 네트워크 정책 결정 도구를 사용하고 있다. 하지만, 인공지능 기반의 네트워크 정책 결정 도구 학습을 위해 네트워크 종단 간 발생한 Data(Delay, Bandwidth, Packet, Drop 등) 수집 어려움이 존재한다. 그래서 본 논문은 인공지능 기반의 네트워크 정책 결정 도구 학습에 필요한 네트워크 종단 간 발생한 Data(Delay, Bandwidth, Packet 수, Drop 수)를 네트워크 시뮬레이션 OMNeT++ 이용하여 Data 추출 방법을 설명하며, OMNeT++ 환경에서의 초 연결형 지능형 연구 개발망(KOREN) 네트워크를 구성하여 인공지능 기반의 네트워크 정책 결정 도구 학습을 위한 Data(Delay, Bandwidth, Packet 수, Drop 수) 추출 후, 선행 연구된 인공지능 기반의 네트워크 정책 결정 도구에 적용하여 초 연결형 지능형 연구 개발망(KOREN) 네트워크상의 종단 간 Delay 예측 후 OMNeT++ 환경에서 종단 간 Delay 비교를 수행하였다.

Ⅰ. 서 론

5세대에서는 서비스 유형에 따라서 각각 다른 기능과 구성을 갖는 전용의 네트워크가 필요하다.[1] 또한 새로운 문화 '언택트'가 탄생하면서 네트워크를 이용하는 사용자 및 서비스 증가는 급속도로 증가하고 있다.[2] 증가된 사용자 및 서비스 때문에 사용하는 네트워크 속도 저하가 발생할 수 있다. 속도 저하가 발행하거나 장애가 생길 경우 사용자 및 서비스 기업에게 많은 치명적인 결과가 발생할 수 있다.

물리적으로 하나의 네트워크를 통해 Device, Access, Transport, Core를 포함하여 종단 간 논리적으로 분리된 네트워크를 만들어 서로 다른 특성을 갖는 다양한 서비스들에 대하여 해당 서비스를 특화된 전용 네트워크를 제공해 주는 네트워크 슬라이스를 효율적이며 효과적으로 정책을 판단하는 인공지능 기반의 네트워크 정책 결정 도구가 필요하다.[3] 인공지능 기반의 네트워크 정책 결정 도구를 학습하기 위해서는 네트워크 종단 간 발생하는 네트워크 핵심성과지표[4]인 Data(Delay, Bandwidth, Packet 수, Drop 수 등)가 필요하다.[5] 하지만 네트워크 종단 간 발생한 실제 Data를 얻는 방법은 시간 및 비용 이 많이 걸리는 단점이 존재하여, 현실적으로 어렵다. 이에 네트워크 시뮬레이터인 OMNeT++를 이용하여 실제 네트워크를 구성할 수 있으며, 인공지능 기반의 네트워크 정책 결정 도구 학습에 필요한 Data(Delay, Bandwidth, Packet 수, Drop 수)를 얻을 수 있다.[6]

초 연결 지능형 연구개발망(KOREN)[7] 이란 전국 10개 대도시 지역 서울, 수원, 판교, 대전, 전주, 광주, 대구, 부산, 제주, 춘천을 10Gbps~360Gbps로 연결하는 백본망을 구축 운영하여 대용량 트래픽 전송이 가능한 국내 유일의 통합연구 시험망이다.

인공지능 기반의 네트워크 정책 결정 도구인 Graph Neural Network(GNN) 모델 기반의 RouteNet은[8] 네트워크 Topology 상에서 Routing table과 네트워크 핵심성과지표 중 Delay, Bandwidth, Packet, Drop 학습하고 Routing table에 따른 Delay를 예측한다. 추후 본 연구에서의 End to End Delay 예측을 수행할 RoutenNet은 언급한 Graph Neural Netwrok (GNN) 모델 기반의 RoutenNet을 통해 KOREN topology 상의 End to End delay를 예측한다.

본 논문에서는 OMNeT++환경에서의 인공지능 기반의 네트워크 정책 결정 도구를 학습하기 위한 네트워크 핵심성과지표 Data 중 Delay, Bandwidth, Packet 수, Drop 수 추출방법을 제안한다. 제안된 추출방법의 Data를 사용하여 KOREN topology 상 End to End Delay를 예측할 수 있다. 제안된 Data 추출 방법의 유효성을 검증하기 위해 RouteNet End to End Delay 예측 값과, OMNeT++환경에서의 End to End Delay 값을 비교한다.

Ⅱ. 관련 연구

Graph Neural Network(GNN) 기반의 RouteNet은 네트워크 상에서 End to End 네트워크 핵심성과지표 중 Delay, Jitter를 예측하는데 사용할 수 있다. 특히 네트워크 Topology, Routing table 및 입력 트래픽 간의 복잡한 관계를 학습하고 모델링이 가능하여 매우 높은 정확도로 End to End Delay, Jitter 예측이 가능하다[9].

그림 1는 (A)NSFNET topology (B)모델의 성능을 알아 보기 위한 실험 결과 이며, OMNeT++ 환경 NSFNET topology 상에서 발생한 네트워크 핵심성과지표 중 Delay, Bandwidth, Packet 수, Drop 수[10]를 학습

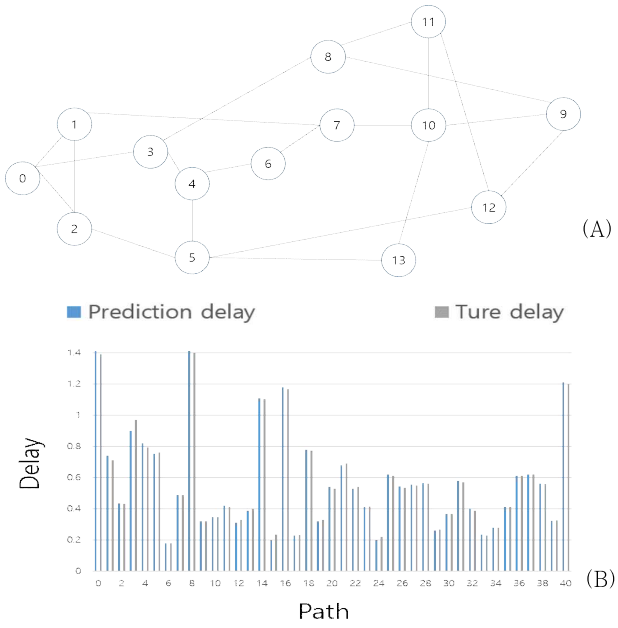


그림 1. (A) NSFNET topology (B) Delay 값 비교

후 End to End Path Delay 예측 값과 NSFNET 네트워크의 실제 측정 Delay 값을 비교하였다. 실험 결과 RouteNet의 예측 값이 네트워크 실제 측정값과 약 $1*10^{-2}$ 이하 오차범위가 발생하였다.

OMNeT++는 C++ 기반으로 개발한 객체지향 이벤트 기반 시뮬레이터 엔진이며, 통신 프로토콜, 컴퓨터 네트워크, 멀티프로세서, 분산시스템 등 다양한 형태의 시뮬레이션을 위해 사용되는 시뮬레이터이다.[11] 또한 매우 큰 크기의 시뮬레이션을 빠른 시간에 수행할 수 있도록 메시지 전달 인터페이스(MPI)를 기반으로 한 병렬 수행을 지원하는 시뮬레이터이다.

Ⅲ. KOREN topology 구성

1. OMNeT++에서 KOREN 네트워크 시뮬레이션 환경

실험에는 Window OS 용 OMNeT++ 시뮬레이터가 사용되었다. 그림 2는 KOREN 네트워크 구성 및 OMNeT++에서의 구현 환경이다.

2. Delay, Bandwidth, Packet 개수, Drop 개수 추출

KOREN topology는 10개의 Node로 구성되어 있으며, 각 Node 별 링크의 Bandwidth는 10Gbps, 100Gbps, 110Gbps, 360Gbps 총 12링크로 구성되어 있다. 하지만 링크가 1씩 존재한 Node는 링크가 파손 및 손실되면 통신할 수 없는 문제점 때문에, 기존의 KOREN topology에서 춘천과 대구, 대구와 부산, 전주와 제주 세 군데의 링크 10Gbps, 100Gbps를 추가하여 시뮬레이션 환경을 구성하였다.

시뮬레이션은 Routing table을 참조하여 시뮬레이션이 동작하며, 그림 3는 OMNeT++에서 인공지능 기반의 네트워크 정책 결정 도구를 학습하기 위한 Data 추출 동작 순서를 도식화하였다. End to End Bandwidth, Delay, packet 수, Drop 수를 얻기 위해 시뮬레이션 시간 동안 각 Node App(Node의 개수)에서 무작위로 생성된 Data packet은 최대 5.000byte 이하 모든 Node에게 전송한다, 각 Node에 존재하는 라우터에서 패킷 도착 시간 - 생성 시간, 총 생성 개수, 받은 개수를 저장 후 시뮬레이션이 끝나고, Statistic 파일에서 계산되어, 시뮬레이션 동안 발생 된 결괏값 [(1) End to End의 Bandwidth (2) End to End의 평균 Delay (3) End to End의 전송 Packet 수 (4) End to End의 Drop packet 수] 을 얻을 수 있다

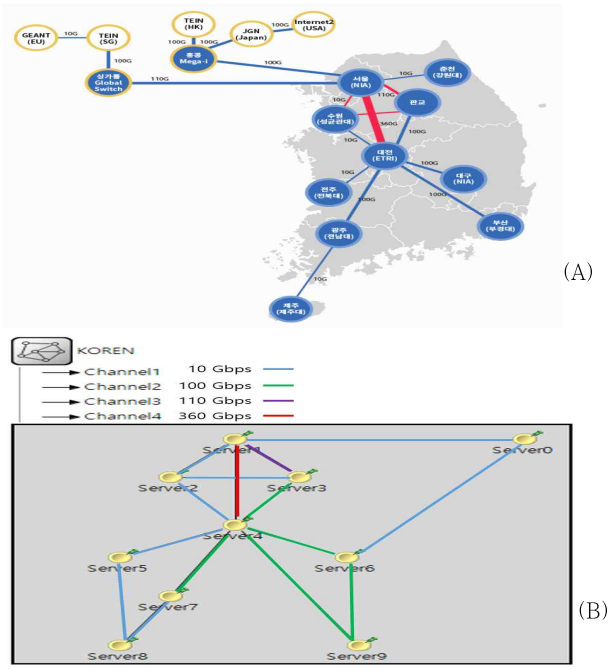


그림 2. (A) KOREN topology (B) OMNeT++ KOREN topology

3. KOREN 네트워크 DataSet

추출된 Data 들은 Result_(lambda).txt 형식으로 구성하였다. Result_(lambda).txt 파일은 네트워크 시뮬레이터인 OMNeT++에서 생성된 Node 들의 End to End Bandwidth, Delay, Num Packet, Drop Packet 을 포함한다. 추출된 DataSet의 lambda(트래픽 강도)는 시뮬레이션의 각 APP[Node의 개수]의 생성되는 Data packet의 비중과 lambda 변수 [9, 12, 15] 값을 곱해 최종 lambda 값이 설정되었다.

lambda 값 9, 12, 15를 설정한 이유는 동일 Routing table을 사용하지만 발생되는 트래픽의 비율을 다르게 하여, 큐잉 딜레이 발생하는 비율[15%, 25%, 50%] 설정하기 위해 9, 12, 15를 사용하였다. 시뮬레이션할 때 동일한 매개 변수(Routing table, lambda)를 사용하여 500번의 실험을 수행하였으며, 얻어진 Result_(lambda).txt 파일의 각 행은 Node[0-9] 10x10 행렬 Routing table에 의해 End to End에서 발행한 [(1) Bandwidth : End to End의 대역폭(kbps), (2) Delay : End to End에서 전송된 패킷에 대한 평균 Delay(s), (3) Numpacket : End to End에서 전송된 패킷 수(Number), (4) Drop : End to End에서 삭제된 패킷 수(Number)]이다.

Routing table은 Shortest Path로 구현하였으며, 총 10개의 Routing table을 사용하여 학습 및 예측에 사용할 KOREN topology DataSet을 구성하였다.

4. RouteNet 적용

학습할 때 사용한 DataSet은 Shortest Path Routing table 8개에서 발생한 [(1) End to End의 Bandwidth (2) End to End의 Delay (3) End to End의 Packet 수 (4) End to End의 Drop packet 수], DataSet를 사용하였으며, 2개의 Shortest Path Routing table에서 발생하는 End to End의 Delay를 예측한다.

5. RouteNet End to End Delay 예측 및 비교

10개의 Node에서 발생하는 End to End Delay를 예측하기 위해 표 1의 Routing table을 사용하였으며, 그림 4·5는 Routing table을 사용하여 RouteNet에 적용하여 얻은 End to End Delay 예측값 이다. 그림 6·7는 Route

Net을 사용한 10개의 Node에서 발생하는 End to End Delay 예측값과 네트워크 시뮬레이터인 OMNeT++ 상에서 얻어진 End to End Delay 비교 수행한 결과이다. 비교 수행한 결과 RouteNet의 예측 값이 OMNeT++ 측정값과 약 $1.5 * 10^{-1}$이하 오차 범위가 발생하였다.

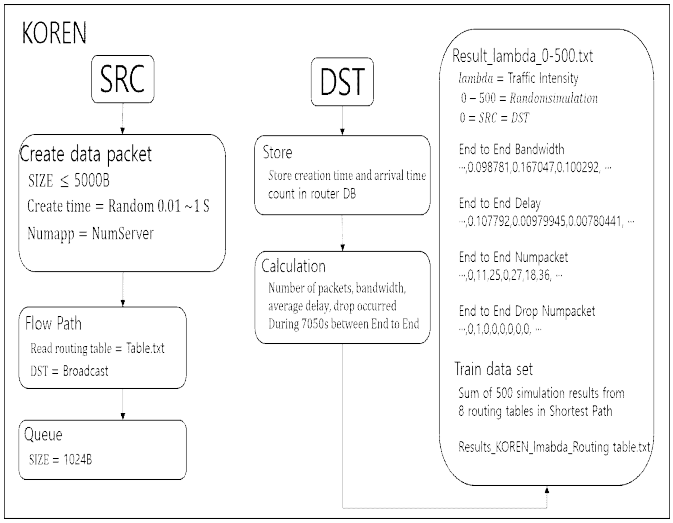


그림 3. KOREN Data추출 순서도

표 1. End to End Delay 예측 하기 위해 사용된 Routing table

Routing Table1	Routing Table2
-1,0,0,0,0,0,1,0,0,1,	-1,0,0,0,0,1,1,1,1,1,
0,-1,1,3,2,2,0,2,2,2,	0,-1,1,1,2,2,0,2,2,0,
0,0,-1,1,2,2,1,2,2,1,	0,0,-1,1,1,1,2,2,2,2,
0,0,1,-1,2,2,0,2,2,0,	1,0,1,-1,2,2,2,2,2,2,
1,1,0,2,-1,3,1,4,4,5,	6,1,0,2,-1,3,6,4,4,5,
0,0,0,0,0,-1,0,1,1,0,	0,0,0,0,0,-1,0,1,1,0,
1,1,0,0,0,0,-1,0,0,2,	1,1,0,0,0,0,-1,0,0,2,
0,0,0,0,0,1,0,-1,1,0,	0,0,0,0,0,1,0,-1,1,0,
1,1,1,1,1,0,1,1,-1,1,	1,1,1,1,1,0,1,1,-1,1,
1,0,0,0,0,0,1,0,0,-1	1,1,0,0,0,0,1,0,0,-1

Ⅳ. 한계점

초 연결 지능형 연구개발망(KOREN) topology 환경을 구축하여 RouteNet 학습에 필요한 네트워크 핵심성과지표 중 End to End Delay, Bandwidth, Packet 수, Drop 수 추출하였으며, 추출된 Data를 이용하여 GNN 기반의 RouteNet의 적용하여, End to End Delay 예측값과 OMNeT++ End to End Delay 값을 비교 검증을 수행하였다.

비교 검증의 한계점인 실제 KOREN topology 환경에서 발생한 Delay 값 비교 검증을 수행하지 못하였으며, 학습을 위한 시뮬레이션의 Data와 실제 Data의 차이에 대한 성능 평가하지 못하였다.

Ⅴ. 결론

5G 융복합 사업에 대한 수요와 산업 현장의 스마트화를 시도하는 기업이 점차 늘어남에 따라, 사용자 및 서비스가 급격하게 증가하였다. 이에 복잡도가 상승된 네트워크 슬라이스 정책 결정을 해결하기 위한 방법인 인공지능 기반의 네트워크 정책 결정 도구를 사용하여 해결하고 있다. 인공지능이 정확한 정책 결정을 하기 위해 실제 네트워크상 발생하는 핵심성과지표 Data로 학습하는 것이 좋다. 하지만 네트워크상 실제 Data를 얻는 방법은 현실적으로 많은 애로사항이 존재하다.

본 논문은 인공지능 기반의 네트워크 정책 결정 도구인 RouteNet을 학습하기 위한, 필요 네트워크 핵심성과지표 Data중 (1) End to End의 Bandwidth (2) End to End의 Delay (3) End to End의 Packet 수 (4) End to End의 Drop packet 수를 네트워크 시뮬레이션인 OMNeT++를 통해 얻는 방법을 제안하였다. 또한 얻은 Data를 이용하여 선행 연구된 GNN기반의 RouteNet의 적용하여 End to End Delay 예측값과 OMNeT++ 환경의 End to End Delay 값을 비교 검증을 수행하였다.

본 연구를 기반으로 네트워크 시뮬레이션 도구를 이용하여 초연결 지능형 연구 개발망(KOREN)을 유연하면서 즉각적으로 서비스 구성·운영 가능하게 만들기 위해 SDN 환경의 KOREN topology 환경을 구축하여, 인공지능 기반 네트워크 정책 결정 도구를 학습하기 위한 필요한 네트워크 핵심성과지표 Data 추출 후 적용할 것이다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음"(IITP-2021-2016-0-00314*)

참 고 문 헌

[1] koreaScience. 5G 망에서의 Network Slicing 요구사항 및 제공 구조. June, 2016.

[2] Cisco. 2020 글로벌 네트워킹 트렌드 보고서. 2020.

[3] ETRI. 네트워크와 AI 기술 동향. 2020.

[4] Zhiyao Xie et al."RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network". IEEE, Conference Paper, Nov, 2018.

[5] M. Ruiz et al. "End-To-End KPI Analysis in Converged Fixed-Mobile Networks". IEEE, July, 2020.

[6] A. Varga, "The omnet++ discrete event simulation system," in Proceedings of the European Simulation Multiconference (ESM), 2001.

[7] KOREN. 미래네트워크선도시험망(KOREN) 이용규정, 1028. (http://www.koren.kr/kor/)

[8] Krzysztof Rusek et al.. "RouteNet: Leveraging Graph Neural Networks for network modeling and optimization in SDN". IEEE, Journal on, Vol.38, pp. 2260-2270. June, 2020.

[9] Hyeseon Jeong, Seongwoong Yeom, Kyungbaek Kim, "지능형 네트워크 관리를 위한 RouteNet 분석", 2020 한국디지털콘텐츠학회.

[10] "Knowledge-defined networking repository," https://github.com/knowledgedefinednetworking/Papers/wiki/RouteNet:-Leveraging-GNN-for-network-modeling-and-optimization-in-SDN, 2019.

[11] OMNeT++ (https://omnetpp.org/documentation/).

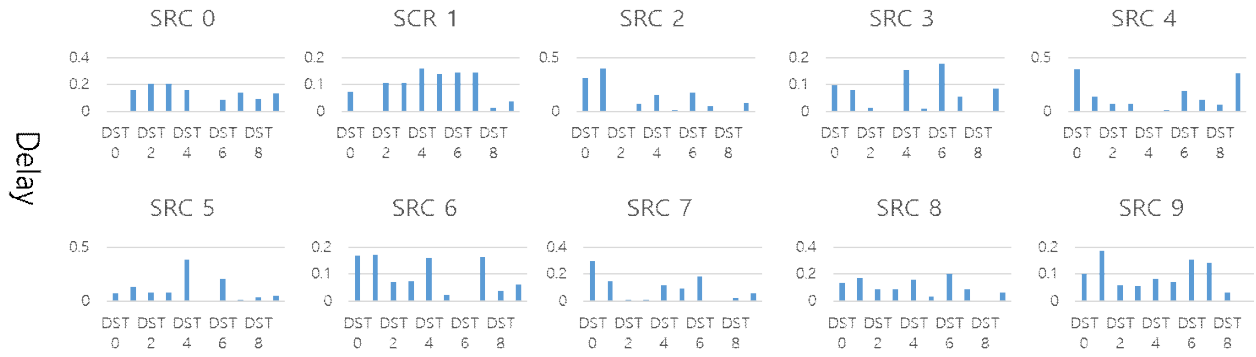


그림 4. Routing table 1 평균 Delay 예측

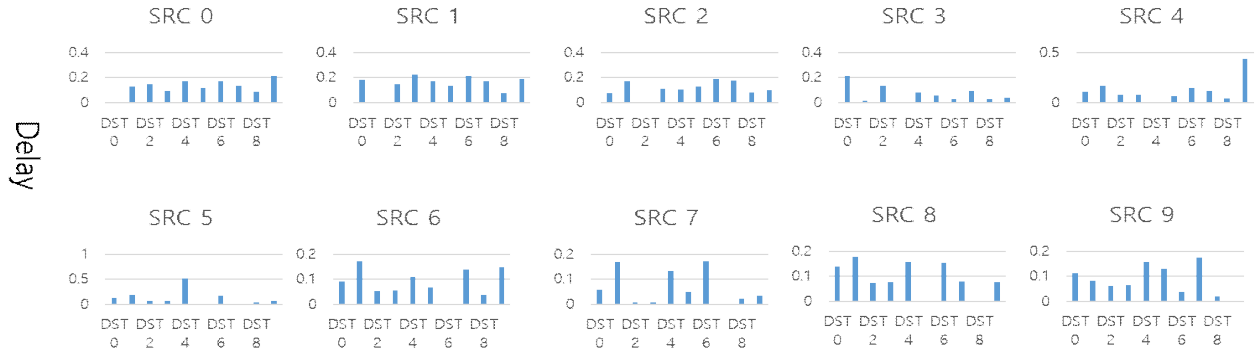


그림 5. Routing table 2 평균 Delay 예측

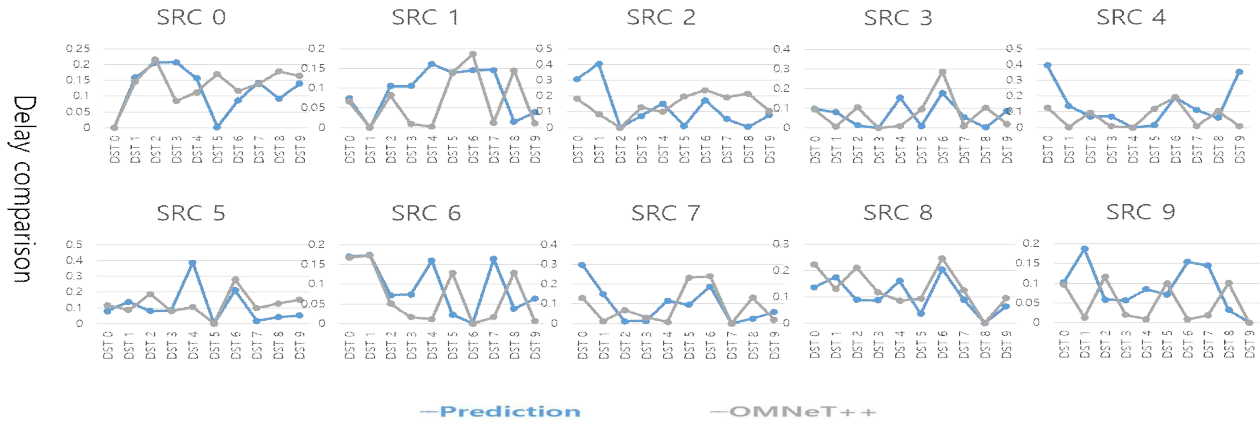


그림 6. Routing table 1 Delay 비교

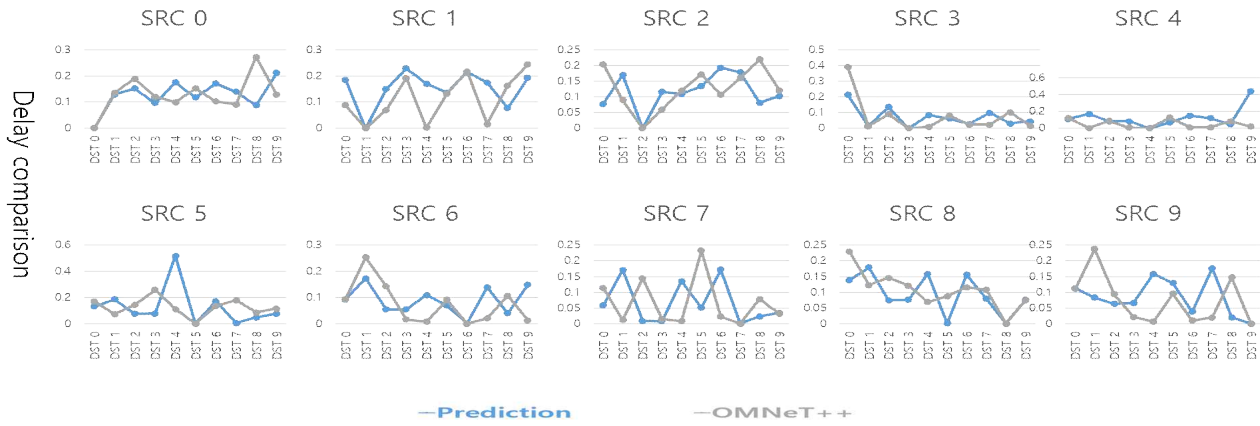


그림 7. Routing table 2 Delay 비교

머신러닝을 이용한 서버 장애 예측 기반 VNF Live Migration

정세연, 유재형, 홍원기
포항공과대학교 컴퓨터공학과

{jsy0906, jhyoo78, jwkhong}@postech.ac.kr

A VNF Live Migration method based on Server Anomaly Prediction using Machine Learning

Seyeon Jeong, Jae-Hyoung Yoo, James Won-Ki Hong
Department of Computer Science and Engineering, POSTECH

요 약

VM (Virtual Machine) live migration 은 동작중인 VM 을 중단시간을 최소화하면서 다른 서버로 이전시키는 기술을 말한다. 클라우드 데이터센터에서는 워크로드(workload) 및 네트워크 트래픽에 대한 로드 밸런싱 목적, 가동되는 서버와 스위치 개수를 최소화하여 전력 소비량을 감소시키는 목적, 서버의 하드웨어 및 소프트웨어 업데이트를 위한 VM 이전과 같은 유지보수 목적 등으로 VM live migration 을 활발히 사용하고 있다. 특히 고장 및 장애 징후를 사전에 탐지해서 예방(prevention) 또는 완화(mitigation)를 위한 수단으로 핵심적이다. 본 논문에서는 NFV (Network Function Virtualization) 환경에서 서버 장애에 따른 제어 복잡도 및 QoS 저하를 낮추기 위한 목적으로, 서버의 장애 예측에 기반한 VNF (Virtual Network Function) live migration 방법을 제안한다. 이는 각 서버에 대한 모니터링 데이터를 머신러닝으로 학습하여 자원 상태 변화에 대한 예측 모델을 생성하고, 예측 결과를 바탕으로 미래에 장애 가능성이 가장 낮은 서버를 최적의 migration 목적지로 선정한다. NFV 테스트베드에서의 실험을 통해 다른 migration 방법 대비 제안하는 방법의 향상된 장애 완화 효과를 검증한다.

Ⅰ. 서론

가상화 기술의 발전으로 다양한 서비스 어플리케이션들이 가상머신 (VM, Virtual Machine) 형태로 동작하게 되면서, 확장성(scalability), 기민성(agility), 신뢰성(reliability) 등 오늘날 클라우드 컴퓨팅 기술의 핵심 목표를 달성할 수 있게 되었다. VM migration 은 동작중인 VM 을 다른 서버로 이전시키는 기술로, 대상 VM 의 특정 시점의 이미지 파일(스냅샷)을 목적지 서버로 복사한 후 하이퍼바이저(hypervisor)를 통해 이미지를 불러들여 VM 을 다시 가동시킨다. VM 을 물리적으로 다른 서버로 이동시킬 수 있지만 이미지 전송 및 VM 이 부팅을 하는 동안에는 서비스를 제공할 수 없다는 문제가 있다.

이러한 문제점을 극복하기 위한 VM live migration [1] 기술은 VM 스냅샷을 목적지 서버로 복사해서 복사본 VM 의 가동 준비가 완료될 때까지 원본 VM 의 동작(서비스)를 유지시킨다. 따라서 두 VM 간의 메모리 동기화를 위한 freezing 시간 및 네트워크 경로 재설정을 포함하는 최소한의 서비스 중단시간(downtime)을 가진다 (그림 1). 이러한 VM live migration 기술은 특히 클라우드 데이터센터에서 워크로드(workload) 및 네트워크 트래픽에 대한 로드 밸런싱 목적, 가동되는 서버와 스위치 개수를 최소화해서 전력 소비량을 감소시키는 목적(VM consolidation), 서버의 하드웨어 및 소프트웨어 업데이트와 같은 정기적인 유지보수를 위해 동작중인 VM 을 다른 서버로 이전시켜서 서비스 제공을 유지하는 목적[2] 등으로 활발히 사용되고 있다. 이에 더하여 앞으로의 VM live migration 기술은 서비스 중단을 야기하는 고장 및 성능 저하를 일으키는 장애 징후를 사전에 탐지하여, 서비스 중단 시간과 사용자가 체감하는 성능 저하를 최소화하는 방향으로 발전되어야 한다. 이러한 고장 예방의 예시로, 장기간 과도한 CPU 사용률을 보이는 과열된 서버의 VM 에 대한 migration 작업을 들 수 있다.

NFV (Network Function Virtualization) 환경 또한 VNF (Virtualized Network Function)에 대한 live migration 을 통해 다양한 관리 효과를 얻을 수 있다. 하지만 NFV 는 SFC (Service Function Chaining)를 통해 여러 VNF 가 연관되어 있고, 다른 NFV 제어 기능(auto-scaling 등)들과 상호작용해야 하므로, 전통적인 VM live migration 방법을 그대로 도입할 수는 없다.

따라서 본 연구에서는 NFV 환경에서 서버 장애에 따른 제어 복잡도 및 QoS 저하를 낮추기 위한 목적으로, 서버 장애에 대한 예측에 기반한 VNF live migration 방법을 제안한다. 본 연구의 contribution 은 아래와 같다.

- 각 서버에 대한 모니터링 데이터를 머신러닝으로 학습하여 서버의 자원 상태 변화에 대한 예측 모델을 생성함.
- VNF live migration 을 통한 서버 장애의 대응 과정에서, 모든 서버의 자원 상태 변화를 예측하여 장애 가능성이 가장 낮은 서버를 migration 목적지로 선정하는 방법을 제안함.
- OpenStack 기반 NFV 환경에서 SFC 를 구성하여 제안하는 VNF live migration 방법을 검증함. 제안하는 방법은 다른 방법 대비 QoS 지표에서 약 15% 수준의 향상을 보임.

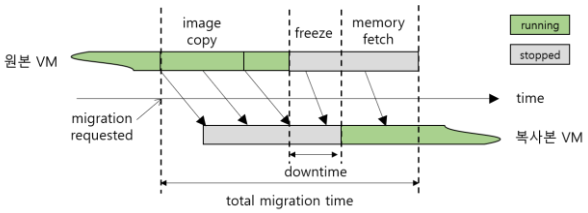


그림 1. VM live migration 동작 과정

Ⅱ. 관련 연구

VM/VNF live migration 에 대한 연구 방향은 아래와 같이 크게 세 가지로 나눌 수 있다. VM/VNF live migration 은 클라우드 데이터센터 관리에 필수적인 기능이지만 일시적인 서비스 중단이 발생하므로, migration 비용을 사용목적에 맞게 적절히 산정하여 적시 적소에만 수행할 필요가 있다.

1. 로드 밸런싱

[3]은 서버 및 네트워크의 자원 사용량에 대한 모니터링을 바탕으로 과부하(overloaded) 상태의 서버 및 링크를 식별하고, 관련된 VNF 들을 부하가 작은 다른 서버로 이전시켜 부하분산을 수행하는 방법을 제안하고 있다. [4]는 SFC reliability 에 대한 정의를 바탕으로 SFC 의 reliability 를 높일 수 있는 대상 VNF 와 목적지 서버를 식별하여 migration 을 수행하며, 결과적으로 migration 을 통해 과부하를 해소하는 방법을 제안하고 있다. [3], [4] 모두 SFC 성능 지표가 향상됨을 보이나, 시뮬레이션 환경에서 수행되어 전체 네트워크에 대한 상태 수집이 가능한 것으로 가정하고 있으며 실제 migration 기능은 시험하지 않았다는 한계가 있다.

2. 전력 소비량 감소

[5]는 데이터센터에서 서비스 성능을 보장하면서 전력 소비를 줄이도록 VM 을 배치하는 방법을 다룬다. 서로 상충되는 지표인 서비스 성능 및 전력 소비량을 강화학습의 보상(reward)에 포함시켜 최대 보상을 가지는 migration 정책(대상 VM 및 목적지 서버)을 학습한다. 강화학습을 통해 트래픽 변화 등 다양한 동적 조건에서 기존 휴리스틱(heuristic) 방식 대비 VM migration 횟수 및 전력 소비량이 감소됨을 보이나, NFV 환경 및 서버 고장/장애 상황은 고려하지 않고 있다.

3. 고장/장애 대응

[6]은 MEC (Mobile Edge Computing) 환경에서 core cloud 에 있는 특정 VNF 의 장애 발생 시, 해당 VNF 를 edge cloud 로 이전하여 운용하는 정책의 비용 적절성 문제를 다룬다. 이를 위해 migration 하지 않을 때의 operator loss 및 migration 할 때의 service path 재설정을 포함한 migration cost 를 모델링하여 전체 비용이 적은 정책을 따른다. 이 과정에서 edge user mobility 예측에 머신러닝을 사용한다. [7]은 운용중인 데이터센터에서 수집된 각 서버의 자원 상태 및 운영 정책을 머신러닝 feature 로 학습하여 미래의 고장 가능성을 예측한 뒤, 고장 점수가 상위권인 서버에서 동작하는 VM 을 하위권 서버로 migration 한다. [6], [7] 모두 migration 정책 결정에 머신러닝을 활용하며, 특히 [7]에서는 고장 예측을 통한 proactive VM live migration 의 필요성을 보이고 있다.

NFV 고장 대응 관련 VNF backup 방법은 master VNF 인스턴스에 대한 slave VNF 인스턴스를 별도로 두어, master VNF 의 고장 시 즉각적으로 slave VNF 로 절체한다 [8]. VNF live migration 방법 대비 중단시간이 작다는 장점이 있으나, master/slave 인스턴스의 실시간 동기화 비용과 over-provisioning 문제가 존재한다.

Ⅲ. 설계 및 구현

본 연구에서 제안하는 서버 장애 예측 기반 VNF live migration 시스템은 크게 Monitoring 모듈, Prediction 모듈, Migration 모듈로 구성되어 NFV MANO (Management and Orchestration) 시스템의 구성요소로 동작한다 (그림 2). NFV 환경은 OpenStack Victoria 릴리즈를 기반으로 VNF 및 SFC 의 제어를 수행하는 별도의 NFVO (NFV Orchestrator) 모듈을 개발하였다. 또한 향후 CNF (Cloud-native Network Function)에 대한 migration 지원을 고려하여 컨테이너(container)를 지원하도록 OpenStack Zun [9] 서비스를 구성하였다.

Monitoring 모듈은 서버 및 VM 의 자원 사용량과 가용 물리/가상 자원에 대한 모니터링 데이터를 주기적으로 수집하여 데이터베이스에 저장한다. 이를 구현하기 위해 각 서버에 collectd 를 모니터링 에이전트로 설치하여 CPU, 메모리, 하드디스크, 네트워크 I/O 사용량 등을 매초마다 수집하고 시계열 데이터베이스인 InfluxDB 에 저장하였다. VM 생성 시 collectd 가 설치된 VNF 이미지를 사용하여 관련 모니터링 프로세스를 자동화하였다. 또한 다른 모듈로부터 다양한 형태의 데이터 쿼리(예, 지난 1 시간 동안 특정 VM 의 CPU 사용량)에 응답하기 위해 swagger 기반 OpenAPI 서버 형태로 Monitoring 모듈을 구축하였다.

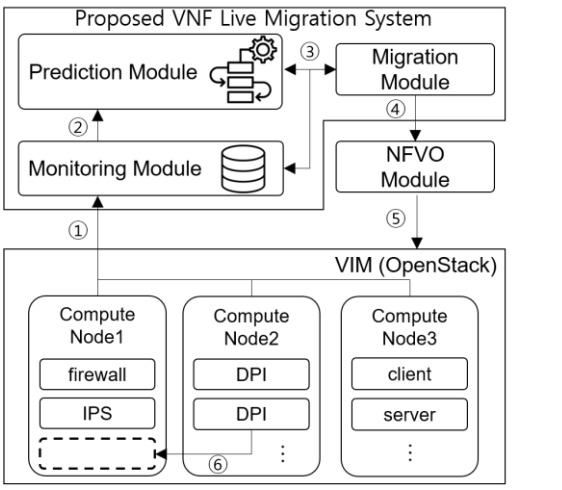


그림 2. 서버 장애 예측 기반 VNF live migration 구조도

Prediction 모듈은 서버 단위로 수집된 모니터링 데이터를 학습하여 서버의 자원 상태를 예측하기 위한 머신러닝 모델을 생성한다. 학습을 끝낸 모델은 Migration 모듈의 요청에 따라 미래 시점의 서버 자원 상태에 대한 예측을 바탕으로 장애 가능성이 가장 낮은 서버를 최적의 migration 목적지로 선정한다. 즉 전체 모델은 일정 기간 모든 서버의 자원 상태를 나타내는 벡터를 입력 받아 최적의 migration 목적지 서버를 출력하는 classification 문제로 정의된다. 이를 위해 서버의 자원 상태에는 일정한 패턴이 있음을 가정한다 (IV. 1 장 실험 시나리오 참고). 본 연구에서는 migration 목적지 선정을 위한 서버 장애 예측에 서버 수준의 모니터링 데이터만을 사용했으나, VM/VNF 수준의 데이터 및 SFC 정보(경로, 길이, SLA 등)를 결합해서 최적의 migration 대상 VNF 를 찾아볼 수 있다 (향후 연구). Prediction 모듈의 전체 모델을 구현하기 위해 Python scikit-learn 라이브러리의 SVM (Support Vector Machine) 알고리즘을 사용하였다.

Migration 모듈은 Monitoring 모듈로부터 VNF live migration 가동 조건(예, 특정 서버의 CPU 사용률 threshold 초과)이 발생하면 최적의 VNF live migration 정책을 결정한다. 본 연구에서는 해당 서버에서 동작하는 VNF 중 하나를 migration 대상으로 선정하며, Prediction 모듈에 질의하여 최적의 migration 목적지 서버를 선정한다. 최종적으로 OpenStack VM live migration API 에 (대상 VNF ID, 목적지 서버 ID)로 구성된 파라미터를 전달한다. 이 과정에서 NFVO 모듈과 연동한다.

IV. 검증

본 연구의 실험에 사용된 서버의 사양은 Intel Xeon 2.67GHz (12 코어), 24GB RAM 이며, OS 로 Ubuntu 18.04 를 사용했다. VNF 가 동작하는 VM 은 1 개의 CPU 코어 및 1GB 메모리를 가지도록 통일하였다. SFC 는 클라이언트의 서비스 요청이 firewall, IPS, DPI 를 순서대로 거쳐 웹 서버에서 응답을 반환하는 서비스를 가정했으며, OpenStack networking-sfc API 를 사용하여 구현하였다.

1. 실험 시나리오

실험에서 VNF live migration 은 특정 서버의 평균 CPU 사용률이 90% 이상일 때 가동된다. 이는 서버 장애 상황의 대표적인 예시로서, 해당 서버의 모든 CPU 코어가 saturated 된 상황을 의미하며 현재 동작중인 VM/VNF 또는 추가적인 서비스 요청에 대해 성능을 보장하지 못하는 상황을 가정한다. 이러한 서버 과부하 상황은 VM 이 예상치 못하게 down 되거나 CPU 과열로 인한 하드웨어 손상 문제로 이어질 수 있다 [10]. 이를 위해 다수의 SFC 에 트래픽을 동시 다발적으로 발생시키는 것에 더해, stress-ng 를 사용하여 모든 서버에 인위적인 부하를 주입하였다. 그림 3 은 시간에 따른 서버 부하 주입을 위한 학습 데이터의 예시를 보이며, 현재 tick 의 과부하 상태 서버에서 동작중인 VNF 를 미래 tick 에서 장애 가능성이 가장 낮은 서버로 migration 하는 정책(화살표)을 보인다. 주입되는 부하의 양과 패턴은 데이터센터에서 일정 주기에 걸쳐 반복되는 트래픽 및 워크로드를 반영하는 것으로 가정한다.

시간	Node1 CPU (%)	Node2 CPU (%)	Node3 CPU (%)	firewall	IPS	DPI
1	90	10	50	N1=>N2	N1	N3
2	100	10	50	N2	N1=>N1	N3
3	10	100	50	N2=>N3	N1	N3
4	50	85	10	N3	N1	N3
5	…			…		
…	확률적으로 일정 random 값 가감			최적 migration 정책 (정답지)		

그림 3. 부하 주입과 migration 정책의 학습 데이터 예시

2. 실험 결과

부하 주입을 통한 서버 자원 상태 예측 모델의 학습 이후, 유사한 부하 패턴을 테스트 데이터로 주입하여 제안하는 서버 장애 예측 기반 VNF live migration 정책을 검증하였다. 또한 migration 을 사용하지 않는 정책(no-mig), 목적지 서버를 임의로 선정하는 정책(random), CPU 사용률이 가장 낮은 서버를 선정하는 정책(least-cpu)과 QoS 성능을 비교하였다.

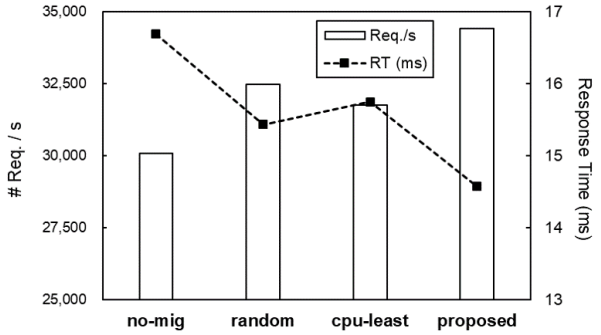


그림 4. VNF live migration 정책 별 SFC 성능

그림 4 는 각 VNF live migration 정책에 따른 SFC 의 성능 지표를 보인다. 초당 평균 request 처리량에 대해 no-mig 대비 random, cpu-least, 제안된 방식(proposed)은 각각 8%, 6%, 15% 증가를 보이며, 평균 response time 에 대해 8%, 6%, 13% 감소를 보였다. 주목할 점은 cpu-least 정책이 random 정책보다 성능이 낮은 것으로, 이는 CPU 사용률이 가장 낮은(least CPU utilization) 서버를 migration 목적지 서버로 선정하는 방식[3][4]이 직관적으로는 적절해 보이지만, 가까운 미래에 해당 서버에 장애(과부화 등)가 발생한다면 오히려 평균 성능이 저하될 수 있음을 의미한다. 제안하는 방식으로 선정된 migration 목적지 서버는 현재 시점에서는 최적이 아닐지라도 미래의 장애 가능성이 가장 낮기 때문에 평균 성능을 향상시킬 수 있다. 또한 본 실험에서는 CPU 과부하와 같은 (일시적인) 서버 장애 상황만을 고려했지만, 서버가 물리적으로 down 되는 고장 상황에서는 QoS 격차가 더욱 커질 것이다.

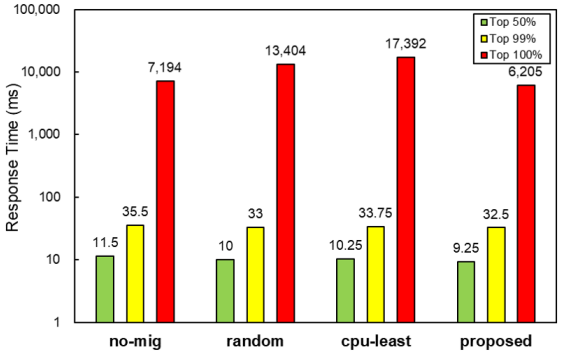


그림 5. VNF live migration 정책 별 response time 비율

그림 5 는 각 VNF live migration 정책에 따른 상위 50%, 99%, 100% response time 을 보인다. 상위 99% 까지의 response time 은 정책에 관계없이 정상적으로 낮은 값을 보이나, 상위 100%에 대해 random 및 cpu-least 정책에서 매우 높은 지연시간을 보이며, 제안하는 방식 또한 no-mig 와 비슷한 수치를 보인다. no-mig 에서 발생하는 tail latency 는 서버 장애에 따른 QoS 저하를 나타내는 반면, random, cpu-least 및 제안하는 방식의 tail latency 는 QoS 저하에 더해 migration 과정(그림 1)에서 발생하는 네트워크 경로 재설정에 따른 일시적인 packet loss 의 영향도 포함한다. 특히 VNF live migration 에서는 SFC 수준의 변경(SFC 경로 등)이 요구되므로 tail latency 에 민감한 서비스에 대해서는 migration 사용에 주의가 요구되며, 이를 줄이기 위한 관련 연구의 필요성을 제기한다.

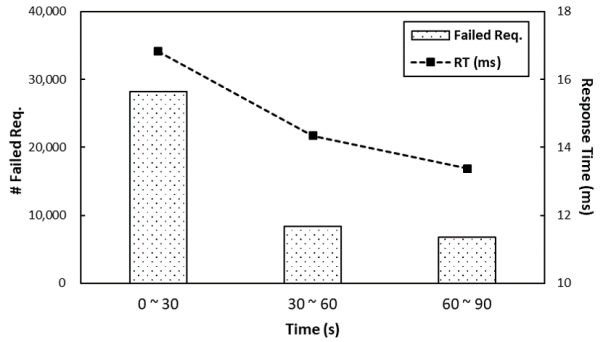


그림 6. VM live migration 요청 이후 시간 별 성능 변화

그림 6 은 동작중인 VM 을 live migration 했을 때 시간에 따른 timeout request 개수 및 평균 response time 의 변화를 보인다. 실험의 OpenStack 테스트베드에서 1 개의 CPU 코어 및 1GB 메모리를 가지는 Ubuntu VM 에 대한 live migration 요청 후 완료까지 평균적으로 45 초가 소요되었다 (OpenStack 로그 기준). 실험 결과에서 migration 요청 직후 30 초 동안은 장애 상태 서버의 원본 VM 에서 서비스를 처리하여 낮은 성능을 보이며, 30~60 초 구간에서 목적지 서버로의 VM migration 이 완료되어 성능이 향상됨을 확인할 수 있다. VM live migration 작업이 완료되는데 걸리는 시간은 VM 의 이미지 크기, 서버의 상태 등에 따라 달라질 수 있다.

V. 결론 및 향후 연구

본 연구는 VM/VNF live migration 기술 및 관련 연구를 소개하고, NFV 환경에서 서버 장애에 대한 완화 수단으로 서버 장애 예측 기반 VNF live migration 방법을 제안한다. 이는 서버의 자원 변화 상태를 머신러닝으로 학습해서 미래의 서버 장애 가능성을 예측하여 가장 적합한 migration 목적지 서버를 선정한다. OpenStack 기반 NFV 테스트베드에서 제안하는 방법의 장애 완화 효과를 QoS 측면에서 비교 검증하였다.

본 연구는 NFV 환경에서의 인공지능 기반 VNF live migration 기술에 대한 사전 연구로서 다양한 향후 연구 방향을 가진다. 첫째, 본 연구는 일종의 reactive VNF live migration 방식으로 임계치(threshold)를 통해 장애 여부를 판단한다. 반면 보다 정교하고 다양한 머신러닝 기술 및 로그(log) 분석 등을 통해 고장/장애 징후를 미리 파악할 수 있다면 proactive VNF live migration 을 통해 고장을 사전에 예방하고 경로 재설정에 따른 packet loss 와 같은 migration 비용을 줄일 수 있을 것이다. 둘째, 본 연구는 특정 SFC 하나를 최적화하기 위해 VNF live migration 을 수행하는 방법을 제안하고 있지만, 다양한 SFC 가 공존할 때 글로벌 최적화를 위한 migration 정책은 다를 수 있다. 이를 위해 SFC 의 end-to-end 지연 시간을 최소화하면서 여러 VNF 를 재배치하는 강화학습 모델을 적용하여 해결할 예정이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)과 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (IITP-2021-2017-0-01633, 대학ICT연구센터육성지원사업)

참 고 문 헌

[1] Jo, Changyeon, Youngsu Cho, and Bernhard Egger. "A machine learning approach to live migration modeling." Proceedings of the 2017 Symposium on Cloud Computing. 2017.

[2] Google Cloud Compute Engine. “Setting instance availability policies”. https://cloud.google.com/compute/docs/instances/setting-instance-scheduling-options. Accessed March 25, 2021.

[3] Yi, Bo, et al. "Design and implementation of network-aware VNF migration mechanism." IEEE Access 8 (2020): 44346-44358.

[4] Rui, Lanlan, et al. "Petri Net-Based Reliability Assessment and Migration Optimization Strategy of SFC." IEEE Transactions on Network and Service Management (2020).

[5] Basu, Debabrota, et al. "Learn-as-you-go with Megh: Efficient live migration of virtual machines." IEEE Transactions on Parallel and Distributed Systems 30.8 (2019): 1786-1801.

[6] Ibrahimpasic, Amina Lejla, Bin Han, and Hans D. Schotten. "AI-Empowered VNF Migration as a Cost-Loss-Effective Solution for Network Resilience." arXiv preprint arXiv:2101.09343 (2021).

[7] Lin, Qingwei, et al. "Predicting node failure in cloud service systems." Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018.

[8] Huang, Huawei, and Song Guo. "Proactive failure recovery for NFV in distributed edge computing." IEEE Communications Magazine 57.5 (2019): 131-137.

[9] OpenStack Foundation. “Welcome to Zun’s documentation!”. https://docs.openstack.org/zun/latest/. Accessed March 25, 2021.

[10] El-Sayed, Nosayba, et al. "Temperature management in data centers: Why some (might) like it hot." Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems. 2012.

Deep Q-Networks 기반 Service Function Chaining 구성 연구

이도영, 유재형, 홍원기

포항공과대학교 컴퓨터공학과

{dylee90, jhyoo78, jwkhong}@postech.ac.kr

A study on Deep Q-Networks-based Service Function Chaining Composition

Doyoung Lee, Jae-Hyoung Yoo, James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH

요 약

5G 시대에서는 급변하는 네트워크 환경 속에서 다양한 요구사항을 갖는 서비스들을 수많은 기기에게 제공하는 것이 요구된다. 이를 위해 5G 네트워크는 소프트웨어 정의 네트워킹 (SDN, Software-Defined Networking)과 네트워크 기능 가상화 (NFV, Network Function Virtualization)를 기반 기술로 활용하여, 유연하고 민첩하게 네트워크를 구축하고 운용한다. SDN/NFV는 새로운 서비스를 신속하게 네트워크에 배치할 수 있는 장점이 있으며, 특히 일련의 네트워크 기능들로 구성된 서비스인 서비스 펑션 체이닝 (SFC, Service Function Chaining) 구현을 용이하게 한다. 하지만, 한편으로는 관리가 필요한 다양한 가상 자원들을 생성하기 때문에 네트워크 및 서비스 운용을 복잡하게 하는 원인이 되기도 한다. 이를 해결하기 위해 본 논문에서는 강화학습 (RL, Reinforcement Learning) 알고리즘 중 하나인 Deep Q-Networks (DQN)을 활용한 SFC 구성 방법을 제안한다. DQN 기반 SFC 구성 방법은 동적인 네트워크 환경에서 서비스의 성능 지표인 서비스 응답 시간을 만족하는 최적의 SFC를 효과적으로 구성한다.

Ⅰ. 서 론

5G 시대는 급변하는 네트워크 상황 속에서 다양한 요구사항을 갖는 서비스들을 수많은 기기에게 제공하는 것이 요구되고 있다. 이를 위해, 5G 네트워크에서는 소프트웨어 정의 네트워킹 (SDN, Software-Defined Networking)과 네트워크 기능 가상화 (NFV, Network Function Virtualization)를 기반 기술로 활용하여, 네트워크를 유연하게 구축하고 운용한다. 특히, NFV는 고가의 전용 하드웨어를 통해 제공되는 네트워크 기능을 소프트웨어 형태로 가상화하여, 일반 상용 서버에서 가상 네트워크 기능 (VNF, Virtual Network Function)으로 제공하는 기술이다. SDN과 NFV는 네트워크 운용 비용을 절감하고 새로운 서비스를 네트워크에 신속하게 배치할 수 있는 장점이 있지만, 한편으로는 관리가 필요한 많은 가상 자원들을 생성하기 때문에 네트워크 관리를 복잡하게 만든다.

복잡한 네트워크 관리 문제를 해결하기 위해 최근에는 기계학습을 네트워크 운용 및 관리에 적용하는 시도가 주목받고 있다. 특히, 지도학습 (Supervised learning)과 비지도학습 (Unsupervised learning)을 통해 트래픽 분류 및 분석, 침임 탐지 등 보안 문제에 활용한 사례가 다수 존재한다 [1]. 하지만, 5G 시대의 NFV 환경에서는 동적으로 서비스를 배치하고 운용하는 것이 필요하기 때문에, 보안 뿐 아니라 VNF 배치 (VNF Deployment), 서비스 펑션 체이닝 (SFC, Service Function Chaining), 오토 스케일링 (Auto-scaling) 등 전반적인 VNF 라이프사이클 (Life-cycle) 관리 기능에 대한 연구가 요구된다. 이들 중, SFC는 네트워크 서비스를 제공하는 방법으로, 일련의 네트워크 기능들을 트래픽에 적용하는 기술이다.

현재까지 SDN/NFV 환경에서 최적 SFC를 구성하는 다양한 방법들이 제안되었지만, 복잡한 5G 네트워크의 급변하는 상황에 대응하여 서비스 요구사항을 만족하는 SFC를 운용하기 위해서는 아직 많은 연구 이슈가 존재한다. 특히, 한정된 네트워크 자원을 고려하여 서비스 성능 요구사항을 만족하는 SFC를 효과적으로 생성할 수 있어야 한다. 따라서, 본 논문에서는 강화학습 (RL, Reinforcement Learning)을 활용하여 최적 SFC를 구성하는 방법을 제안한다. 제안하는 방법은 클라우드 컴퓨팅 환경의 제한된 가용 자원을 고려하며, 서비스 성능 목표인 서비스 시간을 만족하는 SFC를 효과적으로 생성하는 것을 목표로 한다.

Ⅱ. 관련 연구

SFC가 네트워크 서비스를 제공하기 위한 핵심 기술로 자리 잡으면서, SDN/NFV 기반 네트워크에서 기계학습을 활용한 SFC 구성 방법에 대한 많은 연구가 진행되었다. SFC를 구성하는 방법 중 하나는 SFC 경로를 선택하는 것으로, SFC에 포함되는 VNF들이 네트워크에 이미 배치된 상황을 가정한다. [2]에서는 네트워크에 배치된 SF (Service Function)의 CPU 사용률과 대역폭 (Bandwidth)을 고려하여, Temporal Difference (TD) 알고리즘으로 최적 SFC 경로를 선택하는 방법을 제안하였다. 또한, [3]에서는 지도학습을 활용하여 VNF가 배치된 서버들을 연결하는 링크 상태 (Link state)를 학습해 SFC를 통한 패킷 처리량 (Throughput)을 예측한 후, 이를 최적 SFC 경로 선정에 활용하였다.

SFC를 구성하는 다른 방법으로는 SFC 생성 요청이 발생했을 때, SFC에 포함될 VNF들을 새롭게 배치하여 SFC를 구성하는 것이다. [4]에서는 지도학습으로 SFC의 성능 목표를 만족시킬 수 있는 각 VNF 종류의 인스턴스 수를 예측한 후, 해당 개수의 VNF 인스턴스들을 생성하여 SFC를 구성한다. 하지만, 이 방법은 SFC를 구성하는 VNF들의 위치를 고려하지 않았기 때문에, 이를 개선하기 위한 [5]가 제안되었다. [5]에서는 그래프 인공 신경망 (GNN, Graph Neural Network)을 활용하여, SFC를 구성할

VNF가 설치될 서버 위치를 결정한 후, 해당 서버들에 VNF 인스턴스들을 설치하여 SFC를 구성한다.

앞서 설명한 SFC 구성 방법들 중, SFC 경로를 선택하는 방법은 새로운 VNF 인스턴스를 생성하지 않기 때문에 VNF 배치 비용 측면에서 효과적인 SFC 구성이 가능하다. 하지만, 배치된 VNF 상태에 의존하여 SFC를 구성하기 때문에 유연한 SFC 구성에는 한계가 있다. 반면, VNF를 새롭게 배치하여 SFC를 구성하는 방법은 요구사항에 따른 최적 SFC 구성이 가능하지만, 구성 비용이 크다는 단점이 있다. 또한, 네트워크 자원은 한정되어 있기 때문에 수많은 기기들이 접속하는 5G 시대에서는 주어진 자원을 효과적으로 각 서비스들에게 할당하는 것이 요구된다. 즉, SFC 구성을 위해 이미 배치된 VNF를 공유하거나 [6], 새롭게 배치하는 방법을 같이 적용할 필요가 있다.

Ⅲ. Deep Q-Networks 기반 SFC 구성 방법

본 논문에서 제안하는 DQN 기반 SFC 구성 방법은 클라우드 컴퓨팅 환경의 제한된 가용 자원을 고려하여 서비스 성능 목표를 만족하는 SFC를 구성한다. DQN은 심층 강화학습 방법 중 하나로, 인공 신경망 (Neural network)을 통해 수 많은 상태 (State)가 존재하는 최적화 문제를 설계할 수 있다. [그림 1]은 SFC 구성을 위한 DQN 모델을 나타낸다. DQN 모델은 SFC를 구성하는 VNF의 종류와 순서가 명시된 SFC 요청이 있을 때, VNF 인스턴스들을 선택 또는 생성하는 과정을 반복하여 SFC를 구성한다. 즉, 이미 배치된 VNF 인스턴스를 활용하여 성능 목표를 달성할 수 있다면, 해당 인스턴스로 SFC를 구성하기 때문에 VNF 생성 비용이 들지 않아 제한된 자원 환경 속에서 효율적으로 SFC를 생성한다. DQN 모델은 2개의 은닉층을 가지며, 활성화 함수 (Activation function)로 ReLU (Rectified Linear Unit)를 사용한다.

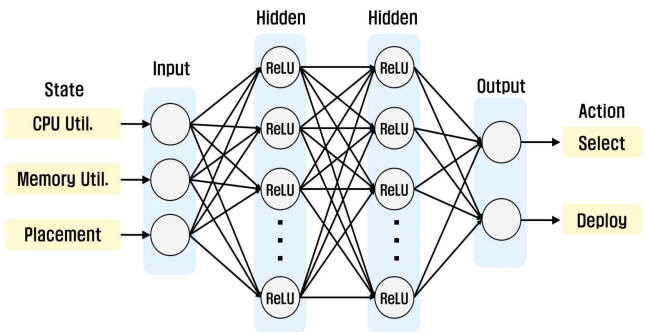


그림 1. Deep Q-networks (DQN) 기반 SFC 구성 모델

DQN 모델의 입력 계층에서는 SFC를 구성할 VNF 인스턴스들의 평균 CPU 사용률 (CPU Util.), 메모리 사용률 (Memory Util.), 배치 상태 (Placement)로 구성된 현재 상태를 입력값으로 받는다. CPU와 메모리는 VNF 인스턴스에서 패킷 처리에 사용되는 자원들이며, 배치 위치는 SFC에서 처리되는 패킷들의 경로로 인한 전파 지연 시간 (Propagation delay)에 영향을 미친다. VNF 인스턴스들의 배치 상태는 이전 순서에 선택된 VNF 인스턴스로부터 다음 순서의 VNF 인스턴스들 사이의 평균 거리를 Hop으로 계산한 값이다. 따라서 DQN 모델의 상태는 선택 대상이 되는 인스턴스들의 패킷 처리 성능과 위치를 고려하여 SFC를 구성한다.

제안하는 모델의 출력 계층에서는 SFC를 구성할 VNF를 기존에 존재하는 VNF 인스턴스들 중 선택할지 (Select), 또는 새로운 VNF 인스턴스를 생성할지 (Deploy)에 대한 행동 값 (Action value)을 출력한다. SFC 구성을 위한 VNF 인스턴스를 선택하는 행동을 결정했을 경우, CPU와 메모리 사용률이 작고, 이전 순서에 선택된 VNF 인스턴스와 가까운 인스턴스를 선택한다. 반면, VNF 인스턴스를 생성하는 행동을 결정했을 때는 이전 순서에 선택된 VNF 인스턴스와 가까운 서버들 중, 가용 자원이 충분한 위치를 선택하여 새로운 인스턴스를 생성한다. SFC를 구성할 때, 기존에 존재하는 VNF 인스턴스들 중 하나를 선택하는 것과 새로운 VNF 인스턴스를 생성하는 것은 SFC 성능과 구성 비용 측면에서 장단점이 존재하기 때문에 이를 고려한 행동 결정이 필요하다. 따라서 본 논문에서는 수식 (1)과 같은 보상 모델을 정의하였다.

$$Reward = -\ln(1 + resTime) \times Cost_{deploy} \quad (1)$$

DQN 모델은 SFC 구성을 위한 최적 정책 (Policy)을 학습하기 위해 행동을 결정할 때마다 그에 대한 보상 (Reward)을 얻는다. 제안하는 방법의 목표는 적은 자원과 비용으로 서비스 성능 요구사항을 만족하는 SFC를 구성하는 것이다. 네트워크 서비스에서 성능을 측정하기 위한 주요 지표 중 하나는 응답 시간이기 때문에 수식 (1)에서는 응답 시간 (resTime)을 고려한다. 응답 시간은 사용자가 생성한 패킷이 VNF 인스턴스에 의해 처리된 후, 그에 대한 응답을 사용자가 받기까지 소요된 시간이다. 제안하는 방법에서 DQN 모델의 행동은 다음 순서의 VNF 인스턴스를 선택하는 것이기 때문에, 보상 계산을 위한 응답 시간은 이전 순서에 선택된 VNF 인스턴스로부터 새롭게 선택 또는 생성된 VNF 인스턴스로 Ping 메시지를 전달하여 millisecond 단위로 측정한다. 또한, 인스턴스 생성에 요구되는 비용은 가중치 ($Cost_{deploy}$)를 통해 측정된 서비스 성능, 즉 응답 시간을 보정한다. 예를 들어, VNF 인스턴스를 선택했을 때와 생성했을 때 동일한 응답 시간이 측정되었을 경우, 인스턴스를 생성했을 때 받는 보상은 가중치 ($Cost_{deploy}$)로 1.2를 곱하여 감소시킨다. 이와 같은 보상 모델을 통하여 SFC 구성을 위한 서비스 성능과 구성 비용을 함께 고려할 수 있다. 마지막으로, 본 논문의 DQN 모델 학습 과정 중, 특정 상황에서 수행할 행동을 선택할 때는 ε-greedy 알고리즘을 활용한다.

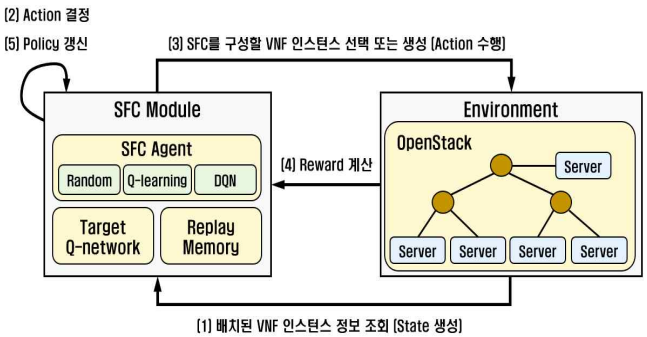


그림 2. SFC 구성 모듈

제안하는 DQN 기반 SFC 구성 방법은 향후 서술할 OpenStack 기반 실험환경에서 동작하는 모듈로 구현되었다. [그림 2]는 SFC 모듈이 동작하는 과정을 보인다. SFC 모듈은 SFC 구성 요청 메시지를 받았을 때, 해당 SFC를 구성하기 위해 환경 (Environment)에 배치된 VNF 인스턴스 정보를 조회한다. 그리고, SFC 구성을 위한 행동을 결정하여 VNF 인스턴스를 SFC에 포함시키고 행동에 대한 보상을 받는다. SFC 구성이 완료될 때까지 VNF 인스턴스 선택 또는 생성을 반복하며, 주기적으로 SFC 구성 정책을 갱신한다. 또한, SFC 모듈이 DQN 알고리즘을 사용할 때, 안정적인 학습을 위해 [7]에서 제안한 Replay memory와 Target Q-network를

활용한다. Replay memory와 Target Q-network는 강화학습 알고리즘의 학습 과정에서 순차적인 행동과 그에 대한 보상 값으로 정책을 갱신할 경우 각 행동들의 상관 관계 (Correlation)로 인해 최적 정책을 찾지 못하는 문제를 해결한다. 구현된 SFC 모듈은 GitHub을 통해 공개되어있다 [8].

Ⅳ. 성능 평가

DQN 기반 SFC 구성 방법의 성능 평가를 위해 [그림 3]과 같이 OpenStack 기반의 실험환경을 구축하였다. 실험환경은 OpenStack 환경을 관리하는 컨트롤러 노드 (Controller Node)와 VNF 인스턴스가 배치되고 SFC가 구성되는 9대의 컴퓨트 노드 (Compute Node)로 구성된다. 또한, 본 논문에서 제안하는 DQN 기반 SFC 구성 알고리즘은 AI 노드에서 실행되며, 알고리즘에 요구되는 모니터링 데이터들은 모니터링 노드 (Monitoring Node)를 통해 수집된다.

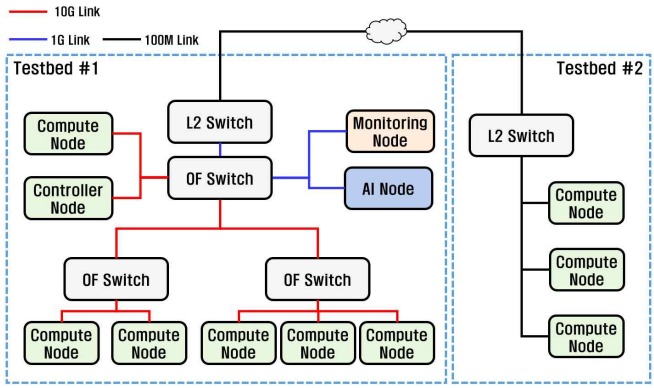


그림 3. SFC 구성 성능 평가 실험환경

성능 평가 시나리오는 [그림 4]와 같다. 성능 평가 시나리오는 미리 구축된 실험환경에서 5개의 VNF로 구성된 SFC를 생성하고, 클라이언트 가상 머신 (VM, Virtual Machine)에서 SFC를 통과하는 패킷을 생성한다. 이를 위해, 패킷의 목적지가 되는 Web VM 2개를 특정 컴퓨트 노드에 생성한 후, 임의의 컴퓨트 노드에 클라이언트 VM을 생성한다. 성능 평가는 클라이언트 VM이 생성한 패킷에 대한 서비스 시간 (Service time)을 측정하여 비교한다. 서비스 시간은 사용자로부터 생성된 패킷이 SFC를 구성하는 모든 VNF 인스턴스에서 처리되어 목적지에 도달한 후, 사용자가 다시 응답 패킷을 받기까지 소요된 시간을 의미한다. 따라서 서비스 시간은 SFC를 구성하는 VNF 인스턴스의 패킷 처리 성능과 각 VNF 인스턴스 사이에서 패킷이 전달되는 시간을 포함한다.

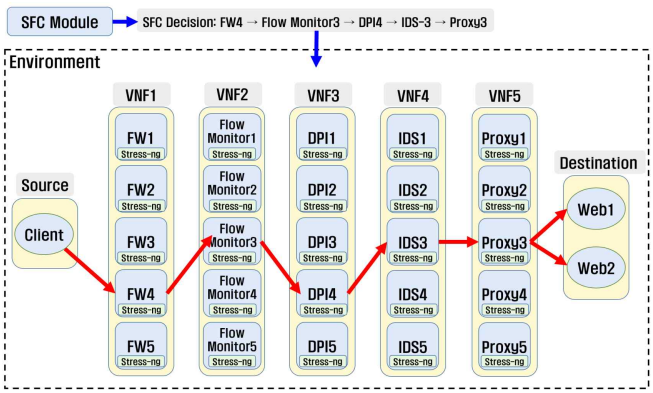


그림 4. SFC 구성 성능 평가 시나리오

SFC는 Firewall, Flow monitor, Deep Packet Inspection (DPI), Intrusion Detection System (IDS), Proxy 기능을 가진 VM들로 구성된다. 각 VNF 기능은 오픈소스 소프트웨어인 Iptables [9], ntopng [10], nDPI [11], Suricata [12], HAProxy [13]를 설치해 구현하였다. VNF 종류마다 5개의 VM들이 실험환경 내 임의의 위치에 미리 배치되어 있으며, 각 VM들은 Stress-ng [14]를 사용하여 동적으로 변하는 임의의 자원 사용률을 가지도록 설정하였다.

표 1. Q-Learning 및 DQN 초기 변수 정보

변수 종류	값
[DQN/Q-learning] η (학습률)	0.01
[DQN/Q-learning] γ (할인율)	0.98
[DQN/Q-learning] ϵ (탐험 확률)	0.10
[DQN] Sampling 크기 (Mini-batch)	16

성능 평가 실험에서는 3개의 서로 다른 SFC 구성 방법을 사용하여 SFC를 생성한다. 첫 번째 방법은 기존에 존재하는 VM들, 즉 VNF 인스턴스들을 임의로 선택하여 SFC를 생성한다 (Random 방법). 두 번째 방법은 본 저자들의 선행 연구 [15]에서 제안한 Q-learning 기반의 SFC 구성 방법 (Q-learning 방법)으로, 미리 배치된 VNF 인스턴스들의 자원 사용량과 배치 위치를 Q-learning으로 학습하여 SFC를 구성한다. 세 번째 방법은 본 논문에서 제안하는 DQN 모델을 기반으로 VNF 인스턴스들을 선택 또는 생성하여 SFC를 구성한다 (DQN 방법). 이 때, Q-learning과 DQN은 학습을 위한 매개변수의 할당이 필요하다. 따라서, 성능 평가 실험에서 Q-learning과 DQN 방법은 (표 1)의 값들을 사용한다.

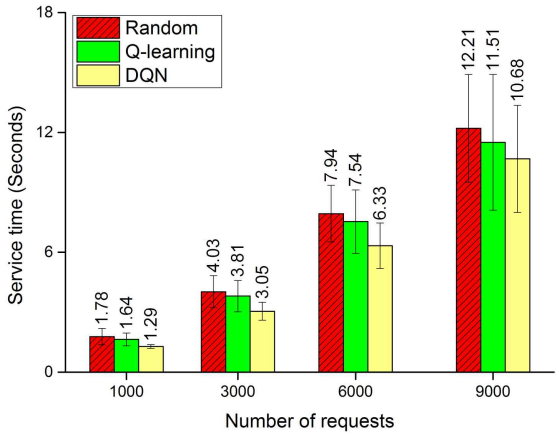


그림 5. SFC를 통한 서비스 시간 측정 결과

모든 설정을 마친 후, Q-learning과 DQN은 3000번 이상 행동을 수행하여 학습한 모델을 통해 성능 평가를 수행하였다. 또한, SFC 구성 방법들을 통해 각 SFC를 구성할 때마다, 클라이언트 VM에서 1,000개, 3,000개, 6,000개, 그리고 9,000개의 HTTP 요청 메시지를 발생시킨 후, 이에 대한 모든 응답을 받기까지 소요된 서비스 시간을 측정하였다. [그림 5]는 각 SFC 구성 방법들을 통해 SFC 구성을 100번 반복하여 계산한 평균 서비스 시간을 보인다. 전체적으로 HTTP 요청 메시지 개수가 늘어날수록 SFC에서 측정되는 서비스 시간도 증가하였으나, DQN 방법으로 SFC를 구성했을 때 가장 짧은 평균 서비스 시간이 측정되었다. 또한, 응답 시간의 표준편차도 DQN으로 SFC를 구성했을 때 가장 작았으며, 이를 통해 본 논문에서 제안하는 방법은 짧고 안정적인 서비스 시간을 제공하는

SFC를 구성할 수 있음을 확인하였다.

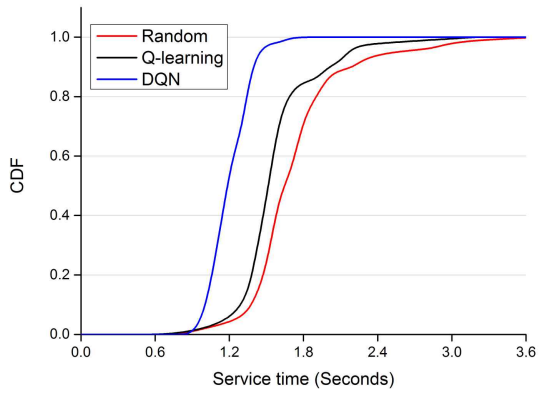


그림 6. SFC 서비스 시간 누적 분포 함수 결과

[그림 6]은 서로 다른 SFC 구성 방법을 100번씩 반복해 생성한 각 SFC에서 HTTP 요청 메시지 1,000개를 발생시켜 측정한 서비스 시간을 누적 분포 함수 (CDF, Cumulative Distribution Function)로 나타낸 결과이다. 이를 통해, DQN 기반 SFC 구성 방법은 95% 이상으로 SFC에서 1.4초 이내의 서비스 시간을 제공하는 것을 확인하였다. 하지만, Random 방법에서는 20%의 SFC와 Q-learning에서는 43%의 SFC만이 1.4초 내의 서비스 시간을 제공하였다. 또한, 제안하는 DQN 기반 SFC 구성 방법에서 현재 배치된 VNF 인스턴스들의 자원 사용률과 배치 상태로 인해 짧은 서비스 시간 보장이 힘들다고 판단할 경우, VNF 인스턴스 1~2개를 새롭게 배치하여 SFC를 구성하였다. 따라서, 제안하는 방식은 최소한의 VNF 인스턴스 추가 비용으로 짧은 서비스 시간을 갖는 SFC를 효과적으로 구성할 수 있다는 것을 확인하였다.

V. 결론 및 향후 연구

본 논문에서는 심층 강화학습 알고리즘인 DQN을 활용하여 클라우드 컴퓨팅 환경에서의 최적 SFC 구성 방법을 제안하였다. 제안하는 방법은 SFC 구성 요청사항을 바탕으로 SFC의 성능 목표인 서비스 시간을 만족시킴과 동시에, 구성 비용을 최소화하는 SFC를 구성한다. 성능 평가 결과, DQN 기반 SFC 구성 방법은 현재 배치된 VNF 인스턴스들의 자원 사용률과 위치를 고려하여 짧은 서비스 시간을 보장하는 SFC를 구성할 수 있었다. 하지만, 제안하는 방법에서 DQN은 VNF 인스턴스와 관련된 데이터만 사용해 학습을 수행하기 때문에, 네트워크 링크 상태, 혼잡 상황 등 네트워크 환경에 대응한 SFC 구성에는 한계가 존재한다.

따라서, 향후 연구로는 동적으로 변하는 네트워크 환경을 고려한 SFC 구성 모델을 설계할 예정이다. 또한, 본 논문의 DQN 모델에서는 간단한 인공 신경망으로 구성되었지만, 향후 개선될 모델에서는 순환 신경망 (RNN, Recurrent Neural Network)을 활용하여, VNF 인스턴스들의 패킷 처리 성능의 추세를 SFC 구성에 고려할 것이다. 마지막으로 다양한 시나리오에서 SFC 구성 방법에 대한 성능 평가를 수행할 계획이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)과 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2021-2017-0-01633).

참 고 문 헌

[1] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications, 9(1), 1-99.

[2] Ku, Hye-Jin, J. H. Jung, and Gu-In Kwon. "A study on reinforcement learning based SFC path selection in SDN/NFV." International Journal of Applied Engineering Research 12, no. 12 (2017): 3439-3443.

[3] Jeong, Seyeon, Heegon Kim, Jae-Hyoung Yoo, and James Won-Ki Hong. "Machine Learning based Link State Aware Service Function Chaining." In 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1-4. IEEE, 2019.

[4] Lange, S., Kim, H. G., Jeong, S. Y., Choi, H., Yoo, J. H., & Hong, J. W. K. (2019, October). Predicting vnf deployment decisions under dynamically changing network conditions. In 2019 15th International Conference on Network and Service Management (CNSM) (pp. 1-9). IEEE.

[5] Kim, H. G., Park, S., Heo, D., Lange, S., Choi, H., Yoo, J. H., & Hong, J. W. K. (2020, November). Graph Neural Network-based Virtual Network Function Deployment Prediction. In 2020 16th International Conference on Network and Service Management (CNSM) (pp. 1-7). IEEE.

[6] Mohamad, A., & Hassanein, H. S. (2020, November). PSVShare: A Priority-based SFC placement with VNF Sharing. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (pp. 25-30). IEEE.

[7] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[8] DPNM, Network Intelligent Project. [Online]. Available: https://github.com/dpnm-ni/ni-sfc-path-module-public

[9] Purdy, Gregor N. Linux iptables Pocket Reference: Firewalls, NAT & Accounting. " O'Reilly Media, Inc.", 2004.

[10] Deri, Luca, Maurizio Martinelli, and Alfredo Cardigliano. "Realtime high-speed network traffic monitoring using ntopng." 28th large installation system administration conference (LISA14). 2014.

[11] Deri, Luca, et al. "ndpi: Open-source high-speed deep packet inspection." 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, 2014.

[12] Open Information Security Foundation, "Suricata: Open Source IDS/IPS/NSM engine", [Online]. Available: https://suricata-ids.org/

[13] HAProxy, "HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer", [Online]. Available: http://www.haproxy.org/

[14] King, Colin Ian. "Stress-ng." URL: http://kernel. ubuntu. com/git/cking/stressng. git/(visited on 28/03/2018) (2017)

[15] Doyoung Lee, Jae-Hyoung Yoo, James Won-Ki Hong, "Q-learning Based Service Function Chaining Using VNF Resource-aware Reward Model", 21st Asia-Pacific Network Operations and Management Symposium (APNOMS 2020), Daegu, Korea, Sep. 23-25, 2020.

로그 및 자원 분석을 통한 VNF 고장 예측에 관한 연구

남석현, 홍지범, 유재형, 홍원기
포항공과대학교 컴퓨터공학과

{obiwan96, hosewq, jhyoo78, jwkhong}@postech.ac.kr

A Study on VNF Failure Prediction through Log and Resource Analysis

Sukhyun Nam, Jibum Hong, Jae-Hyoung Yoo, James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH

요 약

본 논문은 SDN/NFV 환경에서 VNF 및 서버의 고장을 예측하기 위한 기계 학습 모델 및 서비스를 제안한다. 제안하는 모델은 SDN/NFV 환경에서 각 VNF 로부터 발생하는 시스템 로그와 collectd 를 통해 수집되는 자원 사용량 및 트래픽 정보를 함께 활용하는 병렬 CNN 을 사용한다. 병렬 CNN 의 각 채널은 자원 사용량 데이터로부터 feature 를 추출하는 RNN 모델과 로그 데이터로부터 feature 를 추출하는 합성곱층으로 이루어진다. 이 때, 로그 데이터는 사전 처리를 통해 임베딩 행렬로 변환시키며, 추출된 feature 는 softmax 층의 입력으로 사용된다. 제안하는 고장 예측 모델은 병렬 CNN 을 통해 VNF 및 서버에 고장이 발생하기 전에 발생하는 장애 관련 징후로 고장을 예측하고 완화 작용 및 migration 을 통해 서비스 고장을 사전에 방지하는 기능을 제공한다.

Ⅰ. 서 론

오늘날의 네트워크는 그 규모와 구조가 점점 더 커지고 복잡해지고 있다. 소프트웨어 정의 네트워킹 (Software-Defined Networking, SDN)과 네트워크 기능 가상화 (Network Function Virtualization, NFV) 기술 등이 등장하면서 CAPEX/OPEX 를 절감시켰지만, 복잡해진 네트워크 구조로 인해 네트워크의 장애를 진단하고 관리하는 것이 더 어려워지고 있다. 이러한 이유로 SDN/NFV 환경에서 가상 네트워크 기능 (Virtualized Network Function, VNF)의 이상 탐지를 위한 연구가 진행되고 있으나, 서버와 가상 서버 및 VNF 의 고장을 사전에 예측하여 고장 발생 전에 조치를 취하는 기술에 대한 연구는 부족한 상황이다.

대부분의 서버 및 네트워크 장비들은 실시간 로그 (예: syslog) 출력 기능을 제공한다. 최근 서버 및 네트워크의 소프트웨어 구조가 복잡해짐에 따라 로그의 양도 비례하여 늘어나고 있다. 로그는 장비의 상태를 가장 잘 나타내는 데이터 중 하나이며, 장애 발생 시 관련 로그가 발생하기 때문에 네트워크 관리에 로그를 활용하는 연구들이 일부 진행되고 있다 [1, 2, 4]. 하지만 대부분의 로그 데이터는 체계적으로 생성되지 않기 때문에 로그를 이해할 수 있는 전문가가 수작업으로 관리해야 한다는 문제가 있다. 로그 데이터를 자동으로 처리하는 것은 여전히 상당히 어려운 과제로 남아있다.

로그 분석에는 자연 언어 처리 (Natural Language Processing, NLP)의 한 분야인 문장 분류 (sentence classification) 기법을 적용시킬 수 있다. 문장 분류는 영화 리뷰와 같은 텍스트 문서로부터 저자가 해당 문서의 주제에 대해 표현한 의견을 판단하거나, 스팸 메일 분류와 같이 미리 정의된 기준에 따라 문서를 분류하는 분야이다. 단어의 순서가 중요하여 순환 신경망 (Recurrent Neural Network, RNN)이 강세를 보이는 다른 NLP 분야와 달리 각 단어가 분류 결과에 얼마나 영향을 미치는지를 분석하기 위하여 합성곱 신경망 (Convolution Neural Network, CNN)을 주로 사용하는 분야이다 [3].

본 논문에서는 SDN/NFV 환경에서 각 VNF 들의 로그와 시스템 자원 사용량 데이터를 모두 입력 값으로 받는 병렬 CNN 모델을 이용한 서버 고장 예측 모델을 제안한다. 제안한 모델은 입력 데이터를 기준으로 일정 시간 뒤에 VNF 나 물리 서버에서 고장이 발생할 것을 예측한다.

Ⅱ. 관련 연구

선행연구 [3]은 문장 분류 문제에서 CNN 모델을 처음으로 제안하였다. 해당 연구는 문장을 CNN 의 입력 피처로 사용하기 위해 단어를 고밀도 벡터 (dense vector)로 표현하는 기법인 워드 임베딩 (word embedding) 기법 [6]을 사용하였으며, 생성된 임베딩 벡터들로 concatenation 과정을 거쳐 문장 벡터를 생성하였다. 생성된 문장 벡터는 합성곱층 (convolution layer)을 거치는데, 이 때 합성곱층에서 쓰이는 filter 의 크기는 임베딩 벡터의 차원인 h 와 필터의 크기 k 의 곱인 hk 차원의 벡터를 사용하였다. 생성된 feature 에 대해 max pooling 을 적용하여 filter 의 수만큼 feature 값이 생성되고 이에 대해 softmax layer 을 통해 최종 분류를 하였다. 실험 결과 제안한 CNN 모델의 성능이 다른 기계학습 기법에 비해 매우 뛰어나지는 않았지만, 가장 간단한 형태의 CNN 을 활용해 만든 모델임에도 불구하고 다른 모델들과 비교하여 우수한 성능을 보여 문장 분류 문제에 CNN 이 적합함을 보였다.

선행연구 [4]는 시뮬레이션 환경의 무선 통신 시스템에서 로그 데이터를 수집하여 일정한 격차 (gap) 이후의 로그에 에러 메시지가 포함되는지를 예측하는 CNN 모델을 제안하였다. 해당 모델은 로그 데이터에는 숫자와 구두점 등 쓸모 없는 글자가 많기 때문에 이를 제거하고, 어휘 사전에서 빈도수가 적은 단어들을 제거하는 전처리과정을 거친 후 사용하였다. 전처리 후의 로그는 워드 임베딩을 통해 임베딩 벡터로 변환하여 사용하였다. 생성한 임베딩 벡터에 대해 single channel 의 CNN 을 학습시킨 결과 예측 격차가 2000 일 때 정확도 0.7 을 웃도는 성능을 보였으나 예측 시점과의 격차의 기준을 로그 개수로 하여 예측 시점이 일정하지 않다는 한계점이 존재한다.

여러 행렬을 입력으로 받아 각각 합성곱층을 거친 후에 concatenation 하여 사용하는 형태의 CNN 을 multi-channel CNN 이라고 한다. 문장 분석 연구에서 여러 입력 값을 함께 사용하기 위해 multi-channel CNN 을 사용하기도 한다 [3, 5]. 선행연구 [3]는 두 가지 워드 임베딩을 사용하기 위해 2-channel CNN 을 학습시켰다. 선행연구 [5]는 한국어 감성분석에서 형태소 기반의 CNN 을 발전시켜 음절, 자소기반으로 생성한 워드 임베딩을 사용하기 위해 3-channel CNN 을 학습하였다. Multi-channel 을 통해 생성된 다중 feature 들은 softmax layer 에서 분류에 도움이 되는 정보만 남도록 학습되기 때문에 분류에 도움이 되는 다양한 입력 피처를 사용하면 더 높은 성능을 낼 수 있다.

Ⅲ. 병렬 CNN 기반 고장 예측 서비스

본 연구에서는 로그 데이터와 자원 사용량 데이터를 함께 사용하는 병렬 CNN 을 이용하여 물리 서버의 고장을 예측하고 경고하는 서비스를 제안한다. 제안하는 서비스는 예측 시기를 일정하게 하기 위해 시간을 기준으로 예측 시점과의 격차를 정한다. 제안하는 병렬 CNN 기반 VNF 고장 예측 서비스 구조는 그림 1 과 같다.

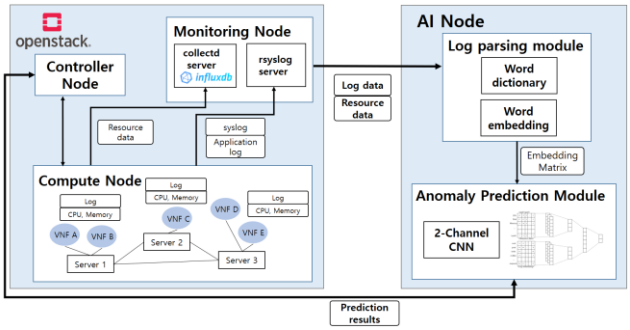


그림 1 병렬 CNN 기반 고장 예측 서비스 구조

제안하는 서비스는 클라우드 컴퓨팅 관리 시스템인 OpenStack 을 이용하여 구축한 NFV Infrastructure (NFVI)를 사용한다. 해당 NFVI 환경은 인공지능 기반의 NFV 관리 플랫폼 선행 연구 [7]를 기반으로 이루어지며, 네트워크 서비스를 제공하기 위해 다양한 VNF (예: IDS, 방화벽) 들을 동작시킨다. 각 VNF 내부에서는 모니터링 데몬인 collectd [8]를 통해 CPU, 메모리 사용량과 같은 자원 사용량 데이터와 네트워크 트래픽 로드와 같은 네트워크 데이터를 실시간으로 수집하여 모니터링 노드로 전송한다. 로그 데이터는 로그 수집 데몬인 Rsyslog [9]를 통해 수집되어 모니터링 노드로 전송된다.

모니터링 노드는 수집된 데이터에서 timestamp 를 이용하여 정해진 window 크기만큼의 데이터를 추출하여 AI 노드로 전송한다.

AI 노드는 사전에 학습된 어휘 사전, 워드 임베딩, CNN 모델을 포함하는 노드이다. 그림 1 의 구조에서 추출된 로그 및 자원 사용량 데이터를 이용하여 진행된다. 해당 데이터는 window 크기만큼의 로그 및 자원 사용량 데이터를 입력 값으로, 일정 시간 뒤에 VNF 및 서버가 정상 작동하는지 여부를 출력 값으로 갖는 데이터이다. 어휘 사전 및 워드 임베딩은 추출된 전체 로그 데이터를 이용하여 생성된다. 어휘 사전은 로그 데이터 전체의 단어들의 빈도수가 계산된 데이터이다. 워드 임베딩은 Google 의 오픈 소스 프로젝트인 word2vec [10]을 이용하여 로그 데이터로 생성시킨다. 공개된 워드 임베딩이 아닌, 로그 데이터를 이용하여 생성한 워드 임베딩을 사용함으로서 로그 데이터 분석에 더 적합한 워드 임베딩을 사용할 수 있다.

AI 노드는 모니터링 노드로부터 전달 받은 로그데이터에 대해 전처리 작업을 거친 후 CNN 모델에 입력시킨다. 전처리 과정은 숫자와 구두점은 제거하고 단어들만 남긴 후 어휘 사전을 통해 로그에서 빈도수가 낮은 단어들은 Unknown 으로 태깅하는 과정이다. Unknown 단어들은 Out of Vocabulary (OOV) 벡터로 치환되는데, 이 때 OOV 벡터는 어휘 사전의 워드

임베딩 벡터들과 가장 멀리 있는 벡터로 사전에 생성된 벡터이다. 나머지 단어들은 사전에 학습된 워드 임베딩 벡터들로 치환된다. 그렇게 생성된 임베딩 데이터와 자원 사용량 데이터를 병렬 CNN 의 입력 값으로 사용한다. 사용하는 병렬 CNN 모델은 그림 2 와 같다.

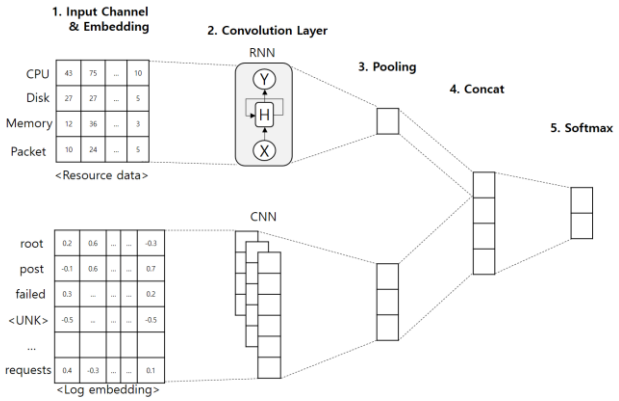


그림 2 병렬 CNN 구조

제안하는 CNN 모델은 자원 사용량 데이터와 로그 데이터를 모두 입력으로 받는 구조로서, multi-channel CNN 의 일종으로 볼 수 있으나, 한 채널은 RNN 을 사용하기 때문에 병렬 CNN 으로 표기하였다. 로그 데이터는 [3]에서 제안한 CNN 모델과 마찬가지로 합성곱층을 거쳐 feature 벡터가 추출된다. 자원 사용량 데이터는 기존의 연구들과 다르게 convolution layer 로서 RNN 을 활용하는데, 이는 자원 사용량 데이터는 로그 데이터와 달리 시간에 따른 값의 변화를 분석해야 하는 시계열 데이터이기 때문이다. 자원 사용량 데이터에서도 마찬가지로 RNN 을 통하여 feature 가 추출되면 두 channel 에서 추출된 feature 를 concatenation 을 통해 feature 벡터를 생성할 수 있다. 최종적으로 softmax layer 를 통해 고장인지 아닌지 판별할 수 있게 된다. 네트워크 장비 및 VNF 는 고장이 일어나기 전에 장애 관련 로그를 발생시킬 것이며, 이를 로그 및 자원 사용량 데이터 분석을 통해 예측할 수 있을 것이다. 본 모델을 통해 제안하는 모델은 일정 시간 이후에 서버에 고장이 생기는지 여부를 예측할 수 있을 것으로 기대된다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 SDN/NFV 환경 관리를 위한 병렬 CNN 기반 고장 예측 모델 및 서비스를 제안하였다. 제안하는 모델은 NFV 환경에서 추출되는 로그 데이터와 자원 사용량 데이터를 기반으로 일정한 시간 격차 뒤의 물리 서버 및 VNF 의 고장 발생 여부를 학습한다. 학습된 모델은 NFV 환경에서 AI 노드에 포함되어 고장이 일어날 것을 미리 예측할 수 있으며, 고장이 예측될 경우 controller node 에 경고한다. 이를 통해 추후 controller 노드에서는 migration 및 auto-scaling 에 활용 가능하다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)과 2021 년도 정부(산업통상자원부)의 재원으로 산업기술평가관리원의 지원을 받아 수행된 연구임 (No.2009633, 초저지연 네트워크 서비스를 위한 SDN 기반 인공지능 관제 시스템 개발).

참 고 문 헌

[1] Q. Fu, J. Lou, Y. Wang and J. Li, "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis," 2009 Ninth IEEE International Conference on Data Mining, 2009, pp. 149-158.

[2] S. He, J. Zhu, P. He and M. R. Lyu, "Experience Report: System Log Analysis for Anomaly Detection," 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, Canada, 2016, pp. 207-218.

[3] Y. Kim, "Convolutional neural networks for sentence classification," In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746-1751.

[4] W. Ji, S. Duan, R. Chen, S. Wang and Q. Ling, "A CNN-based network failure prediction method with logs," 2018 Chinese Control And Decision Conference (CCDC), 2018, pp. 4087-4090.

[5] 김민, 변증현, 이충희, 이연수, “Multi-channel CNN 을 이용한 한국어 감성분석,” 제 30 회 한글 및 한국어 정보처리 학술대회 논문집, 2018, pp. 79-83.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", In Advances on Neural information Processing Systems, 2013.

[7] 정세연, 이도영, 유재형, 홍원기, “인공지능 기반 NFV 관리 플랫폼,” In KNOM Conference 2019, pp. 40-42, May 2019.

[8] “collectd – The system statistics collection daemon,” Available : https://collectd.org/

[9] Adiscon GmbH, “The rocket-fast Syslog Server,” [Online]. Available: https://www.rsyslog.com/

[10] Google, “word2vec,” 2013. [Online]. Available: https://code.google.com/archive/p/word2vec/ .

그리드 컴퓨팅 시스템의 보안 위험성 탐구

황선홍, 엄익채*
전남대학교 정보보안협동과정(대학원생), *전남대학교 시스템보안연구센터(교수)
f.killrra@gmail.com, *iceuom@chonnam.ac.kr

Study on Security Risk of Grid Computing System

Hwang Sun Hong, Euom Ieck Chae*
Interdisciplinary Program of Information Security, Chonnam National University (Master's student)
*System Security Research Center, Chonnam National University (Professor)

요 약

그리드 컴퓨팅은 분산 병렬 컴퓨팅의 일종으로 네트워크에 연결된 여러 사용자의 PC를 하나의 슈퍼컴퓨터처럼 사용할 수 있는 기술을 의미한다. 이러한 그리드 컴퓨팅 기술은 P2P 기반 서비스를 제공하는 플랫폼들에 의해 주로 활용되고 있으며 본 연구에서는 그리드 컴퓨팅이 활용되는 서비스 중 대표적으로 실시간 스트리밍 서비스에서 사용되는 그리드 딜리버리 소프트웨어의 네트워크 통신을 분석하여 보안 위험성을 탐구하였다. 결과적으로 본 연구를 통해 화면 변조, 데이터 탈취, 패킷 변조를 통한 코드 실행 등의 세 가지 시나리오를 도출하였고, 공격 가능성에 대해 증명하였다.

Ⅰ. 서 론

그리드 컴퓨팅[1][2]은 분산 병렬 컴퓨팅의 일종으로 네트워크에 연결된 여러 사용자 PC의 자원을 활용하여 하나의 슈퍼컴퓨터처럼 사용할 수 있는 기술이다.[3] 이러한 그리드 컴퓨팅 기술은 주로 P2P 기반의 서비스를 제공하는 플랫폼들에 의해 활용되고 있다.

최근 COVID-19의 영향으로 수업, 행사를 위해 많은 단체에서 실시간 스트리밍 플랫폼을 이용하고 있으며[4] 실시간 스트리밍 플랫폼 또한 그리드 컴퓨팅 기술을 이용하여 사용자에게 고화질 영상 송출 서비스를 제공하고 있다. 사용자들 간의 내부 리소스를 공유하는 방식인 그리드 컴퓨팅 기술은 같은 채널을 시청하고 있는 사용자들 간에 데이터를 주고받기 때문에 보안에 철저한 관리가 필요하다. 하지만 이러한 그리드 컴퓨팅 시스템의 보안성을 지적하는 연구는 많이 이뤄지고 있지 않다. 따라서 본 논문에서는 이러한 그리드 컴퓨팅 기술을 사용하는 서비스 중 실시간 스트리밍 플랫폼을 대상으로 발생될 수 있는 보안 위험성에 대해 다루고자 한다.

본 논문의 본론 1장에서는 그리드 컴퓨팅의 구조에 대해 설명하고, 2장에서는 그리드 컴퓨팅 기술과 실시간 스트리밍 서비스의 관계에 대해 서술하며 3장에서는 분석과정에서 도출된 Attack Surface에 대해 설명한다.

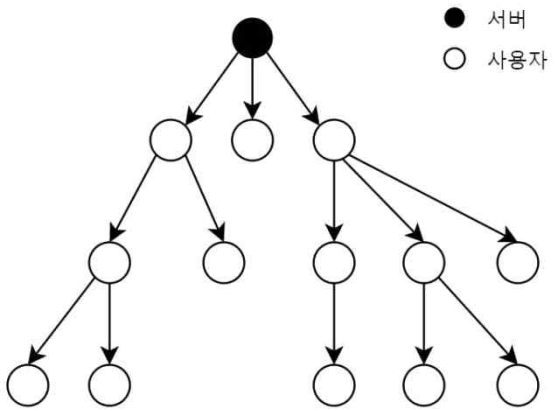
Ⅱ. 본론

1. 그리드 컴퓨팅 구조

본 장에서는 그리드 컴퓨팅을 크게 두 가지로 분류한다. 특히 국내 실시간 스트리밍에서의 그리드 컴퓨팅은 트리, 메쉬 구조를 이루고 있다.

1.1. 트리 구조

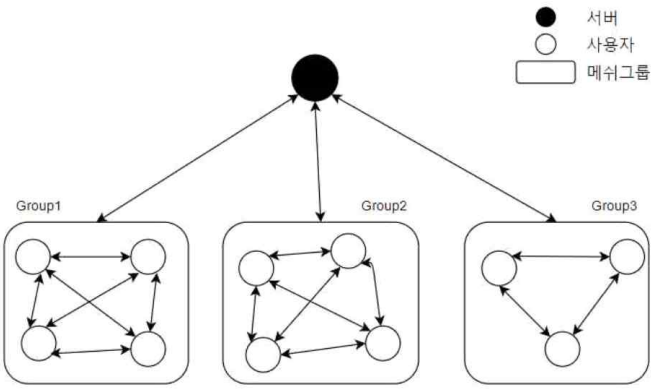
트리 구조의 그리드 컴퓨팅 방식[5]은 사용자가 부모 노드로부터 데이터를 수신한 뒤 자식 노드에게 전달하는 방식이다. [그림 1]과 같이 단방향으로 부모 노드에서 자식 노드로 데이터가 전달된다. 이때, 부모 노드의 사용자가 하위 자식 노드들에게 데이터를 변조하여 전송하게 될 경우 부모 노드에게 연결되어있는 모든 자식 노드들은 변조된 데이터를 수신하게 되며 흐름 제어에 용이하여 공격에 효율적으로 작용될 수 있다.



[그림 1] 트리 구조의 그리드 컴퓨팅 방식

1.2. 메쉬 구조

메쉬 구조의 그리드 컴퓨팅 방식[6][7]은 같은 그룹에 연결된 서로 다른 사용자들 간에 영상 데이터가 송수신되는 방식이다. [그림 2]와 같이 단방향으로 부모 노드가 데이터 전송을 하는 트리 구조 그리드 컴퓨팅 방식과 구별된다. 이렇게 하나의 노드가 아닌 동일 그룹의 사용자들 간 데이터를 송수신할 경우 트리 구조의 그리드 컴퓨팅 방식에서의 위험성을 축소시킨다는 장점이 있다[8]. 하지만 여전히 데이터 변조를 통해 같은 그룹 내의 사용자들에 대한 공격 가능성이 존재한다.



[그림 2] 메쉬 구조의 그리드 컴퓨팅 방식

2. 실시간 스트리밍 서비스와 그리드 컴퓨팅

그리드 컴퓨팅은 P2P 기반 서비스를 제공하는 플랫폼들에 의해 활용되고 있으며 본 논문에서는 이러한 그리드 컴퓨팅 기술을 사용하는 국내 실시간 스트리밍 플랫폼을 대상으로 위험성을 분석하였으며 아프리카TV, 카카오TV, 네이버TV 등에서 고화질 스트리밍 서비스 제공을 위해 그리드 컴퓨팅 기술을 사용하는 것을 확인하였다.

실시간 스트리밍 서비스 제공자는 서버에서 데이터를 송신할 송신자를 선정하고 데이터를 전송한다. 선정된 송신자들은 동일한 채널을 시청하고 있는 다른 사용자들에게 데이터를 전송한다. 이는 그리드 컴퓨팅을 위한 소프트웨어를 통해 진행되는데, 본 논문에서는 이를 그리드 딜리버리라고 칭한다. 이 그리드 딜리버리는 다른 사용자 혹은 서버로부터 받은 데이터를 송수신한 후 이를 브라우저 및 응용 프로그램으로 전달하여 사용자의 화면에 송출하는 방식으로 서비스를 제공한다.

3. 그리드 컴퓨팅의 위험성과 공격 시나리오

3.1. 그리드 컴퓨팅의 위험성

실시간 스트리밍 서비스는 특성상 실시간성을 갖으며 따라서 통신 속도가 중요하다. 국내 실시간 스트리밍 플랫폼의 그리드 딜리버리를 분석한 결과 빠른 통신 속도를 위해 인증, 암복화 과정을 생략하고 최소한의 데이터 처리만 하는 것이 확인되었다. 이로 인해 공격자는 임의로 영상 데이터 혹은 데이터 프로토콜 헤더를 변조하여 수신자에게 전송할 수 있게 되며 이로 인해 보안 취약점을 야기한다. 또한 성인 채널, 비밀 채널 등의 영상 정보를 인가되지 않은 비인가자에 의해 탈취될 수 있는 요소가 존재한다. 또한 실시간 스트리밍 서비스에 사용되는 그리드 딜리버리는 영상 데이터를 송신 및 수신하는 과정에서 사용되게 된다. 따라서 공격자가 변조된 데이터를 하위 노드 혹은 같은 그룹 내의 노드들에게 전송하게 될 경우 그리드 딜리버리 내부에서 메모리 오염 등의 취약점으로 이어질 가능성이 존재한다.

3.2. 공격 시나리오

본 장에서는 그리드 컴퓨팅 기술을 사용하는 국내 실시간 스트리밍 서비스 플랫폼에서 예상할 수 있는 공격 시나리오인 화면 변조, 데이터 탈취, 패킷 변조를 통한 코드 실행에 대해 설명한다.

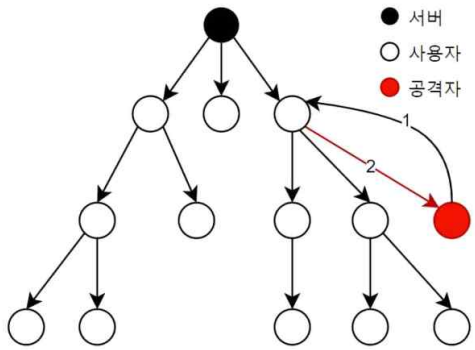
사용자 간 송수신되는 패킷에 암호화가 적용되어 있지 않다면 송신자가 수신자에게 데이터를 전송할 때 해당 데이터를 변조할 수 있는 가능성이 있다. 따라서 공격자가 이를 악용한다면 수신자에게 변조된 화면을 송출하여 수신자가 원치 않는 영상을 강제로 송출할 수 있다. 해당 공격 시나리오는 그리드 컴퓨팅의 트리 구조 사용 시 상위 노드에 공격자가 위치해 있을 때 가능하며 공격 성공 시 하위 노드들의 화면이 [그림 3]과 같이 변조되어 파급력이 크다.



[그림 3] 변조된 사용자 화면

또한 실시간 스트리밍 플랫폼에는 인증된 일부 사용자에게만 시청 권한을 부여하는 형태의 서비스가 존재한다. 이러한 경우 [그림 4]와 같이 접근 권한을 부여하지 않은 공격자가 방송을 시청하고 있는 송신자에게 강제로 소켓 연결을 요청할 수 있으며 취약한 인증으로 인해 권한 없이 영상 데이터를 수신할 수 있다.

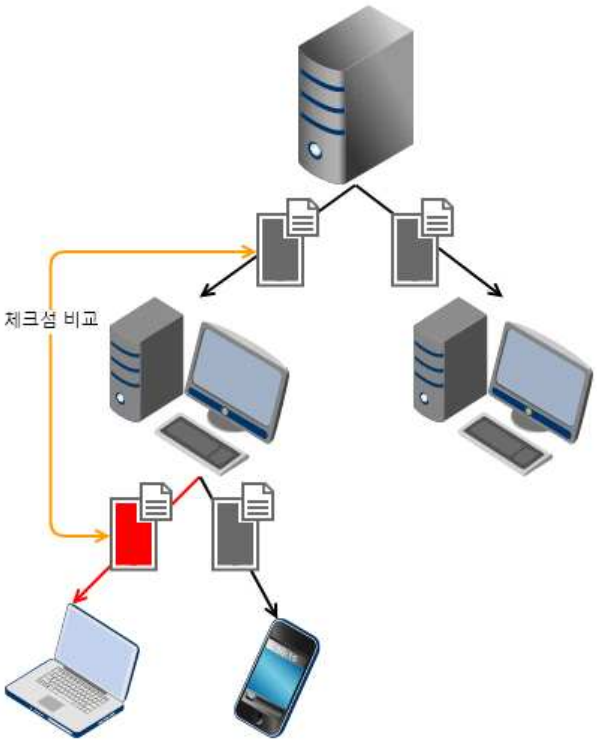
마지막으로 수신자의 그리드 딜리버리는 .exe형태의 실행파일로서 전송된 데이터 패킷을 입력값으로 받게 된다. 이때 공격자가 특정 로직 버그나 메모리 오염을 야기하도록 데이터를 변조하여 전송하게 될 시 원격에서 사용자의 PC에 임의 코드를 실행할 수 있다.



[그림 4] 취약한 인증으로 인한 데이터 탈취 시나리오

4. 보안성 강화 방안

본 장에서는 도출된 공격 시나리오를 토대로 그리드 컴퓨팅 시스템의 보안성 강화를 위한 대책을 제안한다. 앞서 제안된 공격 시나리오는 사용자 간의 인증과 데이터의 무결성 검증의 부재로 인해 발생한다. 따라서 그리드 컴퓨팅 기술을 사용할 경우 데이터를 송신하는 송신자에 대한 인증과정과 해당 송신자가 송신하는 데이터에 대한 암호화가 필요하다. 하지만 실시간 스트리밍 서비스의 경우 실시간성과 빠른 데이터 전송이 요구되기 때문에 데이터 암호화 과정에 어려움이 있다. 따라서 [그림 5]와 같이 체크섬을 통해 서버에서 전송되는 데이터와 송신자 노드가 하위 노드에게 전송하는 데이터를 확인하는 절차가 필요하다.



[그림 5] 체크섬 비교를 통한 데이터 무결성 검증

Ⅲ. 결론

본 논문에서는 그리드 컴퓨팅 기술의 보안성 검증을 위해 국내에서 쉽게 접할 수 있는 실시간 스트리밍 플랫폼의 그리드 딜리버리 소프트웨어를 분석하였고 결과적으로 화면 변조, 데이터 탈취, 패킷 변조로 인한 임의 코드 실행 등 세 가지 위험성에 대해 제시한다. 이러한 위험성은 일반 사용자들의 개인정보 유출과 경제적 피해로 이어질 수 있으며 하나의 사용자만이 공격의 대상이 되는 것이 아닌 네트워크 웜처럼 동작할 수 있음을 제시하였다. 따라서 이러한 그리드 컴퓨팅 시스템을 사용할 경우 사용자 간의 인증과 데이터에 대한 무결성 검증과정이 필요하다.

Ⅳ. 향후 연구

본 논문에서 제시한 시나리오를 토대로 그리드 컴퓨팅 시스템에서 발생할 수 있는 악용 시나리오에 대한 연구를 진행하여 발생할 수 있는 위험에 있어 선제적으로 방어하는 연구를 진행할 계획이다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(NO, 2019-0-01343, 융합보안핵심인재양성)

참 고 문 헌

[1] CZAJKOWSKI, Karl, et al. Grid information services for distributed resource sharing. In: Proceedings 10th IEEE International Symposium on High Performance Distributed Computing. IEEE, 2001. p. 181-194.

[2] 김동균; 이필우; 황일선. 그리드 컴퓨팅. 정보과학회지, 2002, 20.2: 5-10.

[3] 윤영석; 이현우. 개인 방송 플랫폼 기술: 아프리카 TV 와 유튜브를 중심으로. 한국통신학회지 (정보와통신), 2016, 33.4: 56-63

[4] BEECH, Mark. COVID-19 Pushes Up Internet Use 70% And Streaming More Than 12%, First Figures Reveal. ELLIOTT BRENNAN Research Associate, United States Studies Centre, 2020.

[5] LI, Bo; YIN, Hao. Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges [Peer-to-Peer Multimedia Streaming]. IEEE Communications Magazine, 2007, 45.6: 94-99.

[6] MAGHAREI, Nazanin; REJAIE, Reza. Understanding mesh-based peer-to-peer streaming. In: Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video. 2006. p. 1-6.

[7] MERANI, Maria Luisa; NATALI, Laura. Adaptive streaming in P2P live video systems: A distributed rate control approach. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2016, 12.3: 1-23.

[8] 이상훈; 한치근. 연결 정보를 이용한 P2P 스트리밍 네트워크 구조의 개선. 한국컴퓨터정보학회논문지, 2012, 17.5: 49-57.

[9] LIANG, Chao; GUO, Yang; LIU, Yong. Hierarchically clustered p2p streaming system. In: IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference. IEEE, 2007. p. 236-241.

[10] 최순; 변해선; 이미정. 라이브 미디어 스트리밍 서비스를 위한 P2P 기반의 그룹트리 오버레이 구조 및 기법. 한국정보과학회 학술발표논문집, 2009, 36.1D: 389-392.

페이로드 시그니처를 이용한 Adobe 소프트웨어 사용자 행위 탐지

이민성, 최정우, 권윤주, 박지태
고려대학교
{min0764, choigoya97, tkwkf12, pjj5846}@korea.ac.kr

Adobe Software User-behavior Detection Using Payload Signature

Lee Min-Seong, Park Jee-tae, Choi jeong-woo, Kwon Yun-Ju
Korea Univ.

요 약

SaaS는 클라우드 기반 소프트웨어 제공 모델로서 구독의 형태로 서비스를 사용 할 수 있다. SaaS 서비스가 확산되면서 많은 기업 및 단체에서 Office365, Adobe, G-suite 등 많은 서비스들을 구입하여 사용하고 있다. 서비스를 구매하여 사용하다 보니 기업과 단체는 소프트웨어 사용 시 모니터링을 통한 관리가 필요하다. 이러한 모니터링을 진행하기 위해서는 각 서비스별로 사용자가 어떤 소프트웨어를 사용하며, 소프트웨어의 어떤 행위를 했는지 파악을 할 필요가 있다. 본 논문에서는 여러 서비스들 중 Adobe 서비스를 대상으로 사용자가 소프트웨어 사용을 위해 어떤 행동을 해야 하는지 정의하고, 정의된 행위를 탐지할 수 있는 방법론에 대해 제안한다.

Ⅰ. 서 론

SaaS는 클라우드 기반 소프트웨어 제공 모델로서 구독의 형태로 서비스를 배포하고 있다. 마이크로소프트의 Office 365, 구글의 G Suite, Adobe의 Creative Cloud 등이 대표적인 SaaS 서비스에 해당한다. SaaS 어플리케이션은 초기 도입 비용이 상대적으로 적고 필요에 따라 사용을 조절할 수 있기 때문에 많은 기업에서 서비스를 구매하여 사용 하고 있다. 이에 따라 기존 소프트웨어를 SaaS 어플리케이션 형태로 제공하는 소프트웨어 회사들도 증가하였으며 구매의 형태도 다양하게 진행되고 있다. 예를 들어, 가장 많이 사용되어온 소프트웨어들만 구매하여 사용을 할 수 있게 하고, 추가적인 금액으로 더 많은 소프트웨어를 사용 할 수 있도록 한다. 기업 및 단체는 필요한 소프트웨어를 사용하기 위하여 선택적으로 구매하여 SaaS 서비스를 사용 할 수 있다. 기업 및 단체의 입장에서 서비스를 구매하여 사용하다 보니 모니터링을 통한 관리가 필요하게 된다. 구매한 서비스를 사용을 하고 있는지 확인하여 적절한 금액으로 필요한 서비스만 사용 할 수 있도록 관리 할 수 있고, 이는 기업이나 단체의 효율적인 자산 운용을 가능하게 한다. 이러한 모니터링을 진행하기 위해서는 각 서비스별로 사용자가 어떤 소프트웨어를 사용했는지 파악 할 수 있어야 하며, 각 소프트웨어에서 사용자가 필수적으로 어떤 행위를 하게 되는지 정의되어야 한다. 본 논문에서는 Adobe 서비스를 대상으로 트래픽을 분석하여 사용자의 행위를 정의하고 정의된 행위를 탐지 할 수 있는 방법론을 제시한다.

Adobe 트래픽을 분석하기 위하여 접근하기 쉬운 페이로드 시그니처를 정의하여 행위를 탐지 할 수 있도록 한다. Adobe트래픽은 여러 SaaS 서비스들과 마찬가지로 SSL/TLS를 기반으로한 암호화된 패킷으로 통신을 한다.[1][2] 따라서 Adobe 소프트웨어 사용 시 발생하는 모든 플로우를 분석하는 것이 아닌 443Port에 해당하는 플로우들만 수집하여 분석하고 페이로드 시그니처를 정의한다.

Adobe에서 제공하는 여러 서비스들의 트래픽을 분석하기 위하여 여러 행위들을 포함한 트래픽을 수집하고 행위를 정의한다. 본 논문의 구성은 서론에 이어, 2장에서 Adobe 행위 탐지를 위한 행위 분석 및 정의와 트래픽 분석 방법론에 대해 기술하고, 마지막으로 결론 및 향후 연구에 대해 기술한 후 마친다.

Ⅱ. 본 론

본 장에서는 Adobe 트래픽을 분석하여 사용자가 Adobe 소프트웨어를 사용할 때의 행위를 탐지하는 방법론에 대하여 기술한다. Adobe의 소프트웨어를 사용하기 위해서는 여러 소프트웨어를 설치하고 사용할 수 있는 통합 서비스인 Adobe Creative Cloud를 사용하는 방법이 있다. 또한, Adobe Creative Cloud를 통해 각 소프트웨어들을 미리 설치 한 후 원하는 소프트웨어를 개별적으로 실행하는 방법이 있다. 본 논문에서는 Adobe Creative Cloud에서 소프트웨어를 실행하는 것은 제외하였다. 소프트웨어는 Premiere Pro, Photoshop, Illustrator 3가지 소프트웨어를 사용하여 각 소프트웨어에서 발생한 트래픽을 대상으로 분석을 진행하였다.

A. Adobe 사용자 행위 분석

Adobe 사용자의 행위 분석을 위하여 소프트웨어를 사용하기 위하여 필수적으로 진행해야 하는 행위를 정의하였다. 3가지 소프트웨어를 사용하면서 여러 가지 공통된 행위를 진행하여 플로우를 분석하였으며, 분석 시 공통적으로 발생한 플로우를 바탕으로 행위를 정의하였다. 기본적으로 3가지 소프트웨어를 사용하였기 때문에, 어떤 소프트웨어를 사용했는지 분석을 진행해야 한다. 하지만, 현재 페이로드 시그니처 기반 분석 방법으로는 소프트웨어를 구분 할 수 없었다. 따라서 소프트웨어 이름 탐지는 제외하였다.

사용자가 Adobe를 사용할 때 나타나는 행위는 크게 소프트웨어 실행, 로그인, 로그아웃, 소프트웨어 종료로 4가지로 정의하였다. 소프트웨어 실행은 사용자가 소프트웨어를 실행을 할 때의 행위이다. 로그인은 사용자가 Adobe 소프트웨어를 사용하기 위하여 필수적으로 해야 하는 행위이다. Adobe 서비스 자체가 구독의 형태로 진행되기 때문에 로그인이 되어 있지 않으면 소프트웨어를 사용 할 수 없다. 또한, 로그인을 하지 않으면 실

행했던 소프트웨어가 자동으로 종료된다. 로그아웃은 사용자가 로그아웃을 하는 행위이며, 소프트웨어 종료는 소프트웨어 사용을 마치고 종료하는 행위이다. 로그인에서와 마찬가지로, 로그아웃을 하게 되면 소프트웨어가 자동적으로 종료된다.

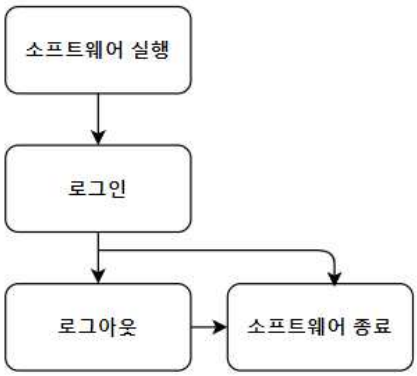


그림 1. 사용자 행위 분석

B. 트래픽 분석 방법 및 행위 탐지

트래픽 분석 방법으로는 각 행위 별 페이로드 시그니처를 정의하여 각 4가지 행위를 탐지하였다. Adobe 소프트웨어를 사용하게 되면 SSL/TLS 기반으로 암호화된 패킷으로 이루어진 플로우가 발생한다. 암호화된 패킷은 443 Port를 사용하여 통신하기 때문에 443 Port에서 발생한 플로우를 기준으로 플로우를 모아서 해당 행위에 대한 페이로드 시그니처를 정의하였다. 정의된 시그니처는 표1에서 확인 할 수 있다.

표 1. 페이로드 시그니처

Behavior	Payload Signature
소프트웨어 실행	"hbrt.adobe.com"
로그인	"oobe.adobe.com"
	"adobeid-nal.services.adobe.com"
	"workflowlicenses.adobe.com"
로그아웃	"sstats.adobe.com"
	"ims-prod06.adobelogin.com"
소프트웨어 종료	"hbrcv.adobe.com"

소프트웨어 실행 시 정의된 페이로드 시그니처를 포함하는 플로우가 발생하며 이를 통해 소프트웨어 실행을 탐지한다. 실행 시 발생하는 시그니처를 가진 플로우는 Adobe 실행 중에도 발생을 한다. 따라서 해당 플로우가 최초로 탐지된 경우에만 실행을 탐지한다. 로그인 행위는 "oobe.adobe.com"시그니처를 가진 플로우 발생 이후 나머지 2가지 시그니처를 포함한 플로우가 5초 이내에 발생하면 탐지된다. 로그아웃 행위 시에는 "sstats.adobe.com"시그니처를 포함한 플로우가 발생 한 후 5초 이내에 나머지 시그니처를 가진 플로우가 발생하면 로그아웃 탐지를 할 수 있다, 소프트웨어 종료 시 정의된 시그니처를 포함한 플로우 발생 시 소프트웨어 종료를 탐지 할 수 있다.

C. 실험 및 검증

개인 PC, 노트북, 학내망 등 여러 환경에서 사용자가 Adobe 서비스를 사용하고 어떤 행위를 하였는지 탐지 하는 실험을 진행하였다. 개인 PC나 노트북에서 수집한 트래픽의 경우, 수집하면서 여러 행동을 조합하여 트래픽을 수집하였으며, 실험 시 해당 행위가 잘 탐지되는지 확인하였다. 실험을 진행하였을 때, 정확하게 사용자의 행위를 탐지 할 수 있었다. 학내망 트래픽의 경우 Adobe 서비스를 사용하는 행위를 탐지 할 수 있었으나 여러 호스트에서 발생하는 트래픽이 모여있기 때문에 정확한 탐지가 되었는지 확인 할 수 없었다. 하지만 호스트 정보와 행위 탐지가 가능하다는 것은 확인하였다.

실험 결과 일반적으로 전체 행위를 진행한다고 가정하였을 때, 소프트웨어가 실행 된 후 로그인을 하고 로그아웃 후 소프트웨어 종료를 진행한다. 하지만 결과에서 소프트웨어 실행 이전에 로그인이 탐지가 되는 경우가 있었다. 이는 소프트웨어 실행 시 발생하는 플로우를 소프트웨어를 실행하자마자 탐지를 진행하는 것이 아닌, 사용자가 작업을 시작하기 위하여 작업 창을 띄우는 시점을 소프트웨어 실행으로 정의하였기 때문이다. 로그인이 되어있지 않을 때, 로그인을 진행하지 않으면 소프트웨어가 종료되고 작업 창을 띄울 수 없다. 따라서 로그인을 한 후 작업창이 뜨게 되는데, 이러한 결과로 소프트웨어 실행시보다 로그인 행위가 먼저 탐지가 되었다. 또한 3가지 소프트웨어(Premiere Pro, Photoshop, Illustrator)를 사용하여 행위를 분석하였는데, 사용자가 같은 행위를 하였을 때 소프트웨어의 구분 없이 공통적으로 행위가 탐지되는 것을 확인하였다. 향후 어떤 서비스의 어떤 행위를 하였는지 탐지하기 위해서는 논문에서 제시한 사용자 행위 탐지 이전에 소프트웨어의 구분이 필요하다.

Ⅲ. 결론 및 향후 연구

SaaS 서비스의 사용량이 많아지면서 기업 및 단체에서 서비스를 구매하여 여러 가지 소프트웨어들을 사용하고 있다. 적절한 구매를 통해 효율적인 자산 운용을 위하여 SaaS 서비스에 대한 모니터링이 필수적이다. 모니터링을 하기 위해서는 SaaS 서비스의 소프트웨어를 사용 할 때 사용자의 행위 분석이 필요하다.

본 논문은 SaaS 서비스 중 Adobe 서비스를 대상으로 사용자 행위를 정의하고 탐지할 수 있는 페이로드 시그니처를 정의하였다. 제시한 방법론으로 사용자가 Adobe 서비스의 소프트웨어 사용 시 어떤 행위를 하였는지 탐지 할 수 있었다. 하지만, 페이로드 시그니처는 버전이 바뀌는 등 페이로드의 내용이 변할 수 있기 때문에 페이로드 시그니처만 사용하여 행위를 정의하는 것은 한계가 있다. 향후 연구로 사용자가 소프트웨어를 사용하면서 발생하는 플로우의 통계 정보를 바탕으로 탐지 알고리즘을 정교화 할 예정이다.

ACKNOWLEDGMENT

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2018R1D1A1B07045742)과 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발)

참 고 문 헌

[1] 김성민, 박준상, 윤성호, 김종현, 최선호, 김명섭, "SSL/TLS 기반 암호화 트래픽의 서비스 식별 방법", 통신학회 논문지 Vol 40 No.11, Nov.2015, pp.2160-2168.

[2] 김성민, 구영훈, 김명섭, "Session ID - Server IP 캐싱 기반의 SSL/TLS 암호화 트래픽의 서비스 식별 방법", 2015년도 한국통신학회 하계종합학술발표회, 라마다호텔, 제주도, Jun. 23-25, 2015.

SNI 정보를 활용한 Office 365 사용자 행위 탐지

최정우, 박지태, 이민성, 권윤주
고려대학교
{choigoya97, pjj5846, min0764, tkwkf12}@korea.ac.kr

Identification of Office 365 User Behaviors using SNI Information

Jeong Woo Choi, Jee-tae Park, Min-Seong Lee, Yun-Ju Kwon
Korea Univ.

요 약

SaaS는 클라우드 기반으로 소프트웨어 제공 모델이다. 최근 많은 기업들에서 SaaS의 사용이 증가하고 있다. 기업의 효과적인 자산 관리를 위해서는 관리 부서에서 SaaS의 구매 및 사용을 관리해야 한다. 하지만 승인하지 않은 클라우드 소프트웨어를 구매하여 사용하는 Shadow IT 현상이 발생하면서 문제가 발생하고 있다. 이러한 문제를 해결하고 효율적으로 사용하기 위해서는 SaaS 사용 모니터링이 필요하다. 본 논문에서는 설치 기반 Office 365를 사용한 트래픽을 IP / Port 기반 분석 방법과 SNI 정보를 활용한 분석 방법을 사용하여 사용자의 행위를 탐지하는 방법론에 대해 제안한다.

Ⅰ. 서 론

SaaS는 공급자나 서비스 제공자가 서버 상에 애플리케이션을 호스팅하고, 고객은 웹 브로우져 등 온라인을 통해 사용한 만큼 비용을 지불하고, 소프트웨어를 서비스로 이용할 수 있도록 하는 소프트웨어 배포 모델을 의미한다[1]. 마이크로소프트의 Office 365, Adobe Creative Cloud 등이 있다. SaaS 서비스는 초기 비용이 상대적으로 적고 사용자의 필요에 따라서 사용할 수 있다는 장점으로 인해서 최근 사용이 많아지고 있다. 특히 기업에서도 많이 사용하는 추세이다. 최근 소프트웨어의 관리에 대한 새로운 방식의 필요성이 대두되면서 SaaS 애플리케이션을 모니터링하고 관리하는 방법에 대한 연구가 진행되고 있다.

모니터링을 통한 관리가 중요한 이유는 Shadow IT 때문이다. Shadow IT란 승인하지 않은 클라우드 소프트웨어를 구입하고, 이를 IT 관리부서나 책임자가 파악하지 못하는 현상을 의미한다.[2] 기업에서 SaaS 애플리케이션을 사용할 때 관리가 제대로 이루어지지 않는 경우가 발생하게 되며, 이는 기업의 손해로 이어진다. 따라서 이러한 문제를 해결하고 기업의 손해를 줄이기 위해서 SaaS 사용 모니터링의 필요성이 대두되고 있다.

트래픽 분석에 사용되는 방법론으로는 IP / Port 기반 분석 방법, 통계 정보를 활용한 분석 방법, 머신 러닝을 활용한 분석 방법, SNI 정보를 활용한 분석 방법 등이 있다. 본 논문에서는 IP / Port 기반 분석 방법과 SNI 정보를 활용한 분석 방법을 사용해서 설치기반 Office 365의 사용자 행위를 탐지하는 방법을 제안한다.

설치 기반 Office 365는 우리가 가장 많이 사용해온 SaaS 애플리케이션이며, 현재도 문서 작업이나 발표를 위해 많이 사용되는 애플리케이션이다. 따라서 본 논문에서는 설치 기반 Office 365를 사용한 트래픽을 분석하여 사용자의 행위를 탐지하여 사용자가 어떤 행위(로그인, 로그아웃, 실행 등)를 했는지에 대한 정보를 제공할 수 있는 방법을 제안하고자 한다.

본 논문의 구성은 본론에서 설치 기반 Office 365 사용자의 행위를 정의한다. 이 후, 설치 기반 Office 365 사용자의 행위를 탐지하는 방법론에 대해 언급한다. 결론에서는 해당 연구를 정리하고 향후 연구에 진행할 내용에 대해 언급하는 순서로 진행된다.

Ⅱ. 본론

본 장에서는 설치 기반 Office 365를 사용한 트래픽을 분석하여 사용자의 행위를 탐지하는 방법론에 대해 기술한다. 트래픽 분석은 SNI 정보를 활용한 분석 방법과 IP / Port 기반 분석 방법을 사용했다. 표 1은 사용자의 행위를 정의해놓은 것이다. 본 논문에서는 기본적인 사용자 행위에 대한 탐지를 목적으로 진행되었기 때문에 Onedrive 접속과 설치형 종료에 대한 탐지는 제외한다.

행위	정의
설치 기반 Office 365 실행	애플리케이션 실행
로그인 시도	ID, Password 입력
로그인 성공	로그인 시도 후 로그인 성공
Onedrive 접속	애플리케이션 실행 후 최초 Onedrive 접속
설치형 종료	애플리케이션 종료
로그아웃	시스템 애플리케이션에서 마이크로소프트 로그아웃

표1.사용자행위정의

각 행위 탐지에 대한 알고리즘의 공통적인 틀은 HTTPS의 Port 번호인 443에 해당하는 플로우들만 대상으로 SNI 정보를 확인해서 각 행위에 대한 시그니처를 정의한다. 이 후, 행위 별로 정의된 시그니처를 포함하는 플로우가 발생하면 해당 행위가 탐지되었다고 판단한다. 그림1, 그림 2, 그림 3 은 각각 설치 기반 Office 365 실행, 로그인 시도 & 로그인 성공, 로그아웃의 알고리즘을 그림으로 표현한 것이다.

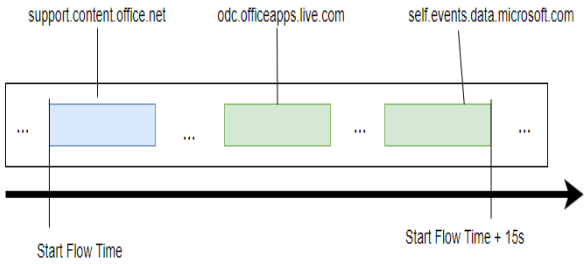


그림 1. 설치 기반 Office 365 실행 탐지

우선 설치 기반 Office 365 실행을 탐지하는 알고리즘에 대해 기술한다. 설치 기반 Office 365 실행을 탐지하는 알고리즘은 HTTPS의 Port 번호인 443 Port에 해당하는 플로우들만 고려한다. 이 후, 설치 기반 Office 365 실행 시 발생하는 공통적인 플로우의 SNI 정보를 바탕으로 정의된 시그니처로는 [support.content.office.net], [odc.officeapps.live.com], [self.events.data.microsoft.com]이 있다. [support.content.office.net]을 포함한 플로우가 발생하면 해당 시간을 저장한다. 이 후, 해당 시점으로부터 15초 이내에 동일한 Host IP에서 [odc.officeapps.live.com], [self.events.data.microsoft.com]을 포함한 플로우가 발생하게 되면 설치 기반 Offce 365 가 실행이 탐지되었다고 판단한다.

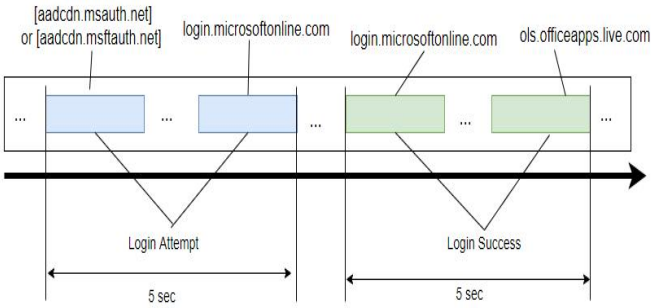


그림 2. 로그인 시도 & 로그인 성공 탐지

본 절에서는 로그인 시도와 로그인 성공을 탐지하는 알고리즘에 대해 기술한다. 로그인 시도, 로그인 성공, 로그아웃의 경우에는 백그라운드에서 작동하는 시스템 앱에서 작동한다. 로그인 과정은 로그인 시도와 로그인 성공으로 구분할 수 있다. 로그인 시도는 ID 입력과 Password를 입력하는 과정을 의미한다. 로그인 성공은 로그인 시도가 발생한 후에 로그인이 성공하는 것을 의미한다. 로그인 시도만 발생하는 경우는 로그인이 실패했다고 판단한다.

로그인 시도 와 로그인 성공을 탐지하는 알고리즘 또한 HTTPS의 Port 번호인 443 Port에 해당하는 플로우들만 고려한다. 이 후, 로그인 시도 시 발생하는 공통적인 플로우의 SNI 정보를 바탕으로 정의된 시그니처로는 [aadcdn.msauth.net], [aadcdn.msftauth.net], [login.microsoftonline.com]이 있다. [aadcdn.msauth.net] 또는 [aadcdn.msftauth.net]을 포함한 플로우가 발생하면 해당 시간을 저장한다. 이후, 해당 시점으로부터 5초 이내에 [login.microsoftonline.com]을 포함한 플로우가 발생하게 되면 로그인 시도가 탐지되었다고 판단한다.

로그인 시도 탐지 후에 로그인 성공 시 발생하는 공통적인 플로우의 SNI 정보를 바탕으로 정의된 시그니처로는 [login.microsoftonline.com], [ols.officeapps.live.com]이 있다. 두 개의 시그니처를 포함한 플로우가 발생하게 되면 로그인 성공이 탐지되었다고 판단한다.

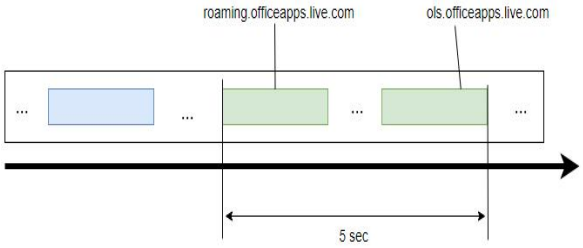


그림 3. 로그아웃 탐지

마지막으로 로그아웃을 탐지하는 알고리즘에 대해 기술한다. 로그아웃은 시스템 애플리케이션에서 마이크로소프트 로그아웃하는 것을 의미한다. 로그아웃을 탐지하는 알고리즘도 HTTPS의 Port 번호인 443 Port에 해당하는 플로우들만 고려한다. 로그아웃 시 발생하는 공통적인 플로우의 SNI 정보를 바탕으로 정의된 시그니처로는 [roaming.officeapps.live.com]이 있다. 해당 시그니처를 포함한 플로우가 발생하면 해당 시간을 저장한다. 이 후, 해당 시점으로부터 5초 이내에 [ols.officeapps.live.com]을 포함한 플로우가 발생하게 되면 로그아웃이 탐지되었다고 판단한다.

Ⅲ. 결론

본 논문에서는 SNI 정보를 활용하여 설치 기반 Office 365 사용자의 행위를 탐지하는 방법론에 대해 제안한다. 최근 많은 기업들에서 SaaS 어플리케이션의 많은 장점들 때문에 사용이 증가하는 추세이다. 하지만 기업에서는 관리되지 않고 승인되지 않은 사용이 발생하면, 이는 기업의 손실로 돌아온다. 이러한 Shadow IT 현상을 해결하기 위한 방법으로 SaaS 모니터링 기술의 필요성이 대두되고 있다.

본 논문에서는 SaaS 어플리케이션 중에서도 우리가 많이 사용하는 파워포인트, 엑셀, 워드와 같은 서비스를 제공하는 마이크로소프트사의 설치 기반 Office 365를 대상으로 트래픽 분석을 진행했다. 트래픽 분석 방법으로는 IP / Port 기반 분석 방법과 플로우의 SNI 정보를 활용하는 분석방법을 사용한다. HTTPS의 Port 번호인 443 Port에 해당하는 플로우들만 고려하여 SNI 정보를 바탕으로 시그니처를 정의해 설치 기반 Office 365 실행, 로그인 시도와 로그인 성공, 로그아웃에 대한 사용자 행위를 탐지하는 방법론에 대해 제안한다.

향후 연구로 본 논문에서 제안한 방법론을 활용해서 Onedrive 접속, 설치형 종료와 같은 추가적인 행위들에 대해서도 분석을 진행할 예정이다. 또한 Adobe Creative Cloud와 같은 다른 SaaS 애플리케이션 서비스에 대해서도 사용자 행위 탐지를 위한 트래픽 분석을 진행할 예정이다.

ACKNOWLEDGMENT

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2018R1D1A1B07045742)과 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발)

참 고 문 헌

[1] 김형환 외 12명, “SaaS 기술 개발 동향”, 전자통신 동향분석, 제24권, 제4호, 2008.

[2] 이민성, 박지태, 최정우, 김명섭, “페이로드 시그니처를 이용한 마이크로소프트 Office 365 서비스 탐지”, 2020년도 한국통신학회 추계종합학술발표회, Nov. 13, 2020

웹 기반 영상회의에서 딥 러닝 모델 적용을 위한 방법 연구

유상우, 고경찬, 홍원기
포항공과대학교
{rswoo, kkc90, jwkhong}@postech.ac.kr

Research on Applying Deep Learning Methods on Web-based Video Conferencing Systems

Sangwoo Ryu[1], Kyungchan Ko[2], James Won-Ki Hong[2]
Graduate School of Artificial Intelligence, POSTECH[1]
Department of Computer Science and Engineering, POSTECH[2]

요 약

인공지능 기술의 발전으로 공장 자동화, 자율주행 등 다양한 산업 분야에서 인공지능이 사용되고 있다. 영상회의 시스템에서 또한 기존 알고리즘의 한계를 극복하기 위해 인공지능을 이용하는 기능들을 추가해 왔고, 대표적으로 초해상화(Super Resolution), 이미지 세분화를 이용한 가상배경 기능 등이 있다. 하지만 웹 기반 영상회의에서는 제한된 웹 환경으로 인해 이런 기능들의 적용이 제한된다. 본 논문에서는 이러한 웹 기반 서비스에서 딥 러닝 모델을 사용하는 기능을 제공하기 위해 웹 환경에서 딥 러닝을 적용하는 여러 방식을 소개하고, 각 방식에서 가상 배경 기능을 위해 사용하는 이미지 세분화 모델을 소개하고 성능을 평가한다. 마지막으로 웹 기반 영상회의에 딥 러닝 모델을 적용하기 위해 고려해야 하는 부분을 논의한다.

Ⅰ. 서론

코로나-19 이후, 대면 업무를 지양하는 분위기가 형성되면서, Zoom, Webex 등 영상회의 서비스를 이용한 업무가 늘어나고 있고, 영상회의를 사용하는 고객의 요구사항 또한 늘어났다. 이에 따라 영상회의 서비스 제공자는 고객의 요구사항을 충족시키고, 사용 만족도를 높이기 위해 다양한 부가 기능을 추가하고 있다. 그 중 일부 기능들은 딥 러닝(Deep Learning, 심층학습)[1] 이전 방식보다 딥 러닝을 적용했을 때 더 좋은 성능을 보여주는데[24], 예를 들어 화질을 개선하기 위한 초해상화(Super Resolution), 얼굴의 정면을 자동으로 보여주는 얼굴 정렬(Face Alignment), 사용자의 배경을 바꾸는 가상 배경(Virtual Background) 등 다양한 기능에서 딥 러닝이 사용될 수 있다.

설치형 영상회의 서비스들은 Python/C++ 와 같은 프로그래밍 언어들을 직접적으로 사용할 수 있고 많은 라이브러리들이 존재해 딥 러닝을 쉽게 적용할 수 있다. 반면에 웹 앱들은 대부분 자바스크립트를 통해 기능이 제공되기 때문에, 웹 기반 영상회의 서비스들은 보다 제한적인 환경에서 모델 적용을 해야 한다. 이 논문에서는 딥 러닝의 여러 적용 사례들 중, 가상 배경 기능에 초점을 맞추어, 이 기능을 적용하기 위한 이미지 세분화(Image Segmentation) 작업을 수행하는 모델을 웹 환경에 적용하기 위한 방법들을 서버, 클라이언트라는 두 적용 위치에 따라 소개한다. 특히 딥 러닝 모델을 클라이언트에 적용하는 경우 WebRTC [13] 기반 영상회의 서비스인 Vmeeting[17]에 다양한 방법으로 적용 및 성능 평가를 수행한다. 마지막으로, 웹 기반 영상회의에 딥 러닝 모델을 적용하기 위해 고려해야 하는 부분을 정리하고 차후 성능 및 사용자 경험 개선을 위해 필요한 연구들을 제시한다.

Ⅱ. 배경

1. 웹 환경에 딥 러닝 적용

웹 브라우저에 딥 러닝을 적용하는 방법은 모델을 적용하는 위치에 따라 나눌 수 있다.

딥 러닝 모델이 서버에 위치하는 경우에는, 브라우저와 별개로 서버의 자원(CPU, GPU)과 서버 측 코드를 이용해 딥 러닝 모델이 동작할 수 있기 때문에, 기존 방식을 그대로 이용할 수 있다. 다만 클라이언트 사이드와 입력 및 출력 데이터를 주고받아야 하기 때문에, 추가적인 대역폭 사용이 필요하다. 따라서 서비스 제공자의 입장에서 서버 구축 및 대역폭 확보를 위해 많은 비용이 소모될 수 있으며, 데이터의 종류에 따라 프라이버시 문제가 생길 수 있고, 서버에서의 연산으로 실시간 처리에 지연이 생길 수 있다.

딥 러닝 모델이 클라이언트에 위치하는 경우에는, 클라이언트의 자원을 이용해 딥 러닝 모델이 동작해야 하고, 영상회의 서비스 제공자가 그에 맞는 구현을 제공해야 한다. 클라이언트마다 서로 다른 기기를 사용하기 때문에, 모두에게 적절한 사용 환경을 제공하지 못할 수 있다는 단점이 존재한다.

1) 자바스크립트 (JavaScript) 라이브러리

구글 오픈소스 프로젝트 TensorFlow[2]에서는 웹에서 머신러닝을 사용하기 위해 TensorFlow.js 라이브러리를 제공하고 있다. TensorFlow.js에서는 일반적인 사용 사례들인 이미지 분류, 객체 감지, 신체 분절화 등에 대해서는 선행 학습된 모델을 제공하고 있고, 이미 존재하는 모델을 재학습시킬 수 있도록 지원한다. TensorFlow.js에서는 저장소 및 수학 연산을 위해 여러

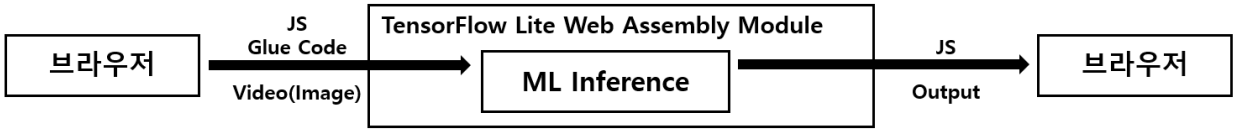


그림 1. TensorFlow Lite 웹 어셈블리 파이프라인

백엔드를 지원하는데 CPU, WebGL, WASM(Web Assembly) [18] 백엔드를 지원한다. 자바스크립트는 브라우저뿐만 아니라 React Native[19]를 통해 모바일 환경, Node.js[20]를 통해 IoT 기기로도 확장하여 사용할 수 있다.

TensorFlow.js 이외에도 Keras.js, ConvNetJS와 같은 자바스크립트 라이브러리에서 웹 브라우저에 딥 러닝을 적용하기 위한 도구들을 제공하고 있다[3].

2) 웹 어셈블리 (Web Assembly)

웹 어셈블리[4]는 웹에서 C/C++ 등으로 작성된 코드를 실행할 수 있도록 바이너리 형식으로 변환해, 가상 머신에서 실행하는 기술이다. TensorFlow에서 모바일 및 IoT 기기에서의 머신러닝 프레임워크인 TensorFlow Lite[2, 21]를 웹 어셈블리로 빌드하거나, 해당 라이브러리를 포함해 추론 과정 전체를 포함하는 프로그램을 웹 어셈블리로 빌드해 자바스크립트에서 해당 모듈을 불러와 모델을 사용한다. 모듈을 자바스크립트를 통해 불러오지만, 실제 연산은 순수 자바스크립트에서 수행하지 않기 때문에 빠른 속도를 보여준다. Google Meet에서는 실시간 영상에 기계학습 적용을 위한 구글 오픈소스 프레임워크인 Mediapipe를 웹 어셈블리로 빌드하여 가상 배경 기능을 위한 배경 분리를 딥 러닝 모델을 이용해 수행하였다[5]. 웹 어셈블리는 현재 Chrome, Safari, Firefox 등의 브라우저에서 2017년부터 지원하여 약 90%의 기기에서 지원을 하고 있다[6]. 자바스크립트를 통해 사용을 하기 때문에, 자바스크립트 라이브러리를 사용하는 경우와 마찬가지로 React Native 또는 Node를 이용해 모바일/IoT 기기에서 사용할 수 있다.

그림 1은 TensorFlow Lite를 웹 어셈블리로 빌드해 사용하는 경우의 파이프라인을 보여준다.

TensorFlow.js를 통해 WASM 백엔드를 사용하는 경우와 순수 웹 어셈블리를 사용하는 경우 모두 SIMD (Single Instruction, Multiple Data), 멀티 스레드(Multi Threads)와 같이 딥 러닝 연산을 가속화할 수 있는 추가적인 옵션을 선택할 수 있다는 장점이 있다.

2. 이미지 세분화 (Image segmentation)

이미지 세분화는 이미지에서 개체가 있는 위치, 모양, 픽셀을 판단해 픽셀 단위의 마스크를 출력하는 작업이다. 이미지 세분화는 자율 주행 자동차에서 도로, 차, 사람 등을 구분하기 위해 사용되거나, 의료 영상에서 특정 종양의 위치를 판단하는 등의 영역에서 사용될 수 있다. 영상 회의에서는 사람과 사람이 아닌 위치를 구분해 마스크를 만들고, 사람이 아닌 위치에 배경을 그리는 가상 배경 기능에 사용될 수 있으며, 그림 2는 실제로 영상회의에 가상 배경을 적용한 예시를 보여준다.

이미지 세분화를 위한 대표적인 딥 러닝 모델로는 DeepLab[7, 8]이 있으며, 특히 모바일 환경에 특화된 MobileNet[9, 10, 11] 또한 지속적으로 연구가 되고 있다.

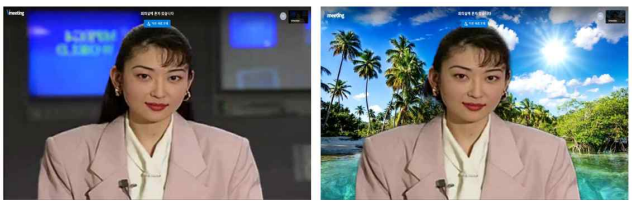


그림 2. 왼쪽 - 원본 영상, 오른쪽 - 가상 배경 적용

3. 웹 기반 영상회의 시스템

그림 3은 WebRTC 기반 영상회의 시스템의 기본적인 구조를 보여준다. 세션 기술 프로토콜(SDP, Session Description Protocol)[12]을 이용해 신호를 교환해 클라이언트들의 연결 지점을 알려주는 시그널링 서버와, 비디오, 오디오 등의 데이터를 전달하는 릴레이 서버로 이루어져 있다.

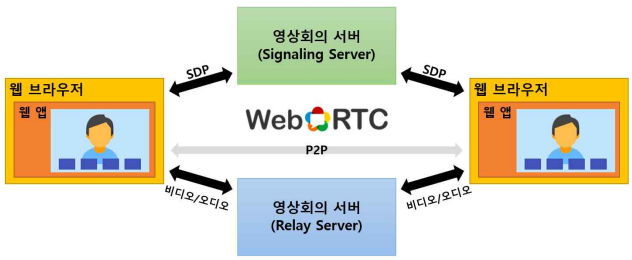


그림 3. WebRTC 기반 영상회의 구조

Ⅲ. 성능 분석

이 장에서는 딥 러닝 모델을 서버에 적용했을 때 이론적 대역폭 사용을 보여주고, 딥 러닝 모델을 Vmeeting 클라이언트에 적용했을 때 여러 적용 방식 별 성능을 보여준다.

1. 서버 측 적용

딥 러닝 모델을 서버에 적용하는 경우, 클라이언트의 자원을 사용하지 않기 때문에 서버의 자원(CPU, GPU)의 성능에 따라 성능이 결정된다. 다만 데이터를 서버로 전송하고 클라이언트로 다시 전송을 해야 하기 때문에 초해상화(Super Resolution)[22, 23]와 같이 해상도와 직접적인 관계가 있는 작업들의 경우 서버에서 클라이언트 방향 대역폭 사용에 직접적인 영향을 주기 때문에, 이를 사용하기 위해서는 미리 대역폭 사용량에 관한 계산이 필요하다. 가상 배경 기능을 위한 배경 분리와 같은 작업들의 경우 해상도와 큰 관련이 없기 때문에, 대역폭에 영향을 주지 않는다.

2. 사용자 측 적용

딥 러닝 모델을 클라이언트에 적용하는 경우, 클라이언트가 사용하는 기기(데스크탑, 모바일 기기 등)의 자원에 따라 성능이 결정된다. 서버에 적용할 때와 마찬가지로 수행하려는 작업과 방식에 따라 대역폭 사용에 영향을 준다. 초해상화(Super Resolution) 작업의 경우, 영상회의에서 처음 비디오를 만들어내는 과정에서 초해상화를 수행할 때는 클라이언트에서 서버로, 서버에서 클라이언트로 대역폭 두 곳 다 영향을 주지만, 영상회의에서 영상을 받는 쪽에서 화면에 표시되는 사람의 영상에 대해서만 수행할 때는 일반 영상회의와 큰 차이가 없다.

그림 4는 N명이 참여하고 참여자에서 영상회의 서버로 전송하는 기본 대역폭이 M kbps인 영상회의에서 영상의 너비, 높이를 2배씩 증가시키는 2x Super Resolution을 수행할 때, 기본 상태와 서버에 적용하는 경우, 클라이언트는 보내는 시점, 받는 시점에 적용하는 경우 총 4가지 경우에서 예상 대역폭 사용을 보여준다.

아래에서는 가상 배경 기능을 위해 각 적용 방식에서 제공하거나 사용할 수 있는 이미지 세분화(Image Segmentation) 모델을 WebRTC[13] 기반 영상회의인 Vmeeting에 적용했을 때의 성능을 보여준다. 이 모델은 영상의 해상도를 변경하지 않기 때문에 대역폭에 큰 영향을 주지 않는다.

FPS 및 추론 시간은 2.90GHz i5-9400F CPU, 16.0GB RAM 데스크탑 PC의 Chrome 브라우저에서 100초 간 수행한 결과의 평균을 이용하였다. 라이브 스트리밍 지원 프로그램인 OBS Studio[14]에서 제공하는 가상 카메라를 이용하였으며, JVT(Joint Video Team) 테스트 영상들 중 영상회의와 유사하게 실내 환경에서 상체만 나오는 10초 길이의 “Akiyo” 영상을 반복 재생하였다.

1) 자바스크립트 - TensorFlow.js

TensorFlow.js에서는 사람-배경 분리를 위한 모델로, Body-pix[15]를 제공하고 있으며, 딥 러닝 모델로 ResNet-50[16]과 MobileNet-V1[9]을 선택할 수 있다. 각 모델에 대해 출력 스트라이드 (Output Stride), 양자화 (Quantization) 정도를 조절할 수 있으며, MobileNet-V1의 경우 컨볼루션

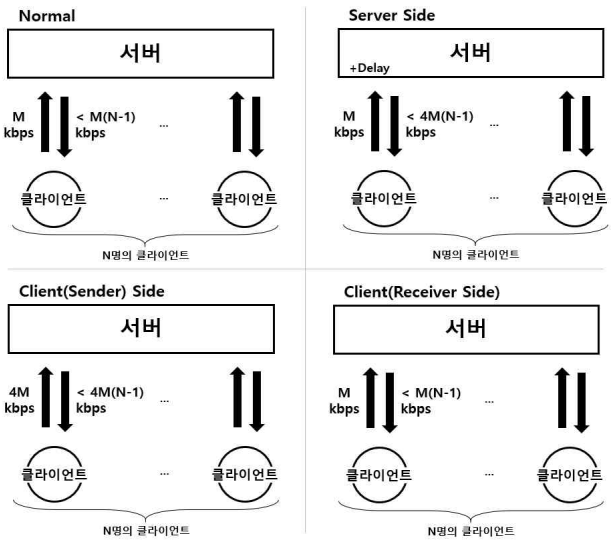


그림 4. 2x Super-Resolution 연산에서 예상 대역폭 사용

(Convolution) 연산에서의 채널 수를 조절해 전체 모델의 크기를 0.6MB부터 13MB까지 조정할 수 있다. 제공하는 모델 이외에도 필요한 경우 커스텀 모델을 불러와 사용할 수 있다.

TensorFlow 공식 문서에서는 중간 크기 모델 (~100-500M multiply-adds)은 WASM 백엔드가 WebGL 백엔드보다 느리고, 가벼운 모델 (~20-60M)은 WASM 백엔드가 빠르다고 설명한다.

표1은 중간 크기 모델인 MobileNet-V1을 적용하고 WebGL 백엔드를 사용한 경우 영상회의에서 배경 분리 작업의 성능을 보여준다.

Multiplier	quantBytes=2			quantBytes=1		
	크기	FPS	추론 시간 (ms)	크기	FPS	추론 시간 (ms)
0.75	~2MB	8.06	80.28	~1MB	7.97	76.94
0.5	~2MB	8.93	70.32	~0.6MB	8.94	67.93

표 1 . TensorFlow.js MobileNet-V1 WebGL 백엔드 성능

모델의 채널 수(Multiplier), 양자화(quantBytes) 정도를 조정해 모델의 크기와 연산의 수를 조정하여 추론 속도(추론 시간)를 조절할 수 있는 것을 확인할 수 있다. 다만 이에 따라 추론 정확도가 달라질 수 있다.

2) 웹 어셈블리 (Web Assembly)

웹 어셈블리를 이용하는 경우 사용하려는 머신러닝 라이브러리에 맞는 커스텀 모델을 사용해야 하는데, TensorFlow Lite를 사용하는 경우, TensorFlow Lite에서 제공하는 이미지 세분화 모델 등 TensorFlow Lite에서 동작할 수 있는 모델들을 사용할 수 있다.

표 2는 Google Meet에서 Apache 2.0 라이센스로 제공했던 MoblieNetV3-small 개선 모델을 적용한 영상회의에서 배경 분리 작업의 성능을 보여준다.

Non-SIMD			SIMD		
크기	FPS	추론 시간 (ms)	크기	FPS	추론 시간 (ms)
400KB	16.97	8.38	400KB	17.45	6.91

표 2 . 웹 어셈블리 - Google Meet Segmentation 모델 성능

표 3은 TensorFlow Lite에서 이미지 세분화를 위해 제공하는 DeepLab-V3[7]을 적용한 영상회의에서 배경 분리 작업의 성능을 보여준다.

Non-SIMD			SIMD		
크기	FPS	추론 시간 (ms)	크기	FPS	추론 시간 (ms)
2.7MB	3.16	260.44	2.7MB	8.11	63.98

표 3 . 웹 어셈블리 - Google Meet Segmentation 모델 성능

Google Meet Segmentation 모델은 양자화 및 최적화가 수행된 모델이기 때문에 상대적으로 크기가 큰 DeepLab-V3 모델보다 더 좋은 성능을 보여주었고, 두 모델 모두 SIMD를 사용했을 때 더 좋은 성능을 보여준다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 기존 딥 러닝을 사용하는 일반적인 환경과 달리 제한적인 웹 환경에서의 딥 러닝 적용을 위해, 웹 기반 영상회의에서의 딥 러닝 모델 적용 방식들을 서버와 사용자라는 두 적용 위치에 따라 설명하였다. 특

히 사용자 측 적용을 했을 때 사용할 수 있는 자바스크립트 라이브러리와 웹 어셈블리 두 방식을 소개하였고, 각 방식을 사용했을 때 대역폭 사용 및 성능을 분석하였다.

웹 기반 영상회의에 딥 러닝을 적용하기 위해서는 다양한 요소들이 고려되어야 한다. 먼저 딥 러닝 모델이 수행하려고 하는 작업에 따라 자원 사용 및 네트워크 대역폭 사용량이 달라질 수 있기 때문에, 이를 미리 분석하고 적절한 모델 적용 위치를 선택하는 것이 필요하다. 또한 본 논문에서 소개한 TensorFlow.js의 백엔드들과 웹 어셈블리를 이용한 머신 러닝 추론 방식들은 모델 크기 및 계산 량에 따라 속도가 달라지기 때문에, 사용하려는 모델에 맞게 모델 적용 방식을 선택해야한다. 딥 러닝 모델 연구가 성능을 높이기 위해서만이 아닌, 모델을 최적화하기 위한 연구들도 함께 진행이 되고 있다. 영상회의와 같은 웹 서비스에서는 실시간으로 이미지 처리가 수행되어야 하기 때문에, 성능과 최적화 두 부분을 모두 고려한 딥 러닝 모델 선택이 필요하다. 마지막으로, 딥 러닝 모델의 결과물을 정확하고 자연스럽게 보여주기 위한 방법을 연구할 필요가 있다.

클라이언트에서 딥 러닝 모델을 사용하는 경우에는 모델의 성능이 클라이언트 기기 성능에 의존하기 때문에, 실시간 영상회의 환경에서 사용자 경험을 해치지 않기 위해 환경에 적응할 수 있는 딥 러닝 모델 적용 연구가 필요하다. 이와 동시에 MobileNet 모델과 같이 저성능 디바이스용 모델들도 지속적인 개선 연구가 필요하다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발, 2017-0-01633, 대학ICT연구센터육성지원사업).

참 고 문 헌

[1] LeCun, Y., Bengio, Y. & Hinton, G. "Deep learning," Nature 521, pp. 436 - 444, 2015, (https://doi.org/10.1038/nature14539).

[2] Martín Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, (https://tensorflow.org).

[3] Ma, Y. et al.. "Moving Deep Learning into Web Browser: How Far Can We Go?," World Wide Web Conference, pp. 1234 - 1244, Association for Computing Machinery, 2015

[4] Haas, A., et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185 - 200, 2017

[5] Background features in google meet, powered by web ml. (2020, October 30). Retrieved March 23, 2021, from https://ai.googleblog.com/2020/10/background-features-in-google-meet.html

[6] Can I use... support tables for HTML5, css3, etc. (n.d.). Retrieved March 23, 2021, from https://caniuse.com/wasm

[7] Chen, L.C. et al. "Rethinking Atrous Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1706.05587, 2017.

[8] Chen, L.C. et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1802.02611, 2018.

[9] Howard, A. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv e-prints, arXiv:1704.04861, 2017.

[10] Sandler, M. et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv e-prints, arXiv:1801.04381, 2018.

[11] Howard, A. et al. "Searching for MobileNetV3," arXiv e-prints, arXiv:1905.02244, 2019.

[12] Handley, Mark et al. "SDP: Session Description Protocol," IETF, doi:10.17487/RFC4566, RFC 4566, July 2006.

[13] A. Bergkvist et al. "WebRTC 1.0: Real-time communication between browsers," W3C Working Draft, Feb. 2015.

[14] OBS studio, https://obsproject.com/

[15] Tensorflow. BodyPix - Person Segmentation in the Browser, https://github.com/tensorflow/tfjs-models/tree/master/body-pix

[16] He, Kaiming et al. "Deep Residual Learning for Image Recognition," pp. 770-778. 10.1109/CVPR.2016.90, 2016.

[17] Vmeeting, https://vmeeting.io

[18] Haas, A. et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185 - 200, 2017.

[19] React Native, https://reactnative.dev/

[20] Node.js, https://nodejs.org/

[21] TensorFlow Lite, https://www.tensorflow.org/lite

[22] Yang, W. et al.. "Deep Learning for Single Image Super-Resolution: A Brief Review," arXiv e-prints, arXiv:1808.03344, 2018.

[23] Anwar, S. et al. "A Deep Journey into Super-resolution: A survey," arXiv e-prints, arXiv:1904.07523, 2019.

[24] Alom, Md. Zahangir et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures," Electronics. 8. 292. 10.3390/electronics8030292, 2019.

기계학습을 이용한 Office365 서비스 분류 알고리즘 성능 평가

권윤주, 이민성, 최정우, 박지태
고려대학교

{tkwkf12, min0764, choigoya97 pjj5846 }@korea.ac.kr

Performance Evaluation of Office365 Service Traffic Classification Algorithm Using Machine Learning

Yun-Ju Kwon, Min-Seong Lee, Jeong-Woo Choi, Jee-Tae Park
Korea Univ.

요 약

응용 프로그램을 실행할 때 발생하는 네트워크 트래픽들을 수집해 분류하는 것은 수많은 다른 네트워크 활동에 근본적으로 중요한 역할을 한다. 본 논문의 목적은 Decision Tree, Gaussian Naive Bayes 등의 두 가지 지도학습 알고리즘을 사용해 플로우의 통계 정보를 가지고 Office365 응용프로그램을 분류하고 두 알고리즘들의 성능을 서로 비교하는 것이다. SaaS(Software-as-a-Service) 어플리케이션을 실행시켜 모니터링을 한 다음 발생하는 트래픽으로 트래픽 데이터셋을 만든다. 데이터셋을 두 가지 분류모델에 적용해 비교하면 Decision Tree 분류 모델이 어플리케이션을 구별하는 데 Gaussian Naïve Bayes 분류 모델보다 더 효율적일 거라는 결과를 확인할 수 있다.

Ⅰ. 서 론

응용 프로그램을 실행할 때 발생하는 응용 트래픽을 분류하는 일은 오늘날 수많은 네트워크 분야, 즉 보안 모니터링, QoS(Quality of Service), 프로그램 사용자들의 행위탐지 등을 포함한 다양한 네트워크 분야에서 근본적으로 중요한 역할을 한다. 본 논문에서는 많은 네트워크 트래픽 중에서 Office 365 클라우드 서비스의 트래픽을 수집해 사용자가 사용한 office365 서비스가 무엇인지 분류하고 식별한다. Office365 의 대표적인 응용 프로그램인 Word, PowerPoint, Excel 으로 총 3 가지 프로그램을 구분한다. 클라우드 서비스는 사용자에게 인터넷 기반의 컴퓨팅 자원을 빌려주고 사용한 만큼 비용을 청구한다. 기업의 서비스 운영에 비용 절감과 현실적이고 효율적인 자산 활용에 도움을 주기 때문에 오늘날 기업들은 클라우드 서비스를 도입하고 있다.

본 논문은 기계 학습을 사용해 SaaS 클라우드 서비스 중 Office 365 트래픽을 분류하여 기업들이 트래픽 분석하는 일에 현실성과 효율성에 도움을 주고자 한다. Microsoft Network Monitor 를 사용하여 Office 365 응용프로그램의 네트워크 트래픽을 수집한다. 수집한 네트워크 트래픽 cap 파일 형식을 5-tuple 이 일치한 패킷들을 정의한 플로우(flow) 단위로 저장한 FWP(Flow With Packet)파일 형식으로 변환한다. 그런 다음에 플로우 별로 통계정보를 수집하여 데이터 셋으로 만들어 두 Decision Tree 분류 모델과 Gaussian Naïve Bayes 분류 모델에 적용해 정확도와 시간을 비교한다. [1]

[1] 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2018R1D1A1B07045742)과 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발)

이 두 모델은 네트워크 트래픽 플로우를 가지고 사용자들이 3 가지 Office 365 의 응용프로그램 중 어떤 프로그램을 서비스를 이용했는지 식별하는 것을 목표로 한다. Decision Tree 분류 모델이 Gaussian Naïve Bayes 보다 조금 더 높은 정확도를 제공한다는 결과를 보여준다.

본 장을 제외한 나머지 부분에서는 SaaS 클라우드 서비스 트래픽 분류 모델을 단계적으로 살펴보고 두가지 기계학습 분류모델들의 성능을 비교분석하고 마지막에는 실험 결과와 향후 연구방향을 제시한다.

Ⅱ. 본론

본 절에서는 어떻게 SaaS 클라우드 서비스의 트래픽 플로우를 식별하고 분류하는지를 두가지 기계학습 분류모델을 사용하여 단계적으로 설명을 한다.

A. 데이터 셋

실시간으로 발생하는 Office365 의 3 가지 응용 프로그램들(Word, PowerPoint, Excel)의 네트워크 트래픽을 수집한다. 패킷 단위로 수집한 트래픽은 .cap 파일 형식으로 저장을 한 다음에 .cap 파일을 5-tuple(source port, destination port, source ip address, destination ip address, protocol)이 일치한 패킷들의 집합 단위인 플로우(flow)로 [2] 저장한 FWP(Flow With Packet)파일 형식으로 변환을 시킨다. 플로우의 통계정보(패킷의 개수, 패킷 크기, 패킷 발생시간)를 수집을 하고 추출한 통계정보들로 분류 모델 입력에 적합하도록 전처리를 한다. 그런 다음, 수집했던 응용프로그램의 트래픽에 따라 통계정보를 라벨링을 한다. 지도학습 알고리즘은 데이터 세트 중 일부 데이터를 훈련용 데이터로 만들어 모델을 훈련시킨 다음 나머지를 테스트용 데이터로 사용해 예측을 출력하게 되는데 본 논문 실험에서는 전체 데이터셋 중에 테스트 사이즈를 0.3 을 주어 훈련용

데이터로 만들고 나머지는 테스트용 데이터로 만들어 실험한다.

B. 분류 모델

분류 모델로는 Decision tree 와 Gaussian Naïve Bayes 을 사용한다.

Decision Tree 알고리즘은 최소한의 시도와 적은 시간으로 예측을 출력할 수 있기 때문에 트래픽을 분석하는 데 중요한 알고리즘이다.

Gaussian Naïve Bayes 알고리즘은 확률기반으로 하는 예측방법이다. Naïve Bayes 분류 모델은 많은 메모리 량을 요구하지 않고 예측 정확도가 좋기 때문에 다른 분류 모델과 비교하면 성능이 우수하다. [1]

Ⅲ. 실험 및 결과

정확도를 높이는 실험결과를 도출하기위해 응용프로그램을 이진분할과 삼중분할을 하여 정확도를 비교한다. 암호화되지 않은 트래픽 플로우들이 담긴 Common 파일, 암호화된 트래픽 플로우들이 담긴 Encrypted 파일, Common 파일과 Encrypted 파일을 합친 Entire 파일 총 3 가지 데이터셋들을 가지고 각각 두 분류모델을 학습시켜 예측의 정확도를 비교 분석한다.

그림 1 은 Decision Tree 분류모델의 트리 깊이를 1 부터 200 까지 뒀을 때 나오는 정확도와 Gaussian Naïve Bayes 분류 모델을 200 번 돌렸을 때 나오는 정확도의 평균값, 최대값, 최솟값을 비교하여 보여준 표이다.

그림 2 은 Decision Tree 에서 데이터 셋 별로 트리의 최대 깊이에 따라 변하는 응용프로그램 트래픽의 이진분할과 삼진분할의 정확도를 나타낸 그래프이다. Decision Tree 분류모델 Gaussian Naïve Bayes 분류 모델을 200 번 돌렸을 때 나오는 정확도의 평균값, 최대값, 최솟값을 나타낸 것이다.

		Average(%)		Maximum(%)		Minimum(%)	
		DT	NB	DT	NB	DT	NB
Word <-> PPT <-> Excel	Common	62.069	36.827	70.989	43.344	42.32	31.058
	Encrypted	70.018	38.595	78.498	46.416	40.273	31.399
	Entire	65.422	37.75	72.696	30.375	41.979	49.146
Word <-> PPT	Common	74.819	52.871	83.582	41.293	56.218	60.695
	Encrypted	82.287	55.831	90.547	45.771	58.208	64.766
	Entire	77.429	58.486	86.567	45.275	59.701	68.159
PPT <-> Excel	Common	74.518	49.152	84.375	40.625	59.895	58.333
	Encrypted	81.168	58.33	87.5	48.958	60.937	69.27
	Entire	76.69	52.434	84.375	41.666	58.333	64.062
Word <-> Excel	Common	69.908	54.362	77.835	43.298	56.701	61.855
	Encrypted	73.776	51.253	82.989	43.8144	54.639	61.34
	Entire	71.693	54.027	81.443	43.298	60.309	61.34

그림 1 두 분류 모델 정확도와 비교

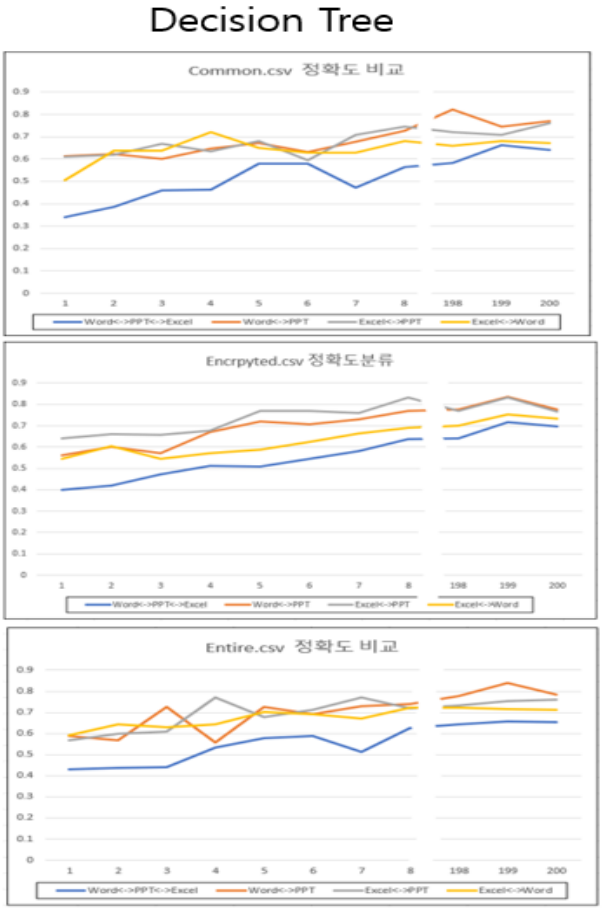


그림 2 트리 깊이에 따른 Decision Tree 정확도 비교

결과는 두 분류모델을 비교해 봤을 때 Decision Tree 분류모델이 Gaussian Naïve Bayes 분류모델보다 훨씬 더 높은 정확도를 보여준다.

Ⅳ. 결론 및 향후 연구 계획

본 논문은 클라우드 서비스 중 Office365 의 트래픽 플로우들로 총 3 가지 응용프로그램(Word, PowerPoint, Excel)으로 분류하기 위한 두가지 기계학습 알고리즘(Gaussian Naïve Bayes, Decision Tree)을 사용해 어떻게 네트워크 트래픽 분류 기술을 적용할지에 대한 방법론을 제시한다.

향후 연구로는 더 효율적인 분류 알고리즘을 찾기 위해 계속 조사하고 비교분석 할 예정이다.

참 고 문 헌

[1] SHAFIQ, Muhammad; YU, Xiangzhan; LAGHARI, Asif Ali. WeChat text messages service flow traffic classification using machine learning technique. In: 2016 6th International Conference on IT Convergence and Security (ICITCS). IEEE, 2016. p. 1-5.

[2] 정구현; 이봉환; 양동민. 패킷 페이로드 내 특정 패턴 탐지 알고리즘들의 성능 분석에 관한 연구. 한국정보통신학회논문지, 2018, 22.5: 794-804.

멀티 스텝 트래픽 예측을 위한 Bi-LSTM 인코더 디코더 모델

염성웅, 최철웅, 홍 남 퀘엣, 김경백
전남대학교
yeomsw0421@gmail.com, sentilemon02@gmail.com, quachhongnam1995@gmail.com, kyungbaekkim@jnu.ac.kr

Bi-LSTM Encoder Decoder Model for Multi-Step Traffic Prediction

Sungwoong Yeom, Chulwoong Choi, Hong-Nam Quach, Kyungbaek Kim
Dept. Artificial Intelligence Convergence, Chonnam National University.

요 약

최근 IoT 기기들의 활성화에 따라 네트워크가 복잡해지고 이를 관리하기 위해 네트워크를 동적으로 변경하는 비용이 높아짐에 따라 네트워크 자원 할당 계획, 제어 및 관리를 위한 단일 단계 트래픽 예측은 한계가 있다. 이를 위해 단일 단계 트래픽 예측 모델의 결과를 입력으로 다시 사용하는 재귀 전략을 적용함으로써 여러 단계를 예측하는 멀티 스텝 트래픽 예측으로 확장되었지만 단계가 진행됨에 따라 예측 성능이 저하되는 단점이 있다. 본 논문에서는 멀티 스텝 트래픽 예측을 위해 다중 출력 전략을 적용한 Bi-LSTM 기반 인코더 디코더 모델을 제안한다. 제안된 기법은 비정상성 트래픽과 정상성 트래픽에 대해 포괄적인 실험을 수행하였고 제안된 기법의 우수한 성능을 확인한다.

Ⅰ. 서 론

최근 IoT 기기들의 활성화로 인해 네트워크에서 관찰되는 트래픽 데이터가 폭발적으로 증가함에 따라 인터넷 서비스 제공업체는 인터넷 운영 효율성을 향상시키기 위해 정확하고 시기적절한 트래픽 흐름 정보를 사용하여 대역폭을 할당하고, 이상을 감지하고, 혼잡을 제어하는 연구가 활성화되고 있다 [1]. 이 트래픽 흐름 정보를 파악하기 위한 모니터링 및 스케줄링을 지원하는 인프라가 적기 때문에 인터넷 관리자는 장단기 트래픽 스케줄링을 수행하기 위해 트래픽에 대한 정확한 예측이 필요하다.

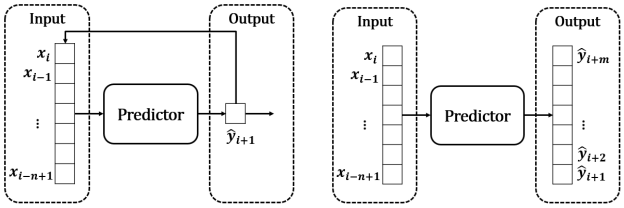
이러한 단기 트래픽 흐름을 예측하기 위해 자기 회귀 통합 이동 평균 (ARIMA) 모델과 같은 통계적 접근 방식을 연구되었다 [2]. 이 기법은 비정상성 또는 정상성과 같은 네트워크 특성에 따라 트래픽의 자기상관을 고려하여 트래픽을 예측한다. 하지만, 이러한 통계적 기법은 급격한 변화가 드러나는 비선형적 특성이 드러나는 네트워크에서는 낮은 성능을 보여준다. 최근 시계열 딥러닝 신경망의 활성화로 인해 시계열 신경망 기반 단기 네트워크 트래픽 볼륨 예측 기법은 낮은 오류율을 보이고 있다 [3]. 하지만, 대역폭, 패킷 소실 및 대기 시간 등과 같은 정보는 네트워크 상태에 따라 원하는 시간에 측정이 불가능할 수 있다. 또한, 고속 네트워크 관리에서는 네트워크의 즉각적인 변경이 비용이 많이 들거나 실행 불가능한 상황이 많다. 예를 들어 광 네트워크에서 파장 (또는 람다)을 설정하는데 필요한 시간은 종종 몇 분 정도이므로 즉시 수행할 수 없다. 이러한 트래픽 성능 측정은 단일 단계를 넘어 향후 단계의 트래픽 적절한 결정을 내릴 수 있는 충분한 시간이 필요하다.

이러한 상황에서 사전 예방적 관리에 충분한 시간을 제공하기 위해 멀티 스텝 트래픽 예측 기법이 연구되었다 [4]. 멀티 스텝 시계열 예측을 위한 두 가지 일반적인 전략은 직접 (또는 병렬) 예측 기법과 반복 (또는 순진한) 예측 기법이다. 직접적인 접근 방식에서는 서로 다른 모델이 병렬로 학습되어 미리 한 단계 앞서 예측한다. 반면에 반복적 접근 방식에서는 연속적인 단계의 출력이 다음 예측 단계의 입력으로 사용되는 반복적인 한 단계 앞서 예측을 수행하여 멀티 스텝 시계열 예측을 수행한다. 재귀적으로 한 단계 앞선 예측은 필요한 시간 범위까지 반복할 수 있으며 단일 예측 모델만 사용된다. 하지만, 이러한 재귀 기법은 이전 단계의 예측 오류 누적은 예측 범위가 증가함에 따라 증가한다.

본 논문에서는 인코더 디코더 구조의 Bi-LSTM을 기반으로한 멀티 스텝 트래픽 예측 모델을 제안한다. 이 모델은 트래픽 데이터를 활용하여 단일 단계의 트래픽뿐만 아니라 멀티 스텝 트래픽을 예측할 수 있다. 제안된 기법은 유효성을 검증하기 위해 비정상성 트래픽과 정상성 트래픽이 관찰되는 실제 DNS 요청을 기반으로 히스토리 상관관계를 고려하여 실험을 수행하여 MAPE를 평가한다.

Ⅱ. 관련 연구

트래픽 예측의 중요성으로 인해 지난 수십 년 동안 많은 트래픽 예측 기법이 제안되었다. 이 트래픽 예측 기법은 트래픽 특성에 따라 성능이 달라질 수 있다. 이 트래픽 특성은 크게 정상성과 비정상성으로 구분할 수 있다. 정상성은 시계열의 확률적인 성질들이 시간에 상관없이 변하지 않는 특성을 가지고 있으며, 비정상성은 반대로 시간에 따라 통계적 특성이 변하는 특성을 가지고 있다. 이러한 정상성과 비정상성을 고려한 접근법으로 추세와 계절성을 기반한 지수 평활 [6], 자기상관을 표현하는데 목적이 있는 ARIMA 모델 [5] 등이 포함된다. 그 중 ARIMA 모델은 트래픽 예측 모델을 구축하기위한 프레임워크로 널리 알려져 있다 [7]. 이후, 계절적 ARIMA [8] 모델과 같은 일부 개선된 방법도 단기 트래픽 예측에 사용되었다. 하지만, 네트워크 유저의 수가 증가함에 따라 네트워크 트래픽 흐름이 무작위 및 비선형으로 인해 많은 전통적인 방법이 불충분한 것으로 나타났다 [9]. 논문 [10]는 노드 간 트래픽 양을 예측하고 실제 데이터를 사용하여 모델을 학습시키기 위해 SDN에 모델을 배포하여 대용량 트래픽 예측을 위한 LSTM 기반 모델을 제시했다. 논문 [11]는 소스 측에서 관찰된 트래픽 계절성을 학습하여 LSTM 기반 네트워크 트래픽 볼륨 추정 방법을 제안했다. 논문 [12]는 네트워크 트래픽 폭주와 불확실성에 대처하기 위해 LSTM 기반의 실시간 네트워크 트래픽 예측 모델을 제안했다. 논문



(a) 재귀 전략 (b) 다중 출력 전략

그림 1. 멀티 스텝 예측 모델 구조

[13]는 GEANT 백본 네트워크 데이터를 사용하여 LSTM 신경망을 통해 트래픽의 비선형 특성을 학습하고 예측했다.

이 딥러닝 기반 트래픽 예측 기법은 다른 방법에 비해 장점을 보여주었다. 앞서 언급한 접근 방식은 단기 시퀀스를 정확하게 예측할 수 있지만 장기 예측 성능에서보다 빠르게 감쇠되기 때문에 효과적인 예측이 어려워진다. 이를 위해 자연어 처리 분야에서 자주 사용되는 심층 신경망을 사용하여 모델 입력을 고정 벡터로 인코딩한 다음 다른 심층 신경망을 사용하여 예측을 위해 고정 벡터에서 출력 시퀀스를 디코딩하는 시퀀스 대 시퀀스 모델 구조를 트래픽 예측 분야에서 활용할 필요가 있다 [14]. 이 논문에서는 고전적인 통계 및 기계 학습 방법과는 달리 시퀀스 대 시퀀스 학습 구조를 통해 멀티 스텝 예측 모델을 제안한다.

Ⅲ. Bi-LSTM 기반 멀티 스텝 트래픽 예측 기법

1. 멀티 스텝 트래픽 예측 방법론

멀티 스텝 모델의 트래픽 입력 창 슬라이드 X_i와 그라운드 트루스 Y_i는 i번째 시간 창에서 n, m 차원으로 구성되어 과거와 미래에 대한 이해를 돕는다. X_i와 Y_i는 수식 (1)과 (2) 같이 표현한다.

$$X_i = [x_{i-n+1}, x_{i-n+2}, \ldots, x_i] \tag{1}$$

$$Y_i = [y_{i+1}, y_{i+2}, \ldots, y_{i+m}] \tag{2}$$

n 개의 과거 단계를 걸쳐 m 개의 미래 단계에 대한 정확한 트래픽 추정을 목표로 하는 멀티 스텝 트래픽 예측 함수$f(\,)$는 전략에 따라 다르게 공식화할 수 있다. 일반적으로 멀티 스텝 예측을 위한 두 가지 효과적인 상태는 재귀 전략과 다중 출력 전략이다.

재귀 전략은 그림 1 (a)와 같다. 재귀 전략은 단일 단계 예측 모델에서 i번째 트래픽 창 슬라이드 X_i를 입력하여 $i+1$번째 트래픽 $\hat{y}_{i+1}$을 출력한다. 트래픽 $\hat{y}_{i+1}$에 대한 추정은 수식 (3)과 같이 표현한다.

$$\hat{y}_{i+1} = f(X_i) = f(x_{i-n+1}, x_{i-n+2}, \ldots, x_i) \tag{3}$$

예측된 트래픽 $\hat{y}_{i+1}$은 다음 예측을 위한 입력으로 구성되며, 이를 m번 반복한다. 이 전략은 트래픽의 동적 특성을 고려한다. 그러나 예측된 결과가 입력으로 사용되기 때문에 예측을 반복할수록 오류가 축적된다. 즉, 모델의 예측 성능은 단계 수가 증가함에 따라 저하되고 예측은 실제 관찰에서 조금씩 벗어나게 된다.

다중 출력 전략은 그림 1 (b)와 같다. 다중 출력 모델에서 i번째 트래픽 창 슬라이드 X_i를 입력하여 i번째부터 $i+m$ 번째까지의 트래픽으로

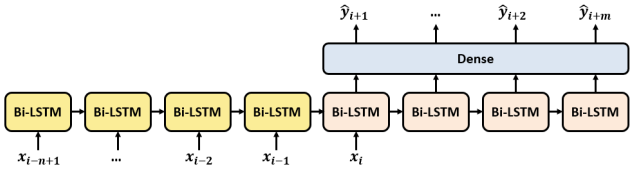


그림 2. Bi-LSTM 기반 인코더 디코더 트래픽 예측 모델

구성된 트래픽 창 슬라이드 $\hat{Y}_i$를 출력함으로써 멀티 스텝 예측을 달성한다. 트래픽 출력 창 슬라이드 $\hat{Y}_i$에 대한 추정은 수식 (4)와 (5) 같이 표현한다.

$$\hat{Y}_i = f(X_i) = f(x_{i-n+1}, x_{i-n+2}, \ldots, x_i) \tag{4}$$

$$\hat{Y}_i = [\hat{y}_{i+1}, \hat{y}_{i+2}, \ldots, \hat{y}_{i+m}] \tag{5}$$

이 전략은 오류 누적을 다소 피할 수 있지만 데이터의 동적 특성을 무시하고 시계열 프로브를 정적 문제로 취급하여 예측의 정확도를 떨어뜨릴 수 있다.

2. 제안된 다중 출력 트래픽 예측 기법

제안된 접근 방식은 네트워크 트래픽을 $i+1$ 번째 시간 창부터 $i+m$ 번째 시간 창까지 비교적 정확하게 단계적으로 추정하는 기법이며, 이 추정을 위해 시계열 신경망인 Bi-LSTM을 인코더 디코더 형태로 사용한다. 그림 2는 제안된 LSTM 기반 인코더 디코더 트래픽 예측 모델을 보여준다. 이 모델은 1개의 Bi-LSTM 레이어로 구성된 인코더와 디코더와 1개의 Dense 레이어로 구성된다. 이 Dense 레이어의 활성화 함수는 선형 함수를 사용한다. 인코더와 디코더는 각각 10개의 노드가 포함된다.

제안된 Bi-LSTM 기반 인코더 디코더 모델의 입력벡터는 관찰된 트래픽 볼륨 s_i와 시간 창 인덱스 t_i로 구성된다. i번째 시간 창에서 구성된 입력벡터는 $x_i = (s_i, t_i)$로 표현한다. 제안된 모델의 학습은 $i-9$ 번째 시간 창부터 i 번째 시간 창까지의 입력 벡터 $x_{i-n+1}, \ldots, x_{i-1}, x_i$와 $i+1$ 번째 시간 창부터 $i+10$ 번째 시간 창까지의 그라운드 트루스 $y_{i+1}, y_{i+2}, \ldots, y_{i+10}$을 사용한다. 제안된 모델의 예측은 $i-9$ 번째 시간 창부터 i 번째 시간 창까지의 입력 벡터 $x_{i-n+1}, \ldots, x_{i-1}, x_i$를 입력으로 사용하여 연속적인 트래픽 $\hat{y}_{i+1}, \hat{y}_{i+2}, \ldots, \hat{y}_{i+10}$을 예측한다.

제안된 모델은 정상성 트래픽과 비정상성 트래픽에 의해 영향을 받을 수 있다. 입력 벡터의 첫 번째 도메인은 트래픽 볼륨으로 정상성과 비정상성을 고려하여 두 가지 전처리를 적용할 수 있다. 첫 번째 전처리 방식은 계절성과 같은 비정상성 트래픽을 고려한 정규화이며, 이는 Bi-LSTM에서 기울기 폭발 또는 손실이 발생하지 않도록 한다. 두 번째 전처리 방식은 편차가 거의 없는 정상성 트래픽을 고려한 트래픽 볼륨을 트래픽 변화율로 바꾸는 것이다. 입력 벡터의 두 번째 도메인은 트래픽 볼륨에 대한 시간 창 인덱스이다. 이 논문에서는 i번째 시간 창에서 관찰된 트래픽과 $i-1$번째 시간 창 사이 간격을 5분으로 설정하며, 하루동안 관찰된 트래픽의 시간 창 인덱스의 최대 길이를 288로 설정한다. 따라서 시간 창 인덱스 t_i의 값은 1부터 288가 된다. 이 시간 창 인덱스 또한 Bi-LSTM에서 기울기 폭발 또는 손실이 발생하지 않도록 하기 위해 정규화를 적용한다.

이와 같이 다중 출력 전략을 고려한 제안된 Bi-LSTM 기반 인코더 디코더 모델은 멀티 스텝 트래픽 예측의 전반적인 성능을 향상시킬 수 있다.

Ⅳ. 실험 및 검증

제안된 Bi-LSTM 기반 인코더 디코더 모델 (Bi-LSTM_S2S)을 평가하기 위해 LSTM 기반 인코더 디코더 모델 (LSTM_S2S), Bi-LSTM 기반 회귀 모델 (Bi-LSTM_RE) 그리고 LSTM 기반 회귀 모델 (LSTM_RE)와 비교한다. 제안된 기법을 평가하기위해 ICANN에서 운영하는 DNS-STAT: Hedgehog의 데이터 셋 중 중국 상하이에서 한 달 (2020년 9월 01일 - 2020년 9월 31일) 동안 수집된 실제 DNS 요청 트래픽을 사용한다. 데이터 세트의 첫 20일에 대한 트래픽은 시계열 딥러닝 모델의 학습에 사용하며, 데이터 세트의 다음 10일은 테스트에 사용한다. 제안된 기법은 예측값에 대한 오차율을 측정하기 위해 가장 보편적으로 사용되는 MAPE를 사용한다.

이 멀티 스텝 트래픽 기법은 트래픽에서 관찰되는 정상성이나 비정상성과 같은 특성에 의해 영향을 받을 수 있다. 또한, 트래픽 시간 창 사이의 상관관계에 따라 성능이 영향을 받을 수 있다. 이를 위해 비정상성이 관찰되는 상하이 DNS 요청 트래픽을 전처리를 하고, 정상성 트래픽과 비정상성 트래픽에 대해 트래픽 시간 창 간의 상관관계를 히트맵을 사용하여 시각화할 필요가 있다. 첫 번째 전처리는 비정상성 트래픽을 그대로 사용하기 위한 정규화이고, 이 전처리를 사용하는 기법은 M1으로 표현한다. 두 번째 전처리는 비정상성 트래픽을 정상성화하기 위해 트래픽을 변화율로 변환하고, 이 전처리를 사용하는 기법은 M2로 표현한다. 우리는 각각의 전처리에 따라 멀티 스탭 트래픽 예측 기법들의 성능을 비교할 필요가 있다.

그림 3은 정규화를 통해 전처리된 비정상성 트래픽 시간 창 간의 상관관계를 히트맵으로 시각화하였다. 히트맵의 색상이 어두울수록 해당 기간 동안 상관 계수가 높아진다. 이 비정상성 트래픽에서는 시간 창 사이의 스텝 간격이 좁아질수록 양의 상관계수의 값이 상승하는 것을 확인할 수 있다. 이에 따라, 멀티 스텝 트래픽 기법은 다음 단계의 예측을 위해 이전 단계의 출력이 모델의 입력으로 사용하기 때문에 이전 단계의 결과에 영향을 받을 수 있다. 하지만, 이전 단계의 결과에 대한 오류가 섞여 입력으로 다시 사용된다면 단계가 진행됨에 따라 오류는 점점 더 쌓이게 되어 성능이 점점 낮아질 수 있다.

그림 4는 트래픽에서 비정상성이 관찰될 때 재귀 기법과 다중 출력 기법이 각 단계의 예측한 결과에 대한 MAPE를 보여준다. 제안된 Bi-LSTM_S2S_M1 모델은 Bi-LSTM_RE_M1 모델에 비해 모든 단계에서 MAPE가 약 3% 낮다. 다수의 결과를 한꺼번에 출력하는 인코더 디코더 모델은 기존의 재귀적 단일 스텝 예측 모듈에 비해 장기적 특징을 학습하기 때문에 전반적으로 낮은 에러율을 보여준다.

그림 5는 트래픽 변화율 변환을 통해 전처리된 정상성 트래픽 시간 창 간의 상관관계를 히트맵으로 시각화하였다. 이 정상성 트래픽에서는 전반적으로 변화가 두드러지지 않아 어떤 사이 간격의 시간 창에서도 상관계수가 낮다. 단계가 진행됨에 따라 오류가 쌓임에도 불구하고 많은 영향을 받지 않을 수 있다.

그림 6는 정상성 트래픽를 재귀 기법과 다중 출력 기법이 각 단계의 예측한 결과에 대한 MAPE를 보여준다. 제안된 Bi-LSTM_S2S_M2 모델은 LSTM_RE_M2 모델을 제외한 나머지 모델과 비슷한 성능을 보여준다.

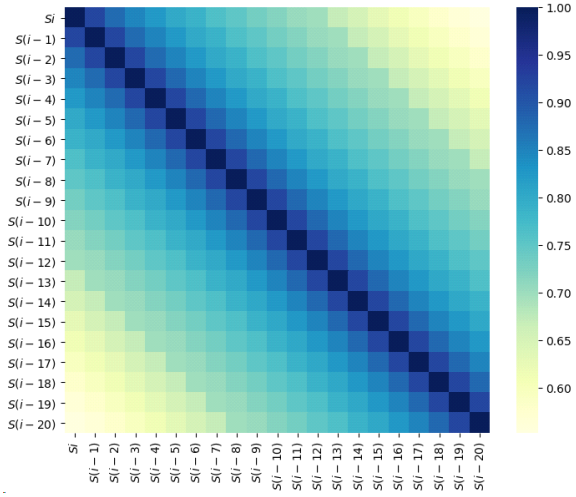


그림 3. 비정상성 트래픽 시간 창 간의 상관관계에 대한 히트맵 시각화

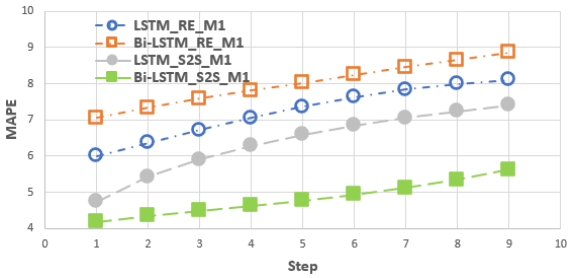


그림 4. 비정상성 트래픽에 대한 10 단계 앞선 예측

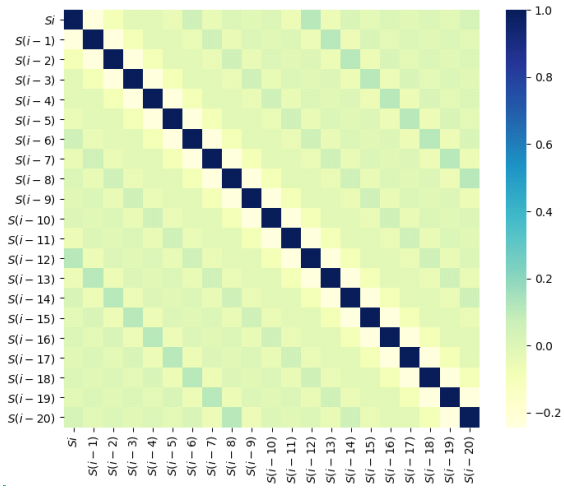


그림 5. 정상성 트래픽 시간 창 간의 상관관계에 대한 히트맵 시각화

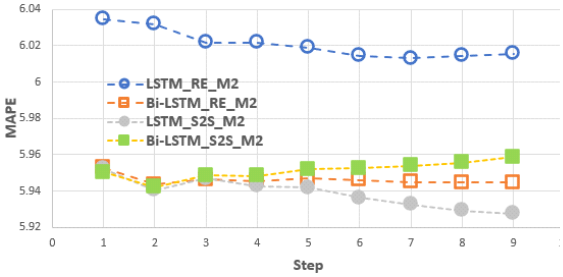


그림 6. 정상성 트래픽에 대한 10 단계 앞선 예측

LSTM_RE_M2은 높은 에러율의 첫 스텝 결과를 회귀하여서 이후 스텝의 에러율은 비교적 높은 에러율을 보여준다.

V. 결 론

본 논문에서는 멀티 스텝 트래픽 예측을 위해 다중 출력 전략을 적용한 Bi-LSTM 기반 인코더 디코더 모델을 제안한다. 실제 DNS 네트워크 트래픽 기반 평가를 통해 제안된 기법이 비정상성 관찰되는 트래픽에 대해 성능을 향상시키는데 효과적이다. 또한, 제안된 기법은 정상성이 관찰되는 트래픽에서 이전의 접근 방식과 유사한 성능을 달성하였다. 차후에 우리는 전반적인 네트워크 트래픽 흐름을 파악하기 위해 그래프 신경망을 네트워크 트래픽 예측 모델에 적용할 계획이다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2021-2016-0-00314*). 본 논문은 2021년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과입니다.

참 고 문 헌

[1] Bhuyan, Monowar H., Dhruba Kumar Bhattacharyya, and Jugal K. Kalita. "Network anomaly detection: methods, systems and tools." Ieee communications surveys & tutorials 16.1 (2013): 303-336.

[2] T. Alghamdi, K. Elgazzar, M. Bayoumi, T. Sharaf and S. Shah, "Forecasting Traffic Congestion Using ARIMA Modeling," 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019, pp. 1227-1232, doi: 10.1109/IWCMC.2019.8766698.

[3] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2017. Applying deep learning approaches for network traffic prediction. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2353 - 2358.

[4] Zang, Chuanyun. "Deep Learning in Multiple Multistep Time Series Prediction." arXiv preprint arXiv:1710.04373 (2017).

[5] Haviluddin, Haviluddin, and Rayner Alfred. "Forecasting network activities using ARIMA method." (2014).

[6] Chan, Kit Yan, et al. "Traffic flow forecasting neural networks based on exponential smoothing method." 2011 6th IEEE Conference on Industrial Electronics and Applications. IEEE, 2011.

[7] Rutka, G. "Network traffic prediction using ARIMA and neural networks models." Elektronika ir Elektrotechnika 84.4 (2008): 53-58.

[8] Shu, Yantai, et al. "Wireless traffic modeling and prediction using seasonal ARIMA models." IEICE transactions on communications 88.10 (2005): 3992-3999.

[9] Somenath Mukherjee, Rajdeep Ray, Rajkumar Samanta, Mofazzal H Khondekar, and Goutam Sanyal. 2017. Nonlinearity and chaos in wireless network traffic. Chaos, Solitons & Fractals 96 (2017), 23 - 29.

[10] A. Azzouni and G. Pujolle. 2018. NeuTM: A neural network-based framework for traffic matrix prediction in SDN. In NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. 1 - 5.

[11] Giang-Truong Nguyen, Van-Quyet Nguyen, Huu-Duy Nguyen, and Kyungbaek Kim. 2018. LSTM based Network Traffic Volume Prediction. In Proceedings of 2018 KIPS Spring Conference.

[12] H. Lu and F. Yang. 2018. Research on Network Traffic Prediction Based on Long Short-Term Memory Neural Network. In 2018 IEEE 4th International Conference on Computer and Communications (ICCC). 1109 - 1113.

[13] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2017. Applying deep learning approaches for network traffic prediction. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2353 - 2358.

[14] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." arXiv preprint arXiv:1409.3215 (2014).

뉴스 데이터를 이용한 머신러닝 기반 비트코인 가격 등락 예측

강민규,김보선,신희종
고려대학교
{cxz3619,boseon12,tlswhd0215}@korea.ac.kr

Machine Learning based Prediction of Bitcoin Price Fluctuation Using News Data

MinGyu Gang, Boseon Kim, Hee-Jong Shin
Korea Univ

요 약

비트코인이 발표된 후, 비트코인은 현재 두 번째 열풍을 맞고 있다. 이러한 열풍과 관심에 비례하여 늘어나는 소셜 미디어, 뉴스 기사들은 많은 연구자들의 연구대상이 되었다. 기존의 연구들은 소셜 미디어와 뉴스 기사를 활용하여 비트코인의 가격 예측을 시도했으며, 많은 연구들이 감성분석을 이용했다. 감성분석을 이용한 방법은 감성분석 시 많은 데이터가 손실되고 감성 분석 사전이 비트코인 분야의 감성을 잘 읽지 못할 가능성이 있다. 본 논문은 감성분석 대신 뉴스 기사의 TF-IDF(Term Frequency-Inverse Document Frequency) 벡터 값과 뉴스 기사의 수를 이용해 다음날의 비트코인 종가를 예측한다. 예측 과정에서 8개 모델 간의 성능비교와 가격 데이터만 쓴 경우와 뉴스 데이터와 가격데이터를 함께 쓴 경우를 비교한다.

Ⅰ. 서 론

비트코인은 2008년 10월 31일 사토시 나카모토가 발표한 최초의 블록체인 기반의 탈중앙화 암호 화폐이다. 비트코인이 발표 된 후 비트코인의 거래량과 거래 가격은 점점 증가하여 2018년 1월 정점을 찍은 후, 2021년 3월에 다시 엄청난 열풍을 불러일으키고 있다. 이러한 열풍에 비례하는 관심만큼 소셜 미디어의 트윗과 비트코인 관련 기사 또한 증가하고 있다. 늘어가는 텍스트 데이터들을 활용하기 위해, 소셜 미디어나 비트코인 관련 기사들을 이용해 비트코인의 가격을 예측하는 많은 선행연구들이 수행되었다.

[1]은 뉴스 데이터를 Textblob 패키지를 이용해 감성분석 한 후, LSTM(Long Short-Term Memory)모델을 써서 비트코인의 가격등락을 예측했다. [2]는 Word2Vec을 이용해 주가 지수 사전을 구축한 후, 구축된 사전을 바탕으로 감성분석의 결과 값과 주가지수의 방향성을 비교했다. [3]은 뉴스 데이터를 CountVectorizer를 이용해 백터화 한 후,로지스틱 회귀,SVM(Support Vector Machine),나이브 베이즈 등의 분류기를 통해 비트코인, 이더리움, 라이트코인의 가격 변동을 예측했다.

뉴스 데이터를 이용해 주가를 예측하는 방법에는 주로 감성분석을 통해 뉴스 데이터를 수치화 하거나 분류한 후, 가격예측의 feature로 활용 한다. 하지만 뉴스 데이터를 감성분석 하는 경우, 많은 데이터의 손실이 있고 감성분석 사전이 비트코인 분야의 감성을 잘 분석하지 못할 가능성이 있다. 따라서 본 논문은 뉴스데이터를 감성분석하지 않고 TF-IDF 알고리즘을 이용해 나온 벡터와 뉴스 기사 수를 이용한 비트코인 가격 등락 예측 방법을 제안한다.

본 논문의 구성은 서론에 이어, 본론에서 데이터 수집, 데이터 전처리, 실험, 실험결과에 대해 설명하고, 마지막으로 결론 및 향후연구에 대해 언급한다.

Ⅱ. 본론

A.데이터 수집

[1]의 뉴스 순위에 따라 2013년에서 2020년 사이에 비트코인 관련 기사를 가장 많이 쓴 상위 4개의 언론사에서 기사를 수집했다. 수집된 기사들은 언론사, 날짜 ,제목 ,본문 4개의 열로 이루어져있으며 그림1과 같다.

	Press	Date	Title	Paragraph
0	BitcoinNews	2016-7-1	NewdatashowsChineseexchangeshaveaccountedfor42...	In the New York Times, writer Nathaniel Popper...
1	BitcoinNews	2016-7-1	OTCTradingForRegularPeopleSearchFortheUberofBi...	More and more people are looking to get starte...
2	BitcoinNews	2016-7-1	BitcoinContestTellIdeapodHowBlockchainCanChang...	The two influential thinkers, Don and Alex Tap...

그림 1.수집된 기사의 형태

B.데이터 전처리

기사들을 벡터화하기 앞서, 기사들은 불용어와 온점(“.”)등을 포함하고 있고 같은 형태의 복수형 또한 다른 단어로 인식하기에 벡터화 하는 것에 문제가 된다. 따라서 파이썬 *nltk* 패키지의 *word_tokenize*와 *stopword*를 활용해 토큰화 시킨 후, 불용어를 제거했다.

다음으로, 우리는 텍스트 데이터인 뉴스기사를 TF-IDF를 이용하여 뉴스 기사의 특징 벡터를 추출한다. TF-IDF는 정보 검색론 분야에서 사용하는 가중치를 구하는 알고리즘으로, 단어 빈도 수를 기반으로 가중치를 구한다.

$$tfidf(t,d,N) = tf(t,d)*idf(t,d) \quad (1)$$

$$idf(t,d) = \log\frac{N}{1+df(t)} \quad (2)$$

가중치인 $tfidf(t,d,N)$는 단어 빈도의 가중치인 $tf(t,d)$와 문서의 역 빈도 가중치인 $idf(t,d)$의 곱으로 이루어져 있다.$tf(t,d)$는 특정 문서 d에서 특정 단어 t의 등장 횟수를 의미하며, $idf(t,d)$는 특정 단어 t가 등장한 문서의 수의 역수 형태이다. $tfidf(t,d,N)$ 가중치는 단어의 빈도수가 높고, 전체 문장에서 적게 등장할수록 높아진다.

하루 단위로 합쳐진 기사의 제목과 본문에 각각 TF-IDF 알고리즘을 적용해 벡터화시켰으며 하루 단위의 기사 수 또한 추출했다. 추출된 데이터는 3개의 열[뉴스 기사 수, 제목의 TF-IDF값, 본문의 TF-iDF값]이며 행

은 950개의 일자이다. 데이터 예시는 그림2와 같다.

	date	num_news	paragraph_tfidf	title_tfidf
0	2018-08-01	15	[0, 0, 0, 0.287682, 0, 0.693147, 0.287682, 0, ...	[0, 0.182612, 0, 0.157864, 0, 0, 0, 0, 0.28768...
1	2018-08-02	32	[0, 0, 0.191147, 0, 0.575364, 0, 0, 0, 0.69314...	[0, 0.693147, 0.693147, 0, 0.575364, 0, 0, 0, ...
2	2018-08-03	43	[0, 0.693147, 0.693147, 0, 0.575364, 0, 0, 0, ...	[0, 0, 0, 0.287682, 0, 0.693147, 0.287682, 0, ...

그림 2. 데이터 전처리 결과

C.실험

본 절에서는 머신러닝 분류 분야에서 주로 쓰이는 8개의 분류모델을 이용했으며 *python*의 *scikit-learn* 라이브러리를 통해 제공되는 모델들과 파라미터들을 이용해 실험을 진행했다. 사용한 분류 모델들은 다음과 같다.

-로지스틱 회귀

로지스틱 회귀는 로지스틱 함수를 사용하여 데이터가 어느 1과0 사이의 값으로 예측하고 확률에 따라 가능성이 더 높은 범주로 분류해주는 모델이다.

-나이브 베이즈

나이브 베이즈는 조건부 확률을 이용한 나이브 베이즈 정리를 기반으로 하는 지도학습 알고리즘이며 분류하려는 대상의 긱 확률을 측정하여 분류하는 모델이다.

-KNN(K-Nearest Neighbors)

KNN모델은 새로운 데이터가 주어졌을 때, 기존 데이터 중 가장 가까운 k의 데이터를 바탕으로 새로운 데이터를 예측하는 모델이다.

-SVM(Suport Vector machine)

SVM 모델은 데이터 군들로부터 가장 먼 분류 경계면을 찾고 경계면을 기반으로 새로운 점이 어느 쪽에 확인하여 분류하는 모델이다.

-랜덤 포레스트

랜덤 포레스트는 여러 개의 피쳐 중 랜덤한 수의 피처를 선택한 후, 다수의 결정트리들을 학습하는 앙상블 기법을 이용한 모델이다.

-Extra Tree

Extra Tree는 랜덤 포레스트와 유사한 결정 트리이지만, 랜덤 포레스트와 다르게 피처 또한 무작위로 분할한 후 최상의 분할을 선택하는 앙상블 기법을 이용한 모델이다.

-AdaBoost

AdaBoost는 Boosting 알고리즘의 한 종류로서, 이전 약한 학습기 결과의 오차에 가중치를 두어 다음 약한 학습기의 입력으로 주어 학습하는 Boosting 모델이다.

-XGBoost

XGBoost는 Gradient Boosting을 개선시킨 알고리즘으로, 경사하강법과 병렬 처리 기법을 사용해 과적합 문제를 해결하고, 처리시간을 개선시킨 모델이다.

실험은 8개의 모델에 입력 값을 다르게 주는 방식으로 두 번에 걸쳐 비교했다. 첫 번째 입력 값은 ['전날의 종가']만을 넣은 방식이고, 두 번째 입력 값은 첫 번째 입력 값에 뉴스 데이터('기사 수','제목의 TF-IDF 값','본문의 TF-IDF 값')을 추가한 값이다. 이것은 뉴스 데이터를 이용한 예측이 가격 데이터만 이용했을 때와 비교해보기 위함이다.

D.실험 결과

모델 평가 결과, 입력 값으로 전날의 종가만을 준 모델들은 40% 후반 대에서 50% 후반까지 고르게 분포한 모습을 보여주었으며, XGBoost 모델이 가장 좋은 정확도인 60%를 보여주었다. 또한 입력 값으로 가격 데이터와 함께 뉴스 데이터(기사 수,제목의 TF-IDF값,본문의 TF-IDF값)를 넣어 주었을 때 나이브 베이즈를 제외한 모든 모델에서 precision, recall, f1-score, accuracy가 소폭 상승하거나 유지했다. 상승치가 없는 모델의 경우, 다른 라벨에 대해 precison과 recall이 0으로 잘 학습되지 못함 모델들이였고, 모든 수치가 크게 하락한 나이브 베이즈 모델 또한 잘 학습하지 못한 모델이었다. 등락을 잘 예측한 모델 중, accuracy를 기준으로 가장 큰 폭으로 상승한 모델은 랜덤 포레스트 모델이며, 가장 높은 accuracy를 기록한 모델은 62%를 기록한 adaBoost 모델이다.

자세한 수치는 표 3에 표시되며, 각각의 왼쪽 수치는 가격 정보만을 사용했을 때 수치이고, 오른쪽의 수치는 가격 정보와 뉴스 데이터를 함께 썼을 때 변화 폭이다.

	precision		recall		f1-score		accuracy	
로지스틱 회귀	58%	-	100%	-	73%	-	58%	-
나이브 베이즈	55%	-3%	100%	-46 %	73%	-	58%	-10 %
KNN	59%	**+5%**	56%	**+6%**	58%	**+5%**	52%	**+5%**
SVM	58%	-	100%	-	73%	-	58%	-
랜덤 포레스트	55%	**+5%**	51%	**+9%**	53%	**+8%**	48%	**+6%**
Extra Tree	55%	**+2%**	51%	**+3%**	53%	**+3%**	47%	**+3%**
AdaBoost	61%	**+2%**	85%	-	71%	**+1%**	59%	**+3%**
XGBoost	62%	-1%	79%	**+6%**	69%	**+1%**	60%	**+1%**

표 3.모델들의 성능 정보

Ⅲ. 결론

본 논문에서는 뉴스 데이터를 이용해 비트코인 가격의 등락을 예측하는 방법을 제시했으며, 8개 모델간의 성능을 비교하여 평가했다. 또한 정보만 쓰는 것 보다 뉴스 정보를 함께 사용하는 것이 비트코인의 등락을 예측하는데 도움이 된다는 것을 보여주었다. 그러나 모델들의 성능 향상에 있어 뉴스데이터의 존재는 그리 크지 않은 수준이다. 따라서 향후 연구로는 뉴스 데이터에서 조회 수, 감성분석 결과 등 더 많은 종류의 뉴스 데이터를 추가하고 모델의 최적화를 통해 성능을 고도화할 계획이다.

ACKNOWLEDGMENT

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2018R1D1A1B07045742)과 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발)

참 고 문 헌

[1]강민규, 김보선, 신무곤, 백의준, 김명섭, "LSTM 기반 감성분석을 이용한 비트코인 가격 등락 예측" 2020년도 한국통신학회 추계종합학술발표회, 온라인 개최, Nov. 13, 2020, pp. 1-2

[2] 김다예 · 이영인. "Word2Vec을 활용한 뉴스 기반 주가지수 방향성 예측용 감성 사전 구축" ,한국빅데이터학회지 제3권 제1호, 2018, pp. 13-20

[3] Connor Lamon, Eric Nielsen, Eric Redondo. "Cryptocurrency Price Prediction Using News and Social Media Sentiment" ,http://cs229.stanford.edu/proj2017/final-reports/5237280.pdf

배터리 디지털 트윈을 위한 IoT 시스템 아키텍처

아르디, 김영현, 최덕재
전남 대학교

ardi@ejnu.net, zero9231@naver.com, dchoi@jnu.ac.kr

IoT System Architecture for Battery Digital Twin

Ardiansyah, Yeonghyeon Kim, Deokjai Choi
Chonnam National University

Abstract

The digital twin concept may be used during the design, manufacturing, and operation phases of a smart energy device. In the case of battery energy storage systems (BESS), the adoption of a digital twin in the pioneering works has shown the potential benefits of reducing the BESS ownership cost and improving its performance in both stationary and vehicular applications. However, the existing battery digital twin system architecture is designed with several significant drawbacks. It does not have an interoperability mechanism for multi-protocol communication, and the co-simulation interfaces to interact with BESS's digital counterpart were not implemented. To tackle these problems, in this paper, we design a battery digital twin system architecture applying the Open Platform Communications Unified Architecture (OPC UA) internet of things (IoT) data exchange protocol. We also formulate our system architecture with the co-simulation interfaces for data sharing and collaboration, then discuss some security and privacy-preserving challenges while using this interface in a 5G mobile edge computing (MEC) environment. We believe that the work presented in this paper will aid in realizing reliable and resilient battery digital twin for intelligent and interconnected battery management in the future.

I . INTRODUCTION

The digital twin is a digital replica of the physical assets that equipped with three following characteristics: 1) the ability to simulate system lifecycle, 2) real-time synchronization with the physical asset, and 3) active data acquisition [1]. In smart energy systems, the digital twin concept can be used during the design, manufacturing, and operation phases of a smart energy device or a smart grid process. The goal is to understand, predict, optimize, and control the energy system assets based on their models and real-time data in virtual space.

In a battery energy storage system (BESS), the digital twin's adoption in the pioneering works [2,3,4] has shown several potential benefits that can be divided into four aspects, i.e., monitoring and diagnostics, lifetime prognostics, fault detection and prediction, and evaluation and optimization. These benefits will help to reduce the total cost of BESS management and increase its performance. However, the works mentioned above suffer from several significant drawbacks as follows. Firstly, the developed system architectures did not have an interoperability mechanism to handle actuator and sensor devices with different internet of things (IoT) communication protocols. Secondly, it did not have its own intelligence to automatically reacts to environmental changes. Lastly, co-simulation interfaces to learn from the inputs or other digital twins were also not implemented. These features are essential in the industry 4.0 context [5], including smart grid applications.

This paper aims to bridge the aforementioned research gap mentioned above and describe our initial work that focuses on designing IoT system architecture for battery digital twin. The main contribution of this paper is as follows:

- We design an IoT system architecture that supports service-oriented architecture, extensible and comprehensive information modeling. The Open Platform Communications Unified Architecture (OPC UA) data exchange protocol [6] is embedded in the proposed IoT gateway to exchange the structured and unified data from devices with different communication protocols.
- We also formulate the higher level of the proposed architecture with the co-simulation interfaces to interact with the BESS's digital counterpart in the 5G mobile edge computing (MEC) environment. Several challenges for secure and privacy-preserving data collaboration and sharing are also discussed.

The rest of this paper is organized as follows. In the next section, we provide a brief description of the battery digital twin. Then, section III presents our proposed architecture and discusses several security challenges. Finally, section IV concludes this paper.

Ⅱ. BATTERY DIGITAL TWIN

In recent years, BESS has emerged as one of the most popular energy storage systems, both in stationary and vehicular applications. In stationary applications, BESS is one of the most flexible elements that can handle real-time power grid operation's stability challenges, particularly in the condition of gradual replacement of dispatchable fossil-fuel-based power plants by intermittent renewable power generation. Therefore, as depicted in the PJM Interconnection LLC market [7], BESS participation in the grid ancillary service markets is increasing rapidly, from zero in 2005 to over 280 MW in 2017.

BESS participation in the ancillary service market requires a strategy bound to various decision-making activities to find the trade-off between BESS operation's benefit and cost in pay-for-performance-based schemes. The optimal decision is dependent on the forecast uncertainties of coupled resources and corresponding stochastic processes that predetermine the development of the BESS state from time to time. On the other hand, BESS is one of the most important and expensive components in battery electric vehicles (BEV) [4]. The cost of BESS is approximately 35% of the BEV total cost. Therefore, for its efficient utilization, the need for long battery life is indispensable, which depends on its operational strategy. Similar to BESS in stationary applications, developing an optimal BESS operation strategy for vehicular applications requires knowledge of the current battery state, determined by numerous factors.

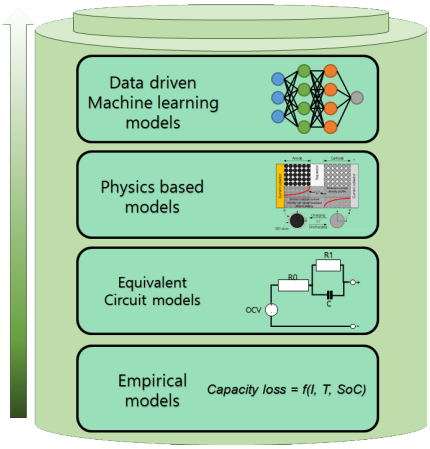


Figure 1. Evolution of Battery Modeling Approaches

Over the years, battery state modeling approaches have evolved from an empirical approach towards more data-driven techniques, as depicted in Figure 1. Battery models based on equivalent circuits are preferred for system-level development and control applications due to their relative simplicity. Engineers use equivalent circuits to model the thermo-electric behavior of batteries, parameterizing their nonlinear elements with correlation techniques that combine models and experimental measurements via optimization. For example, open-circuit voltage (OCV) measurement combined with current integration may give reasonable state of charge (SOC) estimation accuracy. However, it requires a lot of time to estimate the battery state and hence difficult to be utilized in real-world applications. On the other hand, a physics-based model describes battery behavior accurately, including the electrochemical (e.g., solid-electrolyte interphase (SEI) thickness) degradation process. The challenge, however, also takes minutes to solve, and enabling equipment for this measurement is currently quite expensive.

Table 1. Example of machine learning-based battery modeling and the required IoT sensor data

Modeling	Required Data
SOC and SOH estimation	Time, DC Current, DC Power, Temperature, DC Voltage, Charging Status
SOC and SOH prediction/forecasting	Time, SOC estimation, SOH estimation
Battery anomaly detection	Time, DC Voltage, DC Current, DC Temperature, DC Power
Optimization of Battery to Grid Integration	SOC estimation, SOC prediction, SOH estimation, SOH prediction, regulation signal, capacity bidding, time

Recently, with the wealth of data, machine learning (ML)-based approaches have received renewed attention in developing data-driven estimation and forecasting techniques of battery state models, including the SOC, state of health (SOH), the state of available power (SOAP), and the remaining useful lifetime (RUL) [3]. Table 1 depicted several machine learning-based battery state modeling and the required IoT sensor data. However, with the increase of battery cell number and algorithm complexity, onboard battery management system (BMS) is faced with problems in data storage and computational power for precise and accurate estimation and forecasting of the battery's states.

The digital twin concept has been adopted to overcome the problem mentioned above. Adopting the digital twin concept, [2] developed a cloud BMS for BESS in stationary applications, in which BMS is enhanced with cloud computing and IoT technologies. Utilizing the build-up digital twin for battery systems, the computation and data storage capabilities increase exponentially. All battery-relevant data can be measured and transmitted seamlessly to a cloud platform via an IoT gateway. However, the proposed battery digital twin appears with a lack of technical detail on the IoT technologies used for BESS sensing and actuating. Furthermore, the developed IoT system architecture does not have an interoperability mechanism for devices' connectivity with different protocols.

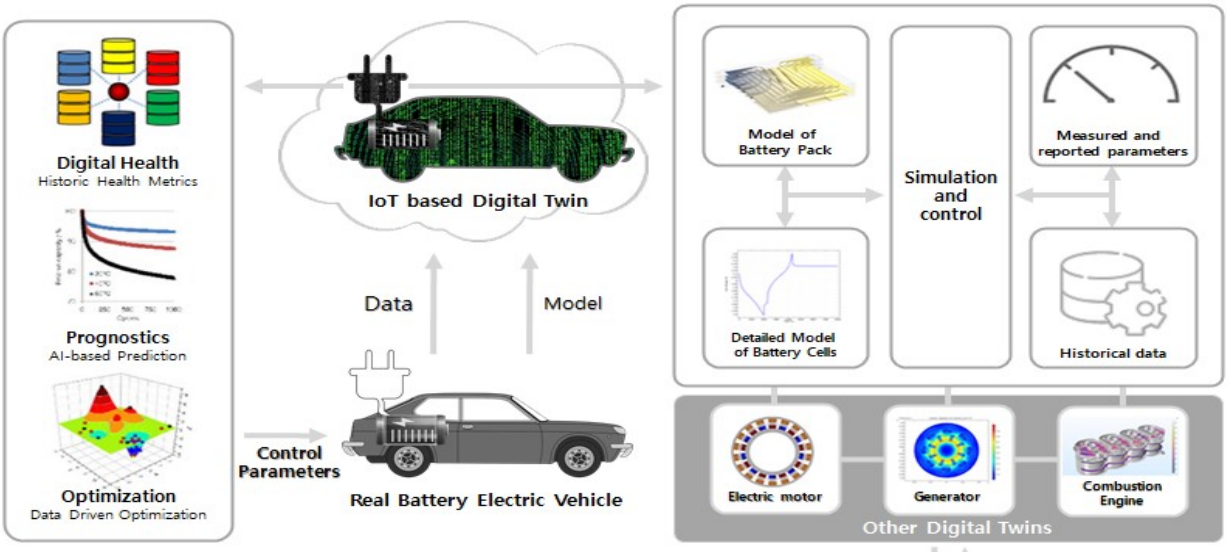


Figure 2. Battery Digital Twin in BEV.

On the other hand, [3-4] introduced the battery digital twin for BESS in vehicular applications. In this case, the battery digital twin may be a multiphysics and multiscale system model that contains historical data for the specific battery pack. It may also include data from other battery packs of the same model. Using the data-driven machine learning models, the measured and reported data could be used to produce a very accurate representation of the conditions inside the battery pack. Furthermore, the models can then be used to make predictions about the battery pack's operation and automatically compute control parameters for battery management. In most cases, the battery digital twin in BEV is not an isolated system. They are part of a more extensive system that may also co-work with other devices and their digital twins, such as the electric motor, the generator, and the combustion engine components. Therefore, the need for secure co-simulation interfaces is also indispensable. In general, the concept of battery digital twin in BEV is depicted in Figure 2.

Ⅲ. PROPOSED IOT SYSTEM ARCHITECTURE

As depicted in Figure 3, IoT technologies play an essential role in battery digital twin applications, enabling sensed data collection through smart devices embedded in BESS. However, actuator and sensor devices may communicate using industrial protocols such as programmable logic controller (PLC) or wireless communication protocols like Wireless Fidelity (Wi-Fi), Bluetooth Low Energy (BLE), and ZigBee. Therefore, to solve this problem, the use of OPC UA that enables interoperability among devices with different protocols is the best option. At its core, OPC UA defines

- A client-server communication protocol built upon TCP, HTTP, or SOAP that defines the exchange of messages via sessions on top of secure communication channels. Also, a type of system for protocol messages with a binary and XML-based encoding scheme.
- A meta-model for information modeling that combines object orientation with semantic relations. The relation between OPC UA and Asset Administration Shell (AAS) meta-model can be seen in [8].
- OPC UA can be built on existing Internet protocol models such as OSI and TCP/IP. For low-power applications, OPC UA's layering over constrained application protocol (COAP) is depicted in [9].

Following services can be provided with OPC-UA and IoT device integration: 1) real-time data access (RDA), e.g., reading, writing, monitoring variables of BESS from time to time, 2) alarm and events (AE), e.g., perform events management and inform the BESS users about these events, 3) historical record access (HRA), e.g., provide query methods for reading the BESS historical data from the archive for analysis. In order to interconnect the actuator and sensor devices with different protocols, a dedicated IoT Gateway with OPC UA supported is used. This gateway acts as a protocol translator, enabling device communication without recognizing the differences among used protocols.

For battery digital twin computing needs, the 5G MEC system placed at the wireless network's edge is used as an alternative technology to provide cloud computing resources and capabilities. This can tackle the need for big data storage and high computational power for battery digital twin, but with better reliability and lower latency compared than the traditional cloud computing systems. Furthermore, to enable data-sharing and collaboration with other digital twins, we also add the co-simulation interfaces in each digital twin instance. However, it will bring new security and privacy problems due to the several concerns such as BESS usage behavior, location, and user private information, so on.

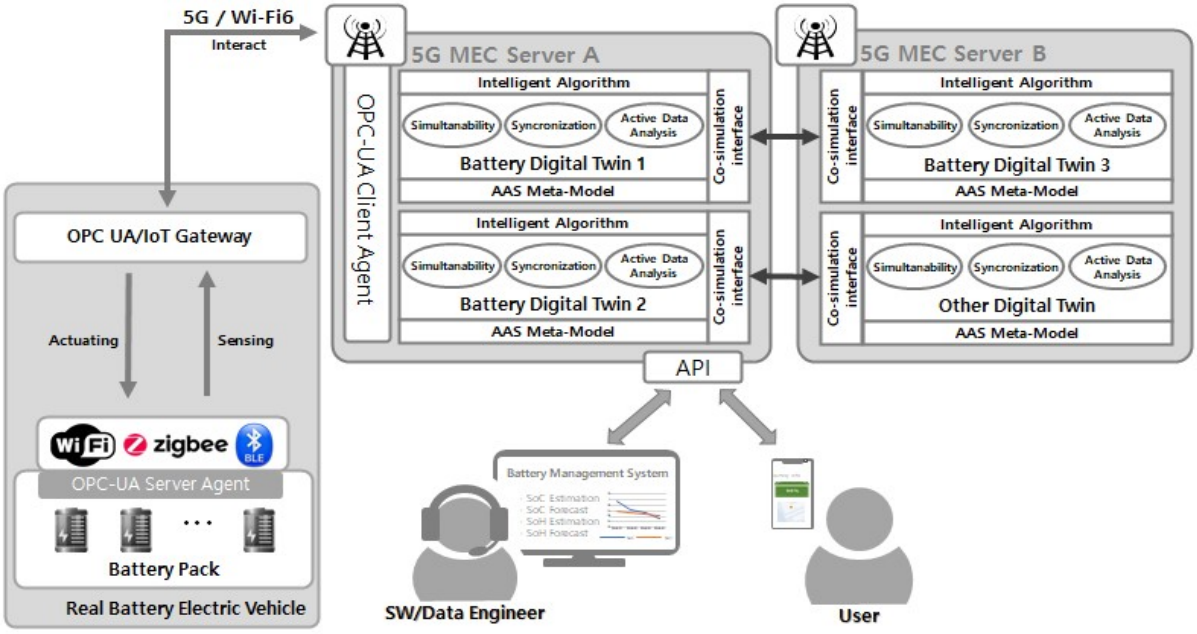


Figure 3. Proposed IoT System Architecture for Battery Digital Twin

To deal with the privacy and security issues, we plan to implement a multi-entity secure data learning scheme. The developed security scheme should allow multiple parties to find the predictive models from their overall data while not revealing their own private data to one another at the same time. Furthermore, the system may also adopt a blockchain mechanism. The blockchain's main task is to process data sharing transactions among digital twin instances located in 5G MEC servers. Using this system, we expect to provide an adaptive privacy-preserving mechanism according to privacy demands and available system resources.

IV. CONCLUSION

Design and build a system architecture for battery digital twin is challenging due to the complexity of arranging many aspects, ranging from interoperability mechanism to data sharing security issues. Using the OPC UA IoT data exchange protocol, we enable feasible development of battery digital twin that support service-oriented architecture, extensible and comprehensive information modeling. The use of 5G MEC servers can tackle the need for big data resources of battery digital twin. Furthermore, the evaluation of potential security and privacy challenges can be a guideline for us to implement a resilient and reliable application of battery digital twin in the future. For the next steps, we will evaluate the proposed architecture using an experimental prototype and propose new solutions to the mentioned challenges.

ACKNOWLEDGMENT

This research is supported by the Brain Korea (BK) 21 AI Convergence Human Resource Education and Research Center at Chonnam National University.

REFERENCES

[1] Ashtari Talkhestani, et al. "An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System" Automatisierungstechnik, vol. 67, no. 9, 2019, pp. 762-782.

[2] Weihan Li, et al. “Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation” Journal of Energy Storage, Vol. 30, 2020.

[3] Billy Wu, et al. “Battery digital twins: Perspectives on the fusion of models, data and artificial intelligence for smart battery management systems” Energy and AI, Vol. 1, 2020.

[4] Lukas Merkle, et al. “Estimate e-Golf Battery State Using Diagnostic Data and a Digital Twin” Batteries, Volume 7, 2021.

[5] Herman Haskamp, et al. “Implementing an OPC UA interface for legacy PLC-based automation systems using the Azure cloud: An ICPS-architecture with a retrofitted RFID system” IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, 2018, pp. 115-121.

[6] Peter Peniak, et al. “Extended gateway model for OPC UA/IoT device integration,” IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 2021, pp. 000155-000160.

[7] Bolun Xu, et al. ”Optimal Battery Participation in Frequency Regulation Markets” IEEE Transactions on Power Systems, vol. 33, issue 6, pp. 6715-6725, 2018.

[8] Jonathan Fuchs, et al. “I4.0-compliant integration of assets utilizing the Asset Administration Shell” 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 1243-1247.

[9] Wang P, et al. “OPC UA Message Transmission Method over CoAP” Internet Draft, IETF, 2018.

군집 연합학습을 이용한 태양광발전량 예측 연구

유은근, 고한얼*
고려대학교

{eg_yoo, heko}@korea.ac.kr

A Study on the Solar Power Forecasting using Clustered Federated Learning

Eungeun Yoo, Haneul Ko*
Korea Univ.

요 약

최근 태양광에너지의 사용이 가파르게 증가하고 있으며 향후에도 정책적 지원이 계속될 것으로 예상됨에 따라 태양광발전을 이용한 에너지 이용은 지속적으로 확대될 전망이다. 하지만 태양광발전량은 기후에 따라 변동성이 높고 예측이 어려워 안정적인 전력망 운영에 많은 부담이 되고 있다. 본 연구에서는 최근 활발하게 연구가 진행되고 있는 연합학습과 Edge Cloud 기술을 적용한 태양광발전량 예측 방법을 제안하고 실제 태양광에너지 데이터를 이용한 실험을 통해 성능을 확인하였다.

Ⅰ. 서 론

최근 화석연료의 고갈 및 환경문제에 대한 관심 증가로 신재생에너지의 사용이 가파르게 증가하고 있다. 신재생에너지는 태양광, 풍력, 바이오, 파력 등이 있으나 그 중 태양광발전은 신재생에너지 중 절반 이상의 높은 비율을 차지하고 있으며, 향후에도 지속적인 정책 지원과 비용 절감으로 태양광발전의 확장은 더욱 빠르게 가속화될 것으로 예상된다. 하지만 태양광발전량은 기후에 따른 변동성과 불확실성이 높아 예측이 어려워 안정적인 전력망 운영에 많은 부담을 주고 있는 상황이다[1].

이를 해결하기 위해 정확한 태양광발전량 예측을 위한 연구가 활발하게 진행되고 있으며 최근에는 딥러닝 기술 중 LSTM(Long Short-Term Memory)을 이용한 예측이 제안되어 기존의 예측방법(SVR, ARIMA 등) 보다 높은 성과를 보이고 있다[2]. 하지만 이런 딥러닝 모델은 과적합에 따른 성능저하를 회피하기 위해 많은 데이터를 필요로 한다. 이에 따라 여러 다른 장소의 대용량 태양광발전량 데이터를 중앙서버에 저장하여 학습하는 것이 성능향상을 위해 중요하다.

태양광발전량 예측은 장기적인 관점뿐 아니라 단기적인 관점에서의 예측도 중요하다[2]. 하지만 학습을 위해 중앙서버로 데이터 이관 후 학습을 수행하는 경우 많은 시간이 소요되고 이에 따라 실시간 예측이 불가능한 문제가 발생한다. 뿐만 아니라 데이터 전송에 따른 통신 부하 및 데이터 유출 등의 문제도 부가적으로 발생하게 된다.

이런 문제를 해결하기 위해 최근 머신러닝 분야에서 연합학습(FL, Federated Learning)이 등장하였다. FL 은 분산학습의 한 분야로 edge device 에서 학습이 수행되고 중앙서버에는 데이터가 아닌 오직 학습된 모델의 가중치만 전송한다. 중앙서버에서는 학습을 수행하지 않으며 각 edge device 에서 보내온 가중치를 평균하여 집계한 후 새로운 모델(global model)을 만들게 된다. 이에 따라 위에서 제기한 데이터 전송에 따른 여러 문제들을 해결할 수 있다. 이런 이점에 따라 FL 을 응용하여 당면한 여러 문제를 해결하고자 하는 많은 연구가 활발이 이루어지고 있으며 또한 FL 에 내재되어 있는 여러 문제를 해결하고자 하는 연구도 함께 진행되고 있다.

FL 이 가지고 있는 문제 중 대표적인 것으로 Non-IID(Independent Identically Distributed) 문제가 있다. 각 edge device 들이 데이터를 수집하고 학습하는 환경이 동일할 수 없기 때문에 자연스럽게 각 edge device 가 보유하고 있는 데이터는 전체 데이터의 분포와 다르게 된다. 이에 따라 중앙에서 취합된 global model 이 상이한 데이터 분포를 가지고 있는 edge device 에 배포될 때 해당 모델이 각 edge device 의 환경과 맞지 않아 정확도가 감소하게 된다. 예를 들어 일조량이 풍부한 평야지역과 일조량이 부족한 고산지역에 edge device 가 존재하고 이 edge device 들이 학습한 태양광 예측 모델의 가중치를 통합하여 global 모델을 생성, 배포하게 되는 경우 두 지역 모두 낮은 정확도를 나타내게 된다. 위의 문제를 해결하기 위해 다양한 방법이 제시되었으며 그 중 대표적인 논문인 [3]에서는 유사한 모델 가중치를 생성하는 edge device 들을 묶어 cluster 를 생성하고 각 cluster 별로 global 모델을 생성하는 방안을 제시했다. 하지만 태양광발전량 예측을 위한 cluster 기반의 연합학습에 대한 연구는 아직 미미한 상황이다.

본 논문에서는 각 edge device 의 위도, 경도와 같은 위치정보의 유사성을 기반으로 여러 개의 cluster 를 구성하고 해당 cluster 를 기반으로 global 모델을 생성하여 태양광발전량 예측의 정확도를 향상시키는 연구를 진행한다.

본 논문의 구성은 다음과 같다. Ⅱ. 본론에서는 본 논문에서 태양광발전량 예측을 위해 제안하는 구조의 구성요소와 동작 방식에 대하여 설명하고 Ⅲ. 실험 결과에서

는 기존의 예측방법과 비교하는 실험을 통해 제안한 방법의 우수성을 보여준다. Ⅳ. 결론에서는 해당 연구의 의의를 정리하고 향후 연구 방향을 제시한다.

Ⅱ. 본론

A. 시스템 모델

본 논문의 시스템 모델은 크게 중앙서버(Central Server)와 Edge Cloud 로 나누어진다. 중앙서버는 각 태양광발전소의 cluster 구성 및 global model 의 배포·집계를 담당한다. Edge cloud 는 edge 단에서 학습을 수행하는 edge computing 환경을 의미하며 edge device 는 각 태양광발전소안에 위치한 제어·모니터링 시스템이 될 수 있다. Edge device 는 자신이 생성한 발전 데이터를 이용하여 배포 받은 global model 을 학습하고 학습된 가중치를 중앙서버로 전송한다. 전체적인 구성은 그림 1 에서 확인할 수 있으며, 동작 과정은 다음과 같다.

1. 중앙서버는 edge device 의 위치정보(위도·경도)의 유사성을 기반으로 모든 edge device 가 포함되는 여러개의 cluster 를 구성한다.
2. 중앙서버는 각 cluster 별로 global model 을 edge device 에게 전송한다.
3. 각 edge device 는 자신의 태양광발전 데이터와 global model 을 이용하여 학습한다.
4. 각 edge device 는 학습이 완료된 local model 의 가중치를 중앙서버에게 전송한다.
5. 중앙서버는 각 cluster 별로 수신된 local model 의 가중치를 평균하여 cluster 별로 새로운 global model 을 생성한다.
6. 2~6 과정을 특정 횟수 또는 특정 정확도에 도달할 때까지 반복한다.

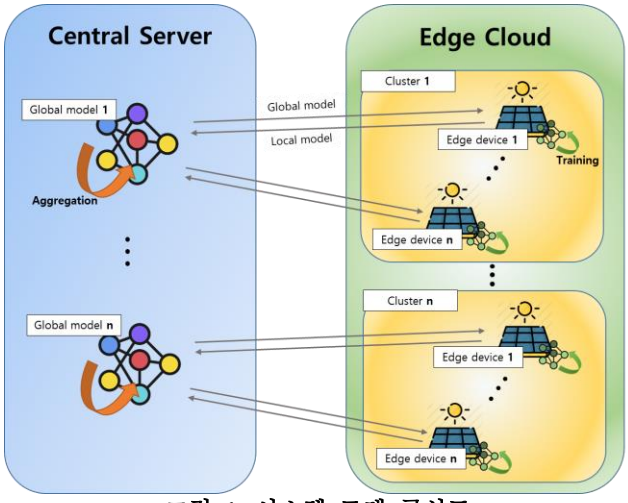


그림 1. 시스템 모델 구성도

B. 태양광 발전소의 Cluster 구성

각 태양광 발전소를 적절한 cluster 에 배치하는 것은 전체적인 성능향상을 위해 중요하다. [4]에서는 한 발전소에서 기상정보가 유사한 시기의 데이터를 기준으로 k-means 알고리즘을 적용하여 cluster 를 구성하는 것을 제안하였다. 본 연구에서는 하나의 발전소의 데이터를 군집화하는 것이 아닌 다수의 발전소에 대한 FL 의 성능을 시험하는 것이 목적이다. 이를 위해 각 발전소의 위치가 가까울 수록 기상정보가 유사할 것이라는 가정을 바탕으로 각 발전소가 위치한 위도·경도 정보를 기반으로한 k-means 알고리즘을 적용하여 cluster 를 구성하였다.

Ⅲ. 실험결과

A. 실험 환경 구성

본 실험을 위해 kaggle 에서 시행한 AMS 2013-2014 Solar Energy Prediction Contest[1]의 데이터를 이용하였다. 해당 데이터는 미국 오클라호마주에 위치한 기상관측소인 mesonet 에서 측정한 태양광에너지 정보를 포함하고 있으며 오클라호마주 전역에 걸쳐 98 개소의 데이터를 제공하고 있다. 수집 기간은 1994 년부터 2007 년까지이며 1 일 단위로 누적된 태양광에너지량을 제공하고 있다.

본 실험에서는 98 개소의 기상관측소를 edge device 로 가정하고 각 기상관측소가 가지고 있는 데이터는 정규화(Min-Max Normalization) 후 7:3 의 비율로 나누어 학습과 시험을 진행했다. 실험을 단순화 하기 위해 32 개의 뉴런을 갖고 있는 1 계층의 LSTM 모델을 이용하며 30 일간의 데이터를 입력으로 하고 1 일 이후의 태양광에너지량을 예측하는 간단한 모델을 이용했다. FL 진행시 학습에 참여하는 edge device 는 10 개를 무작위로 선택하였고 local 에서의 학습은 5 회(epoch) 수행, 중앙서버에서의 모델 배포와 가중치 집계(round)는 20 회 수행했다. 개발은 Tensorflow Federated 를 이용하여 Google Colab 에서 진행했다.

B. 세부 실험 결과

FL 및 cluster 구성의 효과를 검증하기 위해 3 가지 시나리오를 구성하여 실험을 진행하였으며 실험결과는 표 1 과 같다. NoFed 는 각 태양광발전소가 자신의 데이터를 이용하여 학습한 결과의 평균을 나타내며, Federated 는 FL 에서 가장 많이 사용되는 FedAvg 알고리즘을 이용한 결과이며, Clustered FL 은 본 연구에서 제안하고 있는 지리적 위치기반의 cluster 를 이용한 FL 결과를 나타낸다. Clustered FL 우측의 숫자는 생성한 cluster 의 수를 의미한다.

표 1. 학습 수행 방법에 따른 성능 비교

Scenario		MAE(Mean Absolute Error)
NoFed		0.542
Federated		0.498
Clustered FL	2	0.487
	4	0.470
	8	**0.468**
	16	0.481
	32	0.501

표 1 에서 보는 바와 같이 본 논문에서 제안하는 위치기반의 cluster 를 이용한 FL 방법이 자신의 데이터만 이용하여 학습하는 경우보다 cluster 의 수가 8 일때 최대 15.8% 향상됨을 알 수 있다. 또한, cluster 를 사용하지 않는 FL 에 비해서도 최대 6.4% 향상되는 것을 알 수 있다. 이를 통해 여러 개의 발전소가 FL 을 통하여 공동으로 예측 모델을 생성하는 것이 성능 향상을 위해 중요하며 또한, 본 실험에서 구성한 cluster 안의 edge device 들이 유사한 데이터 분포를 갖도록 적합하게 cluster 가 구성됨에 따라 위치 정보 기반의 clustered FL 이 기존의 FL 보다 좋은 성능을 보이게 됨을 알 수 있다.

[1] https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest/data

Ⅳ. 결론

본 논문은 태양광발전량 예측의 정확도 향상을 위해 다양한 지역에 위치한 태양광발전소들을 위치정보를 기반으로 cluster 를 구성하여 FL 을 수행하는 방안을 제안하였다. 이를 통해 각 태양광발전소가 자신의 데이터만 이용하여 학습하는 경우에 비해 15.8 %의 정확도 향상을 얻을 수 있었다. 향후에는 FL 진행중에 edge device 가 학습한 모델의 가중치를 이용하여 동적으로 cluster 를 재구축하는 방법 및 최적의 cluster 수와 크기를 결정할 수 있는 방법에 대한 제안을 통해 최적의 cluster 구축을 위한 연구를 진행할 것이다.

ACKNOWLEDGMENT

본 연구는 2019 년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(과제번호: 2019R1C1C1004352)

참 고 문 헌

[1] C. Wan, J. Zhao, Y. Song, Z. Xu, J. Lin, and Z. Hu, “Photovoltaic and solar power forecasting for smart grid energy management,” CSEE Journal of Power and Energy Systems, vol. 1, Issue 4, pp. 38-46, Dec. 2015.

[2] Y. Lin, D. Duan, X. Hong, X. Cheng, L. Yang, and S. Cui, “Very-Short-Term Solar Forecasting with Long Short-Term Memory (LSTM) Network,” Asia Energy and Electrical Engineering Symposium (AEEES) 2020, May 2020.

[3] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-Center Federated Learning," arXiv:2005.01026v1, May 2020.

[4] M. Rana, I. Koprinska, and V. Agelidis, “Solar power forecasting using weather type clustering and ensembles of neural networks,” International Joint Conference on Neural Networks (IJCNN) 2016, Jul. 2016.

스마트 공장을 위한 IEEE 802.11n 기반의 무선 네트워크 설계 및 시뮬레이션

김민철, 김영탁*
영남대학교 대학원 정보통신공학과
kmc724@ynu.ac.kr, *ytkim@yu.ac.kr

Design and Simulation of Wireless Network based on IEEE 802.11n for Smart Factory

Min-Cheol Kim, Young-Tak Kim*
Dept. of Information & Communication, Graduate School, Yeungnam University

요 약

4차산업 시대에 들어서며, 기존의 공장들이 여러 개의 센서와 액추에이터를 생산 시설이나 로봇 등에 설치하여 생산량을 주기적으로 확인하고 목표 생산량에 도달하고 있으며, 이러한 공장들을 스마트 공장이라고 불려지고 있다. 스마트 공장에 설치된 기기들 중 일부는 낮은 지연율과 높은 안정성을 가져야 한다. 본 논문에서는 스마트 공장에 설치되는 많은 센서들과 액추에이터 뿐만 아니라 로봇 간의 통신에도 사용할 수 있으며, 낮은 지연율과 높은 안정성을 가진 IEEE 802.11n 기반의 네트워크 기법을 연구하였으며 이에 대한 구조 및 설명과 시뮬레이션으로 측정한 결과를 분석한다.

Ⅰ. 서 론

4차 산업 시대에 들어서며, 다양한 관련 연구들이 활발히 진행되고 있다. 대표적으로 인공지능과 사물인터넷, 블록체인 그리고 빅데이터 기술이 차세대 기술로써, 정부에서도 이들을 주시하며 지원하고 있다 [1-3]. 특히 사물인터넷 분야는 한 보고서에 따르면 2025년에 예상되는 잠재 경제력은 최소 3.9조불에서 최대 11.11조불로 예상하고 있으며 자동화 공장에 사용할 수 있는 사물인터넷 분야의 잠재 경제력은 최소 1200억불에서 최대 3700억불까지 예상하였다 [4]. 이러한 흐름에 여러 공장들이 유지 비용 및 운영 비용을 절감하고 제품의 생산 관리를 주기적으로 확인할 수 있는 사물인터넷 (IoT) 기반의 자동화 공장 및 스마트 공장으로의 변환을 진행하고 있다. 스마트 공장으로의 변화는 급증하는 노동력 부족 문제와 낮은 질의 제품 생산력 문제를 해결할 수 있다 [5].

스마트 공장을 구축하기 위해서는 공장 내의 기기들의 역할에 따른 계층 분리가 필요하다. 예를 들어 현장에서 운영자가 중앙에서 전체 공정을 관리하는 엔터프라이즈 계층과 실제 공정을 담당하는 기기의 컨트롤러 및 운영자가 이들을 세부적으로 관리하는 컨트롤 계층, 그리고 실제 기기들의 필드 계층으로 나눌 수 있으며 엔터프라이즈 계층은 컨트롤 계층과 통신할 수 있으며, 컨트롤 계층은 필드 계층과 통신이 가능하다. 엔터프라이즈는 1초에서 1시간 간격의 주기적인 데이터를 전송하여 관리자에게 공장 전체의 공정에 대해 정보를 알려주며, 컨트롤 계층에서는 100ms 간격의 주기적인 데이터를 전송하여 엔터프라이즈 계층에서 이를 수집하고 가공한다. 다만 필드 계층에서는 기계 내부의 통신으로써, 데이터의 무결성 및 초저지연성을 보장하여야 하기 때문에 유선으로 구성되어 있어야 하며 데이터 전송 주기는 250μs ~ 1ms 내로 작아야 한다 [6].

본 논문에서는 스마트 공장에 사용할 수 있는 저지연 고성능 네트워크 기법에 대해 연구하였으며, 이에 대한 설명과 시뮬레이션으로 측정한 결과에 대해 분석하였다. 본 논문의 구성은 다음과 같다. Ⅱ절에서 공장 네트워크를 구성한 다양한 기법들에 대해 간략히 설명하며, Ⅲ절에서는 본 논문에서 제안하는 네트워크 기법에 대해 설명하며, Ⅳ절에서는 제안하는 네트워크 기법의 성능에 대한 결과 및 분석을 하였으며, Ⅴ절에서 결론과 함께 향후 연구 개발을 소개한다.

Ⅱ. 관련 연구

2.1 ISA100 (International Society of Automation)

ISA100은 필드 레벨 (Level 0)를 중심으로 자동화 및 제어 환경에서 무선 시스템 구현을 위한 기술과 절차를 정의하는 표준 제정을 진행하였다. ISA100에서 공장 내의 통신을 위해 3개의 영역과 5개의 클래스로 기기를 분류할 수 있도록 표준으로 정의하였다. 최상위단인 Safety 영역의 클래스 0은 Emergency action으로써, 돌발 상황 발생시 사용되는 메시지로써, 지연시간이 가장 작아야 한다. 다음 상위 영역인 Control 에는 3개의 클래스가 존재하며 해당 영역의 데이터들은 주기성을 가지고 있다. Monitoring 영역에는 2개의 클래스가 존재하며 각각 Alert과 Log 메시지를 의미한다. Alert은 운영자에게 공정이 완료되었거나, 목표 생산량을 달성했을 때 알려주는 메시지를 의미하며 Log는 네트워크의 상태를 기록한 메시지를 의미한다 [7].

2.2 OPC UA (OPC Unified Architecture)

OPC UA는 OPC 재단이 개발한 산업용 기계간 통신 (M2M) 프로토콜이다. 이는 서버/클라이언트 통신 기반으로 개발되었으며, 산업용 기기들의 데이터 수집 및 제어에 초점이 맞추어져있다. OPC UA는 GPL 2.0 라이센스를 사용하여 무료로 공개되어 있으며, 여러 시스템에서 동작할 수 있는 Cross-platform 기능을 지원한다. OPC UA는 기존의 다양한 프로토콜 (SERCOS, ProfiNet, CAN 등) 들의 통합을 주 목표로 개발하였으며, 이들은 기기들의 Application Layer에서 데이터 통신의 보안성 및 무결성을 보장하도록 설계 및 동작한다.

표 1. ISA100 Usage Classes

Domain	Usage Class	Description
Safety	Class 0: Emergency action	Always critical
Control	Class 1: Closed-loop regularity control	Often critical
	Class 2: Closed-loop supervisory control	Usually, Non-Critical
	Class 3: Open-loop control	Human in the loop
Monitoring	Class 4: Alerting	Short-term operational consequence (e.g., event-based maintenance)
	Class 5: Logging and downloading /uploading	No immediate operational consequence (e.g, history collection, sequence of events, preventive maintenance)

2.3 WIA-FA (Wireless networks for Industrial Automation - factory Automation)

WIA-FA는 IWCNs (Industrial Wireless Control Networks) 표준으로써 IEC (International Electrotechnical Commission)에서 무선 공장 네트워크의 시작인 IEC 62591, WirelessHART (Wireless sensor network technology based on the Highway Address Remote Transducer) 를 2010년에 발표한 이후, 2011년에 IEC 62601, WIA-PA (Wireless network for Industrial Automation - Process Automation), 2014년에 IEC 62743, ISA100.11a를 발표하고 2017년에 802.11n 기반의 TDMA 기반의 WIA-FA를 발표하였다. WIA-FA는 Superframe 기반으로 여러 개의 시간 슬롯으로 Superframe을 나누고, 첫 번째 슬롯을 비콘 타임 슬롯으로 사용하고, 나머지 슬롯에서 데이터 전송이 이루어진다 [9-12].

Ⅲ. 스마트 공장을 위한 IEEE 802.11n 기반의 무선 네트워크 설계

3.1 비콘 프레임 구조

비콘 프레임을 통해 스테이션은 AP (Access Point) 와의 시동기가 맞추어 지며, AP가 현재의 네트워크 상황을 알려주기 위해 프레임 안에 상태를 알려주기도 한다. 본 논문에서 제안하는 네트워크 기법 또한 스테이션과 AP사이의 시동기를 조절하고, 네트워크 상태를 알려주며, 접속된 스테이션의 개수와 사용할 데이터 속도를 알려준다. 현재의 네트워크 상태는 Frame Control (Frame Ctrl) 과 Sequence Control (Seq Ctrl) 로 확인할 수 있으며 시동기는 비콘 프레임을 수신한 시간과 프레임 내부의 비콘 메시지를 전송한 시간 Timestamp와 Beacon Interval을 통하여 조절을 할 수 있다. Num Stations은 접속된 스테이션의 개수를 의미하는데, 목표 생산량을 늘리기 위해 기계를 더 추가하거나 비용 절감을 위해 기계를 제거할 수도 있다. 또한 각 스테이션 별로 정보를 할당하는 것 보다, 스테이션의 개수를 알려줌으로써, 각 스테이션이 사용할 시간 슬롯 (T-Slot)을 직접 계산할 수 있도록 하여 비콘 프레임의 전송 시간을 낮춘다. 그리고 전송 속도 정보를 통해 채널에서 사용할 전송 속도를 각 스테이션에게 알려주어 주어진 속도로 변경할 수 있도록 한다. 이를 통해 물리적인 상황과 채널의 상태를 매 비콘 주기마다 확인할 수 있고, 이에 따라 자동으로 변경이 된다.

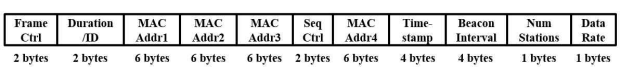


그림 1. 비콘 프레임 구조

3.2 Autonomous Set Data Rate

AP는 접속된 스테이션의 개수만큼 자동으로 데이터 속도를 조절할 수 있어야 한다. 이는 스테이션이 전송하는 데이터 크기에 따라 달라진다. 본 논문에서는 IEEE 802.11n Modulation Coding Scheme (MCS) 인덱스 0 부터 7 까지 사용한다. AP는 접속된 스테이션의 개수와 중요 메시지를 전송할 CSMA/CA 슬롯 (C-Slot) 개수 N_{CSLOT} 를 합쳐서 한 비콘 주기에 사용할 수 있는 슬롯의 개수 N_{slot} 를 정한다. 그리고 비콘 주기 BI 를 정하고, 프레임의 크기 L_{frame} 을 설정한다. 그리고 비콘 메시지의 전송 시간 T_{Beacon} 을 구하고 TRX 모드를 변경하는 시간 $T_{TRXDelay}$ 을 구하여 최종적으로 사용 가능한 슬롯의 크기를 구한다. 여기서, 각 인덱스에 해당하는 속도들이 사용할 수 있는 최소 슬롯의 크기를 구하여야 하는데, 각 속도의 최소 슬롯은 데이터 전송 시간 T_{NMbps} 과 ACK 메시지 전송 시간 T_{ACK} 를 합하면 된다. 이를 수도코드로 작성하면 다음과 같다.

Function SetDataRate

1: BI: Beacon interval
2: T_{Beacon}: Beacon message transmission time
3: L_{frame}: Frame size
4: N_{STA}: Number of Stations
5: N_{CSLOT}: Number of CSMA/CA Slots
6: N_{Slot}: $N_{STA} + N_{CSLOT}$
7: T_{ACK}: ACK message duration
8: $T_{TRXDelay}$: Delay of change TX/RX mode
9: $DR = \{MCS0, MCS1, MCS2, MCS3, MCS4, MCS5, MCS6, MCS7\}$
10: T_{NMbps}: $((L_{frame} \times 1000000) / DR_N) + T_{ACK},\ DR_N \subset DR$
11: T_{Slot}: $(BI - T_{TRXDelay} - T_{Beacon}) / N_{Slot}$
12: **if** ($T_{6Mbps} < T_{Slot}$)
13: datarate = 6Mbps
14: **else if** ($T_{9Mbps} < T_{Slot} < T_{6Mbps}$)
15: datarate = 9Mbps
16: **else if** ($T_{12Mbps} < T_{Slot} < T_{9Mbps}$)
17: datarate = 12Mbps
18: **else if** ($T_{18Mbps} < T_{Slot} < T_{12Mbps}$)
19: datarate = 18Mbps
20: **else if** ($T_{24Mbps} < T_{Slot} < T_{18Mbps}$)
21: datarate = 24Mbps
22: **else if** ($T_{36Mbps} < T_{Slot} < T_{24Mbps}$)
23: datarate = 36Mbps
24: **else if** ($T_{48Mbps} < T_{Slot} < T_{36Mbps}$)
25: datarate = 48Mbps
26: **else if** ($T_{54Mbps} < T_{Slot} < T_{48Mbps}$)
27: datarate = 54Mbps
28: **end if**
29: **end**

수도코드 1. Autonomous Set Data Rate

3.3 Periodic Message

공장의 기계들은 주기적으로 데이터를 생성하여 전송하며, 현장에서 사용되는 기계의 역할이 정해져 있다. 또한 한 AP로 접속하는 기기의 개수는 넓은 범위의 공장을 운영한다면 150대가 동작할 수 있다. 본 논문에서는 한 비콘 주기에 접속한 기기의 개수만큼 T-Slot을 할당시키고 기기들은 AP로부터 할당 받은 AID를 통해 데이터를 목적지로 송신할 수 있도록 설계하였다. 또한 3.2절을 통하여 자동으로 속도를 조절 가능한 기능을 통해 새로운 기기를 배치하여도 접속하여도 문제없이 통신이 가능하도록 설계하였다.

3.4 Mission Critical Message

공장 네트워크에서, 각 기기들은 주기적으로 데이터를 전송할 때가 대부분이지만, 돌발 상황이 발생했거나 특정한 일을 수행 후, 운영자에게 메

시지를 전달할 필요가 있는 경우도 있다. 3.2절에서 C-Slot을 배치하는 것도 이러한 이유 때문이다. C-Slot은 CSMA/CA 기법을 사용하므로 기본적으로 Contention-based 통신 방식을 사용한다. 다만, 데이터 생성 시간이 길 경우, T-Slot을 사용할 수 있도록 하여 전송 안정성을 증가시킬 수 있다. 또한 T-Slot에 Critical 메시지를 전송하였는데, Periodic 메시지가 생성 될 경우에는 채널의 상태를 확인하여 C-Slot을 사용하여 메시지를 전송할 수 있도록 하였다.

3.5 Requirements

스마트 공장 구현을 위해 [8] 에서는 클래스 별로 지연시간 및 전송 안정성에 대해 정의하였다. 클래스 1은 10ms 이내의 지연시간 및 반드시 도착해야 하는 안정성이 필요하며, 클래스 2와 3은 10ms ~ 100ms 이내의 지연시간 및 99.99% 이상의 안정성이 필요하다. 클래스 4와 5는 평균 100ms의 지연시간과 99% 이상의 안정성을 보이면 된다. 즉, 스마트 공장을 구현하기 위해서는 100%에 근접하는 안정성을 보장해야 하며, 지연시간 또한 최대 100ms로써 굉장히 짧아야 가능하다. 본 논문에서는 이러한 요건을 충족하는 네트워크 기법을 연구하였으며 4절에서 이를 설명한다.

표 2. 스마트 공장 설계를 위한 성능 요건

Attribute		Requirements
Delay	Class 1	〈 10ms
	Class 2-3	10ms ~ 100ms
	Class 4-5	100ms average
Transmission Reliability	Class 1	99.99999% ~ 100%
	Class 2-3	≥ 99.99%
	Class 4-5	≥ 99%

IV. 시뮬레이터 설계 및 결과

4.1 시뮬레이터 설계

본 논문에서 설계한 네트워크를 실험하기 위해 NS-3 (Network Simulator-3) 시뮬레이터를 사용하였다. IEEE 802.11n 기반으로 시작 속도는 6Mbps로 설정하였다. 그리고 C-Slot을 10개로 정하고 스테이션의 개수는 30대에서 100대를 배치하였으며, 일반 데이터의 생성 주기 및 크기는 각각 10ms, 100Bytes로 설정하였다. 중요 데이터의 생성 주기는 1초에서 128초로 설정하였고 크기는 10Bytes로 설정하였다. 그리고 100초동안 시뮬레이터를 실행하였고 결과를 도출하였다.

표 3. 시뮬레이터 파라미터

Parameters	Value	
Central Frequency	5.18 GHz	
Bandwidth	20MHz	
Station Tx Gain	0.0	
Station Rx Gain	0.0	
AP Tx Gain	3.0	
AP Rx Gain	3.0	
Data Rate	6Mbps, 9Mbps, 12Mbps, 18Mbps, 24Mbps, 36Mbps, 48Mbps, 54Mbps,	
SIFS	16μs	
Min CW	3	
Max CW	7	
Aifsn	2	
Beacon Interval	10ms	
Beacon size	46 bytes	
ACK size	14 bytes	
Number of C-Slot	10	
Number of Stations	30 ~ 100	
Payload size	Normal	100 Bytes
	Critical	10 Bytes
데이터 생성 주기	Normal	10ms
	Critical	1s ~ 128s
실험 시간	100초	

4.2 비콘 슈퍼 프레임

비콘 전송 시간은 약 80μs 이며, 10개의 C-Slot을 배치하였다. 마지막 250μs는 Tx/Rx 모드 변경 지연 시간으로써, 해당 시간에 전송 및 수신이 발생할 수 없다. T-Slot은 스테이션의 개수에 따라 변하며, 슬롯의 크기 또한 스테이션의 개수와 전송할 프레임의 크기 및 전송 속도에 따라 변한다.

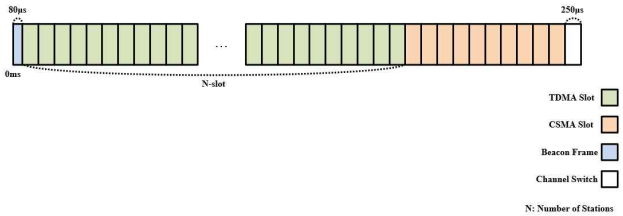


그림 2. 비콘 슈퍼 프레임

4.3 스테이션 개수 및 프레임 크기에 따른 속도 변경

3.2절에서 스테이션의 개수와 프레임의 크기에 따라 데이터의 속도를 변경하는 수도코드를 작성하였는데, 결과는 다음과 같다. 스테이션의 개수가 20대일 때, 6Mbps로 설정이 되었고, 30대일 때 9Mbps, 40대, 50일 때는 12Mbps, 60대일 때 18Mbps, 70, 80대일 때 24Mbps, 90대일 때 36Mbps, 100대일 때 48Mbps로 속도가 설정이 되었다.

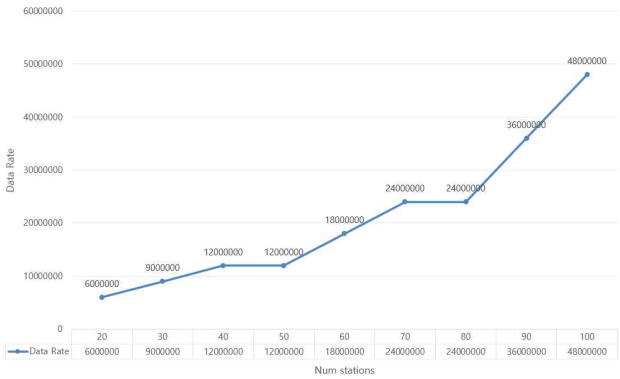


그림 3. 스테이션 개수에 따른 속도 변화

4.4 AP 및 스테이션 설계

NS-3는 WI-FI 모듈을 지원하는데, PHY (Physical) 레이어와 MAC (Medium Access Control) 레이어를 IEEE 802.11 표준에 맞게 작성되어 있다. PHY는 사용할 수 있지만, MAC은 그대로 사용하기에는 비콘 프레임의 크기가 다르며 TDMA 기법이 사용되지 않기 때문에 새로 작성해야 한다. 본 논문에서는 AP MAC과 스테이션의 MAC을 각각 IoTApWifiMac, IoTStaWifiMac으로 정의하였다. 상위 계층에서는 NS-3가 제공하는 PacketSocket을 이용하여 IP 스택을 쓰지 않고 MAC으로 바로 데이터를 전송할 수 있다. 어플리케이션은 파라미터의 Payload size와 데이터 생성 주기를 스크래치 파일에서 작성하고 이를 받아올 수 있도록 작성하였다. 해당 시뮬레이터 자료는 [13]에서 확인할 수 있다.

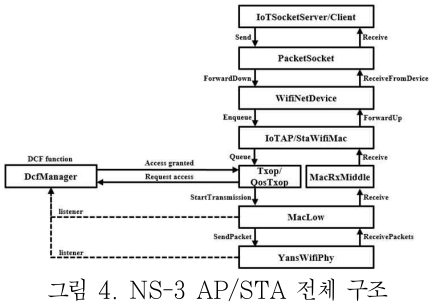


그림 4. NS-3 AP/STA 전체 구조

4.5 시뮬레이터 결과

표 3의 파라미터를 시뮬레이터에 입력하여 스테이션의 개수에 따른 Normal 데이터와 Critical 데이터의 PDR (Packet Delivery Rate) 및 지연시간을 측정하였다. PDR은 Normal 메시지가 Critical 메시지처럼 모두 도착하지는 않았는데, 통신 중간에 C-Slot을 사용하여 전송이 실패한 것으로 보이며, 이 부분에 대한 구체적인 문제 분석과 최적하가 필요한 것으로 파악된다. Critical 메시지의 경우 모든 데이터가 제대로 도착하였다. 일반 메시지와 Critical 메시지의 지연시간은 두 메시지 평균값이 모두 5ms 이내로 10ms 이내로 전부 도착한 것을 확인할 수 있다. 여기서 스테이션의 개수가 늘어남에 따라 지연시간도 증가하게 되었는데, 요청하는 스테이션이 늘어난 것이 원인으로 보인다.

표 4. 시뮬레이터 결과

Num Stations	PDR		Delay	
	Normal	Critical	Normal	Critical
30	0.999	1	3.81ms	3.74ms
40	0.999	1	4.06ms	4.01ms
50	0.999	1	4.25ms	4.20ms
60	0.999	1	4.31ms	4.30ms
70	0.999	1	4.59ms	4.37ms
80	0.999	1	4.57ms	4.47ms
90	0.999	1	4.64ms	4.52ms
100	0.999	1	4.62ms	4.58ms

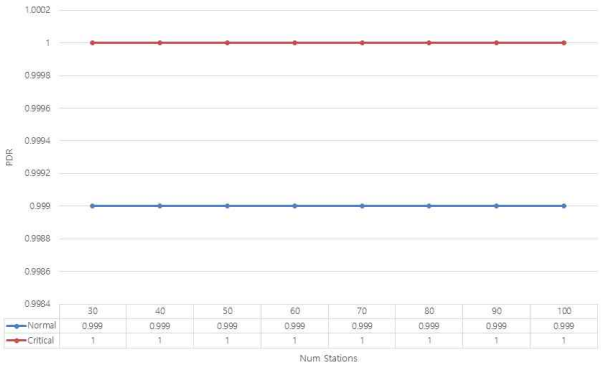


그림 5. 스테이션 개수에 따른 Normal 메시지와 Critical 메시지의 평균 PDR

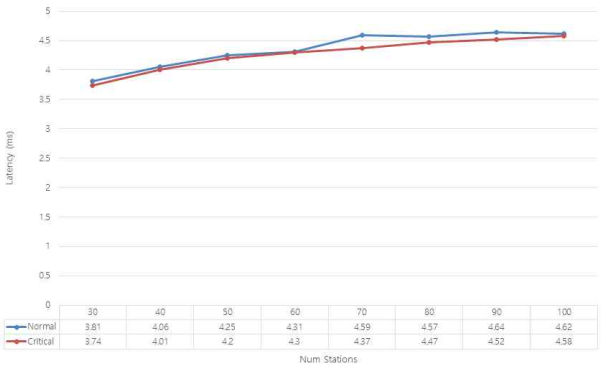


그림 6. 스테이션 개수에 따른 Normal 메시지와 Critical 메시지의 평균 지연 시간

V. 결론

본 논문에서는 스마트 공장을 위한 네트워크 기법을 연구하고 시뮬레이터로 실험한 결과를 분석하였다. 스마트 공장을 설계하기 위해서는 기기들의 역할을 분류하고 해당 역할들 끼리 계층을 나누어 통신이 이루어져야 한다. 예를 들어, 운영자와 컨트롤러 간의 통신과 컨트롤러와 기기간의 통신, 그리고 기기와 기기간의 통신이 공장 내에서 이루어지며, 각 통신에 맞는 요건들을 맞추어야 한다. 이를 위해 IEC에서는 5개의 계층을 나누어 역할에 따른 통신 요건을 표준으로 정의하였다. 본 논문에서 제안하는 네트워크 기법을 이용해 기기간의 통신뿐만 아닌 필드 간의 통신과 필드와 컨트롤 타워간의 통신 기법에 대해 더 연구하고, 실제 실험을 통해서 검증해 볼 것이다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT연구육성지원사업의 연구결과로 수행되었음. (IITP-2020-2016-0-00313)

참 고 문 헌

[1] 대통령 직속 4차산업혁명위원회, “4차 산업혁명 대정부 권고안 - 권고문”, 2021-04-04

[2] 중소기업벤처부, “스마트 제조혁신 가속화 붐 조성 추진”, 2019-05-28

[3] 기술혁신정책과, “2019년 스마트공장 보급확산사업 지원계획 공고”, 2019-02-13

[4] J.Manykia, M.Chui, P.Bisson, J.Woetzei, R.Dobbs, J.Bunghin and D.Aharon, “The Internet of Things: Mapping the Value Beyond the Hype Executive Summary”, McKinsey Global Institute, June 2015

[5] Ryota Yamada, “Introduction of Radio-Integrated System Applications in the Field of Factory Automation”, 2020 IEEE Internation Symposium on Radio-Frequency Integration Technology (RFIT), 2-4 Sept. 2020

[6] Andreas Eckhardt, Sebastian Müller, Ludwig Leurs, “An evaluation of the application of OPC UA Publish Subscribe on factory automation use-case”, 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 4-7 Sept 2018

[7] “Technical Report ISA-TR100.00.03-2011“, International Society of Automation, May 2011

[8] Wei Liang, “WIA-FA and Its Applications to Digital Factory: A Wireless Network Solution for Factory Automation”, Proceedings of the IEEE (Volume: 107, Issue: 6, June 2019), 22 Februrary 2019

[9] Industrial Networks - Wireless Communication Network and Communication Profiles - WirelessHART, Standard IEC 62591 Ed. 2.0, Dec. 2016

[10] Industrial Networks - Wireless Communication Network and Communication Profiles - WIA-PA, Standard IEC 62601 Ed. 2.0, Dec. 2015

[11] Industrial Networks - Wireless Communication Network and Communication Profiles - ISA100.11a Standard IEC 62734 Ed. 2.0, Dec. 2014

[12] Industrial Networks - Wireless Communication Network and Communication Profiles - WIA-FA Standard IEC 62948 Ed. 2.0, Dec. 2017

[13] NS3 for Smart Factory based on IEEE802.11n - https://github.com/kmc000724/NS3-for-Smart-Factory-based-on-IEEE802.11n

Edge Computing Assisted Deep Reinforcement Learning-based Approach for Smart Grid Stability Detection

Luyao Zou and Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Korea
email:{zouluyao, cshong}@khu.ac.kr

Abstract

Smart grid, which has a huge power production unit, is recognized as a critical type of cyber-physical system (CPS) nowadays plays a key role in efficient and reliable energy management. Nevertheless, the smart grid faces the risk of instability, due to it highly depends on the communication network. In addition, the energy data (e.g., energy generation data) has uncertain characteristics which also increases the vulnerability of the smart grid system. Consequently, there is a dire need to detect the stability of the smart grid. In this regard, we propose a docker container-based edge computing-assisted deep reinforcement learning (DRL) approach in this work. In particular, the docker container deployed in the base station is responsible for collecting the energy data from the sensors and transferring the collected data to the remote cloud datacenter. In cloud datacenter, we adopt the DRL with experience replay to detect the smart grid stability. The experiment results show the proposed method achieves a significant gain. Specifically, with the proposed method, the accuracy for detecting the smart grid stability can reach 99.35%.

I. Introduction

Smart grid presently has been considered as the cyber-physical system (CPS) [1], which is of paramount importance in producing and managing the huge energy. Nevertheless, because the smart grid has the characteristic of strongly relying on the communication network and the energy data is uncertain, the vulnerability of the smart grid is increased [1] leading to the instability (e.g., outages, extensive blackouts) that happens in the smart grid. Therefore, it makes sense and necessary to detect the smart grid stability such that to enhance the smart grid.

Common methods for stability detection contains the impedance relays utilization [2], wavelet transform [3] and current assessment [4]. However, these methods will cause unnecessary outages [1]. Thus, unlike these methods, this paper proposes a container-based edge computing-assisted deep reinforcement learning with experience replay (DRL) approach to intelligently detect the smart grid stability.

The key contributions of this work are stated as follows:

- Firstly, we propose a conceptual view of a system model, in which we introduce the docker container in the base station [5] for collecting data from sensors and transferring the collected data to the remote cloud data center for further analysis, etc.
- Secondly, in the cloud data center, we apply DRL to detect the stability of the smart grid by obtaining the optimal policy.
- Finally, the experimental results show the proposed method can realize high accuracy for smart grid stability detection.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. 2020R1A4A1018607). *Dr. CS Hong is the corresponding author.

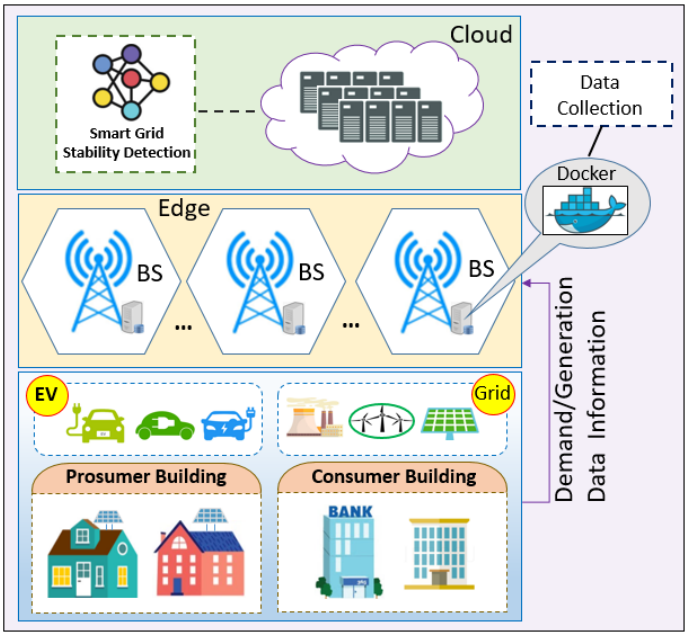


Fig. 1: Conceptual View Of The System Model

II. System Model

This section illustrates the conceptual view of the system model which consists of multiple prosumers, multiple consumers, power grid, the base station (BS), and cloud datacenter, shown as Fig. 1. Besides, the Docker container which is helpful for prompt deployment of algorithms [6] is deployed in the base station for receiving the energy data from the sensors such as energy generation data of the prosumers, energy demand data, etc. And then, the collected data will be submitted to the remote cloud data center for further analysis.

Consider a set $\mathcal{M} = \{1, 2, .., M\}$ of prosumers and con-

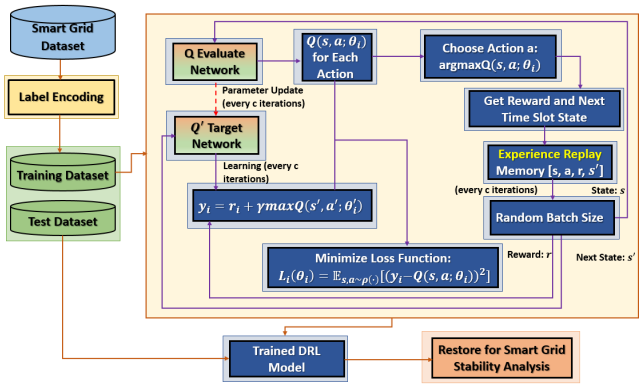


Fig. 2: Workflow Of Proposed Method

sumers and each $i \in \mathcal{M}$ generates energy or consume energy $P_i(t)$ at time slot, where $P_i(t) > 0$ represents generates energy, $P_i(t) < 0$ represents consume energy. According to [7], the stability of smart grid can be calculated as follows:

$$\frac{d^2\varphi_i}{dt^2} = P_i - \beta_i \frac{d\varphi_i}{dt} + \Sigma_{\forall j \in \mathcal{M}} \Upsilon_{ij} sin(\varphi_j - \varphi_i) - \frac{\eta_i}{T} \int_{t-T}^{t} \frac{d\varphi_i}{dt}(t' - \tau). \quad (1)$$

Here, $\frac{d\varphi_i}{dt}$ is the angular frequency deviation, $\varphi_i(t)$ denotes the rotor angle, β_i represents the damping constant and Υ_{ij} is the coupling strength between i and j, and τ is the delay. Besides, η_i is the coefficient which is proportional to the price elasticity of each $i \in \mathcal{M}$ [7]. Denote $\zeta(t) = \frac{d^2\varphi_i}{dt^2}$, the eq. (1) can be rewritten as follows [7]:

$$\zeta(t) = P_i - \beta_i \frac{d\varphi_i}{dt} + \Sigma_{\forall j \in \mathcal{M}} \Upsilon_{ij} sin(\varphi_j - \varphi_i) - \frac{\eta_i}{T}[\varphi_i(t-\tau) - \varphi_i(t-\tau-T)], \forall i \in \mathcal{M}. \quad (2)$$

In eq. (2), if $\zeta(t) < 0$, the smart grid is considered as stable, while if $\zeta(t) > 0$, the smart grid is instable [7].

III. Solution For Smart Grid Stability Analysis

To analyze the smart grid stability, we apply the deep Q learning (DQN) with experience replay, which aims to analyzing the stability of smart grid with high accuracy. In this method, we define the state space as $\mathcal{S} = \{1, 2, ..., S\}$, where $s_t \in \mathcal{S}$ contains a $3 \times M$ elements at time slot t, $s_t : (\tau_1, P_1, \eta_1, ..., \tau_M, P_M, \eta_M)$. Besides, we define the action space as $\mathcal{A} = \{1, 2, ..., A\}$, where an action $a_t \in \mathcal{A}$ is composed of two actions tuple (δ^0, δ^1), in which δ_t^0 represents the smart grid stable decision and δ_t^1 represents the smart grid unstable decision at time slot t. The objective of this work is to detect the stability of the smart grid with high accuracy. Hence, we define the immediate reward as follows:

$$R(t) = \begin{cases} 1, & \text{if correctly detect} \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Besides, we adopt the deep neural network (DNN) to approximate the Q network, where the loss function is defined as [8],

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t, a_t) \sim \rho(\cdot)}[R(t) + \gamma max_{a'} Q(s', a'; \theta^n) - Q(s_t, a_t; \theta)^2]. \quad (4)$$

Here, s', a' and θ^n represents next state, next action and the parameter of training DNN in n^{th} iteration, respectively. $\rho(\cdot)$ represents the probability distribution over the state sequences s and the actions a. After training the DNN, we can obtain the optimal policy as,

$$\pi^*(s) = \arg\max_{a'} Q^*(s_t, a_t'; \theta). \quad (5)$$

In the eq. 5, $Q^*(s_t, a_t'; \theta)$ denotes the optimal Q-value.

The workflow of the proposed method is shown as Fig. 2. In this workflow, firstly, we use the label Encoding method: **sklearn.preprocessing.LabelEncoder** [9] to transform the textual information into numerical information. Secondly, we split the data into the training dataset and test dataset by 8:2 [10], where 80% data is used for training and 20% data is leveraged for testing. Thirdly, we use the training dataset to train the proposed method to obtain the optimal policy $\pi_\theta{}^*(s)$.

IV. Experiment Result and Analysis

To evaluate the proposed method, we utilize the dataset that is gathered from the UCI machine learning repository [11], which has 10000 energy data records. Besides, we implement the proposed A3C method on the python platform via multi-threading programming along with the TensorFlow APIs.

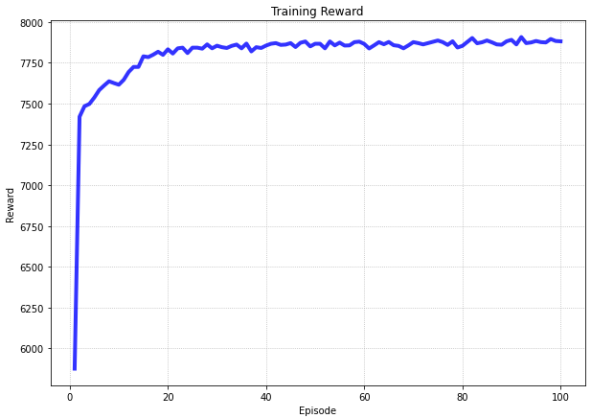


Fig. 3: Reward Of Using Training Dataset

Fig. 3 illustrates the reward of using the training dataset. It can be seen from this figure that the reward first climbs sharply, and around the 20^{th} episode, it converges. In Fig. 4, we compare the result of the proposed method with ground truth. The two sets of data are very close. To further evaluate the proposed method, we list the confusion matrix in Fig. 5. Through this figure, we can get the following terms [12]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{721 + 1266}{721 + 1266 + 5 + 8} = 99.35\%. \quad (6)$$

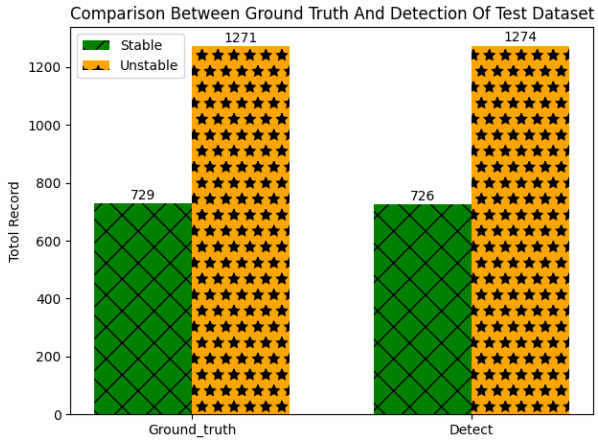


Fig. 4: Comparison Between Ground Truth And Detection of Test Dataset

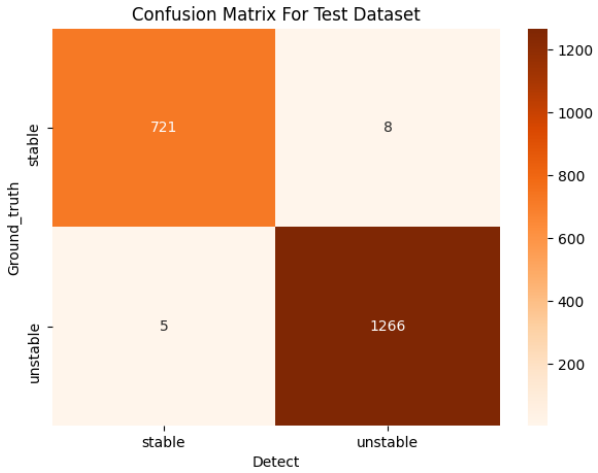


Fig. 5: Confusion Matrix for Test Dataset

$$Precision = \frac{TP}{TP + FP} = \frac{721}{721 + 5} \approx 99.31\%. \quad (7)$$

$$Recall = \frac{TP}{TP + FN} = \frac{721}{721 + 8} \approx 98.9\%. \quad (8)$$

$$F1_Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 99.31\% * 98.9\%}{99.31\% + 98.9\%} \approx 99.1\%. \quad (9)$$

TABLE I: Comparison Among Various Split Ratio Of Test Dataset

Split Ratio	Accuracy	F1 Score
6:4	94.05%	91.08%
7:3	99%	98.6%
8:2	**99.35%**	**99.1%**

Through analyze the aforementioned terms: accuracy, precision, recall and F1-Score, we can induce that the proposed method can achieve significant gain. Specifically, detecting the smart grid stability with the proposed DRL method can achieve 99.35% accuracy, and the precision can reach 99.31%. As for recall and F1-Score, our proposed method can get 98.9% recall and 99.1% F1-Score, respectively. Besides, we compare the accuracy and F1 Score among different split ratio of test dataset, shown as Table I. This table shows split test dataset by 8:2 can achieve the highest accuracy to 99.35% and F1 score to 99.1%, when comparing with a split by 6:4 and 7:3.

V. Conclusion

Smart grid is an important application of the CPS which is responsible for managing energy intelligently. However, it strongly relies on the communication technology, and also, the energy generation and demand are random over time, which results in the smart grid system become vulnerable and unstable. Hence, it is necessary to design an intelligent smart grid stability detection scheme. To this end, this paper introduces a container-based edge computing-assisted deep reinforcement learning approach for intelligently detecting the stability of the smart grid. The evaluation results show the proposed method can achieve high accuracy to 99.35% for the test dataset. Other terms: recall, precision, and F1-score are also very high. As a result, we can induce that the proposed method is efficient and can easily adapt to new data.

References

[1] F. Darbandi, A. Jafari, H. Karimipour, A. Dehghantanha, F. Derakhshan, K. R. Choo, "Realtime stability assessment in smart cyberphysical grids: a deep learning approach," *IET Smart Grid*, May 2020.

[2] U. J. Patel, N. G. Chothani and P. J. Bhatt, "Distance Relaying with Power Swing Detection based on Voltage and Reactive Power Sensitivity," *International Journal of Emerging Electric Power Systems*, vol. 17, November 2015.

[3] M. R. Salimian, F. Haghjoo and M. R. Aghamohammadi, "Out of Step Detection and Protection Using Online WT," *International Electrical Engineering Journal (IEEJ)*, vol. 4, pp. 1133-1139, 2013.

[4] S. M. Hashemi and M. Sanaye-Pasand, "Current-Based Out-of-Step Detection Method to Enhance Line Differential Protection," *in IEEE Transactions on Power Delivery*, vol. 34, no. 2, pp. 448-456, April 2019, doi: 10.1109/TPWRD.2018.2873698.

[5] R. Torre et al., "Enhanced Driving with 5G: A New Approach for Alleviating Traffic Congestion," *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Dallas, TX, USA, 2019, pp. 1-2, doi: 10.1109/NFV-SDN47374.2019.9040009.

[6] V. Divya and R. S. Leena, "Docker based Intelligent Fall Detection using Edge-Fog Cloud Infrastructure," *in IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2020.3042502.

[7] B. Schfer, C. Grabow, S. Auer, J. Kurths, D. Witthaut and M. Timme , "Taming Instabilities in Power Grid Networks by Decentralized Control," *The European Physical Journal Special Topics*, vol. 225, no.3, pp. 569-582, 2016.

[8] L. Zou and C. S. Hong, "Smart Energy Scheduling of B2V-enabled Prosumer Community as EVSE Based on SDN Network: A Deep Reinforcement Learning Approach," *Korea Software Congress 2020*, pp. 234-236, December 2020 (in Korea).

[9] sklearn.preprocessing.LabelEncoder, [Online] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (Visited on March 31, 2021)

[10] E. Alajrami et al., "Handwritten Signature Verification using Deep Learning," *International Journal of Academic Multidisciplinary Research (IJAMR)*, December 2019.

[11] Electrical Grid Stability Simulated Data Data Set," *UCI Machine Learning Repository*, [Online] https://archive.ics.uci.edu/ml/datasets/Electrical+Grid+Stability+Simulated +Data+ (Visited on 30 March, 2021)
[12] W. Zhang, H. Liu, W. Wu, L. Zhan and J. Wei, "Mapping Rice Paddy Based on Machine Learning with Sentinel-2 Multi-Temporal Data: Model Comparison and Transferability," *MDPI*, May 2020.

Evaluation of BBRv2 and PCC Vivace algorithms over KREONet

Tergel Munkhbat[1,2], Buseung Cho[1,2]
[1]Korea Institute of Science and Technology Information, Korea
[2]University of Science & Technology, Korea
{tergelmunkhbat, bscho}@kisti.re.kr

Abstract

A significant concern for research institutes is exchanging terabytes and petabytes of data between geographically distributed locations in a short amount of time. However, the fundamental difficulty with this transfer results from the Congestion Control (CC) algorithms. Recently, several new methods have been proposed to address congestion. Hence, this paper determines whether BBRv2 and PPC Vivace algorithms, which are state-of-art CCs, fulfill the requirement of large-data transmission. The key difference of this paper to others in the field is that our evaluation has been conducted over production level high-speed networks (100G) rather than evaluating in the simulation environment because high-speed networks have characteristics that are difficult to emulate. Our results reveal these CC algorithms still have drawbacks to satisfy the demands of large-data transmissions. BBRv2 results present an immense number of packet losses even though BBRv2 includes packet loss as an input. PCC Vivace overestimates the optimal point and doubles the minimum delay in some situations. This paper suggests that there is still a gap in all CC algorithms to satisfy the high-performance network transmission.

Ⅰ. Introduction

One of the biggest challenges the Internet has faced today is to transfer tens of petabytes of data that was unprecedented in the past. Interestingly, the challenge results from the initial design philosophy of the current Internet technology [1]. Around 1980, there were two ideas to ensure reliability in the case of failure. The first was at the gateway, where the intermediate packet switching nodes could protect from congestion and loss. However, it was complex to implement because every state information should be stored and replicated in the intermediate devices. The second method, which was selected, was at the source. In other words, senders speculate about whether the rate will be cut or increase to prevent congestion.

The default senders' decisions of rates are based on the presumption that packet loss events are congestion called loss-based congestion control (CC) algorithms. A study [2] for data-intensive science assured, however, that even a tiny number of packet losses makes a negative impact on throughput as the latency increase for loss-based CCs (such as CUBIC TCP [3]). Hence, researchers are actively seeking the panacea for network congestion under all networking circumstances.

Several CC design philosophies have been proposed for the last decade, such as hybrid mechanisms or learning-based approaches. For instance, the most modern design in the hybrid CC algorithm is BBRv2 [4], which combines rate-based and model-based approaches. This algorithm constructs a model of an end-to-end path by measuring bandwidth, the RTT, and the packet loss rate. Moreover, Performance-oriented Congestion Control (PCC) Vivace is learning-oriented design philosophy [5]. PCC Vivace uses linear regression techniques to adjust its sending rate. Various researchers have already evaluated these two algorithms in terms of performance and fairness. In particular, many authors attempted to answer whether these algorithms are feasible to disrupt the traditional loss-based congestion mechanism [6-13]. However, most of these authors [6-11, 13] evaluate these two algorithms in the emulation environment (emulating latency, loss, bottleneck bandwidth, buffer size, and so on). Even though the logic behind these algorithms can be seen through the emulation tests, today's networks are becoming much more complicated and difficult to emulate due to their various BDPs, unknown intermediate devices, the use of different queueing algorithms, and so on. Although other researchers [12] evaluate them in the real-world environment, such as on the Internet, such evaluations are not the sensible approach either. It is difficult to know how the background traffic influences the experiments (i.e., a burst transmission could affect by an anonymous user). Thus, there is still a gap in the evaluations of these algorithms.

In this paper, BBRv2, PCC Vivace, CUBIC, and HTCP [14] algorithms are evaluated over high-speed (100G) KREONet networks, which is the national science and research network of Korea. The key difference of our evaluation with others is that the background traffic of the whole end-to-end path is explicitly measured through SNMP and Argus flow collector [15]. Such experiments help to recognize and distinguish the effect of the background traffic. Moreover, another difference is our experiment carries over some R&E flow characteristics such as jumbo frames and elephant flows. As a result, this

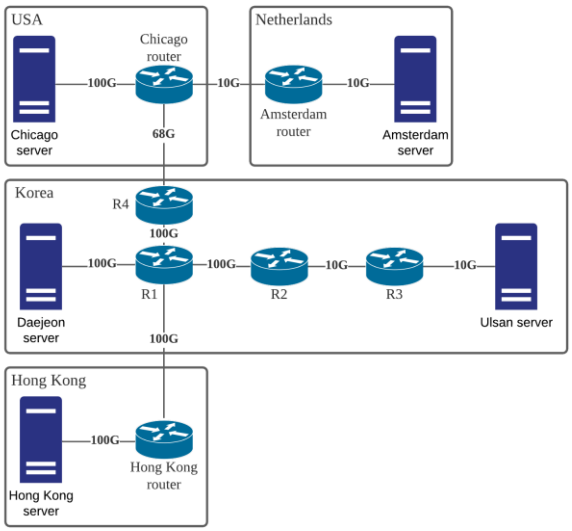
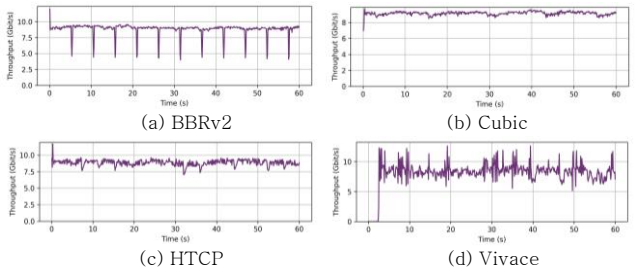


Figure 1. Network topology for evaluations

Table 1 Minimum RTT and average background traffic of bottleneck links from Daejeon to the destination nodes

	RTT min	Average background traffic
Ulsan	5.1 ms	175.3 Mbps
Hong Kong	39.3 ms	1054.6 Mbps
Chicago	154.5 ms	2150.5 Mbps
Amsterdam	264.1 ms	489.8 Mbps



(a) BBRv2 (b) Cubic

(c) HTCP (d) Vivace

Figure 2. Throughput behavior to Ulsan server

paper gives an insight into which algorithm is better to transfer tens of petabytes of data among these state-of-arts algorithms.

The rest of the paper is organized as follows. *Section II* describes the experimental setup to evaluate these algorithms, and *Section III* presents experimental results. *Section IV* concludes the results.

Ⅱ. Experimental environment

To evaluate these CC algorithms, the network topology, shown in Figure 1, was used. This topology consists of five servers and seven core routers located in different sites around the world. The sender node located in Daejeon is directly connected to the KREONet backbone network with a 100Gbps link. The others are receiver nodes directly connected to the KREONet domestic and international networks with 10Gbps and 100Gbps links, as shown in Figure 1. For Ulsan and Amsterdam servers, 10Gbps bottleneck links exist. All the destination sites have different RTTs, packet loss rates, and background traffic, as displayed in Table 1. The Argus flow collector and SNMP protocol provide detailed inspection over the bottleneck links, and they help to be aware of the effect of the background traffic.

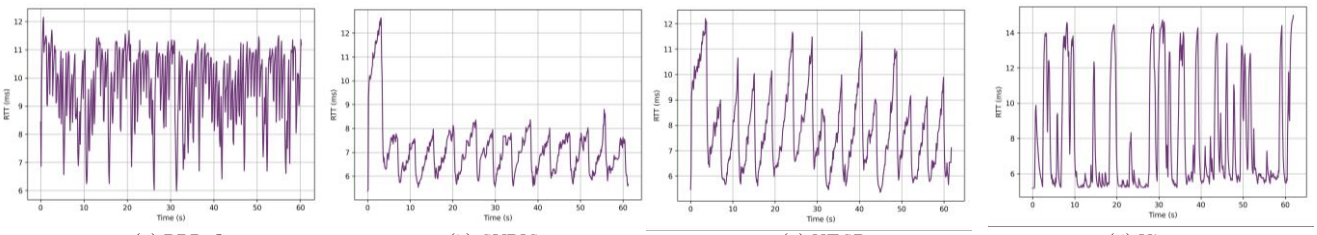
Sender server run on the Supermicro X11DPU board that is equipped with two Intel Xeon Gold 6226R CPUs and Mellanox ConnectX-5/VPI 100G NIC with EM. The operating system of the sender is Ubuntu 18.04 running kernel 5.10 including the implementation of BBRv2alpha-2021-01-15 [16] and PCC Vivace [17, 18]. In contrast, the receivers are Centos 7.7 and use kernel 3.10. Moreover, the maximum limit of the buffer size was increased to 2G, and the intermediate devices are all Cisco Nexus Series switches. The tool used to measure throughput performance and packet loss is *iperf3*, and RTT assessment was gathered by *TCPlog* [19] at the sender side. Each experiment was repeated 15 times and lasted for 300 seconds, and all experiments use jumbo frames.

Ⅲ. Results and analyses

A. Behavior of the CC algorithms

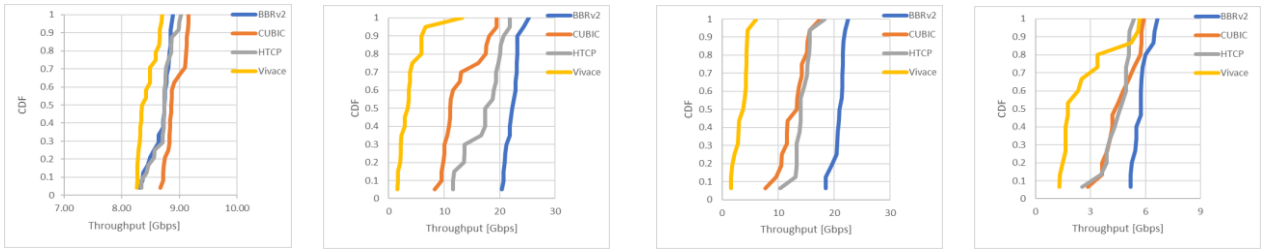
The throughput and the RTT behaviors of these algorithms are illustrated in Figure 2 and 3, where shows only the first 60 seconds of a single run of the Ulsan test for clarity. The average throughput of all four algorithms is similar to around 9Gbps. The path of the Ulsan destination is quite clean-path (packet loss rate is low). Hence, there is no sudden drop for the loss-based algorithms such as CUBIC and HTCP (Figure 2b and 2c, respectively). The average throughput of BBRv2 is slightly lower than loss-based CC algorithms due to the ProbeRTT phase, which is used to measure the minimum RTT of the end-to-end path for BDP calculation. Hence, BBRv2 enters this phase every 5 seconds unless there is a newly measured RTT that is lower than the last minimum RTT of the ProbeRTT stage. Moreover, during this phase, the current throughput is set to half of the measured BDP to reduce the buffer usage, as shown in Figure 2a. For the PCC Vivace, its utility function overestimates the optimal point of throughput. Moreover, it sticks around two seconds to reach the optimal throughput during the startup phase (Figure 2d), and the overall throughput is slightly lower than the other three algorithms, around 8.7Gbps.

Interestingly, Figure 3 shows the highest average RTT among these algorithms depicted by BBRv2, which is 9.7ms. BBRv2 RTT increases up to about 11ms except for the initial startup behavior because the behavior of loss-based algorithms fills the buffer, and the BBRv2 RTT is influenced by loss-based background traffic. Furthermore, the RTT drops to about 6-7 ms during the ProbeRTT phase, as displayed in Figure 3a. CUBIC shows the lowest RTT, around 6.8ms. Figure 3b shows the CUBIC additive-increase/multiplicative-decrease (AIMD) behavior, and it fluctuates between about 6ms and 8ms because of packet losses except the initial stage. HTCP has a similar behavior to CUBIC but fluctuates around 6-11ms (Figure 3c). For the Vivace algorithm, it



(a) BBRv2 (b) CUBIC (c) HTCP (d) Vivace

Figure 3. RTT behavior from Daejeon to Ulsan server



(a) Ulsan (5ms) (b) Hong Kong (40ms) (c) Chicago (155ms) (d) Amsterdam (264ms)

Figure 4. CDF of the throughput

monitors the utility values, and due to its overestimation, RTT reaches up to 14ms, as shown in Figure 3d. However, the overall RTT of Vivace algorithm is lower than BBR.

In summary, when the loss rate is low in the network, BBRv2 and PCC Vivace are slightly lower performances than loss-based algorithms. Moreover, the average RTT of BBRv2 and PCC Vivace is higher than loss-based algorithms. It means that loss-based algorithms are still better than the new ones when the packet loss rate is low and the distance is short.

B. Impact of the different distances

This part evaluates the impact of the distance among these algorithms. Figure 4 shows the CDFs of the throughput performance of this effect.

Figure 4(a) displays that CUBIC is better performance than other algorithms when the delay is around 5ms. The bottleneck link capacity is 10Gbps and the background traffic is about 175Mbps during the tests. BBRv2 and HTCP are similar throughputs and greater than PCC Vivace, but slower than CUBIC.

Figure 4(b) shows the result when the lowest delay is about 40ms. There is no bottleneck link, meaning the end-to-end path is 100Gbps. Our RTT results show that there is no significant change in the delay during the test. In that case, BBRv2 presents the highest throughput than others and following by HTCP. The CUBIC algorithm is lower than these two, meaning CUBIC is not a good performance at 100G high-speed and 50ms delay networks. For the PCC Vivace, it is still lower than other algorithms.

Figure 4(c) shows the case when the minimum delay increased to around 154ms, and the bottleneck link is 68Gbps. BBRv2 is more performance than others. Although HTCP is slightly better than CUBIC, their performance is quite similar when CDF percentage increases more than 50%. The reason is that the throughput recovery process depends on the ACK for loss-based CC algorithms. It means that since the RTT is high, this ACK exchange process is slow, leading slow recovery process after congestion is detected. Hence, these loss-based algorithms show similar results. PCC Vivace is still not a good performance in this case.

Figure 4(d) shows the result when the delay increased to 264ms. This path has a bottleneck link 10Gbps, and the average background traffic is about 500Mbps. The throughput of BBRv2 is still better than others. HTCP and CUBIC throughput performance are similar to each other. For the PCC Vivace algorithm, it is slower than others until the CDF percentage is less than 85%.

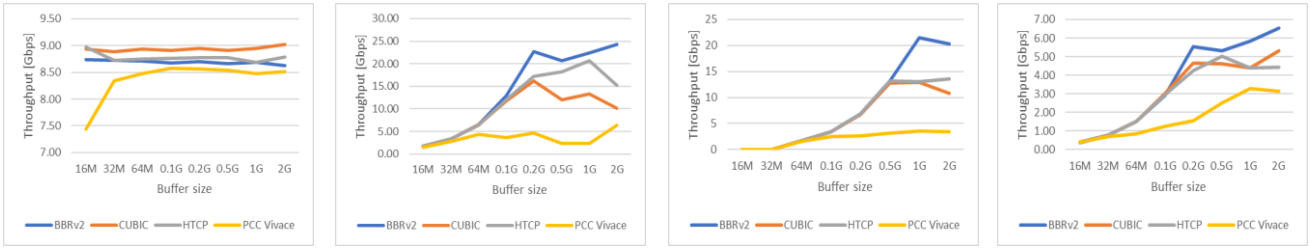
In summary, when the RTT is lower than 10ms, CUBIC is a better performance than BBRv2 and others. As the RTT increases more than 40ms, BBRv2 presents better than the other algorithms. However, PCC Vivace and BBRv2 have higher packet loss rate even though BBRv2 has included packet loss as an input compares to the previous version. This packet loss result will be discussed in the next part.

C. Changing destination buffer sizes and the packet loss rates

The following experiment presents a scenario where these algorithms react to changes in destination buffer size.

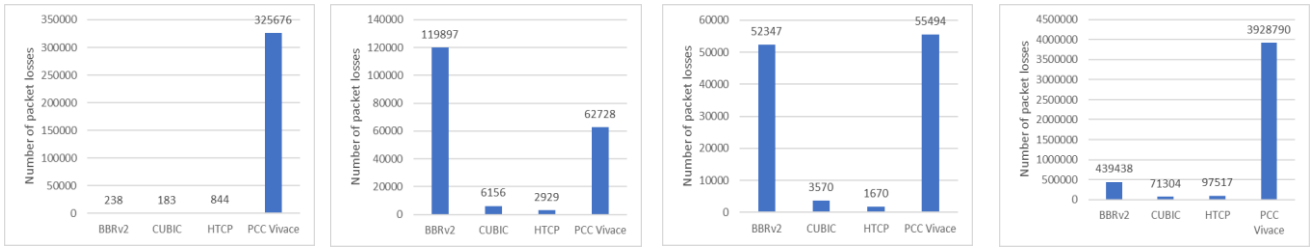
Figure 5 shows the throughput changes according to destination buffer sizes. When the lowest delay is around 5ms and the bottleneck link is 10Gbps, thus BDP is 50Mb (bottleneck * delay). Even though the destination buffer size is lower than BDP, the average throughput is not changed in the Ulsan test (Figure 5a) except PCC Vivace. The reason is that high packet loss occurs (Figure 6a) for PCC Vivace, meaning more packets are sent than other algorithms.

Figure 5b shows when the delay is 40ms and throughput is 100Gbps, meaning BDP is 4Gb. When the destination buffer size is 128Mb (0.1G), the average throughput reduces to 12Gbps, and when the buffer size is 64Mb, the throughput declines to 6.4Gbps. However, a massive number of packet losses



(a) Ulsan (5ms) (b) Hong Kong (40ms) (c) Chicago (155ms) (d) Amsterdam (264ms)

Figure 5. Throughput with various buffer sizes



(a) Ulsan (5ms) (b) Hong Kong (40ms) (c) Chicago (155ms) (d) Amsterdam (264ms)

Figure 6. Number of packet losses

occurs for BBRv2 when the throughput is not limited by the buffer size, as shown in Figure 6b. Most HTCP and CUBIC packet loss occur due to the initial exponential increase. After that, the number of packet losses is low for loss-based CC algorithms.

Figure 5c shows when the delay is 154ms and throughput is also 100Gbps so that BDP is 15Gb. When the destination buffer size is 0.5Gb, throughput declines to 13Gbps. The packet loss number is also high for BBRv2 and PCC Vivace (Figure 6c).

Figure 5d presents the case when the delay is 264ms and bottleneck link is 10Gbps, meaning BDP is 2.6Gb. The number of CUBIC and HTCP packet loss is higher than in previous tests (Figure 6d). The reason is that most of the packet loss occurs at the initial startup stage because when the RTT is high, its convergence time is also longer than short RTTs. For the packet losses of BBRv2 and Vivace, they overestimate the optimal point constantly.

To sum up, even though BBRv2 is better throughput performance when the RTT is high, an immense number of packet losses occur. PCC presents a vast number of packet losses, no matter what the RTT is. For loss-based algorithms, most of the packet loss occurs during the initial stage. After that the packet loss rate is low. Moreover, when the throughput is only limited by destination buffer size, the following equation satisfies our whole various buffer tests.

$$\text{Throughput} = 4 * \text{destination buffer size} / \text{RTT} \quad (1)$$

IV. Conclusion

High-performance data transfer heavily relies on CC algorithms. This paper evaluates recently designed algorithms called BBRv2 and PCC Vivace. Our result shows that when the loss rate is low and the distance is short, CUBIC is better performance than others. As the distance increases, BBRv2 is a higher performance than others. However, a vast number of packet losses are observed even though this version includes packet loss as an input. The reason is that with limited information, such as RTT and packet loss, it overestimates the optimal point of throughput over today's complicated network. For PCC Vivace, it is possible to conclude that it is not designed to transfer big data due to its immense packet losses and low throughput. Moreover, when throughput is only limited by a host buffer size, equation (1) could give an insight into the estimation of throughput. Overall, there is still a gap for the congestion control philosophy among all CC algorithms for high-performance data transmission with a long-distance.

References

[1] Clark, David. "The design philosophy of the DARPA Internet protocols." Symposium proceedings on Communications architectures and protocols. 1988.
[2] Dart, Eli, et al. "The science dmz: A network design pattern for data-intensive science." Scientific Programming 22.2 (2014): 173-185.
[3] Ha, Sangtae, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." ACM SIGOPS operating systems review 42.5 (2008): 64-74.
[4] Cardwell, Neal, et al. "Bbrv2: A model-based congestion control." Presentation in ICCRG at IETF 104th meeting. 2019.
[5] Dong, Mo, et al. "{PCC} vivace: Online-learning congestion control." 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). 2018.
[6] Gomez, Jose, et al. "A performance evaluation of TCP BBRv2 alpha." 2020 43rd International Conference on Telecommunications and Signal Processing (TSP). IEEE, 2020.
[7] Kfoury, Elie F., et al. "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband." Computer Communications 161 (2020): 212-224.
[8] Nandagiri, Aarti, et al. "BBRvl vs BBRv2: Examining Performance Differences through Experimental Evaluation." 2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN. IEEE, 2020.
[9] Song, Yeong-Jun, et al. "Understanding of BBRv2: Evaluation and Comparison with BBRv1 Congestion Control Algorithm." IEEE Access (2021).
[10] Song, Yeong-Jun, et al. "Intra-protocol Convergence Problem in BBRv2's Bandwidth Probing." 2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020.
[11] Fejes, Ferenc, et al. "On the Incompatibility of Scalable Congestion Controls over the Internet." 2020 IFIP Networking Conference (Networking). IEEE, 2020.
[12] Abbasloo, Soheil, Chen-Yu Yen, and H. Jonathan Chao. "Classic meets modern: A pragmatic learning-based congestion control for the Internet." Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. 2020.
[13] Jay, Nathan, et al. "A deep reinforcement learning perspective on internet congestion control." International Conference on Machine Learning. PMLR, 2019.
[14] Leith, Douglas, R. Shorten, and Y. Lee. "H-TCP: A framework for congestion control in high-speed and long-distance networks." PFLDnet Workshop. 2005.
[15] Argus flow collector. Accessed: Dec 11, 2020. [Online]. Available: https://openargus.org/
[16] TCP BBRv2 Alpha. Accessed: Jan 18, 2021. [Online]. Available: https://github.com/google/bbr/blob/v2alpha-2021-01-15/README.md
[17] TCP PCC Vivace. Accessed: Jan 18, 2021. [Online]. Available: https://github.com/PCCproject/PCC-Kernel/tree/vivace
[18] Jay, Nathan, et al. "A PCC-Vivace Kernel Module for Congestion Control." (2018).
[19] TCPlog tool. Accessed: Jan 19, 2021. [Online]. Available: https://git.scc.kit.edu/CPUnetLOG/TCPlog

Optimal Resource Allocation and Association in Space-Aerial Assisted Networks

Nway Nway Ei and Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Korea
email:{nwayei, cshong}@khu.ac.kr

Abstract

Integrating low earth orbit satellites (LEOSat) into the aerial network is a promising solution to provide global coverage and continuous services to the massive ground IoT devices. In this paper, we study a space-aerial assisted network in which UAVs transmit the data collected from the IoT devices to the LEOSats. The optimal UAV to LEOSat association and power allocation problem is formulated to maximize the energy efficiency of the system. Then, the formulated problem is addressed by applying Lagrangian multiplier method. The simulation results are provided to show the effectiveness of the proposed approach.

Keywords – *Low earth orbit satellite, unmanned aerial vehicle.*

I. Introduction

The low earth orbit satellites (LEOSats) have recently gained public attention since they can provide seamless coverage and continuous services to the massive ground IoT devices. However, it is mostly impossible for such energy-constrained IoT devices to directly access to the LEOSats through a long distance. Therefore, the deployment of unmanned aerial vehicle as a relay has become a promising solution due to its reliable communication link with not only ground devices but also with the LEOSats. They can collect the data from the IoT devices and relay it to the LEOSats for further processing. However, it is challenging for the UAVs to determine the association with LEOSats which are in different orbits. Moreover, each UAV needs to decide its optimal transmission power to guarantee its QoS requirement. Hence, in this work, we investigate the optimal resource allocation and association problem in a space-aerial assisted network so that the system energy efficiency is maximized.

The remainder of the paper is organized as follows. In Section II, we present the proposed system model and problem formulation. The simulation results are provided in Section III and we conclude the paper in Section IV.

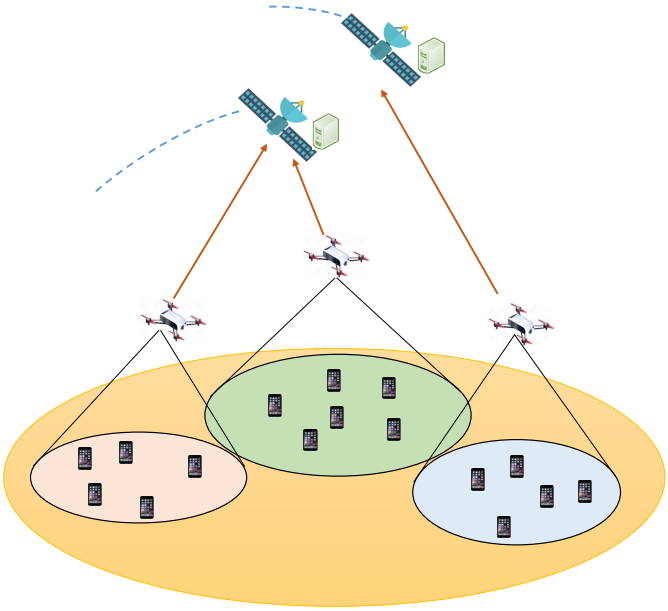


Fig. 1: Space-aerial assisted network.

II. System Model and Problem Formulation

A. System Model

In the proposed system model, as shown in Fig. 1, we consider a space-aerial assisted network with multiple LEOSats and UAVs, where there are a set of unmanned aerial vehicles (UAVs) $\mathcal{M} = \{1, 2, ..., M\}$ and a set of LEO satellites (LEOSats) $\mathcal{S} = \{1, 2, ..., S\}$. Each UAV has the data collected from the ground IoT devices which need to be processed. Here, we assume that the data of IoT devices has already been aggregated at the UAVs and so we do not consider the data transmission between IoT devices and UAVs. Since the onboard processing capability and energy of UAVs are very limited, they further transmit the aggregated information to the LEOSats which have much more resources. Let us define the association variable to indicate whether UAV m is associated with LEOSat s or not as follow:

$$a_{m,s} = \begin{cases} 1, & \text{if UAV } m \text{ is associated with LEOSat } s, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here, we adopt the Rician fading channel model with AWGN to establish the link between UAVs and LEOSats. The channel gain between UAV m and LEOSat s is described as [1]

$$g_{m,s} = \frac{|h_{m,s}|^2}{(d_{m,s})^{\alpha}}, \tag{2}$$

where $h_{m,s}$ is the channel fading coefficient for the air to space link and α is the path loss exponent. $d_{m,s}$ =

$\sqrt{[(H_s - H_m) + R_e]^2 - R_e^2 \cos^2\phi} - R_e \sin\phi$ is the distance between UAV m and LEOSat s, which generally depends on the orbital height $(H_s - H_m)$ and the elevation angle, ϕ between UAV and LEOSat. R_e is the radius of the earth. Then the signal to noise ratio (SNR) of UAV m which transmits data to LEOSat s is given by

$$\xi_{m,s} = \frac{p_{m,s} g_{m,s}}{\sigma^2}, \tag{3}$$

where $p_{m,s}$ is the transmit power of UAV m to LEOSat s and σ^2 is the noise power spectral. We assume that each UAV will orthogonally be allocated fixed bandwidth $w_{m,s}$. The achievable data rate for UAV m to LEOSat s data transmission is given as

$$R_{m,s} = w_{m,s} \log_2(1 + \xi_{m,s}). \tag{4}$$

Therefore, the total throughput of the system can be written as

$$R = \sum_{s=1}^{S} \sum_{m=1}^{M} a_{m,s} R_{m,s}. \tag{5}$$

While considering the power consumption of UAVs, the hovering power is taken into account in addition to the transmission power. Hence, the total power consumption of the system can be expressed as

$$P = \sum_{s=1}^{S} \sum_{m=1}^{M} (a_{m,s} p_{m,s} + p_m^{\text{hov}}), \tag{6}$$

where p_m^{hov} is the hovering power of UAV m which is assumed to be constant in this work. Then we define the energy efficiency of the system (in bit per joule) as the ratio of the total throughput to the total power consumption as follow:

$$\eta = \frac{R}{P} = \frac{\sum_{s=1}^{S} \sum_{m=1}^{M} a_{m,s} R_{m,s}}{\sum_{s=1}^{S} \sum_{m=1}^{M} (a_{m,s} p_{m,s} + p_m^{\text{hov}})}. \tag{7}$$

B. Problem Formulation

The objective is to maximize the energy efficiency of the system and the optimization problem is formulated as follow:

$$\begin{aligned} &\max_{\boldsymbol{A},\boldsymbol{P}} \ \eta \qquad (8)\\ &\text{s.t.}\\ &\quad C1: \sum_{s=1}^{S} a_{m,s} R_{m,s} \geq r_{\text{min}}, \forall m \in \mathcal{M},\\ &\quad C2: \sum_{s=1}^{S} a_{m,s} \leq 1, \forall m \in \mathcal{M},\\ &\quad C3: p_{m,s} + p_m^{\text{hov}} \leq P_m^{\text{max}}, \forall m \in \mathcal{M},\\ &\quad C4: a_{m,s} \in \{0,1\}, \forall m \in \mathcal{M}, \forall s \in \mathcal{S},\\ &\quad C5: p_{m,s} \geq 0, \forall m \in \mathcal{M}, \forall s \in \mathcal{S}, \end{aligned}$$

where C1 ensures the QoS requirement of the UAVs and C2 states that each UAV can associate to at most one LEOSat. C3 guarantees that the total power consumption of each UAV for data transmission and hovering should not exceed its maximum power. The formulated problem is intractable due to the fractional objective function and the binary association variable. To address this problem, the fractional objective function is transformed into an equivalent subtraction form by a non-negative factor η [2]. Therefore, by transforming the fractional objective function into the subtraction form and relaxing the association variable into continuous variable, the problem in (8) is rewritten as follow:

$$\begin{aligned} &\max_{\boldsymbol{A},\boldsymbol{P}} \ R - \eta P \qquad (9)\\ &\text{s.t.}\\ &\quad C1 - C3, C5,\\ &\quad C4: a_{m,s} \in [0,1], \forall m \in \mathcal{M}, \forall s \in \mathcal{S}. \end{aligned}$$

We apply Lagrangian dual decomposition method to solve problem (9) which is the simpler and more tractable form of problem (8). The Lagrangian function of problem (9) can be written as

$$\begin{aligned} \mathcal{L}(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = &\sum_{s=1}^{S} \sum_{m=1}^{M} a_{m,s} R_{m,s}\\ &- \eta \Big[\sum_{s=1}^{S} \sum_{m=1}^{M} \big(a_{m,s} p_{m,s} + p_m^{\text{hov}}\big) \Big]\\ &+ \sum_{m=1}^{M} \theta_m \big(\sum_{s=1}^{S} a_{m,s} R_{m,s} - r_{\text{min}} \big)\\ &+ \sum_{m=1}^{M} \lambda_m \big(1 - \sum_{s=1}^{S} a_{m,s}\big)\\ &+ \sum_{m=1}^{M} \delta_m \big(P_m^{\text{max}} - p_{m,s} - p_m^{\text{hov}}\big)\\ &+ \sum_{s=1}^{S} \sum_{m=1}^{M} x_{m,s} a_{m,s}\\ &+ \sum_{s=1}^{S} \sum_{m=1}^{M} y_{m,s} (1 - a_{m,s})\\ &+ \sum_{s=1}^{S} \sum_{m=1}^{M} z_{m,s} p_{m,s}, \end{aligned} \tag{10}$$

where θ_m, λ_m, δ_m, $x_{m,s}$, $y_{m,s}$, and $z_{m,s}$ are Lagarangian multipliers. Then we can easily derive the Hessian matrix from (10) by differentiating with respect to $a_{m,s}$ and $p_{m,s}$ as H = $\begin{bmatrix} 0 & \frac{(1+\theta_m) w_{m,s} A}{(1+p_{m,s} A) Ln2} \\ \frac{(1+\theta_m) w_{m,s} A}{(1+p_{m,s} A) Ln2} & \frac{-(1+\theta_m) a_{m,s} w_{m,s} A^2}{(1+p_{m,s} A)^2 Ln2} \end{bmatrix}$, where $A = \frac{|h_{m,s}|^2}{(d_{m,s})^{\alpha} \sigma^2}$. Since the Hessian matrix obtained is negative definite, the problem (9) is convex and can be solved by using CVXPY [3].

III. Simulation Results

For the simulation results, we consider 3 LEOSats which are orbiting in different altitudes and the locations of UAVs are randomly generated. To verify the effectiveness of our proposed algorithm, we design a random approach in which

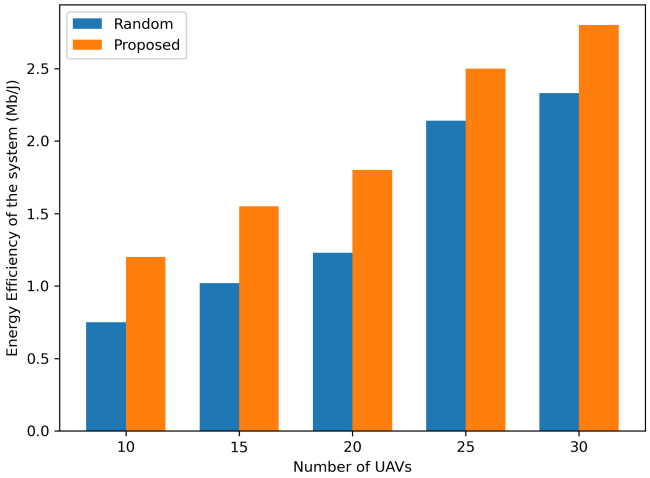


Fig. 2: The energy efficiency vs number of UAVs.

UAVs are randomly associated to LEOSats. As we observe in Fig. 2, the energy efficiency of the system increases with the number of UAVs. However, the energy efficiency of the system in the proposed approach is higher than the random one. The reason is that the UAVs may consume much power by randomly associating the LEOSat through a long distance so that its QoS requirement is met. As a result, the energy efficiency of the system degrades.

IV. Conclusion

In this paper, we introduce a space-aerial assisted network in which UAVs relay the aggregated data from the massive IoT devices to the LEOSats. The optimal UAV-LEOSat association and power allocation problem is investigated to maximize the system energy efficiency. We apply Lagrangian multiplier method to solve the formulated problem. The simulation results are illustrated to show that the proposed algorithm can achieve the maximum system energy efficiency.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1A4A1018607). *Dr. CS Hong is the corresponding author.

References

[1] Zhang, Long and Zhang, Hongliang and Guo, Chao and Xu, Haitao and Song, Lingyang and Han, Zhu, “Satellite-Aerial Integrated Computing in Disasters: User Association and Offloading Decision,” 2020 IEEE International Conference on Communications (ICC), pp. 554–559.

[2] Li, Zhendong and Wang, Ying and Liu, Man and Sun, Ruijin and Chen, Yuanbin and Yuan, Jun and Li, Jiuchao, “Energy efficient resource allocation for UAV-assisted space-air-ground Internet of remote things networks,” IEEE Access, vol. 7, pp. 145348-145362, Oct. 2019.

[3] Steven Diamond and Stephen Boyd, “CVXPY: A Python-embedded modeling language for convex optimization”, vol. 17, no. 83, pp. 1–5, 2016.

C-V2X 사이드링크 모드 4 통신에서 자원 할당을 위한 센싱 기반의 후보 검출법

알리 모인, 김영탁*
영남대학교 대학원 정보통신공학과

alimoin@ynu.ac.kr, *ytkim@yu.ac.kr

Sensing based Candidating for Resource Allocation in C-V2X Sidelink Mode 4

Moin Ali, Young-Tak Kim*
Dept. of Information & Communication, Graduate School, Yeungnam University

Abstract

Over the last few years, the number of connected vehicles is increased rapidly, and it is difficult for current wireless access technologies to satisfy all the requirements (high reliability and low latency) of vehicular networks. The main challenge is the collision probability especially in congested scenarios which make its performance far from the required results. The third generation partnership project (3GPP) designed a standard called C-V2X or LTE-V to support vehicles to everything in Release 14 (Rel-14). C-V2X Mode 4 specially designed for Vehicle-to-Vehicle (V2V) communications using PC5 sidelink interface without any support of cellular infrastructure where vehicles select and manage their resources autonomously. In order to increase the efficiency of C-V2X, we analyze the impact of different parameters on system performance and present a sensing based candidating of resources to allocate resources for the vehicles in future in which we sense the resource through sensing window and choose the candidate resource and broadcast the information of candidate resource to avoid collision and by broadcasting reselection counter value the collision is counter, through this mechanism, we achieve better performance than the compared mechanism, and show results of packet reception ration with different number of vehicle by simulation results.

Ⅰ. Introduction

To realize the goals of the future automobile industry, an immense research is being carried out on cellular-Vehicle-to-Everything (C-V2X) communication to assist numerous services, such as autonomous driving, collision avoidance to provide more effective traffic control and to deter road accidents [1]. In order to provide above mentioned services, information sharing is allowed between Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrain (V2P) and Vehicle-to-Network (V2N). C-V2X can be considered as an enhancemnet of dedicated short-range communication (DSRC) as both of them deal with user equipment (UE) control signaling [2].

In 3rd generation partnership project (3GPP) Release 12 (Rel-12), proximity services for device-to-device (D2D) communication is enabled so that by establishing a direct link between devices, UEs exchange data over short distances. Using this method, long term evolution (LTE) traffic at eNodeB (eNB) can be effectively decreased [3]. In addition, to expand the coverage area for the network UEs act as relays for the delivery of data to nearby devices [4]. According to 3GPP specifications, two modes have been defined for short range D2D communications namely Mode 1 and Mode 2. Mode 1 employs under network-controlled scheduling and Mode 2 operates autonomously without the support of core network. The main motive of these two modes is to expand network range as described earlier and improve the UEs battery life [5].

Two more modes have been defined in C-V2X for transmit resource allocation, namely Mode 3 and Mode 4 where, Mode 3 base station is used to allocate resources for the vehicles in coverage area and in Mode 4 vehicles communicate directly with each other [6], [7]. Both modes can be used together and vehicles will turn to Mode 4 when they will be out of coverage area. For best performance, Mode 4 is supposed to be the default mode and this is the main focus of this paper.

The mainly drwaback of the sensing-based semi-persistent scheduling alogrithm, is that every vehicle selects the resources contiguously every second to avoid half duplex problem. During the selection process vehicles do not announce the location of the resource they select for the next transmission run. Although in SPS, the sensing components helps to lower the probability of packet collision consequently. Since, in SPS the nature of resources is semi-persistent so the collisions lasts for a series of packets if it happens [8].

In this paper, we show that how to reduce collision by using sensing-based Candidating for resource allocation in C-V2X mode 4 and prove it by simultaion results that performance can be enhanced through candidating the resources before their actual usage.

Ⅱ. Related Work

In Release 14, 3GPP mainly focuses on safety related applications and in 15 and 16 it has been working on 5G New Radio (NR)-based V2X whose main purpose is to facilitate advance features of LTE V2X.

Many works propose improvements in SPS. Molina-Masegosa et al. [9] discuss about two important aspects. Firstly, they discuss four different error types in packet delivery in which most of collision happens in safety-critical range. Secondly, they focus that the advantage of SPS decreases as the distance between receiver and transmitter increases, and further in [10] Molina-Masegosa et al. lighten the impact of SPS parameters and show that for best PDR, Keep Resource Probability can be regulated according to traffic load in mobile situations compared to the study by Bazzi et al. [11]. Authors also consider the Reference Signal Received Power (RSRP) through which the resource are filtered for next transmission.

Furthermore, some of the studies which is closely related to proposed work in this paper. In [12] He et al. show that packet collision can be reduced by separating control and data payload and let them carry the information about reservation for each other in linked manner. However, like SPS it is only for next coming control or data payload. In addition, their proposal requires cross layer processing to carry reservation information on data channel. Bonjorn et al. [13] propose to broadcast Reselection Counter value and resolve collision by forcefully changing the RC Value of the vehicles that have same RC values. The deficiency found in this paper is that the resource utilization is 25 percent lesser, and moreover it is difficult to change RC values when it is applied on congested scenarios because there are limited number of values in Reselection Counter.

Similarly, in [14] Jeon et al. propose to broadcast the location of resource instantly before the reselection of resource and similarly in [8] Jeon et al. reserve the resource one second before the actual transmission and broadcast the location of resource in current packet run and show that it achieves higher performance, specifically in congested scenarios. However, the deficiency in this work is the wastage of resources if passing vehicles in the opposite line and moreover the collision of new vehicles because new coming vehicle may not hear the reservation in formation by the vehicles in the ongoing traffic.

Our proposed sensing-based candidating resource allocation scheme candidates the resources in half of the current packet run and broadcast the reselection counter value of the current packet run and the location of candidate resource and avoid from collision. In this way, there is no wastage of resources in reservation also avoids new vehicle collision problem as well and it fulfill the requirement of high reliability and low latency which is shown in section IV.

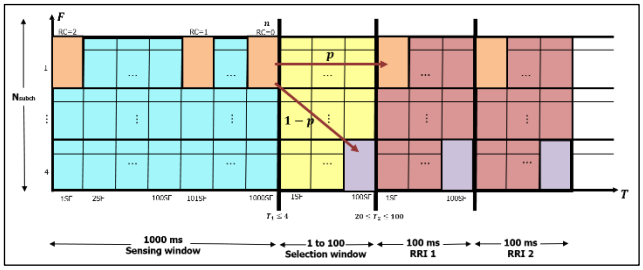


Figure 1: Sensing-based Semi-persistent Scheduling

Ⅲ. Proposed Scheme

In this section, we handle the wastage of resources and consecutive collision probability between vehicles Firstly we introduced the standard SPS mechanism, and secondly we show our proposed scheme.

3.1 Sensing-based Semi-Persistent Scheduling (SB-SPS)

3GPP used SB-SPS as scheduling protocol, for selection of the resources autonomously in C-V2X Mode 4. In this protocol, vehicular user equipment (V-UE) select and reserve subchannel for the series of the consecutive transmissions in frequency domain. Subchannels per subframe totally dependent on data size which have to be transmitted.

Figure 1, shows the mechanism of SB-SPS, where a vehicle at time n select the resources for next packet run by sensing the resources in previous $1000ms$. Resources are selected in the selection window from a list of resources that are not occupied by the other vehicles. Selection window lower bound $(T_1 \leq 4)$ is dependent on V-UE configuration and upper bound is $(20 \leq T_2 \leq 100)$ describes the maximum time allowed for packet inter reception (PIR). In condition of busy resources, V-UE will select the occupied resource with less received power.

After selection of resource, V-UE will do transmission of periodic messages, in specific time interval which is described by resource reservation interval. After using resource for Reselection counter value $(5\ to\ 15)$ the resource keep probability $(0.0\ to\ 0.8)$ mechanism is triggered.

3.2 Sensing-based Candidating Semi-persistent Scheduling (SbC-SPS)

The main goal of our proposed scheme is to enhance the performance of SPS by performing following steps.

- Sensing the Candidate resources and select them for next packet run.
- Broadcast the reselection counter value of current packet run and the location of candidate resource to inform other vehicles to avoid collision and increase reliability.

Initially, vehicles (V-UEs) follow the default SB-SPS mechanism and after first packet run, V-UE will sense the candidate resources and produce a list of resources that are not occupied by others and choose them as their candidate resources. Process of Sensing for candidate resource is shown in Figure 2 as below.

By using candidate sensing window more resources are

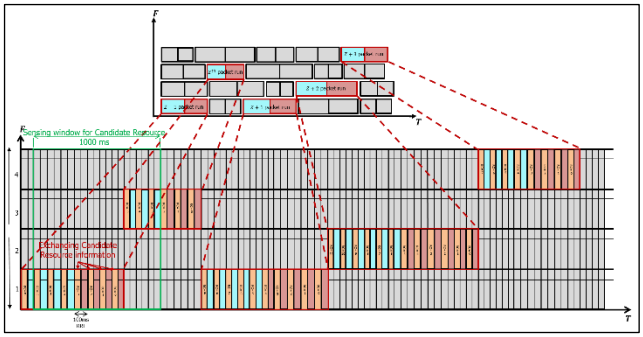


Figure 2: Sensing and candidating of resources in SbC-SPS.

filtered and it helps to avoid packet collision and enhance the performance and more specifically it is increment in default mechanism to enhance the performance therefore in worst conditions if it's not work in any case the default mechanism will continue its working.

The second step of our proposed scheme is to broadcast the Reselection counter value and the location of candidate resource. In current packet when reselection counter value reached at the threshold value ($threshold\ value = RC/2$) it selects the candidate resource and start to broadcast its location and current packet run reselection counter value to inform other vehicles about the candidate resource usage of current vehicle in future and it is shown by Figure 3. The purpose of broadcast the location of candidate resource and reselection counter value after selecting the candidate is to avoid collision. SbC-SPS is described in Algorithm 1.

Algorithm 1: Sensing and candidating the resources in SbC-SPS

Procedure SbC-SPS $(RRI,\ C_1,\ C_2,\ N_{subCh},\ P_{kr})$
  $txSubCh \leftarrow random(1, N_{subCh})$
  $txSubFr \leftarrow random(1,\ RRI)$
  $RC \leftarrow random(C_1,\ C_2)$
  $RCthreshold == RC/2$
  $T \leftarrow 0$
  ---
  while True **do**
    if $T == txSubFr$ **then**
      $txpkt(txSubCh)$
      if $RC > 0$ **then**
        $txSubFr \leftarrow txSubFr + RRI$
        $RC \leftarrow RC - 1$
        if $RC == RCthreshold$ **then**
          Update_Cand_sensing_window ( )
          Collision_Resolution ()
          if $RC == 0$
            $RC_N \leftarrow CrRC$
            $txSubCh_N \leftarrow CrtxSubCh$
            if $random(0,\ 1) < P_{kr}$ **then**
              $CrtxSubFr \leftarrow txSubFr + RRI \times (RC + 1)$
              $CrtxSubCh \leftarrow txSubCh$
            else
              select_candidate_resource ( )
          else
            Update_sensing_window ( )
    $T \leftarrow T + 1$
  End while
End Procedure

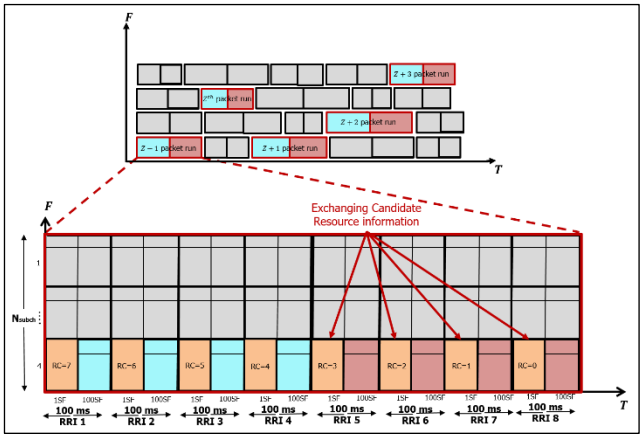


Figure 3: SbC-SPS Packet Run

For collision avoidance Collision resolution algorithm is described in Algorithm 2. In Current packet run, if two or more vehicles after reaching their threshold value, select the same candidate resource then they compare their reselection counter values and vehicle with lesser value receive the resource for future usage and other vehicle selects the candidate resource again. The main reason behind selecting the vehicle with less reselection counter value is that to avoid collision, wastage of resource and continuity of transmission.

Algorithm 2: Collision Resolution Algorithm

Procedure Collision_Resolution $(RC, N_{subCh}, txSubCh,\ txSubFr, RRI, C_1,\ C_2,\ N_{Veh},\ Veh)$
  // candidate resource parameters
  $CrtxSubCh \leftarrow random(1, N_{subCh})$
  $CrtxSubFr \leftarrow txSubFr + RRI + RCrandom + (1,\ RRI)$
  $CrRC \leftarrow random(C_1,\ C_2)$
  ---
  while True **do**
    for $(V_{ID1} \leftarrow 0; V_{ID1} < N_{Veh}; V_{ID1} + +)$ **do**
      $txSubCh_1 \leftarrow Veh[V_{ID1}].txSubCh$
      $txSubFr_1 \leftarrow Veh[V_{ID1}].txSubFr$
      $CrtxSubCh_1 \leftarrow Veh[V_{ID1}].CrtxSubCh$
      $CrtxSubFr_1 \leftarrow Veh[V_{ID1}].CrtxSubFr$
      $RC_1 \leftarrow Veh[V_{ID1}].RC$
      for $(V_{ID2} \leftarrow 0; V_{ID2} < N_{Veh}; V_{ID2} + +)$ **do**
        if $(V_{ID1} == V_{ID2})$ **then**
          Continue
        $txSubCh_2 \leftarrow Veh[V_{ID2}].txSubCh$
        $txSubFr_2 \leftarrow Veh[V_{ID2}].txSubFr$
        $CrtxSubCh_2 \leftarrow Veh[V_{ID2}].CrtxSubCh$
        $CrtxSubFr_1 \leftarrow Veh[V_{ID1}].CrtxSubFr$
        $RC_2 \leftarrow Veh[V_{ID2}].RC$
        if $CrtxSubCh_1 == CrtxSubCh_2$ **and** $CrtxSubFr_1 == CrtxSubFr_2$ **then**
          if $RC_1 < RC_2$ **then**
            $RC_1 \leftarrow CrRC$
            $txSubCh_1 \leftarrow CrtxSubCh$
            $txSubFr_1 \leftarrow CrtxSubFr$
          else
            $RC_2 \leftarrow CrRC$
            $txSubCh_2 \leftarrow CrtxSubCh$
            $txSubFr_2 \leftarrow CrtxSubFr$
      End for
    End for
  End while
End procedure

IV. Simulation Results

Simulations are conducted on NS-3 simulator. The parameters of simulation are shown in Table 1. Figure 4 shows the packet reception ratio (PRR) when the number of Vehicles (V-UE) increases. The results indicate that the proposed method has higher value of PRR for similar congestion scenario compared to the C-V2X Sim algorithm. However, higher congestion decreases the PRR. The improvement in PRR is due to collision avoidance in proposed method.

Simulation Parameters	
Number of V-UEs	0...250
Base distance	150m
Message Size	190 bytes
Transmission power	23dBm
Resource reservation period	100ms
Modulation and Coding Scheme	20
Channel bandwidth	20 MHz
RBs per subchannel	10
Number of subchannels	10

Table 1: Simulation Parameters

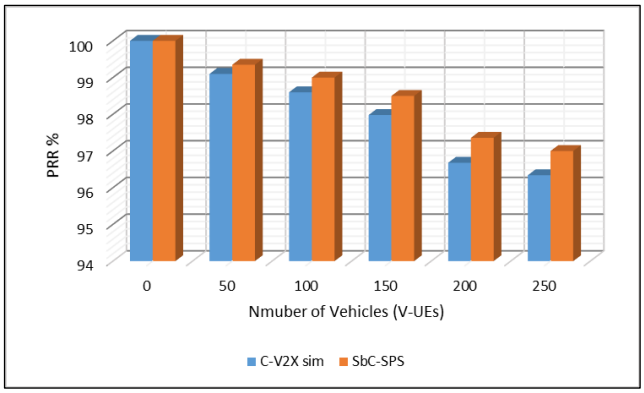


Figure 4: Packet Reception Ratio for an Increasing number of V-UEs

V. Conclusion

Collision can be happened through various reasons in semi persistent scheduling such as change of traffic topology, different parameters of SPS. In this paper, we purpose a solution to reduce the collision probability by candidating the resource before actual transmission, for this purpose we set a threshold value where vehicle select candidate resources and secondly vehicle broadcast the value of reselection counter and the location of candidate resource. In the end, we show some simulation result and enhance the performance.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT 연구육성지원사업의 연구결과로 수행되었음. (IITP-2020-2016-0-00313).

References

[1] S. Chen et al., "Vehicle-to-Everything (v2x) Services Supported by LTE-Based Systems and 5G," *IEEE Communications Standards Magazine,* vol. 1, no. 2, pp. 70-76, 2017.

[2] M. Wang et al., "Comparison of LTE and DSRC-Based Connectivity for Intelligent Transportation Systems," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, Sydney, 2017.

[3] E. Fakhfakh, S. Hamouda and S. Tabbane, "Enhanced traffic offloading with D2D communications under noise rise constraint," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, Messina, 2016.

[4] H. Tang and Z. Ding, "Mixed Mode Transmission and Resource Allocation for D2D Communication," *IEEE Transactions on Wireless Communications,* vol. 15, no. 1, pp. 162-175, 2016.

[5] J. Kim, S. Kim, J. Bang and D. Hong, "Adaptive Mode Selection in D2D Communications Considering the Bursty Traffic Model," *IEEE Communications Letters,* vol. 20, no. 4, pp. 712-715, 2016.

[6] "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures (V14.3.0, Release 14), document TR 36.213," 3GPP, 2017.

[7] "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures (V14.3.0, Release 14), document TR 36.312," 3GPP, 2017.

[8] Y. Jeon and H. Kim, "An Explicit Reservation-Augmented Resource Allocation Scheme for C-V2X Sidelink Mode 4," *IEEE Access,* vol. vol. 8, pp. pp. 147241-147255, 2020.

[9] R. Molina-Masegosa and J. Gozalvez, "System Level Evaluation of LTE-V2V Mode 4 Communications and Its Distributed Scheduling," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, Sydney, NSW, Australia, 2017, pp. 1-5.

[10] R. Molina-Masegosa, J. Gozalvez and M. Sepulcre, "Configuration of the C-V2X Mode 4 Sidelink PC5 Interface for Vehicular Communication," in *2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, Shenyang, China, 2018.

[11] A. Bazzi, G. Cecchini, A. Zanella and B. Masini, "Study of the Impact of PHY and MAC Parameters in 3GPP C-V2V Mode 4," *IEEE Access,* vol. vol. 6, pp. pp. 71685-71698, 2018.

[12] J. He, Z. Tang, Z. Fan and J. Zhang, "Enhanced Collision Avoidance for Distributed LTE Vehicle to Vehicle Broadcast Communications," *IEEE Communications Letters,* vol. vol. 22, no. doi: 10.1109/LCOMM.2018.2791399, pp. pp. 630-633, 2018.

[13] N. Bonjorn, F. Foukalas and P. Pop, "Enhanced 5G V2X services using sidelink device-to-device communications," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Capri, Italy, 2018.

[14] Y. Jeon, S. Kuk and H. Kim, "Reducing Message Collisions in Sensing-Based Semi-Persistent Scheduling (SPS) by Using Reselection Lookaheads in Cellular V2X," *Sensors,* vol. 4388, p. 18, 2018.

블록체인 기반 CBDC 아키텍처 및 트랜잭션 키 관리 시스템 설계

한정수, 김정헌, 허강욱*, 전윤서*, 우종수, 홍원기
포항공과대학교, *하나금융
{saw1515, kjheon1118, woojs, jwkhong}@postech.ac.kr
*{kwheo, yunsuh.chun}@hanafn.com

Design of Blockchain-based CBDC Architecture and Transaction Key Management System

Jungsu Han, Jeonheon Kim, *Kangwuk Heo, *Yunsuh Chun, †Jongsoo Woo, James Won-Ki Hong
Department of Computer Science and Engineering, POSTECH
* Hana Financial Group
†Graduate School of Information Technology, POSTECH

요 약

전자상거래와 전자결제 시스템의 등장으로 종이 화폐의 사용이 줄고 있다. 대부분의 종이 화폐가 사라지고 디지털 화폐가 주류가 될 것임은 충분히 예견할 수 있는 사실이다. 이러한 현상은 블록체인 기술 발전과 COVID-19 으로 인해 더욱 가속화 되었다. 중앙은행 디지털화폐(CBDC)가 최근 주목을 받기 시작한 것 역시 이와 같은 이유 때문이다. CBDC는 중앙은행이 발행하는 디지털 형태의 화폐이다. 현재 CBDC는 여러 국가에서 활발하게 연구되고 있으며 실제로 파일럿 테스트 중인 국가도 존재한다. 그러나 기존 CBDC 연구는 주로 필요성과 요구조건에 대한 내용들을 중심으로 진행되고 있으며 세부적인 기술 연구는 부족한 것이 현실이다. 본 논문에서는 CBDC를 구현하기 위한 아키텍처 설계와 필요한 구체적인 기술에 대한 제안을 담고 있다. 또한 기술적인 문제뿐만 아니라 CBDC가 실제 서비스단계에서 정상적으로 동작하기 위한 방안에 대해서도 다루고 있다.

Ⅰ. 서 론

블록체인의 기술 발전, 암호자산의 확산, 달러의 약세화 등을 배경으로 여러 국가에서 중앙은행 디지털화폐(Central Bank Digital Currency, CBDC)의 관심이 커지고 있다[1]. 국제결제은행 (Bank for International Settlements, BIS)는 CBDC를 중앙은행이 직접 책임지며 국가 회계 단위로 표시되는 디지털 결제수단으로 정의하였다[2]. 예를 들어 가치가 일정한 스테이블코인 (Stablecoin)[3]의 경우에는 중앙은행이 직접 발행하지 않기 때문에 CBDC가 될 수 없다. 또한 디지털 달러의 경우에는 한국 은행이 발행을 하더라도, 한국이 책임질 수 있는 회계 단위는 원화 밖에 없기 때문에 CBDC라고 부를 수 없다. 즉, 국가의 중앙은행에서 직접 발행하고 책임질 수 있는 지불 수단이 바로 CBDC 이다.

현재 전 세계 중앙은행 중 80%가 CBDC 도입을 검토하고 있으며, 이들은 다음과 같은 6가지 이유로 CBDC에 관심을 갖고 있다. 첫째는 금융 포용력 때문이다. 현재 종이 화폐를 대신하여 신용카드, 페이앱 등 다양한 방법으로 결제가 이루어지고 있다. 이처럼 현금이 신뢰할 수 있는 결제 수단이긴 하지만, 현금은 점점 접근성과 유용성이 떨어지고 있다. 그렇기 때문에 여러 국가들에서 민간 결제수단을 대체할 수 있고 안정성이 보장된 결제 시스템을 CBDC를 통해 구축하고자 한다. 두번째는 복원력 때문이다. 결제 청산 시스템에서 셧다운이 일어났을 때 기존 시스템들은 단일 장애점 (Single Point of Failure) 문제를 겪고 있다. 블록체인을 기반으로 한 CBDC의 경우 분산시스템을 통해 이런 문제를 해결할 수 있는 기술력을 제공하므로 주목받고 있다. 세번째는 결제수단의 다양성 때문이다. 기존 민간 결제 플랫폼으로 인해 결제 시각 플랫폼에 맞는 QR 코드나 결제 방법을 사용해야 되는 불편함과 복잡성으로 인해 비용이 발생하게 된다. 하지만 특정 기업의 결제 방식으로 통일하기에는 독과점의 문제가 존재한다. 그렇기 때문에 중앙은행이 정하는 표준 시스템이 필요하며 CBDC가 이 문제에 대한 해결책으로 제시되고 있다. 네번째는 역외 거래 때문이다. 국가간의 송금이나 결제 시 복잡한 과정으로 인한 비효율성과 결제 외 비용이 발생하며, 이 과정이 불투명하게 진행된다는 문제가 존재한다. CBDC는 국가간의 표준 제정 및 시스템 호환성을 통해 이 문제를 해결하는 것이 가능하다. 다섯 번째는 추적 가능한 익명성 때문이다. 기존의 현금은 추적이 전혀 불가능 문제가 있어 범죄에 악용될 여지가 크다. 하지만 CBDC는 추적 가능한 익명성을 통해 프라이버시를 보장하면서 범죄 예방에 도움을 줄 수 있다. 마지막으로 재정 이전 때문이다. 예를 들어 COVID-19 상황 같이, 국가 재난 지원금을 금융 취약 계층도 받을 수 있는 시스템이 필요하며 DID (Decentralized Identity) 기반 신원 증명 서비스[4]와 연결되어 적절하고 안전하게 지급 될 수 있는 국가적 시스템이 필요하다. CBDC는 이런 형태의 재정 이전에서 효과적인 수단으로 주목받고 있다.

본 논문에서는 CBDC 구현을 위한 기술적 이슈에 대한 논의와 함께 시스템 아키텍처에 대한 제안을 다루고 있다. 중앙은행과 시중은행, 일반 고객의 계층적 구조에서 각 참여자의 역할과 요구조건에 대한 내용을 다루고 있으며 안전하고 신뢰할 수 있는 CBDC 유통과 거래 과정을 위한 기술적 요소들을 제시하고 있다.

Ⅱ. 관련 연구

1. Blockchain

블록체인 기반 CBDC의 개발을 위해서는 어떤 블록체인 플랫폼을 사용할 것인지를 결정하는 것이 중요한 이슈이다. 블록체인의 핵심 원장(core ledger)은 은행이나 고객 간의 거래 순서를 결정하며 거래 정보가 저장되어

야 하며 위변조가 불가능한 시스템이어야 한다. 또한 다수의 사용자들의 동시 거래를 지원하는 처리량과 짧은 지연 시간 내에 거래가 체결되도록 보장해야한다. 이를 위하여 [5]에서는 R3 Corda를, 그리고 [6]에서는 Hyperledger Fabric을 CBDC를 위한 블록체인 플랫폼으로 택하고 있지만, 이 플랫폼들은 역외 거래(cross border payment)를 구현하기에 적합하지 않다는 문제점이 있다. 이처럼 현재 CBDC 시스템을 위한 규격화된 블록체인 플랫폼이 제시되지 않은 상태이기 때문에 핵심 원장 개발 시 국가별로 상이한 플랫폼을 독자적으로 개발하거나, 기존 블록체인 플랫폼을 변형하여 개발 중인 경우가 대부분이다. 이 문제를 해결하기 위해 [7]은 역외 거래와 다른 블록체인 간의 연결을 지원하는 상호 운용성 프로토콜(interoperability protocol)의 중요성을 강조하고 있다. 본 논문에서는 블록체인 간 커뮤니케이션 기능을 가능하게 하는 인터체인 기반의 CBDC 시스템을 제안하고 있다.

2. Key Management

[8]은 블록체인을 통해 CBDC 트랜잭션이 이루어질 때 CBDC 실제 소유자의 개인키를 이용하여 트랜잭션에 서명하는 방법을 제안하고 있다. 개인키로 서명한 트랜잭션을 검증자(validator)가 공개키를 이용하여 검증하는 방식을 통해서 트랜잭션의 유효성을 검증하게 된다. 이때 만약 개인키가 외부로 노출될 경우 타인이 악의적인 목적을 가지고 임의로 유효한 서명을 생성하여 금전적인 피해를 줄 수가 있다. 그렇기 때문에 개인키를 안전하게 보관하고 관리하는 것이 CBDC 시스템을 구축할 때 중요한 요구사항 중 하나이다. 일반적으로 디지털 지갑이 개인키를 관리하게 되며 공개키 기반 구조(Public Key Infrastructure, PKI)[9]의 다양한 방법들이 제시되고 있다.

PKI 시스템의 보안 자체는 안전하지만 통신 중간 과정에서나 지갑 소프트웨어에서 공격 받아 개인키 유출이 될 가능성은 항상 존재한다. 그렇기 때문에 트랜잭션에 서명 시 다수의 관리자의 동의가 있어야만 서명이 가능한 방식들이 오랜 기간 연구되어왔다. 대표적인 예시가 다중 서명(multi signature)[10]과 다자간 연산(Multi Party Computation, MPC)[11] 방식이 존재한다. 다중 서명 방식의 경우 트랜잭션을 위한 복수의 키를 생성하여 키를 가진 자들이 동시에 서명해야만 트랜잭션의 서명이 유효한 방식을 말한다. 이와 달리 다자간 연산 방식의 경우 하나의 키를 생성하기 위해 서로 신뢰하지 않는 다수가 각자의 입력 값을 공유하지 않고, 암호화된 입력 값을 통해 서명 키를 생성하는 방식을 말한다. 본 논문에서는 CBDC 네트워크의 참여자들이 다자간 연산 방식을 통해 트랜잭션 서명을 생성하는 키 관리 시스템을 제안한다.

Ⅲ. 아키텍처 설계 및 구현

1. Entity

CBDC 시스템 참여자(entity)는 크게 중앙은행, 시중은행, 일반 고객 세가지로 나뉘게 된다. 그림 1 은 CBDC의 상위설계 구조를 보여주고 있다. 중앙은행은 기본적으로 CBDC 발행 및 배정 권한을 갖고 있으며 CBDC 원장(ledger)의 관리와 책임을 갖게 된다. 시중은행은 CBDC의 원활한 유통을 담당하며 일반 고객들의 CBDC 관리를 담당하게 된다. 일반 고객의 경우에는 CBDC 실사용 계층으로 KYC(Know-Your-Customer)[12] 유저와 non-KYC 유저로 나뉠 수 있다. KYC란 은행에서 고객의 신원을 확인하는 것을 말하며 은행 계좌를 통해 CBDC 거래를 하기 위해 KYC 과정이 필수적이다. 이에 반해 현재 사용되는 종이 화폐와 같이 오프라인 CBDC 거래에서는 사용자의 신원을 확인할 수 없기 때문에 KYC 과정없이 CBDC 거래를 하게 된다.

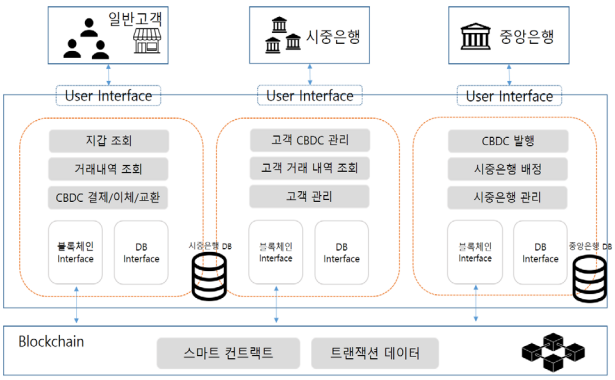


그림 1. CBDC High-level System Design

2. Blockchain

모든 CBDC 트랜잭션 기록은 블록체인 상에 저장된다. 블록체인에 요청을 보내면 이전 트랜잭션 기록을 간단하게 조회할 수 있다. 그렇지만 모든 단순 거래 조회나 검색과 같이 블록체인에 기록할 필요가 없는 CBDC 트랜잭션을 위해 블록체인에 수 많은 RPC (Remote Procedure Call) 요청을 보내는 것은 블록체인 노드의 병목현상으로 인해 지연이 발생시키고, 이는 CBDC 사용자의 불편함으로 이어질 가능성이 높다. 그렇기 때문에 새로운 블록체인에 트랜잭션이 기록되어야 하는 작업과 그렇지 않은 작업으로 나누어 노드의 부하를 최소화 시켜야 CBDC 사용성이 높아진다. 이는 블록체인의 트랜잭션 처리 속도와는 별개로 수 백, 수 천 만명의 CBDC 사용자의 요청에 대해 CBDC 블록체인 노드의 처리량이 감당할 수 있는지에 대한 것이다. 데이터 접근성과 관리의 용이성 측면에서 CBDC 블록체인과 실시간으로 동기화되는 별도의 데이터베이스를 생성하는 것이 가능하다. 즉, 사용자의 CBDC 트랜잭션 요청에 대해 은행 서버의 데이터베이스에서 응답할 정보가 존재할 경우에는 즉각적으로 전달하고, 만약 데이터가 부재한다면 블록체인에 스마트 컨트랙트를 호출하여 사용자에게 전달할 데이터를 받아오는 구조를 제안한다. 자세한 진행 과정은 그림 2 와 같으며 다음과 같은 과정으로 진행된다.

1. 사용자가 CBDC 이체, 교환, 결제, 조회 등의 트랜잭션을 발생시킨다.
2. 시중은행(server)은 사용자의 신원 인증을 요청한다.
3. 사용자는 자신의 신원 인증 정보를 담은 내용을 전달하고 은행은 이를 확인한다.
4.
   a. 시중은행 데이터베이스에 트랜잭션을 처리할 데이터가 존재하는 지 확인하고, 존재할 경우 다음 단계로 넘어간다. 만약 데이터가 없을 경우에는 4-b 와 같이 동작한다.
   b. 블록체인 상의 데이터를 호출하거나 트랜잭션을 발생시키기 위해 스마트 컨트랙트를 호출한다. 블록체인에서는 트랜잭션에 이상이 없는 지를 확인하고, 없다면 블록에 트랜잭션 정보를 기록한다. 시중은행 데이터베이스는 블록체인의 정보

를 실시간으로 동기화하여 동일한 데이터 요청이 있을 경우를 대비한다.

5. 트랜잭션 요청의 결과가 전달된다.
6. 사용자에게 트랜잭션 결과가 보여지게 된다.

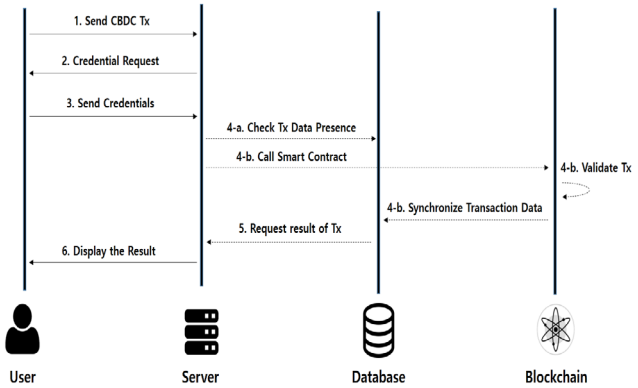


그림 2. CBDC Transaction Workflow

시중은행의 데이터베이스는 트랜잭션의 이상 유무를 확인하지 않으며 블록체인에 저장된 데이터와 동기화되어 응답 지연시간을 줄이기 위한 미들웨어 역할만을 하게 된다. 이를 통해 은행마다 자신들에게 특화된 데이터베이스 스키마를 설계하여 관리가 용이하며, 비교적 데이터 호출 시간이 긴 블록체인 RPC 요청을 최소화 하여 데이터 접근성을 높일 수 있다. 이는 현존하는 블록체인 플랫폼들의 낮은 트랜잭션 처리량(TPS)를 극복하기 위한 방법이다. 하지만 데이터베이스는 블록체인 원장에 비해 쉽게 위변조 될 가능성이 있고, 동기화를 위한 비용이 추가적으로 발생하기 때문에 이 방법은 근본적인 해결책이 아니다.

본 논문에서는 인터체인 기반의 블록체인을 CBDC에 가장 적합한 플랫폼으로 제안한다. 그 이유는 인터체인은 일종의 사이드체인(sidechain) 역할을 할 수 있기 때문에 앞서 말한 트랜잭션 처리량을 노드의 수평적 확장(horizontal scaling)을 통해 향상시키는 것이 가능하기 때문이다. 비트코인과 이더리움과 같이 단일 원장을 갖는 블록체인들과 달리 인터체인은 블록 데이터를 분산하여 저장할 수 있다. 이를 통해 메인체인이 모든 데이터를 저장하며 점점 무거워져 TPS가 낮아지고 트랜잭션 수수료가 증가하는 것을 방지하는 것이 가능하다. 또한 인터체인 기반의 블록체인은 현존하는 타 블록체인 플랫폼에 비해 역외 거래 및 은행 간 송금에 특화되어 있다는 장점이 있다. 기존 블록체인 프로토콜들은 신뢰할 수 있는 제 3 자 없이 상호 정보를 교환할 능력이 없는 폐쇄적인 구조를 띄고 있다. 즉, 비트코인이 이더리움과 거래 내역을 공유할 수 있는 해결책이 부재한다는 뜻이다. [1]은 이런 문제로 인해 국가별로 상이한 CBDC 블록체인 프로토콜로 인해 추후 국가 간의 역외 거래에 큰 장애물이 될 가능성이 높을 것이라는 지적을 하였다. 그렇기 때문에 [13, 14, 15]와 같이 블록체인 간의 통신을 가능하게 하는 인터블록체인을 CBDC 블록체인 플랫폼으로 선정될 가능성이 높다고 예상된다.

[13]과 같이 인터체인은 여러 개의 독립적인 병렬 블록체인인 존(zone)과 각 존을 연결하는 허브(hub)로 구성하는 것이 가능하다. 각 존은 국가, 정책, 사용 목적 등 다양한 특징에 특화된 형태로 독립적인 블록체인 네트워크를 형성하는 것이 가능하다. 허브는 각 존들 간의 트랜잭션이 가능하게 하여 상호 연계되어 동작하도록 한다. 이를 CBDC에 적용시킬 경우 은행 별 상황에 맞는 최적의 블록체인 플랫폼으로 개별 존으로 구현하고 이를 IBC (Inter-Blockchain Communication) 프로토콜을 이용하여 허브로 거래 정보를 전달하여 다른 존의 블록체인과 토큰을 교환하는 것이 가능하게 된다.

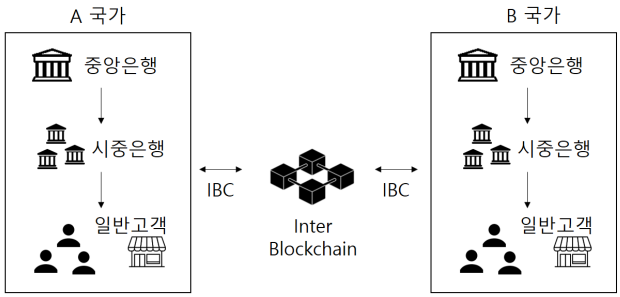


그림 3. 인터체인을 이용한 은행간 거래 시나리오

그림 3 에 볼 수 있듯이 이를 통해 은행 별 CBDC 블록체인 네트워크는 각각 하나의 존을 형성하게 되고 이는 허브를 통해 거래나 토큰 정보를 공유하는 것이 가능하다. 즉, A 국가에서 발행한 CBDC 화폐를 B 국가의 CBDC 화폐로 전환하는 것이 가능하다. 이를 통해 은행이나 환전소에 지불해야할 수수료를 줄이는 것이 가능하며 이중 환전과 같은 불필요한 과정이 사라질 것으로 예상된다.

3. Key management

본 논문에서는 CBDC 트랜잭션 생성 시 다자간 연산(MPC) 방식을 통한 서명 방식을 제안한다. MPC 는 서명 참여자의 민감 정보를 공유하지 않는 암호 기술로 별도의 키 관리 방법을 제시하지는 않는다. 그렇기 때문에 트랜잭션의 민감 정보 보호와 효율적인 키 관리를 위해서 본 논문은 MPC 와 함께 [16]에서 제시한 그룹 키 관리 시스템(Group Key Management, GKM) 을 사용하였다. GKM 은 계층적 구조를 갖고 있는 다수 노드들 간의 효율적인 키 관리에 적합한 방법이다. CBDC 블록체인 네트워크에 참여하는 사용자들은 트랜잭션을 생성할 권한 수준에 차등이 필요하다. 예를 들어, CBDC 를 발행하는 권한은 중앙은행에서만 가져야하며, CBDC 의 원활한 유통과 고객 관리를 위해서 시중은행은 일반고객보다는 더 많은 권한을 가져야 한다. 또한 무분별한 키 생성이 가능할 경우 KYC 과정이 어려워져 범죄에 사용될 가능성이 있다. 그렇기 때문에 CBDC 트랜잭션에 사용될 키 관리를 위한 계층적인 시스템이 필요하다. 그림 4 에서 볼 수 있듯이 CBDC 아키텍처에서 GKM 시스템은 각각 중앙은행, 시중은행, 일반 고객으로 구성되는 것이 가능하다.

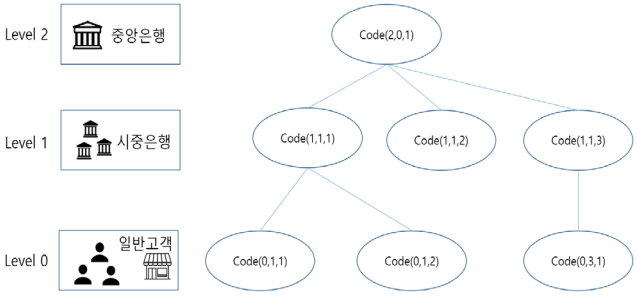


그림 4. Group Key Management in CBDC

각 레벨의 노드는 여러 개의 노드로 구성될 수 있으며, 상위 계층의 노드가 하위 계층의 노드 보다 더 많은 권한을 보유하고 있고 동일 계층에서 노드 간의 권한은 동일한 것으로 가정한다. 각 그룹에 적힌 숫자는 그룹을 식별하기 위한 코드로 Code (i, j, k)로 표기된다. 이 때 i는 자신의 그룹 레벨을 의미하며, j는 상위 그룹에서 부모 그룹의 위치를 의미하고, k는 현재 레벨에서의 자신의 그룹 위치를 의미한다. 예를 들어 (1, 1, 2)의 경우 레벨 1 에 상위 레벨의 첫 부모 그룹의 자식이며 현재 레이어에서 2 번째 위치에 해당하는 것을 의미한다. 부모 그룹이 없을 경우에는 (2, 0, 1)와 같이 j 값은 0 으로 표기한다. $GK_{i,j,k}$ 를Code (i, j, k)를 가진 그룹 키라고 할 때 $c_1, c_2, c_3 \ldots$ 자식 그룹에 대해 그룹 키는 다음과 같이 계산이 된다.

$$GK_{i,j,k} = f\left(GK_{i-1,k,c_1}, GK_{i-1,k,c_2}, \ldots\ldots GK_{i-1,k,c_l}\right)$$

이 때 f는 단방향 함수로 입력 값과 출력 값의 길이는 동일하다. 이를 통해 상위 계층의 그룹 키가 하위 계층의 그룹 키 값에 따라 갱신되게 된다. 그렇기 때문에 하위 계층의 경우 부모 그룹 내 노드 간의 합의를 통해 자식 그룹에 포함될 지 여부가 결정된다면 그룹 키가 배정되고, 이후 부모 그룹 키는 갱신된다. 또한 자식 그룹들의 그룹 키를 계산하는 f에서 MPC 방식을 이용하면 동일 그룹 내에서 그룹 키의 정보를 공유하지 않으며 $GK_{i,j,k}$ 를 만드는 것이 가능하다. 이를 통해 키 관리 시스템을 계층적으로 구현함으로써 트랜잭션 생성 시 계층 수준에 따라 권한을 차등하여 부여하는 것이 가능하게 된다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 인터체인을 이용한 CBDC 아키텍처와 다자간 연산 방식을 활용한 그룹 키 관리 시스템을 제안하였다. 기존에 존재하는 대부분의 CBDC 관련 연구들은 시스템 요구사항이나 범용적인 아키텍처를 제안하는 것에 그치고 있다. 이와 달리, 본 논문은 구체적인 CBDC 시스템 개발을 위한 기술적 방법을 제안하였다. 인터체인이 CBDC에서 갖는 이점에 대하여 설명하였으며, MPC 기반의 그룹 키 시스템을 이용한 안전하고 효율적인 키 관리 방법을 제시하였다.

향후 연구에서는 실제 블록체인을 이용한 인터체인 기반의 CBDC 파일럿 프로젝트 진행할 예정이다. 특히 인터체인의 경우 수평적 확장이 가능한 장점이 있어 실제 CBDC가 상용화 되기 위한 트랜잭션 처리량를 달성하기에 적합하다는 평가를 받고 있다. 중앙은행 및 시중은행 간 거래 시나리오를 가정하여 블록체인 간의 커뮤니케이션의 정량적 성능 평가와 기존 시스템과의 차별점에 대해 연구할 계획이다.

ACKNOWLEDGMENT

본 연구는 2021 년도 하나금융그룹의 지원과 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터 육성지원사업의 연구결과로 수행되었음 (IITP-2021-2017-0-01633*).

참 고 문 헌

[1] Auer, R. A., Cornelli, G., & Frost, J., “Rise of the central bank digital currencies: drivers, approaches and technologies,” CESifo Working Paper, no. 8655, 2020.

[2] Boar, C., Holden, H., & Wadsworth, A., “Impending arrival-a sequel to the survey on central bank digital currency,” BIS paper, no. 107, p. 19, 2020.

[3] Mita, M., Ito, K., Ohsawa, S., & Tanaka, H., “What is stablecoin?: A survey on price stabilization mechanisms for decentralized payment systems,” 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 60-66), July 2019.

[4] Luecking, M., Fries, C., Lamberti, R., & Stork, W., “Decentralized identity and trust management framework for Internet of Things.” 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC 2020), pp. 1-9, Toronto, Canada, May 2020.

[5] Calle, G., & Eidan, D., “Central Bank Digital Currency: an innovation in payments,” R3 White Paper, 2020.

[6] Maharjan, S., Ko, K., Kang, C., Woo, J., & Hong, J. W., "A Study of CBDC Model Applicable for the Current Banking Environment", KNOM Conference 2020, pp. 56-60, 2020.

[7] Qasse, I. A., Abu Talib, M., & Nasir, Q. “Inter blockchain communication: A survey,” ArabWIC 6th Annual International Conference Research Track, pp. 1-6, 2019.

[8] Han, X., Yuan, Y., & Wang, F. Y., “A blockchain-based framework for central bank digital currency,” IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), pp. 263-268, 2019.

[9] Pal, O., Alam, B., Thakur, V., & Singh, S., “Key management for blockchain technology,” ICT Express, 2019.

[10] Ohta, K., & Okamoto, T., “Multi-signature schemes secure against active insider attacks,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 82, no. 1, pp. 21-31, 1999

[11] Yao, A. C., “Protocols for secure computations,” In 23rd Annual Symposium on Foundations of Computer Science, pp. 160-164. 1982.

[12] Kapsoulis, N., et al., “Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture,” Future Internet, 12(2), 41, 2020.

[13] “Cosmos” [Online]. Available at https://cosmos.network/docs/resources/whitepaper.html

[14] “ICON” [Online]. Available at https://docs.icon.foundation/ICON-Whitepaper-EN-Draft.pdf

[15] “Aion” [Online]. Available at https://whitepaper.io/document/31/aion-whitepaper

[16] Pal, O., Alam, B., Thakur, V., & Singh, S., “Key management for blockchain technology,” ICT Express, 2019.

비트코인 노드의 압축 블록 전달 지연 원인 분석

김애리, 주홍택
계명대학교 컴퓨터공학과
AeriKim@stu.kmu.ac.kr, juht@kmu.ac.kr

Analysis of the cause of delay in compact block delivery of Bitcoin nodes

Aeri Kim, Hongtaek Ju
Dept. of Computer Engineering, Keimyung Univ.

요 약

비트코인 노드들은 P2P 네트워크로 연결되며, 새롭게 생성된 블록은 비트코인 네트워크에서 전체 노드로 플러딩(flooding) 방식을 통해 전파된다. 블록은 노드를 거쳐 전체 네트워크로 전파되기 때문에 전체 네트워크로 블록이 전파되는 시간은 노드 간의 전달 시간에 영향을 받게 된다. 노드 간 블록 전달 시간을 단축하면 전체 노드에 블록이 전파되는 시간을 단축할 수 있고 비트코인의 성능이 향상된다. 블록 전달 시간을 단축하기 위해서는 우선 전달 시간을 측정하고 분석하여 지연 원인을 찾아 이를 해결하는 방법을 제시해야 한다. 본 논문은 직접 연결된 비트코인 노드들 간에 압축 블록(Compact Block) 전달 시간을 측정하고 전달 시간 지연 원인을 분석한다. 실험은 한 대의 PC에 Docker 컨테이너를 사용하여 비트코인 노드를 2개 설치하고, 하나는 비트코인 메인 네트워크로, 나머지는 이 노드와 직접 연결하여 두 노드 간 전달 시간을 측정하고 분석한다. 두 노드 간 블록 전달 방식은 현재 비트코인에서 주로 사용되는 고대역폭 압축 전달 블록 방식(CBR: Compact Block Relay)을 따른다. 두 노드의 로그 데이터를 통하여 전달 시간을 계산하고, 전달 과정에서 지연이 발생한 구간을 찾는다. 전달 지연 시간이 큰 블록과 전달 시간이 작은 블록을 선정하여 두 블록의 로그와 BitcoinRPC 명령어를 통해 수집한 블록의 정보를 비교 분석해 전달 지연의 원인을 찾는다.

Ⅰ. 서 론

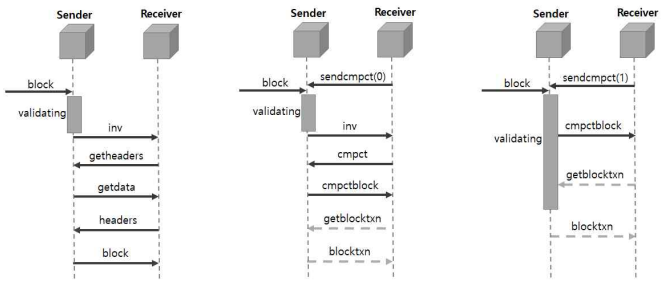
비트코인은 2008년 나카모토 사토시라는 익명의 개발자에 의해 개발된 블록체인 플랫폼이다 [1]. 비트코인 네트워크는 분산형 구조인 P2P 방식을 채택해 동작한다 [2]. 비트코인에서 트랜잭션은 전파되고, 검증된 후에 새로운 블록에 포함된다. 새롭게 생성된 블록은 비트코인 네트워크에 참여 중인 노드로 플러딩(flooding) 방식을 통해 전파된다. 이처럼 블록은 노드를 거쳐서 전체 네트워크로 전파되기 때문에, 전체 네트워크로 블록이 전파되는 시간은 노드 간의 전달 시간에 영향을 받으며, 블록 전달 시간을 단축하면 전체 노드에 전파되는 블록 전파 시간을 단축할 수 있고 비트코인 성능이 향상된다.

현재 비트코인 네트워크의 블록 전달 방식은 기존의 Legacy Protocol과 Matt Corallo가 개발한 Compact Block Relay로 동작한다 [3]. Compact Block Relay 도입을 통해 블록 전달 시간과 대역폭의 감소 등 블록 전달의 효율성을 증대시킬 것으로 예상되었지만, 정확한 성능 향상의 실험 및 분석 결과가 제공되지 않았다 [4].

본 논문에서는 Compact Block Relay 방식에서의 전달 시간을 측정하고, 전달 시간이 지연된 원인을 분석하여 블록 전달 시간을 단축하기 위한 연구에 기초 데이터를 제공한다. 본 논문을 바탕으로 블록 전달 시간을 단축하고 비트코인의 성능을 향상할 수 있다.

Ⅱ. 관련 연구

2.1 배경지식



(a) Legacy Protocol (b) Low Bandwidth (c) High Bandwidth

그림 1. 프로토콜 별 블록 전달과정에서 주고받는 메시지

서론에서도 밝힌 바와 같이 현재 블록 전달 방식으로는 기존에 존재하던 Legacy Protocol과 전달 시간 단축을 위해 개발된 Compact Block Relay 프로토콜이 있다. 그림 1의 Legacy Protocol은 블록을 전달할 때 블록 헤더와 함께 블록에 포함된 전체 트랜잭션을 전달한다. Compact Block Relay 방식은 블록에 포함된 트랜잭션 중에서 수신하는 노드의 메모리 풀에 이미 저장된 트랜잭션은 전달하지 않음으로써 블록 전달을 효율적으로 수행하는 방식이다. 이 방식은 저대역폭 릴레이(Low Bandwidth Relaying)방식과 고대역폭 릴레이(High Bandwidth Relaying)방식이 있고, 기존 연구에서는 60% 이상의 블록 전달에서 코인베이스를 제외하고 전달될 블록에 포함된 트랜잭션들이 수신하는 노드의 메모리 풀에 이미 모두 저장되어 있어 전달되는 블록에 트랜잭션을 채울 필요 없이 즉시 전

달될 수 있음이 확인되었다 [3]. 특히, 고대역폭 방식에서는 그림 1의 (c)에서 보여주는 바와 같이 주고받는 메시지의 절차가 다른 전달 방식에 비해 간단하고, 이전 노드에서 아직 블록의 검증이 끝나지 않더라도 다음 노드로 바로 전달하여 블록 전달 시간이 감소한다 [5].

Compact Block Relay 방식의 전달과정을 자세히 설명하면, 그림 1 (b), (c)의 전달하는 노드(sender)가 cmpctblock 메시지를 통해 블록의 헤더와 블록이 포함하고 있는 트랜잭션 리스트를 보내고, 전달받는 노드(receiver)는 자신의 메모리 풀과 전달받은 트랜잭션 리스트를 비교하여 존재하지 않는 트랜잭션 데이터를 getblocktxn 으로 요청한다. getblocktxn을 받은 Sender는 요청한 트랜잭션을 blocktxn으로 전달해주고, Receiver는 필요한 트랜잭션을 전달받으면 메모리 풀에 저장된 트랜잭션을 합쳐서 블록을 완성한다.

2.2. 관련 연구

비트코인 네트워크에서 블록 전달 시간과 관련하여, 이를 측정 하고 분석하는 논문은 현재 많이 없다. Ryunokuke 등은 압축 블록 전달 방식의 성능 개선 측정 결과를 시뮬레이션으로 보여주고 있으나, 실제 노드 간 전달 시간을 측정한 것이 아니다 [6]. 본 논문에서는 Compact Block Relay 방식에 대하여 실제 비트코인 네트워크 내에 구성된 노드를 사용해 전달 시간을 측정한다. 또한, 전달 지연을 분석하여 블록 전달 시간을 향상할 수 있는 기초 데이터를 제공한다.

Ⅲ. 압축 블록 전달 시간 측정 환경구성 및 방법

3.1. 환경 구성

본 연구에서 사용된 PC는 i5-7500T CPU, 8GB의 메모리를 갖고 있으며, 저장 공간은 SSD 1TB 하드디스크를 사용한다. 운영체제는 리눅스 18.04 LTS 이다. 비트코인 코어를 “Bitcoin Core version v0.20.99.0-30568d3f1” 버전으로 설치하고 최신 블록까지 동기화를 진행한 다음에 블록 전파 시간을 측정한다 [7].

시스템 성능, 시스템 시간 그리고 네트워크 상태 등을 동일한 환경으로 구성하기 위하여 하나의 PC에 Docker 컨테이너를 사용해 두 개의 비트코인 노드를 각각 설치한다 [8]. 그림 2에서 첫 번째 노드(MainNode)는 비트코인 네트워크와 연결하여 일반적인 노드와 같이 8개 이상의 노드들과 연결되고, 두 번째 노드(SubNode)는 첫 번째 노드와 직접 연결한다.

지연 원인 분석의 4.2.1 네트워크 부분에서 추가적인 실험을 진행한다. 같은 PC 4대를 각각 대한민국의 포항, 춘천, 대구, 세종, 총 4개의 지역에 설치하여 같은 PC, 다른 네트워크라는 조건을 주어 네트워크가 전달 시간에 영향을 미치는지 파악한다. 이때 PC의 사양은 ‘AMD Ryzen 7 3700X 8-Core Processor’의 CPU와 ‘32GB RAM’의 메모리, ‘SSD 1TB’의 스토리지를 사용한다. 그리고 블록 높이 656000~656500까지 총 500개를 측정하여 전달 시간을 분석한다.

3.2. 압축 블록 전달 시간 측정

본 실험에서는 블록 높이 654000부터 656000까지 총 2,000개 블록의 전달 데이터를 수집 및 분석하며, 수집한 날짜는 2020년 10월 23일부터 11월 8일까지로 약 17일이다. 하루에 약 125개 정도의 블록 전달이 발생한다. 측정한 블록은 고대역폭 전달과정을 따른다.

그림 2는 노드 구성 방식과 고 대역폭 압축 블록 전달과정을 보여준다.

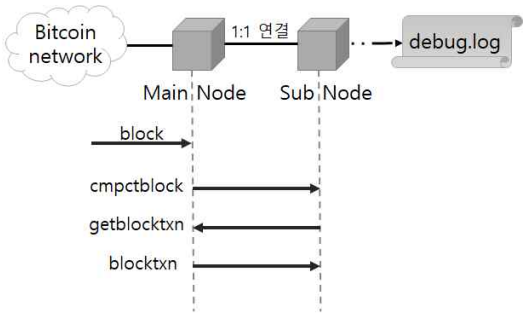


그림 2. 블록 전달 시간 측정 환경 구성도 및 블록 전달과정

SubNode는 블록을 전달받기만 하는 말단 노드에 해당하며 블록 수신 과정을 debug.log라는 이름의 로그 파일에 기록한다. 본 연구에서는 SubNode의 debug.log 파일에 기록되는 로그 정보를 분석하여 블록 전달 시간을 측정하고 분석한다.

로그 기록 시간의 최소 단위를 초 단위에서 마이크로초 단위로 변경하여 정밀하게 시간을 측정한다. 또, 컴팩트 블록(cmpctblock), 네트워크(net) 로그만 출력하게 설정하여 필요한 데이터만 수집한다.

그림 2 하단의 고 대역폭 압축 블록 전달 방식을 보면, 전파의 시작은 cmpctblock 패킷 메시지이다. cmpctblock 패킷은 압축 블록을 전달해주는 메시지로 블록 헤더와 이 블록에 포함된 트랜잭션 리스트를(shortID) 가지고 있다. 블록 전달에 앞서 미리 트랜잭션을 전달받을 수 있기 때문에 블록에 포함된 트랜잭션들 중에서 이미 노드 메모리 풀에 저장되어 있는 트랜잭션이 있을 수 있다. 따라서 트랜잭션 리스트 중 메모리 풀에 없어서 부족한 트랜잭션만을 다시 송신 노드에 요청하여 블록을 완성한다. 이때 사용하는 메시지가 getblocktxn과 blocktxn 이다. getblocktxn 패킷으로 트랜잭션을 요청하고, blocktxn 패킷으로 요청한 트랜잭션을 받는다. 이 두 패킷 사이의 시간은 블록을 완성하는데 필요한 트랜잭션을 요청하는 시간이다.

cmpctblock 패킷을 전달받은 시각부터 blocktxn을 받은 이후 해당 압축 블록을 성공적으로 조립한 시각이 주요 수집 데이터이며, 두 데이터의 시각차가 블록 전달 시간이 된다.

블록 전달 시간을 알아낸 이후에는 블록 전달 시간이 지연되었을 때 영향을 준 원인을 밝혀내기 위하여 블록 전달 시간과 비교할 데이터를 추가로 수집한다.

로그 파일에서는 getblocktxn으로 요청한 트랜잭션 수와 요청한 트랜잭션의 사이즈 데이터를 수집한다. 또한 블록이 노드의 체인에 포함될 때 정보(UpdateTip 로그)도 수집하여 전달 시간이 큰 블록과 작은 블록이 어떠한 차이를 보이는지 분석한다.

BitcoinRPC 명령어 ‘getblock’으로 블록에 대한 19가지의 데이터(해시, 사이즈, 높이, 버전, 트랜잭션 아이디 리스트, 논스, 블록에 포함된 트랜잭션 수, etc.)를 수집한다. 데이터를 전달할 때 일반적으로 데이터의 크기가 전송 시간에 영향을 주기 때문에, 수집한 데이터 중에서 블록에 포함된 트랜잭션 수(nTx)와 사이즈를 중점적으로 분석한다. 다음 절에서는 4장에서 제시하는 분석에 적용되는 분석 방법에 대하여 설명한다.

3.2.1. 상관 관계 분석

상관 분석은 두 변수 간에 어떤 선형적 또는 비선형적 관계를 가지고 있는지 알아보는 분석 방법이다. 두 변수는 서로 독립적인 관계이거나 상관된 관계일 수 있으며 관계의 강도를 상관 계수로 나타낸다. 상관 계수를 계산하는 식은 수식 1과 같다. X와 Y는 상관 관계 분석에 사용되는 두 배열이 값이다.

$$Correl(X,Y)=\frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

수식 1. Correl 상수 계산식

상관 계수가 +1 또는 -1에 가까울수록 두 값이 관계가 있음을 나타낸다. +1은 정비례, -1은 반비례를 의미한다. 보통 ±0.8 정도면 두 값이 연관이 있다고 판단한다. 두 값이 정비례, 또는 양의 값으로 비례한다면 그래프는 우측의 위로 상승하는 그래프를 그리며, 반대의 경우에는 그래프가 우측 아래로 하락하는 그래프를 그린다. 상관 관계 분석을 통해 전달 시간과 지연 원인 요소 간의 상관관계를 밝혀, 해당 요소가 정말 전달 시간에 영향을 미치는지 파악한다.

3.2.2. 주기도(Periodogram) 분석

신호처리에서 주기도는 신호의 스펙트럼 밀도를 추정하는 것이다. 주기도 계산은 다음 수식 2를 이용한다.

$$\mathcal{F}\{x(t)*x(-t)\}=X(f)\cdot X^*(f)=|X(f)|^2.$$

* x(t) : 자기상관 함수인 푸리에 변환 (Fourier transform, FT)
** t : 입력 신호

수식 2. 주기도 계산식

주기도는 신호가 시간에 따라 어느 정도의 특성 변화를 가지는지 나타내는 자기상관 함수인 푸리에 변환을 사용하여 계산한다. 푸리에 변환은 시간이나 공간에 대한 함수를 시간 또는 공간 주파수 성분으로 분해하는 변환이다. 정확히 이산 푸리에 변환과 이산 시간 푸리에 변환이 주기도에 사용된다.

주기도 함수는 MATLAB 툴을 통해 사용할 수 있다. MATLAB는 MathWork 사에서 개발한 수치 해석 및 프로그래밍 환경을 제공하는 공학용 소프트웨어이다 [9]. 그림 3은 주기도를 그리는데 사용되는 함수의 형태이다.

```
[pxx,f] = periodogram(arrivedTime_period,[],[],dataNum);
```

그림 3. 주기도 함수

* arrivedTime_period : 입력 신호
** dataNum : 단위 시간당 샘플 개수
*** pxx, f : 주기의 규모와 빈도 (그래프의 x,y축)

주기도 함수를 사용하여 그려진 그래프는 x축이 빈도, y축이 규모의 값을 갖는다. 단위 시간당 몇 번의 주기를 갖는지 빈도를 이용하여 알 수 있으며, 이를 통해 블록의 전달 시간에 주기성이 있는지 파악한다. 특히 네트워크의 상태와 전달 시간을 비교 분석할 때 사용하여 네트워크가 전달 시간에 영향을 미치는 요소인지 밝힌다.

Ⅳ. 전달 시간 측정 결과와 지연 원인 분석

4.1. 압축 블록 전달 시간

다음은 비트코인 네트워크 내에서 MainNode와 SubNode가 직접 연결되어 있을 때, SubNode에서 블록 전달 시간을 측정한 결과이다. 블록 높이 654000부터 656000까지 총 2,000개를 측정하여 그래프로 표현한 것이 그림 4이다.

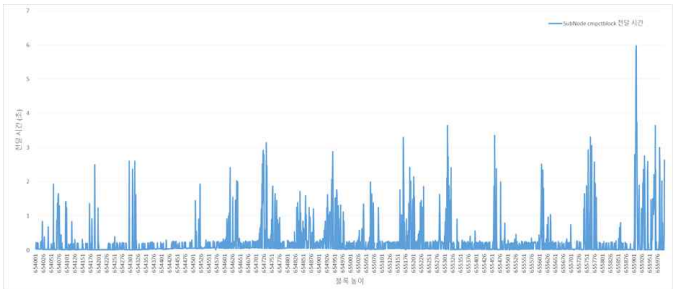


그림 4. SubNode의 블록 전달 시간

그림 4는 블록 전달 시간이 균일하지 않고, 편차가 크다는 것을 보여준다. 정확히 수치로 나타난 것은 표 1이다.

표 1. SubNode의 블록 전달 시간 통계

평균	최소	최대	분산	표준편차
0.266874 초	0.000275 초	5.987413 초	0.311813 초	0.558402 초

평균은 0.266874초이고, 최솟값은 0.000275초, 최댓값은 5.987413초로, 최댓값이 최솟값의 약 20,000배이다. 분산과 표준편차까지 함께 보면 평균에서 변량들이 거리가 멀고, 퍼져 있지 않으며 편차가 크다. 즉, 전달 시간은 균일하지 않고 특정한 블록 전달에서 지연을 발생시키는 원인이 존재한다.

파레토 법칙과 반대되는 긴꼬리는 전체에서 상위 20%를 제외한 값이며, 인터넷의 발달로 경제적, 통계적으로 의미를 갖게 된다 [10]. 전달 시간의 누적 정규 분포 그래프를 그리고 긴꼬리를 찾은 것이 그림 5 이다.

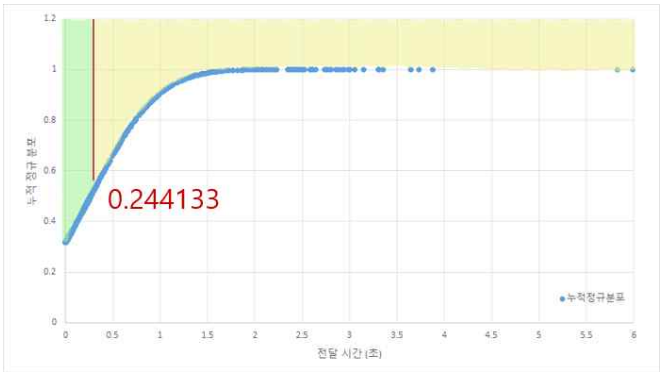


그림 5. SubNode의 블록 전달 시간 누적 정규 분포 그래프

20%와 80%를 가르는 것이 1600번째 값인 0.244133이다. 이 값보다 오래 걸리는 값들이 긴꼬리에 해당하며 긴꼬리 중에서도 마지막 2개의 값, 즉 최대로 걸린 2개의 값을 추출하고 해당 개수에 맞춰 최솟값도 2개를 선택해 전달 시간 지연 원인을 분석할 때 사용한다. 표 2와 표 3이 설명한 값을 나타낸 표이다.

표 2. SubNode에서 전달 시간이 최대로 걸린 블록

순서	블록 높이	전달 시간
1	655909	5.987413 초
2	655908	5.829158 초

표 3. SubNode에서 전달 시간이 최소로 걸린 블록

순서	블록 높이	전달 시간
1	654574	0.000275 초
2	655332	0.000554 초

전달 시간이 균일하지 않다는 것과 지연이 존재한다는 것을 알고, 표 2, 표 3에 나온 최대와 최솟값의 상윗값들이 갖는 차이를 비교 분석하여 왜 균일하지 않은 것인지, 왜 지연이 존재하는지 그 원인을 다음에서 밝힌다.

4.2. 압축 블록 전달 시간 지연 원인 분석

4.1에서 측정한 결과를 바탕으로 블록을 분석하여 전달 시간에 영향을 준 요인이 무엇인지 파악한다.

4.1의 그림 4 SubNode의 블록 전달 시간에 대해 보면 지연 시간이 큰 블록들이 특정 구간에 몰려 있는 양상을 띄고 있다. 해당 구간의 블록 자체에 특징이 있는 것인지, 네트워크 상황에 영향을 받은 것인지 알아볼 필요가 있다.

따라서 분석한 요인은 총 6개로 먼저 네트워크의 영향에 대해 분석한다. 이후에 블록 자체에 특징이 있는 것인지 파악하기 위하여 블록의 체인 연결과 시스템 메모리양, 블록의 사이즈와 트랜잭션 보유 수, 로그의 흐름, getblockTxn으로 요청한 트랜잭션 개수, 요청한 트랜잭션의 사이즈에 대해 살펴본다.

4.2.1. 네트워크

4.1의 그림 4를 보면, 전달 시간이 큰 블록들의 간격이 일정하게 보여 전달 시간 그래프가 주기성을 갖고 있음을 육안으로 확인 가능하다. 이러한 주기성은 네트워크의 영향을 받았을 때에 일어날 확률이 높기 때문에 네트워크가 정말 전달 시간에 영향을 주는 것인지 실험과 분석을 통해 파악한다.

네트워크와 전달 시간의 관계를 알기 위해서, 시간에 중점을 두고 분석한다. 하루 내 특정 시간의 네트워크 상태가 전달 시간에 영향을 주는지 분석하는 것이다.

우선, 그래프의 주기성을 육안이 아니라 주기도를 통해 정확히 한다. 주기도에 사용할 단위 시간당 샘플 개수를 측정하기 위해서, 측정하였던 블록 높이 654000부터 656000까지 총 2,000개 블록의 데이터가 수집된 날짜를 구한다. 날짜는 2020년 10월 23일부터 11월 8일까지로 약 17일 정도 수집한 데이터이며 영국 표준 시간대를 따른다. 하루 내에 주기성을 파악하기 위해서 각 블록을 일일 단위로 자른다.

일일 블록 생성 개수 평균을 구하면 125.2이며, 단위 시간당 샘플 개수를 125개로 정한다. 전달 시간 데이터와 단위 시간당 샘플 개수를 이용하여 그린 주기도 그래프가 그림 5이다.

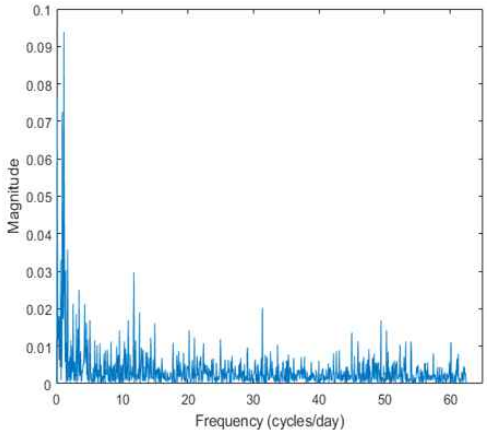


그림 5. 블록 전달 시간 주기도

그림 5의 주기도에서 Peak는 1.09이다. 이는 약 하루마다 한 번씩 주기가 있음을 의미한다. 하루에 한 번 주기가 있다는 사실을 바탕으로 하루 한 번의 주기가 언제인지에 대해서 전달 시간 그래프의 Peak를 분석하여 밝힌다.

그림 6은 4.1의 그림 4의 가로축을 블록 높이가 아니라 시각으로 표현한 그래프이다. 이때, 시간은 영국 표준 시간대이며, 블록을 전달받았을 때의 시각이다. 그림 6에서 전달 시간이 1초 이상인 블록이 있는 구간 총 15개를 선정한다. 선정한 15개의 구간을 파란 점으로 표시하고, 이 중 전달 시간이 최대로 걸린 블록의 높이와 전달 시간, 그리고 받은 시간을 표로 정리한 것이 표 4이다.

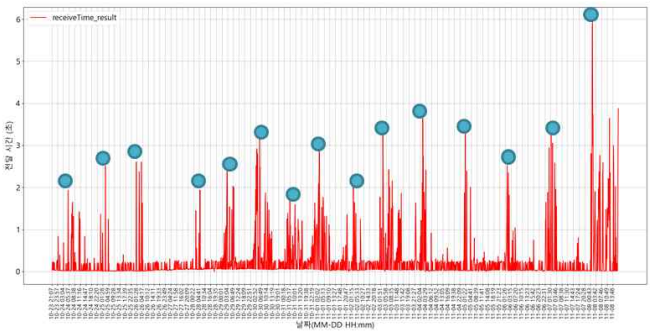


그림 6. SubNode의 블록 전달 시간 (날짜 및 시각)

표 4. 15개의 구간 별 전달 시간이 큰 블록의 높이와 받은 시각

순서	Height	블록 받은 시각 (MM-DD hh:mm)	전달 시간 (초)
1	654058	10-24 05:08	1.932877
2	654190	10-25 03:00	2.504438
3	654299	10-26 00:41	2.60732
4	654524	10-28 05:11	1.934891
5	654619	10-29 02:21	2.415461
6	654734	10-30 05:27	3.151263
7	654841	10-31 05:17	1.714497
8	654945	11-01 02:18	2.887972
9	655065	11-02 01:49	1.998577
10	655169	11-03 03:59	3.302347
11	655310	11-04 02:36	3.644544
12	655460	11-05 01:15	3.356491
13	655608	11-06 01:44	2.523082
14	655763	11-07 01:38	3.311777
15	655909	11-08 01:16	5.987413

표 4의 블록을 받은 시각을 보면 주로 오전 1시에서 5시까지이다. 이는 우리나라, 서울 시간대로 오전 10시에서 오후 2시까지이다. 이는 Internet Rush hour와 연관된다.

대다수의 인터넷 사용자들이 동시에 온라인에 접속하는 기간을 Internet Rush hour 라고 하는데, 이때 네트워크 트래픽이 증가한다. 2017년 Google Analytics report에 따르면 Internet Rush hour 은 오전 9시부터 정오까지이다 [11]. 표 4의 블록을 받은 시각과 겹친다. 즉, 블록 전달 시간이 오래 걸린 블록들은 네트워크 트래픽이 많은 시간대에 있다는 의미이다. 네트워크의 영향을 받는다는 것과도 같다.

이를 정확히 하기 위해, 한가지 추가적인 실험으로 증명한다. 같은 PC 4대를 각각 대한민국의 포항, 춘천, 대구, 세종, 총 4개의 지역에 설치한다. 이러한 설치는 연구에서 같은 PC, 다른 네트워크라는 조건을 주어 네트워크가 전달 시간에 영향을 미치는지 파악 가능하다. 그림 7이 포항, 대구, 춘천, 세종의 블록 전달 시간을 합쳐서 그린 그래프이다.

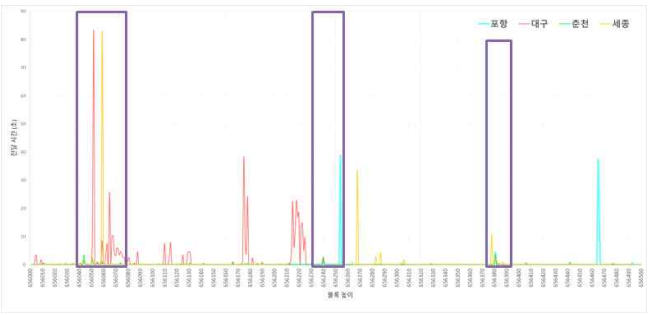


그림 7. 대한민국 포항, 대구, 춘천, 세종의 블록 전달 시간 그래프

블록 높이 656000~656500까지의 블록을 측정한 시간은 2020년 11월 9일 오전 1시부터 11월 12일 오전 6시까지로, Internet Rush Hour가 그림 7 그래프에 총 3번 나타나야 한다.

보라색으로 네모를 친 부분이 각 날짜 별로 Internet Rush Hour에 해당하는 구간이다. 보라색 구간에서 4개 지역의 전달 시간이 급격히 튀는 경향을 보인다. 수치로 판단하기 위하여 전체 500개에 대한 각 지역 별 전달 시간 평균과 보라색 구간, 즉 Internet Rush Hour에서 각 지역 별로 전달 시간 평균을 비교한 것이 표 5이다.

표 5. 4개 지역 별 전체와 Internet Rush Hour 구간의 전달 시간 평균 비교 (단위 : 초)

	포항	대구	춘천	세종
전체 (500개)	0.22695	0.999995	0.045059	0.36076
Internet Rush Hour	0.695844	2.051231	0.188847	1.751719

표 5를 통해서 포항, 대구, 춘천, 세종 각 4개의 지역이 전체의 평균보다 Internet Rush Hour 때에 전달 시간이 2배에서 5배 정도 더 오래 걸림을 알 수 있다. 즉, 전달 시간이 네트워크에 영향을 받는다는 것을 다시 확인 가능하다. 하지만, 그림 6과 그림 7에서도 Internet Rush Hour가 아니지만 전달 시간이 오래 걸린 블록들이 있는데, 이는 네트워크 이외에도 전달 시간에 영향을 주는 요소가 있음을 의미한다. 4.2.2 부터는 네트워크를 제외하고 전달 시간에 영향을 주는 요소가 무엇인지 밝힌다.

4.2.2. 블록의 체인 연결과 시스템 메모리양

SubNode의 로그 중에서 UpdateTip 로그는 새로운 블록을 발견하고 블록체인에 연결했을 때 출력되는 로그로, 블록과 체인에 대한 정보를 담고 있다. 해당 로그를 분석함으로써 블록이 체인에 포함되는 과정에서 전달 시간에 영향을 미치는 요인이 있는지 찾는다.

블록이 체인에 연결될 때 찍히는 로그에는 블록의 해시값, 블록 높이, 마이닝 합의 알고리즘 버전, 블록 완성 시간, UTXO 캐시량, 체인의 전체 트랜잭션 수 등이 있다. 표 6 에서 표 9까지는 4.1에서 선택한 전달 시간의 최소, 최댓값을 갖는 블록들이 체인에 연결될 때 찍힌 로그를 정리한 것이다.

표 6. 전달 시간이 최대로 걸린 블록의 로그 1

New Best	Height	Version
000000000000000000ab489daa30db5df905c5a6e1e919640019c5f12ab5106	655909	0x20000000 (소프트 포크)
Date	**Progress**	**Cache**
2020-11-08T01:15:33Z	1.000000	126.3MiB(941690txo)
Log2_work	**Tx**	**Warning**
92.427036	584797176	73 of last 100 blocks have unexpected version

표 7. 전달 시간이 최대로 걸린 블록의 로그 2

New Best	Height	Version
000000000000000000084f4e693644069c20b72e9f370b326d12040b0532b060	655908	0x20800000 (에이식부스트)
Date	**Progress**	**Cache**
2020-11-08T01:12:45Z	1.000000	125.6MiB(935328txo)
Log2_work	**Tx**	**Warning**
92.427020	584796613	74 of last 100 blocks have unexpected version

표 8. 전달 시간이 최소로 걸린 블록의 로그 1

New Best	Height	Version
000000000000000000005afb341111fab2cf4f97df2068286bebb4ca6fdf191e	654574	0x20000000 (소프트 포크)
Date	**Progress**	**Cache**
2020-10-28T18:44:37Z	1.000000	77.2MiB(540117txo)
Log2_work	**Tx**	**Warning**
92.404128	581807660	69 of last 100 blocks have unexpected version

표 9. 전달 시간이 최소로 걸린 블록의 로그 2

New Best	Height	Version
00000000000000000005c10b1d8159e02e0db27397917d9af63a6cbd232fbb86	655332	0x3fffe000 (에이식부스트)
Date	**Progress**	**Cache**
2020-11-04T05:28:55Z	1.000000	15.4MiB(35478txo)
Log2_work	**Tx**	**Warning**
92.417992	583601592	71 of last 100 blocks have unexpected version

블록 전달 시간이 최소로 걸린 블록과 최대로 걸린 블록의 차이점은 cache 항목에서 나타난다. 최대로 걸린 블록들은 cache의 값이 126.3MiB, 125.6MiB 이고, 최소로 걸린 블록들은 77.2MiB, 15.4MiB로 각 값의 평균은 125, 46이며, 약 3배가 차이 난다.

cache는 메모리에 할당된 UTXO 캐시의 양으로, 블록을 검증할 때 필요한 UTXO를 디스크에서 가져와 메모리에 할당할 수 있을 만큼 할당하여 사용한다 [12]. 블록을 체인에 연결했을 때 당시를 기준으로 찍힌 cache의 양이기 때문에, 해당 블록을 검증하기 위해서 사용한 UTXO들과 해당 UTXO를 할당하기 위해 사용한 메모리의 양이다. 따라서 cache 값이 크다는 것은 시스템 자원을 많이 소모함은 물론, 블록을 검증하기 위한 연산에 걸린 시간이 길 것이다라는 가정을 세울 수 있다. 물론 본 실험에 사용된 고대역폭 압축 블록은 블록을 전달받은 즉시 검증을 시작하여 검증 과정이 끝나지 않고도 블록 조립을 시작하기 때문에 검증 시간이 전달 시간에 영향을 미친다는 가정이 잘못되었을 수 있으나, 블록 검증 시간이 조립에 필요한 시간보다 오래 걸려 지연될 수 있으니 추가로 분석을 진행한다.

위 사실과 가정을 바탕으로 cache 값이 전달 시간에 정말 영향을 미치는 것인지 확인하기 위해 최대, 최소로 전달 시간이 걸린 블록 2개 값을 제외하고 각각 나머지 하위 5개 블록을 찾아 cache를 중심으로 분석한 결과가 표 10, 표 11이다.

표 10. 전달 시간이 최대로 걸린 블록의 cache

Height	Cache	전달 시간 (초)
656000	81.6MiB(575661txo)	0.222576
655911	128.1MiB(955621txo)	0.185435
655969	53.4MiB(345747txo)	0.033455
655310	142.8MiB(1076333txo)	0.227047
655460	135.9MiB(1020125txo)	0.032648

표 11. 전달 시간이 최소로 걸린 블록의 cache

Height	Cache	전달 시간 (초)
655736	106.7MiB(781529txo)	0.000587
655049	110.7MiB(812267txo)	0.000699
655534	72.9MiB(505173txo)	0.001984
655320	151.3MiB(1146012txo)	0.004287
655711	90.1MiB(646032txo)	0.004992

상위 2개의 블록을 봤을 때는 차이가 있었으나, 표 10, 표 11의 나머지 하위 5개 블록에서는 각 평균 cache 값이 107, 105로 차이가 별로 없다. 표 6에서 표 9까지의 분석과 추가로 5개를 더 분석한 것의 결과가 다르다. 그렇다면 측정한 654000부터 656000 블록의 cache 값 전체를 분석하여 cache 값이 전달 시간에 영향을 미치는 것이 맞는지 정확히 한다. 블록 높이 별 그래프로 그린 것이 그림 8, 평균과 중앙값, 최대, 최솟값을 찾아 통계를 낸 것이 표 12이다.

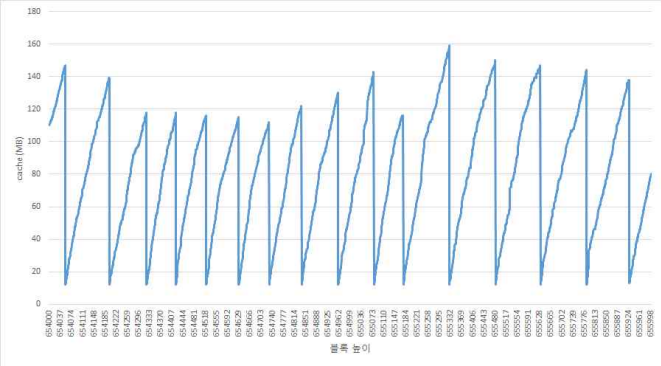


그림 8. 블록 높이 별 cache

표 12. 블록 높이 654000부터 656000까지 cache 값 통계 (단위 : MB)

평균	중앙값	최대	최소
79.556	81	159	12

그림 8을 보면 cache값의 그래프가 규칙적이며 점점 증가했다가 감소하기를 반복한다는 것을 알 수 있다. 그림 8의 그래프를 통해서 UpdateTip 로그의 cache는 해당 블록을 검증하기 위해서 사용한 UTXO라는 추측은 틀린 것이며, 단지 비트코인 노드가 해당 블록을 연결하기 이전의 블록들도 포함하여 블록 검증을 위해 가져왔던 UTXO들이 누적된 cache의 양인 것을 알 수 있다. 따라서 서서히 증가했다가 급감하는 그래프를 그리는 이유는 검증을 위해 UTXO를 차례로 disk에서 메모리로 할당하다가 비트코인 노드가 사용할 수 있도록 정해진 메모리 용량이 가득 차서, 혹은 여러 오류로부터 보호하기 위해 노드 자체에서 정기적으로 Flush 하기 때문이다. 또한 Flush 했을 때, 표 12의 cache 최솟값이 0이 아니라 12인 이유는 모든 UTXO를 Flush 하면 디스크에서 UTXO를 읽어 와야 하고, 그로 인해 검증 속도가 느려질 수 있어서이다.

그렇다면, cache의 값이 단순히 크다 해서 해당 블록의 연산이 많아 전달 시간이 길어졌다는 가정 역시도 틀렸고, 오히려 cache의 양이 적을 때에 디스크에서 UTXO를 읽어 와야하기 때문에 전달 시간이 오래 걸릴 수 있다는 가정이 더 논리적이다.

정기적으로 증감하는 cache의 구간에서 각 최대와 최솟값, 즉 Flush를 했을 때와 Flush하기 직전일 때 블록의 전달 시간을 조사하여 cache와 cache의 Flush가 전달 시간에 영향을 미치는지 분석한다. 해당 블록들을 구분하여 전달 시간의 평균을 구한 것이 표 13이다. 또한, 표 12에서 cache의 중앙값 81을 중심으로 cache의 값이 81일 이상일 때(클 때)와 81미만일 때 (작을 때)를 반으로 나누어 전달 시간 평균을 낸다. 표 14가 그 결과이다.

표 13. Flush 했을 때(cache의 최소)와 하기 직전일 때(cache의 최대) 블록의 전달 시간 평균 (단위 : 초)

Flush 했을 때 (cache 최소)	Flush 하기 직전일 때 (cache 최대)
0.14192	0.381076

표 14. cache의 값이 중앙값 이상일 때(클 때)와 중앙값 이하일 때(작을 때), 블록의 전달 시간 평균 (단위 : 초)

중앙값 이상	중앙값 이하
0.297012	0.236675

표 13에서 Flush를 했을 때, 즉 cache의 값이 최소일 때 오히려 전달 시간이 최대일 때에 비하여 약 2.5배 정도 작다. 따라서 cache의 양이 적을 때 전달 시간이 오래 걸린다는 두 번째 가정 역시도 틀렸음을 안다. 표 14를 통해서도 cache의 값이 크거나 작거나 평균값이 비슷하기 때문에, 현재 메모리에 할당된 UTXO의 cache는 전달 시간에 영향을 미치지 않는다는 것을 알 수 있다. 즉, 블록의 체인 연결과 비트코인 노드가 사용하는 시스템 메모리양은 전달 시간에 영향을 미치지 않는다.

4.2.3. 블록의 사이즈와 트랜잭션 수

Bitcoin RPC인 bitcoin-cli의 getblock 명령어를 사용하여 해시, 사이즈, 높이, 버전, 트랜잭션 아이디 리스트, 논스, 블록에 포함된 트랜잭션 수 등 19가지의 블록에 대한 정보를 받는다.

일반적으로 데이터를 전달할 때, 데이터의 크기나 개수가 클수록 전달 시간이 오래 걸린다. 그렇기에 받아온 정보 중 블록의 크기를 나타내는 'size'와 블록이 포함하고 있는 트랜잭션의 수를 의미하는 'nTx'를 집중적으로 분석하고, size와 nTx가 클수록 전달 시간이 오래 걸린다는 가정을 세운다.

선정하였던 전달 시간이 최대로 걸린 블록과 최소로 걸린 블록 2개를 포함하여 5개를 추가하고 size와 nTx를 알아본 것이 표 15, 표 16이다.

표 15. 전달 시간이 최대로 걸린 블록의 size와 nTx

Height	nTx	size (bytes)
655909	563	1,933,801
655908	563	1,425,866
656000	390	1,775,055
655911	977	1,170,639
655969	570	1,715,906
655310	2034	1,352,233
655460	779	1,587,855

표 16. 전달 시간이 최소로 걸린 블록의 size와 nTx

Height	nTx	size (bytes)
654574	1	366
655332	1	287
655736	1	290
655049	1	280
655534	1	310
655320	1	287
655711	1	338

두 표를 보면 전달 시간이 최대로 걸린 블록의 경우 nTx의 평균이 839개, size의 평균은 1,565,908 bytes이며, 전달 시간이 최소로 걸린 블록의 nTx 평균은 1, size 평균은 308 bytes이다. 해당 결과만 보면 size와 nTx가 클수록 전달 시간이 오래 걸린다는 가정이 맞는다.

하지만 표 15에서 전달 시간이 오래 걸린 블록의 사이즈와 트랜잭션 수가 전달 시간에 정비례하지 않는다. 예를 들어 655909가 가장 오래 걸렸지만 nTx가 563개인 것에 반하여 655310은 nTx 가 2,034개이다. 따라서 두 값의 비례관계를 정확히 하기 위하여 우선 두 값을 블록 높이 별로 같이 그려 그림 9를 통해 보인다.

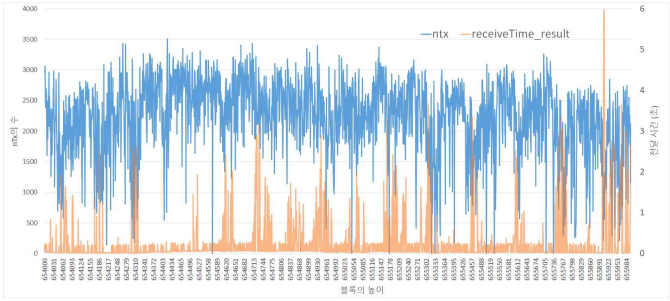


그림 9. SubNode의 블록 별 nTx와 전달 시간

그림 9의 그래프가 전달 시간과 nTx가 겹쳐져 서로 상승하는 그래프를 그리기 기대했으나 그렇지 않다. 특히나 전달 시간이 오래 걸린 655909, 655908 블록 부분에서 전달 시간이 급상승 하는데 비해 nTx 그래프는 하락한다. 그래프로 보기에는 정확하지 않아, 두 값 사이의 관계를 정확하게 판단하기 위하여 분석 방법 3.2.1의 상관 관계 분석 방법으로 상관 계수를 구하고 분포도를 그린다. 상관 분석의 변수로 사용된 것은 전달 시간과 nTx이다. 그림 10은 두 값의 분포도를 보여준다.

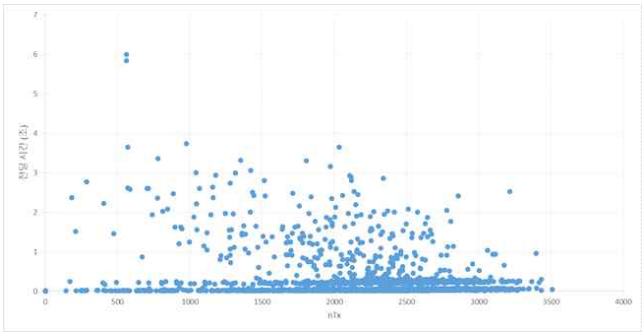


그림 10 SubNode의 블록 별 nTx와 전달 시간 분포도

만약 두 값이 정비례, 또는 양의 값으로 비례한다면 그래프는 우측의 위로 상승하는 그래프를 그려야 한다. 하지만 그림 10의 그래프는 값이 고루 분포되어 있으며 비례하지 않는다. 상관 계수는 -0.21971614로 음(-)의 값을 갖고 있으며, 0.8에도 미치지 않는다. 즉, nTx와 전달 시간은 연관이 없다.

하지만 값 중에는 전달 시간이 적은 값이 많아 상관 분석에 영향을 미칠 수 있다.



그림 11. SubNode의 전달 시간 구간 별 블록의 개수 빈도

그림 11은 SubNode의 전달 시간을 0.5초 단위로 나누어서 구간을 지정하고, 전체 2,000개의 블록 중 몇 개의 블록이 각 구간에 해당하는지 빈도

를 그린 그래프이다. 그림 11을 통하여 0초에서 0.5초 안에 있는 블록의 개수가 많음을 알 수 있다. 따라서 전체 2,000개 블록에 대한 전달 시간의 평균인 0.2초 이하로 걸린 블록은 제외하고 상관 분석을 진행한다. 결과는 -0.49486911 이다. 값 자체는 커졌지만, 여전히 0.8보다 작으며 음의 값으로 전달 시간과 nTx가 반비례 관계임을 의미하고 있다.

nTx와 전달 시간의 관계가 그래프, 분포도로 알 수 없었기 때문에 size와 전달 시간과의 관계는 그래프와 분포도를 그리지 않고, 상관 계수로만 비교해본다. 이 역시 0.2초 이하로 걸린 블록은 제외한 값이다. size와 전달 시간의 상관 계수는 nTx의 상수보다 더 작은 0.070191323 으로 비례관계를 갖긴 하지만 값이 너무 작기때문에 연관이 없다.

따라서 두 결과 모두 처음 세웠던 블록의 size와 nTx가 클수록 전달 시간이 오래 걸린다는 가정과 맞지 않는다.

그림 11을 바탕으로 각 전달 시간을 0.5초 단위로 나누어 각 구간마다 전달 시간과 nTx, size 간의 상관 계수와 nTx, size의 평균을 구하여 정리한다. 그러면 전달 시간과 nTx 및 size가 서로 영향을 미치는지 더 정확히 알 수 있다. 다음 표 17이 이를 정리한 표이다.

표 17. 전달 시간 구간 별
전달 시간과 nTx, Size 상관 계수와 nTx, Size 평균

전달 시간 구간	전달 시간과 nTx 간 상관 계수	nTx 평균	전달 시간과 size 간 상관 계수	size 평균
0 - 0.5	0.172515	2,265	-0.04788	1311656
0.5 - 1	-0.19656	2,197	0.118948	1308239
1 - 1.5	-0.04847	1,903	0.042081	1324587
1.5 - 2	0.09898	1,826	-0.17976	1288374
2 - 2.5	-0.20318	1,655	-0.05029	1340375
2.5 - 3	0.065008	1,415	-0.2715	1326898
3 - 3.5	-0.44259	1,467	-0.4495	1262989
3.5 - 4	-0.28807	1,193	-0.42917	1257839

표 17의 nTx 평균값만 보면, 전달 시간이 오래 걸릴수록 nTx의 개수가 줄어들고 있기 때문에 앞서 상관 관계를 분석하였던 결과와 마찬가지로 전달 시간과 nTx는 반비례 관계를 가지고 있다고 말할 수 있다. 하지만 표 17의 전달 시간과 nTx 간 상관 계수를 보면, 각 전달 시간 구간마다 반비례 관계와 비례관계가 불규칙적으로 나타나고 있음은 물론 그 값 역시도 모두 0.2 미만으로 두 값 간에 어떠한 관계가 있다고 말할 수 없다. size 역시도 마찬가지이다. 즉, 사이즈와 트랜잭션 수는 전달 시간에 영향을 미치지 않는다.

4.2.4. 로그의 흐름

다음은 MainNode와 직접 연결된 SubNode의 compact block protocol 로그 흐름을 분석한 결과이다. 전달 시간과 트랜잭션을 요청하는 것이 어떤 관계가 있는지 로그의 흐름을 분석하면서 알아본다.

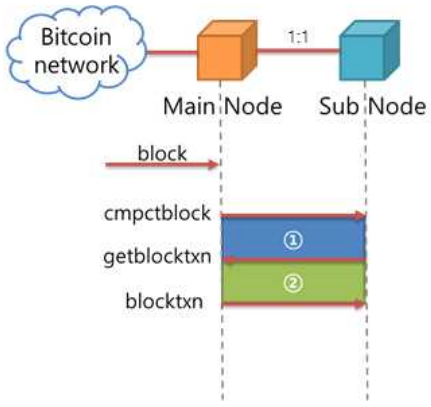


그림 12. Compact Block 전달 과정

우선 컴팩트 블록의 전달과정을 이해하기 위해 그림 12를 보면, 두 노드는 cmpctblock, getblocktxn, blocktxn 총 3개의 패킷 메시지를 주고받으면서 컴팩트 블록을 전달받아 완성 시킨다. 그림 12를 바탕으로 로그 파일에서 나타나는 최대, 최소로 전달 시간이 걸린 블록들의 로그 흐름을 분석한다.

분석 결과, 전달 시간이 최대로 걸린 블록들의 로그 흐름은 그림 12와 같이 정석대로 패킷을 주고받으나, 최소로 걸린 블록들은 getblocktxn을 보낸 로그가 없고, blocktxn만 있다. 즉 그림 12의 초록색으로 칠해진 2번 과정이 생략되어 있다. 또한, 로그 앞의 타임 스탬프를 확인해보면 getblocktxn 패킷과 blocktxn 패킷 사이에서 시간이 소요됨을 알 수 있다. 이 시간을 트랜잭션 요청 시간이라 정의하고, 전달 시간에서 트랜잭션 요청 시간이 차지하는 비율을 계산한 표가 표 18이다. 전달 시간이 최대로 걸린 655909, 655908 블록에 대해 조사한 것이다.

표 18. 전달 시간이 최대로 걸린 블록의 트랜잭션 요청 시간과 비율

블록 높이	전달 시간	트랜잭션 요청 시간	트랜잭션 요청 시간이 차지하는 비율
655909	5.987413 초	5.924182 초	98 %
655908	5.829158 초	5.816636 초	99 %

표 18을 통해 트랜잭션 요청 시간이 차지하는 비율이 98~99%로 대부분임을 알 수 있다. 앞의 실험들과 같이 추가로 최대, 최소 전달 시간이 걸린 블록을 5개 더 조사해본 결과 전달 시간이 최대로 걸린 블록들과 달리 최소로 걸린 블록들은 모두 getblockTxn 로그가 없다.

getblocktxn 로그가 없는게 정말 getblocktxn 패킷을 보내지 않았다는 것을 의미하는지는 로그의 흐름을 통해서는 알 수 없으나 최소로 걸린 블록들의 로그에서 blocktxn의 패킷이 33bytes로 찍혀 있는 것을 통해 알 수 있다. 이는 트랜잭션을 받지 않았다는 것을 의미하기 때문에 요청한 트랜잭션이 없고, 받지도 않았음은 확실하다. blocktxn의 패킷에 대한 것은 다음 4.2.6에서 자세히 설명한다.

즉, 로그로만 판단했을 때에는 과정이 하나 생략되면서 시간이 단축되었음을 의미한다. 덧붙여 전체 전달 시간을 압축 블록을 받는 구간과 트랜잭션을 요청하는 구간으로 나누었을 때, 트랜잭션을 요청하는 구간의 시간 평균이 압축 블록을 받는 구간보다 약 100배 정도 차이가 있다는 것이 이전 연구의 결론이다 [13].

이는 getblocktxn의 전송 유무, 즉 트랜잭션 요청 여부가 전달 시간에 영향을 미친다는 의미이다. 트랜잭션을 요청하는 것이 전달 시간에 영향을

미친다면, 트랜잭션을 요청한 개수와 사이즈는 어떨까. 이에 대해서는 4.2.5와 4.2.6에서 차례대로 밝힌다.

4.2.5. getblockTxn으로 요청한 트랜잭션 개수

트랜잭션을 요청한 개수는 압축 블록을 성공적으로 조립했을 때 찍히는 로그에서 수집한다. 해당 로그의 마지막 부분에 블록 조립 시 필요하여 요청한 트랜잭션 개수가 적혀있다. 요청한 트랜잭션 개수와 전달 시간의 관계를 알아보기 위해 분포도를 그리고 상관 계수를 알아낸다. 그림 13은 두 값의 분포도를 그린 그림이다.

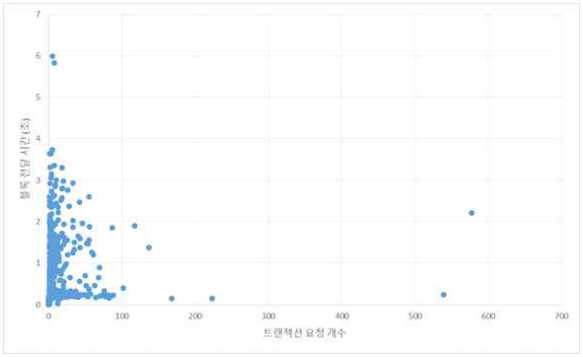


그림 13. 요청한 트랜잭션 개수와 전달 시간 분포 그래프

그림 13의 그래프를 보면, 상승하는 듯하나 정비례 관계가 뚜렷하게 보이진 않는다. 두 값의 상관 계수는 0.191522246 이다. 비례관계이지만 약 0.2 정도의 작은 숫자가 나와 거의 연관이 없음을 의미한다. 3.2.3에서 블록의 nTx와 size를 분석하였던 것처럼 전달 시간이 0.2초 이하로 걸린 블록들은 제외하고 상관 계수를 계산하였을 때에는 0.06510961로, 전체에 대해 상관 계수를 계산했을 때보다 더 작은 값을 보인다. 따라서 요청한 트랜잭션 개수와 전달 시간은 관계가 없다.

getblockTxn으로 요청한 트랜잭션 개수는 연관이 없다고 밝혔으나 3.2.4에서 말했듯이 getblockTxn 으로 트랜잭션을 요청한 여부는 전달 시간에 영향을 미친다. 이를 정확히 하기 위해 그린 그래프가 그림 14이다. 그림 14는 트랜잭션 요청 개수를 정렬 기준으로 삼고, 전달 시간과 함께 내림차순으로 정렬 한 것이다. 가로축은 본 연구에서 측정한 2,000개의 블록 개수를 단순히 적어 놓은 것이다.

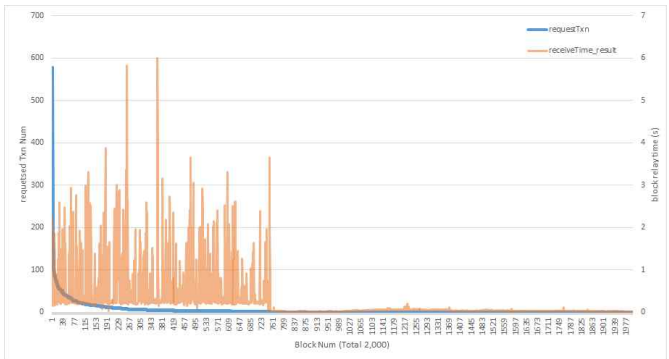


그림 14. 트랜잭션 요청 개수와 전달 시간 정렬 그래프

총 2,000개의 블록 중에 getblocktxn으로 트랜잭션을 요청한 블록은 754개로 전체의 37.7%를 차지한다. 그림 14에서 가로축의 754번째 부근을 보면 receivedTime_result 그래프가 급격히 하락하며 x 축과 매우 인접하게 그려지는 것을 확인할 수 있다. 이는 요청한 트랜잭션이 없을 때, 즉 트랜잭션을 요청하지 않을 때에 전달 시간이 급감함을 의미한다.

트랜잭션 요청 개수가 0인 것(트랜잭션 요청을 하지 않음)과 트랜잭션 요청 개수가 0이 아닌 것(트랜잭션 요청을 함)으로 나누어서 정확히 분석하고 통계로 낸다. 표 19과 표 20이 그 결과다.

표 19. 트랜잭션을 요청한 블록 통계 (단위 : 초)

평균	최소	최대	중앙값	표준편차
0.655025	0.01637	5.987413	0.251284	0.766578

표 20. 트랜잭션을 요청하지 않은 블록 통계 (단위 : 초)

평균	최소	최대	중앙값	표준편차
0.032987	0.000275	0.192786	0.027641	0.018523

표 19와 20에서 주요하게 살펴볼 것은 전달 시간의 최댓값과 평균이다. 트랜잭션을 요청했을 때는 최댓값이 약 6초, 요청하지 않을 때는 최댓값이 약 0.2초 정도로 30배 차이가 난다. 트랜잭션을 요청했을 때의 평균은 약 0.6초, 요청하지 않았을 때의 평균은 약 0.03초로 12배의 차이가 난다.

표준편차 역시도 의미가 있는데, 트랜잭션을 요청했을 때보다 트랜잭션을 요청하지 않았을 때의 표준편차가 더 작다. 트랜잭션을 요청했을 때는 각 값이 평균에 비해서 고르게 퍼져 있으나, 트랜잭션을 요청하지 않았을 때는 각 값이 평균에 근접하게 뭉쳐있다는 뜻이다.

종합하자면, 트랜잭션을 요청하지 않으면 요청했을 때 비해 12배나 적은 시간인 0.03초대로 대부분이 전달 시간을 소요한다.

3.2.5의 실험 결과 트랜잭션의 개수는 전달 시간에 영향을 미치지 않지만, 트랜잭션의 요청 여부는 전달 시간과 밀접한 관계를 갖음을 안다.

4.2.6. 요청한 트랜잭션의 사이즈

압축 블록 조립 시에 요청 여부와 달리 필요한 트랜잭션의 개수 자체는 전달 시간에 영향을 미치지 않음을 안다. 하지만, 사이즈는 영향을 미칠 수 있다. 개수와 사이즈는 비슷하면서도 다른 요소로 개수가 적어도 사이즈는 클 수 있기 때문이다. 따라서, 요청한 트랜잭션의 사이즈 데이터를 얻고 전달 시간과의 상관관계에 대해서 분석한다.

요청한 트랜잭션의 사이즈는 blocktxn 로그에서 얻는다. getblocktxn으로 블록을 조립할 때 필요한 트랜잭션을 요청한 이후, SubNode가 MainNode에게 요청한 트랜잭션을 받았을 때 blocktxn 패킷의 사이즈가 로그에 기록된다. 기론된 blocktxn 패킷 사이즈는 요청한 트랜잭션의 사이즈가 아니고, 기본 패킷 사이즈를 포함하고 있다. 패킷 구성을 나타내는 표 21을 참고해서 자세히 설명한다.

표 21. blocktxn packet 구성

Field Name	Type	Size	Encoding	Purpose
blockhash	Binary blob	32 bytes	The output from a double-SHA256 of the block header, as used elsewhere	제공되는 트랜잭션이 담긴 블록의 해시
transactions_length	Compact Size	1 or 3 bytes	As used to encode array lengths elsewhere	제공되는 트랜잭션 수
transactions	List of Transactions	variable	As encoded in "tx" messages in response to getdata MSG_TX	트랜잭션 리스트

표 21의 blockhash는 제공되는 트랜잭션이 담긴 블록의 해시이며, transactions_length는 제공되는 트랜잭션 수이다. 이 두 필드는 blocktxn의 기본적인 필드로 항상 존재한다. transactions는 트랜잭션 리스트이다[14]. transactions는 부가적인 필드로 getblocktxn으로 요청한 트랜잭션의 개수에 따라 다르다. 여기서 blockhash 와 transaction_length만 합하면 33bytes가 된다. 따라서 요청한 트랜잭션의 사이즈는 blocktxn 패킷 사이즈에서 33bytes를 제외하고 계산해야 한다. 4.2.4 로그의 흐름에서 전달 시간이 최소로 걸린 블록들의 blocktxn 패킷이 33bytes인 것이 트랜잭션을 받지 않았다는 것을 의미한다는 것은 표 21을 통해 설명할 수 있다. 요청한 트랜잭션의 사이즈와 전달 시간의 관계를 알기 위해 그려낸 두 값의 분포도가 그림 15이다.

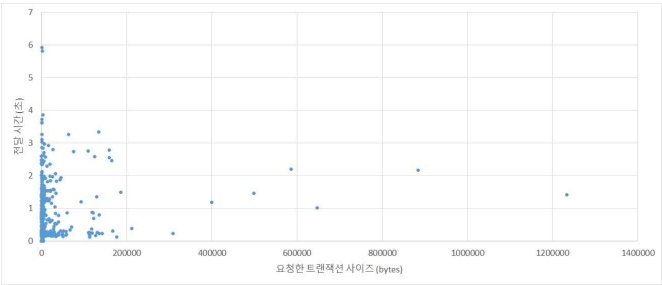


그림 15. 요청한 트랜잭션 사이즈와 전달 시간 분포도

그림 15 분포도에서 오른쪽 위로 상승하는 그래프를 기대했으나, 결과로 보면 뚜렷한 비례 관계가 보이지 않고 4.2.5의 요청한 트랜잭션 개수에서 그렸던 분포도 그림 13과 유사하게 그려진다.

두 값에 대하여 상관 계수는 0.244951 값이 나온다. 양의 비례관계를 가지긴 하지만 그 정도가 미미하다. 이전의 분석과 마찬가지로 전달 시간이 0.2초 이하인 블록을 제외하고 상관 계수를 계산하여도 0.136077로 전체를 계산했을 때보다 더 작은 숫자를 보이며 여전히 그 정도가 미미하다. 즉, 요청한 트랜잭션 사이즈와 전달 시간은 관계가 없다.

Ⅴ. 지연 원인 분석 결과

지연 원인을 찾기 위해 비교 분석으로 사용한 6가지 요소를 표로 정리하고, 정말 전달 시간 지연의 원인이 맞는지 분석한 결과 표가 다음 표 22이다.

표 22. 전달 시간 지연 원인 분석에 사용된 요소와 여부

요소	전달 지연 원인 여부
네트워크	○
블록의 체인 연결과 시스템 메모리양	X
블록의 사이즈와 트랜잭션 보유 수	X
로그의 흐름	○
getblockTxn으로 요청한 트랜잭션 개수	△
요청한 트랜잭션의 사이즈	X

요소에 대해 차례대로 설명하면, 네트워크의 경우에는 Internet Rush Hour(오전 9시 ~ 오후 12시) 때에 트래픽량이 증가하면서 블록 전달 시간을 지연시킨다. 블록의 체인 연결과 비트코인 노드가 사용하는 시스템 메모리양의 경우 블록이 노드의 체인에 포함되면서 찍히는 로그를 분석한 것이다. 그 과정에서 블록 검증에 사용한 UTXO가 메모리에 할당된 값을 보여주는 cache의 항목에서 전달 시간이 최소, 최대로 걸린 블록 간에 차이가 보여 추가로 분석하였으나 결론적으로 블록의 체인 연결과 비트코인 노드가 사용하는 시스템 메모리양은 전달 시간 지연에 영향을 미치지 않는다. 블록의 사이즈와 트랜잭션 보유 수는, 데이터를 전달할 때 보통 시간이 오래 걸리는 요인이 사이즈이기 때문에 블록의 정보 중에서 두 값을 중점적으로 분석하였으나 이 역시 전달 시간과 관계가 없다. 로그의 흐름은 SubNode의 로그에서 압축 블록 전달 과정에 대한 로그의 흐름을 분석한 것이다. 해당 분석에서는 전달 시간이 최대, 최소로 걸린 블록의 차이가 getblockTxn 로그의 여부다. 전달 시간이 최대로 걸릴 때는 있지만, 최소로 걸릴 때는 없다. 또한, 트랜잭션을 요청한 getblocktxn 패킷과 요청받은 blocktxn 패킷 사이의 시간이 블록의 전달 시간에서 99% 정도를 차지하여, 트랜잭션을 요청하는 것이 지연의 원인임을 밝힌다. getblockTxn으로 요청한 트랜잭션 개수를 통해서는 요청한 개수 자체는 영향을 주지 않지만 요청한 개수가 0인 것과 0이 아닌 것으로 나누어 분석하였을 때, 트랜잭션을 요청하지 않으면 요청했을 때 비해 12배나 적은 시간인 0.03초대로 대부분이 전달 시간을 소요함을 안다. 따라서 로그의 흐름에서처럼 트랜잭션을 요청 여부에 따라 전달 시간 지연 원인이 됨을 확실히 안다. 이에 표 22에서 세모로 표시하여 간접적으로 지연 원인을 밝혔다는 것을 표현한다. 요청한 트랜잭션 개수와 사이즈는 비례하지 않을 수 있기에 사이즈에 대해서도 분석해보았으나 사이즈 역시 그 자체로는 전달 시간에 영향을 미치지 않는다.

Ⅵ. 결론 및 향후 연구

본 논문에서는 하나의 PC에 비트코인 노드 두 개를 설치하고 두 노드를 직접 연결한 후 비트코인 노드 간 블록 전파 시간을 측정한다. 시간과 시스템이 동일하게 구성되어 있으며 네트워크 구성도 로컬로 동작하기 때문에 다른 문제를 검토할 필요 없이 측정할 수 있다. 2020년 10월 23일부터 11월 8일까지로 약 17일 정도 측정했으며 블록 높이 654000부터 656000까지 총 2,000개의 블록 데이터에 대해 분석한다.

고 대역폭 압축 블록 전달 방식으로 블록을 전달받는 노드에서, 블록 전달 시간은 평균 0.0266874초로 측정되었으며 최대 최솟값의 차이가 약 20,000배 나면서 전달 시간이 균일하지 않고 지연이 존재함을 확인한다.

지연의 원인을 분석하기 위해서 최대, 최솟값의 상위 2개 값을 선정하여 해당 블록 간의 차이가 어떤 것이 있는지 찾는다. 선정된 블록은 각 655909, 655908 과 654574, 655332이다. 지연 원인을 찾기 위해 비교 분석으로 사용한 요소는 총 6가지로 네트워크, 블록의 체인 연결과 비트코인 노드가 사용하는 시스템 메모리양, 블록의 사이즈와 트랜잭션 보유 수, 로그의 흐름, getblockTxn으로 요청한 트랜잭션 개수, 요청한 트랜잭션의 사이즈이다.

비트코인 노드의 압축 블록 전달 시간 측정 및 지연 원인 분석의 결과를 종합하면 다음과 같다. 블록 전달 시간은 네트워크의 상태와 압축 블록을 조립할 때 트랜잭션을 요청하는지 여부에 따라 영향을 받는다. 이는 블록의 필요 트랜잭션을 본인의 메모리 풀에 보유하고 있는지에 따라 지연될 수 있음을 뜻한다. 따라서 전달 시간 지연의 원인을 해결하기 위해서는 메모리 풀과 전달 시간 간의 추가 연구가 필요하다.

메모리 풀과 전달 시간 사이의 추가 연구를 통해서, 메모리 풀에 전달받는 트랜잭션의 특징을 파악하고 왜 본인의 메모리 풀에 필요한 트랜잭션이 없는지에 대한 이유를 밝혀야 한다. 또한 필요한 트랜잭션의 특징이 무엇인지, 어떤 트랜잭션인지 파악하여 getblockTxn으로 요청할 트랜잭션을 예측하고 먼저 메모리 풀에 받게 함으로써 블록이 트랜잭션을 요청하지 않도록 할 수 있다. 트랜잭션을 요청하는 블록이 줄어들면 전체적으로 전달 시간도 단축되며 비트코인 네트워크의 성능이 향상될 것이다.

또한, 본 논문에서는 고대역폭 전달 프로토콜에 대해서만 다루고 있으나

실제로 비트코인 노드는 소개하였던 Legacy, Low Bandwidth 프로토콜까지 총 3개의 프로토콜을 모두 사용하고 있기 때문에, 실제로 블록마다 어떤 프로토콜을 선택해야 전달 시간이 단축 되는지에 대해서도 연구가 필요하다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2018R1D1A1B07050380).

참 고 문 헌

[1] Satoshi Nakamoto. (2008). Bitcoin: A peer-to-peer electronic cash system

[2] F. M. Benčić and I. Podnar Žarko. (2018). Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, pp. 1569-1570, doi: 10.1109/ICDCS.2018.00171.

[3] Matt Corallo, BIP 152: Compact block relay, https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki (accessed: March 2021)

[4] BitcoinCore, Compact Blocks FAQ, https://bitcoincore.org/en/2016/06/07/compact-blocks-faq (accessed: March 2021)

[5] C. Decker and R. Wattenhofer. (2013). Information propagation in the Bitcoin network. IEEE P2P 2013 Proceedings, Trento, pp. 1-10, doi: 10.1109/P2P.2013.6688704.

[6] Nagayama, Ryunosuke, K. Shudo and Ryohei Banno. (2019). Simulation of the Bitcoin Network Considering Compact Block Relay and Internet Improvements. ArXiv abs/1912.05208 : n. pag.

[7] GitHub, bitcoin, https://github.com/bitcoin/bitcoin, (accessed : March 2021)

[8] docker, docker, https://www.docker.com, (accessed : March 2021)

[9] MathWorks, MATLAB, https://kr.mathworks.com/products/matlab.html (accessed : March 2021)

[10] Anderson, Chris (2006). The Long Tail: Why the Future of Business is Selling Less of More. Hyperion. ISBN 1-4013-0237-8

[11] Hallam, Google Analytics: hour of day & day of week reports, https://www.hallaminternet.com/google-analytics-hour-of-day-day-of-week-reports (accessed : March 2021)

[12] Delgado-Segura S., Pérez-Solà C., Navarro-Arribas G., Herrera-Joancomartí J. (2019) Analysis of the Bitcoin UTXO Set. In: Zohar A. et al. (eds) Financial Cryptography and Data Security. FC 2018. Lecture Notes in Computer Science, vol 10958. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-58820-8_6

[13] 김애리, 맹수훈, 에싸이드 매리얌, 주홍택. (2020). 로그 출력에 기반한 직접 연결된 비트코인 노드의 압축블록 전달 시간 측정 및 분석. 한국정보과학회 학술발표논문집, (), 1539-1541.

[14] Bitcoin Developer, P2P Network - BlockTxn, https://bitcoin.org/en/developer-reference (accessed: March 2021)

Gossip 기반 P2P 라우팅을 통한 블록체인 성능 개선 연구

최원석, 강창훈, 홍원기
포항공과대학교 컴퓨터공학과

{ws4583, chkang, jwkhong}@postech.ac.kr

A Study on Improvement of Blockchain Performance through Gossip-based P2P Routing

Wonseok Choi, Changhoon Kang, James Won-Ki Hong
Department of Computer Science and Engineering, POSTECH

요 약

Peer-to-Peer(P2P) 네트워크는 기존의 클라이언트-서버 모델과 달리 모든 노드들이 자원을 공유하며 서비스 요청자와 서비스 제공자의 역할을 수행한다. 다양한 P2P 서비스들의 등장과 더불어 현재 혁신적인 기술로 큰 주목을 받고 있는 블록체인 기술 역시 P2P 네트워크를 기반으로 하고 있다. 현재 블록체인의 주요 관심사 중 하나는 실용화를 위한 성능 개선에 있다. 이러한 성능개선에 관한 연구는 대부분 합의 알고리즘 단계에서 이루어지고 있으며 여러 블록체인들이 독자적인 합의 알고리즘을 적용하여 성능을 개선하고 있다. 하지만 블록체인에서의 P2P 라우팅의 경우 메시지 전파를 위해 여러 P2P 서비스에서 사용하고 있는 Gossip 기반의 라우팅을 적용하고 있다. 합의 알고리즘뿐만 아니라 P2P 라우팅 방식 역시 블록체인에서 노드 간의 메시지 전달에 주요한 역할을 하기 때문에 P2P 라우팅의 개선 또한 블록체인의 성능 개선으로 이어질 수 있다. 본 논문에서는 기존의 Gossip 기반 P2P 라우팅 방식들에 대해 분석하여 블록체인에 적합한 형태의 Gossip 기반 P2P 라우팅을 어떻게 구성해야 할지에 대해 제안함으로써 블록체인의 성능 개선에 기여하고자 한다.

Ⅰ. 서 론

Peer-to-Peer(P2P)[1] 네트워크는 별도의 중앙 서버가 존재하지 않는 분산된 시스템으로 모든 노드들이 동등한 권한을 가진다. 기존의 클라이언트-서버 모델과 달리 P2P 네트워크에서는 노드들이 서비스 요청자와 서비스 제공자의 역할을 동시에 수행하기 때문에 모든 노드들이 자원을 공유 해야 하는 시스템에서 주로 사용된다. 사토시 나카모토라는 익명의 개발자로부터 처음 제안된 비트코인[2]에서부터 비롯되어 현재 많은 연구가 진행되고 있는 분산 컴퓨팅 기술인 블록체인 기술 역시 P2P 네트워크를 기반으로 하고 있다.

블록체인은 탈중앙성이나 익명성 등의 장점으로 많은 주목을 받고 있으나 실용화를 위해서는 여전히 성능적인 부분에서 문제가 존재한다. 대표적인 블록체인인 비트코인에서는 대략 10 정도의 초당 거래량(TPS), 이더리움에서는 15~20 정도의 TPS 를 보이고 있다. 대부분의 블록체인들은 이러한 TPS 문제를 개선하기 위해 기존의 합의 알고리즘을 개선하는 것에서 방법을 있다. 대표적으로 Cosmos[3] 에서는 DPoS(Delegated Proof of Stake)[3]와 PBFT(Practical Byzantine Fault Tolerance)[4]를 결합한 텐더민트 합의 알고리즘[3]을 사용하고 있으며 Algorand[5] 역시 DPoS 를 이용해 TPS 를 개선하였다.

하지만 P2P 라우팅 단계에서는 별다른 개선 없이 대부분 기존 Gossip 기반의 라우팅을 사용하고 있다. P2P 네트워크에서는 하나의 노드로부터 빠르게, 많은 노드들에게 메시지를 전송하는 것이 중요하다. 이를 위해 효율적인 멀티캐스팅[6]이 요구되며 현재 P2P 네트워크에서 대표적으로 사용되는 것이 Gossip 기반의 멀티캐스팅[7] 방식이다. Gossip 프로토콜에서 노드들은 주기적으로 이웃 노드들에게 메시지를 전송하며 메시지를 받은 노드들 역시 동일하게 이를 이웃 노드들에게 전송한다. 모든 이웃 노드들에게 메시지를 보내는 대신 메시지를 보낼 이웃 노드를 선정하는 알고리즘을 달리 하여 성능 개선이 가능하고 메시지 전송 방식에도 Eager Push, Lazy Push, Pull 등의 접근 방식[8]이 존재한다.

Gossip 프로토콜에도 여러 접근 방식이 존재하고 이를 통한 성능 개선이 가능하다. 본 논문에서는 Gossip 기반 P2P 라우팅을 분석하고 블록체인에 사용하기 적합한 형태의 Gossip 프로토콜을 찾기 위한 고려 사항들을 분석함으로써 블록체인 성능 개선에 기여하고자 한다.

Ⅱ. 관련 연구

GossipPINE[9] 에서는 Gossip 프로토콜에서 확률적으로 메시지를 보낼 이웃 노드를 선정하는 Probabilistic Neighbor-Aware Gossip 알고리즘인 Probabilistic Inverse Neighbor-degree Edge 알고리즘을 제안하였다. GossipPINE 은 이웃 노드들의 차수(Degree)를 기반으로 하며 이웃 노드의 차수가 낮을수록 해당 노드에게 메시지를 전송할 확률이 높아지게 된다. 이 접근 방식을 통해 차수가 지나치게 낮은 노드의 경우 장애가 발생했을 것이라 예측할 수 있으며 차수가 높은 노드에게는 중복된 메시지를 보낼 확률이 낮아지게 된다. GossipPINE 은 기존의 Probabilistic Neighbor-Aware Gossip 알고리즘들보다 낮은 Latency 와 높은 Reliability 를 보였다.

Plumtree[8] 에서는 Push-Lazy-Push Multicast Tree 프로토콜을 제안하여 멀티캐스트를 개선하였다. Plumtree 는 랜덤하게 선정된 노드에서 받은 메시지를 바로 전송하는 Eager Push 방식과 처음에는 메시지의 식별자를 전송한 후 이를 받은 노드가 해당 메시지의 페이로드를 요청하는 Lazy Push 방식을 혼합해서 사용하였다. 처음 메시지를 전송한 노드들과는 Eager Push 방식으로 통신하고 중복된 메시지를 전송한 노드들과는 Lazy Push 방식으로 통신함으로써 전체 노드들의 신장 트리를 구성하였다. Plumtree 또한 기존의 Gossip 프로토콜보다 overhead 가 적고 높은 reliability 를 보였다.

Ⅲ. 블록체인에서의 Gossip 기반 라우팅

1. 기존 블록체인에서의 Gossip 기반 라우팅

비트코인에서는 Lazy Push 방식의 Gossip 기반 브로드캐스트를 사용한다. 메시지 발신자는 연결된 노드들에게 Inventory 메시지를 전송 하고 메시지를 수신한 노드들은 GetData 메시지로 응답한다. 그리고 메시지 발신자는 GetData 메시지로 응답한 노드들에 데이터를 전송함으로써 메시지 전달 과정이 마무리 된다. 비트코인은 여기서 Lazy Push 방식을 좀 더 개선하기 위해 Compact Block Relay 를 도입하였다. 노드들 간에 블록 정보를 교환할 때 노드들은 새 블록이 발견되면 블록 헤더와 트랜잭션 해시 정보들만을 전송하며 이를 전달받은 노드들은 트랜잭션 해시 정보들만으로 블록을 구성할 수 없을 경우에만 추가적으로 트랜잭션 정보를 요청한다. Compact Block Relay 방식은 전체 bandwidth 와 블록 전파 시간을 크게 단축할 수 있지만 최악의 경우 메시지 교환 과정이 늘어나 latency 면에서 더 나빠질 수 있다는 문제점이 존재한다.

비트코인과 같이 가장 유명한 블록체인 중 하나인 이더리움에서는 노드들이 새로운 블록 정보를 받을 경우 이웃 노드들 중 랜덤하게 일부 노드들을 선정한다. 그리고 선정된 노드들에게는 블록 헤더만을 검증한 이후 모든 블록 데이터를 전송하고 나머지 노드들에게는 전체 블록 정보를 검증한 뒤 블록 해시 값만을 전송한다. 블록 해시 값을 전송받은 노드는 일정시간을 대기한 후 블록 정보를 아는 주변 노드들에게 블록 헤더를 요청한다. 블록 헤더를 전달 받으면 다시 일정 시간을 대기한 뒤 블록 바디 정보를 요청하여 전달받는다. 이러한 방식은 대기 시간으로 인해 노드들이 전체 블록 정보를 얻기 까지 오랜 시간이 걸린다는 문제점이 존재한다.

2. Gossip 기반 라우팅 성능 개선 방안

P2P 라우팅의 성능은 크게 세 가지 관점에서 바라볼 수 있다. 첫 번째는 latency 이다. 당연하게도 노드와 노드간의 메시지 전달 속도는 메시지를 모든 노드에게 전달하는 데 걸리는 시간에 큰 영향을 미친다. 두 번째는 reliability 이다. 노드의 수가 적은 환경이라면 모든 노드를 서로 연결함으로써 안전하게 모든 노드들에게 동일한 메시지를 전송하는 것이 가능하다. 하지만 퍼블릭 블록체인과 같이 노드의 수가 많은 환경이라면 모든 노드를 연결하는 것은 불가능하고 하나의 노드에 연결되는 노드의 수가 제한된다. 하나의 노드에서 메시지를 보낼 때 해당 노드는 자신과 연결된 모든 노드, 혹은 일부 노드에게 메시지를 보내게 되는데 이때 네트워크 연결이 불안정 할 경우 메시지를 전달 받지 못하는 노드가 발생할 수 있다. 네트워크의 reliability 를 높이기 위해서는 노드와 연결되는 노드의 수를 적절히 선정하는 것이 필요하다. 마지막으로 redundancy 이다. 노드와 연결되는 노드의 수가 많아질 경우 메시지를 전송하는 과정에서 동일한 메시지를 중복해서 수신할 가능성이 높아진다. 이러한 불필요한 메시지는 네트워크의 트래픽을 높여 결과적으로 전체 네트워크의 메시지 전달 속도를 낮추게 된다. 따라서 불필요하게 중복되는 메시지를 줄여 redundancy 를 낮춤으로써 Gossip 프로토콜의 성능을 개선할 수 있다.

Latency, reliability, redundancy 와 같은 특성들은 네트워크 토폴로지에 따라서도 다르게 나타날 수 있다. 따라서 주어진 토폴로지 환경에 적합한 라우팅 방식을 선정하는 것 역시 중요하다. 또한, reliability 를 높이기 위해 노드와 연결되는 노드 수를 늘릴 경우 redundancy 가 낮아지는 문제가 존재한다. 따라서 단순히 노드와 연결되는 노드의 수를 조정하는 것이 아닌, 연결된 노드 중 메시지를 전달할 노드를 적절하게 선정할 필요가 있다. 이러한 방법으로 메시지를 전송하는 노드의 수를 제한하거나 랜덤하게 선정된 노드들에게만 메시지를 전송하는 등의 방안들이 존재한다.

Gossip 프로토콜을 구성하기 위해서는 먼저 네트워크에서 메시지를 전송하는 주체가 되는 메시지 발신자에 대한 정의가 필요하다. 일반적인 퍼블릭 블록체인에서는 모든 노드들이 트랜잭션을 생성하거나 블록을 생성할 수 있기 때문에 모든 노드들이 최초 메시지 발신자가 될 수 있다.

그리고 최초 메시지 발신자로부터 메시지를 수신한 이웃 노드들 역시 이후에 메시지 발신자가 되어 해당 메시지를 주변 노드에게 메시지를 전송하게 된다. 이때, 노드들이 메시지를 전송할 때 메시지에 자신의 서명을 추가함으로써 메시지 수신자로 하여금 동일한 노드로부터 받은 중복된 메시지를 확인할 수 있게 하는 것이 가능하다. 메시지 전송 방식은 기본적으로 모든 정보를 보내는 Eager Push 나 간략한 정보만을 보낸 뒤 응답을 받은 뒤에 모든 정보를 보내는 Lazy Push 방식으로 정할 수 있다.

신장 트리(spanning tree) 기반의 라우팅 역시 하나의 가능성이 될 수 있다. 신장 트리 기반의 라우팅은 P2P 네트워크 상의 모든 노드를 포함하는 최소 신장 트리를 구성하여 메시지를 전파함으로써 메시지 전송 과정에서 중복된 메시지의 전달을 없앨 수 있다. 따라서 신장 트리 기반의 라우팅은 redundancy 가 적고 reliability 가 높다는 장점이 있다. 하지만 신장 트리 구조로 노드를 구성할 경우 노드에 장애가 발생했을 때 메시지를 전달할 수 없게 되기 때문에 신장 트리를 빠르게 재구성할 수 있어야 한다. 그림 1 은 신장 트리 예시를 보여준다.

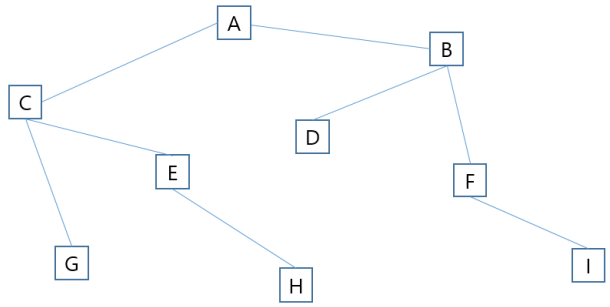


그림 1. 신장 트리 예시

신장 트리 기반의 라우팅을 개발하기 위해서는 우선 트리를 구성할 때 메시지 발신자 마다 개별적인 트리를 구성할 것인지, 혹은 모든 노드들이 공유하는 하나의 트리를 구성할 것인지에 대한 고려가 필요하다. 메시지 발신자 마다 개별적인 트리를 구성할 경우 블록체인에서는 모든 노드들이 메시지 발신자가 될 수 있기 때문에 노드 수만큼의 트리가 필요하다. 따라서 이를 유지하는 데 큰 비용이 발생할 수 있다. 모든 노드들이 공유하는 하나의 트리를 구성할 경우 메시지를 전송하는 경로가 기존 Gossip 기반의 라우팅보다 길어질 수 있다. 따라서 어떠한 방식을 적용할 것인지, 어떻게 해당 문제들을 극복할 것인지에 대해 고려하여야 한다.

신장 트리가 올바르게 작동하기 위해서는 트리를 생성할 때 노드와 연결할 노드들을 선정해야 하며 각 노드들이 최소 하나의 정상적인 다른 노드와 연결되어야 한다. 또한, 노드가 네트워크에 참여하거나 네트워크에서 나갈 때 신장 트리에 바로 반영이 되어야 한다. 노드가 네트워크에 참여하면 트리에 해당 노드가 추가되어야 하며 연결된 노드를 주기적으로 확인하여 해당 노드가 네트워크에서 나갈 경우 트리에서 제거되어야 한다. 트리 구조에서 하나의 노드에 장애가 발생 할 경우 해당 노드가 속해있는 가지에 문제가 발생한다. 따라서 연결된 노드가 일정 시간 동안 응답이 없다면 기존의 다른 노드와 연결하여 메시지를 전달 받아야 한다. 이 경우 어떤 노드와 새롭게 연결할 지를 고려해야하며 기존 연결을 끊고 새로운 노드와 연결되었을 때 신장 트리에 루프가 발생하지 않도록 해야한다.

Ⅳ. 결 론

본 논문에서는 블록체인의 성능을 높이기 위한 방안으로 Gossip 기반의 P2P 라우팅에서의 개선을 제안하였다. 그리고 Gossip 기반의 P2P 라우팅을 개선한 연구들을 소개하였다. GossipPINE 에서는 새로운 이웃 노드 선정 방식을 도입하여 성능을 개선하였으며 PlumTree 에서는 신장 트리를 기반으로 한 Gossip 프로토콜을 제안하였다. 대표적인 블록체인인 비트코인과 이더리움에서의 Gossip 프로토콜을 분석하였으며 이를 통해 블록체인에서 사용될 Gossip 프로토콜을 구성하기 위해 고려해야할 사항들을 분석하였다. Gossip 프로토콜을 개선하기 위해서는 크게 Latency, Reliability, Redundancy 세 가지 측면에서 접근할 수 있으며 이웃 노드 선정 알고리즘의 개선, 혹은 신장 트리와 같은 네트워크 토폴로지 도입을 통해 이들을 개선 가능하다.

향후 연구로는 동일한 환경에서 여러 이웃 노드 선정 알고리즘을 분석 및 비교하여 성능을 측정해보고 최적의 이웃 노드 선정 알고리즘을 찾아내고자 한다. 또한 신장 트리에서 학습을 기반으로 한 메타 휴리스틱 기술을 통해 최적의 신장 트리 라우팅 경로를 찾아내고자 한다.

ACKNOWLEDGMENT

본 연구는 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발, IITP-2021-2017-0-01633*, 대학 ICT 연구센터육성지원사업)

참 고 문 헌

[1] Lua, Eng Keong, et al. "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys & Tutorials* 7.2 (2005): 72-93.

[2] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." *URL: https://bitcoin.org/bitcoin.pdf (accessed: 01.04. 2010)* (2008).

[3] Cosmos, “Cosmos Whitepaper.” *URL: https://v1.cosmos.network/resources/whitepaper (accessed: 01.04. 2021)*.

[4] Castro, Miguel, and Barbara Liskov. "Practical byzantine fault tolerance." *OSDI*. Vol. 99. No. 1999. 1999.

[5] Gilad, Yossi, et al. "Algorand: Scaling byzantine agreements for cryptocurrencies." Proceedings of the 26th Symposium on Operating Systems Principles. 2017.

[6] Défago, Xavier, André Schiper, and Péter Urbán. "Total order broadcast and multicast algorithms: Taxonomy and survey." *ACM Computing Surveys (CSUR)* 36.4 (2004): 372-421.

[7] Birman, Kenneth P., et al. "Bimodal multicast." *ACM Transactions on Computer Systems (TOCS)* 17.2 (1999): 41-88.

[8] Leitao, Joao, Jose Pereira, and Luis Rodrigues. "Epidemic broadcast trees." *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*. IEEE, 2007.

[9] Hu, Ruijing, and Leander Jehl. "Reliable Probabilistic Gossip over Large-Scale Random Topologies." *arXiv preprint arXiv:1704.05808* (2017).

이더리움 노드 탐색 프로토콜 분석을 위한 와이어샤크 해석기 개발

김정연*, 주홍택

*계명대학교 컴퓨터공학과

*jungyeonkim@stu.kmu.ac.kr, juht@kmu.ac.kr

Development of Wireshark Dissector for Ethereum Node Discovery Protocol Analysis

Jungyeon Kim*, Hongteak Ju

*Dept. of Computer Engineering, Keimyung Univ.

요 약

이더리움은 P2P 네트워크에서 노드 정보를 검색하고 저장하기 위해 Kademlia 기반의 노드 탐색 프로토콜을 사용하고 있다. 와이어샤크에서 비트코인 프로토콜은 해석기를 이용하여 네트워크 패킷을 분석할 수 있지만 이더리움 프로토콜 해석기 개발은 미흡하다. 본 논문은 와이어샤크에서 이더리움 노드 탐색 프로토콜을 분석하기 위한 해석기를 개발한다. 이더리움 노드 탐색 과정 분석으로 네트워크 성능 향상과 취약성 연구를 위한 기초 자료로 활용될 수 있다.

Ⅰ. 서 론

이더리움(Ethereum)은 블록체인 기술을 기반으로 스마트 계약 기능을 지원하기 위한 분산 컴퓨팅 플랫폼이다. 이더리움은 탈중앙화된 분산 P2P 네트워크를 좁은 직경의 토폴로지로 생성하기 위해서 Kademlia 기반의 노드 탐색 프로토콜(Node Discovery Protocol)을 사용하고 있다[1]. 이더리움 네트워크 확장에 따라 네트워크 장애 및 프로토콜 문제점을 확인해야할 필요성이 높아졌으나 와이어샤크(Wireshark)에서 이더리움 패킷 해석을 지원하지 않는다.

본 논문은 와이어샤크에서 이더리움 노드 탐색 프로토콜을 분석하기 위해 이더리움 네트워크 패킷을 수집하고 노드 탐색 프로토콜 해석기를 개발한다. 이더리움 노드 탐색 과정을 수집·분석하여 네트워크 성능 향상 연구를 위한 기초 자료로 사용할 수 있다.

Ⅱ. 관련 연구

이더리움은 P2P(Peer-to-Peer) 네트워크에 의해 통신된다. 이더리움의 네트워크 통신은 다음의 3가지의 프로토콜로 구성되어 있다. 노드 탐색을 실시하는 Node Discovery, TCP 연결 및 암호화 통신을 위한 RLPx 그리고 이더리움 애플리케이션 레벨 프로토콜인 Subprotocol에 의해 이더리움 네트워크가 관리된다[2]. 이더리움 네트워크에 가입하는 새로운 노드의 일반적인 워크플로우는 그림 1과 같다.

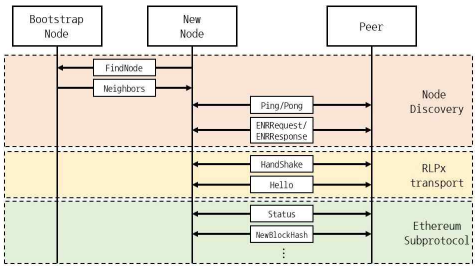


그림 1. 이더리움 신규 노드의 워크플로우

2.1 이더리움 노드 탐색 프로토콜

이더리움은 탈중앙화된 분산 P2P 네트워크를 좁은 직경의 토폴로지로 생성하기 위해서 Kademlia 기반의 노드 탐색 프로토콜을 사용한다. 노드 탐색 프로토콜은 이더리움 노드에 대한 정보를 검색하고 저장하기 위한 UDP(User Datagram Protocol) 기반 프로토콜이다. 신규 노드는 이더리움 네트워크에 참여하기 위해 라우팅 테이블에 하드 코딩된 부트스트랩(Bootstrap) 노드 ID를 추가한다. 모든 노드에는 secp256k1 공개키인 ID가 있다. 라우팅 테이블은 노드 자체의 노드 ID와 인접 노드 ID 사이의 XOR 거리에 따라 160개의 버켓(k-bucket)으로 분할된다[3]. 각 버켓은 XOR 거리가 $2i$에서 $2i+1\,(0 \le i < 256)$사이의 노드 목록을 유지한다. 버켓에는 최대 16개(k)의 노드 항목이 포함된다[2]. 새로운 피어를 찾기 위해 FindNode 메시지를 부트스트랩 노드로 보내 ID에 가장 가까운 노드에 대한 정보를 요청한다. 응답으로 새로운 노드 ID를 얻게 되면 버켓에 저장되고 모든 버켓을 채울 때까지 노드 탐색을 반복한다. 다음으로, 새로운 피어와 연결하기 위해서 노드의 온라인 상태를 확인하는 Ping ,Pong 메시지를 주고받는다. 마지막으로 TCP 연결을 위한 노드 레코드(Ethereum Node Record) 정보를 주고받으며 노드 탐색 프로토콜 과정이 완료된다. 이후에는 RLPx 프로토콜과 애플리케이션 레벨 프로토콜이 사용된다. 현재 이더리움 노드 탐색 프로토콜 버전 4에서 사용 중인 패킷은 표 1과 같다.

이름	필드	설명
Ping	0x01	수신 노드의 온라인 상태 확인 메시지
Pong	0x02	Ping에 대한 응답메시지
FindNode	0x03	특정 노드 ID에 가까운 노드에 대한 정보 요청 메시지
Neighbors	0x04	노드 ID에 가장 가까운 16개의 노드 정보 전달 메시지
ENRRequest	0x05	수신 노드의 노드 레코드 요청 메시지
ENRResponse	0x06	노드 레코드 전달 메시지

표 1. 노드 탐색 프로토콜 패킷 종류

2.2 와이어샤크 이더리움 프로토콜 해석기

와이어샤크는 네트워크 패킷 데이터를 분석하는 대표적인 프로그램이다. 각기 다른 네트워크 프로토콜이 규정한 패킷의 의미와 더불어 필드와 요약 정보를 보여줄 수 있다. 와이어샤크는 오픈 소스 소프트웨어 프로젝트이며 GNU GPL(General Public License)에 따라 출시되었다. 모든 사용자가 와이어샤크를 자유롭게 사용할 수 있을 뿐만 아니라 와이어샤크에 새로운 프로토콜을 추가할 수 있다. 와이어샤크에서 기본적으로 내장되어 있는 해석기가 없는 경우, 플러그인 방식 또는 소스 내장 방식으로 구현할 수 있다. 플러그인은 Lua로 작성된 스크립트이거나 C 또는 C++로 구현될 수 있다.

현재 대표적인 암호화폐인 비트코인은 와이어샤크에서 기본적으로 제공되는 해석기로 네트워크 프로토콜 분석이 가능하다. 버전 1.10.0에서 3.4.4까지 개발되어있다[4]. 2016년 8월 BIP-152를 기반으로 업그레이드된 압축 블록 전달 프로토콜(Compack Block Relay)을 제외한 비트코인 프로토콜은 해석 가능하다. 반면, 와이어샤크에서 이더리움 네트워크 프로토콜 분석을 위한 해석기는 제공되지 않는다. 이더리움에서는 ConsenSys의 프로토콜 엔지니어링 그룹 및 시스템 팀(PegaSys)에서 개발한 이더리움 노드 발견 프로토콜 버전 4의 오픈 소스 해석기를 와이어샤크 이더리움 프로토콜 해석기로 사용할 것을 추천하고 있다. PegaSys의 해석기는 C 기반으로, 이 해석기를 컴파일하고 플러그인을 만들려면 해석기 소스 이외에 CMakeLists.txt 파일을 포함한 몇 가지 지원 파일을 추가 및 수정하고 재빌드해야 한다.

Ⅲ. 이더리움 프로토콜 해석기 구현

3.1 개발 환경

이더리움 네트워크 패킷을 수집하기 위해 1개의 노드에 이더리움 코어를 설치하고 이더리움 네트워크에 연결하여 블록 동기화 및 노드 탐색 완료 후 진행한다. 이더리움 네트워크 패킷 수집 시간은 버켓을 채운 후, 정리(Refresh)하는 일정주기(Refresh Time, 1시간)를 기준으로 3시간 동안 진행한다[5]. 수집 방법은 와이어샤크를 활용하여 이더리움 프로토콜 패킷을 전송 계층 프로토콜(Transport Protocol)에서 TCP와 UDP로 분류하고 pcap 파일 형식으로 저장한다. 표 2는 이더리움 노드 탐색 프로토콜 해석기를 구현하는 동안 사용된 소프트웨어를 설명한다.

표 2. 이더리움 프로토콜 해석기 구현 구성 요소

구성 요소	사 양
Linux	Ver 18.04
Ethereum Client	Ver 1.9.12
Go-Language	Ver 1.13.6
Wireshark	Ver 3.2.7
Lua	Ver 5.3

본 논문에서는 프로그램 확장 편의성을 고려하여 와이어샤크의 플러그인 해석기를 Lua 스크립트로 구현하여 진행했다. 와이어샤크의 Lua 인터프리터는 와이어샤크의 글로벌 구성 디렉토리에 위치한 init.lua에서 지정된 파일을 로드하여 시작한다. Lua 스크립트로 구현된 해석기는 와이어샤크를 재빌드하거나 추가적인 수정이 없어 사용이 용이한 것이 장점이다. 개발자가 오픈 소스로 배포 시에 다른 사용자들이 바로 와이어샤크에서 실행할 수 있다.

3.2 이더리움 프로토콜 해독 과정

이더리움 노드 탐색 프로토콜 패킷은 UDP 전송 방식으로 주로 30303번 포트를 사용하여 통신되고 있다. 이더리움 프로토콜 해석기가 구현되지 않은 상태에서는 와이어샤크가 이더리움 패킷을 UDP 패킷으로 캡처·해석하고 저장한다. 응용 계층(Application layer) 데이터는 원시 상태인 Hex값으로 출력된다. 본 논문에서 구현한 ethereum.lua 파일은 기존 해석기 테이블 중 udp.port 항목이 30303번인 경우에 해석하도록 설정한다.

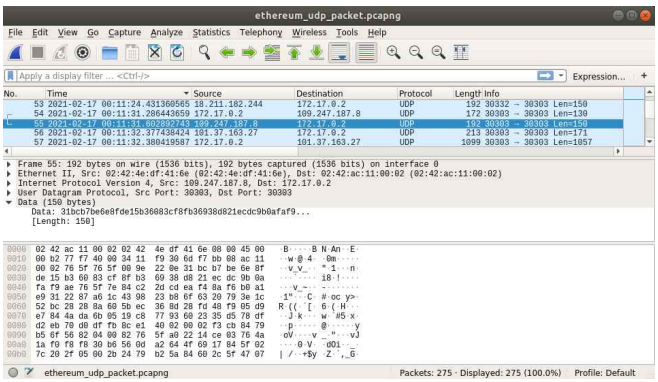


그림 2. 해석되지 않는 이더리움 노드 탐색 프로토콜 패킷

이더리움 UDP 패킷의 최대 크기는 1280Byte로 페이로드가 최대 크기를 초과할 경우 여러 개의 패킷으로 나누어 전송하고 있다. 노드 탐색 프로토콜 패킷의 경우에는 패킷 재조립 과정이 필요하지 않다.

이더리움의 UDP 패킷 헤더는 Hash, Signature, Packet Type으로 구성되어 있다. Packet Type 필드는 메시지 유형을 정의하는 단일 바이트다. Ping 메시지부터 ENRResponse 메시지까지 순서대로 0x01부터 0x06까지의 값으로 Packet Type 필드가 정의되어 있다. 패킷 유형에 따라 메시지 구조가 다르며 RLP(Recursive Length Prefix) 목록으로 인코딩된다[6]. Neighbors 메시지의 데이터 구조는 다음 표와 같다.

표 3. Neighbors 메시지 데이터 구조

패킷 필드 이름	크기(Byte)	데이터 종류
Nodes	Variable	[]rpcNode
Expiration	4byte	timestamp(UCT)

표 4. rpcNode 데이터 구조

패킷 필드 이름	크기(Byte)	데이터 종류
IP	4 or 16 byte	IPv4 or IPv6
UDP	Variable	UDP port
TCP	Variable	TCP port
ID	encPubkey	secp256k1 public key

Neighbors 메시지는 FindNode 메세지의 target ID에 가장 가까운 노드 16개의 정보를 rpcNode 데이터 구조로 응답한다. rpcNode 필드가 가변적이기 때문에 이더리움에서 사용하는 RLP에 따른 디코딩 과정으로 얻은 필드 길이에 맞게 해석해야 한다. Neighbors 메시지을 해석하는 수도코드는 다음과 같다.

```
void function ethereum_protocol.dissector(buffer, pinfo, tree){
    local maintree = tree:add(ethereum_protocol, buffer(), type[packet_type])
    local offset = 0
    if packet_type == 4 {
        Nodes_len = rlp_decode(offset++)
        while(offset < Nodes_len){
            local subtree = maintree:add(ethereum_protocol, buffer(), "Nodes")
            rpcNode_len = rlp_decode(offset++)
            while(offset < rpcNode_len){
                field_len = rlp_decode(offset++)
                subtree:add(ethereum.field, buffer(offset, field_len))
                offset += field_len
            }
        }
        expiration_len = rlp_decode(offset++)
        maintree:add(ethereum.expiration, buffer(offset, expiration_len))
        offset += expiration_len
    }
}

```

그림 3. Neighbors 메시지 해석 수도코드

3.3 이더리움 프로토콜 해석기 구현 결과

이더리움 네트워크 패킷을 수집할 때, 패킷 목록 창에서 패킷 커맨드 값이 나오고, 상세 창에서 프로토콜에 맞춰 해석된 패킷 데이터를 그림 4에서 확인할 수 있다. FindNode를 송신한 노드에게 노드의 피어 정보를 전달하기 위해 Neighbors 메시지로 정상적으로 응답하고 있다. 패킷 최대 크기를 초과하지 않도록 16개의 이웃 노드 정보를 2개의 패킷으로 나누어 전송한다.

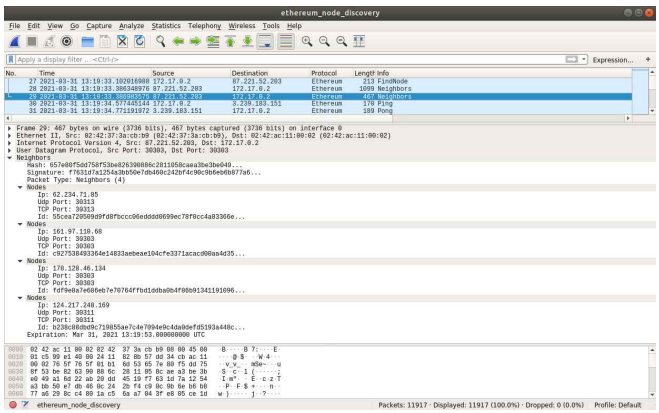


그림 4. 해석기를 적용한 Neighbors 메시지

해석기를 적용한 와이어샤크로 패킷 수집 기간 동안 총 11,895개의 이더리움 노드 탐색 프로토콜 패킷을 수집하였다. 총 패킷에서 11,875개의 패킷은 노드 탐색 프로토콜 패킷이고, 20개의 패킷은 RTCP(Real-time Transport Control Protocol)패킷이다. RTCP는 패킷 손실 및 지연 시간 등의 통계 및 제어 정보를 제공하여 통신 서비스 품질의 피드백을 제공한다. 노드 탐색 프로토콜 패킷 유형에 따라 분류하였을 때, Ping 4,611개(38.7%), Pong 3432개(28.8%), FindNode 1919개 (16.1%), Neighbors 1659개(13.9%), ENRRequest 164개(1.4%), ENRResponse 90개(0.8%)로 구성되어 있다.

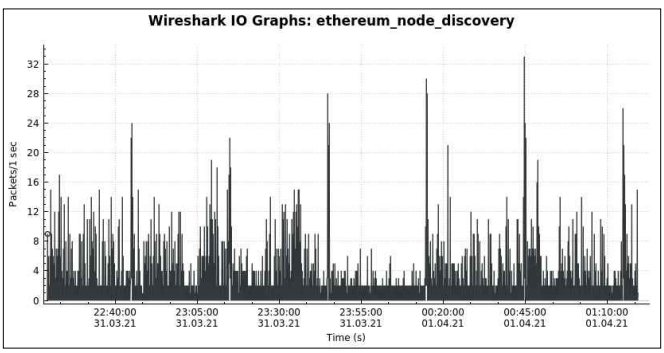


그림 5. 이더리움 노드 탐색 프로토콜 패킷 입출력 그래프

이더리움 노드 탐색 프로토콜 패킷 입출력 그래프를 보았을 때, 30분 주기로 패킷 송수신량이 급증하는 것을 확인할 수 있다. 영향을 주는 패킷을 확인하기 위해, 시간에 따른 패킷별 송수신량을 측정하였다.

패킷 송수신량 급증의 원인은 FindNode와 Neighbors 메시지로, 연결된 피어가 Ping 메시지에 응답하지 않아 새로운 피어를 찾는 경우나 버켓의 노드 목록을 정리하는 주기로 인해 두 가지 메시지 종류의 패킷량이 증가했다.

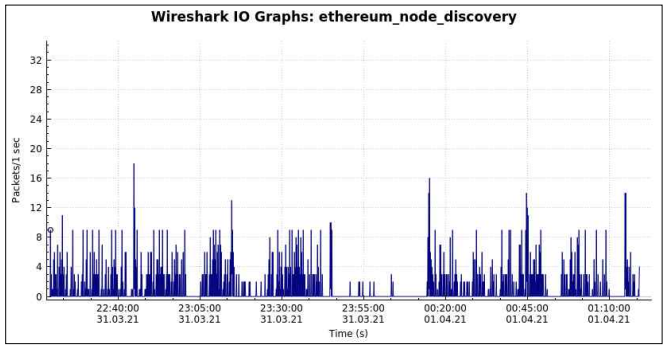


그림 6. FindNode와 Neighbors 메시지 입출력 그래프

Ⅳ. 결 론

본 논문에서는 분산 P2P 네트워크 향상을 위한 이더리움의 노드 탐색 프로토콜을 와이어샤크 해석기에 반영하여 패킷을 수집하고 패킷 메시지 단위로 분석할 수 있게 하였다. 또한 해석기를 사용하여 와이어샤크에서 이더리움 노드 탐색 프로토콜의 네트워크 장애 및 프로토콜 문제점을 확인할 수 있다. 본 논문의 분석을 바탕으로 이더리움 노드 탐색을 측정·분석하여 이더리움 네트워크 성능 향상 연구를 진행할 수 있다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2018R1D1A1B07050380).

참 고 문 헌

[1] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." white paper 3.37 (2014).

[2] Ethereum devp2p,

"https://github.com/ethereum/devp2p", 3월 접속

[3] Kim, Seoung Kyun, et al. "Measuring ethereum network peers." Proceedings of the Internet Measurement Conference 2018. 2018.

[4] Wireshark Display Filter Reference,

"https://www.wireshark.org/docs/dfref/b/bitcoin.html", 3월 접속

[5] Go-Ethereum Client,

"https://github.com/ethereum/go-ethereum", 3월 접속

[6] Ethereum Wiki,

"https://eth.wiki/fundamentals/rlp", 3월 접속

네트워크 침입 탐지를 위한 Stacked Denoising Autoencoder-Deep CNN 모델

이종화[1], 김종욱[1], 최미정[1,2,*]

강원대학교

{jonghwalee, jw.kim, mjchoi}@kangwon.ac.kr

Stacked Denoising Autoencoder-Deep CNN Model for Network Intrusion Detection

Jong-Hwa Lee[1], Jong-Wouk Kim[1], Mi-Jung Choi[1,2,*]

IGP. in Medical Bigdata Convergence, Kangwon National Univ.[1]
Dept. of Computer Science, Kangwon National Univ.[2]

요 약

IT 기술의 발전으로 엣지 컴퓨팅 환경을 사용하는 서비스 공급업체는 사용자에게 높은 수준의 서비스를 제공하고 있다. 이에 따라 사용자와 관련된 다양한 정보들이 단말 장치에 저장되는 동시에 탐지하기 더욱 어려운 최신 사이버 공격의 핵심 목표가 됐다. 단말 장치의 보안을 위해 대표적으로 방화벽, 침입 탐지 시스템 등의 보안 시스템들이 자주 활용되지만, 기존의 침입 탐지 시스템은 탐지 정확도가 낮은 문제점이 존재했다. 사용자에게 더욱 안전한 서비스를 제공하기 위해 많은 분야에서 활용되는 기계 학습을 이용한 침입 탐지 시스템 개발 연구가 활발히 진행되고 있다. 따라서 본 논문에서는 엣지 컴퓨팅 환경에서 단말 장치의 침입 탐지를 위한 기계 학습 모델을 제안한다. 제안하는 모델은 노이즈를 사용하여 손상된 입력 데이터를 압축 및 복구하는 SDAE(Stacked Denoising Autoencoder)와 CNN(Convolutional Neural Network) 모델을 결합한 준지도학습 모델을 제안한다. 또한, 제안한 모델의 가우시안 노이즈 계수를 변경하면서 입력 데이터의 손상된 정도에 따라 모델의 정확도, 예측 시간, 자원의 소모량의 차이를 비교 및 분석했다.

Ⅰ. 서 론

컴퓨터와 모바일 네트워크의 발전으로 클라우드 컴퓨팅 환경이 엣지 컴퓨팅 환경으로 변화하고 있다. 서비스 공급자들은 사용자들에게 더욱 빠르고 질 높은 서비스를 제공할 수 있게 되었다. 이에 따라, 단말 장치에 사용자와 관련된 이름, 전화번호, 주소 등의 개인정보를 수집 및 저장한다[1]. 따라서 단말 장치를 포함한 엣지 컴퓨팅 환경은 DDoS, Flooding 등 다양한 네트워크 공격들의 목표가 된다. 그러므로 사이버 공격의 침입을 막기 위한 방화벽, 안티 바이러스, 침입 탐지 시스템 등의 보안 요소가 필수적이다. 이 중에서 침입 탐지 시스템(Intrusion Detection System)은 의심되거나 불법적인 행위를 탐지하여 사용자에게 알려주는 시스템을 의미한다. 침입 탐지 시스템은 네트워크 침입 탐지에 가장 일반적으로 사용되었지만 낮은 탐지 비율과 높은 오탐률을 가지는 문제가 존재한다[2, 3]. 이러한 문제를 해결하기 위하여 인공 지능을 접목한 침입 탐지 시스템 연구가 진행되고 있다.

기계 학습은 인공 지능의 한 분야로써 크게 지도 학습, 비지도 학습, 강화 학습으로 나누어진다. 지도 학습은 데이터셋의 정답이 표시되어 있는 출력(output) 데이터를 이용하여 데이터를 분류하거나 특정 값을 예측한다. 비지도 학습은 지도 학습과 다르게 정답이 표시되어 있지 않은 데이터를 이용하여 데이터를 분류하거나 상관관계를 분석한다[4]. 강화학습은 인간의 판단 과정을 구현한 학습 방법으로 어떠한 행위를 이행하였을 때 최대한의 보상을 제공하는 방향으로 다음 판단을 결정하는 학습 방법이다[5]. 이 중에서 지도 학습 방법은 상대적으로 학습 데이터에 특히 영향을 많이 받는다. 학습 데이터셋의 특징을 지나치게 반영한 과대적합 문제와 특징이 적게 반영되는 과소적합 문제가 발생할 수 있다. 그러므로 지도 학습은 데이터셋의 적절한 전처리가 필수적이다.

따라서 본 연구에서는 비지도 학습과 지도 학습을 결합한 준지도학습 방법의 학습 모델을 제안한다. 제안하는 모델의 SDAE는 Encoder와 Decoder로 이루어져있다. Encoder에서 입력 데이터에 노이즈를 추가하고 Decoder를 통해 원본 데이터를 복구한다. 그리고 1D-CNN을 여러 개 쌓은 Deep CNN이 복구된 데이터의 특징을 학습하여 정상인지 비정상인지 분류한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에 사용한 모델의 연구 동향을 조사하고, 3장에서는 학습에 사용한 UNSW-NB15 데이터셋에 대한 소개 및 데이터셋의 전처리 과정을 설명하고 제안하는 모델을 설계했다. 4장에서는 전처리가 완료된 데이터를 학습한 SDAE-Deep CNN 모델의 가우시안 노이즈 계수를 변경하면서 비교 실험을 진행했다. 수행한 실험 결과를 통해 학습 모델의 정확도, 예측 시간, 하드웨어의 자원 소모량 등을 비교 분석하고 결론을 다루며 논문을 마친다.

Ⅱ. 연구 동향

1) Denoising Autoencoder

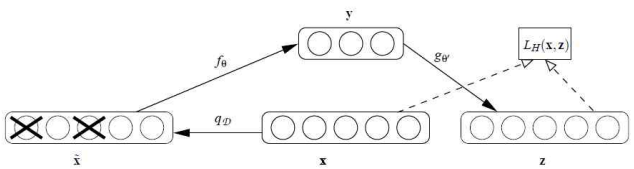
Pacal Vincent et al. 의 연구에서 입력 데이터를 재구성하여 출력하는 Autoencoder 모델을 변형하여 데이터의 노이즈를 제거하는 DAE(Denoising Autoencoder)를 연구했으며 그림 1과 같다[6]. 입력 데이터 x는 확률적으로 노이즈를 만들어내는 함수 q_D를 통해 노이즈가 더해진 $\tilde{x}$ 데이터를 만든다. 그리고 Encoder f_θ는 데이터를 압축하여 y를 생성하고 Decoder $g_{\theta'}$는 y를 재구성하여 복원된 z 데이터를 만들어낸다.

마지막으로 원본 데이터 x와 재구성된 데이터 z를 통해 손실 함수 $L_{H(x,z)}$가 손실율을 계산한다.

DAE는 기존의 DBN(Deep Belief Network)보다 성능이 좋고 분류 문제의 오류를 줄였으며 SVM(Support Vector Machine)의 낮은 분류 성능 문제도 해결했다. 뿐만 아니라 DAE는 이미지의 윤곽을 처리하는 Gabor filter와 유사한 기능을 학습할 수 있으며, 숫자 이미지에서 더 많은 스트로크를 탐지할 수 있다. 완전히 깨끗한 입력(input) 데이터를 학습하는 것보다 손상된 입력 데이터를 복구하여 얻은 입력 데이터를 학습하는 것이 더욱더 객관적이고 안정적인 구조의 데이터를 얻을 수 있어 과적합 문제를 해결할 수도 있다. 따라서 모델이 학습 할 때 더 유용한 특징들을 발견하기 위해 노이즈를 제거하고 고차원 데이터를 저차원 데이터로 변환하는 과정이 더 높은 수준의 재표현이 가능하다고 설명했다. Safa Mohamed et al.의 연구에서 네트워크 침입 탐지를 위한 특정 유닛의 가중치 값을 제거하는 Dropout 기반의 DAE 모델을 제안했으며, 해당 모델은 네트워크 침입 탐지에서 자주 사용하는 NSL-KDD 데이터셋의 정상 데이터만 학습했다[7]. DAE의 은닉 층에서 입력 데이터를 무작위로 제거하였으며, 이것을 학습 데이터의 노이즈라고 정의했다. 해당 연구에서 은닉 층의 유닛수를 32, 24, 16, 8개로 변경하면서 실험했으며, 은닉 층의 유닛수가 8개일 때 90.3%의 가장 높은 정확도를 도출했다.

2) Convolutional Neural Network

CNN은 인간의 뇌에서 시각 감각을 처리하는 메커니즘을 이용하여 만들어졌다. CNN은 커널(kernel)이라는 추출 필터를 이용하여 이미지로부터 자동적으로 복잡한 특징들을 추출한다. Meliboev Azizjon et al.의 연구에서 네트워크 침입 탐지를 위해 1D-CNN 기반의 침입 탐지 모델을 제안했다[8]. 해당 연구에서 Random Oversampling 방법을 이용하여 UNSW-NB15 데이터셋의 불균형성 문제를 해결했다. 그리고 1차원 데이터를 처리하는 1D-CNN을 이용하여 최고 91.2%의 정확도와 91.59%의 F1-score를 도출했다. Peilun Wu et al. 의 연구에서는 네트워크 침입 탐지를 위해 CNN과 RNN(Recurrent Nerual Network)을 결합한 계층적 신경망인 LuNet을 제안했다[9]. LuNet은 CNN과 RNN을 결합하여 LuNet Block를 구성하고 여러 층을 쌓은 모델이다. 해당 연구에서는 학습을 위해 NSL-KDD 데이터셋과 UNSW-NB15 데이터셋을 사용했다. 이진 분류에서는 각각 97.4%와 97.7%의 정확도를 달성했으며, 다중 분류에서는 각각 99.1%와 85.0%의 정확도를 도출했다.



(그림 1) DAE의 구성도.

(표 1) UNSW-NB15 데이터셋의 정상 데이터와 비정상 데이터의 수.

UNSW-NB15	Training set		Testing set		Total Set
	Normal	Abnormal	Normal	Abnormal	
Data	56,000 (22%)	119,341 (46%)	37,000 (14%)	45,332 (18%)	257,673 (100%)

Ⅲ. 모델 설계

1) 데이터셋

본 연구에서는 UNSW-NB15 데이터셋을 사용한다. UNSW-NB15 데이터셋은 Cyber Range Lab에서 IXIA PerfectStorm을 사용하여 실제 네트워크 패킷을 수집 및 가공하여 만든 공공데이터셋이다. UNSW-NB15 데이터셋은 175,341개의 학습 데이터와 82,232개의 실험 데이터가 파일에 저장되어 배포되며 데이터의 수는 표 1과 같다. 표 2는 UNSW-NB15 데이터셋의 45개의 특징들을 정리한 것이다. 또한, UNSB-NB15 데이터셋은 'normal', 'reconnaissance', 'backdoor', 'DoS', 'exploits', 'analysis', 'fuzzers', 'worms', 'shellcode', 'generic' 의 1개의 정상과 9개의 공격 유형으로 분류했다.

2) 데이터 전처리

모델의 학습을 위해 UNSW-NB15 데이터셋의 45개의 특징에서 학습에 불필요한 'id', 'attack_cat' 특징을 제거하고 정답이 표시된 'label' 열을 분리했다. 그리고 기계 학습이 사용하지 못하는 범주형 데이터를 처리하기 위해 'proto', 'service', 'state' 특징들을 실수형 데이터로 변환했다. 데이터셋의 데이터들 간의 편차가 클수록 학습 모델은 데이터의 중요도를 찾을 수 없는 문제가 있다. 따라서 이러한 문제를 해결하기 위해 데이터들을 0과 1 사이의 값으로 정규화를 진행했다.

3) Stacked Denoising Autoencoder & Deep CNN

본 연구에서 제안하는 전체 모델은 그림 4와 같다. 그림 4는 SDAE와 Deep CNN을 결합한 모델이며 SDAE는 그림 2와 같이 DAE의 은닉층을 2개 이상 쌓은 모델이다. Deep CNN 모델의 구성은 그림 3과 같으며 2개의 1D-CNN, Maxpool, Dropoout 계층이 4번 연결된 모델이다. SDAE를 통해 특징들이 더욱 압축 및 재구성되어 분류하기 더욱 좋은 데이터를 얻을 수 있다. SDAE를 통해 데이터의 노이즈를 제거하기 위해 입력 층에서 노이즈를 추가해야 한다. 이때 사용한 노이즈 입력 방법은 가장 일반적인 노이즈 모델인 가우시안 노이즈 모델을 사용했다. 기존 연구에서 가장

(표 2) UNSW-NB15 데이터셋 특징들의 이름과 형태.

Name	Type	Name	Type
id	numeric	dwin	numeric
dur	numeric	tcprtt	numeric
proto	nominal	synack	numeric
service	nominal	ackdat	numeric
state	nominal	smean	numeric
spkts	numeric	dmean	numeric
dpkts	numeric	trans_depth	numeric
sbytes	numeric	response_body_len	numeric
dbytes	numeric	ct_srv_src	numeric
rate	numeric	ct_state_ttl	numeric
sttl	numeric	ct_dst_ltm	numeric
dttl	numeric	ct_src_dport_ltm	numeric
sload	numeric	ct_dst_sport_ltm	numeric
dload	numeric	ct_dst_src_ltm	numeric
sloss	numeric	is_ftp_login	numeric
dloss	numeric	ct_ftp_cmd	numeric
sinpkt	numeric	ct_flw_http_mthd	numeric
dinpkt	numeric	ct_src_ltm	numeric
sjit	numeric	ct_srv_dst	numeric
djit	numeric	is_sm_ips_ports	numeric
swin	numeric	attack_cat	nominal
stcpb	numeric	label	numeric
dtcpb	numeric		

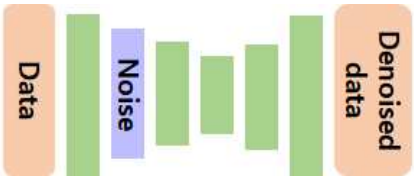
성능이 좋았던 1D-CNN 모델을 이용하여 Deep CNN 모델을 구성한다 [10]. Deep CNN은 여러 층의 CNN 계층을 쌓은 것이다. CNN 모델은 전체가 아닌 필터를 통해 일부분의 데이터에서 특징 맵을 추출하고 상관관계를 계산한다. 따라서 여러 CNN 계층을 통해 데이터셋을 대표할 수 있는 특징을 반복적으로 학습한다. Maxpool 계층은 필터를 통해 추출된 특징 맵에서 비교적 상관관계가 낮은 데이터를 제외한다. 그리고 과대적합 문제를 해결하기 위해 Dropout 계층을 지난다. 마지막으로 분류를 위해 Flatten 계층과 4개의 Dense 계층을 통과하여 정상인지 비정상인지 분류한다.

즉, 제안하는 모델에서 데이터셋의 데이터들을 분류하는 특징들이 노이즈가 추가되고 복구하는 과정에서 낮은 상관관계를 가진 특징들은 제거되고, 높은 상관관계를 가진 특징들을 높은 수준으로 재구성 및 추출한다. 따라서 깨끗한 데이터를 학습하는 것보다 노이즈를 추가하고 복구한 데이터를 학습한 경우가 성능이 좋을 것으로 예상한다.

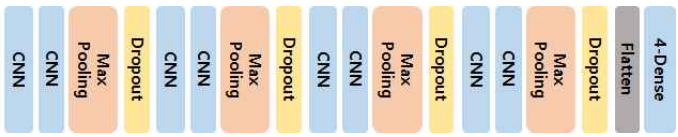
Ⅳ. 실험 및 실험 결과

1) 실험

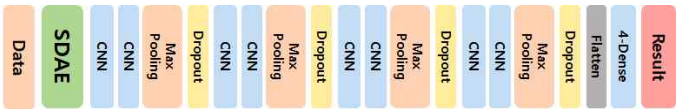
가우시안 노이즈 계수에 따른 SDAE-Deep CNN 모델의 성능 평가를 위해 정확도, F1-score를 측정했다. 그리고 실시간성이 중요한 네트워크 환경에서 빠르고 효율적인 탐지를 위해 예측 시간, GPU 사용량, CPU 사용률을 추가로 측정했다. 정확도 및 F1-score를 계산하는 방법은 수식 (1) - (4)와 같다. TP(True Positive)는 실제 양성인 것을 양성이라고 바르게 예측한 것이고, FP(False Positive)는 실제 음성인 정답을 틀리게 예측한 것이다. TN(True Negative)은 실제 음성인 것을 음성이라고 바르게 예측한 것이고, FN(False Negative)는 실제 양성인 것을 음성이라고 틀리게 예측한 것이다. 실험을 위해 사용한 시스템의 제원은 표 3과 같다.



(그림 2) SDAE의 구성도.



(그림 3) Deep CNN의 구성도.



(그림 4) SDAE-Deep CNN의 구성도.

(표 3) 기계 학습 시스템 제원.

Device	Specification
OS	Ubuntu 18.04.1
CPU	Intel(R) Xeon(R) E3-1275 3.60GHz
RAM	pc4-2133 16GB * 2ea
GPU	NVIDIA RTX 2080 super 8G

가우시안 노이즈는 평균이 0이고 표준 편차가 1인 표준정규분포를 바탕으로 노이즈를 생성한다. 이때 노이즈의 계수는 표준 편차를 의미하며 노이즈의 정도를 나타낸다. 노이즈 계수가 크다는 것은 표준편차가 큰 것을 의미하며 학습 데이터들의 평균으로부터 점점 관련성이 떨어지며 무의미한 노이즈를 생성하게 된다. 따라서 가우시안 노이즈 계수는 0.0, 0.2 - 1.0일 때의 값을 사용한다. 노이즈 계수가 0.0인 경우는 SDAE에 노이즈가 추가되지 않으므로 깨끗한 데이터를 학습한 것이다. 또한, 실험 과정에서 모델의 학습이 정상적으로 진행되는지 확인하기 위하여 학습 데이터의 20%를 비복원추출하여 검증 데이터로 사용했다.

2) 실험 결과

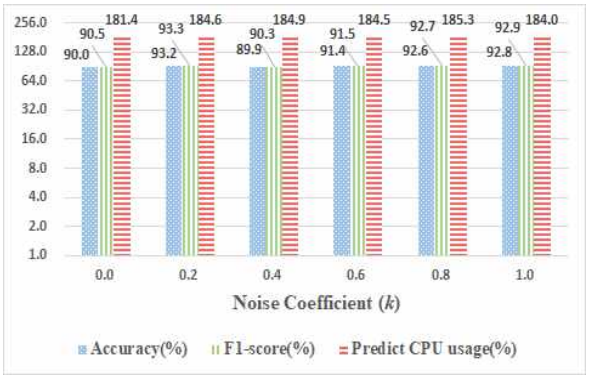
가우시안 노이즈 계수에 따른 실험한 결과는 그림 5와 표 4와 같다. GPU 사용량의 경우 모두 1,371MB로 동일했다. 그 이유는 SDAE와 Deep CNN의 모델 구성이 같으므로 학습할 때 사용된 유닛의 수는 변함이 없기 때문이다. 그림 5에서 노이즈 계수(k)가 0.0인 깨끗한 데이터를 학습할 때보다 계수가 0.2일 때 93.2%로 가장 높은 정확도를 도출했다. 그리고 계수가 0.4일 때 89.9%로 가장 낮은 정확도를 도출했다. 그림 5의 F1-score에서 노이즈가 없는 경우에 90.5%를 도출했지만, 노이즈 계수가 0.2일 때

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$



(그림 5) 노이즈 정도에 따른 정확도, F1-score, CPU 사용률.

(표 4) 노이즈 정도에 따른 예측 시간.

Noise Coefficient (k)	Predict time(s)
0.0	8.5
0.2	7.8
0.4	8.0
0.6	8.3
0.8	7.7
1.0	8.0

데이터를 학습한 경우에 93.3%로 F1-score가 2.8% 더 높다. 즉, 원본 데이터를 그대로 학습하기보다 노이즈를 일정 수준 삽입한 후 복원한 데이터를 학습하는 것이 더 높은 성능을 보인다. 또한, 거듭된 실험을 통해 적당한 노이즈를 삽입하는 것이 더 높은 성능을 보인다는 것을 알 수 있었다. CPU 사용률은 노이즈 계수가 0.0일 때 CPU 사용률이 가장 낮았으며, 계수가 0.8일 때 CPU 사용률이 가장 높았다. 이를 통해 노이즈를 제거하고 복구하는 과정에서 CPU의 연산을 많이 요구하는 것을 알 수 있다. 표 4에서는 계수가 0.0일 때 가장 느리게 결과를 예측했으며 계수가 0.8일 때 가장 빠르게 결과를 예측했다. 표 4를 통해 노이즈의 유무는 예측 시간에 큰 영향을 주지 않는다는 것을 알 수 있다.

노이즈 계수에 따라 예측 시간 및 GPU 사용량에 큰 연관 관계는 찾아볼 수 없지만, 정확도, F1-score 및 CPU 사용률의 경우 큰 연관 관계를 찾아볼 수 있다. 노이즈가 없는 입력 데이터를 학습하는 것보다 SDAE에서 노이즈를 추가하고 복구된 데이터를 Deep CNN을 통해 학습하는 경우가 정확도, F1-score가 더 높았다. 또한, 노이즈를 추가하고 복구하는 과정이 CPU의 연산을 많이 요구한다는 사실을 알아냈다. 즉, 실험을 통해 노이즈 계수가 0.2 일 때 노이즈로 인하여 손상된 데이터를 복구하는 과정(denoising)에서 데이터를 분류할 때 상관관계가 없는 노이즈로 손상된 특징은 제거하고, 큰 상관관계가 있는 특징을 잘 추출하여 높은 성능을 도출했다.

Ⅳ. 결론

클라우드 컴퓨팅 환경의 문제를 보완하기 위해 새롭게 등장한 엣지 컴퓨팅 환경을 공격 대상으로 하는 새로운 공격들이 등장하고 있다. 하지만 기존의 IDS는 이러한 공격들을 잘 탐지하지 못하는 문제가 있다. IDS의 성능을 높이기 위해 기계 학습을 이용한 IDS 연구가 활발히 진행되고 있으며 높은 성능을 도출하고 있다. 따라서 본 논문에서는 엣지 컴퓨팅 환경에서 효율적으로 침입을 탐지할 수 있는 기계 학습의 준지도학습을 이용한 SDAE-Deep CNN 모델을 제안했다. SDAE-Deep CNN 모델은 가우시안 노이즈를 입력 데이터에 추가하고 복구 및 재구성 과정을 통해 데이터의 객관적인 대표성을 가지는 안정적인 특징을 추출한다. 그리고 추출된 특징을 Deep CNN을 통해 학습하고 정상과 비정상 패킷을 분류한다. 실험을 통해 가우시안 노이즈 계수가 0.2일 때 최고 93.2%의 정확도를 도출했으며, 노이즈가 희미하게 존재하거나 많이 존재하는 상태가 더욱 학습 효과가 뛰어났다. 즉, 노이즈가 없는 깨끗한 데이터를 학습하는 것보다 SDAE를 통해 손상된 데이터를 재구성하는 과정에서 데이터를 더욱 잘 분류할 수 있는 유의미한 특징들을 더욱 잘 추출하고 학습했다.

학습 데이터셋의 클래스별 크기가 다를 경우 학습 과정에서 다음과 같은 문제가 발생할 수 있다. 학습 데이터의 수가 적은 클래스는 과소적합 문제가 나타나고, 학습 데이터의 수가 많은 클래스의 경우 과대적합 문제가 나타날 수 있다. 이러한 문제를 클래스 불균형이라고 지칭하며 학습 모델의 정확도를 떨어트리는 원인이 된다. 따라서 향후 연구로 SMOTE 방법 등과 같은 다양한 표본재추출 방법을 적용하여 제안한 SDAE-Deep CNN 모델의 분류 정확도를 높이는 연구를 진행할 예정이다. 또한, UNSW-NB15데이터셋 뿐만 아니라 네트워크 침입 탐지를 위한 다양한 데이터셋을 사용하여 SDAE-Deep CNN 모델의 효과를 검증할 예정이다.

ACKNOWLEDGMENT

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임.(NRF-2020R1A2C1012117).

참 고 문 헌

[1] Jiale Zhang, Bing Chen, Yanchao Zhao, Xiang Cheng and Feng Hu, “Data security and privacy-preserving in edge computing paradigm: survey and open issues,” *Journal of IEEE Access*, vol. 6, pp. 18209 - 18237, Mar. 2018.

[2] Sydney Mambwe Kasongo and Yanxia Sun, “A deep learning method wih filter based feature engineering for wireless intrusion detection system,” *Journal of IEEE Access*, vol. 7, pp. 38597 - 38607, Mar. 2019.

[3] Iftikhar Ahmad, Mohammad Basheri, Muhammad Javed Iqbal and Aneel Rahim, “Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection,” *Journal of IEEE Access*, vol. 6, pp. 33789 - 33795, May 2018.

[4] Jesper Van Engelen and Holger Hoos, “A survey on semi-supervised learning,” *Journal of Machine Learning*, vol. 109, no. 2, pp. 373 - 440, Nov. 2019.

[5] GwiHoon Kim and YongGeun Hong, “Machine learning technology trends in the network,” *Journal of The Korean Institute of Communication Sciences*, vol. 34, no. 10, pp. 38 - 44, Sept. 2017.

[6] Pacal Vincent, Hugo Larochelle, Isabelle Lajoie, Yshua Bengio and Pierre-Antoine Manzagol, “Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, no. 12, pp. 3371 - 3408, Dec. 2010.

[7] Safa Mohamed, Ridha Ejbali and Mourad Zaied, “Denoising autoencoder with dropout based network anomaly detection,” in *proc. of ICSEA 2019: The 14th International Conference on Software Engineering Advances*, pp. 111 - 116, Osaka, Japan, Aug. 2019.

[8] Meliboev Azizjon, Alikhanov Jurnabek and Wooseong Kim, “1D CNN based network intrusion detection with normalization on imbalanced data,” in *proc. of 2020 International Conference on Artificial Intelligence in Information and Communication*, pp. 218 - 224, Fukuoka, Japan, Apr. 2020.

[9] Peilun Wu and Hui Guo, “LuNet: a deep nerual network for network intrusion detection,” in *proc. of 2019 IEEE Symposium Series on Computational Intelligence(SSCI)*, pp. 617 - 624, Xiamen, China, Dec. 2019.

[10] JongHwa Lee, Jiwon Bang, JongWouk Kim and MiJung Choi, “Experimental comparison of network intrusion detection models solving imbalanced data problem,” *Journal of KNOM Review*, vol. 23, no. 2, pp. 18 - 28, Dec. 2020.

네트워크 침입 탐지 성능 향상을 위한 최적화 연구

이지원, 김형태, 김경백

전남대학교

maum-@hanmail.net, akkain0513@gmail.com, kyungbaekkim@jnu.ac.kr

Study of Optimization for improving the Performance of Network Intrusion Detection

Jiwon Lee, Hyungtae Kim, Kyungbaek Kim

Chonnam National University.

요 약

우리의 생활이 네트워크 환경과 밀접해지면서 네트워크 위협이 증가하고 있다. 이에 이상 탐지기반 침입 탐지 연구가 활발히 이루어지고 있으며, 네트워크 침입 탐지 데이터셋과 이를 활용한 기계학습 모델들이 주목받고 있다. 현재 탐지율 향상을 위해 다양한 기계학습 모델들이 사용되고 있지만, 연구 대부분이 다양한 환경을 고려하기 보다는 주어진 한가지 데이터셋에 성능을 향상시키기 위해 모델의 변형 및 고도화에 집중되어 있다. 이러한 모델들은 기존 사용하던 데이터셋과 다른 데이터셋을 사용하면 기존의 성능을 끌어내지 못하였다. 본 논문에서는 최소한의 기능의 순수한 모델을 만들고, 각 모델들의 모델 성능을 끌어내기 위해 하이퍼 파라미터 최적화를 이용하여 각 데이터셋 마다 성능을 측정하였다. 이를 이용하여 각 환경에서 유연한 네트워크 탐지 모델 개발에 도움을 주고자 한다.

Ⅰ. 서 론

현재 사회에서 네트워크 환경은 우리 일상에 더욱더 가까워지고 있다. 그와 동시에 다양한 네트워크 환경에 노출되며, 예상치 못한 사이버 공격이 증가하고 있다. 현재 기계학습 연구가 발달함으로써 기계학습을 도입한 네트워크 침입 탐지 연구가 활발히 이루어 지고, 그에 필요한 다양한 환경의 네트워크 데이터셋 자료 또 한 공개되었다. 이에 다양한 데이터셋에 적용 가능한 범용적인 모델이 필요하지만, 많은 연구들이 다른 환경의 데이터셋은 고려하지 않고, 한가지 데이터셋에 최적화된 앙상블 모델을 개발하는 등 성능 향상을 위해 모델 고도화에 몰두되고 있다[1, 2]. 데이터셋에 불균형을 해소하기 위해 오버샘플링을 적용한 연구도 존재한다[3,4]. 랜덤 오버샘플링을 적용하여 KNN, LSTM 등 다양한 모델로 실험을 한 연구에서는 기계학습 모델에서는 기본값과 딥러닝에서는 임의로 생성한 모델을 사용하여 적용하였다[5]. 우리는 하이퍼 파리미터 설정만으로 충분히 높은 성능을 기대하며, 실험 구성을 위해 간단한 모델을 만들었다. 또 한 최적의 하이퍼 파라미터를 적용하기 위해 Bayesian optimization를 사용했으며, NSL-KDD, UNSW-NB15데이터셋의 사용하여 결과를 내놓았다.

Ⅱ. 관련연구

2.1 데이터셋

우리는 NSL-KDD, UNSW-NB15 데이터셋을 선택했다. 두 데이터의 특징으로는 다양한 공격 방법들이 포함되어 있고, 피처의 개수가 비슷하고 일부 공통된 피처도 존재하지만, 기계학습 모델마다 성능이 다르게 측정되었다. 또 한 다른 데이터와 달리 학습 데이터셋, 테스트 데이터셋이 나뉘어서 제공되기 때문에 좀 더 객관적인 측정이 가능하다는 점이다. 이런 특징을 가지고 있어 활발하게 연구들이 이루어져 있다. [1] 같은 경우에는 Decision Tree모델을 이용한 앙상블 모델을 만들어 높은 정확도를 보이고 있다. Decision Tree 모델 이외에도 딥러닝 모델을 포함한 4가지의 이상의 모델들을 결합한 앙상블 모델에서도 85% 내외에 정확도를 보여주고 있다[2]. Chuanlong Yin의 연구에서는 LSTM의 하이퍼 파라미터 중에 노드의 개수에 대한 성능을 보여주고 있다[11]. UNSW-NB15 데이터셋에 Autoencoder와 SVM을 결합한 모델로 측정한 정확도는 89%, J48로 측정한 결과로는 88%로 의사결정나무 모델도 높은 정확도를 보인다[6,7]. 또 한 데이터셋에 불균형을 해결하기 위해 데이터셋에 랜덤 오버샘플링을 적용한 후 Conv1D모델을 측정한 결과에도 비슷한 정확도를 보여주고 있다[3].

표 1 관련 연구 정확도

Method	Dataset	Result
Decision Tree Bagging Ensemble[1]	NSL-KDD	Accurancy = 85.81%
Ansemble model[2]	NSL-KDD	Accurancy =85.2%
LSTM[6]	NSL-KDD	Accurancy = 83.28%
Autoencoder & SVM [7]	UNSW-NB15	Accurancy = 89.134%
Decision Tree(J48) [8]	UNSW-NB15	Accurancy = 88.3%
Conv1D[5]	UNSW-NB15	Accurancy = 89.4%

2.1 기계학습 모델

K-Nearest Neighbors(KNN)은 입력 후에 학습 데이터들을 k개의 이웃 무리로 구분 짓고, 이웃 중에서 근접한 이웃끼리 거리를 맞춰서 대입하는 것이다. 게으른 학습 방법의 하나로써 테스트 속도가 느리지만, 특정 부류를 구분하는 작업에서 효율이 좋다. 주 파라미터는 K값으로써 이웃들의 수를 설정할 수 있다.

Logistic Regression(LR)는 선형 회귀 모델과 같이 입력된 데이터 특성의 가중치 합을 계산한다. 하지만 선형 회귀와 다르게 입력 데이터에 대한 값이 아닌, 그것에 값이 특정 영역에 속할 확률을 계산하는데 사용된다. 그리고 그 모델의 강도를 조절하는데 역수인 C를 사용하여, C값이 클수록 모델의 규제를 줄일 수 있다.

Support Vector Machine(SVM)는 학습 데이터들을 구분 짓기 위

한 경계를 찾는 것이 목적으로, 경계와 데이터 사이에 마진이라는 여백 공간을 최대로 하기 위해, 마진과 가장 가까운 값인 서포트 벡터를 기준으로 마진을 최대화 하는 경계를 찾는 것이 목적이다.

의사결정나무(DT) 모델은 학습 데이터로부터 특징을 추출하여 트리 형태를 만드는 알고리즘으로써, 다양한 데이터셋에 활용이 가능한 다재다능의 알고리즘이다. 아래에 있는 모델들의 부모격인 알고리즘으로써, 다양한 파생모델들이 존재한다. 주요 파라미터는 max_depth, leaf이다.

Random Forest(RF)는 훈련 데이터에 중복을 허용하지 않는 pasting(페이스팅) 방법을 사용하여 만든 결정 트리의 앙상블이다. 결정나무 모델은 bagging처럼 중복을 허용하여 최고의 특징을 뽑아내기 위해 초점이 맞춰져 있다. 반면 랜덤 포레스트는 무작위로 다양한 특징들을 뽑아낸 후, 그 특징들에서 최적의 해를 뽑아낸다. 그렇기에 편향된 데이터를 분산시켜 과적합을 피할 수 있다.

eXtreme Gradient Boosting(XGB)은 Gradient Boosting Machine(GBM)의 하나의 방법으로 약한 학습기를 결합하여 이전 값의 오차를 결합 나가는 앙상블 모델이다. GBM 보다 좀더 속도 측면에서 개선되었지만, 여전히 속도는 다른 모델에 비해서 느린 편이다. 과접합 문제를 해결하기 위한 여러 가지 기능들이 내장되어 다양한 하이퍼 파라미터를 적용할 수 있다. 많은 하이퍼 파라미터 때문에 다른 검색 방법보다 Bayesian optimization이 하이퍼 파라미터를 찾는데 우수한 성능을 보였다[12].

Light Gradient Boosting Machine(LGBM)은 XGB의 개량 모델로 속도가 XGB보다 빠르다. 일반적으로 의사결정나무 기반 모델들은 균형 트리 분할이지만 LGBM은 리프 중심 트리 분할방식이여 트리 모양이 비대칭인 특징을 가진다. 트리를 생성할 때 최대 손실값을 가진 리프 노드를 지속적으로 분할 하면서 깊은 트리를 만드는 구조이기에 적은 데이터 사용 시 과적합 발생이 쉽다는 단점이 있다.

Deep Neural Network(DNN)은 인공신경망 모델에서 은닉층을 늘려서 사용하는 모델로써, 은닉층이 2개 이상인 학습 모델이다. 딥러닝 중에 기초 모델로써 이를 응용한 여러 모델들이 파생되었다.

Long Short-Term Memory models(LSTM)은 순환신경망 모델로써 Recurrent Neural Networks(RNN)에서의 장기 메모리 문제를 해결하기 위해 만들었다. Gated Recurrent Units (GRU)는 LSTM에서 좀 더 간소화된 모델로 LSTM과 유사한 점이 있어서 두 모델의 성능을 비교한 연구들이 많다.

2.3 최적화 방법

현재 나오는 기계학습 모델 중에는 모델에 사용자가 직접 설정하는 변수를 지정할 수 있으며 이 개념을 하이퍼 파라미터라고 한다. 기계학습 모델 중에서 의사결정나무 모델 같은 경우 트리의 깊이와 리프 노드의 최대 개수등 이 있으며, 딥러닝 모델에서는 epoch 값이나 옵티마이저 설정 등이 있다.

이러한 하이퍼 파라미터를 이용한 최적화 탐색 방법으로는 직접 값을 지정하여 넣는 Manual Search 방법과 모델 하이퍼 파라미터에 넣을 수 있는 변수를 여러 개 집어넣어서 우수한 모델을 찾는 Grid serach 가 대표적이다. 반대로 말 그대로 하이퍼 파라미터를 랜덤하게 집어넣고 그중에서 우수한 값을 뽑아낼 수 있는 Random search 방법이 있으며, Grid Search 보다 좋은 성능을 도출했다[9]. 하지만 위의 방법들은 좋지 않을 확률이 상대적으로 높거나 성능을 확인하기 위한 시간이 너무 오래 걸린다.

그렇기에 우리는 새로운 해결 방안으로 주목되고 있는 Bayesian optimization을 적용하였다[10]. Bayesian optimization은 어느 입력값을 받는 미지의 목적 함수를 상정 후, 그 함숫값을 최대로 만드는 최적의 해를 찾는 것으로 목적을 두어 좋은 파라미터를 찾는 것이다. 현재 대표적인 툴로 Hyperopt가 있다[11].

III. 실험

우리는 머신러닝 모델에서는 KNN, SVM, Logistic regression과 의사결정나무 계열에서는 DecisionTree, RandomForest, LGBM, XGB 모델 그리고 딥러닝 계열의 모델로는 DNN, LSTM, GNU를 사용하였다. 사용한 데이터는 KDD-NSL, UNSW-NB15를 사용했으며 전처리 과정에서 공격 분류는 normal, unomal로 이진 분류하였고, 스트링 기반의 피처들은 라벨 인코딩을 사용하였다.

표 2. default 구성

Model	Configuration	
KNN	default	
SVM	default	
LR	default	
Model	Configuration	
DT	default	
RF	default	
XGB	default	
LGBM	default	
Model	Configuration	compile
DNN	Dense(f*4)-Dropout(0.2)- Dense(f*2)-Dropout(0.2)- Dense(f)-Dropout(0.2)- Dense(1)	optimizer=adam epoch=10
LSTM	LSTM(f)- Dense(1)	optimizer=adam epoch=10
GRU	GRU(f)- Dense(1)	optimizer=adam epoch=10

표 3. 하이퍼 파라미터 검색 조건

Model	Configuration	
KNN	n_neighbors=(1-20), weights=(uniform, distance)	
SVM	C=(0.001-100), max_iter=(50-500)	
LR	C=(0.001-100), max_iter=(50-500), solver =(newton-cg, lbfgs, liblinear, sag, saga)	
Model	Configuration	
DT	max_depth=(3-18), min_samples_leaf=(1-10) min_samples_split=(2-10), criterion=(gini, entropy)	
RF	max_depth=(3-18), min_samples_leaf=(1-10) min_samples_split=(2-10), criterion=(gini, entropy) n_estimators=(50-200)	
XGB	eta=(0.01-0.5), min_child_weight=(1-10), max_depth=(3-18), subsample=(0.1-1), gamma=(0.1-10)	
LGBM	learning_rate=(0.01-0.5), min_child_weight=(1-10), max_depth=(3-18), colsample_bytree=(0.1-1)	
Model	Configuration	compile
DNN	Dense(10-320)-Dropout(0.1-0.3)- Dense(10-320)-Dropout(0.1-0.3)- Dense(10-320)-Dropout(0.1-0.3)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)
LSTM	LSTM(10-320)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)
GRU	GRU(10-320)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)

우리는 모델의 default값과 Bayesian optimization를 사용한 모델의 성능

을 비교하기 위해 각 모델을 default와 optimization으로 구성하였다. 기계학습 모델과 의사결정나무 모델에서는 최소한의 하이퍼 파라미터를 적용한 default 모델과 Hyperopt 툴을 적용한 하이퍼 파라미터 구한 모델을 준비하였다. 딥러닝 default 모델은 표 2처럼 DNN은 3개의 은닉층에 각 피쳐의 개수의 4배, 2배, 1배수의 노드와 과접합 방지를 위해 Dropout비율을 0.2로 설정하였으며, LSTM, GRU에는 한 개의 층으로 노드 개수는 피쳐의 개수로 설정하였다. compile 부분에는 epoch는 10, optimzier은 adam으로 설정하였다. 딥러닝 optimization모델의 경우에는 표 3과 같이 각 노드 개수와 Dropout 함수의 범위 epoch 값과 optimizer을 adam, nadam 중에 선택하는 옵션을 Hyperopt 툴에 적용하여 찾은 하이퍼 파라미터를 적용했다.

이후 KNN, LR XGB, LGBM 같은 모델에서는 난수 생성기를 사용하지 않거나, 일정하게 값이 나오는 모델들은 1회, 나머지 모델들은 각 10회 실험 진행하여 최댓값, 최솟값, 평균을 구하였다.

표 4. KDD-NSL Accurancy

Model	default			optimization		
	max	min	mean	max	min	mean
KNN	0.7720			0.7681		
SVM	0.7690	0.6425	0.7124	0.7760	0.6372	0.7297
LR	0.7021			0.7025		
DT	0.7902	0.7667	0.7785	0.7918	0.7699	0.7773
RP	0.7898	0.7657	0.7721	0.7827	0.7607	0.7731
LGBM	0.7935			0.7931		
XGB	0.7940			0.7933		
DNN	0.8071	0.7332	0.7591	0.7716	0.7350	0.7493
LSTM	0.8111	0.6964	0.7680	0.8006	0.7567	0.7808
GRU	0.8007	0.7520	0.7745	0.8389	0.7776	0.8037

표 5. UNSW-NB15 Accurancy

Model	default			optimization		
	max	min	mean	max	min	mean
KNN	0.8099			0.7819		
SVM	0.8049	0.5577	0.6654	0.8398	0.4682	0.6995
LR	0.743466			0.7434		
DT	0.8968	0.8941	0.8954	0.9029	0.9014	0.9020
RP	0.9030	0.9012	0.9018	0.9037	0.9016	0.9025
LGBM	0.9091			0.9172		
XGB	0.9137			0.9089		
DNN	0.7998	0.7316	0.7741	0.8101	0.7268	0.7921
LSTM	0.9050	0.8648	0.8775	0.9015	0.8791	0.8897
GRU	0.8914	0.8730	0.8833	0.8983	0.8337	0.88255

실험 결과로 대체적으로 default 상태로 돌린것보다 optimization를 설정한 값이 대체적으로 높았다. 특히 딥러닝에서는 대부분의 모델들이 default 모델 보다 optimization 모델이 1-2프로 높은 성능을 보여줬다. 반면 성능에 큰 차이가 없는 모델들도 존재했다. 의사결정나무 계열의 모델들에도 다양한 하이퍼 파라미터가 존재하기에 이를 적용한 otimization 성능에서 딥러닝 모델만큼의 효율을 보여줄 것을 기대했으나, 성능 향상에 큰 영향을 주지 않았다.

IV. 결론

본 논문에서는 최적화를 적용하여 각 모델별로 성능을 측정하였다. 의사결정나무 모델이 자원 효율까지 고려할 경우 딥러닝 모델보다 우위에 있었지만, 딥러닝의 막대한 하이퍼 파라미터 적용 범위에서 일부만 수정하여도 성능 개선의 여지를 보여주었다. 현재 연구에서는 다른 연구보다 딥러닝 연구에 비해 간결하지만, 이러한 기본적인 모델에 하이퍼 파라미터 최적화를 적용하여 성능을 어느 정도 끌어낼 수 있는지 알아보았다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2021-2016-0 - 00314*)

참 고 문 헌

[1]Poulmanogo Illy, Georges Kaddoum et al., "Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning," IEEE Wireless Communications and Networking Conference, 15- 18 April 2019

[2]Xianwei Gao, Chun Shan et al., "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," IEEE Access, Volume 7, 82512 - 8252, June 2019.

[3]Yun-Gyung Cheong, Kinam Park, Hyunjoo Kim, Jonghyun Kim and angwon Hyun, "Machine learning based intrusion detection systems for class imbalanced datasets," Journal of Korea Institute of Information Security & Cryptology, vol. 27, no. 6, Dec. 2017

[4]Abhishek Divekar, Meet Parekh et al., "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," in Proc. of 2018 IEEE 3rd International Conference on Computing, mmunication and Security, pp. 1-8, Kathmandu, Nepal, Oct. 2018.

[5]Jong-Hwa Leew, Jiwon Bang et al., "Experimental Comparison of Network Intrusion Detection Models Solving Imbalanced Data Problem, " KNOM Review '20-02 Vol.23 No.02, 2020.

[6]Chuanlong Yin, Yuefei Zhu, Jinlong Fei and Xinzheng He, "A deep learning approach for intrusion detection using recurrent neural networks," Journal of IEEE Access, vol. 5, pp. 21954-21961, Oct. 2017.

[7]A. F. A. Khan, A. Gumaei and A.Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," IEEE Access, vol. 7, pp. 30373 - 30385, 2019.

[8]Hebatallah Mostafa Anwer, Mohamed Farouk and Ayman A. Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," in 2018 9th International Conference on Information and Communication Systems (ICICS), pp. 157 - 162, IEEE, 2018.

[9]J. Bergstra, and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, Vol. 13, pp. 281-305, February 2012.

[10]J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," Advances in neural information processing systems, pp. 2951-2959, 2012.

[11]James Bergstra, Dan Yamins, David D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms, "PROC. OF THE 12th PYTHON IN SCIENCE CONF. (SCIPY 2013)

[12]Sayan Putatunda , Kiran Rama, "A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost," SPML '18: Proceedings of the 2018 International Conference on Signal Processing and Machine, pp 6 - 10, November 2018.

STIX 기반 사이버 위협 인텔리전스를 활용한 산업제어시스템 네트워크 위협 행위 분석 방법

신동혁, 엄익채*

전남대학교 시스템보안연구센터

206413@jnu.ac.kr, *iceuom@chonnam.ac.kr

Using STIX-based cyber threat intelligence ICS threat behavior analysis method

Shin dong hyuk, Euom Ieck Chae*

System Security Research Center, Chonnam National Univ.

요 약

고도화된 사이버 위협이 지속됨에 따라 신속한 대응을 위한 사이버 위협 정보 공유 체계는 국제적으로 더욱 중요한 관심사가 되었다. 위협 정보 공유를 위한 표준 및 플랫폼 개발 연구가 활발히 진행되고 있어 기업 간 또는 국가 간 표준화된 사이버 위협 인텔리전스를 수집하고, 공유하고 있다. 그러나, 발전소, 제조 시설과 같은 산업제어시스템에 대한 사이버 위협 사례는 지속해서 증가하는 추세임에 반해, 제어시스템 특성을 반영한 위협 인텔리전스의 활용은 아직까지 미미한 단계에 그쳐있다. 본 논문에서는 사이버 위협 인텔리전스를 표준화된 형식으로 그래프 데이터베이스에 구축하고, 저장된 정보를 통해 위협 활동을 탐지하기 위한 방법을 제안한다. 위협 정보는 공격자의 행동 지식 기반 데이터베이스인 MITRE ATT&CK 프레임워크에서 추출한 데이터를 활용하고, 위협 정보를 그래프 데이터베이스에 저장함으로써 기대할 수 있는 이점에 대해 논의한다.

Ⅰ. 서 론

최근 사이버 공격은 IT(Information Technology) 인프라뿐만 아니라 기반시설, 제조 기업 등 OT(Operational Technology) 인프라까지 위협하고 있다. 2020년 기준으로, 2018년 대비 OT 취약점이 약 33% 더 증가하였으며, 취약점의 71%는 원격 네트워크를 통한 공격으로 악용되었다.[1] 이러한 취약점을 이용하여 제어시스템에 대한 공격이 성공하게 되었을 때, 시스템 가용성에 손실을 입을 수 있으며, 심각한 경우 인명 피해까지 발생할 수 있다.

제어시스템에 대한 지능형 지속 위협(APT, Advanced Persistent Threat) 공격은 수개월에 걸쳐 위협 탐지 체계를 우회하여 침투하며, 공격 대상을 모니터링하고, 침투 흔적을 남기지 않도록 은밀하게 수행된다. 일반적으로 IDS/IPS 시스템을 통한 네트워크 모니터링 및 SIEM과 같은 보안 솔루션을 적용하여 위협 탐지를 수행하고 있으나, APT 공격의 특성상 새로운 공격에 대한 침해 대응에 한계가 있다. 공격 유형이 점점 지능화됨에 따라 APT 공격에 대응하기 위해 정보 공유의 필요성이 제기되었다. 위협 정보, 공격자의 행위, 공격 도구 등을 분석한 사이버 위협 인텔리전스(CTI, Cyber Threat Intelligence)는 이러한 정보 공유를 위해 국내외에서 활용되고 있으며, CTI 공유를 위한 국제 표준 규격도 제시되었다.

CTI 공유를 위한 표준은 국제전기통신연합인 ITU-T에서 제정한 CYBEX(Cybersecurity Information Exchange techniques)와 미국 국토안보부와 비영리 연구단체 MITRE에서 개발한 STIX/TAXII(Structured Threat Information Expression/Trusted Automated Exchange of Indicator Information) 등이 있다.

현재 국내외에서 STIX/TAXII 표준이 주류로 자리 잡고 있으며, 본 논문에서는 STIX 표준 규격으로 표현된 CTI 데이터를 그래프 데이터베이스에 저장함으로써 보안 위협 정보 및 상관관계를 시각화한다. 그리고 그래프를 통해 제어시스템에서 발생할 수 있는 네트워크 위협 행위를 식별하기 위한 방법을 제안한다.

Ⅱ. 배경 및 관련 연구

2.1 제어시스템 네트워크 구조 및 보안 위협 정보

MITRE에서는 실제 발생한 침해사고 분석 데이터를 기반으로 공격자가 공격을 설계하고 실행하는데 사용한 TTP(Tactic, Technique, Procedure)를 설명하는 프레임워크를 공개하였다. 또한, 최근 MITRE ATT&CK for ICS를 배포하여 [그림 1]로 나타낸 산업제어시스템 네트워크 환경에 대한 다양한 위협 정보를 제공한다. 또한, 해당 위협 정보들을 STIX 2.0 표준 규격으로 공개하여 많은 보안 솔루션 제조사들이 탐지 및 대응을 위한 기반 기술로써 활용하고 있다.

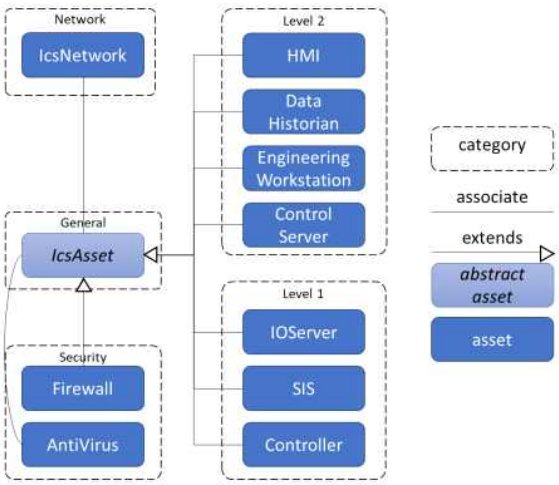


그림 1. 산업제어시스템 네트워크 구조

2.2 STIX 표준

STIX는 사이버 위협 정보를 규격화하고 각 위협 정보의 특성을 관계별로 표현할 수 있는 규격으로, STIX 표준이 2017년에 OASIS CTI CT에서 2.0으로 채택되면서 [표 1]과 같이 12개의 SDO(STIX Domain Object)와 2개의 SRO(STIX Relationship Object)로 구성되었다. STIX 2.0 표준은 위협 분석, 위협 공유 자동화, 탐지 및 대응 자동화와 같은 기능에 활용할 수 있도록 JSON 형식으로 표현된다.

객체	설명
SDO(STIX Domain Object)	
Attack Pattern	공격자가 특정 대상을 장악하기 위해 시도하는 전술, 기술, 절차 (TTP)
Campaign	사이버 공격 관련 공격자의 행동
Course of Action	공격을 예방하고 대응하기 위한 작업
Identity	개인, 조직, 그룹 등을 표현할 수 있는 객체
Indicator	악의적 행동을 탐지하는데 사용할 수 있는 패턴
Intrusion Set	악성 행위의 공통 속성을 가진 악의적 동작과 자원의 그룹화된 집합
Malware	특정 대상을 공격하기 위해 사용되는 악성코드
Observed Data	시스템 및 네트워크에서 관찰된 정보
Report	특정 주제에 관해 분석된 위협 정보의 집합
Threat Actor	악의적인 의도를 가지고 활동하는 것으로 판단되는 개인, 그룹, 조직
Tool	공격을 위해서 사용하는 정상적인 소프트웨어
Vulnerability	공격자가 침투에 사용할 수 있는 취약점
SRO(STIX Relationship Objects)	
Relationship	SDO 객체 간의 관계를 표현
Sighting	CTI의 요소가 관찰되었다는 사실

표 1. STIX 2.0 구성 요소 정의

2.3 CTI를 활용한 보안 위협 분석

국내에서는 제어시스템 대상으로 보안 정보를 수집하기 위해 제어시스템의 자산들을 분류하고, 수집한 로그를 STIX 포맷으로 변환하여 관리하는 연구가 진행되었다.[2] 다른 연구에서는 공개된 여러 보안 위협 분석 보고서에서 CTI를 수집하고, STIX 표준 기반으로 상관관계를 분석하여 그래프 데이터베이스를 구축하였다.[3] 그러나 CTI 수집 범위가 산업제어시스템을 고려하지 않았기 때문에 제어시스템 보안에서 중요한 가시성 확보에 어려움이 있다. 따라서 본 연구는 MITRE ATT&CK for ICS의 데이터 소스를 활용하여 TTP를 고려한 그래프 기반 데이터베이스를 구현하고, 제어시스템에 대한 위협 정보 및 위협 관계를 시각화하여 보안 가시성 확보를 위한 방법을 제안한다.

Ⅲ. CTI 분류 및 관계 분석

MITRE ATT&CK 유형은 [표 2]와 같이 STIX 2.0 표준 규격으로 매핑되어 JSON 형식의 데이터 소스로 활용된다. MITRE ATT&CK for ICS에서 Matrix는 11개의 공격 전술(Tactic)에 대한 공격자의 공격 기법(Technique) 81개를 나타내는데, Matrix와 Tactic은 STIX 객체에 확장 속성으로 표현하여 STIX 2.0에 포함되지 않는 개념을 설명한다. 또한, 50개의 위협 해소(Mitigation), 10개의 공격 그룹(Group), 17개의 공격 도구(Software)를 정의하고 있어, STIX 2.0 SDO로 매핑이 가능하다.. 절차(Procedure)는 ATT&CK에서 정의하고 있지 않은 개념이지만, ATT&CK 유형 간의 관계를 표현하기 위해 STIX 2.0 SRO 형태로 매핑한다.

ATT&CK 유형	STIX 객체 유형	확장 여부
Matrix	x-mitre-matirx	O
Tactic	x-mitre-tactic	O
Technique	attack-pattern	X
Procedure	relationship	X
Mitigation	course-of-action	X
Group	intrusion-set	X
Software	malware or tool	X

표 2. ATT&CK 유형에 대한 STIX 2.0 SDO 매핑

이렇게 매핑된 ATT&CK 유형은 STIX Relationship 객체를 통해 공격에 활용된 기술(Attack Pattern), 공격 그룹(Intrusion Set), 공격 도구(Malware or Tool), 대응 방법(Course of Action)에 대해 사용(Uses)과 위협 해소(Mitigate) 관계로 나타낸다. 각 SDO간의 관계는 [그림 2]와 같이 설명할 수 있다.

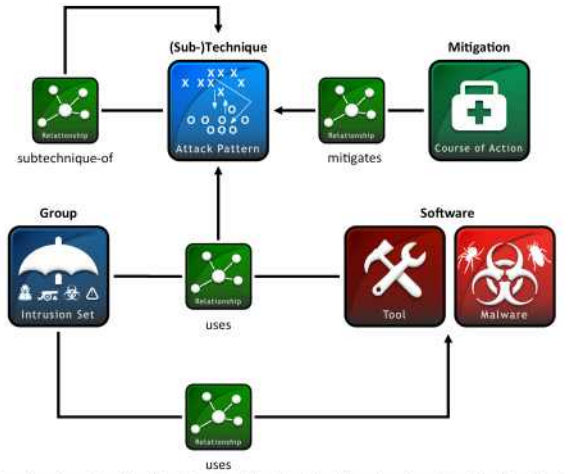


그림 2. ATT&CK 유형과 STIX 2.0 SDO간 관계도[4]

Ⅳ. 그래프 데이터베이스 구현 및 위협 행위 분석

STIX 2.0 형식으로 표현된 MITRE ATT&CK for ICS는 그래프 데이터베이스를 구현할 때 노드를 간소화하기 위하여 [그림 3-(a)]와 같이 4개의 노드로 표현하였다. Attack-pattern과 course-of-action은 위협 해소(mitigation) 및 사용(uses) 관계로 구분할 수 있으므로, Technique 노드로 통합하고, Malware는 Software 노드로, Intrusion-set은 Group 노드로 구분하였다. 더욱 상세한 위협 분석을 위해, 별칭(Alias) 노드를 추가하여 malware 및 intrusion-set에서 관계가 있는 다른 공격 도구나 공격 그룹이 존재하면, 포함하도록 구현하였다.

결과적으로 MITRE ATT&CK for ICS에서 추출한 데이터 소스를 그래프 데이터베이스에 저장하면, [그림 3-(b)]와 같이 시각화된 정보로 객체별, 관계별 CTI를 관리하고 분석할 수 있다. [그림 3-(b)]으로 표현된 그래프는 Group, Technique, Software, Alias로 구성된 총 203개의 노드와 Alias, uses, mitigation으로 정의된 총 552개의 간선으로 구성된다.

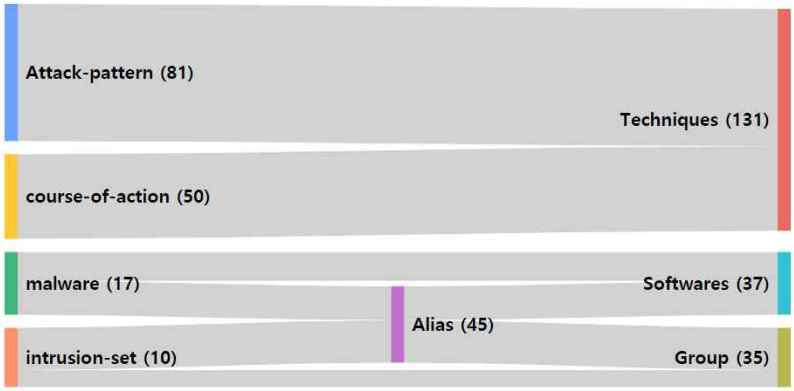


그림 3-(a) 그래프 데이터베이스 구축을 위한 노드 재구성

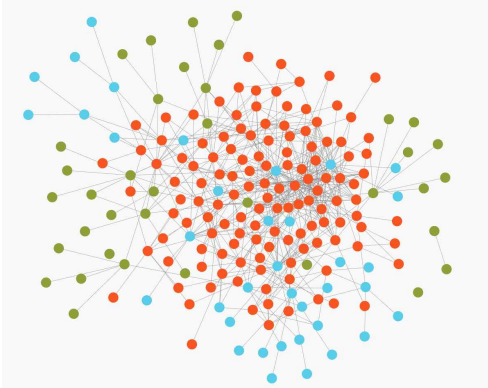


그림 3-(b) ATT&CK 그래프 데이터베이스 구축

본 연구에서는 Neo4j 그래프 데이터베이스를 사용하였으며, 사이버 위협 행위 분석은 Cypher 쿼리로 위협 정보를 시각화하였다. 2017년에 발생한 NotPetya 랜섬웨어를 예로 들면, [그림 4]와 같이 MITRE ATT&CK for ICS의 TTP 매트릭스 기반으로 관련 정보가 추출되는 것을 볼 수 있다. 또한, 쿼리 결과에서 그래프가 아닌 Text 형식으로 출력하면 [그림 5]와 같이 관련된 공격자와 공격 기법을 명시하고, 다른 출처에서 같은 공격으로 활용되었던 악성코드, 대응 전략 및 분석 보고서 등 추가적인 세부 정보(Description)를 얻을 수 있다.

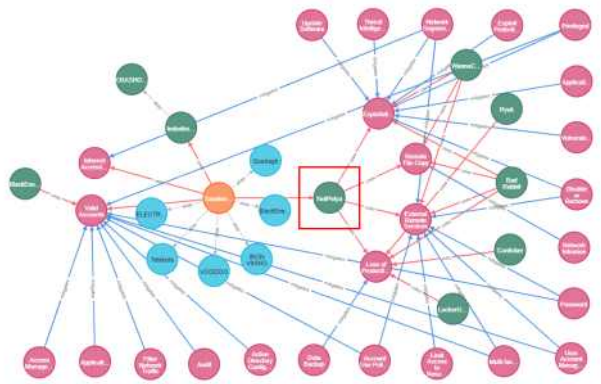


그림 4 NotPetya 랜섬웨어 쿼리 결과


```
  "properties": {
"created": "2019-03-26T15:02:14.907Z",
"name": "NotPetya",
"modified": "2020-06-18T20:27:49.511Z",
"description": "[NotPetya]
(https://attack.mitre.org/software/S0368) is malware that
was first seen in a worldwide attack starting on June 27,
2017. The main purpose of the malware appeared to be to
effectively destroy data and disk structures on
compromised systems. Though [NotPetya]
(https://attack.mitre.org/software/S0368) presents itself
as a form of ransomware, it appears likely that the
attackers never intended to make the encrypted data
recoverable. As such, [NotPetya]
(https://attack.mitre.org/software/S0368) may be more
appropriately thought of as a form of wiper malware.
[NotPetya](https://attack.mitre.org/software/S0368)
contains worm-like features to spread itself across a
computer network using the SMBv1 exploits EternalBlue and
EternalRomance.(Citation: Talos Nyetya June 2017)
(Citation: Talos Nyetya June 2017)(Citation: US-CERT
NotPetya 2017)(Citation: ESET Telebots June 2017)",
"id": "malware--5719af9d-6b16-46f9-9b28-fb019541ddbb",
"type": "malware",
"version": "1.2"
  }
```


그림 5 text로 표현된 NotPetya 랜섬웨어 쿼리 결과

V. 결론

본 연구에서는 STIX 표준으로 작성된 산업제어시스템 CTI를 활용하여 그래프 데이터베이스로 시각화하고, 시각화된 정보를 통해 위협 분석을 수행하는 방법을 제안하였다. 제어시스템은 서로 다른 필드 장비와 산업용 프로토콜을 사용하므로 복잡한 네트워크 구성으로 인해 사이버 공격으로부터 위협 행위를 식별하기 어렵고, 공격 간의 관계를 명확히 표현하기가 어려우나 본 연구가 제안하는 방법으로 보안 위협 분석을 위한 가시성 확보 문제를 개선하였다. 향후에는 CTI에 대한 신뢰도 할당 모델을 적용하여 우선순위에 따른 위협 탐지 기법을 연구할 계획이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(NO.2019-0-01343. 융합보안핵심인재양성)

참 고 문 헌

[1] Claroty, “Claroty Biannual ICS Risk & Vulnerability Report: 2H 2020”, 2021

[2] 최종원, 김예솔, 민병길, CTI 모델 활용 제어시스템 보안정보 수집 방안 연구. 정보보호학회논문지, 28(2), pp. 471-484, 2018

[3] Lee, Seulgi, et al. "Managing cyber threat intelligence in a graph database: Methods of analyzing intrusion sets, threat actors, and campaigns." 2018 International Conference on Platform Technology and Service (PlatCon). IEEE, 2018.

[4] MITER Corporation, “How ATT&CK implements and extends the STIX format”, 2021. (https://github.com/mitre/cti/blob/master/USAGE.md)

Opcode 디스어셈블러와 기계 학습 기반의 트랜잭션 패턴 분석을 이용한 이더리움 스캠 컨트랙트 및 악성 유저 탐지 시스템 설계

이채현 O*, 고경찬*, 우종수**, 홍원기*

*포항공과대학교 컴퓨터공학과

**포항공과대학교 정보통신연구소

{chlee0211, kkc90, woojs, jwkhong}@postech.ac.kr

A Design of Ethereum Scam Contract and Malicious User Detection System using Opcode Disassembler and Machine Learning based Transaction Pattern Analysis

Chaehyeon Lee O*, Kyunchan Ko*, Jongsoo Woo**, James Won-Ki Hong*

*Department of Computer Science and Engineering, POSTECH

**Graduation School of Information Technology, POSTECH

요 약

블록체인의 데이터 불변성, 데이터 투명성은 사람들로 하여금 블록체인상의 데이터를 과도하게 신뢰하도록 하며, 특히 스마트 컨트랙트 코드의 타당성을 일반 사용자가 판단하는 것이 매우 어렵다. 이는 사용자들을 여러 스캠에 취약한 상태로 만들며 실제로 막대한 피해를 초래하는 이더리움 기반의 스캠이 많이 발생하고 있다. 따라서 본 연구에서는 이더리움 네트워크상의 스캠 컨트랙트와 이를 배포하는 악성 유저 (어카운트)를 식별할 수 있는 탐지 시스템을 제안한다. 스마트 컨트랙트의 바이트 코드를 분석하고, 스캠 컨트랙트와 연관된 트랜잭션 패턴 분석을 통해 스캠 컨트랙트 및 악성 유저를 탐지할 수 있는 방법을 소개하고 향후 연구 방안에 대해 기술한다.

I. 서 론

이더리움 [1]은 비트코인 이후 등장한 2 세대 블록체인이다. 비트코인이 화폐의 기능, 즉, 전자 지불 시스템으로써의 역할에 집중한다면, 이더리움은 스마트 컨트랙트 개념을 지원함으로써 거래나 결제 이외에도 다양한 목적으로 애플리케이션을 개발하고, 운영할 수 있게 플랫폼으로써의 기능을 제공한다.

스마트 컨트랙트는 Nick Szabo 에 의해 제안된 것으로, 기존의 서면 계약을 코드로 구현하고 특정 조건이 만족되었을 시 계약이 이행되도록 프로그램화하여 자동화하는 데 목표가 있다. 이더리움은 이러한 스마트 컨트랙트 개념을 블록체인에 도입하여, 신뢰하지 않는 당사자 사이에서도 자동화된 계약을 실행할 수 있게 한다. 따라서 특정 기능을 포함한 디지털 계약을 작성하고자 하는 사용자는 누구든 스마트 컨트랙트를 작성하고, 컴파일한 후 이더리움 네트워크에 배포할 수 있다. 블록체인을 기반으로 함에 따라 누구든 스마트 컨트랙트 실행과 관련한 데이터를 확인할 수 있고, 삭제할 수 없다.

하지만 이러한 데이터 불변성, 데이터 투명성을 제공하는 블록체인 기술의 장점은 사용자로 하여금 블록체인 네트워크의 사용자들은 모두 올바른 방향으로 행동한다고 생각하게 만들어 스마트 컨트랙트를 포함한 블록체인상의 데이터를 과도하게 신뢰하도록 현혹한다. 또한 배포된 스마트 컨트랙트의 코드는 퍼블릭 하게 공개되어 있어 누구나 확인할 수 있지만, 일반 사용자들이 스마트 컨트랙트 코드의 타당성을 판단하는 것이 매우 힘들다. 따라서 스마트 컨트랙트의 안정성을 간과하게 함에 따라 여러 스캠(사기)에 취약한 상태가 된다.

실제로 불특정 다수를 속이기 위한 신용 사기가 여러 암호화폐를 이용해 빈번하게 발생하고 있고, 폰지 스캠, 피싱 스캠 등 다양한 형태의 스캠 컨트랙트가 무분별하게 배포되어 막대한 피해를 초래하고 있다. 2019 년에는 암호화폐 스캠으로 인한 피해 금액이 40 억 달러에 이르렀으며 암호화폐 스캠으로 인해 이더리움 21,000 개가 도난당한 사례가 있다. 따라서 블록체인 사용자들을 불법 스캠으로부터 보호하기 위한 시스템이 필요하다고 판단된다.

따라서 본 연구에서는 사용자를 스캠으로부터 보호하기 위해 이더리움 네트워크상에 배포된 주요한 스캠 컨트랙트를 식별하고, 스캠 컨트랙트를 배포하는 악의적인 유저 (악성 어카운트)를 탐지할 수 있는 시스템을 설계하였다. 배포된 스마트 컨트랙트의 바이트코드를 Opcode 로 변환해 주는 Opcode 디스어셈블러와 기계 학습을 이용해 스캠 컨트랙트와 연관된 트랜잭션 패턴을 분석하여 특정 스마트 컨트랙트의 스캠 여부와 스캠 컨트랙트를 배포하는 악성 유저를 식별할 수 있는 시스템을 제안한다.

II. Background

스캠 컨트랙트는 스캠의 종류에 따라 정상 컨트랙트와 구별되는 트랜잭션 패턴을 보일 수 있다. 특히, 사용자는 스캠의 종류에 따라 스마트 컨트랙트에 일정 코인을 전송하거나, 스마트 컨트랙트로부터 일정 코인을 전달받는다. 폰지 스캠의 경우, 일정 코인을 보내면 해당 코인보다 더 많은 코인을 되돌려 받는 패턴을 보이게 되는데 그림 1 은 폰지 스캠의 유형 중 하나의 예시를 보여준다. 투자자가 스마트 컨트랙트에게 일정 금액을 보내면, 스마트 컨트랙트는 해당 투자자의 어카운트와 투자 금액을 기록한다. 이후 새로운 투자자가 스마트 컨트랙트에 기록되어 있는 것보다 10% 많은 이더를 보낼 시, 스마트

컨트랙트는 자신이 기억하고 있는 이전 투자자에게 해당 금액을 전송하고 새로운 투자자의 정보로 기록을 업데이트한다. 이 과정이 반복되어 새로운 투자자가 계속해서 스마트 컨트랙트에 이더를 보낸다면 이전 투자자는 자신이 투자한 금액보다 10% 이상의 이익을 얻게 된다. 스마트 컨트랙트는 아무 이더도 지니지 않고 새로운 투자자로부터 이전 투자자에게 이익을 전달하는 역할만을 수행

```
pragma solidity >=0.4.0 < 0.7.0;

contract Ponzi {
   address payable public currentInvestor;
   uint public currentInvestment = 0;

   function () external payable {
      uint minimumInvestment = currentInvestment * 11/10;
      require(msg.value > minimumInvestment);

      address payable previousInvestor = currentInvestor;
      currentInvestor = msg.sender;
      currentInvestment = msg.value;

      previousInvestor.send(msg.value);
   }
}
```

하며, 스마트 컨트랙트의 구현에 따라 다양한 형태의 폰지 스캠을 작성할 수 있다.

그림 1 폰지 스캠 예시

피싱 스캠은 대가를 지불하기 위한 목적으로 코인을 스마트 컨트랙트에 전송하기 때문에 피싱 스캠 컨트랙트는 특정 패턴의 이더를 꾸준히 수신하는 경향이 있다.

III. 이더리움 스캠 컨트랙트 및 악성 유저 탐지 시스템

본 연구에서는 스마트 컨트랙트의 바이트 코드와 트랜잭션 패턴 분석을 통해 이더리움 네트워크상의 스캠 관련 컨트랙트를 식별하고, 악성 유저(악성 이더리움 어카운트)를 탐지하는 시스템을 디자인하였다.

1. 전체 아키텍처

그림 2 는 이더리움 스캠 컨트랙트 및 악성 유저 탐지 시스템의 전체 아키텍처를 보여준다. 탐지 시스템은 스캠/정상 스마트 컨트랙트 수집 모듈, 스마트 컨트랙트 Opcode 분석기, 트랜잭션 수집 및 특징 추출 모듈, 기계학습 기반의 악성 유저 판별 모듈, 데이터 베이스로 구성된다. 각 모듈은 별도로 개발되어 독립적으로 운영된다.

일부 공개 사이트에서는 스캠이라고 판별된 스마트 컨트랙트의 바이트 코드 또는 스캠 컨트랙트 어카운트를 공개하고 있다. 스캠/ 정상 스마트 컨트랙트 수집 모듈에서는 공개된 사이트로부터 스캠 컨트랙트와 정상 컨트랙트를 수집한다. 수집된 두 분류의 컨트랙트는 분류가 결정되지 않은 임의의 스마트 컨트랙트의 분류를 결정하기 위해 사용될 수 있다. 수집된 스마트 컨트랙트는 Ⅲ-2 에서 소개할 스마트 컨트랙트 Opcode 분석기의 입력으로 전달된다. Opcode 분석기를 이용하면 사전에 공개되지 않은 스캠 컨트랙트를 식별할 수 있어 데이터 셋을 확장하는데 활용할 수 있다. 스마트 컨트랙트 Opcode 분석기는 Ⅲ-2 에서 자세한 명세를 제공한다.

Ⅱ에서 설명한 바와 같이 스캠 컨트랙트는 스캠의 종류에 따라 특정 트랜잭션 패턴을 보일 수 있다. 스캠 컨트랙트는 스캠 서비스를 이용하는 여러 사용자들과 상호 작용하게 되고, 여러 트랜잭션을 발생시킨다. 동일한 스캠은 일정한 트랜잭션 패턴을 보일 것이라 예상됨에 따라 특정 컨트랙트와 연관된 트랜잭션의 패턴을 분석하여 해당 컨트랙트가 스캠 컨트랙트인지 아닌지, 그리고 임의의 두 컨트랙트가 동일한 유형의 컨트랙트인지를 판별하는데 활용할 수 있다. 따라서 트랜잭션 수집 및 특징 추출 모듈에서는 스마트 컨트랙트와 연관된 트랜잭션을 이더리움 풀 노드로부터 수집하고, 트랜잭션 패턴을 정의하기 위해 트랜잭션과 연관된 특징들을 추출한다. 트랜잭션 패턴 분석에 활용 가능한 특징들로는 스마트 컨트랙트로 코인을 전송한 트랜잭션의 수, 스마트 컨트랙트가 코인을 전달한 트랜잭션의 수, 컨트랙트로 전송된 이더의 양, 컨트랙트로부터 전달받은 이더의 양, 스마트 컨트랙트와 연관된 트랜잭션의 생애 주기, 연관된 두 트랜잭션 사이의 평균 시간 등이 있다.

기계 학습 기반의 악성 유저 판별 모듈에서는 추출된 트랜잭션 특징들을 기계학습을 통해 학습한다. 스캠인지 아닌지를 이진 분류하기 위해 GBM, DRF, XGBoost 등의 기계학습 분류 모델을 활용한다. 최적의 분류 모델을 생성하기 위해 H2O 의 AutoML 기능을 활용하여 자동으로 학습 모델을 생성하고 최상의 성능을 도출하는 모델을 선정한다. 학습된 분류 모델은 서버에 저장되어 특정 컨트랙트 어카운트의 트랜잭션 특징이 입력으로 전달되면 해당 어카운트가 스캠 컨트랙트를 배포한 악성 유저인지, 아닌지를 식별하는데 사용할 수 있다.

2. 스마트 컨트랙트 Opcode 분석기

동일한 유형의 스캠 컨트랙트의 경우, 스마트 컨트랙트의 코드 사이에 유사성이 있을 수 있다. 스마트 컨트랙트는 EVM 이 실행할 수 있는 형태인 바이트 코드로 변환되어 저장되는데 이러한 바이트 코드는 스마트 컨트랙트의 내용을 이해하기에 적합하지 않은 형태로 바이트 코드를 이용해 스마트 컨트랙트 간 유사성을 파악하는 것이 어렵다. 따라서 스마트 컨트랙트 Opcode 분석기를 이용해 분류가 정해지지 않은 스마트 컨트랙트가 스캠 컨트랙트인지 아닌지를 식별하고자 한다.

스마트 컨트랙트 Opcode 분석기는 바이트 코드 디스어셈블러, Opcode 분석 모듈, 스캠 컨트랙트 판별 모듈로 구성된다(그림 3). 먼저, 바이트 코드 디스어셈블러를 이용해 사전에 수집된 정상 스마트 컨트랙트와 스캠 컨트랙트의 바이트 코드를 Opcode 로 변환한다. 바이트 코드의 변환을 위해서 공개되어 있는 디컴파일러 또는 디스어셈블러를 활용한다. 사용 가능한 오픈 툴은 Ⅳ-2 에서 소개한다.

변환된 Opcode 는 스마트 컨트랙트 구현에 따라 다수의 Opcode 명령으로 이루어져있고, 스캠의 종류에 따라 특정한 Opcode 패턴을 가질 것이라고 가정한다. 스캠 컨트랙트의 경우, 정상 컨트랙트와 구별할 수 있는 특정한 Opcode 특성을 가질 수 있기 때문에, Opcode 분석 모듈은 변환된 Opcode 의 명령어를 통계적으로 분석하여 컨트랙트를 구성하는 명령어 집합을 구한다. 또한, 이 과정에서는 컨트랙트 별 특성을 파악하기 위해 기계 학습을 활용할 수 있다. 통계 분석 및 기계 학습 결과를 통해

스캠 컨트랙트 판별 모듈에서는 주어진 스마트 컨트랙트가 스캠의 성향을 띠는지 아닌지를 구별할 수 있다.

IV. 관련 연구

1. 관련 연구

[2] 는 폰지 스캠의 유형과 유형별 보안 문제에 대해 소개하고, 공개되어 있는 컨트랙트를 유형별로 통계 분석하였다. 이더리움상의 폰지 스캠과 관련한 트랜잭션의 수를 비교하고 스캠의 생애 주기에 대해 설명함으로써 폰지 스캠이 얼마나 많은 영향을 주고 있는지 분석했다.

[3] 는 스마트 컨트랙트 바이트 코드의 Opcode 를 기반으로 특징을 추출하여 이더리움 네트워크상의 폰지 스캠 컨트랙트를 탐지할 수 있는 모델을 디자인했다. operand 와 suffix 를 제외하고 74 개의 서로 다른 Opcode 를 이용해 스마트 컨트랙트를 표현할 수 있는 Opcode set 을 정의한다. 그리고 Opcode 시퀀스와 set 의 길이 등을 feature 로 하는 모델링 전략을 제시하였다. [4] 는 데이터 마이닝 기술과 기계 학습을 이용해 폰지 스캠 컨트랙트를 탐지할 수 있는 모델을 제안했다. 폰지 스캠의 사기 행위를 식별하기 위해 해당 연구에서는 Ether flow graph 를 이용해 7 가지의 특징을 추출하여 account feature 로 활용하였다. 그리고 Opcode 를 code feature 로 활용하였다. Account feature, opcode feature, Account + Opcode feature 의 세 카테고리로 기계 학습을 이용해 폰지 스캠 컨트랙트를 분류해본 결과, Account + Opcode 를 이용했을 때 0.94 의 Precision 을 얻을 수 있었다. [5] 에서는 스마트 컨트랙트의 바이트 코드를 Opcode 로 변환하고, Opcode 별 빈출 정도 등을 통계적으로 분석하여 폰지 스캠 컨트랙트와 정상 컨트랙트를 구분해보았다.

[6] 에서는 네트워크 임베딩을 통해 이더리움 네트워크의 피싱 스캠을 탐지하는 방법론을 제안한다. 이더리움 트랜잭션 히스토리로부터 특징을 추출하고 네트워크 임베딩 알고리즘을 이용하여 피싱 노드를 식별하였다.

소개한 관련 연구에서는 온라인상에 공개되어 있는 소스들로부터 스캠 컨트랙트를 수집하고, 자주 등장하는 Opcode 또는 폰지 스캠 컨트랙트와 상호 작용하는 유저의 특징 등을 이용해 폰지 스캠 컨트랙트를 구별하는 방법론을 제안했다. 공개되어 있는 데이터가 매우 한정되어 있어 기계 학습 및 스캠 컨트랙트 식별을 위해 활용 가능한 데이터 셋이 매우 적다고 판단된다. 따라서 본 연구에서는 Opcode 의 특징을 이용해 자체적으로 여러 스캠 컨트랙트를 수집하여 스캠 컨트랙트 식별을 위한 기계 학습의 데이터 셋으로 활용하고자 한다. 또한, 특정 스캠에만 한정되지 않고 여러 종류의 스캠 컨트랙트를 탐지할 수 있는 범용 시스템을 제안한다.

2. 바이트 코드 디스어셈블러

본 논문에서 제안하는 스마트 컨트랙트 Opcode 분석기 내의 바이트 코드 디스어셈블러는 공개되어 있는 오픈 소스 및 툴을 이용할 수 있다.

Etherscan 은 바이트 코드를 Opcode 로 변환하는 온라인 Opcode 디스어셈블러를 제공한다. 웹사이트에 접속하여 변환을 원하는 스마트 컨트랙트의 바이트코드를 입력하면 Opcode 로 변환하여 결과를 출력한다. 그림 4 는 Etherscan 의 디스어셈블러 예시를 보여준다. 바이트 코드를 입력한 결과, 해당 코드를 구성하는 Opcode 의 목록을 반환해주었다.

EVM Bytecode Decompiler 는 node.js 기반의 바이트 코드 디컴파일러 기능을 제공한다. API 를 제공하여 raw 바이트 코드를 가져오거나, 바이트 코드로부터 Opcode, function, event 등을 추출하는 기능을 제공한다.

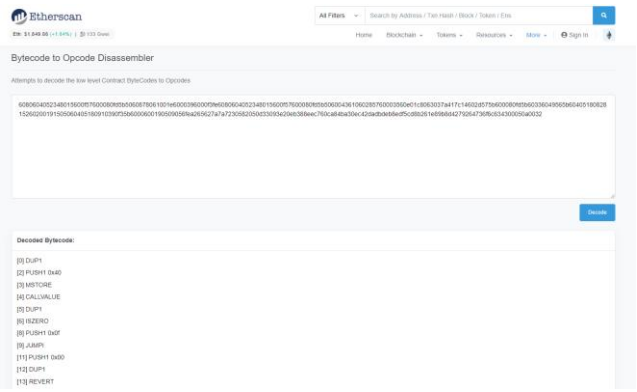


그림 4 Etherscan Bytecode Disassembler 예시 화면

V. 결론 및 향후 연구

본 연구에서는 이더리움 네트워크상의 스캠과 연관된 스마트 컨트랙트를 식별하고, 스캠 활동과 관련 있는 악성 유저를 탐지하기 위한 시스템을 제안한다. 스마트 컨트랙트의 바이트 코드를 Opcode 로 변환한 후 통계 분석하여 스캠 컨트랙트의 특성을 띠는지 확인하고, 트랜잭션 패턴 분석을 통해 스캠 컨트랙트와 연관된 악성 유저 및 악성 행위를 탐지하고자 한다. 본 디자인을 통해 특정 스캠에 한정 짓지 않고 범용 스캠 컨트랙트를 탐지할 수 있을 것이라 기대한다. 향후 연구를 통해 본 연구에서 제안하는 시스템을 구현하고, 시스템을 통해 수집한 실제 데이터 셋을 이용해 탐지 정확도를 측정해 본다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (IITP-2021-2017-0-01633*, 대학 ICT 연구센터육성지원사업, 2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)

참 고 문 헌

[1] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151.2014 (2014): 1-32.

[2] Torres, Christof Ferreira, and Mathis Steichen. "The art of the scam: Demystifying honeypots in ethereum smart contracts." 28th {USENIX} Security Symposium ({USENIX} Security 19). 2019.

[3] Peng, Jianxi, and Guijiao Xiao. "Detection of Smart Ponzi Schemes Using Opcode." International Conference on Blockchain and Trustworthy Systems. Springer, Singapore, 2020.

[4] Chen, Weili, et al. "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology." Proceedings of the 2018 World Wide Web Conference. 2018.

[5] Jung, Eunjin, et al. "Data mining-based ethereum fraud detection." 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019.

[6] Wu, Jiajing, et al. "Who are the phishers? phishing scam detection on ethereum via network embedding." IEEE

Transactions on Systems, Man, and Cybernetics: Systems (2020).

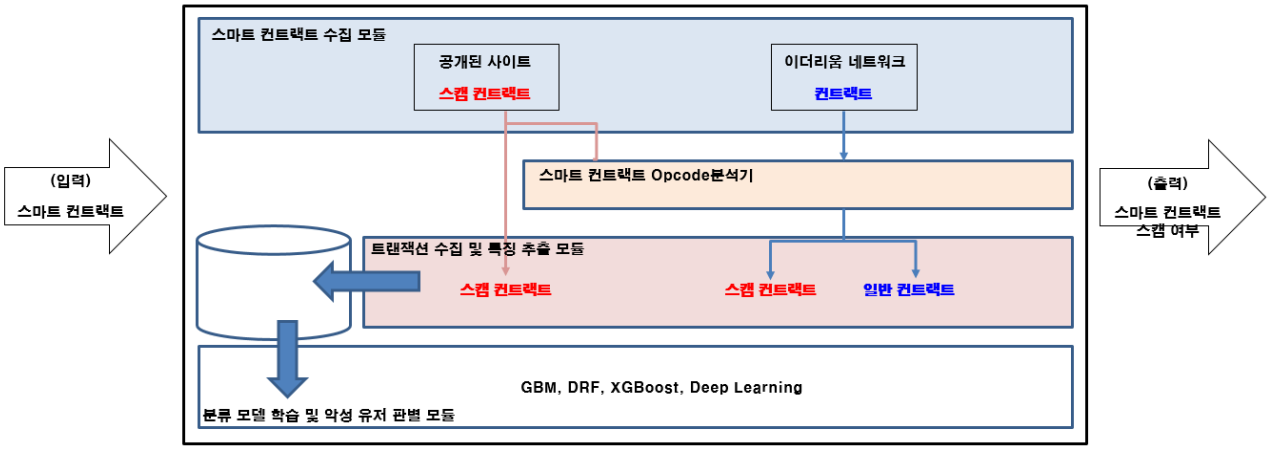


그림 2 이더리움 스캠 컨트랙트 및 악성 유저 (어카운트) 탐지 시스템

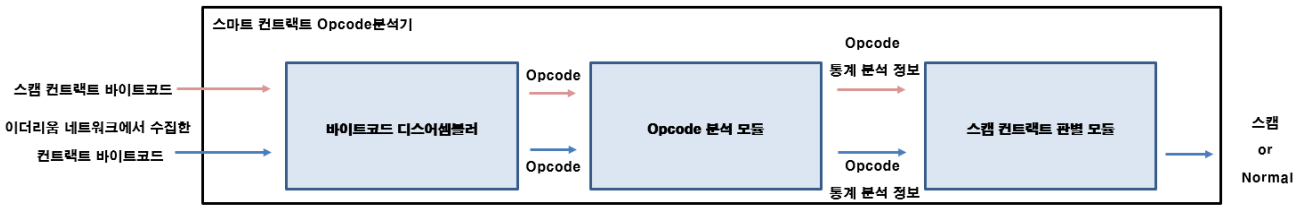


그림 1 스마트 컨트랙트 Opcode 분석기

비트코인과 이더리움의 구조적 차이에 따른 지갑 주소 클러스터링 방법 비교 및 분석

신혜영, 주홍택
계명대학교 컴퓨터공학과
yeoung@stu.kmu.ac.kr, juht@kmu.ac.kr

Comparison and analysis of wallet address clustering method due to differences between Bitcoin and Ethereum

Hye-yeong Shin, Hongtaek Ju
Dept. of Computer Engineering, Keimyung Universit

요 약

본 논문에서는 대표적인 블록체인 플랫폼인 비트코인과 이더리움의 지갑 주소 군집화 방식을 비교 분석한다. 비트코인은 UTXO를 기반으로 거래가 이루어지는 대표적인 블록체인 플랫폼이며, 이와 관련된 연구가 많이 진행되었다. 하지만, 어카운트 기반 거래가 이루어지는 블록체인의 경우 많은 연구가 진행되지 않았다. 불법적인 행위에 사용된 블록체인을 추적하기 위해서는 어카운트 기반 거래가 이루어지는 블록체인의 비익명화에 대한 연구 또한 진행되어야 한다. 해당 연구를 진행하기에 앞서, 본 논문에서는 이더리움의 지갑 주소 군집화와 관련된 연구를 조사한다.

Ⅰ. 서 론

블록체인(Blockchain) [1]은 중앙통제시스템 없이 탈중앙화된 P2P 네트워크로 운영된다. 블록체인 네트워크에서 발생한 거래(Transaction)는 블록에 저장되고, 블록은 합의 알고리즘(Consensus Algorithm)에 따라 유효성을 검증받는다. 검증된 블록은 그림 1과 같이 기존 블록체인에 연결된다.

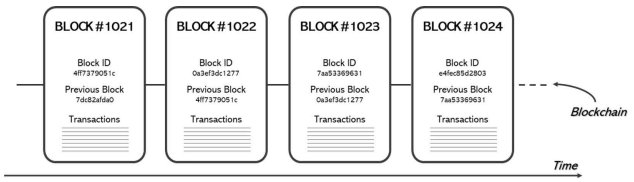


그림 1. 블록체인 기본 구조

블록체인은 사용자의 익명성(Anonymity)이 보장되는데, 이러한 점을 이용하여 악의적인 사용자가 암호화폐(Crypto-currency)를 합법적이지 않은 거래에 사용할 수 있는 문제가 있다 [2-4]. 불법 거래나 돈세탁 등 불법 거래에 참여한 당사자가 누구인지 알아내기 위해서는, 사용자에 대한 비익명화 작업이 진행되어야 한다. 하지만 이러한 사실 정보는 블록체인에서 발생한 정보만으로는 알아내는 것이 불가능하므로, 불법 거래가 주로 이루어지는 다크 웹(Dark Web) 등에서 추가로 정보를 수집해야 한다. 불법 거래와 관련된 정보를 수집한 뒤에는, 거래가 기록된 트랜잭션 데이터와 수집데이터를 이용하여 추적할 수 있다.

Reid, F. 등 [5]은 비트코인(Bitcoin) 트랜잭션 데이터를 최초로 분석하여, 지갑 주소를 군집화하는 작업을 진행했다. 군집화 작업을 통해 같은 지갑에서 관리될 것으로 추정되는 지갑 주소 그룹을 생성했다. Dorit, D. 등 [6]은 [5]에서 제시한 방법을 이용하여 관련 지갑 주소들을 군집화하는 작업을 진행하고, 지갑 주소의 거래 활동에 대해 분석했다. 그들은 분석 결과를 통해 가장 많은 BTC를 가진 주소가 무엇인지 찾을 수 있었다. 또한, 트랜잭션과 지갑 주소와의 관계를 시각적으로 표현하여 거래 흐름을 쉽게 알 수 있는 분석 결과를 도출했다. [5]와 [6]에서 사용된 방법은 비트코인 트랜잭션 정보를 이용하여 동일한 지갑에서 관리되는 지갑 주소를 추정하는 기초연구라 할 수 있다.

[5, 6] 연구를 기반으로, 해당 연구에서 제시한 방법과 SNS, 다크 웹 등에서 수집한 정보로 사용자 비익명화를 진행한 연구[7-13]들이 다양하게 진행되었다. 비트코인과 마찬가지로 UTXO(Unspent Transaction Output)를 기반으로 거래가 이루어지는 모네로(Monero)[13-18], Zcash[13, 19-23] 등에서도 비익명화와 관련된 많은 연구가 진행되었다.

UTXO 기반 거래가 이루어지는 블록체인의 사용자 비익명화와 관련된 연구는 많이 진행되었지만, 이더리움(Ethereun) [24]처럼 어카운트(Account)를 기반으로 거래가 이루어지는 블록체인의 경우 그렇지 않다. 불법적인 행위에 사용된 블록체인을 추적하고, 사용자를 밝혀내기 위해서는 UTXO 기반 거래뿐만 아니라 어카운트 기반 거래가 이루어지는 블록체인의 비익명화 작업이 진행되어야 한다.

따라서 본 논문은 이더리움 지갑 주소[1]를 비익명화하는 연구를 진행하기에 앞서, 이와 관련된 연구를 조사하고 어떤 방식으로 지갑 주소 군집화를 진행하였는지 알아본다. 이더리움과 비트코인의 차이점과 지갑 주소 군집화 방식의 차이점을 알아본다.

본 논문의 구성은 다음과 같다. 2장에서는 비트코인에 대한 기본적인 개념을, 3장에서는 이더리움에 대한 기본 개념을 설명한다. 또한, 4장에서는 비트코인과 이더리움의 차이에 대해 기술한다. 5장에서는 비트코인과 이더리움의 지갑 주소 군집화 방식에 대해 알아보고, 6장에서는 군집화 방식을 비교 분석한다. 마지막으로 7장에서는 조사한 군집화 방식을 이용한 향후 연구에 관해 논의하며 본 논문을 마친다.

1) 본 논문에서 언급하는, 이더리움의 지갑 주소는 지갑에서 관리되는 어카운트 주소를 의미한다. 비트코인 지갑 주소와의 용어 통일을 위해 사용하도록 한다. 비트코인에서의 지갑과 이더리움에서의 지갑은 비슷한 듯하지만 서로 다르므로, 동일시하는 것은 좋지 않다.

Ⅱ. 비트코인

비트코인은 블록체인 기술을 기반으로 만들어진 최초의 암호화폐 플랫폼이다 [1]. 비트코인은 P2P 네트워크에서 동작하며, 동등한 노드들로 구성되어 있다. 각 노드는 트랜잭션이 포함된 블록을 PoW 합의 알고리즘에 따라 유효성을 검증한다. 검증된 블록은 기존 블록체인에 연결되며, 블록을 생성한 채굴자는 채굴 보상(Mining Fee)을 받게 된다.

1. 블록

비트코인의 블록체인은 그림 2와 같이 연결된다.

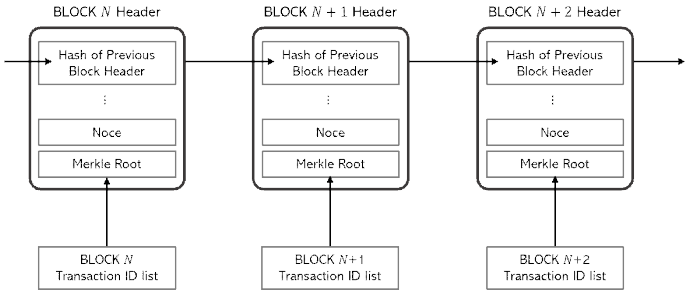


그림 2. 비트코인의 블록체인 기본 구조

비트코인 블록에는 블록헤더와 nTx(블록에 포함된 트랜잭션 개수), 트랜잭션 ID 리스트 정보가 기록된다. 블록헤더는 표 1과 같은 정보를 포함하며, 총 80바이트로 구성된다. 블록헤더는 블록의 메타데이터로 생각할 수 있다.

Name	Description	byte
Version	비트코인 소프트웨어 버전 정보	4
Hash of Previous Block Header	이전 블록헤더의 해시값 (이전 블록을 가리키는 해시 포인터)	32
Merkle root	머클 트리의 루트 노드 값 머클 트리는 블록에 포함된 트랜잭션 해시값으로 생성된 이진 트리	32
Timestamp	블록 생성 시간(UTC 표현)	4
Bits	블록헤더 해시값의 목푯값 또는 상한선	4
Nonce	블록헤더 해시값이 Bits보다 작아지게 하는 값	4

표 1. 비트코인 블록헤더 구조

제네시스 블록을 제외한 모든 블록은 이전 블록헤더의 해시값을 갖는다. 이 값은 이전 블록을 가리키는 해시 포인터의 역할을 한다. 블록 안에 있는 트랜잭션 값이 위조되면 헤더의 해시값이 변하게 되는데, 이 값으로 인해 블록 내 모든 기록은 변경할 수 없다.

머클 트리(Merkle Tree 또는 Merkle Trie)는 특정 트랜잭션이 블록에 포함되었는지 효율적으로 탐색할 수 있도록 고안된 자료구조다. 블록에 포함된 모든 트랜잭션의 해시값과 위트니스(Witness)[2] 해시값을 요약한 정보를 포함하고 있다. 머클 트리의 최상단 노드를 머클 루트라 하고, 이 값이 블록헤더에 기록된다. 머클 트리를 통해 트랜잭션을 신속하게 검증할 수 있으며, 이중 지불(Double Spending) 여부[3]를 확인할 수 있다.

2. 지갑

비트코인 지갑은 개인키(Private Key)와 공개키(Public Key) 그리고 UTXO를 관리한다. 비트코인 지갑은 논리적인 객체로 개인키와 공개키 쌍을 필요에 따라 생성하고 사용한다.

개인키는 타원곡선 암호 [25]로 공개키를 생성하고, 공개키 해시[4]를 계산하여 지갑 주소를 생성[5]한다. 이때 생성된 지갑 주소를 이용하여 트랜잭션에 사용한다. 개인키로부터 생성된 지갑 주소에서 발생한 UTXO는 하나의 트랜잭션에서 사용할 수 있다.

공개키 해시로 공개키를 역변환할 수 없으므로, 개인키를 잃어버리게 되는 경우 해당 지갑에서 관리되는 UTXO를 사용할 수 없다. 따라서, 지갑의 개인키 관리는 매우 중요하다.

3. 트랜잭션

트랜잭션은 UTXO에 기록된 BTC의 소유권을 이전한 기록이다. 통상적으로 비트코인 지갑이 관리하는 UTXO의 값을 합한 것이 잔액이라고 하지만, 비트코인에서는 잔액이라는 개념이 존재하지 않는다.

비트코인에서 발생한 모든 트랜잭션은 UTXO에 기록된 내용을 바탕으로 이루어지기 때문에, UTXO는 비트코인 트랜잭션을 구성하는 기본 단위로 생각할 수 있다. 트랜잭션을 발생시키고자 할 때는, 지갑 주소에서 관리되는 하나 이상의 UTXO를 사용할 수 있다[6]. 또한, 동일한 개인키로 관리되는 여러 개의 지갑 주소에서 관리는 UTXO를 사용할 수 있다.

Name		Description
txid		전자서명 값을 제외한 트랜잭션 ID
hash		전자서명 값을 포함한 트랜잭션 ID
size		전자서명 값을 제외한 트랜잭션 사이즈
vsize		전자서명 값을 포함한 트랜잭션 사이즈
weight		트랜잭션 weight
version		트랜잭션 버전
vin	txid	입력부의 UTXO
	vout	UTXO의 출력부(vout) index
	scriptsig	입력부 스크립트 내용(unlocking script) 전자서명과 공개키 정보 등
vout	value	BTC
	n	출력부 index
	scriptPub Key	출력부 스크립트 내용(locking script) 출금 주소, 공개키와 주소 형식 정보 등

표 2. 비트코인 트랜잭션 구조

트랜잭션 데이터는 트랜잭션 id, vin, vout 등으로 구성된다 (표2). 일반적으로 vin은 입력부, vout은 출력부라 한다. 입력부의 txid와 vout은 송금 주소가 사용하고자 하는 UTXO가 기록된 트랜잭션과 트랜잭션에서의 출력부 위치(vout의 n 필드)를 의미한다. 또한, 전자서명과 공개키는 비트코인에서 자신이 UTXO의 소유주임을 증명하고 다른 사

2) 위트니스는 블록에 포함된 트랜잭션의 전자서명 값으로, 트랜잭션에 hash로 저장된다. witness 트랜잭션 ID 형태로 저장

3) 블록체인 전체 데이터를 보유하지 않고, 블록헤더 정보만을 갖고있는 라이트 노드(SPV/Lightweight 노드)는 자신의 노드에서 발생한 트랜잭션의 유효성과 승인 여부만을 확인한다. 머클트리에서 트랜잭션 정보를 확인하는 경우, 이중 지불 없이 확정된 트랜잭션으로 간주할 수 있다.

4) 공개키 해시는 공개키에 SHA-256 해시 알고리즘을 두 번 적용한 뒤, RIPEMD 160 해시 알고리즘을 적용한 값

5) 지갑 주소는 공개키 해시값을 Base58Check로 인코딩한 값

6) 하나의 지갑 주소는 0개 또는 하나 이상의 UTXO를 관리한다. 여러 개의 UTXO를 갖고있는 주소 중 대표적으로 제네시스 블록 채굴 주소[26]가 있다.

용자들이 이를 검증할 때 사용된다.

출력부에는 출금 주소와 송금 주소로부터 받은 BTC이 기록되는데 이를 트랜잭션의 출력값이라 한다. 이때, 트랜잭션의 출력값이 다른 트랜잭션에서 사용되지 않았다면 이를 UTXO라 한다. 트랜잭션의 출력값은 어느 한 시점에 UTXO 상태로 지갑 주소에서 관리된다.

UTXO는 나눠지지 않는 하나의 덩어리이므로, UTXO에 기록된 출력값보다 작은 값의 BTC를 소비하게 되는 경우 나머지는 송금 주소로 다시 보내지거나, 다른 지갑 주소로 보내진다.

그림 3, 그림 4는 지갑 주소 A가 36BTC의 출력값을 가지는 UTXO에서 지갑 주소 B에게 30BTC를 전송하게 되는 경우 발생하는 거래 방법을 나타낸다.

그림 3은 지갑 주소 A가 B에게 30BTC를 전송한 후, 나머지 6BTC를 자신의 지갑 주소로 보내는 것이다. 그림 4는 그림 3과 달리 나머지 6BTC를 다른 지갑 주소 C로 보내는 것이다.

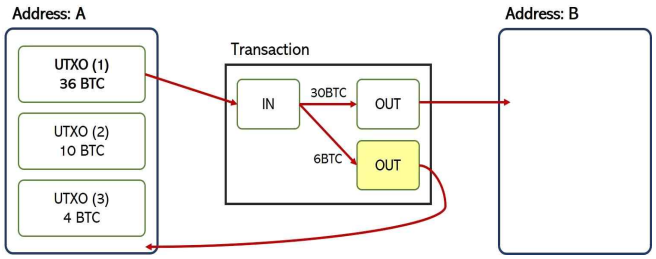


그림 3. UTXO를 이용한 거래 1

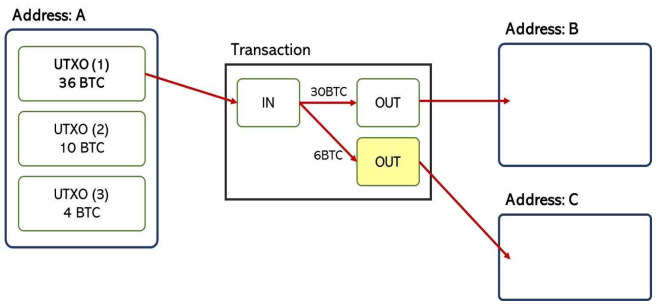


그림 4. UTXO를 이용한 거래 2

만약, 지갑 주소 A가 B에게 14BTC를 전송하고자 할 때는 UTXO(2)와 UTXO(3)를 사용하는 것이 가장 이상적인 방법이다.

Ⅲ. 이더리움

이더리움은 비탈린 부테릭(Vitalik Buterin)에 의해 구현되었으며, 블록체인 기술을 기반으로 스마트 컨트랙트 (Smart Contract) 기능을 지원하는 블록체인 애플리케이션 플랫폼이다 [24]. 이더리움에서는 암호화폐 거래뿐만 아니라, 서로 신뢰할 수 없는 대상 간에 계약을 진행할 수 있다.

이더리움은 P2P 네트워크에서 동작하며, 동등한 노드들로 구성된다. 각 노드들은 PoW 합의 알고리즘에 따라 블록을 생성하고 유효성을 검증한다. 검증된 블록은 기존 블록체인에 연결되며, 블록을 생성한 채굴자는 이더(ETH; Ether)를 채굴 보상으로 받게 된다.

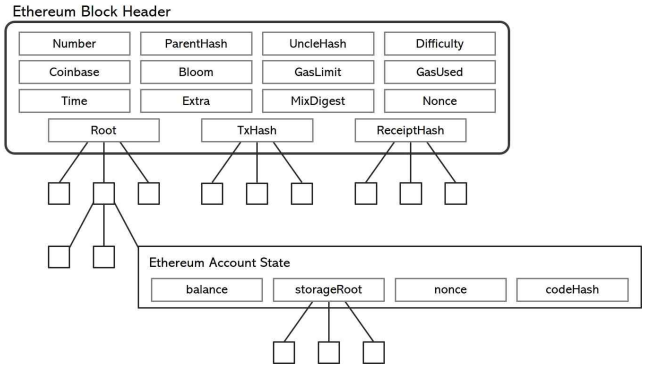


그림 5. 이더리움의 블록 기본 구조

1. 블록

이더리움의 블록에는 블록헤더와 엉클 블록(Uncle Block Hash List), 그리고 트랜잭션(트랜잭션 ID 리스트) 정보가 기록된다. 블록헤더는 그림 5와 같이 총 15개의 항목으로 구성된다.

이더리움 블록을 구성하고 있는 각 항목에 대한 설명은 표 3과 같다.

Name	Description
Number	현재 블록 번호
ParentHash	이전 블록 해시값
UncleHash	블록에 포함된 엉클 블록의 해시값
Coinbase	블록 채굴 후 이더를 지급받을 지갑 주소
Bloom	블록에 포함된 트랜잭션 로그 정보에 대한 블룸 필터
Difficulty	블록 생성 난이도
GasLimit	블록당 현재 지급 가능한 최대 가스 총량
GasUsed	현재 블록 내에 트랜잭션에 의해 사용된 가스[7] 총량
Time	블록 생성 시간(UTC 표현)
Extra	현재 블록과 관련된 기타 추가 정보
Nonce	블록 해시값을 찾기 위해 조정하는 값
MixDigest	Nonce 값의 해시값
Root	이더리움 전체 상태 정보가 저장되어있는 머클 패트리시아 트리의 루트 노드 해시값
TxHash	현재 블록 내에 모든 트랜잭션의 머클 트리의 루트 노드 해시값
ReceiptHash	현재 블록 내 모든 트랜잭션의 리시트[8]들의 머클 트리의 루트 노드 해시값

표 3. 이더리움 블록헤더 구조

블룸 필터(Bloom Filter)는 컨트랙트 어카운트(Contract Account)와 컨트랙트 트랜잭션이 실행될 때 생성된 모든 로그 정보를 인덱싱한 값이 저장되어있다. 블룸 필터로 트랜잭션 목록이나 해당 트랜잭션에서 생성된 로그들이 중복으로 저장되는 것을 방지할 수 있다.

2. 어카운트

어카운트는 이더리움 플랫폼에서 모든 트랜잭션의 실행 주체이자 기

7) 가스(Gas)는 이더리움 시스템의 운영 토큰이다. 이더리움에서 스마트 컨트랙트나 트랜잭션이 발생하였을 때, 마이너에게 트랜잭션 처리 비용을 가스로 지급해야 처리된다. 1Gas는 약 10^{-7}Ether이다. 비트코인에서의 트랜잭션 수수료 개념으로 생각할 수 있다.

8) 리시트(receipt)는 트랜잭션의 실행 과정 중 발생한 기록으로, 트랜잭션의 상태 정보(처리 결과)를 확인할 수 있다.

본 단위다. 모든 어카운트는 중복되지 않는 식별자로 주소를 부여받는데, 이 주소를 어카운트 주소 또는 지갑 주소라 한다.

어카운트의 잔액(balance), 트랜잭션 발생 횟수, 어카운트 타입 등의 정보를 상태(state)라 한다. 이더리움 전체 상태는 이더리움에 존재하는 모든 어카운트의 상태 정보를 의미하며, 머클 패트리시아 트리로 저장 및 관리된다.

이더리움에는 두 가지 타입의 어카운트가 존재한다.

A. 외부 소유 어카운트(EOA, Externally Owned Account)

EOA는 이더리움 플랫폼의 일반 사용자들이 사용하는 어카운트다. EOA는 개인키로 관리되며, 이를 사용하여 전자 서명된 트랜잭션을 생성할 수 있다. 만약, 개인키를 분실하게 되는 경우 EOA를 사용할 수 없다.

B. 컨트랙트 어카운트(CA, Contract Account)

CA는 스마트 컨트랙트를 블록체인에 배포할 때 생성되는 어카운트로, 컨트랙트를 가리키는 포인터 역할을 한다. 동일한 동작을 하는 컨트랙트가 존재하더라도, CA는 고유성을 지닌다.

이더리움 어카운트는 표 4와 같이 4개의 필드로 구성되어 있다.

Name	Description
Nonce	해당 어카운트로 보내진 트랜잭션의 수 (CA인 경우, 어카운트에 의해 생성된 트랜잭션의 수)
Balance	해당 어카운트의 이더 잔고(Wei 기준)
Root	해당 어카운트가 저장될 머클 패트리시아 트리의 루트 노드
CodeHash	해당 어카운트가 CA인 경우 바이트 코드의 해시값 (EOA인 경우, nil 값으로 표기)

표 4. 이더리움 어카운트 정보

3. 스마트 컨트랙트

스마트 컨트랙트는 특정 계약을 스스로 수립, 검증 및 이해하기 위한 프로토콜이다. 이더리움은 P2P 네트워크 상에 배포되어 블록체인 내에 상태 정보로 존재한다. 모든 사용자는 스마트 컨트랙트를 생성할 수 있으며, 다른 사용자가 생성한 스마트 컨트랙트를 사용할 수 있다. 한번 배포된 스마트 컨트랙트는 수정할 수 없다.

스마트 컨트랙트는 EVM((Ethereum Virtual Machine)[9]에서 작동되며 어카운트의 상태 전이를 유발한다.

4. 트랜잭션

이더리움에서는 EOA의 개인키로 전자 서명된 트랜잭션과 이더리움 네트워크에 스마트 컨트랙트가 배포될 때 생성되는 트랜잭션(Contract creation transaction)이 존재한다. 하지만, 본 논문에서는 EOA에 의해 생성되는 트랜잭션을 설명하도록 한다.

EOA에 의해 생성되는 트랜잭션은 지급(payment) 트랜잭션과 호출(invocation) 트랜잭션이 있다. 지급 트랜잭션은 하나의 EOA에서 다른 EOA로 ETH를 전송하는 것이고, 호출 트랜잭션은 EOA가 스마트 컨트랙트의 특정 함수를 호출/실행하는 것이다.

EOA에 의해 생성된 지급/호출 트랜잭션은 RLP(Recursive Lengh Prefix)로 인코딩되어, 암호 해시 알고리즘 Keccak-256를 통해 해시값으로 변환된다. 해시값은 ECDSA 알고리즘[10] [27]을 통해 EOA의 개인키로 전자 서명된다. 전자서명 된 트랜잭션은 표 5와 같은 트랜잭션 데이터 구조를 갖는다.

Name	Description
Nonce	송신 지갑 주소(EOA)에서 발생한 트랜잭션 개수
GasPrice	트랜잭션의 송신 지갑 주소가 지급하는 가스의 가격(wei)
GasLimit	트랜잭션에서 지급 가능한 가스의 최대 범위
Recipient	트랜잭션을 수신받는 주소 (CA인 경우, nil 값으로 표기)
Value	송신자가 수신자에게 전송할 ETH 양
Data	가변 길이 바이너리 데이터
v, r, s	송신 지갑 주소의 ECDSA 전자서명을 만드는 데 사용되는 값

표 5. 이더리움 트랜잭션 기본 구조

논스(nonce)는 송신 지갑 주소에 의해서 발생한 트랜잭션 개수로, 중복되지 않고 순차적으로 1씩 증가한다. 논스는 각 송신 지갑 주소에서 발생한 트랜잭션 건수에 대한 통계이며, getTransactionCount로 확인할 수 있다. 이더리움에서는 트랜잭션의 넌스 값으로 이중 지불 문제를 해결한다.

트랜잭션 데이터에서는 '송신 지갑 주소' 필드가 없다. 송신 지갑 주소는 v, r, s로 복구된 공개키로 계산할 수 있기 때문이다. r과 s는 개인키로 서명된 트랜잭션의 결과 값이다. r과 s값으로 2개의 공개키를 계산할 수 있으며, v로 공개키가 무엇인지 알 수 있다.

트랜잭션의 주요 페이로드(payload)는 value와 data 2개의 필드에 포함된다. 트랜잭션은 value와 data에 담긴 값에 따라 4가지로 분류할 수 있다. 표 6은 value와 data 값에 따른 트랜잭션 종류이다.

payload		Description
value	data	
○	×	지급 트랜잭션
×	○	호출 트랜잭션
○	○	컨트랙트의 잔액을 증가시키거나 ETH 전송을 거부할 수 있음
×	×	발생할 수는 있으나, 가스 낭비

표 6. 페이로드로 분류한 트랜잭션

EOA의 개인키로 전자서명 된 트랜잭션은 RLP로 인코딩되고, Keccak-256를 통해 해시값으로 변환된다. 이 해시값이 트랜잭션 ID다.

5. Internal Transaction

CA가 다른 CA의 컨트랙트에 호출 메시지를 보내는 경우, 이것을 Internal 트랜잭션이라 한다. Internal 트랜잭션은 다른 컨트랙트 함수를 사용하기 위해 호출하는 경우 주로 사용된다.

Internal 트랜잭션과 호출 트랜잭션은 전혀 다른 개념이다. 호출 트랜잭션은 EOA가 컨트랙트를 실행하기 위해 CA에게 보내는 전자 서명된 메시지이며, 블록체인에 기록된다. 하지만, Internal 트랜잭션은 이더리움 블록체인에 기록되지 않는다. Internal 트랜잭션 정보는 geth, paritty 등의 이더리움 클라이언트 API를 사용하여 확인할 수 있다.

Ⅳ. 비트코인과 이더리움 구조 비교

2장과 3장에서 알아본 비트코인과 이더리움의 기본 개념을 바탕으로 블록체인 플랫폼을 비교할 수 있다. 4장에서는 비트코인과 이더리움 지갑

9) 이더리움 스마트 컨트랙트를 실행하는 스택 기반의 실행 환경
10) 비대칭 암호화 알고리즘 - 256bit 타원형 곡선 방식

주소 군집화 방법을 알아보기 전에, 해당 연구에서 사용되는 핵심적인 개념들을 비교한다.

비트코인과 이더리움은 블록체인 기술을 기반으로 만들어진 암호화폐 플랫폼이란 공통점이 존재한다. 하지만, 이더리움은 비트코인과 달리 스마트 컨트랙트를 지원함으로써 좀 더 발전된 형태의 블록체인 플랫폼 서비스를 사용자에게 지원한다.

1. 지갑 주소

비트코인은 UTXO를 기반으로 거래가 이루어지며, 잔액이라는 개념이 존재하지 않기 때문에 지갑 주소에서 직관적으로 보유하고 있는 UTXOs의 총합을 알 수 없다. 개인키를 제외한 내용들이 트랜잭션에 기록되어 있기 때문이다. BTC를 전송하고자 하는 경우, 하나 이상의 입력 주소와 출력 주소를 기재할 수 있다. 단, 입력 주소는 동일한 개인키로 관리되는 지갑 주소여야만 한다. 이 방법을 통해 매 거래마다 새로운 지갑 주소를 사용할 수 있게 되는데, 이는 사용자의 익명성을 지키기 위해 사토시 나카모토가 착안한 방법이다.

반면에, 이더리움은 어카운트를 기반으로 거래가 이루어지며 지갑 주소 정보가 상태 정보로 저장된다. 상태 정보를 통해 직관적으로 지갑 주소의 잔액 및 다른 정보를 확인할 수 있다. 이더리움에서는 지갑 주소로 ETH를 전송하고자 하는 경우, 트랜잭션에 하나의 입력 주소와 출력 주소만을 기재 할 수 있다. 이것은 지갑 주소 재사용률을 높이기 때문에 사용자의 익명성이 지켜지지 않을 수 있다. 물론 스마트 컨트랙트를 통해 여러 개의 입력 주소 또는 출력 주소를 기재할 수 있지만, 일반적인 트랜잭션에서는 하나씩 기재가 가능하다.

2. 이중 지불

두 블록체인 플랫폼은 이중 지불 문제를 해결하는 방법 또한 다르다.

비트코인은 트랜잭션을 만든 사용자가 전자서명을 생성하고, 트랜잭션을 전파받은 사람은 전자서명을 검증한다. 트랜잭션 검증에는 생성된 트랜잭션의 ScriptSig와 UTXO의 ScriptPubKey가 사용된다.

풀 노드는 UTXO의 검증을 위해 UTXO set에 UTXO 정보를 검색한다. 해당하는 UTXO 정보를 찾지 못한 경우, 이는 이중 지불로 간주한다. 정보를 찾게 되는 경우 검증 과정을 거치게 되고, 블록에 포함되기를 기다리고 있는 상태가 된다. 블록에 트랜잭션이 포함되는 경우, 해당 UTXO 정보는 UTXO set에서 삭제된다.

이더리움은 트랜잭션의 논스 값을 이용하여 해결한다. 논스 값은 트랜잭션을 생성한 송신 지갑 주소에서 발생한 개수를 의미한다. 즉, 송신 주소에서 전송한 트랜잭션의 통계수치다. 트랜잭션의 검증을 위해 getTransactionCount를 이용한다. getTransactionCount의 결과 값과 현 트랜잭션의 논스 값의 차이가 1인 경우, 해당 트랜잭션은 유효하다. 하지만, 값의 차가 0 또는 음수의 값을 갖게 되는 경우 트랜잭션은 이중 지불로 간주된다. 이더리움에서는 논스의 값에 따라 순차적으로 트랜잭션이 처리 되는 것을 알 수 있다.

3. 트랜잭션

비트코인은 트랜잭션을 인코딩하는 과정이 없지만, 이더리움의 경우 총 2번의 인코딩 과정을 거친다. 서명 전 트랜잭션을 인코딩하고, 서명 후 추가로 인코딩 과정이 진행된다. 서명된 트랜잭션이 RLP로 인코딩되고 난 후, Keccak-256를 통해 해시값으로 변환되는데 이 해시값이 트랜잭션 id다.

이더리움은 비트코인과 달리, 트랜잭션에 포함되는 데이터와 트랜잭션 간의 관계가 비교적 간단하다는 것을 알 수 있다. 두 플랫폼 모두 송신 지갑 주소에 대한 정보가 없다. 이더리움은 v, r, s 필드로 공개키를 복구시켜 지갑 주소 정보를 알 수 있다. 하지만, 비트코인은 입력 주소의 트랜잭션 id와 index를 확인해야 주소를 확인할 수 있다.

V. 관련 연구

5장에서는 비트코인과 이더리움 지갑 주소 군집화 방식에 관한 관련 연구를 알아보도록 한다. 또한, 주로 트랜잭션 데이터를 통해 지갑 주소를 군집화하는 연구를 대상으로 조사를 진행했다.

1. 비트코인 지갑 주소 군집화

Reid, F. 등[5]은 비트코인 트랜잭션 입력부 데이터를 분석하여, 지갑 주소를 군집화하는 작업을 진행했다. [5]는 트랜잭션의 입력부에 기재 가능한 지갑 주소는 같은 개인키로 관리되는 것을 전제로 연구가 진행되었다. 비트코인 트랜잭션 데이터를 수집하여, 하나의 지갑에서 관리되는 주소를 추정했다. 또한, [5]에서는 사용자 익명성의 한계에 대해 언급하며 이를 보안할 필요가 있다고 주장했다.

Ron, D. 등[6]은 [5]에서 제시한 방법을 이용하여 같은 지갑에서 관리될 것으로 추정되는 특정 주소들의 모든 거래 활동에 대해 분석했다. [6]에서는 실험이 진행된 2012년까지 비트코인 네트워크에서 발생한 트랜잭션들의 통계적 특징에 대하여 설명했다. 하지만, 어떤 방식으로 통계적 특징을 추출했는지에 대한 내용은 언급되지 않았다. [6]은 한번도 다른 주소로 BTC를 보내지 않은 지갑 주소와 해당 주소들에서의 UTXO 총합(7,019,100BTC)을 추출하였고, 가장 많은 트랜잭션을 발생시킨 지갑 주소가 무엇인지 등에 관한 결과를 제시했다. [6]에서는 특정 방법을 사용하지 않고, 트랜잭션의 거래 흐름만을 추적하기 매우 어렵다고 언급했다.

[5, 6]에서 제시한 방법은 비트코인 지갑 주소 군집화 방법에 관한 기초적인 연구 방법이라 할 수 있다. 신혜영 등[10]은 [5, 6]에서 제시한 방법을 이용하여 사토시 나카모토의 재정 활동을 분석하는 연구를 진행하였다. 0번 블록부터 653,000블록에 포함된 트랜잭션 데이터 분석을 통해 사토시가 최소 1,011개의 지갑 주소와 20,143.438BTC를 보유했음을 추정했다.

[12]는 SNS, 웹 등에 개시된 비트코인 주소를 스크랩하고, 해당 주소들에서 발생한 트랜잭션을 Spagnuolo 등[33]에 의해 개발된 지갑 주소 클러스터링 도구 BitIodine을 통해 분석했다. BitIodine은 [5, 6]에서 사용한 방법에 잔돈 지갑 주소의 개념을 더해 개발된 지갑 주소 군집화 도구다. [12]는 BitIodine을 통해 지갑 주소 군집화를 진행한 후, 수집한 비트코인 주소 정보들로 비익명화 작업을 진행했다.

[5, 6]에서 사용한 방법 이외에도 지갑 주소와 관련된 트랜잭션 데이터의 특징을 추출한 연구 또한 진행되었다. Kanemura, K 등[8]은 불법 거래에 사용된 지갑 주소들과 트랜잭션들을 분석하고 DNM(Darnet Market) 주소를 식별하는 방법을 제안했다. [8]은 약 20만 개 이상의 비트코인 지갑 주소 웹 크롤링으로 수집했으며, 수집한 지갑 주소와 관련된 트랜잭션 분석을 통해 73개의 Features를 추출했다. 추출한 Features로 비트코인 지갑 주소를 식별/분류하기 위해 Supervised 분류기를 이용해 데이터를 학습시켰다. 또한 학습 결과를 통한 지갑 주소 분류를 진행하기 위해 voting-based system을 제안했다. [8]에서는 DNM 주소가 다른 사용자들 보다 더 높은 수수료를 지불했음을 확인했으며, 이것이 DNM 주소와 non-DNM 주소를 식별할 때 중요한 역할을 한다는 것을 발견했다. [8]은 제안한 Majoriyu voting 기반의 voting method를 통해 81%에 해당하는 분류 정확도를 확인했다. 분류

비정확도를 보인 19%에 대한 이유로는 coin-mixing service를 언급했다. 또한, 비트코인 네트워크에 존재하는 지갑 주소를 수집하는 방식을 개발할 필요가 있음을 주장했다.

2. 이더리움 지갑 주소 군집화

Klusman 등[29]은 이더리움 비익명화에 대한 연구를 진행하기 위해, 비트코인에서의 비익명화 기술 [12, 32, 33]을 모두 적용하려고 시도했다. [32]는 트랜잭션을 처음 전파하는 노드가, 트랜잭션을 생성한 사람일 것이라는 전재로 연구를 진행했다. 비트코인 네트워크에 존재하는 모든 노드에 연결을 시도하여, IP 주소와 지갑 주소를 연결하여 비트코인 클라이언트를 비익명화하는 연구를 진행했다. [32] 연구를 진행하기 위해서는 고정된 노드 정보가 필요하다. 하지만 이더리움의 경우 노드와 연결된 이웃 노드가 주기적으로 변경되기 때문에 해당 방법으로 연구를 진행할 수 없다. 또한, [32] 연구를 진행하기 위해서는 모든 노드에 연결해야 하므로 자원이 많이 소모되며 네트워크의 크기가 증가하는 경우 대입하기 힘들다는 단점 또한 존재한다. 즉, P2P 운영 방식과 자원 소모가 많이 되기 때문에 진행하지 못한다는 것을 발견했다.

또한, [12]와 같이 SNS, 웹 등에 있는 주소 정보를 수집하여 트랜잭션을 분석하고자 했으나 트랜잭션 구조 차이로 인해 진행하지 못했다. [29]는 비트코인에서 적용된 비익명화 기술이 P2P 운영 방식과, 트랜잭션 구성의 차이로 인해 이더리움에 적용되지 않음을 확인했다.

Linoy 등[30]은 Stylemetry 방식을 사용하여, 스마트 컨트랙트의 작성자와 CA 주소를 비익명화 하는 연구를 진행하였다. 해당 연구는 이더리움에서 불법적인 거래와 관련된 스마트 컨트랙트를 식별하는 데 사용된다. 하지만, 해당 연구에서는 EOA와 관련된 연구가 진행되지 않았다.

Béres 등[31]은 ENS(Ethereum Name Service)에 기록된 지갑 주소 정보와 트위터, 다크넷 등에 기록되어 있는 이더리움 지갑 주소 및 사용자 정보를 수집하였다. [31]은 이더리움의 지급/호출 트랜잭션 데이터를 바탕으로 graph representation 학습을 진행했다. 뿐만 아니라, 지갑 주소의 일일 활동 및 트랜잭션 수수료 등으로 지갑 주소 특징을 추출했다. [31]은 graph representation 학습 결과와 추출된 데이터를 바탕으로 동일한 사용자가 소유한 지갑 주소를 군집화하는 방법을 연구했다. 또한, Torando Cash 믹서와 같은 Coin-mixing service로 이더리움 사용자의 익명성이 강화되었음을 강조했다.

Ⅵ. 지갑 주소 군집화 방법 비교 및 분석

6장에서는 5장 관련 연구에서 조사한 연구 방식을 비교 및 분석한다.

비트코인은 트랜잭션에 저장되어있는 데이터를 토대로 지갑 주소 군집화 연구들이 많이 진행되었다. 트랜잭션 데이터뿐만 아니라 웹에서 수집한 데이터들로 군집화가 이루어진 주소들의 사용자를 비익명화할 수 있었다 [8].

하지만, 이더리움은 비트코인과는 달리 트랜잭션과 어카운트에 기록된 데이터들을 토대로 지갑 주소를 군집화하는 것은 힘들다. 비트코인의 경우, 동일한 개인키로 관리되는 지갑 주소들이 트랜잭션에 기재될 수 있다. 하지만, 이더리움의 경우 하나의 트랜잭션에는 하나의 송신 주소만을 기재할 수 있기 때문에 동일한 방법으로 진행하기 힘들다. 이런 이유로 이더리움에서 지갑 주소 군집화를 위해 [31]에서는 지갑 주소와 트랜잭션 데이터 특징을 분석 및 추출하는 연구가 진행됐다.

Ⅶ. 결론 및 향후 연구

블록체인의 사용자의 비익명화는 불법 거래에 참여한 지갑 주소 사용자들을 추적하기 위해 굉장히 중요한 연구라 할 수 있다.

비트코인과 같이 UTXO를 기반으로 거래가 이루어지는 블록체인의 사용자 비익명화에 대한 연구는 많이 진행되어 왔다. 비트코인의 경우, 트랜잭션에 기록된 데이터를 통해 지갑 주소를 군집화하는 연구를 진행할 수 있었다. 또한, 다크넷 등에서 수집한 정보들로 특징을 추출하여 해당 지갑 주소의 DNM 여부를 확인하는 연구가 진행되었다. 더 나아가, 트랜잭션 데이터와 불법 거래에 참여한 지갑 주소 데이터들의 특징을 추출하여 불법적인 행위에 사용된 지갑 주소를 추적하는 연구를 진행할 수 있을 것이라 기대한다.

하지만, 어카운트 기반 거래가 이루어지는 블록체인의 경우 많은 연구가 진행되지 않았다. 이더리움의 기본 개념과 [27]을 통해 트랜잭션에 기록된 데이터를 통해 지갑 주소를 군집화하는 연구를 진행하기 힘들다는 것을 확인할 수 있었다.

향후 연구에서는 지갑 주소별 특징을 추출하고 분석하여 지갑 주소를 군집화하고 사용자의 비익명화를 진행하는 연구를 통해 블록체인 불법 거래의 영역을 확장 시킬 수 있을 것이라 기대한다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2018R1D1A1B07050380).

참 고 문 헌

[1] Nakamoto, Satoshi. "*Bitcoin: A peer-to-peer electronic cash system,*" 2008.

[2] Foley, S., Karlsen, J. R., Putniņš, T. J. "*Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies?,*" The Review of Financial Studies, Vol. 32, No. 5, pp. 1798-1853, May-April, 2019.

[3] Paquet-Clouston, M., Haslhofer, B., Dupont, B. "*Ransomware payments in the bitcoin ecosystem,*" Journal of Cybersecurity, Vol. 5, No. 1, 2019.

[4] N. Christin, "*Traveling the silk road: A measurement analysis of a large anonymous online marketplace,*" in Proceedings of ACM International Conference on World Wide Web, pp. 213 - 224, 2013.

[5] Reid, F., Harrigan, M. "*An analysis of anonymity in the bitcoin system,*" Security and privacy in social networks, Springer, pp. 197-223, 2012.

[6] Ron, D., Shamir, A. "*Quantitative analysis of the full bitcoin transaction graph,*" International Conference on Financial Cryptography and Data Security, Springer, pp. 6-24, April, 2013.

[7] Meiklejohn, S., Pomarole, M., Jordan, G., et al. "*A fistful of bitcoins: characterizing payments among men with no names,*" In Proceedings of the 2013 conference on Internet measurement conference, pp. 127 - 140, October, 2013.

[8] Kanemura, K., Toyoda, K., Ohtsuki, T. "*Identification of darknet markets' bitcoin addresses by voting per-address classification results,*" 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, pp. 154-158, May, 2019.

[9] 정세진, 곽노현, 강병훈, "*불법 커뮤니티를 통한 비트코인 거래 추적 방법에 관한 연구,*" 정보보호학회논문지, Vol. 28, No. 3, pp. 717-727, 2018.

[10] 신혜영, 주홍택, "*해시레이트와 트랜잭션 분석을 통한 나카모토 사토시 비트코인 규모 추정 연구,*" 2020 통신망 운용관리 학술대회 논문집, pp. 47-55, 5월, 2020.

[11] 신혜영, 주홍택, "*비트코인 트랜잭션 데이터 분석을 통한 거래 추적 및 분석 연구,*" 석사학위논문, 계명대학교, 대구, 2021.

[12] Al Jawaheri, H., Al Sabah, M., Boshmaf, Y., et al., "*Deanonymizing tor hidden service users through bitcoin transactions analysis,*" Computers & Security, 89, 101684, 2020

[13] Biryukov, A., Tikhomirov, S., "*Security and privacy of mobile wallet users in bitcoin, dash, monero, and zcash,*" Pervasive and Mobile Computing, 59:101030, 2019.

[14] Alonso, Kurt M., "*Zero to monero,*" 2020.

[15] Monero, https://www.getmonero.org/, cited 2021 March. 27.

[16] Chervinkski, Joao Otávio Massari, Kreutz, D., "*Floodxmr: Low-cost transaction flooding attack with monero's bulletproof protocol,*" IACR Cryptology ePrint Archive, 2019:455, 2019.

[17] Möser, M., Soska, K., Heilman, E., et al. "*An empirical analysis of traceability in the monero blockchain,*" arXiv preprint arXiv:1704.04299, 2017.

[18] Li, Y., Yang, G., Susilo, W., et al. "*Traceable monero: Anonymous cryptocurrency with enhanced accountability,*" IEEE Transactions on Dependable and Secure Computing, 2019.

[19] Zcash, https://z.cash/technology/, cited 2021 March. 27.

[20] Biryukov, A., Feher, D., "Deanonymization of hidden transactions in zcash," University of Luxembourg, 2018.

[21] Biryukov, A., Feher, D., Vitto, G., "*Privacy aspects and subliminal channels in zcash,*" In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1813-1830, 2019.

[22] Tramer, F., Boneh, D., Paterson, K. G., "*Ping and reject: The impact of side-channels on zcash privacy,*" 2019.

[23] Zhang, Z., Li, W., Liu, H., et al. "A refined analysis of zcash anonymity," IEEE Access 8: 31845-31853, 2020.

[24] Buterin, V., "*Ethereum: A next-generation smart contract and decentralized application platform,*" URL https://github.com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper, 7, 2014.

[25] "SEC2 ver2", http://www.secg.org/sec2-v2.pdf, cited 2021 March. 29.

[26] "Nakamoto Satoshi's Bitcoin Wallet address", https://www.blockchain.com/btc/address/1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa, cited 2021 Marh. 31.

[27] "ECDSA Algorithms", http://bit.ly/2r0HhGB, cited 2021 Marh. 31.

[28] Sompolinsky, Y., Zohar, A., "Accelerating bitcoin's transaction processing," fast money grows on trees, not chains, 2013. URL https://eprint.iacr.org/, 2013.

[29] Klusman, R., Dijkhuizen, T., "Deanonymisation in ethereum using existing methods for bitcoin," 2018.

[30] Linoy, S., Stakhanova, N., Matyukhina, A., "*Exploring Ethereum's Blockchain Anonymity Using Smart Contract Code Attribution,*" In 2019 15th International Conference on Network and Service Management (CNSM), IEEE, pp. 1-9, 2019

[31] Béres, F., Seres, I. A., Benczúr, A., et al. "*Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users,*" arXiv preprint arXiv:2005.14051, 2020.

[32] Biryukov, A., Khovratovich, D., Pustogarov, I., "*Deanonymisation of clients in Bitcoin P2P network,*" Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 15-29, 2014.

[33] Spagnuolo, M., Maggi, F., Zanero, S. "*Bitiodine: Extracting intelligence from the bitcoin network,*" International conference on financial cryptography and data security. pp. 457-468, Springer, Berlin, Heidelberg, 2014.

이더리움 네트워크에서 액티브 프루빙을 이용한 정확도 높은 토폴로지 탐색 방법

맹수훈*, 주홍택

*계명대학교

*shmaeng@kmu.kr, juht@kmu.ac.kr

A High-Accuracy Dynamic Topology Inference Method Using Active Probing in Ethereum Network

SooHoon Maeng*, Hongteak Ju

*Keimyung Univ.

요 약

이더리움 네트워크는 P2P 네트워크 중에 잘 알려진 카뎀리아(Kademlia)를 사용하여 P2P 네트워크에 참여하고 새로운 노드를 탐색하며 이웃 노드와의 연결을 변경한다. 이더리움 네트워크는 DDoS공격, 51%공격, 시빌 공격과 같은 보안 공격에 대처해야 하고 네트워크가 확장됨에 따라 초당 거래 처리 속도(TPS)가 느려지는 확장성(Scalability)문제를 해결해야 한다. 이러한 문제 해결 방안은 동적으로 변화하는 토폴로지 분석과 토폴로지를 구성하고 있는 노드 간 연결에 대한 기초 연구를 바탕으로 도출되어야 한다. 따라서 본 논문에서는 노드 탐색과정을 반복하여 패시브 프루빙 데이터를 수집하고 액티브 프루빙 방법을 사용하여 이더리움 네트워크에 참여하고 있는 노드의 활동 유무를 확인한다. 또한 토폴로지에 대한 분석 및 시각화를 통해 이더리움 네트워크에 대한 분석 자료를 제공한다.

Ⅰ. 서 론

이더리움(Ethereum)은 비탈릭 부테릭(Vitalik Buterin)에 의해 튜링 완전(Turing Completeness)언어를 접목시켜 여러 가지 탈중앙화 어플리케이션(Decentralized Application)구현이 가능한 블록체인 플랫폼이다[1]. 이더리움 네트워크는 P2P 네트워크 중에 잘 알려진 카뎀리아(Kademlia)[2]를 사용하여 P2P 네트워크에 참여하고 새로운 노드를 탐색하며 이웃 노드와의 연결을 변경한다.

이더리움 네트워크는 DDoS공격, 51%공격, 시빌 공격과 같은 보안 공격[3, 4]에 대처해야 하고 네트워크가 확장됨에 따라 초당 거래 처리 속도(TPS)가 느려지는 확장성(Scalability)문제를 해결해야 한다. 이러한 문제 해결 방안은 동적으로 변화하는 토폴로지 분석과 토폴로지를 구성하고 있는 노드 간 연결에 대한 기초 연구를 바탕으로 도출되어야 한다. 이더리움 네트워크에서 노드 탐색을 기반으로 토폴로지를 분석하기 위한 연구[5]와 피어의 속성을 분석하는 연구[6]가 있었으나 연결 정보를 패시브 프루빙(Passive Probing)으로 수집한 후에 IP를 기반으로 피어 간 연결을 기반으로 액티브 프루빙(Active Probing)과정을 수행하여 이더리움 네트워크의 토폴로지 분석에 대한 정확도를 높인 연구는 없었다.

본 논문에서는 이더리움 네트워크에 참여하고 있는 노드를 카뎀리아 방식을 활용하여 탐색하고 동적으로 변화하는 토폴로지를 확인하는 방법을 제안한다. 본 논문에서는 이더리움 네트워크의 토폴로지를 측정하기 위해 카뎀리아를 기반으로 노드를 탐색하고 패시브 프루빙 데이터를 수집한다. 이더리움 카뎀리아를 활용하여 탐색한 노드는 실제로 동작하지 않는 이웃 피어의 정보가 포함될 수 있다. 이는 블록체인 네트워크의 천(Chrun)현상[7]과 사용자들의 네트워크 참여 의사에 따라 변경될 수 있다.

수집된 패시브 프루빙 데이터는 네트워크 스캐너과정을 통해 액티브 프루빙 과정을 수행한다. 액티브 프루빙은 토폴로지는 최소한의 시간 동안 변경되지 않은 상태로 유지된다고 가정한다. 본 연구에서 토폴로지를 측정하는 최소한의 시간은 네트워크 스캐너가 동작하는 시간이다. 이를 통해 이더리움 네트워크에서 노드 활성화 유무를 통해 파악된 토폴로지는 그래프 이론에 입각하여 특징을 분석하고 이더리움 네트워크의 토폴로지 측정에 대한 정확도를 높인다. 본 논문의 결과는 향후 블록체인 토폴로지에서 영향력 있는 피어 선정을 통한 네트워크 성능 향상, 중복 메시지 제거에 대한 기초 연구와 이더리움 네트워크 토폴로지가 변경되는 최소한의 시간에 대한 연구의 기초 자료로 활용될 수 있다.

본 논문의 2장은 기존 비트코인 토폴로지 분석 연구를 제시한다. 3장은 이더리움 네트워크에서 동적 토폴로지 탐색 방법에 대해 설명한다. 4장에서 실험에 대한 분석 결과를 제시하고 5장은 본 논문의 결론을 도출한다.

Ⅱ. 관련 연구

2.1 비트코인 토폴로지 분석

M. Grundmann [8]은 비트코인 네트워크에 참여하고 있는 노드의 이웃 노드를 트랙잭션을 생성해서 보내고 수신함으로써 탐색하는 방법을 제안했다. 특정 노드의 이웃 노드를 확인하는 방법으로 각 노드마다 다른 트랜잭션을 보내거나 이중 지불 트랜잭션을 보내는 방법으로 탐색했고 거래 수수료문제로 인해 메인넷에 적용은 어렵다는 결론을 도출 했다.

Andrew Miller [9]는 어드레스 프로브(AddressProve)을 통해 비트코인 네트워크에 참여한 노드들 간의 링크를 발견하고 동적 토폴로지에 적용하는 방법을 제안했다. 그 결과로 비트코인 네트워크에서 노드 간 서로 연결된 커뮤니티가 존재를 확인하고 측정된 동적 토폴로지의 결과를 도출 했

다.

Essaid Meryam [10]는 비트코인 네트워크에서 활동하고 있는 노드를 재귀적으로 스캔하여 노트를 탐색하는 노드프로브(NodeProbe)를 도입하는 방법을 제시 했다. 이를 통해 비트코인의 시간 변화에 따라 활동 노드 수 , 영구 노드 등의 변화를 확인하고 노드 수가 같은 랜덤 네트워크 모델보다 커뮤니티 수가 4배 이상 많고 특정 노드가 비트코인 네트워크의 백본 역할을 분석을 통해 확인했다.

비트코인 토폴로지 분석은 이중 지불 트랜잭션, 어드레스 프로브(AddressProve), 노드프로브(NodeProbe)를 이용하여 동적 토폴로지에 대한 분석이 수행되었다. 이더리움 네트워크에서는 연결차수에 의해 영향력을 확인하는 연구[14]는 노드 간 연결을 확인할 수 있지만 피어가 제공한 IP는 피어의 버켓 정보로 특정 시간 네트워크 참여 유무를 확인할 수 없다. 또한 NodFinder[15]에서는 이더리움 노드 정보 제공 웹[16]보다 약 3배 많은 노드를 발견했지만 노드 간 연결 관계를 확인 할 수 없기 때문에 전체 토폴로지를 시각화하지 못했다. 이처럼 이더리움 네트워크에서 정확도 높은 토폴로지에 대한 분석 및 시각화 연구는 수행되지 않았다.

Ⅲ. 동적 토폴로지 탐색 방법 및 분석 방법

3.1 동적 토폴로지 탐색 방법

본 연구는 이더리움 네트워크에서 동적으로 변화하는 노드를 탐색하기 위해 전처리 과정과 분석엔진 2가지 과정을 수행한다. 또한 대용량 데이터를 수집 및 분석하기 위해 4개의 데이터베이스를 사용하며 그림 1과 같다. 전처리 기능(PreProcess)은 Peer Controller, Data Collector로 나누어지고 Peer Controller과정은 많은 피어를 수집하기 위해 탐지 노드(Go-ethereum Client)와 연결된 피어를 버켓에서 삭제하며 피어 정보를 수집한다. 탐지 노드는 FIND_NODE을 통해 응답 받은 이웃 피어 정보를 수집하기 위해 이더리움 클라이언트의 로그파일에 저장하는 기능을 추가하고 수집된 정보는 피어와 이웃 피어의 연결성을 함께 로그 파일(Geth.log)에 저장한다. 또한 Data Collector과정은 로그파일 정보를 파싱(Parsing)하여 피어 IP, 피어 Port, 이웃 피어 IP, 이웃피어 Port, 타임스탬프정보를 Raw Data 데이터베이스에 저장한다. 분석엔진의 Duplication Filter과정은 Raw Data 데이터베이스에 저장된 피어의 IP, Port를 중복 제거하고 피어와 이웃 피어로 구성된 연결 정보에 대한 중복도 데이터를 제거한 후 Duplication Filter Data 데이터베이스에 저장한다. Duplication Filter Data 데이터베이스는 피어와 이웃 피어의 IP, Port의 중복 값을 제거하고 피어 IP와 Port값이 쌍으로 저장되며 이때의 피어와 이웃 피어의 연결성은 배제되어도 된다. Duplication Filter를 수행 후 TCP/IP Ping 액티브 프루빙으로 네트워크에서 해당 노드가 연결되어 동작하고 있는지 조사하고 그 결과를 Active Node 데이터베이스에 저장한다. 이 과정이 기존 연구에 추가된 액티브 프루빙 과정이다. Network Scanner는 TCP/IP Ping 패킷을 보내며 실험 속도를 높이기 위해 50ms 간격의 쓰레드(Thread)로 구현되었다. 활동 중인 노드를 정확하게 측정하기 위해 2번의 시도로 수행한다. 또한 이더리움 네트워크에서 카뎀리아의 노드 탐색과정에서 'Hello'메시지는 500ms의 대기시간을 가지므로 Network Scanner 모듈은 응답 대기 최대 시간은 1초로 설정해 충분한 응답 대기 시간을 가지도록 설계했다[13]. Network Scanner는 파이썬 기반의 Network Socket에서 생성 함수 AF_INET, SOCK_STREAM를 사용하여 신뢰성을 높였다.

로그 파일로 수집된 패시브 프루빙 정보와 Ping 프로토콜로 확인한 액티브 프루빙 정보의 정확도 향상 정도를 분석하기 위해 Raw Data 데이터베이스와 Active Node 데이터베이스의 결과를 서로 비교하는 분석을 Compare Connection 모듈이 수행한다. 동적 토폴로지는 Network Scanner 모듈의 결과로 활동 중인 노드의 IP, Port와 Raw Data의 피어와 이웃 피어 연결을 비교하여 동적으로 변화하는 이더리움 네트워크의 토폴로지를 확인한다. Dynamic Topology 데이터베이스에는 Network Scanner와 Raw Data을 비교 하는 알고리즘은 단순하게 선형 탐색(Linear Search) 방법을 사용하여 각 요소를 순차적으로 확인한다.

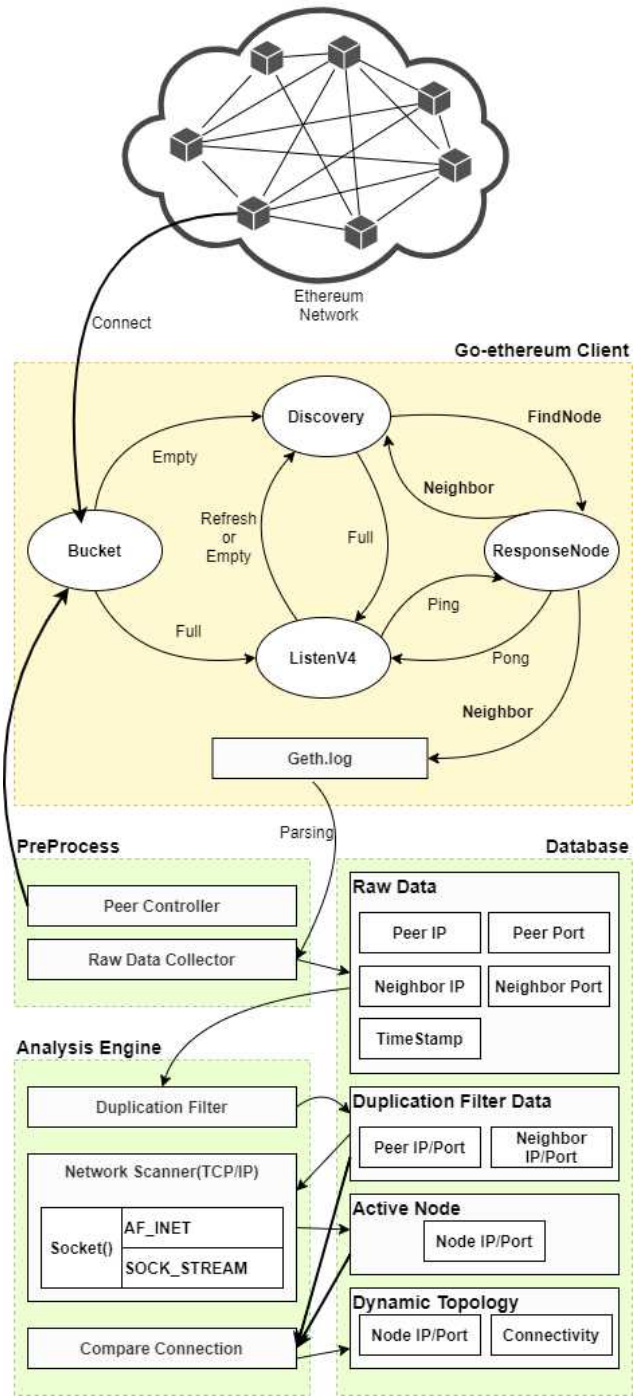


그림 1. 동적 토폴로지 탐색 및 분석 시스템 구성

3.2 동적 토폴로지 분석 방법

본 논문에서는 동적 토폴로지를 분석하기 위해 평균 연결 차수, 평균 경

로 길이, 직경, 평균 클러스터링 계수, 근접 중심성, 매개 중심성으로 나누어 표 2와 같은 수식을 이용하여 분석을 진행했다.

평균 연결 차수는 임의의 노드와 이웃하는 노드들 사이에 연결된 선의 수를 평균을 의미하며 평균 경로 길이는 토폴로지 내 임의의 두 노드가 연결되어 있을 때 반복하지 않는 선들의 연결의 평균 길이를 의미한다. 직경은 토폴로지 내의 최대 거리를 의미하며 토폴로지 크기와 같은 의미이다. 클러스터링 계수는 특정 노드와 연결된 임의의 두 노드가 서로 연결되어 있을 확률을 의미하며 임의의 네트워크에서 군집이 존재하지 않으면 평균 클러스터링 계수 값은 0에 가깝다. 근접 중심성(Closeness Centrality)은 전체 네트워크에서 임의의 노드가 중앙에 위치하고 있는지를 확인한다. 또한 근접 중심성은 중요한 노드일수록 다른 노드까지 도달하는 경로가 짧은 것을 의미하며 작은 값일수록 중심에 가깝다. 매개 중심성(Between Centrality)은 임의의 두 노드사이의 가장 짧은 경로가 노드를 거쳐가는 노드 수를 의미하고 매개 중심성이 높은 노드는 다른 노드들 사이에서 중간 역할을 하며 네트워크에 큰 영향력주고 있다.

구분	수식
Average Degree	$D_{Avg}\frac{1}{N}\sum_{1}^{N}K_i$
Average Path Length	$APL_{Avg}=\frac{\sum_{i,j}^{N}Dis(n_i,n_j)}{\sum_{i,j}^{N}Path(n_i,n_j)}$
Diameter	$D=\max_{i,j}(Path(i,j))$
Average Clustering Coefficient	(1) $C_i=k_i\frac{2T_i}{(k_i-1)}$ (2) $C_{Avg}=\frac{1}{N}\sum_{N}^{i}C_i$
Closeness Centrality($i\neq j$)	$C_c(N_i)=\frac{1}{[\sum_{j=1}^{g}=d(N_i,N_j)]},i\neq j$
Betweenness Centrality	$C_B(N_i)=\sum_{j<k}\frac{g_{jk}(N_i)}{g_{jk}}$

표 1. 동적 토폴로지 분석 내용

Ⅳ. 동적 토폴로지 분석 결과

4.1 실험 환경

본 연구에 사용된 탐지 노드는 많이 사용이 되는 이더리움 클라이언트(Ethereum Client)[12, 14]탐지 노드가 설치된 컴퓨터는 Intel Core i7-6700 CPU@3.40GHZ 8코어, 1TB SSD, 16GB의 RAM을 사용하며 운영체제는 Linux 18.04 버전이며 이더리움 네트워크 노드 탐색을 위한 클라이언트는 Go-ethereum은 1.9.12 버전을 사용하고 Go-Language는 1.13.6을 설치했다. 탐지 노드는 일반 인터넷 서비스 제공자(ISP)에 연결되고 접속 속도는 약 20Mbps이며 파이어월과 같은 보안 장비가 없이 인터넷에 연결되어 있다. 패시브 프루빙 데이터 수집 기간은 2020년 12월부터 2021년 2월까지 2개월이고 액티브 프루빙과정을 통한 토폴로지 조사는 2021년 2월 28일에 약 30분 동안 수행했다. 이더리움 네트워크는 노드들의 참여와 탈퇴를 반복하고 새로운 노드가 추가되기 때문에 패시브 프루빙 데이터는 수집 기간이 필요하다. 패시브 프루빙 데이터 수집 기간은 실험 환경으로 인해 2개월로 설정되었지만 기간을 확장하면 정확도 높은 토폴로지를 측정할 수 있다.

4.2 이더리움 네트워크 노드 탐색 결과

본 연구는 이더리움 네트워크에서 참여하는 클라이언트 전체 로직에 영향을 주지 않기 위해 FIND_NODE의 응답을 로그 파일에 저장하였고 2개월간 수집된 데이터는 131,776개의 피어와 각 피어에 연결된 8,643,225개의 이웃 피어 연결이다. Duplication Filter의 과정에서 중복을 제거하여 총 131,776개의 노드 IP가 만들어졌고 이 숫자는 2개월 수집 기간 동안 전체 이더리움 네트워크에 한번이라도 참여한 노드들의 수 이다. 또한 피어와 이웃 피어의 연결도 중복을 제거하여 총 1,464,853개의 연결이 수집되었다. Network Scanner과정은 Duplication Filter과정에서 수집된 데이터를 이용하여 약 30분 동안 수행했고 최종적으로 13,261개의 노드가 동작중인 것이 확인되었다. 이 결과는 상용화된 이더리움 노드 추적 웹에서 발견된 노드의 1.5배 이상이다. 또한 동작 중인 피어와 동작 중인 이웃 피어의 연결을 Network Scanner와 Raw Data를 선형 탐색한 결과 총 353,606개의 연결을 확인했다.

4.3 이더리움 동적 토폴로지 분석 결과

동작 중인 노드들 간의 연결된 총 353,606개의 연결로 분석된 결과는 표 2와 같다. 노드들의 평균 차수는 27.583이고 이는 피어가 이웃 피어의 평균 연결 수를 의미한다. 평균 경로 길이는 각각의 노드가 제 2의 노드에게 도달하는 최소한의 경로의 평균으로 3.694이고 이는 노드 간 평균적으로 이 만큼의 거리를 두는 것을 의미한다. 이더리움 네트워크의 직경은 8로 관측되었으며 노드 간 밀도를 나타내는 결집계수는 0.478의 수치를 보여준다. 근접 중심도(Closeness Centrality)와 매개 중심도(Betweenness Centrality)의 평균은 표 2와 같다.

구분	결과 값
Average Degree	27.583
Average Path Length	3.694
Diameter	8
Average Clustering Coefficient	0.478
Closeness Centrality	0.19
Betweenness Centrality	0.001

표 2. 동적 토폴로지 분석 결과

이더리움 네트워크에서 활동 중인 노드를 단순히 연결 차수에 따라 헤비 노드(50개 이상 연결), 미들 노드(2 ~ 50개 연결), 라이트 노드(1개 연결)로 분류하여 4,860개의 헤비노드, 7,613개의 미들 노드, 788개의 라이트 노드를 발견했다. 이더리움 네트워크에 연결 가능한 피어의 기본 값은 50이며 단순히 이 연결 개수에 의해 헤비, 미들, 라이트 노드를 분류한다. 또한 근접 중심성과 매개 중심성은 35.xx.xx.xx가 0.484, 0.087로 가장 높은 결과를 보였고 특정 노드의 정보보호를 위해 IP주소는 8비트만 공개한다. 본 연구를 통해 연결 차수가 가장 높은(1,146개)노드는 부트스트랩 노드가 아닌 아마존 ISP를 사용하고 있다.

Compare Connection단계의 결과인 353,606개의 연결성을 바탕으로 이더리움 네트워크의 토폴로지를 그림 2와 같이 시각화했다. 실험 결과로 이더리움 네트워크는 노드 연결 차수가 높은 노드를 발견했고 노드들은 약 9개의 커뮤니티를 구성하여 이더리움 네트워크에 참여하고 있는 사실

을 확인했다.

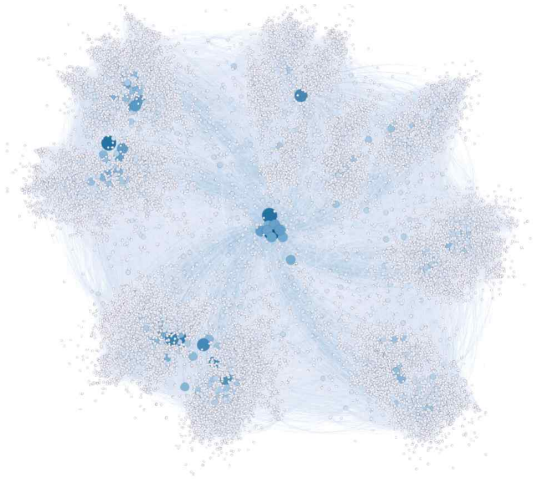


그림 2. 이더리움 동적 토폴로지 스냅샷

이전 연구의 토폴로지 스냅샷과 비교하여 토폴로지는 커뮤니티가 상대적으로 명확하게 확인되고 커뮤니티의 가장자리 노드가 작게 발견되었다. 이는 네트워크에 영향력을 작게 미치는 노드 즉 네트워크에 참여와 탈퇴를 반복하는 노드들을 제외한 결과이고 정확도 높은 토폴로지를 확인할 수 있다.

V. 결론

본 논문에서는 이더리움 네트워크에서 동적으로 변화하는 토폴로지를 액티브 프루빙 방법을 사용하여 정확도 높은 토폴로지 탐색 방법을 제시한다.

이전 연구[11]의 노드 탐색과정을 반복하여 패시브 프루빙 데이터를 수집한다. 패시브 프루빙 데이터는 탐지 노드와 연결된 피어의 노드 목록이고 이더리움 네트워크에 특정 시간동안 참여 유무를 확인하기 어렵다. 이에 따라 본 논문에서는 액티브 프루빙 방법을 사용하여 이더리움 네트워크에 참여하고 있는 노드의 활성화 유무를 확인한다. 이더리움 네트워크의 토폴로지를 측정하기 위해서는 패시브 프루빙 데이터와 액티브 프루빙 데이터를 선형 탐색(Linear Search)방법으로 토폴로지 데이터를 수집 한다. 이더리움 동적 토폴로지 실험 결과 13,261개의 노드와 353,606개의 노드들의 연결을 확인했다. 이는 이더리움 노드 추적 웹에서 발견된 노드의 1.5배 이상의 수치를 도출했다.

본 연구는 이더리움 네트워크의 토폴로지를 측정하는 방법을 제시한다. 이를 통해 네트워크에 참여와 이탈을 반복하는 노드들의 토폴로지를 확인할 수 있다. 또한 이더리움 토폴로지의 분석 결과와 시각화는 이더리움 네트워크를 이해하는데 사용된다. 따라서 향후 블록체인 토폴로지에서 영향력 있는 피어 선정을 통한 네트워크 성능 향상, 중복 메시지 제거에 대한 연구를 진행할 예정이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2018R1D1A1B07050380).

참 고 문 헌

[1] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." white paper 3.37 (2014).

[2] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." International Workshop on Peer-to-Peer Systems. Springer, Berlin, Heidelberg, 2002.

[3] Saad, Muhammad, et al. "Exploring the attack surface of blockchain: A systematic overview." arXiv preprint arXiv:1904.03487, 2019.

[4] Wang, Xu, et al. "Attack and defence of ethereum remote apis." 2018 IEEE Globecom Workshops (GC Wkshps). IEEE, 2018.

[5] Kim, Seoung Kyun, et al. "Measuring ethereum network peers." Proceedings of the Internet Measurement Conference 2018.

[6] Gao, Yue, et al. "Topology Measurement and Analysis on Ethereum P2P Network." 2019 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2019.

[7] Imtiaz, Muhammad Anas, et al. "Churn in the bitcoin network: Characterization and impact." 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019.

[8] Grundmann, Matthias, Till Neudecker, and Hannes Hartenstein. "Exploiting transaction accumulation and double spends for topology inference in bitcoin." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018.

[9] Miller, Andrew, et al. "Discovering bitcoin's public topology and influential nodes." et al., 2015.

[10] Essaid, Meryam, Sejin Park, and Hongteak Ju. "Visualising Bitcoin's Dynamic P2P Network Topoogy and Performance." 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019.

[11] Maeng, Soo Hoon, Meryam Essaid, and Hong Taek Ju. "Analysis of Ethereum Network Properties and Behavior of Influential Nodes." 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2020, p. 203-20.

[12] Ethereum devp2p,

"https://github.com/ethereum/devp2p", 3월 접속

[13] Go-Ethereum Client,

"https://github.com/ethereum/go-ethereum", 3월 접속

[14] Ethereum Requirements,

"https://ethereum.org/ko/developers/docs/nodes-and-clients/", 3월 접속

[15] Kim, Seoung Kyun, et al. "Measuring ethereum network peers." Proceedings of the Internet Measurement Conference 2018. 2018. p. 91-104.

[16] Ethereum Node Explorer, https://www.ethernodes.org/ 2018, 9월, 12일 접속

사물인터넷(IoT)에서의 블록체인 활용 관련 연구동향 분석

강창훈, 최원석, 우종수, 홍원기
포항공과대학교 컴퓨터공학과

{chkang, ws4583, woojs, jwkhong}@postech.ac.kr

Analysis of Research Trends Related to the Use of Blockchain in the Internet of Things (IoT)

Changhoon Kang[1], Wonseok Choi[1], Jongsoo Woo[2], James Won-Ki Hong[1]
[1]Department of Computer Science and Engineering, POSTECH
[2]Graduate School of Information Technology, POSTECH

요 약

사물인터넷(Internet of Things, IoT)은 사물에 연산 및 통신 기능을 부여함으로써 이들을 연결해주는 기술을 말한다. 지금까지 스마트홈, 스마트팩토리, 커넥티드카 등의 분야에서 IoT 기술이 많이 활용되고 있으며, 점점 더 다양한 영역으로 활용 범위가 넓어질 것으로 기대된다. 기존의 전통적인 IoT 시스템은 중앙화 된 주체가 네트워크에 연결된 모든 기기들의 인증이나 접근 권한 등을 관리한다. 하지만 이런 형태의 모델은 단일 장애점 (Single Point of Failure)과 같은 위험을 가지고 있으며, 수많은 기기들의 신원 관리와 그들로부터 수집한 데이터의 무결성을 보장하는 것이 어렵다. 따라서 탈중앙화 되어 있고 신원 관리 기능을 제공하며 조작이 불가능한 특성을 가지는 블록체인을 IoT 와 접목시켜 해당 어려움을 해결하려는 여러 연구가 진행되고 있다. 일반적인 블록체인들은 시스템 운영을 위해, 연결된 노드들이 충분한 연산 능력과 많은 저장 공간을 갖고 있어야 한다. 반면에 대부분의 IoT 기기들은 제한된 크기의 연산 능력과 저장 공간을 갖고 있기 때문에, 기존의 블록체인 시스템에서 노드로 동작하기에는 어려움이 있다. 본 논문에서는 IoT 기기와 같이 자원이 제한된 디바이스들로 구성된 네트워크에서 블록체인을 효율적으로 활용하기 위한 연구의 동향을 분석했다. 블록체인의 경량화를 통해 IoT 기기가 블록체인 노드로써 동작할 수 있게 하거나, 충분한 자원을 가진 기기를 IoT 네트워크에 추가하고 동시에 블록체인 노드 역할을 하도록 하는, 현재 크게 두 가지 다른 방식의 접근법으로 연구가 진행되고 있다.

Ⅰ. 서론

IoT [1] 기술은 현재 여러 가전제품과 자동차 등 다양한 기기에 이미 적용되어 일상속에서 많이 사용되고 있다. 인터넷을 통해 연결된 수많은 IoT 기기들은 방대한 양의 데이터를 생성하고 축적한다. 센서와 같은 작은 디바이스가 생산하는 많은 양의 데이터를 수집함으로써 이후 빅데이터 분석을 통해 더욱 발전된 서비스를 제공할 수 있게 된다. 일반적으로 IoT 시스템은 client/server paradigm 으로 디자인되었기 때문에, 네트워크에 속해 있는 기기로부터 생산되는 데이터와 기기들의 접근 제어 (Access Control)를 모두 중앙화 된 주체가 담당해야 한다. 이런 구조는 단일 장애점 (Single Point of Failure) 위험을 야기하고, 데이터의 무결성을 보장할 수 없다. 중앙화 된 주체가 공격받는다면 전체 시스템이 동작할 수 없게 되고, 생산되었던 모든 데이터들이 소실될 수 있다. 따라서 이런 문제점을 블록체인과의 융합을 통해 해결하려는 시도들이 이어지고 있다.

블록체인은 데이터를 블록 단위로 가공하여 사슬 구조로 원장에 저장하고, Peer-to-Peer (P2P) 네트워크 [2] 내에 속한 분산된 모든 노드들이 각자 동일한 원장을 관리한다. 따라서 누구도 임의로 데이터를 몰래 수정할 수 없다. 또한 모든 참여자들이 하나의 온전한 원장을 저장하기 때문에 일부 노드에 Failure 가 발생하더라도 시스템이 동작하는 것에는 아무런 문제가 없다. 이러한 블록체인의 특성 덕분에 블록체인을 IoT 에 결합한다면 앞서 언급된 기존 IoT 시스템의 문제점을 해결할 수 있다.

Proof-of-Work (PoW) [2] 합의 알고리즘을 사용하는 일반적인 블록체인 시스템이 동작하기 위해서는 높은 연산능력 (Computing Power)와 충분한 데이터 저장 공간이 필요하다. [3] 블록 생성을 위해 복잡한 암호학적인 계산들이 필요하고, 시간이 지남에 따라 매번 생성되는 모든 블록들을 영구적으

로 보관해야 하기 때문이다. 하지만 IoT 기기들은 사용할 수 있는 CPU, 메모리, 스토리지 등의 자원이 제한되어 있기 때문에 블록체인 시스템의 노드로써 이용되기에 적합하지 않다. 적정한 수준 이상의 보안을 유지하기 위해서 기존과 같은 수준의 복잡도를 가지는 계산이 필요하다면, 블록을 생성하고 저장하는데 아주 오랜 시간이 걸리게 된다. 또한 시스템 동작 이후 얼마 가지 않아서 스토리지가 가득 차는 상황이 발생할 것이다. 이런 문제점을 해결하기 위해 효율적으로 IoT 와 블록체인 기술을 융합하는 것이 해당 분야의 주요 연구 과제이다.

본 논문에서는 이를 위해 어떤 연구들이 수행되고 있는지 동향에 대해 알아본다. 연구는 크게 두 가지 다른 접근법으로 나눌 수 있었다. 첫 번째 접근은 블록체인의 경량화를 통해 IoT 기기가 블록체인 노드로써 동작할 수 있도록 하는 것이다. 두 번째 방법은 충분한 자원을 가진 기기를 IoT 네트워크에 추가하여 동시에 블록체인 노드의 역할을 수행하도록 하는 것이다. 본 논문의 나머지 부분에서는 각 접근 방식에 해당하는 연구들을 소개한다.

II. IoT 와 블록체인의 융합

본 논문에서는 IoT 와 블록체인의 융합에 대한 연구를 크게 두 가지 다른 접근 방식을 정의하여 분류했다. IoT 기기를 그대로 블록체인 노드로 사용할 수 있도록 블록체인의 경량화를 진행하는 연구 방식이 한 가지 접근법이다. 반대로 IoT 네트워크에 자원이 충분한 기기를 추가하여 이것이 블록체인 노드의 역할을 수행하도록 하는 접근법이 존재한다. 대부분의 연구들이 이 두 가지 접근법 중 하나에 포함되는 것을 확인했다.

1. IoT 기기를 블록체인 노드로 사용

첫번째 접근법은 제한된 크기의 자원을 가진 IoT 기기들을 직접 블록체인 노드로 활용하는 방식이다. 대부분의 연구에서 기기들의 연산능력과 스토리지의 부족을 극복하기 위해, 블록체인을 구성하는 블록이나 트랜잭션의 구조를 수정하고 더욱 가벼운 합의 메커니즘을 새롭게 제안하는 등 블록체인의 경량화에 초점을 두고 있다.

Koshy, Babu, and Manoj [4]는 IoT 시스템 적용에 적합하도록 블록체인 구조를 수정한 Sliding Window Blockchain (SWBC)을 제안했다. (그림 1) 이 연구의 주요 목적은 IoT 시스템의 보안 문제를 블록체인 적용을 통해 해결하는 것이다. 해당 연구에서는 IoT 기기들을 각각 하나의 블록체인 노드로써 동작하도록 했고, IoT 기기들의 제한된 자원만으로도 블록체인 시스템이 운영될 수 있도록 Sliding Window 라는 개념을 통해 새로운 블록을 생성 및 관리하는 방법을 제안했다. 기존 블록체인 구조에서는 블록 헤더에 이전 블록의 Hash 값을 저장한다. 따라서 기록된 데이터를 수정하는 것은 그 이후로 생성된 모든 블록들을 새롭게 생성해야 하기 때문

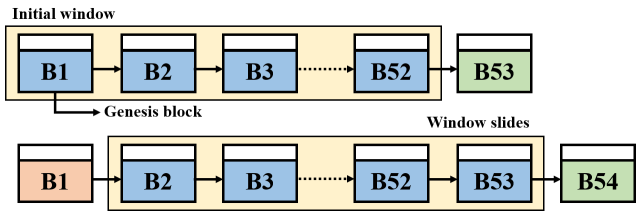


그림 1 Sliding Window Blockchain

에 매우 어렵다. 제안된 방법은 이전 블록 하나의 Hash 를 저장하는 것이 아니라 Sliding Window 크기만큼의 최근 블록들의 hash 를 저장한다. Sliding Window 의 크기는 현재 N 개의 블록이 있다면 1 과 (N-1) 사이의 값을 임의로 가질 수 있다. 즉 마이닝을 하기 위해서는 Window Size 와 그만큼의 최신 블록들을 알고 있어야 하고 이 Window Size 는 마이너들에게만 공개된다. 이 방법을 사용하면 IoT 기기들은 Window Size 보다 앞서서 생성된 블록들을 로컬 스토리지에 저장해둘 필요가 없기 때문에 이를 삭제함으로써 기기의 스토리지 오버헤드를 줄일 수 있다. 모든 블록 데이터들은 IoT 기기에서는 시간이 지나면서 삭제되지만 프라이빗 클라우드에 따로 저장해 계속 보관된다. 또한 이 방법에서는 Proof-of-Work (PoW)의 Difficulty 를 불필요하게 높이지 않고 상황에 따라 알맞게 선택하는 기술을 이용해 PoW 를 간소화함으로써 효율성을 높였다. 결론적으로 스토리지와 연산의 오버헤드를 줄임으로써 IoT 시스템에 블록체인이 적용될 수 있도록 했고, 이를 통해 기존 IoT 시스템의 보안 문제를 해결할 수 있었다.

Huang, et al. [5]는 엣지 컴퓨팅 환경에서 효율적으로 블록체인의 블록 데이터를 저장하는 방법을 제안하고 에너지 소비를 줄일 수 있는 새로운 Proof-of-Stake (PoS) [6] 메커니즘을 제안했다. 이 방법에서 달성하고자 하는 주요 목표는 효율적인 스토리지 사용과 IoT 기기들의 에너지 소비를 낮추는 것이다. IoT 기기들처럼 엣지 컴퓨팅 환경의 기기들은 블록체인의 전체 데이터를 모두 저장하기에는 저장 공간이 매우 부족하다. 따라서 특정 블록 데이터를 모든 기기가 저장하는 것이 아니라 할당된 기기의 저장 공간에만 저장한다. IoT 기기들은 모두 제한된 크기의 스토리지를 갖고 있지만 기기마다 전체 크기나 남은 용량의 차이가 존재한다. 이 기법의 핵심 아이디어는 남은 용량이나 접근의 용이성에 따라 최적의 기기를 데이터 저장을 위해 할당하는 것이다. 시스템은 모든 기기들이 동등한 비율의 저장공간을 사용하도록 한다. 즉, 자원이 많은 기기에는 그만큼 더 많은 공간의 할당이 발생한다. 데이터에 쉽게 접근하기 위해서는 무선 통신에서 데이터 손실을 줄이기 위해 노드의 이동성 (Mobility)이 작은 것이 유리하다. 따라서 특정 두 노드들 간의 거리와 각자의 이동성에 대한 비용을 계산하여 이를 스토리지 할당에 활용했다. 결과적으로 시스템은 효율적으로 모든 기기들의 저장 공간을 관리할 수 있고, 블록 데이터의 탐색이 빨라진다.

이 연구에서는 에너지 소비가 심한 PoW 대신 새로운 형태의 PoS 합의 메커니즘을 제안했다. 각 기기의 시스템에 대한 기여도에 따라 마이닝 성공 확률이 달라진다. 많은 저장 공간이 할당되어 있을수록 더 높은 확률로 블록 채굴에 성공할 수 있다. 이런 특징 덕분에 데이터를 생성하는 기기가 아니더라도 단순히 저장 공간을 제공하기 위해 시스템에 더 많은 노드들이 참여하는 것을 유도할 수도 있다.

2. 충분한 자원의 기기를 블록체인 노드로 사용

두 번째 접근법은 자원이 충분한 다른 기기들을 IoT 기기 대신에 블록체인 노드로 사용하는 방식이다. 이 방법에서는 IoT 기기들을 그룹화하고, 각 그룹마다 블록체인 노드 역할을 하는 충분한 자원의 기기를 둔다. 따라서 해당 충분한 자원의 기기들을 통해서 IoT 기기가 블록체인과 상호작용하게 된다.

Yazdinejad, et al. [7]는 SDN controller 를 블록체인 노드로 사용하는 아키텍처를 제안했다. 이 연구의 핵심 아이디어는 자원이 한정되어 있는 IoT 기기들 대신 이미 네트워크 상에 존재하면서 블록체인 노드로써 역할을 하기에 충분한 자원을 가진 SDN controller 를 활용하는 것이다. 이 연구가 제안한 아키텍처에서는 IoT 기기들이 클러스터 구조로 나누어져 있고, 각 클러스터를 담당하는 SDN controller 가 존재한다. SDN controller 들 사이의 퍼블릭 블록체인을 구성하고, 각 클러스터들마다 IoT 기기들 간의 프라이빗 블록체인을 운영한다. 즉, 자원이 부족해도 운영이 가능한 프라이빗 블록체인을 통해 IoT 기기들의 신원 관리를 수행하고, 운영에 충분한 자원이 필요한 퍼블릭 블록체인을 SDN controller 가 관리한다. 이를 통해 데이터 무결성 보장 등 기존 IoT 시스템의 문제점을 해결할 수 있다.

Novo [8]는 블록체인을 이용해 IoT 시스템에서의 분산 접근 제어 시스템을 제안했다. 이 연구는 기존 IoT 시스템의 보안보다는 확장성 문제에 초점을 맞추고 있다. 하나의 중앙화 된 서버에서 수십억 개의 IoT 기기로의 접근을 모두 관리하게 되면 확장성 문제가 발생한다. 접근 제어 과정이 자주 발생하게 되면 병목현상이 일어나 시스템 전체의 성능을 저하시키게 된다. 이 연구에서는 일반적인 형태의 블록체인을 구성하고 하나의 노드가 관리 허브 (Management Hub)를 통해 하나의 지정된 IoT 기기 그룹과 상호작용하도록 한다. (그림 2) 관리 허브는 IoT 기기들과 블록체인 노드 사이의 인터페이스 역할을 수행한다. 접근 제어와 관련된 규칙은 모두 블록체인에 등록된 하나의 스마트 컨트랙트 내에 정의되어 있다. IoT 기기로부터 접근 제어 쿼리가 발생하게 되면 기기가 속한 그룹의 관리 허브를 통해 연결된 노드로 전달된다. 모든 노드는 블록체인 데이터를 다 저장하고 있기 때문에 스마트 컨트랙트에 쿼리를 보낼 필요없이 노드의 로컬 데이터 확인을 통해 전달받은 쿼리에 응답할 수 있다. 결과적으로 IoT 기기들의 그룹마다 별도의 담당 노드를 둠으로써 시스템의 확장성을 더욱 높일 수 있다.

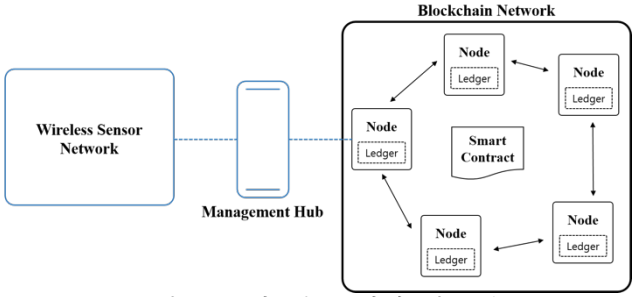


그림 2 분산 접근 제어 시스템

Ⅲ. 결론

본 논문에서는 자원이 제한된 IoT 기기들로 구성된 네트워크에 블록체인을 적용시키기 위해서 진행되고 있는 연구의 동향에 대해 분석하였다. 크게 두 가지 다른 접근법을 정의할 수 있었고, 각각에 해당되는 연구들을 정리해 소개하였다. 첫 번째 방법은 블록체인을 구성하는 요소들의 구조를 변경하거나 새로운 합의 메커니즘을 제안함으로써 블록체인 시스템 자체를 경량화 한다. 따라서 자원이 한정된 IoT 기기들이 직접 블록체인 노드로 네트워크에 참여하는 형태를 갖고 있다. 두 번째 방법은 자원이 충분한 다른 기기들이 블록체인 네트워크의 노드로써 동작하도록 한다. IoT 기기들을 그룹화하여 각 그룹마다 지정된 노드를 통해 블록체인과 상호작용하게 된다.

현재 해당 분야의 연구들은 대부분 기존 IoT 시스템이 client/server paradigm 으로 인해 가지는 보안이나 확장성 문제를 블록체인과의 접목을 통해 해결하기 위해서, IoT 시스템과 블록체인의 효율적인 융합 방법을 찾아내는 것이 목적이다. 블록체인이 가지는 탈중앙성, 데이터 불변성 등의 특징을 해치지 않으면서 자원이 제한된 IoT 기기들로 구성된 시스템에 접목시킬 수 있는 방법을 지속적으로 연구하는 것이 필요해 보인다.

ACKNOWLEDGMENT

본 연구는 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발, IITP-2021-2017-0-01633*, 대학 ICT 연구센터육성지원사업)

참 고 문 헌

[1] ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. Computer networks, 2010, 54.15: 2787-2805.

[2] Zheng, Zibin, et al. "An overview of blockchain technology: Architecture, consensus, and future trends." 2017 IEEE international congress on big data (BigData congress). IEEE, 2017.

[3] GERVAIS, Arthur, et al. On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016. p. 3-16.

[4] KOSHY, Prescilla; BABU, Sarath; MANOJ, B. S. Sliding Window Blockchain Architecture for Internet of Things. IEEE Internet of Things Journal, 2020, 7.4: 3338-3348.

[5] HUANG, Yaodong, et al. Resource allocation and consensus on edge blockchain in pervasive edge computing environments. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019. p. 1476-1486.

[6] King, Sunny, and Scott Nadal. "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake." self-published paper, August 19 (2012): 1.

[7] YAZDINEJAD, Abbas, et al. An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. IEEE Transactions on Services Computing, 2020.

[8] NOVO, Oscar. Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE Internet of Things Journal, 2018, 5.2: 1184-1195.

차량 블랙박스 영상 데이터의 위변조 및 프라이버시 문제 해결을 위한 경량화 블록체인

나동준, 박세진*
계명대학교

nadongjun@kmu.kr, *baksejin@kmu.ac.kr

Lightweight blockchain to solve forgery and privacy issues of vehicle black box image data

Dongjun Na, Sejin Park*
Keimyung University

요 약

본 논문은 교통사고가 발생할 경우에 대한 책임 소재 판단과 사고 예방을 위해 필수적인 역할을 하는 블랙박스 영상 데이터에 대한 위변조와프라이버시 문제 해결을 위한 블록체인 활용 방법을 제안한다. 차량 블랙박스 IoT 또는 커넥티드 카 내부에서 동작 가능한 블록체인을 사용하며 이를 위해 저전력, 저용량 기기에서의 동작을 위해 크기를 경량화하며 PBFT 합의 알고리즘을 통해 무선 네트워크를 사용하는 비동기 분산 시스템 상황에서도 블록체인 노드 간 합의를 위한 메세지의 전달 실패, 지연, 중복으로 인한 시스템 오류를 방지할 수 있다. 본 논문의 경량화 블록체인을 통해 블랙박스 보안 문제를 해결할 수 있다.

Ⅰ. 서 론

최근 교통사고가 발생하는 경우, 사고 발생의 책임 소재에 대한 판단을 용이하게 하고, 사고예방의 효과가 높은 이유로 택시, 버스와 같은 대중교통 시설과 개인 차량에 교통사고 상황을 영상으로 기록할 수 있는 차량용 영상기록 블랙박스 (VEDR: Video Event Data Recorder)의 장착이 증가하고 있다.

시장조사전문기업 엠브레인 트렌드모니터[1]의 월 평균 1 회 이상 직접 운전을 하는 전국 성인남여 1,000 명에 대한 조사 결과 블랙박스의 필요한가라는 질문에 95.3%가 필요하다고 응답하였으며 필요하다고 생각하는 이유로는 “사고 발생에서의 잘못을 가리기 위해”, “잘못을 부인할 경우 증거자료로 사용하기 위해” 였다. 이러한 통계자료로 보아 블랙박스 영상은 교통사고의 사고 증거를 위해 필수적이며 영상 데이터에 대한 보안이 중요하다.

하지만 ETRI 의 차량용 블랙박스 보안 이슈 동향[2]에 따르면, 차량용 블랙박스 데이터에 대한 보안 문제점이 존재한다고 한다. 보안 문제 중 가장 직접적인 문제는 블랙박스에 저장된 영상 데이터의 위변조 문제이다. 차량용 블랙박스 영상 데이터의 경우, 사고 원인의 규명 및 사고 상황을 판단하는데 참고자료로 사용되며 사고 데이터를 법적인 증거로서 활용할 수도 있다.

차량용 블랙박스는 차량 소유자나 운전자와 같이 제한적인 사용자만 접근 가능하기 때문에 차량에 부착된 블랙박스 영상 데이터에 대해 완벽한 접근 권한을 가지고 있으며 공격을 수행하는데 필요한 시간적인 제약이 없다.

따라서 일반적인 차량용 블랙박스 데이터는 위변조 가능성에 항상 노출돼있다. 차량용 블랙박스 영상 데이터는 자체 메모리 또는 SSD 메모리를 사용하며, 저장된 데이터는 필요에 따라 사용자가 불리할 경우 메모리 자체를 파손해서 영상 데이터의 사용 자체를 방해할 수도 있다. 실제 KS 표준[3]에 위변조 방지를 위한 기준이 추가되어 있지만 여전히 위변조 가능성이 존재한다.

또한 메모리에 저장된 일부 영상 데이터에 대하여 사용자의 이익에 따라서 부분적인 삭제와 위조, 변경이 가능하다. 또한 블랙박스는 프라이버시 문제가 존재한다.

사고 발생시 정확한 사고 규명을 위해 상용 차량에서 부착되는 블랙박스의 경우, 차량 내부의 영상과 음성을 촬영을 하여 밀폐된 공간에서의 사용자들의 전화 기록, 대화 등 개인적인 정보를 저장할 수 있어 사생활 침해소지가 있다.

상용 차량 이외에도 개인 차량에서 사용되는 블랙박스도 차량의 운행이 시작되며, 운전자의 이동 경로와 운전 습관 등을 서버로 저장하게 된다. 블랙박스에 저장된 정보가 의도하지 않은 경로로 유출되는 경우에는 운전자에 대한 사생활 침해가 될 수 있다.

이러한 문제를 해결하기 위해서는 블랙박스 영상 데이터에 대한 신뢰성과 무결성 및 기밀성이 필요하다. 블록체인[4] 기술과 PKI(Public key infrastructure) 암호화 기술은 이 문제를 해결하기에 적합하다.

블록체인 기술은 중앙 서버를 사용하지 않고, 관리 대상 데이터를 블록이라고 하는 소규모 데이터들이 P2P 방식을 기반으로 생성된 체인 형태의 연결고리 기반 분산 데이터 저장 환경에 저장하여 누구라도 임의로

수정할 수 없고 누구나 변경의 결과를 열람할 수 있는 분산 컴퓨팅 기술 기반의 원장 관리 기술이다.

Ⅱ. 본론

블록체인은 탈중앙화와 합의 알고리즘으로 블랙박스 영상 데이터에 대한 데이터 신뢰성 및 위변조 방지를 할 수 있지만 모든 노드가 열람 가능한 블록체인에 데이터를 저장하므로 영상 데이터에 대한 프라이버시 문제점을 해결할 수 없다. 또한 블록체인의 크기가 증가하는 문제로 인해 스토리지 용량이 제한된 블랙박스 IOT[5], 커넥티드 카[6]와 같은 곳에서 블록체인 노드를 동작시킬 수 없어 클라우드[7] 또는 엣지 컴퓨팅[8] 기술을 활용하며 이로 인해 데이터를 차량 내부에서 동작 가능하지 않으며 중앙화로 인한 문제점을 해결할 수 없다.

본 논문에서는 블록체인의 크기 문제와 프라이버시 문제를 해결하여 차량 블랙박스 영상 데이터의 위변조와 프라이버시 문제를 해결할 수 있는 블록체인을 제안한다.

크기 경량화를 위해 차량에서 생성되는 블랙박스 영상 데이터를 IPFS(InterPlanetFileSystem)[9]에 업로드 하는 방식으로 활용한다. IPFS 는 분산형 파일 시스템으로 파일을 IPFS 네트워크의 노드에 분산 저장하고 IPFS hash 값을 반환하며 이를 통해 여러 컴퓨터에 분산 저장되어 있는 파일을 가져온 후 하나로 합쳐주는 방식으로 작동한다.

이를 활용하여 분산 저장 후 반환되는 Hash 값만을 블록체인에 저장하여 블록체인을 경량화 할 수 있다. 또한 프라이버시를 위해 IPFS 에서 반환된 Hash 값에 대한 접근제어를 위해 PKI 기반 암호화를 통해 사용자의 공개키로 암호화하여 개인키를 보유한 특정 사용자만 영상 데이터에 접근할 수 있게 한다.

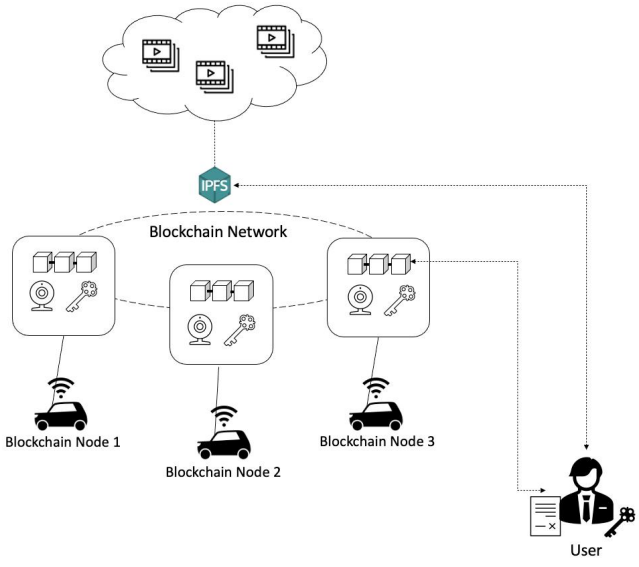


Fig 1. 블록체인 시스템 아키텍처

Fig 1.은 본 논문에서 제안하는 블록체인의 시스템 아키텍처이다. 차량 내 구동되는 블랙박스 또는 커넥티드 카 내부 IoT 에서 블록체인 노드가 유지되며, 차량 블랙박스에서 생성되는 영상을 경량화 위해 IPFS 노드가 존재한다. 이후 블랙박스 영상 데이터를 접근할 수 있는 블록체인 노드의 개인키를 보유한 사용자 또는 관리자가 블록체인에서 암호화된 IPFS hash 를 반환 받고 개인키로 복호화하여 IPFS hash 값을 통해 영상 데이터를 다운로드 할 수 있다.

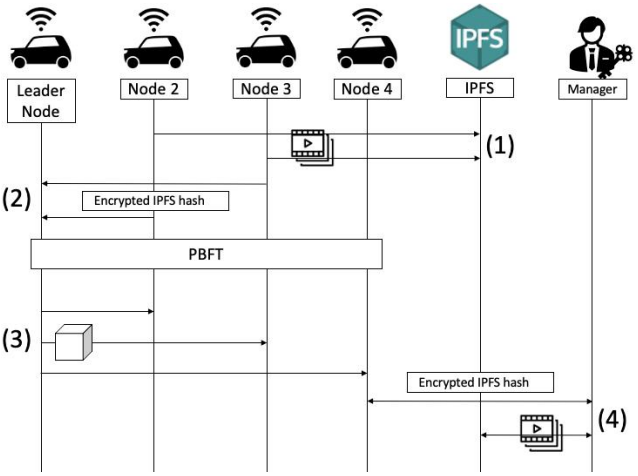


Fig 2. 블록체인 시스템 플로우 차트

Fig 2.는 본 논문에서 제안하는 블록체인의 플로우 차트이다. 블록체인 노드가 4 개, IPFS, Manager 가 존재하며 PBFT 합의를 위해 블록을 생성하는 Leader Node 가 존재한다. 플로우 차트는 다음과 같다.

(1) 블록체인 노드(블랙박스가 존재하는 차량)에서 영상 데이터를 생성하고 IPFS 에 업로드한다.

(2) IPFS hash 를 반환 받고 영상 데이터를 생성한 노드의 공개키로 암호화 받고 Leader Node 에게 전송한다. (3) 암호화된 IPFS hash 를 받고 블록을 생성하고 PBFT 합의를 통해 블록을 검증한다.

(3) 검증한 블록을 모든 블록체인 네트워크의 노드에게 전달한다.

(4) 영상 데이터에 접근 권한(개인키 보유)이 있는 사용자 또는 관리자가 블록체인에서 암호화된 IPFS hash 값을 반환 받고 복호화하여 블랙박스 영상을 다운로드 받는다.

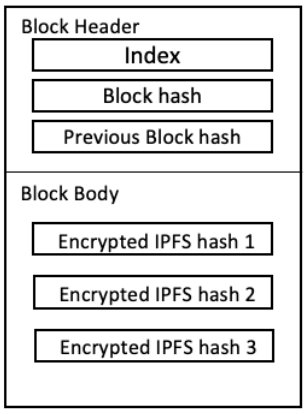


Fig 3. 제안한 블록체인의 블록 구조

Fig 3.은 본 논문에서 제안하는 블록체인의 블록 구조이다. Block Header 에는 블록의 번호인 Index 가 존재하고 블록 내 모든 데이터를 해시화한 블록 해시 이전 블록의 해시값이 존재한다.

Block Body 에는 블록체인 네트워크의 노드들이 보유한 공개키로 암호화한 IPFS hash 값이 존재한다.

사용자 또는 관리자는 개인키를 통해 복호화하여 블랙박스 영상에 접근할 수 있다.

블록체인에 블랙박스 영상을 IPFS 에 업로드하고 다운로드 지연 시간을 측정하여 성능에 끼치는 정도를 평가하는 실험을 진행하였다.

Table 1. 시중 블랙박스 영상 해상도와 프레임

Resolution	FPS
320x240px-2560x1440px	8-60

Table 1.은 시중 블랙박스 영상 데이터의 해상도와 프레임이다.

Equ 1. 영상의 세로 해상도 x 영상의 가로 해상도 x 영상의 색 (비트) x 영상의 초당 프레임 수 / 8 = 영상 용량

Equ 1.은 영상의 용량 계산 방법이다. 해당 식을 통해 시중 블랙박스 영상 데이터의 크기를 계산하여 실험하였다. 영상의 시간은 10 초로 통일하였다.

본 논문의 실험을 위해 사용한 데이터셋이다. 최대 해상도와 최소 해상도를 가진 영상의 fps 는 KS 표준[10]으로 정해진 영상인 프레임율(frame rate)이 30fps 이며 사고발생 전후 각 10 초 분량의 영상을 가정하 여 사용하였다. 영상의 해상도는 1280x720, 용량은 20.4MB 를 사용하였다. 또한 10MB, 30MB, 40MB, 50MB 용량의 영상 데이터를 업로드하고 다운로드하여 용량 증가에 따른 지연시간 증가폭을 확인하였다.

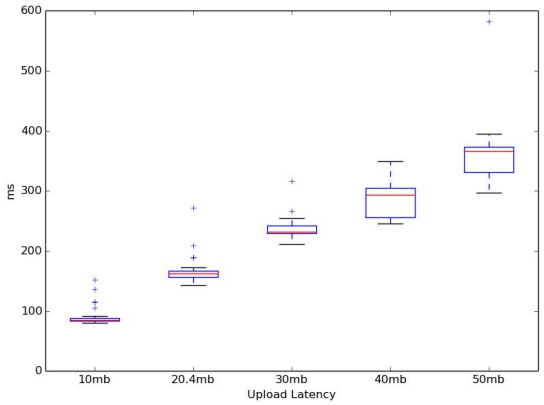


Fig 4. 블랙박스 영상 업로드 지연시간

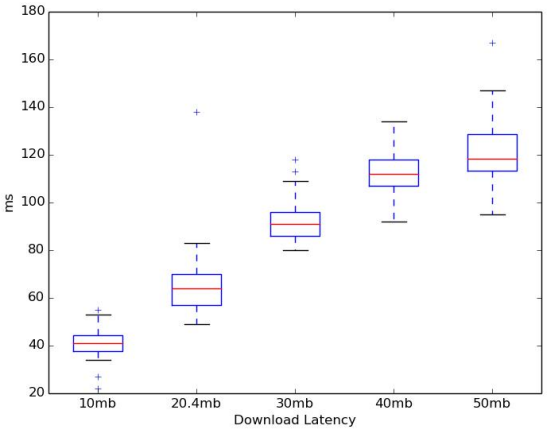


Fig 5. 업로드된 블랙박스 영상 다운로드 지연시간

Fig 4.은 블랙박스 영상의 표준에 포함되는 크기의 영상 데이터인 10MB, 20.4MB, 30MB, 40MB, 50MB 를 IPFS 에 업로드 하였을 때 지연시간을 나타내는 그래프이다. 표준 크기인 20.4MB 일 경우 최소 143, 최대 272ms 가 소요됐으며 업로드 용량에 따른 증가폭 또한 선형적으로 증가하였다.

Fig 5.는 IPFS 에 업로드 후 반환 받은 IPFS hash 값으로 10MB, 20.4MB, 30MB, 40MB, 50MB 의 용량의 영상을 다운로드 했을 경우 그래프이다. 표준 크기인 20.4MB 를 다운로드 받은 경우 최소 49, 최대 138ms 가 소요되었다. 다운로드 또한 용량이 10MB 가 증가할 때 마다 지연시간이 선형적으로 증가하였다.

두 그래프 결과값에서 업로드와 다운로드 모두 20.4MB 의 용량은 성능에 영향을 미치지 않을 만큼 지연시간이 소요됐으며 선형적인 증가폭을 보이며 크기의 증가 에 큰 영향을 받지 않는 것을 확인하였다.

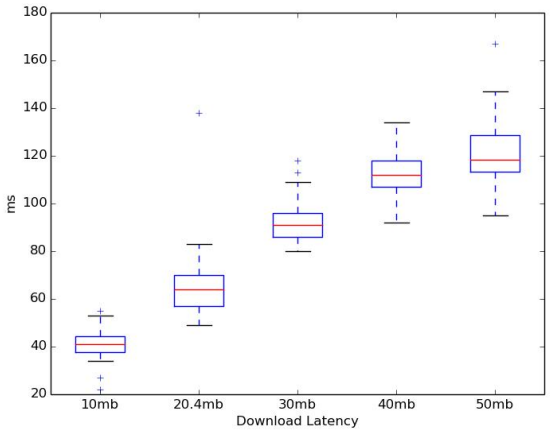


Fig 6. PBFT 지연시간

Fig 6.는 블록체인 노드를 3 개 두고 암호화된 IPFS hash 값에 대한 PBFT 합의 요청을 하였을 경우 총 시간 그래프이다. PBFT 측정 결과 또한 최소 31, 최대 82ms 의 결과를 나타내며 0.1 초 이하의 지연시간을 보여 메모리 사용량과 CPU 연산 능력을 요구하지 않으며 블록체인 노드간 합의를 할 수 있다는 것을 확인하였다.

Ⅲ. 결론

본 논문에서는 블랙박스 영상 데이터에 대한 위변조 가능성과 프라이버시 문제점을 해결하기 위한 경량화 블록체인을 제안하였다. IPFS 에 영상을 업로드하여 영상 크기를 IPFS hash 크기인 46byte 로 경량화 시켜 블록체인에 저장하고 지연시간 또한 성능에 큰 영향을 미치지 않으며 용량에 따른 지연시간 증가량 또한 선형적인 것을 확인하였다. 또한 PBFT 합의를 통해 연산량을 거의 요구하지 않고 블록체인 네트워크 노드 간 데이터에 대한 합의를 하였다.

또한 블록체인에 저장된 IPFS hash 값을 사용자의 공개키로 암호화하여 개인키를 가진 이용자만 영상 데이터에 접근할 수 있게 하여 프라이버시를 보장할 수 있다. 추후 연구에서는 PBFT 합의 알고리즘의 문제점인 노드의 수가 증가할 경우 네트워크 통신량이 많아지는 문제점을 해결해야 한다.

ACKNOWLEDGMENT

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1G1A1100305).

참 고 문 헌

[1] 김세진, 차량용 블랙박스 설치 해마다 증가...설치율 89%, (http://www.datasom.co.kr/news/articleView.html?idxno=99167).

[2] 김무섭, 최수길, 정치윤, 한종욱. "차량용 블랙박스 보안 이슈 동향", 전자통신동향분석 제 27 권 4 호, pp. 123-129, 2012 년 8 월

[3] 김민기, 'KS 인증' 차량용 블랙박스, 사고영상 위·변조 가능 '논란' (https://news.joins.com/article/17356039).

[4] Blockchain. (https://en.wikipedia.org/wiki/Blockchain)

[5] 연찬모, 이통사, 저전력 통신기술 기반 'IoT 블랙박스' 경쟁 나서 (http://biz.newdaily.co.kr/site/data/html/2018/08/27/2018082700002.html)

[6] 심현보, 커넥티드 카의 기술, 한국정보통신학회논문지

[7] Steve Ranger, What is cloud computing? Everything you need to know about the cloud explained, ZDNet(zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/)

[8] IBM, What is edge computing? (https://www.ibm.com/cloud/what-is-edge-computing)

[9] J. Benet, “Ipfs-content addressed, versioned, p2p file system,” arXiv preprint arXiv:1407.3561, 2014

[10] 현종환 , 정재윤 , 리건 , 홍원기. “무결성을 보장하는 차량용 블랙박스 동영상 전송 및 관리 시스템”, KNOM

가상 네트워크 관리를 위한 기계학습 기반 이상 탐지 시스템 설계

홍지범, 남석현, 유재형, 홍원기

포항공과대학교 컴퓨터공학과

{hosewq, obiwan96, jhyoo78, jwkhong}@postech.ac.kr

A Design of Machine Learning-based Anomaly Detection System for Virtual Network Management

Jibum Hong, Jae-Hyoung Yoo, James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH

요 약

5G 네트워크 기술은 유무선 통신 기술을 포함하여 네트워크 슬라이싱 및 클라우드 컴퓨팅을 주요 구성 기술로 활용한다. 이러한 네트워크 슬라이싱 및 클라우드 컴퓨팅의 근간이 되는 기술인 가상화 (Virtualization) 기술은 Software-Defined Networking (SDN), Network Function Virtualization (NFV)를 통해 통신 사업자 및 서비스 제공업체가 제공하는 서비스의 보다 효율적인 관리를 가능하게 한다. 하지만 가상 네트워크가 점점 복잡해짐에 따라 다양한 네트워크 관리 문제가 발생하게 되고, 이러한 관리 문제를 해결하기 위해 네트워크 관리자는 물리 계층의 네트워크만이 아닌 가상 계층의 네트워크에서 운용되는 Virtualized Network Function (VNF) 및 Cloud-native Network Function (CNF)의 자원 사용량 및 네트워크 트래픽을 모니터링하고 분석할 필요가 있다. 본 논문에서는 이러한 네트워크 관리 문제 해결을 위해 장애 관리 기술 중 하나인 이상 탐지 (anomaly detection)에 초점을 맞추어 가상 네트워크 운용 중 발생하는 고장 및 과부하, 서비스 품질 저하 등의 이상 (anomaly) 상태를 탐지하는 시스템을 제안한다. 제안하는 시스템은 물리 및 가상 네트워크에서 수집되는 데이터를 통해 기계학습 (Machine Learning) 모델을 학습시켜 이상 상태를 탐지한다.

Ⅰ. 서 론

오늘날의 5G 네트워크는 초고속 (eMBB), 초연결 (mMTC), 초저지연 (URLLC)을 목표로 기존보다 더 많은 서비스를 효율적으로 지원하기 위해 설계되었다. 5G 네트워크의 주요 구성 기술인 네트워크 슬라이싱과 클라우드 컴퓨팅 기술은 사용자 및 서비스 요구에 따라 여러 개의 논리적이고 독립적인 종단간 (end-to-end) 네트워크를 제공한다. 이러한 기술들은 Software-Defined Networking (SDN) 및 Network Function Virtualization (NFV) 기술을 기반으로 기존 하드웨어 중심의 폐쇄된 네트워크 기능을 소프트웨어화하여 가상 머신을 이용한 Virtualized Network Functions (VNFs), 또는 컨테이너를 이용한 Cloud-native Network Functions (CNFs)로 대체한다. 이에 따라 통신 사업자 및 서비스 프로바이더는 클라우드 및 데이터 센터의 가상화된 컴퓨팅 자원을 보다 효율적으로 사용하고, 다양한 애플리케이션 서비스를 유연하고 효율적으로 배포하는 것을 가능하게 하여 비용을 절감할 수 있다 [1].

하지만 가상화된 자원을 활용하여 제공되는 서비스들이 증가하면서 가상 네트워크가 점점 복잡해지고, 이로 인해 자원 할당, 장애 관리 등과 같이 다양하고 새로운 네트워크 관리 문제가 발생한다. 이는 서비스 및 시스템의 심각한 장애를 유발할 수 있기 때문에 가상 네트워크 환경을 운용하기 위한 물리 서버 및 VNF (또는 CNF)의 자원 사용량과 네트워크 트래픽, 로그 등을 분석하여 서비스 및 시스템의 상태를 탐지하고 예측할 필요가 있다.

본 논문에서는 여러 가상 네트워크의 관리 문제 중 장애 관리를 위한 이상 탐지 (anomaly detection) 시스템을 설계한다. 제안하는 이상 탐지 시스템은 선행 연구 [2]를 개선하여 모니터링 데이터를 바탕으로 기계학습 (Machine Learning) 모델을 학습시켜 실시간으로 가상 네트워크를 구성하는 물리 및 가상 서버의 고장 및 과부하를 포함하여 SLA 위반과 같은 서비스 품질 저하 등의 이상 상태를 탐지하고 예측한다. 그리고 제안하는 이상 탐지 시스템을 활용한 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

이상을 탐지하는 방법 중 하나인 시스템 및 네트워크 과부하 탐지는 CPU, 메모리, 트래픽 로드 등과 같은 측정치 (metric)의 임계값 (threshold)을 설정하여 임계값을 넘을 경우 과부하 상태로 탐지할 수 있다. 하지만 이는 단순히 특정 순간의 측정치를 기반으로 이상 여부를 판단하기 때문에 과부하 상태를 제외한 다른 종류의 이상 상태 탐지에는 많은 오탐 (false alarm)을 유발한다. 따라서 최근의 이상 탐지 연구는 시계열 (time-series) 데이터를 기반으로 통계 (statistical) 기반의 휴리스틱 (heuristic) 알고리즘이나 기계학습 알고리즘을 이용하여 서비스 및 시스템의 상태를 판단한다. 먼저 통계 기반의

접근법을 사용한 선행 연구 [3]는 NFV 환경에서 수집한 시계열 데이터를 바탕으로 linear regression 및 Mahalanobis distance 를 사용하여 CPU, memory, bit rate 와 같은 각 측정치에 대한 임계값을 정의하여 이상 상태를 탐지한다.

다음으로 최근 네트워크 관리에 기계학습과 같은 인공지능 기술을 적용하여 사람의 개입 없이 네트워크를 관리 연구가 활발히 진행되고 있다. 선행 연구 [4]는 vIMS 환경에서 제공하는 서비스에서 정상 응답 비율을 SLA 로 정의하여 이를 이상 상태를 탐지한다. 해당 연구는 서비스 요청에 대한 정상 응답 비율이 낮을 경우, vIMS 에서 SLA 위반을 유발시킨 컴포넌트의 위치를 분석하는 Root-Cause Analysis (RCA) 기능을 지도학습 기반의 Random Forest 알고리즘을 학습시켜 제공한다.

또한 본 논문의 선행 연구 [2]는 Multi-Access Edge Computing (MEC) 토폴로지로 이루어진 NFV 환경에서 지도학습 기반의 기계학습 알고리즘인 Extreme Gradient Boost (XGBoost)를 통해 여러 VNF 들로 구성된 웹 호스팅 및 인증 서비스의 평균 응답 시간 및 가용성 (availability)을 기반으로 SLA 위반을 탐지하고 SLA 위반의 원인이 되는 VNF 의 위치를 제공한다.

Ⅲ. 기계학습 기반 이상 탐지 시스템

기존 VNF 이상 탐지 시스템은 대부분 CPU 과부하, 메모리 부족, 트래픽 과부하 등의 상태만을 이상으로 판단한다는 한계가 존재하며, 이 외의 이상 상태를 탐지하고자 하는 경우 많은 오탐을 유발한다. 이에 본 논문의 선행 연구 [4]에서는 VNF 의 자원 및 네트워크 사용량을 바탕으로 가상 네트워크에서 운용 중인 서비스의 SLA 위반의 발생 여부와 그 원인이 되는 VNF 의 위치를 탐지하는 방법을 제안한다. 본 논문에서 제안하는 이상 탐지 시스템은 기존 선행 연구 [4]의 기능 확장 및 성능 개선을 목표로 한다. 제안하는 이상 탐지 시스템의 구조는 그림 1 과 같다.

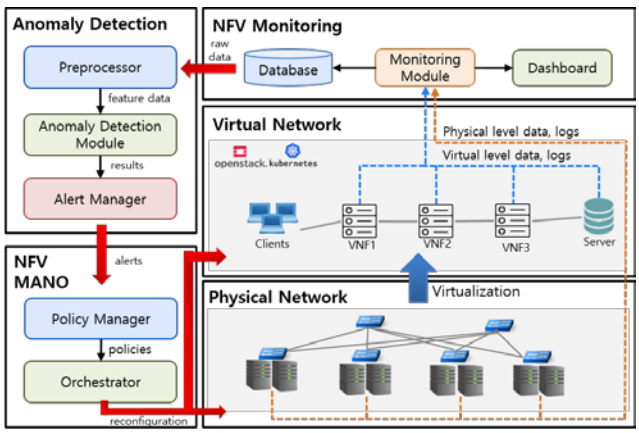


그림 1 기계학습 기반의 이상 탐지 시스템

1) NFV 환경 구축 및 모니터링 기능: 먼저 상용 하드웨어를 통해 구성된 물리 네트워크 환경에서 OpenStack 또는 Kubernetes 와 같은 Virtual Infrastructure Manager (VIM)을 이용하여 NFV 환경을 구축한다. 이러한 NFV 환경은 인공지능 기반의 NFV 관리 플랫폼 연구 [5]를 기반으로 이루어지며, MEC 환경을 모사한 토폴로지로 이루어져 있다. 이를 기반으로 가상 네트워크에서 VNF/CNF 의 SFC 를 구성하여 서비스를 운용한다. 이러한 가상 네트워크 환경에서 모니터링 기능은 기존 선행 연구에서는 VNF 가 동작하는 가상 머신의 CPU, 메모리

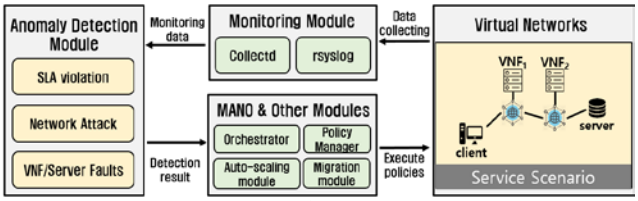


그림 2 이상 탐지 모듈 개념도

와 같은 자원 사용량과 네트워크 트래픽 로드만을 모니터링한 반면, 제안하는 이상 탐지 시스템의 모니터링 기능은 기존 수집하던 가상 네트워크의 데이터를 포함하여 물리 서버의 자원 사용량, 트래픽 로드, 하드웨어의 온도 등을 수집한다. 이 외에도 syslog 를 통해 기록되는 로그 정보를 rsyslog 를 통해 수집한다. 수집된 데이터는 모니터링 모듈로 전달되어 데이터를 대시보드에 전달하여 실시간 모니터링 정보를 시각화하여 제공하거나 데이터베이스에 시계열 형태의 데이터로 저장한다.

2) 기계학습 기반 이상 탐지 기능: 구축된 가상 네트워크 환경 및 모니터링 기능을 기반으로 기계학습 기반의 이상 탐지 기능을 구현한다. 먼저, 데이터베이스에 저장된 데이터 (raw data)를 실시간으로 가져와 전처리기 (preprocessor)를 통해 이상 상태 탐지에 사용하기 위한 feature 데이터로 변환시킨다. 다음으로, 기계학습 기반의 이상 탐지 모듈은 feature 데이터를 기반으로 정상 및 이상 상태를 판단한다. 기존 선행 연구 [2]의 이상 탐지 모듈은 지도학습을 이용하여 학습시킨 탐지 모델을 통해 feature data 를 이용하여 제공 중인 서비스의 SLA (평균 응답 시간 및 가용성) 위반 여부를 예측한다. 모델이 현재 상태를 SLA 위반으로 예측할 경우, 이상 탐지 모듈은 해당 서비스의 SFC 에서 SLA 위반의 원인이 되는 특정 VNF 의 위치를 찾아 네트워크 관리자 혹은 자율 네트워킹 환경에서 NFV 환경의 관리 정책을 세우는 Policy Manager 에게 이상 징후가 발생했음을 알린다.

본 논문에서 제안하는 시스템의 이상 탐지 모듈은 선행 연구를 기능 확장 및 개선하여 그림 2 와 같이 동작한다. 이상 탐지 모듈은 크게 서비스 계층의 이상 상태를 탐지하기 위한 SLA 위반 탐지, DDoS 와 같은 네트워크 공격 탐지, 그리고 물리 서버 및 VNF/CNF 의 고장 탐지로 이루어진다. 이상 탐지 모듈은 실시간 데이터를 바탕으로 각 목표에 따른 탐지 결과를 도출하고 NFV 환경을 관리하는 NFV MANO 모듈의 Policy Manager 에게 이를 알리면 Policy Manager 는 탐지 결과를 기반으로 가상 네트워크 관리를 위한 새로운 관리 정책을 생성하고, 이를 Orchestrator 에게 전달하여 정책을 실제 네트워크에 반영한다. 이 때 Orchestrator 는 필요한 경우 auto-scaling 혹은 migration 과 같은 다른 모듈들과 연계하여 가상 네트워크를 관리한다.

이와 같은 이상 탐지 시스템을 구현하기 위해서는 이상 탐지 모듈에 사용되는 기계학습 기반의 모델을 각 탐지 목표에 따라 학습시켜야 한다. 먼저, SLA 위반 탐지는 선행 연구 [2]가 가지는 scalability 문제를 해결하기 위해 네트워크 토폴로지까지 학습이 가능한 Graph Neural Network (GNN) 알고리즘을 사용하여 SFC 의 길이 및 VNF/CNF 종류가 동적으로 변화하는 환경에서도 탐지가 가능하도록 개선한다. 다음으로 네트워크 공격 탐지의 경우, 최근 DDoS 를 비롯한 네트워크 공격 탐지 연구에 많이 사용되는 Recurrent Neural Network (RNN) 또는 decision tree 계열의 알고리즘을 학습시킨다. 마지막으로 물리 서버 및 VNF/CNF 의 고장 탐지 모델은 자원 사용량 및 syslog 와 같은 로그 데이터를 종합적으로 분석하여 고장 발생 여부를 판단한다. 이 때 로그 데이터 분석은 먼저 unstructured 로그 데이터를 파싱을 통해

structured 로그 데이터로 변환하고, 이를 Convolutional Neural Network (CNN) 또는 RNN 계열의 알고리즘으로 학습시켜 고장 등의 이상 상태를 판단한다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 가상 네트워크 관리를 위한 기계학습 기반의 이상 탐지 시스템을 제안하였다. 제안하는 시스템은 선행 연구[2]를 개선하여 기능 확장과 성능 개선을 목표로 가상 네트워크 환경에서 수집되는 데이터를 통해 기계학습 알고리즘을 각 탐지 목표에 맞게 학습시키고, 학습된 모델을 바탕으로 물리 및 가상 서버의 고장 또는 과부하, 서비스 품질 저하와 같은 이상 징후를 탐지한다. 이를 통해 서비스 또는 시스템의 심각한 장애가 발생하기 전에 네트워크 관리자 또는 [5]와 같은 자율 관리 시스템에 알림으로써 선제적인 가상 네트워크 관리에 활용할 수 있다.

향후 연구로 제안하는 설계를 구현하기 위해 NFV 환경의 모니터링 데이터를 수집한다. 그 후 기계학습 알고리즘을 학습시키고 성능을 비교하여 각 탐지 목표에 맞는 최적 모델을 도출한다. 마지막으로 도출된 모델을 바탕으로 이상 탐지 시스템을 구현하고, 실제 환경에서의 실험을 통해 제안하는 이상 탐지 시스템의 성능을 검증한다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)과 2021 년도 정부(산업통상자원부)의 재원으로 산업기술평가관리원의 지원을 받아 수행된 연구임 (No.2009633, 초저지연 네트워크 서비스를 위한 SDN 기반 인공지능 관제 시스템 개발).

참 고 문 헌

[1] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," International Journal of Computer Applications, vol. 55, no. 3, pp. 38-42, Oct. 2012.

[2] J. Hong, S. Park, J. H. Yoo and J. W. K. Hong, "Machine Learning based SLA-Aware VNF Anomaly Detection for Virtual Network Management," In 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2020, pp. 1-7.

[3] C. Sauvanaud, K. Lazri, M. Kaaniche, and K. Kanoun, "Anomaly detection and root cause localization in virtual network functions," In 2016 IEEE 27th International Symposium on Software Reliability Engineering, pp. 196-206, Oct. 2016.

[4] M. Kourtis, G. Xilouris, G. Gardikis and I. Koutras, "Statistical-based anomaly detection for NFV services," In 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 2016, pp. 161-166.

[5] S. Lange, et al., "A Network Intelligence Architecture for Efficient VNF Lifecycle Management", IEEE Transactions on Network and Service Management (TNSM), August 2020.

비지도 학습 기반 딥러닝 모델을 활용한 가상 네트워크 비정상 행위 탐지에 관한 연구

이청준†, 홍지범§, 최희열†
†한동대학교, §포항공과대학교

chunglee3224@gmail.com, hosewq@postech.ac.kr, heeyoul@gmail.com

Unsupervised learning for anomaly detection in virtual network environment

Chungjun Lee†, Jibeom Hong§, Heeyoul Choi†
†Handong Global Univ., §Pohang Univ. of Science and Technology

Abstract

Software Defined Networking (SDN) and Network Function Virtualization (NFV) have increased flexibility and efficiency in management of service in computer network. However, such development also raised difficulty and complexity of management of computer network and demanded new techniques for an effective management. Anomaly detection is one of such techniques to raise alarm when given service is in abnormal status. There have been many previous works which applied deep learning-based unsupervised anomaly detection in many other domains, but they have not been applied to SFC in virtual network environment. In this paper, we applied deep learning-based auto-encoder, auto-regressor models and thresholding methods to anomaly detection dataset collected from SFC in simulated network environment and achieved detection performance of 78.7% in f1-measure.

I. INTRODUCTION

Softwarization of computer networks led to more flexible utilization of network infrastructures [1]. However, it also led to growth in complexity of management and increase in difficulty for human engineers [2]. Consequently, there have been increasing attention to applying recent development of deep learning models for automatic control and management of softwarized computer network [2]. Especially, anomaly detection is one of important virtual network management techniques which enables quick response to possible quality degradation [2].

There are prior works that addressed anomaly detection problem in different domains using deep learning [3], [4], [5], [6], [7]. The proposed solutions are based on unsupervised learning because of ease for data collection, and difficulty in artificial creation of abnormality. However, the solutions have not been applied to service function chain (SFC) in virtual network environment.

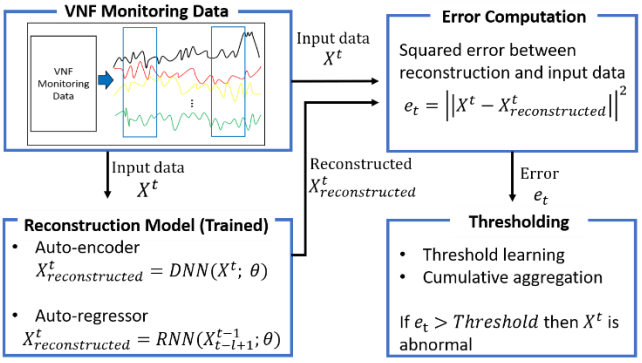


Figure 1: Overview of unsupervised anomaly detection method in virtual network environment.

In this paper, we apply deep learning-based unsupervised learning approach to anomaly detection based on monitoring data of SFC in virtual network environment. Contribution of this paper is to report potential detection performance of unsupervised learning algorithm in such setting. As shown in figure 1, we experiment with two reconstruction models and two thresholding methods to evaluate performances. In result, we observed anomaly detection performance of 78.7% in f1-score without dependence on labelling data.

Ⅱ. BODY

The problem is classifying the status of SFC at a given time. As shown in figure 2, monitoring data is given as X^t where $X^t \in \mathbb{R}^{V \times D}$, and V, D are number of virtual network function (VNF) instances in SFC and number of monitored metrics respectively. t is superscript indicating time index. Constraint is that in training time, we do not have monitoring data corresponding to abnormality. Detection performance is measured by test dataset composed of data and label tuples, (X^t, y^t) where y^t are binary label, and $t \in [1, \dots, N_{testset}]$ which are collected from simulation environment.

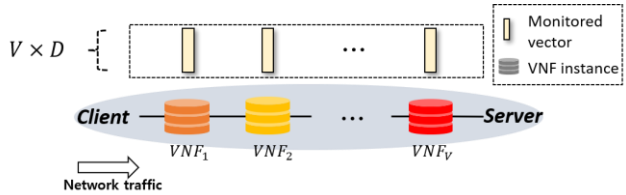


Figure 2: Collection of monitoring data from SFC.

Our unsupervised learning approach follows [5], [6] and [7] where they classify abnormal input data based on deviance from training data. Our proposed method follows 3 steps: (1) learn reconstruction model, (2) compute error and (3) apply threshold. These 3 steps are illustrated in figure 1. We train reconstruction models to accurately reconstruct input data by minimizing the error e_t in equation (1).

$$e_t = \left|\left|X^t - X^t_{reconstructed}\right|\right|^2 \quad (1)$$

The error e_t increases when input data deviates from normal. Reconstruction models are deep learning models like auto-encoder or auto-regressor.

Auto-encoder is deep neural network (DNN) and computes $X^t_{reconstructed}$ based on X^t as in equation (2). In the equation, θ is trainable parameters of the neural network. Auto-encoder learns a function which projects input data X^t into a low dimensional vector and then projects the low dimensional vector to $X^t_{reconstructed}$. On the other hand, auto-regressor is recurrent neural network (RNN), and it learns sequential patterns in training data to compute $X^t_{reconstruected}$ based on previous data sequence as in equation (3). In the equation, l is length of sequence.

$$X^t_{reconstruected} = DNN(X^t;\, \theta), \quad (2)$$

$$X^t_{reconstruected} = RNN(X^{t-1}_{t-l+1};\, \theta). \quad (3)$$

We train the reconstruction models using the following objective function via the gradient descent method. In case of auto-regressor, X^t on left-hand side is replaced by X^{t-1}_{t-l+1} whose subscript indicate start index.

$$J(X^t;\, \theta) = \left|\left|X^t - X^t_{reconstructed}\right|\right|^2. \quad (4)$$

Thresholding is a process of determining normal or abnormal when an error value is computed. The first method is threshold learning where a set of threshold values are evaluated against small portion of labeled dataset to select the best threshold value [7]. The second method is cumulative aggregation which computes a threshold value using mean and standard deviation of error values from the training data [5]. When the computed error e_t exceeds the threshold, it is classified as an abnormal case.

Anomaly detection data was collected from a testbed that simulates SFC in different traffic scenarios [1]. Each instance of VNF in SFC provides various network management functions to traffic. Specifically, WSD dataset is collected from testbed that simulated web service scenario setting whereas LAD1 and LAD2 datasets are collected from that for login authentication scenario setting. Features consist of 23 OS-monitoring metrics from each VNF node that composes SFC. Each dataset has a different number of VNFs as well as

different types of VNFs, and their statistics are in table 1. For test and valid2 sets, they include cases which simulate occurrence of anomalies. The labelling of the anomalous data is based on service level agreement (SLA) standard. The monitoring data collected at the time when response time (less than 0.5 seconds) or availability (over 99.95% success of requests) of service was not satisfied are labeled as abnormal.

Dataset	WSD	LAD1	LAD2
# of VNFs	5	4	4
Total samples	68,731	121,053	121,053
Anomalies	26,354	19,913	44,513
Training set	27,451	65,529	59,470
Valid1 set	3,050	7,281	5,507
Valid2 set	6,496 (2,108)	9,683 (1,593)	9,680 (3,561)
Testing set	13,742 (5,270)	24,206 (3,982)	24,201 (8,902)

Table 1: Statistics of anomaly detection datasets.

Model	WSD	LAD1	LAD2	Average
Auto-encoder	**0.867**	**0.769**	0.725	**0.787**
Auto-regressor	0.844	0.684	**0.755**	0.761

Table 2: f1-measure of the reconstruction models using the threshold learning method.

Model	WSD	LAD1	LAD2	Average
Auto-encoder	0.848	0.635	0.717	0.733
Auto-regressor	**0.824**	0.645	**0.756**	**0.742**

Table 3: f1-measure of the reconstruction models using the cumulative aggregation method.

For experiment, the goal was to measure the anomaly detection performance of the proposed methods: auto-encoder and auto-regressor models. For thresholding, we experimented threshold learning and cumulative aggregation. Experiment results are shown in table 2 and 3. We observed that the threshold is more optimized using threshold learning where auto-encoder and auto-regressor show 78.7 and 76,1% performance in three datasets on average, while the cumulative aggregation shows 73.3 and 74.2%, respectively.

Ⅲ. CONCLUSION

In this paper, we experimented and showed that unsupervised learning models detects anomalies for SFC in virtual network environment without dependence on label information. Although supervised learning model shows relatively higher performance [1], its development is heavily dependent on label information. Future direction is to apply recent development of deep learning-based anomaly detection models to improve detection performance of unsupervised learning models.

ACKNOWLEDGMENT

This research was supported by the Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. 2018-0-00749, Development of virtual network management technology based on artificial intelligence).

REFERENCES

[1] J. Hong, S. Park, J. H. Yoo, and J. W. K. Hong, "Machine learning based sla-aware vnf anomaly detection for virtual network management," in 2020 16th International Conference on Network and Service Management (CNSM), 2020, pp. 1–7.

[2] S. Nedelkoski, J. Cardoso, and O. Kao, "Anomaly detection from system tracing data using multimodal deep learning," in 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 2019, pp. 179–186.

[3] A. Gulenko, F. Schmidt, A. Acker, M. Wallschlager, O. Kao, and F. Liu, "Detecting anomalous behavior of black-box services modeled with distance-based online clustering," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 912–915.

[4] F. Schmidt, F. Suri-Payer, A. Gulenko, M. Wallschlager, A. Acker, and O. Kao, "Unsupervised anomaly event detection for cloud monitoring using online arima," in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 71–76.

[5] F. Schmidt, A. Gulenko, M. Wallschlager, A. Acker, V. Hennig, F. Liu, and O. Kao, "Iftm - unsupervised anomaly detection for virtualized network function services," in 2018 IEEE International Conference on Web Services (ICWS), 2018, pp. 187–194.

[6] Malhotra, P., L. Vig, G. Shroff and Puneet Agarwal. "Long Short Term Memory Networks for Anomaly Detection in Time Series." ESANN (2015).

[7] Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., & Chawla, N.V. (2019). A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. ArXiv, abs/1811.08055.

자연어 처리를 이용한 네트워크 트래픽 이상 탐지 기법

김가영, 엄익채*

전남대학교 정보보안협동과정(대학원생), *전남대학교 시스템보안연구센터(교수)

rkdud8727@gmail.com, *iceuom@chonnam.ac.kr

Anomaly Detection in Network Traffic Using NLP

Kim Ga Yeong, Euom Ieck Chae*

Interdisciplinary Program of Information Security, Chonnam National University (Master's student)

*System Security Research Center, Chonnam National University (Professor)

요 약

최근 사이버 데이터의 양이 계속 증가함에 따라 보안 실무자들은 네트워크 보안을 보장하기 위해 분석해야 하는 데이터양이 증가하고 있다. 또한 새로운 유형이 공격이 지속적으로 생성되고 발견되고 있다. 이를 효과적으로 처리하기 위해 SIEM 같은 전용 보안 솔루션에 인공지능을 활용하는 사례가 늘어나고 있다. 기본적인 머신러닝과 딥러닝을 이용한 연구는 지속되고 있다. 본 논문은 자연어 처리 기법 모델인 BERT를 네트워크 트래픽 이상 탐지 기법에 적용하여 패킷을 토큰화시키고 많은 양의 데이터를 효율적으로 처리 후 이상 탐지까지의 과정을 소개하고 발전 방향을 소개하고자 한다.

Ⅰ. 서 론

최근 보안 솔루션에 인공지능을 활용하는 사례가 늘어나고 있다. 그중 하나인 보안 로그와 이벤트 분석을 처리하는 기존 SIEM 같은 전용 솔루션에 인공지능을 적용하여 오탐을 낮추는 방법을 많은 보안 솔루션 기업들이 고려하고 있다.

사이버 보안 로그는 조직 전체에서 생성되며 엔드 포인트(컴퓨터, 랩탑, 서버), 네트워크 통신 및 주변 장치(VPN노드, 방화벽)를 포함한다. 사이버 데이터 양이 계속 증가함에 따라 사이버 네트워크 보안 기업은 네트워크 보안을 보장하기 위해 분석해야 하는 데이터양이 기하급수적으로 증가하고 있다. 회사 내의 장치가 1000대인 회사를 기준으로 발생하는 로그 추정치를 계산하면 최대 EPS(Event Per Second)가 22,000 이상이며 로그 트래픽은 하루 100GB 이상이 생성된다. 따라서 로그 분석을 하기 위해 수집하는 단계부터 보다 더 유연한 방법이 필요하다.[1]

기본적으로 로그 파일은 이상 감지, 성능 모델링, 네트워크 장애 진단 등에 사용할 수 있는 정보를 포함하고 있다. 로그 파싱은 로그의 소스, 형식 및 시간의 형태가 다르기 때문에 데이터를 분석하려면 필요한 과정이다. 하지만 새로운 로그 형식이 도입되거나 저하된 로그(degraded logs)에 대해서는 새로운 대책이 필요하다.

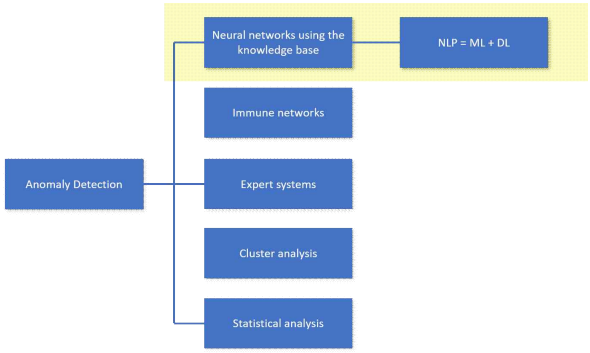
기존 네트워크 이상 탐지 방법은 포트 검사에 의존하는 메커니즘에서부터 출발하여 최근 머신 러닝을 활용하는 방향으로 나아가고 있다. 다수의 연구에서는 라벨링이 필요 없는 비지도 학습의 알고리즘인 K-means를 사용하는 동향을 보이지만 최근 주목받는 기술은 자연어 처리와의 융합이다. 자연어 처리 NLP(Natural Language Processing)는 원래 텍스트 번역, 대화형 사용자 인터페이스, 텍스트 분류, 맞춤법 검사 등으로 활용된다. 자연어 처리의 딥러닝 모델은 트랜스포머(Transformer), BERT(Bidirectional Encoder Representations from Transformers), GPT-3 등이 존재한다. 트랜스포머는 문장 전체의 중요성을 모두 분석하는 대신 중요한 문장만을 집중해서 분석하는 방식인 어텐션 메커니즘(attention mechanism)만을 활용해 자연어를 처리하는 방식인 셀프 어텐션(self-attention)을 사용하였다. 이후 2018년에 개발된 BERT는 셀프 어텐션을 기반으로 만들어졌으며 문장을 분석할 때, '앞에서 뒤', '뒤에서 앞' 양방향으로 분석한다는 특징이 있다. 네트워크 패킷의 구조를 자연어 처리 과정인 토큰화를 통해 데이터를 정제한 후 학습하면 보안 로그 파싱 속도와 효율을 높일 수 있다.

본 논문에서는 인공지능을 활용하여 네트워크 패킷 이상 탐지를 적용한 관련 연구에 대해 설명하고, 머신 러닝을 이용하여 네트워크 트래픽 이상 탐지를 한 연구를 2장에서 비교 분석한 후 3장에서 자연어 처리 기법을 활용하여 이상 탐지를 한 연구에 대해 알아보고 개선점을 제시한다.

Ⅱ. 관련 연구

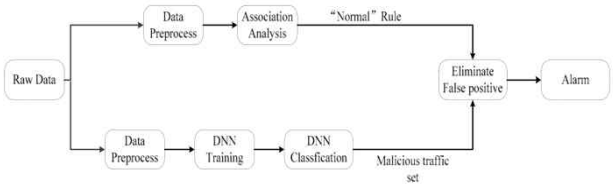
네트워크 트래픽 분석을 위한 이상 탐지 방법에 사용되는 알고리즘은 일반적으로 세 가지로 구분할 수 있다.[2]

첫 번째, 비지도 이상 탐지 기술은 미리 라벨링 된 데이터가 필요하지 않으므로 많이 사용되는 기술이다.[2] K-NN(K-Nearest Neighbor) 기반 기술, 클러스터링 기반 기술, 통계 알고리즘, 하위 공간(subspace) 기반 기술이 존재한다. 두 번째, 지도 이상 탐지 기술은 정상과 비정상 두 가지 클래스로 구성되며, 이후에 새로운 데이터는 두 클래스와 비교하여 해당 데이터가 속한 클래스를 예측한다. 마지막은 준지도 이상 탐지 기술로 특징은 정상적인 행동의 모델만 구성하여 학습하기 때문에 "정상" 클래스만 사용된다.



〔그림 1〕 이상 탐지 기법의 방법

Minghui Gao 등은 딥러닝의 DNN-4 알고리즘을 사용한 이상 탐지 시스템을 연구하였다.[3] 이 연구에서는 DNN을 적용한 이상 자동 탐지 기술은 큰 성과를 거두었지만 여전히 오분류된 트래픽이 존재한다는 것을 문제 제기하였다. 공개 데이터 셋인 NSL-KDD을 대상으로 여러 알고리즘을 적용한 결과 가장 높은 정밀도와 정확도를 보인 DNN-4를 선택하였다. 또한, 오분류를 줄이기 위해 DNN과 연관 분석을 기반으로 그림 2에 표현된 2단계 이상 탐지 시스템을 설계하였다.



〔그림 2〕 2단계 이상 탐지 시스템

분석 순서는 전처리, 학습 및 학습 모델을 이용한 분류, 연관 규칙 분석 순서로 진행된다. 연관 규칙 분석에는 Apiori 알고리즘이 사용되었으며 이는 서로 다른 두 데이터가 얼마만큼 빈번히 발생하였는지에 대한 연관도를 알 수 있다. 이를 이용해 이산화된 특징과 데이터 셋에 숨겨진 정상 레이블 간의 규칙을 찾고 이후 악성 트래픽으로 잘못 분류된 정상 트래픽을 필터링한다.

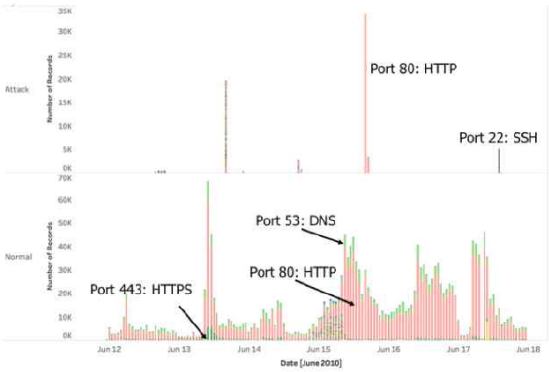
Benjamin 등은 RNN(Recurrent Neural Networks)의 LSTM(Long Short-Term Memory)을 사용한 이상 탐지 시스템을 연구했다.[4] 이 연구는 규칙 기반의 접근 방식은 이미 알려진 패턴만 식별한다는 점을 문제점으로 삼았으며 악의적인 동작 패턴에 의존하지 않아야 이전에 보지 못한 패턴을 탐지할 수 있다고 말한다. 네트워크 트래픽을 하나의 컴퓨터 간의 대화를 나타내는 문장으로 보며 문장을 형성하는 단어를 토큰화하였다. 이 연구는 자연어 처리를 결합한 연구로 문장을 사용하여 새로 생성된 언어의 의미 및 구문 문법을 학습하는 언어 모델을 생성한다. 이후 언어 모델을 이용하여 두 IP간의 통신을 예측하는데 사용되며 관찰된 통신이 일반적인지 비정형적인지를 판단하여 예측을 하게 된다.

Nathanael 등은 소셜 미디어의 텍스트를 이용하여 DoS(Denial-of-Service) 공격을 탐지하기 위한 NLP 모델을 연구하였다.[5] 다수의 네트워크에서 광범위하게 발생한 공격에 대해 네트워크 서비스 상태를 간접적으로 측정하기 위해 소셜 미디어를 활용한 NLP 연구이다. 이 연구는 이산 자료들에 대한 확률적 생성 모델인 LDA(Latent Dirichlet Allocation)을 사용하였다. LDA는 주어진 텍스트에 대해 각 텍스트가 어떤 주제들을 서술하는지에 대한 확률적 토픽 모델 기법 중 하나이다. 미리 알고 있는 주제별 단어 수 분포를 바탕으로 주어진 텍스트에서 발견된 단어 수 분포를 분석함으로써 해당 텍스트가 어떤 주제들을 함께 다루고 있을지를 예측할 수 있다. 연구에서는 두 가지의 목표를 다루고 있는데 공격과 비공격의 자동 분류와 공격이 진화하는 동안 사용자 행동 분석을 할 수 있는 모델이다. 각 보안 커뮤니티에서 공격 주제를 설정하여 학습 모델을 통해 해당 공격에 대한 자동 분류를 실행한다.

Ⅲ. NLP를 이용한 네트워크 트래픽 이상 탐지

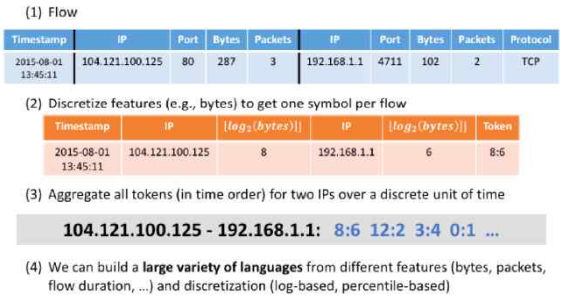
사이버 데이터의 양이 계속 증가함에 따라 보안 실무자들은 네트워크 보안을 보장하기 위해 분석해야 하는 데이터양이 증가하고 있다. 또한 전 세계적으로 새로운 유형의 공격이 지속적으로 생성되고 발견되고 있다. 규칙 기반의 접근 방식은 알려진 공격을 특성화하고 플래그를 지정하는데 효과적이지만 일반적으로 새로운 공격이나 유형의 데이터는 효과적이지 않다. 본 절에는 네트워크 트래픽 이상 탐지에 적용되는 자연어 처리 기반의 기법을 설명한다.

이 연구[6]에서는 기존 컴퓨팅 환경에서 수집된 네트워크 트래픽 데이터인 ISCX(Intrustion Detection Evaluation Dataset)[7]를 이용했다. ISCX 데이터 셋은 네트워크 침입 감지에서 이상 기반 접근 방식은 특히 적절한 데이터 셋의 부족에서 비롯되는 정확한 평가, 비교, 배포에 어려움을 겪는다. 또한, 많은 데이터 셋은 내부 데이터이며 개인 정보 보호 문제로 인해 공유를 할 수 없다는 점을 문제로 보고 실제 테스트 베드 환경을 구성하여 만든 오픈 데이터 셋이다. ISCX 데이터 셋은 그림 3과 같이 2,028,053개의 netflow 레코드를 포함하며 96.6%는 정상이고 나머지3.4%는 공격을 나타낸다.



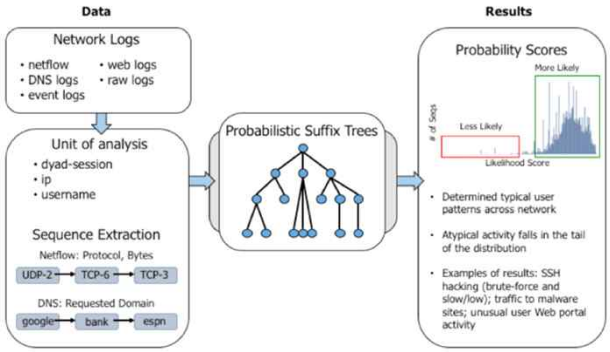
〔그림 3〕 ISCX 데이터 셋의 구성 공격 (위)/정상 (아래)

분석을 하기 전에 네트워크 패킷의 토큰 시퀀스로의 형식 변환이 필요하다. 프로토콜, 포트, 바이트, 패킷 및 기타 기능의 다양한 조합을 개별 토큰으로 인코딩 할 수 있으며 이러한 토큰 시퀀스는 네트워크로 연결된 두 컴퓨터 간의 통신을 효과적으로 압축한다. 변환은 전송된 바이트의 합을 나타내는 프로토콜 식별자인proto-bytes와 전송된 바이트/패킷의 합을 나타내는 프로토콜 식별자인 proto-density을 사용한다. 크기를 조절하기 위해 log2를 취한다. 이를 통해 생성되는 토큰 수를 줄이면서 크기(바이트, KB, MB, GB)를 유지한다.

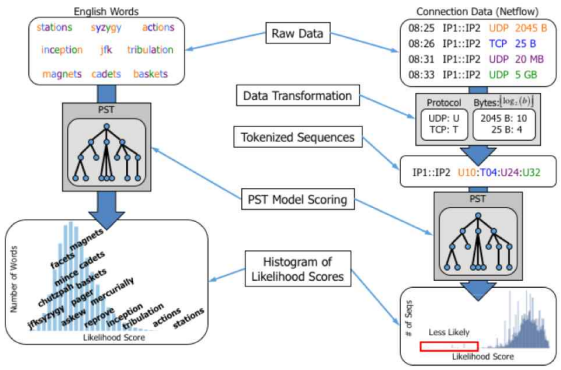


〔그림 4〕 네트워크 패킷 시퀀스 구조

토큰 시퀀스로의 형식 변환 후 시퀀스 인덱싱 기술 중 하나인 예측 모델 PGST(Probabilistic Generalized Suffix Tree) 모델을 만든다. PGST는 PST의 변형이며 작은 시퀀스로 구성된 큰 용량의 접미사 트리이다. 시퀀스와 PST 모델을 만든 후 각 시퀀스에 점수를 매기는 작업을 한다. 그림 5와 같이 각 시퀀스는 0과 1사이의 확률 점수를 받으며 0이 아니며 설정된 점수 사이에 있는 시퀀스는 플래그를 지정한다. 플래그가 지정된 시퀀스는 데이터에 존재할 가능성이 낮은 시퀀스를 말하며 비정상 시퀀스를 의미한다.



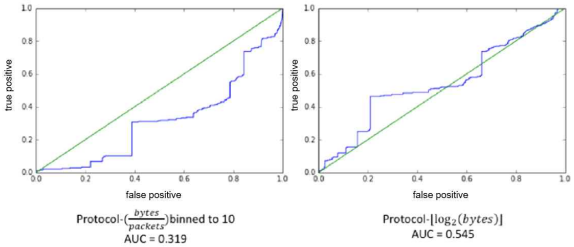
〔그림 5〕 네트워크 트래픽 데이터에서 PGST 모델 생성을 위한 분석



〔그림 6〕 PST 모델링을 적용한 경우
일반 영어 단어(왼쪽) / 네트워크 트래픽 데이터 (오른쪽)

그림 6을 보면 일반 영어 단어에 PST 모델링을 적용한 경우와 네트워크 트래픽 단어에 PST모델링을 적용한 경우의 차이점을 볼 수 있다. 왼쪽 일반 영어 단어를 보면 actions이나 stations 같은 단어는 히스토그램을 보면 오른쪽에 위치한 것을 확인할 수 있으며 chutzpah과 syzygy와 같이 생소한 단어는 히스토그램의 왼쪽에 위치한 것을 확인할 수 있다. 네트워크 패킷에도 똑같이 적용하여 영어 철자 패턴을 분석하는 과정 대신 통신을 나타내는 토큰화된 시퀀스의 철자 패턴을 정량화한다. chutzpah와 syzygy와 같이 히스토그램의 왼쪽에 위치하고 생소한 단어처럼 나타날 가능성이 낮은 시퀀스가 관찰되는 네트워크 환경은 비정상적인 이벤트가 발생했다고 예측할 수 있다.

또한, PST 모델링을 적용하여 성능을 높이는 방법은 하이퍼 파라미터의 조절이다. 모델의 최대 깊이를 설정하는 트리 깊이와 시퀀스의 우선순위 설정과 후보를 추가하는 확률 임계값, 두 개의 평탄화(smoothing)을 위한 파라미터를 조절하여 성능을 높일 수 있다.



〔그림 7〕 하이퍼 파라미터 조정 전 (왼쪽) / 조정 후 (오른쪽)

Ⅳ. 결론

본 논문에서는 자연어 처리를 통해 네트워크 패킷의 이상 탐지를 하는 방법을 알아보았다. 네트워크 패킷을 토큰화하여 각 시퀀스로 전처리 과정을 거친 후 일반 단어에 적용하는 기법을 적용하였다. 이를 통해 기존 데이터와의 유사도를 비교하고 유사도가 낮은 데이터를 비정상 시퀀스로 분류하였다. 또한 하이터 파라미터를 조절하여 기본 성능보다 향상시키는 과정을 소개했다.

향후에는 PST 모델링 기법 외에 다양한 모델을 적용하여 이상 탐지를 할 예정이며, 기존 PST 모델링보다 더 나은 성능이 나올 수 있도록 연구할 예정이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(NO, 2019-0-01343, 융합보안핵심인재양성)

참 고 문 헌

[1] Bartley Richardson. “Neural network, that’s the tech; To free your staff from, bad regex”, Dec. 2019,
(https://medium.com/rapids-ai/cybert-28b35a4c81c4#cid=av01_so-twit_en-us).

[2] Wenguang Song, et al. “ A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection”, pp. 3-4, Mar. 2020.

[3] Minghui Gao, et al. “ Malicious Network Traffic Detection Based on Deep Neural Networks and Association Analysis”, Mar. 2020.

[4] Benjamin J. Radford, et al. “ Network Traffic Anomaly Detection

Using Recurrent Neural Networks", Mar. 2018.

[5] Nathanael Chambers, et al. " Detecting Denial-of-Service Attacks from Social Media Text: Applying NLP to Computer Security", Jun. 2018.

[6] Bartley D. Richardson, et al. " Anomaly Detection in Cyber Network Data Using a Cyber Language Approach", Aug. 2018.

머신러닝 분석을 위한 데이터 모델을 통한 데이터 프로비저닝

왈리드 아크바르, 아팍 모하마드, 칸 탈하 애흐마드, 송 왕 철
{waleedwali786, afaq24, talhajadun}@gmail.com, philo@jejunu.ac.kr

Data Provisioning Through Data Models for Machine Learning Analysis

Waleed Akbar, Afaq Muhammad, Talha Ahmed Khan, Wang-Cheol Song
Department of Computer Engineering, Jeju National University

Abstract

Deployment of virtualized infrastructure is significant importance when managing overly complex and demanding network requirements, as in the telecom sector. Due to the recent increase in high data rate applications, network management and maintenance have become more critical and challenging. SDN (Software-Defined Network) is capable of managing these complex networks due to the integrated virtualized technology. SDN has a central control plane that provides APIs (Application Programming Interfaces) to integrate different applications that react according to the runtime environment. To monitor SDN-based network and take advantage of real-time data, it is essential to analyze generated data and alter network configurations according to different scenarios. In this paper, we have proposed a real-time SDN monitoring system. It monitors the physical resources of deployed hosts and network traffic on links between virtual switches. We deploy agents on each host machine and configure them with a central database to collect and store real-time resource monitoring data. Moreover, we connect Grafana with the database to extract and visualize data. Therefore, we can analyze network traffic behavior and appropriately choose the machine learning model for network performance tuning and optimization.

Ⅰ. Introduction

There is an enormous increase in data generation due to the large-scale deployment of IoT (Internet of Things) devices. This massive amount of data needs to be managed, analyzed, and routed intelligently; otherwise, data will negatively impact network performance [9]. Traditional network devices cannot cope with such massive data. Due to the diversified nature of the network, it is difficult to consistently update the control plane. SDI (Software Defined Infrastructure) has the potential to succeed in this sort of situation. It has a centralized control plane and distributed data plane. Furthermore, another critical issue is to monitor these complex networks and to understand the monitored data. A complete monitoring system assists network administrators in understanding the network behavior better. Therefore, we proposed any open-source softwarized network monitoring system that captures network resource and physical resource utilization data. The monitoring system comprises visualization tools, databases, active agents, and controllers.

This manuscript is organized as follows. Section II describes the related work and motivation, Section III details the real-time monitoring system. Section IV explains the data models, Section V shows the machine learning scenarios and section VI concludes the manuscript.

Ⅱ. Motivation and Related Work

SDN allows deployment of user define polices on the fly. It changes network configurations to handle the traffic and optimally utilized network resources [4]. Futhermore, SDN facilitates deployment of complex policies such as orchestration, slicing and life cycle management. SDN switches are capable of measuring traffic flow and statistics either with push-based [2] or pull-based architectures [3]. The agents are deployed inside controller and switches to monitor the network statistics. The group of switches are selected using greedy approach and they transfer packet thrice before measuring RTT [1].

SDN allows deployment of user define polices on the fly. It changes network configurations to handle the traffic and optimally utilized network resources [4]. Futhermore, SDN facilitates deployment of complex policies such as

orchestration, slicing and life cycle management. SDN switches are capable of measuring traffic flow and statistics either with push-based [2] or pull-based architectures [3]. The agents are deployed inside controller and switches to monitor the network statistics. The group of switches are selected using greedy approach and they transfer packet thrice before measuring RTT [1].

III. System Design

The figure 1 depicted the architecture of integration of our monitoring system with KOREN testbed. The architecture has four main components, visualization, data storage, controller and agent. Grafana allows users to query, visualize and generate alerts「5」. We have deployed Grafana on local PC in our lab and configured it with Prometheus database. It pulls the data via HTTP from multiple nodes (agents) into a single database「6」. We have installed Prometheus on our lab PC and configure it with Node Exporter agents. It collects the data form deployed agent on KOREN hosts at remote location. Node Exporter is an open-source tool, it is used to expose the hardware level, and service-level metrics「7」. The agents run on endpoints in each KOREN hosts, these endpoints are configured with Prometheus to collect the metrics such as CPU utilization, memory utilization, and many other metrics. By default, the metrics are collected periodically after every 10 seconds. Controllers are part of KOREN testbed, they are used to perform CURD operation on switches「8」. Real-time link traffic statistics are collected using BEAM controller APIs. BEEM is a central controller with a view of complete virtualized network. Many controllers are available.

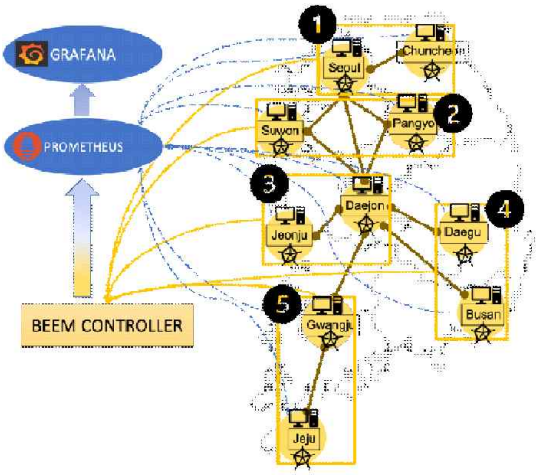


Figure 1: An overall view of monitoring system with controller, Grafana, prometheus and agents

IV. Data Model

Our monitoring system can generate different types of data models such as link traffic statistical data, KOREN host resources utilization data.

Link traffic statistical data is accumulated using main controller, with five sub-controllers. Every sub-controller monitors the traffic and forward it to main controller. We collect the traffic statistic data and topology information using BEEM controller API s. The collected statistics data attributes are transfer/receive bytes, transfer/receive drop and transfer/receive packets and timestamp value. Figure 2 shows the transfer rate of 598.4 Mb/s, 0 packets drop count, 0% drop rate and 6% link utilization from Daejeon and Busan.



Figure 2 : Links monitoring statistics

The deployed agents on KOREN hosts are monitoring network traffic, and resource utilization metrics. There are two types of collected metrics. The first one is network resource utilization metrics includes bytes send/received per interface, port, and many other metrics. The other is physical and virtual resource metrics includes memory utilization, disk utilization and many other metrics of each physical or virtual machine. Figure 3 shows the metrics collected from different KOREN hosts. As seen in figure; the Suwon has 63% of memory utilization and 3% CPU utilization. Different color line graph represents the network traffic on different logical and/or physical network interfaces.

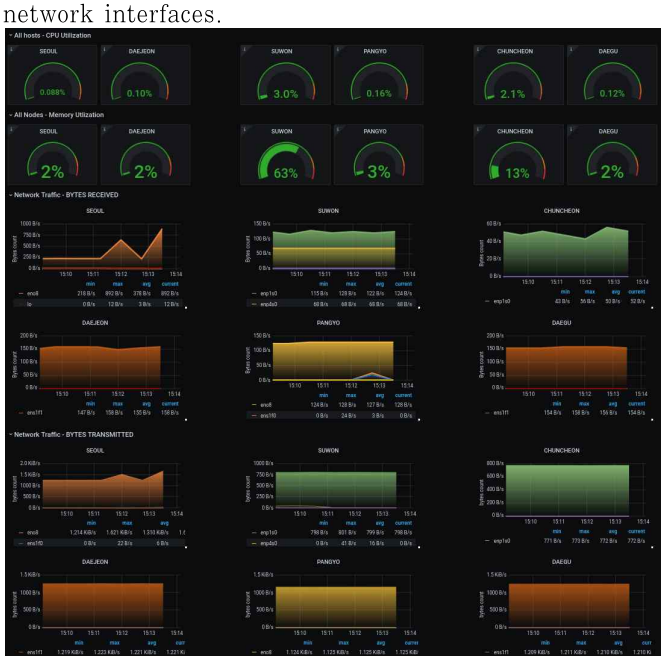


Figure 3 : Real-time monitoring metrics

V. Machine Learning Scenarios

In our scenario, we considered link traffic statistical data model for GNN (Graph Neural Network) and host resource utilization data for LSTM (Long Short-Term Memory).

Due to the availability of topology information in traffic statistical data model. We utilized this data model for GNN training and input topology information as a graph. The dataset is collected for five days and once per minute. The total bytes transferred per minute, total packets transferred per minute, total packets drop per minute, time stamp (60 seconds), routing information are input metrics to model. The model output is link bandwidth utilization for the next timestamp as shown in figure.

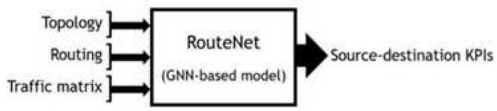


Figure 4: GNN input and output metrics

A real-time data model is time-series data gathered once per minute. LSTM has an inherited structure to work well on sequential data. Therefore, we apply LSTM to a real-time data model. A multi-step bi-LSTM model is used for predictions. The timestamp, bytes received, byte transferred, bits per second, and packets per second are input metrics to model. The model output is link utilization for 10 minutes, 20 minutes, and 60 minutes as shown in figure 5.

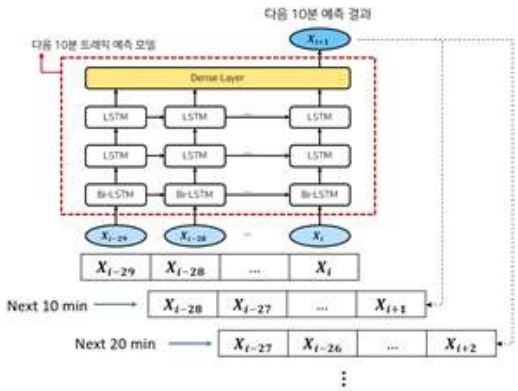


Figure 5 : LSTM Architecture

VI. Conclusion

The rapid growth of data centric application imposed intensive load on traditional network architecture. The monitoring and understanding of data are essential for network automation and optimization. We proposed network monitoring and analytical engine that understands the network behavior and analyze data generation patterns. We further recommend machine learning models according to data generation patterns.

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2017- 0-01633) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B01016322).

REFERENCES

[1] L. Liao, V. C. M. Leung and M. Chen, "An Efficient and Accurate Link Latency Monitoring Method for Low-Latency Software-Defined Networks," in IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 2, pp. 377-391, Feb. 2019, doi: 10.1109/TIM.2018.2849433.

[2] Mao, Yuyi, et al. "A survey on mobile edge computing: The communication perspective." IEEE Communications Surveys & Tutorials 19.4 (2017): 2322-2358.

[3] Tangari, Gioacchino, et al. "Self-adaptive decentralized monitoring in software-defined networks." IEEE Transactions on Network and Service Management 15.4 (2018): 1277-1291.

[4] Abbas, Khizar, et al. "An efficient SDN-based LTE-WiFi spectrum aggregation system for heterogeneous 5G networks." Transactions on Emerging Telecommunications Technologies (2020): e3943.

[5] https://grafana.com/. Accessed on 07-12-2020.

[6] https://prometheus.io/. Accessed on 07-12-2020.

[7] https://github.com/prometheus/node_exporter. Accessed on 07-12-2020.

[8] http://13.209.193.98/beemopenmul/

[9] Rafiq, Adeel, et al. "Intent-based end-to-end network service orchestration system for multi-platforms." Sustainability 12.7 (2020): 2782.

위치 인식 다중 경로 네트워크 프로비저닝

홍남곽, 염성웅, 김경백*

전남대학교 인공지능융합학과

quachhongnam1995@gmail.com, yeomsw0421@gmail.com, kyungbaekkim@jnu.ac.kr

Location-aware Multi-Path Network Provisioning

Hong-Nam Quach, Sungwoong Yeom, Kyungbaek Kim*

Dept of Artificial Intelligence Convergence, Chonnam National University

Abstract

As the number of IoT and mobile devices grows and the concept of 5G technology becomes more mainstream, various network infrastructures are expanding, and demand for customized network services is becoming a hot trend. Also, the demands of request user-specific network services are increasing along with the still restricted network infrastructure. Providing a guaranteed QoS level to users' requirements necessitates taking into account the variety of factors that influence network service efficiency, as well as complex network provisioning. In this paper, we suggest a novel dynamic network provisioning scheme with multi-path planning. In this system, we design a user request processor system that understands user-specific QoS specifications in order to optimize network resource utilization.

Ⅰ. INTRODUCTION

Nowadays, along with network infrastructure and technology development, the demand for personalized networks from users is increasing [1]. Besides, the Covid-19 epidemic broke out, making it impossible for people not to go out, leading to an ever-increasing demand for internet use. The dynamic network provisioning concept is a way to provide flexibility and service customization in existing networks by sharing network resources, which include both physical and logical resources (e.g., computing, storage, etc.) among different service providers. Moreover, location-aware network provisioning allows dynamically generating a subnetwork based on users' requested locations to provide services that match particular needs [2]; users can also input some service and network parameters into their request [3]. However, when many users want to use the network services with the exact location and resources. The ISP needs to calculate for leveraging their resource to provide the several different paths. This helps them ensure available QoS and save the operating expenses.

Recently, pathfinding in a location-aware network has been the subject of many studies in recent years. For example, a method of best pathfinding using Location-aware AODV (Ad-hoc On-demand Distance Vector) for MANET (Mobile Ad-hoc NETwork) is suggested in [4]. This paper used multiple parameters such as node-ID, timestamp, GPS, bandwidth, RTT, packet loss ratio, and others to modify an existing protocol called AODV based on location to find the best path among multi-path routing protocols for MANET. On the other hand, the authors in [5], [6] proposed online algorithms with an auxiliary graph for unicast and multicast requests, including a bandwidth constraint and maximized network throughput. However, these studies do not consider the location-specific information of user requests. For example, when the students send requests to access the online classes in university in their house, the ISPs should determine the requested locations that students would use to make the network their university location, instead of his house's location. This paper suggested a dynamically estimating network switch multi-path for location-aware network provisioning to overcome this problem. This supports the network provider could leverage their resources and save cost operation.

Ⅱ. METHODOLOGY

This section suggested a method for estimating network switch multiple-path based on the user has requested locations. We use our proposed device architecture based on [2] in this paper. After that, we suggest two extensive new features. The web user interface, which is designed to allow several users to

submit their requests simultaneously. Second, we add the Network Path Calculator module, which calculates suitable switch multi-paths to accept as many user requests as possible.

A. *Web application interface for obtaining the location requests*

This section explains the web application, enabling users to send location requests and sending them to a central server for processing, as shown in Fig. 1. A map-based selector-region Web UI is shown in the diagram below. A user may select locations and determine the appropriate QoS level using network parameters such as bandwidth and delay. Then, the location of selected regions, which are expressed with the ID and the parameter of coordinates. These pieces of information are stored in the database and import into the CSV file.

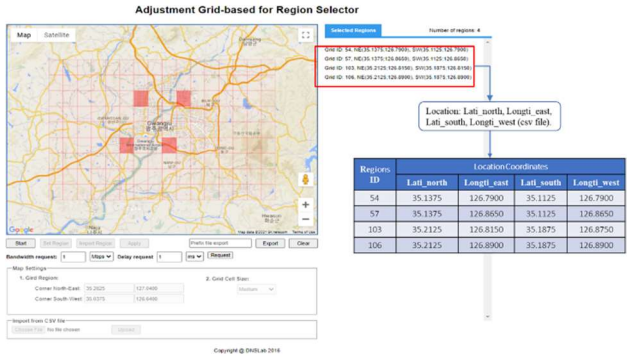


Figure 1: Web-UI with several selected location by a user

B. *Multi-path calculation*

This paper uses an algorithm to identify all network devices that cover the requested locations to create a subnetwork based on the requested locations. The algorithm analyses each area to see if it belongs to a switch's cover field. Finally, the algorithm returns a list of devices to build the subnetwork.

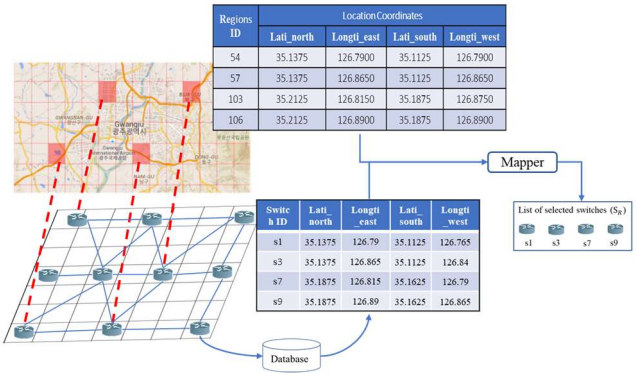


Figure 2: Model of Location-awareness

Users can select locations through a map interface that corresponding with switches connections in the below layer. Two tables contain the ID and necessary coordinates from the user-requested location and covered regions of selected switches.

Algorithm1: Mapping and find the corresponding switches cover selected regions

INPUT: The list of coordinates of selected regions

OUTPUT: List of switches cover selected regions S_{L_R}

Step 1: Get the and extract out the location coordinates

The List location of request: $L_R = \{l_1, l_2, \ldots l_n\}$

l_1 = (lat.ne, lng.ne, lat.sw, lng.sw)

Step 2: Get the information of resource switches in the DB

The resource Switches: $S = \{s_1, s_2, \ldots s_n\}$

s_1 = (reg.lat.ne, reg.lng.ne, reg.lat.sw, reg.lng.sw)

Step 3: Compare:

For each s ∈ S do

For each $l \in L_R$ do

If:

l.lat.ne ≤ s.reg.lat.ne &&

l.lng.ne ≤ s.reg.lng.ne &&

l.lat.sw ≤ s.reg.lat.sw &&

l.lng.sw ≤ s.reg.lng.sw then

S_{L_R}.add(s)

End if

End for

End for

→List of cover switches: S_{L_R}

We created a location-awareness model in which a module uses an algorithm to identify switches with a cover region that contains at least one requested usage location. After that, the Network Path Calculator creates the sub-network topology using the list of switches identified by the location-awareness model and algorithm 2. The procedure "Mapping and finding the corresponding switches cover selected regions" in Algorithm 1 illustrates our deployment for the model of location-awareness. In algorithm 1, two processes: Get and extract locations which return a list of selected locations (Step 1), and get the list switches that return a list of resources switches (Step 2). Next, the algorithm checks and compares if at least one requested position (Area) into the cover region of that switch (Step3) exists for each switch. Finally, the algorithm returns a list of the corresponding switches covering selected regions. This list of cover switches forwarded to the path calculator to find the routes to build a sub-network topology to provide the customers' network services.

We assume that all link connect switches are of the same quality. Many paths to connect all switches in the selected list and make a subnetwork that covers the user has requested locations. The Network Path Calculator deployed the Dijkstra algorithm to find effective routes to create a subnetwork that uses the shortest paths between each intermediate device pair. In other words, it is the active path in the subnetwork. Then, we bring all of the shortest paths together to create a dynamic route used by default.

In the above part, we have designed a system for connecting all of the devices selected. However,

several scenarios in reality that cause lead to the path may not operate effectively, such as one or more connections in this subnetwork are overloaded or malfunctioning connections occur. For example, when so many users request a set of the same switches with different bandwidth, users simultaneously transfer their data in the same subnetwork. This leads links in the subnetwork to become overloading.

This paper deploys the algorithm to find other paths to overcome this issue, which are the alternative routes for the shortest path with similar constraints. We assume the traffic load of the network is set up and define the following:

The traffic load information of each link:

$$T_{(used)} = \sum_{i}^{k} BW_{i_{(required)}}$$

The remaining traffic load information of a link:

$$T_{(remain)} = T_{(original)} - T_{(used)}$$

Where:

$T_{(used)}$: the traffic load was used by users in a link

$BW_{i_{(required)}}$: the bandwidth requirement of the user i^{th}in the all k users send requests

$T_{(original)}$: the traffic load is set up initially on each link

Algorithm 2: Finding the alternative paths

INPUT: Set of the selected switches S_R

OUTPUT: Set of the paths to connect the switches in S_R

Procedure: S_R = {P_{S_R}} pair of switches in S_R

Step 1: Check the current traffic information in network

Step 2: Choose any P_{S_R} in S_R and find the route to connect

Step 3: In P_{S_R}, from the source switch, select the neighbor device.

Step 4: If the link between source and neighbor device have remained traffic.

Compare the remain traffic with bandwidth require.

If $BW_{(required)} < T_{(remain)}$

Add it to expected path

If the link between source and neighbor device have remained traffic.

Compare the remain traffic with bandwidth require.

If $BW_{(required)} < T_{(original)}$

Add it to expected path

Step 5: Repeat the step 4 until to reach the destination.

If not, return no path is found

Step 6: Perform similarly with next other pair in P_{S_R}

Finally, we calculate a set of alternative paths, and after combining these routes, we make a table, which contains the shortest paths for the user.

III. EVALUATED

A. Setting Environment

To conduct the test, we built a virtual machine on VMware with the Operating system Ubuntu 20.04.0.2 LTS. Also, we deploy the framework by using Python2.7, MySQL. To support the framework, we designed and implemented the Web-UI by using socket.io, node.js, and HMTL. We use the editor Visualcode to build the code of the framework and Mininet to build the environment.

Environment	Configuration
Operating system	Ubuntu-20.04.2.0 LTS
Memory	RAM 8GB
Programming language	Python 2.7, NodeJS, HTML
Libary	Python 2.7 Libary
Database	MySQL server
Simulator	Mininet 2.2.0
Protocol	OpenFlow1.3

Table 1: Experimental environment configuration.

In this paper, the experiment is conducted several experiments with various network settings using the mininet to evaluate the proposed method. Then, we measure the number of accepted requests and accumulate bandwidth. In these experiments, we set up switch network size in 13 nodes, and each network link has a bandwidth capacity in the range of 100 ~ 1000 Mbps and 2~5ms delays. Also, we randomly pick 20% of devices in a request and randomly choose the start time and usage duration. The QoS parameter of a request includes bandwidth constraint (1~10Mbps) and delay constraint in range (40ms ~ 200ms).

Performing experiments with the Web-UI helps the customers give the requirements about location, QoS constraints as bandwidth, and delay. The experiments assume the number of requests from users will become in our framework frequently and continuously. We generated requests with the experiments, such as 10, 20, 30, 50, and up to 500 requests to conduct this assume. Based on the location parameters of these will be handled to find the corresponding switches resources. We then used the QoS constraint of these requests to find the shortest path fitting each request. To evaluate our proposed method's performance, we offer the Dijkstra algorithm to compare.

B. Analysis, Evaluate Results

Our proposed method is evaluated with the network with topology size 13 switches, and the number of requests increases up to 100 while other parameters are fixed. Fig.3 plot the performance curves of two different methods from which it can be seen that the proposed method outperforms Dijkstra in all cases.

Specifically, at the very first simulation time, when the number of requests is relatively small (less than 50), the number of accepted requests seems to be the

same

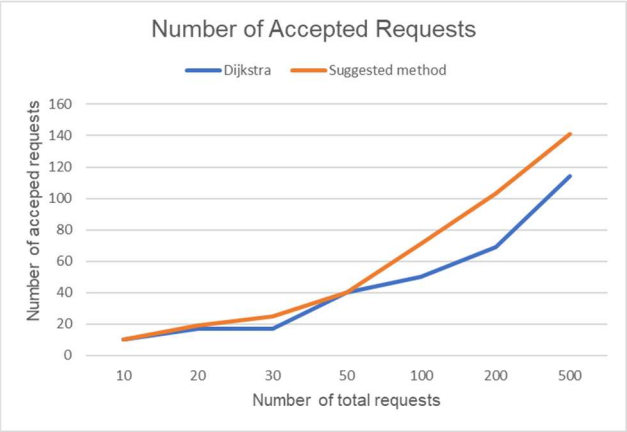


Figure 3: The number of accepted requests

between the two methods. However, as the number of requests grows, our proposed approach accepts a slightly higher number of requests than the other, up to 1.5 times at the end of the simulation. In other words, when more requests come into the system, more requests can be accepted more than the other, as Fig.3. With Dijkstra, when the network resource serves a request, Dijkstra cannot use it for other requests until its duration finishes. However, with the proposed method, the network resource can be shared by several requests within an expected certain amount of time them. Furthermore, the suggested method also finds alternative paths that can fit the user request's QoS constraints

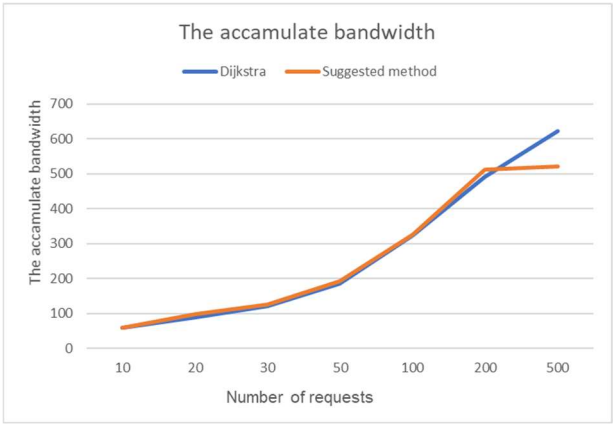


Figure 4: The accamulate bandwidth

Besides, Fig. 4 shows that our proposed approach provides more cumulative bandwidth than the Dijkstra method, despite the difference being very minor due to our simulation network resource constraints. However, we recognize that when the number of requests increases from 200 to 500 at the end of the simulation. The proposed method's cumulative bandwidth decreases more diminutive than the Dijkstra method. The explanation for this is that while the Dijkstra method resource is saturating, the proposed method can accept more requests whose QoS requirements are negligible.

IV. CONCLUSION

In this paper, an approach is proposed that is a framework that includes a web-UI to gather users' requests. The server responsible listens, obtains, and handles the request, which is continuously maintained. This server supports network resources' deployment, finds the routes that respond to user's requests, and creates subnetwork topologies that match the offers' requirements. The server also monitors the network operation to gather, analyze, anticipate, avoid, and respond to network incidents. We are trying to enhance network provisioning performance in the number of accepted requests and QoS constraints and extend the experiment to measure the average bandwidth and delay to evaluate the proposed method's performance.

ACKNOWLEDGMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2016-0-00314) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation). This results was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE).

References

[1] Wellman, Barry. "Physical place and cyberplace: The rise of personalized networking." International journal of urban and regional research 25.2 (2001): 227-252.

[2] Van-Quyet Nguyen, Sinh-Ngoc Nguyen, Deokjai Choi, Kyungbaek Kim. "Location-aware Dynamic Network Provisioning," In Proceedings of the 19th APNOMS, 2017.

[3] Gde Dharma N., Van-Quyet Nguyen, Tiep Vu Duc, Ngoc NguyenSinh, Alvin Prayuda J.D., Kyungbaek Kim, Deokjai Choi "Design of Service Abstraction Model for Enhancing Network Provision in Future Network," In Proceedings of the 18th APNOMS, 2016.

[4] Anagha Raich, Amarsinh Vidhae. "Best Path Finding using Location-aware AODV for MANET," International Journal of Advanced Computer Research Volume 3, Num3, pp.336-340, Sep. 2013.

[5] M.Huang et al., "Dynamic routing for network throughput Maximization in Software-defined networks," in Proceeding of IEEE INFOCOM, USA, 2016, pp. 1-9

[6] Mike Jia, W.Liang, M.Huang, Z.Xu, and Yu Ma, "Routing Cost Minimization and Throughput Maximization of NFV-enabled Unicasting in Software-defined networks," IEEE Transaction and Network service Management, vol. 15, no. 2, pp. 732-745, June 2018.

A distributed, scalable, high-performance, and real-time data collection and processing architecture for a data-intensive science network

Mazahir Hussain[1,2], Buseung Cho[1,2]

1. Korea Institute of Science and Technology Information, KREONet Center,

2. University of Science & Technology,

Daejeon, South Korea

[e-mail: mazahir.hussain@kisti.re.kr, bscho@kisti@re.kr]

*Corresponding author: Buseung Cho

Abstract

The growing nature of the transferring of data is becoming critical in data-intensive science environment like Astronomy, Climate, Genomics, Energy Physics, etc. for research collaboration that leads to the problem of the collection and processing of network traffic data. The traditional mechanism of data collection and processing are not designed for large-scale networking such as the research and education (R&E) network that is optimized for transferring data with the speed of up to 100Gbps. We have proposed and deployed a system architecture featured with horizontal scaling, high-performance, and real-time network traffic data collection and processing for data-intensive network infrastructure with speed up to 100Gbs. Furthermore, the current proposed architecture will be enhanced for network management and operation automation, anomaly detection, and efficient resource utilization by utilizing artificial intelligence in the future.

Keywords: Network data collection, network traffic visualization, system architecture for network data processing

Ⅰ. Introduction

Network data collection is the first step towards monitoring and analyzing network data to ensure performance, reliability, availability, and security. The complexity of managing the network data increases with the large-scale network such as Research and Education Network which operates for data-intensive science between different research and education communities for transferring big data. In networking, many problems occur during transferring of data including network congestion, denial of Service (DoS) attacks, link down, server failure, low latency, and other unexpected errors. To normalize the aforementioned problems, we need a real-time monitoring system comprised of dashboards for data visualization, a database management system to store the data, and algorithms for finding the outrages in the network. The traditional data collection and processing mechanisms are not designed to handle gigabit speeds however, it needs a more robust, distributed, scalable, and high-performance mechanism to collect the data from routers, process the data, store it in a database for future purposes and visualize the flow of the data in real-time.

The goal of designing a distributed, high-performance, scalable, and real-time processing is to collect, process, analyze, store, and visualize the network flow data in large-scale networking infrastructure such as a data-intensive science network environment having the speed of up to 100Gbps. We proposed a system architecture that has the capability to handle the collection and processing of flow data in a large-scale networking infrastructure. The architecture is based on modular and distributed computing utilizing several open-source tools and technologies.

This paper is organized as follows. The next section discusses related works. Section 3 presents the detailed architecture of the proposed system while section 4 recommended the essential configurations for mission-critical and long-running spark jobs. Finally, the paper is concluded in section 5 along with the future works.

Ⅱ. Related Work

In this section, we review several related architectures for large-scale networking infrastructure related to our study. NetSage, which is a monitoring, analyzing, and visualization tool featured with SNMP data, flow data, tstat-based traffic, and personal data [1]. Another study is carried out using In-band Network Telemetry (INT) for automated network management and operation along with Artificial Intelligence tools and technologies by extracting the important network information for analysis and storing [2]. A push-based approach and a system for the measurement of the utilization of the

link are introduced as FlowSense [3]. There are different flow monitoring from the early nineties to currently available tools with different features scaling the speed up to 100Gbps is surveyed in this paper [4]. Recently, a multi-threaded based open-source platform is introduced for collecting, processing, and visualizing the large-scale network data to addressed both streaming and archived data as InSight2 [5].

III. System Architecture

The architecture of the system consists of several modules. The workflow of each module is shown in Figure 1. The first step is Network Traffic Mirroring, in which the flow data are mirroring from the routers. The second stage of the system is Network Traffic Ingestion, which is collecting the data from mirroring servers using argus client and streaming the data on different Kafka topics. The third stage of the system is Network Traffic Processing, which is consuming the data from different topics of Kafka and Processing the streams of data. The fourth stage of the system is Network Traffic Storage, which consists of a time series based RDBMS to store the flow data for real-time visualization and future purposes. The final stage in this architecture is Network Traffic Visualization, which is visualizing the data which is storing in the database.

2.1. Network Traffic Mirroring

Traffic mirroring is also known as port mirroring that sends the copy of traffics flowing from one or more source ports to one or more destination ports. In this architecture, we are using argus (Audit Record Generation and Utilization System) which is flow-based, passive monitoring tool for IP networks which is originally developed by CMU in 1993 [6]. The current version of the argus is an open-source project under QoSient LLC. Argus is reading the packets from the mirroring port and appending the network flow data on the specified socket connected to the argus client. Radium is a real-time Argus Record Multiplexer that reads the flow data from one or more remote hosts and writes to a single point by collecting and processing from multiple argus and Netflow records Figure 2. The following commands can be executed to start the argus server and radium.

argus -e localhost -P 561 -i enp216s0f0 -d

radium -S localhost:561 -d -e 127.0.0.1 -P 562

2.2. Network Traffic Ingestion

Kafka is a high-performance, horizontally scalable, fault-tolerant, and distributed event streaming platform for data pipelines and data integration [7]. The main purpose of using Apache Kafka for streaming the network flow data is the capability of its horizontal scaling and high performance when the network devices will be increased. In this module, Apache Kafka is reading flow data from the socket which is bind with the argus for writing the argus flow. To execute the Kafka producer and consumer, we need to start the zookeeper server and Kafka server along with setting up some of the Kafka related configurations which are part of the Network Traffic Ingestion module that can be seen in Figure 2. We have selected 26 features from the provided features by argus which are following along the command to produce the flow data on the topic of Apache Kafka.

ra -S localhost:562 -s stime saddr daddr proto sport dport flgs sco dco spkts dpkts sbytes dbytes sappbytes dappbytes sloss dloss sretrans dretrans dir sjit djit state tcprtt sload dload -c , | bin/kafka-console-producer.sh --topic argusflow --bootstrap-server localhost:9092

2.3. Network Traffic Processing

Apache Spark is an in-memory cluster computing system designed for nearly real-time stream processing and batch processing for large-scale data processing and faster computation [8]. It was originally developed by the University of California, Berkeley's AMPLab, and later donated to Apache Software Foundation. In the current system, we are utilizing *SPARK STREAMING* which is used for stream processing. It supports many sources for data ingestion including Kafka, Flume, HDFS/S3, Kinesis, Twitter, etc. In this stage, the streaming of the network flow data is reading from the argus topic, processing the data, and storing it in the database using Apache Spark. Apache Spark is chosen for this system because of its features of horizontal scalability, in-memory computation, fault-tolerant stream processing, and its supports for batch, interactive, and stream processing. In the next section, we have explained the configurations for running the spark job for a long period because the default configurations will not be able to make the job stable for a long period. Spark, Hadoop [9], YARN [10], and zookeeper [11] are working in the Network Traffic Processing module Figure 2.



Figure 1: System workflow

2.4. Network Traffic Storage

InfluxDB is one of the most powerful time-series databases having the capability to handle millions of data points per second and open source under MIT license [12]. The network flow data is in the form of time intervals along with the common flow properties so a database that is based on time series will be more efficient and suitable. To store the flow data, we have accomplished InfluxDB because of its high throughput of handling millions of records per second as the data-intensive science environment network is used to transfer data with a speed of up to 100Gbs.

2.5. Network Traffic Visualization

Network flow data visualization is one of the essential parts of monitoring the traffic of the network for network performance, optimizing infrastructure, and detecting malicious traffics. There are different open-source frameworks available for visualizing data supporting several data sources. In this architecture, we have leveraged Grafana which is an open-source platform under the Apache License 2.0 [13]. It allows visualizing, alerting, and creating dynamic monitoring dashboards. In Figure 3, the aggregated flow data for the last 24 hours is showing visualized by Grafana dashboard that is located in the Network Traffic Visualization module in the architecture Figure 2.

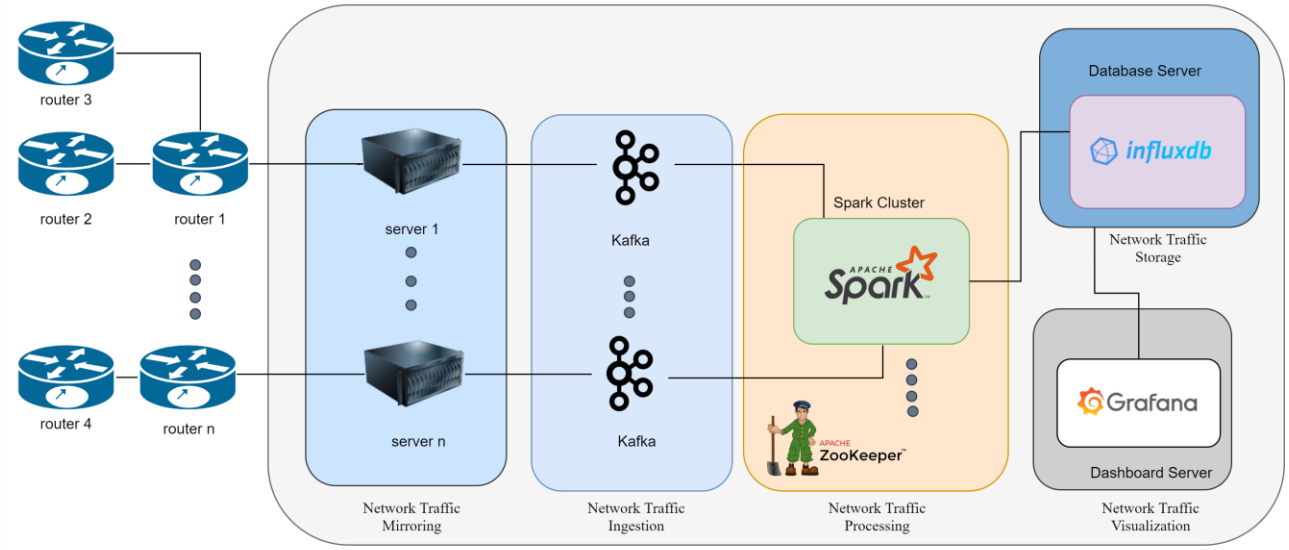
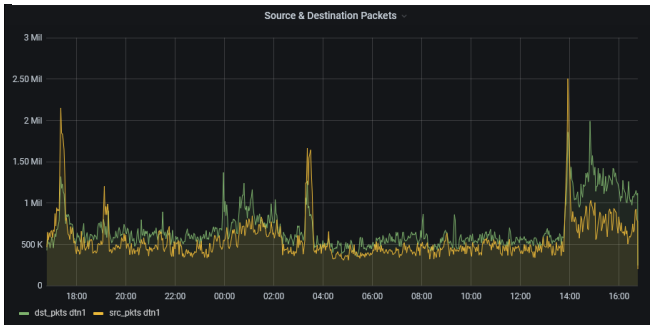
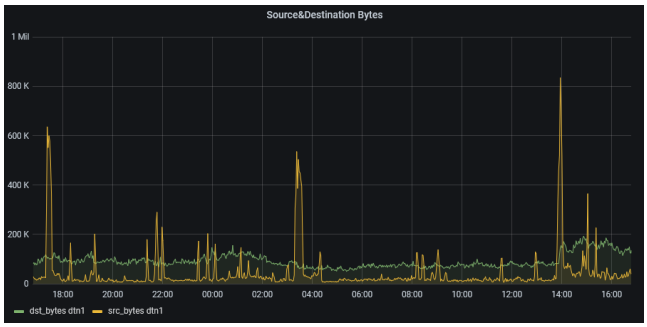


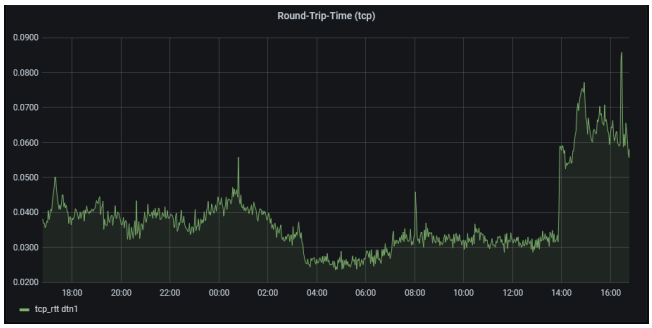
Figure 2: System architecture



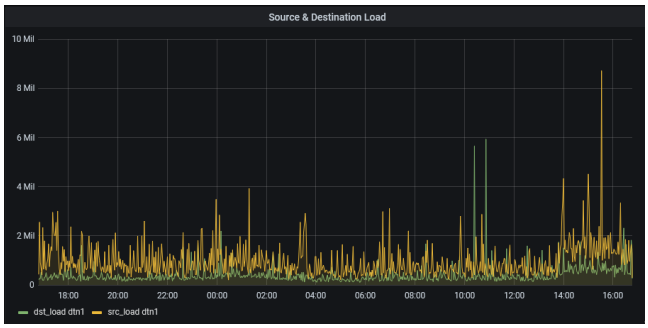
a). Source and destination packets count



b). Source and destination transaction bytes



c). Round trip time for tcp connection setup



d). Source and destination bits per second

Figure 3: Network traffic visualization on Grafana dashboard for last 24 hours

IV. Configurations for long-running spark streaming application

One of the main challenges with a long-running mission-critical application is the interruptions that stop the Spark job that leads to data loss and duplicates. There are different reasons for failing to be stable for a long-time because the apache spark is not designed for handling long-running jobs. We have tailored several configurations including the deployment mode while submitting the spark job. We have used YARN- Yet Another Resource Negotiator, which has mainly two components scheduler for resource allocation and Applications Manager (AM) to accepts the job submissions. In YARN cluster mode, the Application Master and Spark driver runs on the same container. By running the application in YARN cluster mode, the Spark job will not be affected even though the node which has submitted the job went down. As mentioned the spark driver and application master (AM) share a single JVM which can cause any error leading to failing the spark driver. To overcome this issue, we need to specify the maximum number of attempts to re-run the application if it fails. This parameter can be configured using *yarn.resourcemanager.am.max-attempts*. Another property comes along with a validity interval for the maximum executor failures before the main application will be failed. The default one is not suitable for long-running stream processing applications. In our current architecture, we have configured it with *{8 * num_executors}* with an *1h* of validity interval. To avoid the preemption by the Hive query, it is recommended to run the job in a separate queue. Spark supports Kerberos authentication that can be configured by providing the keytab and principle which helps to maintain the Kerberos token indefinitely.

V. Future Work and Conclusion

The next step of the project is to use the cutting-edge technologies of Artificial Intelligence for real-time network traffic classification, network traffic prediction, anomaly detection, suspicious traffic detection, and network management and operation automation to optimize the networking infrastructure for high-performance data transmission and better resource utilization by ensuring security, and, availability in the data-intensive science network environment. In this project, we have proposed and deployed a system architecture featured with horizontal scaling, high-performance, and real-time flow data collection and processing for data-intensive network infrastructure with speed of up to 100Gbs. The architecture is very flexible because of its scaling nature, the computational capability of any module can be scaled hierarchically for maximizing the performance of each module or the overall system. Secondly, the software used in each module can also be replaced with another without affecting the other module.

References

1. Turner, K., et al. *The NetSage Measurement Framework: Design, Development, and Discoveries*. in *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. 2020.
2. Hyun, J., et al., *Real-time and fine-grained network monitoring using in-band network telemetry.* International Journal of Network Management, 2019. **29**(6): p. e2080.
3. Yu, C., et al. *FlowSense: Monitoring Network Utilization with Zero Measurement Cost*. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.
4. Hofstede, R., et al., *Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX.* IEEE Communications Surveys & Tutorials, 2014. **16**(4): p. 2037-2064.
5. Kodituwakku, H.A.D.E., A. Keller, and J. Gregor, *InSight2: A Modular Visual Analysis Platform for Network Situational Awareness in Large-Scale Networks.* Electronics, 2020. **9**(10): p. 1747.
6. *Argus.* https://openargus.org/using-argus.
7. Kreps, J., N. Narkhede, and J. Rao. *Kafka: A distributed messaging system for log processing*. in *Proceedings of the NetDB*. 2011.
8. Zaharia, M., et al., *Spark: Cluster computing with working sets.* HotCloud, 2010. **10**(10-10): p. 95.
9. Shvachko, K., et al. *The Hadoop Distributed File System*. in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 2010.
10. Vavilapalli, V.K., et al., *Apache Hadoop YARN: yet another resource negotiator*, in *Proceedings of the 4th annual Symposium on Cloud Computing*. 2013, Association for Computing Machinery: Santa Clara, California. p. Article 5.
11. Junqueira, F. and B. Reed, *ZooKeeper: distributed process coordination*. 2013: " O'Reilly Media, Inc.".
12. Naqvi, S.N.Z., S. Yfantidou, and E. Zimányi, *Time series databases and influxdb.* Studienarbeit, Université Libre de Bruxelles, 2017: p. 12.
13. Thalmann, D., *An interactive data visualization system.* Software: Practice and Experience, 1984. **14**(3): p. 277-290.

한국통신학회 통신망운용관리연구회
2021년 통신망운용관리 학술대회 논문집
Proceedings of KNOM Conference 2021

ISSN : 2586-0232 (Online)

2021년 4월 29일 인쇄
2021년 4월 29일 발행

발행인 / 석우진 운영위원장
편집인 / 조부승 출판위원
발행처 / 한국통신학회 통신망운용관리연구회
서울시 서초구 서초동 1330-8 현대기림오피스텔 1504동 6호
전화: 02-3453-5555
홈페이지: www.knom.or.kr
디자인 및 편집 / 계명대학교 Comnet



한국통신학회 통신망운용관리연구회