Proceeding of KNOM 2016 Conference

2016년 통신망운용관리학술대회 논문집

일시: 2016. 5. 12 ~ 2016. 5. 13

장소: 강원대학교, 공학1호관 001호 사이버랩

주최: 한국통신학회 통신망운용관리연구회, 강원대학교

주관: 강원대학교 정보통신 연구소

경희대학교 글로벌소프트웨어융합연구소

한국과학기술정보연구원

후원: 한국통신학회, ICBMS Smart Mediator

한국통신학회 통신망운용관리연구회

초대의 말씀

한국통신학회 통신망운용관리 연구회 (KNOM)는 2016년 통신망 운용관리 학술대회 (KNOM Conference 2016)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 마련하였습니다.

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 2016에서는 통신망 전반에 대한 모델링, 설계, 서비스 제공, 운용 관리 및 보안 기술 분야의 최신 연구 개발 결과에 대한 유익한 정보제공과 토론의 장을 제공할 계획입니다. 부디 본 Conference가 운용관리 전반에 걸친 활발한 기술교류와 토론의 장이 될 수 있도록 각 분야의 전문가 여러분들의 적극적인 참여를 기대합니다. 또한 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.

2016. 5.

한국통신학회 통신망운용관리연구회 위원장 송왕철

통신망운용관리학술대회 운영위원장 주홍택

운영위원회

운영위원장	주홍택(계명대)
학술위원회	석승준(경남대)
포스터	석우진(KISTI)
튜토리얼	김윤희(숙명여대)
강연초청	정종문(연세대)
출판	이재오(한국기술교육대)
홍보	윤원용(동아대), 최태상(ETRI), 박용석(삼성전자), 최덕재(전남대), 곽대원(가인정보기술) 이석호(메타비즈), 박수현(국민대), 손지연(ETRI), 이길행(ETRI), 이광휘(창원대)
전시	이영석(충남대)
등록 및 진행	최미정(강원대), 김명섭(고려대)
자문	이영우(KT), 홍충선(경희대), 유재형(미래부), 송왕철(제주대), 김영명(KT) 김영탁(영남대), 이경휴(ETRI), 홍원기(POSTECH)

프로그램

2016년 5월 12일(목)

12:00-13:00 등록(공학1호관 001호 앞 로비)

발표장소 공학1호관 001호

13:00-15:00 **WS1. 공유 환경 구축을 위한 ICT 인프라 Management** | 좌장: 석우진 박사 (KISTI)

15:00-15:10 Coffee Break

15:10-16:30 **TS1. SDN Management** | 좌장: 김명섭 교수 (고려대)

[Tutorial Session] 좌장: 최미정 교수(강원대)

16:30-17:30 **[Tutorial 1] Software Defined User Centric Adaptive Traffic Management** | 최태상 박사 (ETRI)

17:30-18:30 **[Tutorial 2] Preparing SmartX IoT-Cloud Services leveraging Cloud-based Analytics** | 김종원 교수 (JIST)

19:00 만찬

2016년 5월 13일(금)

09:00-09:30 등록

발표장소	공학1호관 001호	공학1호관 309호
09:30-10:50	**TS2. Network Management** (좌장: 조부승 박사, KISTI)	**TS3.Future Internet** (좌장: 이재오 교수, 한국기술교육대)
10:50-11:00	Coffee Break	
11:00-12:20	**TS4. Internet of Things** (좌장: 김윤희 교수, 숙명여대)	**TS5. Cloud Computing** (좌장: 석 승 준 교수, 경남대)
12:20-13:30	Lunch	

발표장소 공학1호관 001호

13:30-14:00 **[Poster Session]** 좌장: 석우진 박사(KISTI)

14:00-14:10 **개회식** (사회: 김명섭 교수, 고려대)
개회사: 통신망운용관리연구회 위원장 송왕철 교수 (제주대)

[Keynote Address] 좌장: 주홍택 교수 (계명대)

14:10-14:50 **[Keynote] 네트워크 기술개발 현황 및 향후 방향** | 유재형 박사 (미래부 CP)

14:50-15:00 Coffee Break

[Invited Speaker Session] 좌장: 정종문 교수 (연세대)

15:00-15:40 **초청 세미나 1. Fraud Call 동향 및 탐지/차단 기술** | 김우태 팀장 (KT)

15:40-16:20 **초청 세미나 2. SDN/NFV 최신 기술 및 발전방향** | 유재욱 연구소장 (아토리서치))

16:20-17:00 **초청 세미나 3. 클라우드 최신 기술 및 발전방향 : Docker를 중심으로** | 조철용 팀장 (이노그리드)

17:00- **[Best Paper Award & 경품 추첨*]** (사회: 최미정 교수, 강원대)
폐회사: KNOM Conference 2016 운영위원장 주홍택 교수 (계명대)

18:00-20:00 KNOM OC Wrap-up Meeting

논문 목차

『Technical Session』

Technical Session 4. Internet of Things (5월 13일(금) 11:00 – 12:20, 공학1호관 001호))

(좌장: 김윤희 교수, 숙명여대)

1.오픈소스하드웨어를 이용한 농업ICT 플랫폼
조원익, 송왕철(제주대학교)

2. Secured Network Architecture and Solution for IoT Services
김우태, 이세희, 이영우(KT)

3. **IoTivity 기반의 출입 관리 방법**
이응기, 김현우, 주홍택(계명대학교)

4. **An Architecture for managing IoT Device Object in the IMS**
왕기초(한국기술교육대학교)

Technical Session 5. Cloud Computing (5월 13일(금) 11:00 – 12:20, 공학1호관 309호)

(좌장: 석승준 교수, 경남대)

1.ICBMS 플랫폼 연동 및 매쉬업 서비스 개발을 위한 Smart Mediator 연구
이도영, 정세연, 정태열, 홍원기(포스텍)

2. 스파크 클러스터의 SLA기반 오토스케일링
오유리, 최지은, 김윤희(숙명여대)

3. SDN/SFA 기반 미래 인터넷 연구자원 플랫폼 연동 구조
김성민, 이수강, 석우진, 김명섭(고려대학교)

4. 차량 클라우드 보안 요구사항 및 연구 이슈
김명수, 장인선, 주석진, 백상헌(고려대학교)

『Poster Session』 5월 13일 (금) 13:30 - 14:00 (좌장: 석우진 박사, KISTI)

1. **SDN 네트워크에서 무선 랜 이동성 지원 방안** 김성실.윤준호.박정재.권혁명.석승준 (경남대학교)

2. **네트워크 가상화 기술 연구 동향** 현종환, 홍원기 (포스텍)

3. **머신 러닝 기법을 활용한 SDN트래픽 엔지니어링 기법 개선 방안** 최준묵, 한윤선, 현종환, 홍원기 (포스텍)

4. **Dynamic ACL 모델을 사용한 SCADA 시스템 내 FTP 서비스의 화이트리스트 표현에 관한 연구**
정우석, 김성민, 구영훈, 김명섭 (고려대학교)

5. **응용 트래픽 분류에 정교한 페이로드 시그니쳐 구조** 구영훈, 이성호, 김성민, 이수강, 김명섭 (고려대학교)

6. **연속된 플로우 정보를 이용한 CDN서비스 트래픽 분류 방법 연구** 이수강, 심규석, 정우석, 김명섭 (고려대학교)

7. **RBAC 기반의 캠퍼스 출입자 권한 제어방법** 임이삭, 김현우, 주홍택 (계명대학교)

8. **출입 관리 서비스를 위한 EIP 기반 매쉬업 프레임워크 설계** 김현우, 권동우, 주홍택 (계명대학교)

9. **무선 네트워크에서의 다양한 TCP 프로토콜의 성능 평가** 정진화, 안동혁 (계명대학교)

10. **얼굴인식 Open API 게이트웨이 설계** 옥기수, 권동우, 주홍택(계명대학교)

11. **SDN기반 드론 무충돌 경로 제어** 백성복, 이영우(KT)

12. **Evolution of Identity Management in OpenStack based Clouds**
Ramneek, Wonjun Choi, Woojin Seok, Yoonjoo Kwon(KISTI)

13. **사물인터넷을 위한 다양한 응용층 프로토콜** 하태영, 정종문(연세대학교)

강연 목차

Tutorial	5월 12일 (목) 16:30—18:30 (좌장: 최미정 교수, 강원대)

1. Software Defined User Centric Adaptive Traffic Management
   최태상 박사 (ETRI)
2. Preparing SmartX IoT-Cloud Services leveraging Cloud-based Analytics
   김종원 교수 (JIST)

Keynote	5월 13일 (금) 14:10 – 14:50 (좌장: 주홍택 교수, 계명대)

1. 네트워크 기술개발 현황 및 향후 방향
   유재형 박사 (미래부 CP)

Invited Speaker	5월 13일 (금) 15:00 – 17:00 (좌장: 정종문 교수, 연세대)

1. Fraud Call 동향 및 탐지/차단 기술
   김우태 팀장 (KT)
2. SDN/NFV 최신 기술 및 발전방향
   유재욱 연구소장 (아토리서치)
3. 클라우드 최신 기술 및 발전방향 : Docker를 중심으로
   조철용 팀장 (이노그리드)

주최

KNOM KICS 1996 통신망운용관리연구회 한국통신학회

KNU 강원대학교 KANGWON NATIONAL UNIVERSITY

주관

KT 강원대학교 정보통신연구소 Kangwon National University Institute of Telecommunications and Information

경희대학교 KYUNG HEE UNIVERSITY

KISTI 한국과학기술정보연구원 Korea Institute of Science and Technology Information

후원

KICS 한국통신학회

ICBMS Smart Mediator

메모

SNMP 를 활용한 SDN 기반의 IMS 네트워크 관리방안

양우석 ○, 김정호, 이재오

한국기술교육대학교 정보통신공학과

{oos13, jungho32, jolee}@koreatech.ac.kr

요 약

5G 통신을 이끌어가기 위해 전도유망한 기술로는 IP Multimedia Subsystem(IMS)과 네트워크 가상화가 있다. IMS 는 멀티미디어 서비스 및 응용 서비스 제공을 위한 플랫폼이다. 이러한 IMS 는 Call Session Control Function(CSCF)라는 프로세스 노드들로 구성되며, 각 노드들은 협업하여 멀티미디어 통신을 지원한다. 하지만 근래의 네트워크 디바이스들의 폭발적인 증가는 CSCF 의 성능을 저하시키거나 지연의 발생을 초래할 수 있다. 최근 통신사업자들은 디바이스들의 증가로 인한 트래픽 문제를 대비해 네트워크 가상화 기술 중 하나인 Software Defined Network(SDN)를 주목하고 있다. 이는 기존의 네트워크 자원과 구조를 이용하여, 최대의 효율을 이끌어 낼 수 있기 때문이다. Simple Network Management Protocol(SNMP)는 네트워크 관리 시스템의 대표적인 프로토콜로, 모든 장비의 네트워크 트래픽을 관리 및 감시 할 수 있다. 본 논문은 5G 통신 환경에서 예상되는 네트워크 문제들을 SNMP 를 통해 SDN 기반의 IMS 로 해결 할 수 있는 방안을 제안한다. 이를 통해 SDN 기반의 IMS 에서의 트래픽 관리와 품질관리가 이뤄지고, 통신 사업자들은 네트워크의 자원과 구조를 효율적으로 사용 하여 최소한의 투자비용으로 높은 질의 서비스를 제공 할 수 있다.

1. 서론

5G 통신에서 가장 우려되는 문제점은 폭발적으로 증가하는 네트워크 디바이스의 수에 따른 신호 트래픽의 관리다. 이를 해결할 방안으로 통신사업자들은 네트워크 가상화에 주목하고 있다. 네트워크 가상화란 기존의 하드웨어와 소프트웨어 자원을 하나의 소프트웨어로 네트워크 자원과 구조를 효율적으로 이용 할 수 있다는 개념으로 대표적으로는 SDN 이 있다.

네트워크 디바이스들의 멀티미디어 서비스를 제공하기 위한 핵심 기술로는 IP 망을 기반으로 하는 IMS 가 있다. 이 IMS 를 SDN 과 접목시켜, SDN Controller 로 IMS 의 노드인 CSCF 들을 제어한다면 향후 IoT 산업에서 급증하는 디바이스로 인한 멀티미디어 서비스의 트래픽 문제를 해결할 수 있다고 본다.

SDN Controller 를 통해 IMS 의 CSCF 들을 제어하기 위해서는 Network Management Sysytem(NMS)이 필요하다. NMS 란 네트워크의 효과적인 운용, 설비 유지 및 품질 유지를 위하여 통신 시스템의 운영 관리와 시스템 내의 각종 자원의 구성, 상태 등을 감시하고 조작하는 것을 말하며, 구체적으로는 트래픽 관리, 품질 유지, 보존 관리 그리고 망 제어를 포함한다. 대표적인 프로토콜로는 SNMP 가 있는데, 본 논문은 SNMP 를 활용하여 SDN 기반의 IMS 네트워크를 관리하는 방안에 대해 제안한다.

2. 관련연구

2.1. IMS

IMS 는 IP 프로토콜을 기반으로 음성, 동영상 등의 멀티미디어를 복합적으로 지원하는 플랫폼이다[1]. IMS 는 기본적으로 IP 기반 멀티미디어 서비스의 제어와 Quality of Service (QoS)제어, 기존 인터넷, Public Switched Telephone Network (PSTN), 이동통신망 등의 기존 네트워크와의 연결, 로밍 등 다양한 서비스를 제공하기 위해 Session Initiation Protocol (SIP) 기반의 신호를 처리한다. IMS 는 호 처리를 위한 CSCF 가 존재한다. CSCF 는 각각의 역할에 따라 P(Proxy)-CSCF, I(Interrogating)-CSCF, S(Serving)-CSCF 라는 프로세스 노드들로 나뉘며, 각 노드들은 다음과 같은 역할을 한다[2].

- P-CSCF 는 단말이 Access 네트워크를 통하여 IMS 망에 접속하는 지점이다. P-CSCF 의 역할은 프록시와 사용자 에이전트(UA)의 역할을 수행하면서 사용자로부터 수신한 SIP 레지스트 메시지를 사용자의 홈 도메인을 참조하여 I-CSCF 로 전달한다. 그리고 사용자로부터 수신한 SIP 메시지를 등록 절차에 따라 수신한 주소를 이용하여 S-CSCF 로 전달하고, 사용자에게 SIP 메시지를 요구 또는 응답한다.
- I-CSCF 는 사용자의 홈 네트워크에 속하는 첫 번째 지점으로써, 하나의 네트워크 도메인에

여러 개 존재할 수 있다. 다른 네트워크에서 수신한 SIP 메시지를 S-CSCF 로 라우팅하고 Home Subscriber Server(HSS)로부터 S-CSCF 의 주소를 획득한다.

· S-CSCF 는 주로 사용자의 세션을 제어하는 HSS 에 등록하고 이후 사용자의 가입자 정보를 다운로드 하여 유지한다. 그리고 해당 서비스를 제공하기 위한 서비스 플랫폼과의 상호작용과 함께 사용자에게 서비스 자원과 관련된 정보를 제공한다.

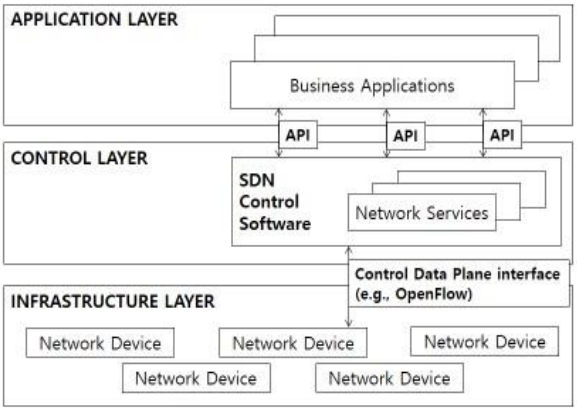
2.2. SDN

SDN 은 네트워크 제어기능을 물리적 네트워크와 분리시킨 네트워크 구조를 말한다. SDN 을 특징짓는 두 가지의 부분은 '첫째, 네트워크 제어 기능을 데이터 전달기능과 분리하여 구현해야 함' 과 '둘째, 네트워크 제어기능이 개발되고 실행될 수 있는 환경을 분리하여 전형적인 낮은 성능의 CPU 가 장착된 하드웨어 스위치에 더 이상 위치시키지 않음' 이다[4].

즉, SDN 이라면 기본적으로 네트워크 제어기능이 기존의 스위치나 라우터 등의 하드웨어와 별도로 분리되어야 하고, 데이터 전달 기능과도 역시 분리되어 실행 될 수 있는 네트워크 구조를 가져야 한다. 이는 SDN 제어 하드웨어가 네트워킹 장치들로 구성된 물리적인 인프라 계층 위에 위치하고 있으며, 이를 이용해 OpenFlow 같은 제어 인터페이스를 통해 통신한다는 의미다. 목표는 네트워크를 유연하고 프로그램 가능한 플랫폼으로 변모시켜 자원 활용을 최적화하여 더욱 비용 효율적이고 확장 가능하도록 하는 것이다.

다음의 [그림 1]은 OpenFlow 에서 묘사한 SDN 아키텍처를 나타낸다. 각 계층에 은 다음과 같은 역할을 한다.

- Application Layer: 네트워크 서비스 응용 계층
- Control Layer: 네트워크 구성 및 제어 계층
- Infrastructure Layer: Packet 전달 계층



[그림 1] SDN 아키텍처

이러한 SDN 의 특성은 소수의 하드웨어 자원에서도 소프트웨어만으로 충분히 네트워크가 이뤄질 수 있도록 한다. 이로 인해 자금, 운영 경비를 낮추어 유연하고 민첩한 해결법을 업계 표준의 서버 하드웨어 구성요소에서 공급할 수 있도록 돕는다[4].

2.3. SNMP

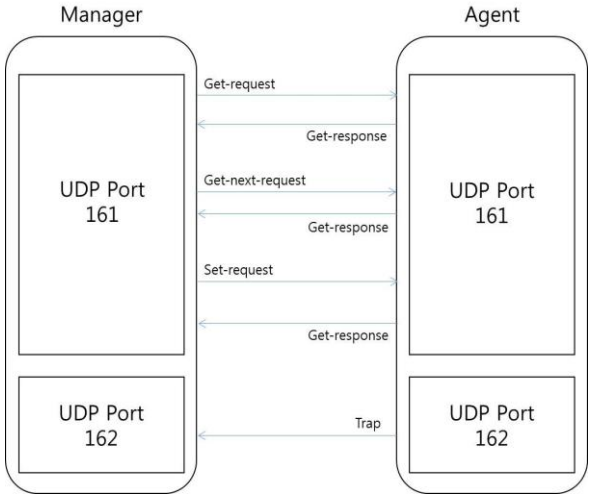
SNMP 는 네트워크 관리 프로토콜의 가장 대표적인 프로토콜로 TCP/IP 기반의 네트워크에서 네트워크상의 각 호스트의 정보를 자동적으로 수집하여 관리하여 문제 발생시 자동적으로 장애상황을 관리 시스템에 통보한다.

SNMP 의 기능은 다음과 같이 크게 4 가지로 나뉜다[5].

- 네트워크 구성관리
- 성능관리
- 장비관리
- 보안관리

SNMP Agent 는 피 관리대상 장비. 즉 호스트, 라우터, 브릿지 그리고 허브와 같은 네트워크 장비에 설치되어있고 관리 시스템의 요구에 따라 관리 정보를 송달하거나 관리 시스템의 요구를 수행한다. 이러한 정보들을 수집하여 저장한 데이터베이스를 Management Information Base(MIB)라 한다.

다음 [그림 2]는 SNMP Manager 와 Agent 간의 통신 흐름이다.

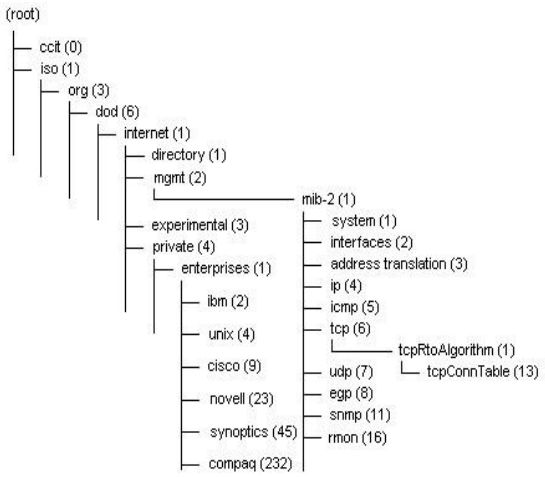


[그림 2]SNMP Manager 와 Agent 의 통신 흐름

SNMP 의 정보 교환은 메시지 단위로 이루어지며 SNMP 메시지는 GetRequest PDU. GetNextRequest PDU, SetRequest PDU, GetResponse PDU, Trap PDU 의 다섯 가지 형태를 포함하며, 기본적으로 폴링 방식을 이용한다. GetRequest PDU 는 Manager 가 하나 또는 그 이상의 Agent 에게 오브젝트 값을 요청하기 위해 사용하며, GetNextRequest PDU 는 요청하는 것은 똑같지만 돌려받는 오브젝트 값은 MIB 상에 정

의된 어휘적인 순서로서의 다음 오브젝트 값이다. SetRequest PDU 는 값을 변경하기 위해 Manager 가 Agent 로 보내는 메시지이다. 위의 세 가지 메시지에 대한 응답은 GetResponse PDU 로 돌아간다. Trap PDU 는 Agent 가 자신에게 일어난 사건들에 대한 정보를 Manager 에게 알리는 메시지다.

MIB 는 SNMP 가 관리해야 할 네트워크 자원 오브젝트의 분류된 정보를 의미한다. 관리할 자원 오브젝트는 시스템정보, 네트워크사용량, 네트워크 인터페이스정보 등이 된다. MIB 로 정리된 값에 대해서 SNMP 는 Agent 의 상태를 파악하거나 그 값을 변경할 수 있다. 결과적으로 Agent 의 장비와 상태도 변경할 수 있으며, 동작을 명령할 수 있다. MIB 는 편한 관리를 위해 Tree 구조를 갖는다. 다음 [그림 3]은 MIB 의 기본구조의 일부를 나타낸다.



[그림 3] MIB 의 기본구조

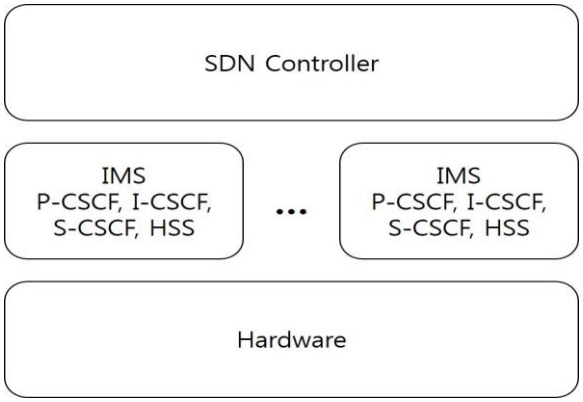
MIB 는 계층적 구조를 가짐으로 필요에 따라 확장해서 사용이 가능하기 때문에 사설로 MIB 를 만들어 사용 할 수 있다. 이러한 사설 MIB 는 private(4)의 enterprise(1)에 정의해서 사용 할 수 있다. 이러한 독자적인 MIB 를 확장 MIB 라고 부른다. MIB 내의 특정 오브젝트는 Object Identifier(OID)로 확인 할 수 있다. [그림 3]의 각 계층의 구조 옆에 명시된 연속된 정수가 OID 다[5].

3. SNMP 를 활용한 SDN 기반의 IMS 관리

본 장에서는 제안하는 관리 방안에 대해 설명한다. 앞서 관련연구에서 언급한 내용들을 토대로 SDN 기반의 IMS 의 구조를 설명한 뒤, SNMP 를 이용하여 어떻게 관리가 이뤄지는 보여준다.

3.1. SDN 기반의 IMS

SDN 기반의 IMS 는 SDN Controller 를 통해 각 CSCF 들의 역할을 관리한다. 다음 [그림 4]는 SDN 기반의 IMS 의 단순 구조를 나타낸다.

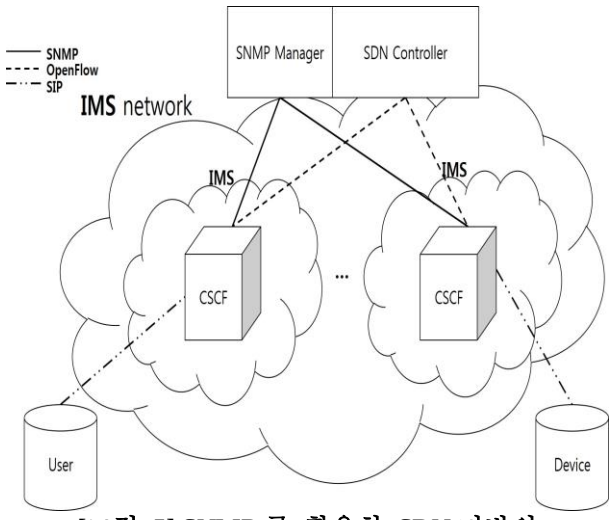


[그림 4] SDN 기반의 IMS 구조

SDN Controller 는 IMS 내의 CSCF 들의 상태를 체크하여 부하가 발생할 경우 다른 IMS 의 CSCF 로 넘길 수 있게끔 신호를 OpenFlow 를 통해 처리를 넘긴다[7]. 이러한 구조의 네트워크 제어를 통해 네트워크 서비스의 유연성을 높여 응용서비스의 개발 및 관리비용을 줄일 수 있다.

3.2. SNMP 를 활용한 네트워크 관리

3.1 에서 언급한 SDN 기반의 IMS 구조를 SNMP 로 보다 신뢰성 있고 간편하게 네트워크 관리를 할 수 있는 방안을 제안하려 한다. [그림 5]는 전체적인 구조를 나타낸 것이다.

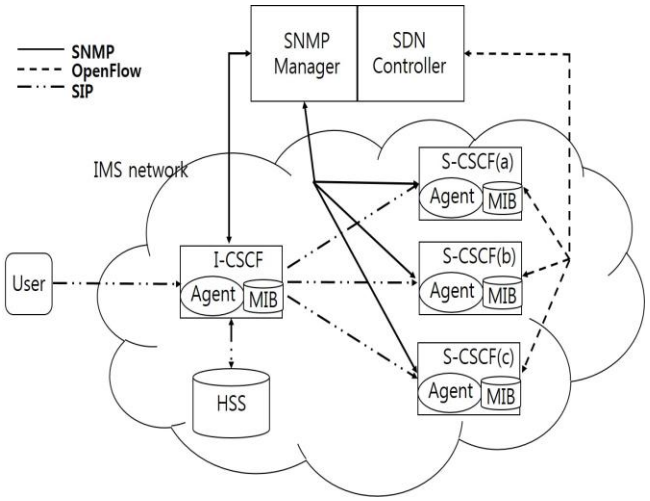


[그림 5] SNMP 를 활용한 SDN 기반의 IMS 네트워크 관리 구조

User 에서 SDP 가 포함된 SIP 신호를 CSCF 로 보내면, CSCF 에 있는 SNMP Agent 는 SNMP 프로토콜을 이용하여 주기적으로 MIB 를 SNMP Manager 와 정보를 주고 받는다. 만약 기존의 CSCF 에 부하가 발생하면 SNMP Manager 는 SDN Controller 가 적절한 CSCF 를 선택하여 SIP 메시지를 전달할 수 있도록 한다. 이를 통해 네트워크 자원과 구조를 효율적으로 이용 할 수 있다.

다음 [그림 6]은 위의 구조를 통하여 I-CSCF 에

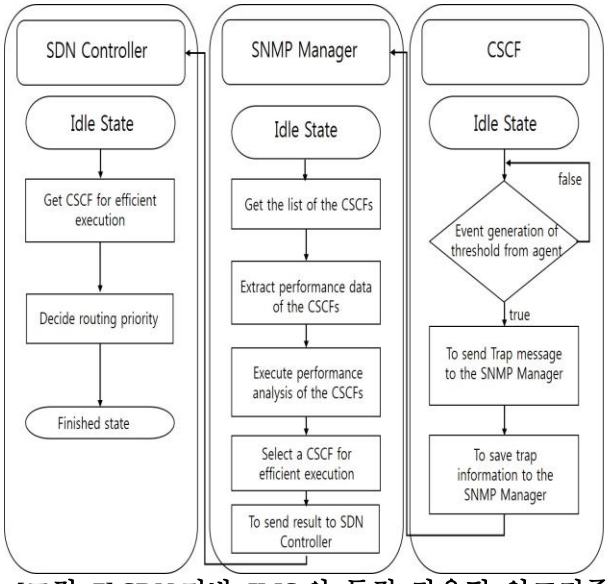
서 S-CSCF 로의 SIP 메시지 흐름과 SNMP Manager 와 SDN Controller 의 제어를 시나리오로 나타낸 것이다.



[그림 6] 메시지 라우팅 시나리오

다수의 S-CSCF(a,b,c)가 존재할 때, 임의의 S-CSCF 에 부하가 발생했다고 가정한다. 부하가 발생한 S-CSCF 는 SNMP Manager 에게 부하의 발생을 알린다. 이후 SNMP Manager 는 라우팅 할 S-CSCF 의 주소를 SDN Controller 에게 알려주고 적절한 S-CSCF 를 선택하여 SIP 메시지를 전달한다.

위와 같은 시나리오가 이뤄지기 위해서는 다음 [그림 7]과 같은 동적 라우팅이 이뤄져야 한다. 동적 라우팅 알고리즘을 이용하는 이유는 네트워크의 규모가 커질 경우 CSCF 의 노드 수는 더 많아지고 관리 해야 할 부분이 많아지기 때문에 부하가 걸릴 시 수행이 힘든 정적 라우팅보다 적절하기 때문이다.



[그림 7] SDN 기반 IMS 의 동적 라우팅 알고리즘

[그림 7]의 알고리즘은 CSCF 와 SNMP Manager 그리고 SDN Controller 세 구성으로 나뉘어 동작한다. 각 구성은 다음과 같은 작업을 수행한다.

· CSCF: CSCF 내의 Agent 로부터 Threshold 가 발생시, Trap 메시지를 SNMP Manager 로 보고 및 저장
· SNMP Manager: CSCF 들의 정보를 비교 분석 후 가장 적합한 CSCF 선택
· SDN Controller: 선택된 CSCF 를 확인 후 라우팅 우선순위 결정

알고리즘의 전체적인 흐름을 설명하자면 SNMP Manager 는 주기적으로 Get 메시지를 통하여 CSCF 로부터 성능 정보에 관한 정보를 수집한다. SNMP Manager 에서 설정한 Threshold(CPU, Memory)를 초과하거나 예외 이벤트가 발생할 경우에 Trap 메시지를 통해 SNMP Manager 에 보고 및 저장한다. Trap 메시지는 CSCF 들이 비정상적인 동작 또는 이벤트가 발생할 경우에 이를 SNMP Manager 에 전달하는 역할을 수행한다. SNMP Manager 는 이렇게 전송된 CSCF 정보들을 추출한 뒤 각 CSCF 들의 성능을 비교 분석한다. 그리고 가장 효율 높은 CSCF 를 선택한 뒤 SDN Controller 에게 이 정보를 전송하면, SDN Controller 는 가장 효율이 좋은 라우팅 경로를 결정하여 수행한다.

4. 결론 및 향후 과제

5G 통신으로 향하는 현 시점에서 급증하는 네트워크 기기의 트래픽 관리를 위한 효율적인 네트워크 관리 방안은 끊임없이 제시되고 있고 앞으로도 유망한 분야가 될 것으로 전망된다. 본 논문은 기존의 멀티미디어 제공 플랫폼인 IMS 에 SDN 을 접합하여 가장 널리 쓰이는 네트워크 관리 프로토콜인 SNMP 로 IMS 내의 부하 관리를 제안한다. 하지만 이는 IMS 내의 부하만을 관리할 뿐, SDN Controller 에 발생할 수 있는 부분은 고려하지 않았다. 때문에 향후 실제 구현을 통해 발생할 수 있는 다른 문제점들을 해결해 보고자 한다. 또한 향후 연구에서는 IMS 와 SDN 을 결합하여 SNMP 를 이용하여 네트워크의 효율이 얼마나 좋아지는지 시뮬레이션의 결과를 보여주고 또 SNMP 의 MIB 정보를 private MIB 로 변경하여 보다 더 높은 효율과 간편화된 네트워크 관리를 검증해 볼 것이다.

5. 참고문헌

[1] 3GPP TS 23.228, “IP Multimedia Subsystem (IMS) V10.1.0” (2010-06).
[2] 조재형, 이재오, “IMS/SDP 구조 및 Presence 서비스 구현 모델”, KNOM, 2008.
[3] P. Demestichas, A. Georgakopoulos, D. Karvounas, K.

Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, "5G on the horizon: key challenges for the radio-access network", Vehicular Technology Magazine, IEEE, vol. 8, no. 3, pp. 47–53, 2013.

[4] "Software-defined networking: the new norm for networks", White paper. Open Networking Foundation. April 13, 2012. Retrieved August 22, 2013.

[5] J. Case "A Simple Network Management Protocol (SNMP), IETF RFC 2790, March2000.

[6] H. Kim and F. N, "Improving network management with software defined networking," Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[7]A. Lara, A. Kolasani and B. & Ramamurthy, "Simplifying network management using Software Defined Networking and OpenFlow," IEEE International Conference on Advanced Networks and Telecommunications Systems, pp.24-29, December 2012

DPI 를 이용한 SDN 트래픽 매니지먼트 시스템

정세연 O, 이도영, 최준묵, 홍원기

포항공과대학교 컴퓨터공학과

{jsy0906, dylee90, juk909090, jwkhong} @postech.ac.kr

요 약

Software-Defined Networking (SDN)은 네트워크 분야에서 주목받는 연구 분야의 하나로써 그 기본 개념 및 목적은 네트워크를 컨트롤 평면과 데이터 평면으로 분리하여 관리 및 운용을 유연하게 만드는 것이다. 컨트롤 평면에 중앙집중화된 형태로 위치하는 컨트롤러는 스위치에 플로우 룰을 설치하고 관리하는 기능 등을 갖는데 이를 통해 효과적으로 네트워크를 관리 및 운용 할 수 있다. 현재까지 공개된 SDN 컨트롤러 중 ONOS 는 SDN 을 구성하는 여러 컴포넌트를 계층화하고 모듈 단위의 API 를 제공함으로써 손쉽게 어플리케이션의 개발 및 실행을 가능하게 한다. 하지만 이러한 편의성과 Programmability 에도 불구하고 현재 ONOS 의 기능을 충분히 활용하는 어플리케이션은 많지 않은 실정이다. 따라서 본 논문에서는 ONOS 가 제공하는 기능들을 활용하여 SDN 트래픽 관리 도구로 동작하는 Firewall 과 Bandwidth Manager 어플리케이션을 제안한다. 두 어플리케이션은 네트워크로 유입되는 플로우의 통과 여부와 Data rate 등을 결정하는 역할을 수행한다.

1. 서론

오늘날 우리는 인터넷이 없는 삶은 상상 할 수 없는 환경에서 살아가고 있다. 네트워크는 등장 이후 극적인 발전을 이루어 오늘과 같은 인터넷 시대를 만들어냈고, 덕분에 수 많은 사람들은 네트워크를 통해 다양한 정보를 얻고 SNS, Email 등의 서비스를 이용하며 살아가고 있다. 하지만 이런 발전에도 불구하고 네트워크 분야는 여전히 해결해야 할 문제들이 많이 남아있는 영역이기도 하다. 특히 네트워크는 이를 구성하는 수 많은 종류의 네트워크 기기들로 이루어져 있는데, 각 기기들이 가지는 인터페이스와 이를 제공하는 회사가 다양하다는 특징이 있다. 이는 다시 말하면 네트워크 설정을 변경하기 위해서는 그 네트워크를 구성하는 모든 기기들의 인터페이스에 맞게 각 기기들의 설정을 변경해야 한다는 것을 의미한다. 때문에 실제로 새로운 서비스 도입이나 네트워크 설정 변경을 목적으로 즉각적으로 네트워크에 변경사항을 적용하는 것은 불가능에 가깝다.

따라서 이러한 문제를 해결하기 위해 많은 노력들이 있어왔는데 그 중에서도 Software-Defined Networking (SDN)은 탁월한 해결방안 중의 하나로 여겨지고 있다. SDN 의 기본 개념은 네트워크를 컨트롤 평면과 데이터 평면으로 분리하여 관리 및 운용을 유연하게 만드는 것이다. 스위치와 라우터로 이루어진 데이터 평면은 패킷을 전달하는 역할을 하는데 이때 어느 경로로 패킷을 전달할 것인지는 컨트롤 평면의 플로우룰 설정에 따른다. 한편 컨트롤 평면은 데이터 평면을 관리 및 제어하는 역할을 갖는데 중앙 집중화된 컨트롤러가 컨트롤평면에 위치하여 다수의 스위치 또는 라우터를 관리한다. 이를 위해 컨트롤러는 자신이 관리하는 네트워크의 데이터 평면, 즉 스위치들과 끊임없이 통신하여 네트워크 상태를 파악해야 한다. 컨트롤러와 스위치들 간 통신을 위해서는 두 요소 간에 표준화된 통신 프로토콜이 요구되었고 그 결과 OpenFlow 프로토콜이 등장하였다. OpenFlow 표준을 따르는 스위치와 컨트롤러 사이에서 새로운 패킷이 스위치에 유입될 경우, 스위치는 패킷의 헤더 일부를 포함한 Packet-In 메시지를 컨트롤러로 전달하고 컨트롤러는 수신한 헤더 정보와 네트워크 토폴로지 정보를 이용하여 패킷의 라우팅 경로를 결정한다. 이후 컨트롤러는 패킷의 진행 경로 상의 스위치들에 OpenFlow 메시지를 통해 플로우룰을 설치한다. 이러한 과정을 통해 SDN 에 특정 플로우의 경로가 결정되고 해당 플로우는 이 경로를 통해 목적지까지 전달된다.

SDN 이 주목 받기 시작하면서 현재까지 수 많은 OpenFlow 기반 오픈소스 SDN 컨트롤러가 개발되었다. 그 중 OpenDaylight [1]는 다중 Southbound 프로토콜 플러그인들과 다양한 서비스 및 어플리케이션들을 제공한다. 이를 통해 어플리케이션 개발자와 연구자들이 네트워크 디바이스들간의 통신보다 SDN API 들에 더 집중할수록 돕고 있다. Ryu [2]는 Python 으로 제작된 컴포넌트 기반의 SDN 프레임워크로써, 체계적으로 정의된 API 와 소프트웨어 컴포넌트들을 제공하여 개발자들이 쉽게 네트워크 관리 어플리케이션을 개발할 수 있는 환경을 제공한다. Floodlight [3]은 Apache-licensed 의 Java 기반 컨트롤러로 Big Switch Networks 에 소속된 많은 개발자들에 의해 개발되어 공개되었다. Floodlight 는 컨트롤러를 쉽게 확장하고 개발할 수 있도록 모듈 로딩

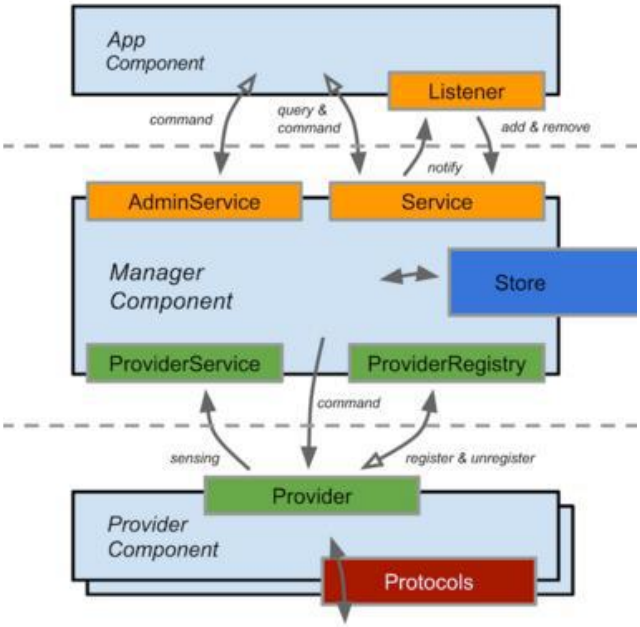
시스템을 제공한다는 특징이 있다. ONOS[4]는 성능, 확장성, 가용성을 중시하는 분산형 SDN 컨트롤러이며 편의성 있는 Web GUI 와 SDN 컴포넌트 단위의 계층 구조를 통해 어플리케이션을 개발하고 컨트롤러에 탑재할 수 있는 기능들을 제공한다. 하지만 ONOS 는 상대적으로 최근에 공개되었고 컨트롤러의 기능 자체에 대한 개발이 우선시 되어왔기 때문에 ONOS 기반 어플리케이션들은 많지 않은 실정이다. 따라서 본 논문에서는 ONOS 가 제공하는 기능을 활용하여 SDN 트래픽 관리 도구로써 동작하는 Firewall 과 Bandwidth Manager 어플리케이션을 제안한다. 이들은 데이터 평면에 존재하는 플로우를 TCP/UDP 포트번호 또는 외부 DPI 프로그램을 통해 분석하여 해당 트래픽을 특정 어플리케이션 및 프로토콜로 분류하며, 이러한 정보를 통해 네트워크의 보안을 강화하고 어플리케이션에 따라 트래픽의 Data rate 을 동적으로 할당하는 등 네트워크를 보다 효율적이고 유연하게 관리하는 것을 목적으로 한다.

본 논문의 이후 구성은 다음과 같다. 우선 2 장에서는 SDN 의 관련 연구 및 배경에 대해 설명하고, 3 장에서는 이 논문에서 제안하는 어플리케이션인 Firewall 과 Bandwidth Manager 의 구체적인 구조 및 구현에 대해 서술한다. 4 장에서는 두 어플리케이션의 실제 검증 결과에 대해 소개하고 마지막 5 장에서는 결론 및 향후 연구 계획에 대해 서술한다.

2. 관련 연구 및 배경

SDN 의 등장 이후, SDN 의 특성을 이용하여 네트워크의 보안성을 강화하기위한 연구가 많이 진행되어왔다. FlowGuard [5]는 SDN 기반의 Firewall 로서 패킷 또는 플로우 단위의 세밀한 수준에서 트래픽을 제어한다. 또한, 이미 존재하는 플로우룰과 FlowGuard 에 의해 새롭게 설치된 플로우룰의 충돌 가능성을 고려함으로써 오픈소스 컨트롤러에 내장된 기본적인 Firewall 보다 더 향상된 기능을 제공한다. 하지만 FlowGuard 는 해당 트래픽을 발생시키는 어플리케이션 또는 프로토콜을 식별할 수 없다는 단점이 있다. 이를 극복하기 위한 노력 중의 하나로 Application-aware SDN 을 구축하기 위해 M Jarschel [6]등은 DPI 를 이용하여 YouTube 트래픽을 분류하고, 해당 트래픽을 발생시키는 호스트에서 버퍼 기반의 네트워크 최적화 기법을 적용시켰다. 이를 통해 네트워크 자원을 효율적으로 사용하는 것이 가능해졌지만 오직 YouTube 트래픽만을 고려한다는 한계가 존재한다. 한편 Zafar Qazi [7]등은 Application-awareness 시스템을 SDN 에 접목한 Atlas 프레임워크를 제안했다. 이는 별도의 DPI 를 이용하지 않고 각 스위치의 버퍼에 쌓인 패킷들을 Machine Learning 기법을 이용하여 특정 어플리케이션의 트래픽으로 분류한다. 하지만 이는 식별된 어플리케이션 트래픽의 Data rate 을 제어할 수 없고 오직 Drop 여부만을 결정 할 수 있다는 한계가 있다.

본 논문에서 제안하는 시스템은 OpenFlow 기반의 SDN 에서 트래픽 분류에 따른 Firewall 기능 및 동적 Data rate 할당 기능을 제공하며, ONOS 컨트롤러에서 동작하는 어플리케이션 형태로 구현되었다. 제안하는 어플리케이션은 ONOS 의 구현 관점의 계층 [그림.1]에서 최상위 계층인 App Component 에서 동작한다. 어플리케이션은 Packet-In 메시지 및 플로우 관련 통계 값들을 하위 계층으로부터 받아오고, 이를 이용하여 특정 네트워크 정책을 결정한다. 그리고 이러한 정책을 네트워크에 적용하기 위한 요청을 하위 계층으로 전달한다. ONOS 는 SDN 을 관리하기 위해 다양한 기능들을 수행하는데 이에 대한 Programmability 를 지원하기 위해 하위 계층에 다양한 Manager 와 Provider 가 존재한다.



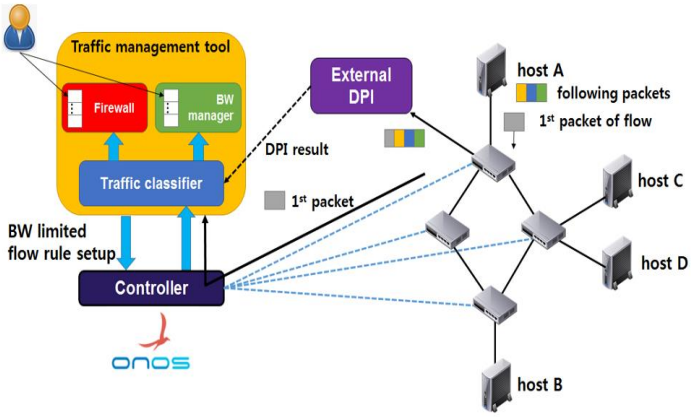
[그림.1] ONOS 구조

3. 시스템 구조 및 구현

본 논문에서 제안하는 시스템은 [그림.2]처럼 다음 세 가지 요소로 구성된다. (1) 트래픽 관리 도구, (2) 호스트와 OpenFlow 프로토콜을 지원하는 스위치들로 구성된 SDN 토폴로지, (3) 외부 DPI 프로그램. 첫 번째 구성요소인 트래픽 관리 도구는 ONOS 컨트롤러 위에서 동작하며, 두번째 구성요소인 OpenFlow 지원 스위치들은 ONOS 컨트롤러에 연결되어 있다. 외부 DPI 프로그램은 Edge 스위치에 새로운 플로우가 유입될 경우 이에 대한 DPI 를 실시하고 분석된 결과(어플리케이션 타입)를 컨트롤러로 전달한다. 이를 통해 컨트롤러는 해당 플로우에 어느 정도의 Data rate 을 할당할지 설정하거나 플로우 자체의 유입을 차단할 수 있다. 이러한 결정은 네트워크 관리자가 ONOS Web GUI 를 통해 트래픽 관리

도구에 설정한 정책을 따른다.

본 연구에서는 (2)에 해당하는 SDN 환경을 구성하기 위해서 Mininet [8] 네트워크 에뮬레이터를 활용해서 가상의 SDN 네트워크를 구성하였다. 그리고 OpenFlow 1.3 에서 지원하는 Meter 테이블 기능을 이용하기 위해 해당 기능들이 구현된 CpqD/ofsoftswitch13 [9]을 OpenFlow 소프트웨어 스위치로 설치하였다. 또한 (3) 외부 DPI 프로그램으로는 오픈 소스 DPI 라이브러리인 nDPI 를 사용하였다. nDPI 는 테스트베드에서 Stand-alone 으로 동작하며 높은 정확도로 200 개 이상의 어플리케이션 트래픽을 분류할 수 있다[10]. nDPI 는 Edge 스위치들 위에서 동작하며 유입된 플로우의 DPI 결과를 REST API 를 통해 ONOS 컨트롤러로 전달한다.

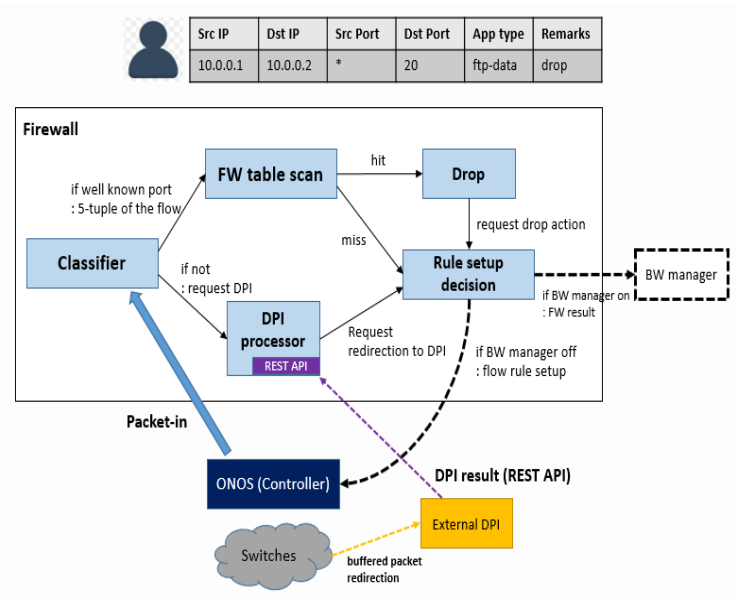


[그림.2] 시스템 전체 구조

A. Firewall

일반적으로 OpenFlow 기반 Reactive 방식의 SDN 에서 새로운 플로우가 유입될 때, 스위치에는 라우팅을 위한 플로우 룰이 설치되어 있지 않다. 이러한 경우에 해당 플로우의 Layer 2 부터 Layer 4 까지의 헤더 정보가 Packet-In 메시지를 통해 컨트롤러로 전달되고, 컨트롤러는 수신한 메시지를 분석하여 경로 설정에 필요한 IP 주소와 TCP/UDP 포트번호 등의 정보를 추출한다. 본 논문에서 제안하는 [그림.3]의 Firewall 은 이러한 정보를 활용하여 TCP/UDP 포트번호 기반 트래픽 분류를 수행, 해당 플로우의 어플리케이션 타입을 식별한다. 만약 해당 플로우가 포트번호 기반으로 분류가 가능할 경우, 플로우의 5-tuple (Source/Destination IP, Source/Destination 포트번호, 프로토콜) 정보를 추출하여 매칭되는 Firewall 룰이 있는지 확인하고 룰의 정책에 따라 해당 플로우를 처리한다. 하지만 동적포트할당(Dynamic Port Allocation) 등의 이유로 포트번호만으로 분류되지 않는 플로우의 경우에는 컨트롤러의 REST API 를 통해 외부 DPI 프로그램으로부터 해당 플로우의 DPI 결과(어플리케이션 타입)를 전달 받아 분류한다. 보다 상세히 설명하면, [그림 2.]를 기준으로 Sender 호스트로부터 발생하는 모든 플로우는 DPI 를 통과하게 되며 분석 중에 새로운 플로우가 식별될 경우 타임스탬프와 해당 플로우의 5-tuple 및 어플리케이션 타입 정보를 컨트롤러로 전달한다(REST API). 컨트롤러는 외부 DPI 프로그램으로부터 전달받은 플로우에 대한 DPI 결과와 스위치로부터 Packet-In 메시지 형태로 전달 받은 플로우 정보를 비교하여 동일한 플로우로 판단되는 경우, 해당 플로우의 5-tuple 과 DPI 결과를 바탕으로 매칭되는 Firewall 룰을 찾아 적용한다. 이와 같은 경우, 외부 DPI 의 플로우 분석 및 REST API 를 통한 결과 전달에 걸리는 시간과 스위치로부터의 Packet-In 메시지의 도착 사이에 시간 차가 존재할 수 있기 때문에 먼저 받은 DPI 결과들을 캐시하거나 DPI 결과가 도착할 때까지 플로우의 처리를 지연시키는 등의 동기적 처리 방식이 요구된다.

Firewall 룰은 관리자가 네트워크를 보호하기 위해 미리 설정하는 것으로 가정하는데, 이 룰들의 명세는 <IP 주소, 포트 번호, 프로토콜(어플리케이션), Action>으로 구성된다. 만약 특정 플로우를 Drop 하는 룰이 정의되어 있다면 Firewall 은 컨트롤러에게 해당 플로우를 Drop 할 것을 요청하고, 컨트롤러는 플로우가 유입된 스위치에 해당 플로우를 Drop 시키는 플로우룰을 설치한다. 결과적으로 네트워크 관리자가 원하지 않은 플로우들은 더 이상 네트워크로 진입하지 못하고 Edge 스위치에서 Drop 된다. 반면, 네트워크 진입이 허용된 플로우의 경우 (Firewall 의 Forward 액션), Bandwidth Manager 로 전달되어 Data rate 제어 및 이후의 플로우 처리 과정이 진행된다.

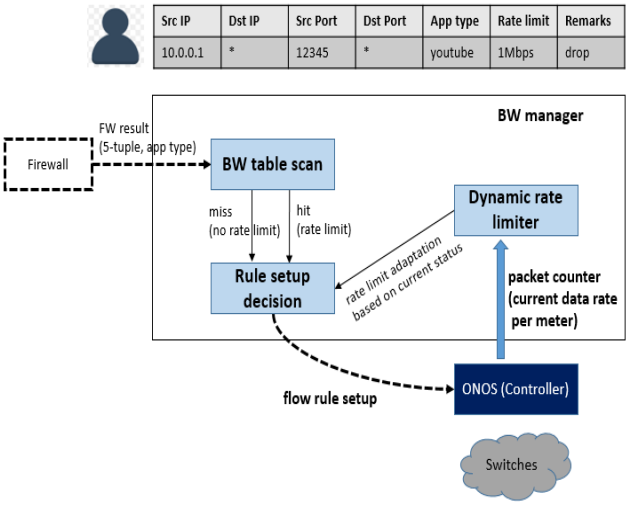


[그림.3] Firewall 설계

B. Bandwidth Manager

Bandwidth Manager 는 [그림.4]와 같이 네트워크 관리자로부터 5-tuple 정보, 어플리케이션 타입 및 Date rate 한도를 Bandwidth 룰로 전달 받는다. 이

룰들은 [그림.5]와 같이 ONOS Web UI 를 통해 네트워크 관리자로부터 입력 받아 테이블 형태로 저장된다. Firewall 을 통과한 플로우는 Bandwidth Manager 에서 미리 설정된 Bandwidth 룰과 매칭되는지 확인된다. 매칭되는 룰이 존재할 경우 Bandwidth Manager 는 해당 플로우의 라우팅을 위한 플로우룰의 설치과정에서 매칭 액션의 한 형태로 OpenFlow 1.3 에서부터 제공되는 Meter band 를 적용한다. 이를 통해 해당 플로우에 적용되는 Data rate 의 한계치를 지정할 수 있다. 컨트롤러는 스위치에 대한 주기적인 OpenFlow 폴링을 통해 플로우룰에 설정된 Meter band 와 관련된 Meter 통계값 (OFPMP_METER)을 Dynamic Rate Limiter 로 전달한다. Meter 통계값은 특정 플로우룰에 적용된 Meter band 의 영향을 받은 패킷 개수와 Byte 정보를 제공한다. Dynamic Rate Limiter 는 설치된 플로우룰의 Meter band 값을 플로우의 상태에 따라 동적으로 조절한다. 예를 들어, 일반적인 플로우룰 패킷 통계값 (OFPMP_FLOW)을 통해 계산된 평균 패킷 처리량과 해당 플로우룰에 설정된 Meter band 에 따라 Drop 된 패킷의 양을 비교했을 때, 일정시간 동안 빠르게 차이가 좁혀지는 경우 설정된 Meter band 의 Data rate 이 플로우의 QoS 를 지나치게 감소시킨다고 판단한다. 따라서 Dynamic Rate Limiter 는 기존에 설정된 Meter band 의 Data rate 을 OFPT_METER_MOD 메시지를 통해 증가시키고 이에 대한 정보를 컨트롤러로 전달해 네트워크 관리자에게 알린다. 이와 같은 과정들은 OpenFlow 1.3 에 소개된 Meter 테이블 기능을 적극 활용하여 이루어진다.

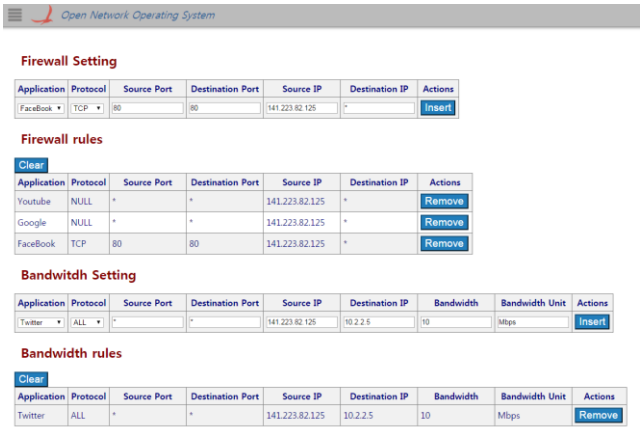


[그림.4] Bandwidth Manager 설계

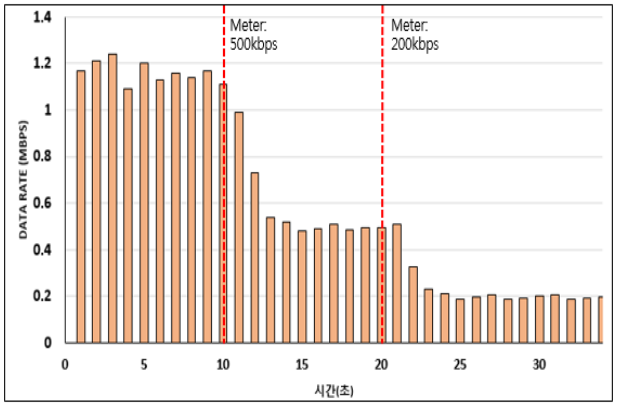
4. 시스템 검증

구현된 프로토타입의 성능평가를 위해 간단한 테스트 베드에서 Bandwidth Manager 의 Meter band 적용에 관한 실험을 진행하였다. 테스트베드는 Mininet 위에 구축된 Sender, Receiver 호스트가 OpenFlow 1.3 을 지원하는 소프트웨어 스위치를 토해 연결되어 있고, 스위치에는 nDPI 가 Stand-alone 으로 동작하여 Sender 와 Receiver 간에 트래픽을 분석한다. 또한 스위치는 제안된 어플리케이션이 동작하고있는 ONOS 컨트롤러와 연결되어 Bandwidth Manager 의 Data rate 정책을 포함하는 플로우룰이 설치된다. Sender 와 Receiver 사이에 FTP 세션을 만들고 Sender 에서 대용량 파일을 전송할 때 네트워크 관리자가 전송 시작 후 10 초 시점에서 Bandwidth Manager 를 통해 해당 세션의 Meter band 를 0.5 Mbps 로 제한하고 20 초 시점에서는 0.2 Mbps 로 설정하는 시나리오를 수행한다. 시나리오의 수행결과는 [그림.6]과 같다. 파일 전송이 시작되면 컨트롤러는 DPI 의 분석결과를 토대로 FTP 플로우를 식별하여 Sender 와 Receiver 사이의 스위치에 플로우룰을 설정한다. 전송 시작부터 10 초 간은 Data rate 의 제한이 없으므로 해당 플로우룰에 적용되는 Meter band 없이 대략 1.2Mbps 의 속도로 전송된다. 10 초 시점에 관리자가 Bandwidth Manager 를 통해 Sender 와 Receiver 사이의 FTP 플로우에 0.5 Mbps 의 Meter 를 적용하면, Receiver 에서는 대략 3 초 후에 0.5 Mbps 의 Data rate 을 측정할 수 있다. 마찬가지로 20 초 시점에서도 0.2 Mbps 의 Meter 가 적용될 때 변화되는 Data rate 을 확인할 수 있다. Meter 의 적용과 감지되는 Data rate 의 시간은 네트워크의 복잡도나 모니터링 프로그램에 따라 달라질 수 있다.

현재 OpenFlow Meter 의 구현은 설정된 Data rate 를 초과하는 패킷을 강제로 Drop 시키거나 IP 헤더의 DSCP 필드에 매칭하여 Drop 의 Precedence 를 조정하는 방식을 지원하고 있다 [11]. 본 논문에서는 강제 Drop 방식을 사용했는데, TCP 기반의 FTP 실험에서 Meter band 를 초과하는 Data rate 이 감지되는 시점 직후에 패킷 Drop 에 따른 Data rate 이 감소했다가 다시 Meter band 값 수준으로 증가하는 패턴을 확인할 수 있다.



[그림.5] 어플리케이션 Web UI



[그림.6] Meter band 적용에 따른 Data rate

5. 결론 및 향후 연구

본 논문에서는 SDN 에서 트래픽 관리를 위해 DPI 를 이용하는 ONOS 어플리케이션을 제안하였다. OpenFlow 기반 SDN 에서 네트워크를 구성하는 스위치가 새로운 플로우의 패킷을 수신할 경우 이에 대한 헤더 정보를 Packet-In 메시지를 통해 컨트롤러에게 전달하고, 컨트롤러는 해당 정보를 통해 패킷이 어느 경로를 통해 전달되어야 하는지를 결정한다. 본 논문에서는 이러한 특징을 바탕으로 Firewall 을 구현하였는데, 플로우의 헤더 정보를 관리자에 의해 미리 설정된 Firewall 룰과 비교하여 해당 플로우를 Drop 하여 Edge 스위치에서 차단할 것인지 또는 라우팅 경로를 설정하여 목적지로 전달 할 것인지를 결정한다. 컨트롤러가 새롭게 유입되는 패킷의 헤더 정보를 확인하는 것은 OpenFlow 기반 스위치와 컨트롤러간에 표준을 따르는 것이기 때문에 Firewall 은 이러한 정보를 손쉽게 제공받을 수 있다. OpenFlow 기반 SDN 의 기본적인 동작 방식에서는 컨트롤러가 전달받은 패킷을 통해 해당 플로우의 라우팅 경로를 결정하고 경로상의 스위치에 적합한 플로우룰을 설치하는 등의 수동적인 역할을 하는 것에 비해, 본 연구에서는 Firewall 을 동작 시킴으로써 해당 플로우의 네트워크 진입 여부를 네트워크 관리자가 제어하게 되어 네트워크의 보안성을 높일 수 있게 된다. 이와 유사하게, Bandwidth Manager 는 플로우의 헤더 정보와 Meter band 를 이용하여 특정 어플리케이션의 트래픽 단위로 Data rate 을 제어할 수 있다. 플로우의 헤더 정보만으로는 트래픽 분류가 어려울 경우 외부 DPI 프로그램의 트래픽 분석 결과를 활용할 수 있다.

본 논문에서 제안한 시스템의 주요기능이 검증을 통해 정상적으로 동작하는 것은 확인되었지만 아직까지는 프로토타입 수준이기 때문에 향후 개선되어야 할 여지가 있다. 특히 보다 크고 다양한 어플리케이션 트래픽이 발생하는 네트워크에서 관리자가 취할 수 있는 여러 시나리오를 통해 제안한 방식이 잘 동작하는지 확장성 측면에서 검증할 필요가 있다. 또한, Bandwidth Manager 에서 이용하는 Meter band 가 OpenFlow 1.3 버전 이상에서 지원되는데, 실제 환경에서 잘 동작하기 위해 해당 기능을 구현하는 소프트웨어 또는 하드웨어 스위치가 필수적이다.

6. 감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-15-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

7. 참고 문헌

[1] Medved, Jan, et al. "Opendaylight: Towards a model-driven sdn controller architecture." *2014 IEEE 15th International Symposium on*. IEEE, 2014.
[2] Ryu. http://osrg.github.io/ryu/.
[3] Floodlight. http://www.projectfloodlight.org/floodlight/.
[4] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS." *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014.
[5] Hu, Hongxin, et al. "FLOWGUARD: building robust firewalls for software-defined networks." *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014.
[6] Jarschel, Michael, et al. "Sdn-based application-aware networking on the example of youtube video streaming." *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. IEEE, 2013.
[7] Qazi, Zafar Ayyub, et al. "Application-awareness in SDN." *ACM SIGCOMM Computer Communication Review*. Vol. 43. No. 4. ACM, 2013.
[8] Mininet. http://mininet.org.
[9] Ofsoftswitch13 cpqd. https://github.com/CPqD/ofsoftswitch13.
[10] Deri, Luca, et al. "nDPI: Open-source high-speed deep packet inspection."*Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 2014.
[11] Mohan, Purnima Murali, Dinil Mon Divakaran, and Mohan Gurusamy. "Performance study of TCP flows with QoS-supported OpenFlow in data center networks." *Networks (ICON), 2013 19th IEEE International Conference on*. IEEE, 2013.

WSN Management Framework based on OpenFlow and NetConf Protocol

Gde Dharma Nugraha, Deokjai Choi

School of Electrical Engineering, Chonnam National University

Gwangju, South Korea

gdebig@gmail.com, dchoi@jnu.ac.kr

Abstract

Recent development of Wireless Sensor Network (WSNs) has improved the capability and decrease the size of the sensor node. The impact of this improvement is the large number of sensors that have to manage by the WSN Management. In the other hand, Software Defined Network (SDN) has emerged as the future technology of the network. SDN network management gets supported by OpenFlow Protocol and NetConf Protocol. In this paper, we propose our WSN Management Framework that incorporates with SDN and combine the OpenFlow Protocol and NetConf Protocol. Furthermore, we propose the generic architecture of the WSN Base Station and WSN Node that appropriate with our proposed WSN Management Framework.

Keywords – software-defined network; OpenFlow; NetConf; Wireless Sensor Network, Management; Configuration

1. Introduction

In recent years, the advancement in micro-electro-mechanic systems, wireless communication and digital electronics have developed Wireless Sensor Networks (WSNs) become powerful and smaller in size. This creates a smart node with wireless connectivity. This creation leads to the development of the Internet of Things (IoT), one of the emerging technology in future internet. Internet of Things connects the objects or the things in the physical world to their virtual representation in the internet.

The number of smart nodes that deployed for IoT may be huge. Although the hardware specification of smart node has increased significantly, the smart node for IoT still have limitations regarding to computing resources, memory size and power sources. For power source, smart nodes for IoT usually get powered by batteries. This has become challenges on management framework development for IoT.

WSN Management has two objects that have to manage such as network management and node management. Network management related to a series of activities, methods, procedures and tools required to operate a network [1]. Node management related to discovery, selection and configuration of smart nodes in the WSN. Most of state-of-the-art research in IoT Management [1]–[5], focus only to one object. It causes current IoT Management become less comprehensive.

Another emerging research that provides better network management system is Software Defined Network (SDN). SDN separates the control plane and data plane and enabling programming capability to control and monitor network data path. SDN simplifies the deployment of new protocols and applications. SDN has supported by many network management protocols. The mostly used currently are OpenFlow Protocol and NetConf Protocol. OpenFlow protocol is used to manage the network and NetConf Protocol is used to manage the network node configuration in SDN.

In this paper, we describe our initial concept and design of the Management Framework to incorporate WSN Technology and SDN to provide comprehensive WSN Management that have the capability to handle WSN Network and WSN Smart Nodes. We argue that using the OpenFlow protocol in SDN can provide a global view of the network. It will simplify the network management of WSN. Utilize NetConf Protocol in the SDN will helps to simplify WSN Smart Node configuration management. Using NetConf Protocol, we can monitor and reconfigure the smart nodes that already deployed.

The rest of this paper is organized as follows. In section II, we overview the capability of OpenFlow Protocol and NetConf Protocol in SDN. In section III, we discuss the proposed Management Framework for WSN based on OpenFlow and NetConf. Finally, in Section IV we present the conclusion and discuss future work.

2. Network Management Protocol in SDN

Currently, the most network management protocols that used in SDN are OpenFlow and NetConf. OpenFlow protocol offers the network management through flow table manipulation and NetConf protocol offers the network management through configuration management of large numbers of networked devices.

A. OpenFlow Protocol in SDN

OpenFlow protocol [6] is already adopts widely by the network for SDN. OpenFlow is vendor independent so OpenFlow can be implemented by using any network device from any vendor. Specification of OpenFlow (OF) describes an open protocol that allows software applications to program the flow table of different switches. OpenFlow consists of three main parts: An OpenFlow compliant switch, a secure channel and a controller. Figure 1 depicts the main component of OpenFlow. An OpenFlow Switch contains a flow table that used to handle the incoming packet. Each entry in flow table has matching field, counters and instructions. The incoming packet will be compared with the entry in the flow table. If there is a match entry, the packet will be handled based on the instruction contained by that entry. If there is no match entry, the switch will encapsulate and forward the packet to the controller. Counter is used to collect the statistics information of the packets.

A controller is a software program responsible for managing and manipulating OpenFlow Switch's Flow Table. Controller interact with OpenFlow Switches through a secure channel. Through secure channel, controller sends control message to the OpenFlow Switches, receive packets and send packets to the switches [1]. The controller manages the network by managing and manipulating OpenFlow Switches' flow tables. Packet coming into the controller when this packet has no match in the OpenFlow Switches' flow table. The controller then process the packet based on the application or rules defined by the network administrator or programmer. The result, then sends back to the switch and modify the OpenFlow Switches' flow table. OpenFlow Switches then will use this new flow table to handle next incoming packets.

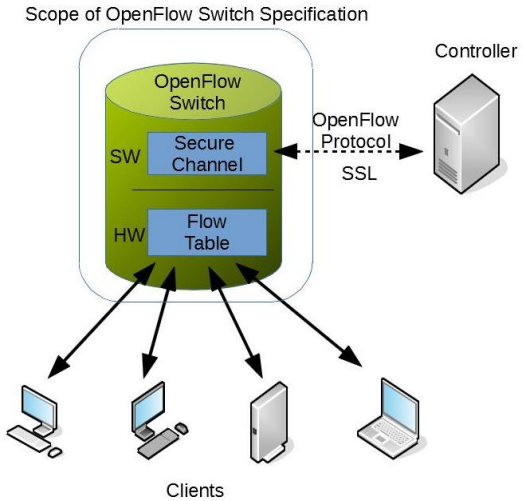


Figure 1. Components of OpenFlow Protocol [6]

The controller in OpenFlow Protocol provides the interface or an Application Programming Interface (API) that can be used by the network administrator to manage the network. Through this API, network administrator can collect the status of the network nodes through the packets that coming into the controller, implement early detection mechanism for any problem in the network, implement new policies or new routing to the network, and programming the controller to handle new node in the network. This capability of OpenFlow controller helps the network administrator to simplify network management of large scale network.

B. NetConf Protocol in SDN

NetConf Protocol, defined by RFC 6241 [7], is an IETF network management protocol that provides mechanisms to install, manipulate and delete the configuration of network devices. NetConf protocol provides the ability to write configuration data to a networked device and collect the data for a networked device. NetConf protocol uses the Remote Procedure Call (RPCs) to transmit the configuration data that encoded in Extensible Markup Language (XML). This transmission uses a secure and connection oriented protocol [8].

NetConf Protocol has a simple layered architecture that shows in figure 2. This layered architecture illustrates the mechanism of the NetConf Protocol to send or retrieve the configuration data for networked devices. As shown in figure 2, secure transport provides secure transport protocol, end-to-end connectivity and ensure reliable delivery of messages. On top of secure transport layer, lies a message layer that uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages. This message layer provides a mechanism for encoding of RPC calls and notifications. The operation layer provides specific operations to manipulate configuration state and content layer consists of configuration and state data which is XML-encoded. The operation layer and content layer use data modelling language to model configuration and state data manipulated, called YANG.

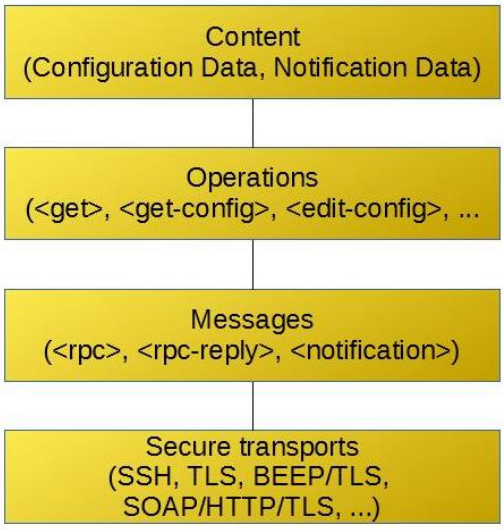


Figure 2. NetConf Protocol Layer

Figure 3 shows the scenario of NetConf Protocol deployment. This scenario uses the assumption that a network-wide configuration or policy system uses the NetConf protocol to push configuration changes to NetConf enabled devices. In this deployment NetConf Server and NetConf Client is used interchangeably. NetConf client

resides in NetConf policy manager and acting as policy decision point. This node will request the status of each client or deploy the configuration to each client through NetConf Server that resides in the NetConf enabled devices. NetConf server acting as a policy enforcement to the managed devices.

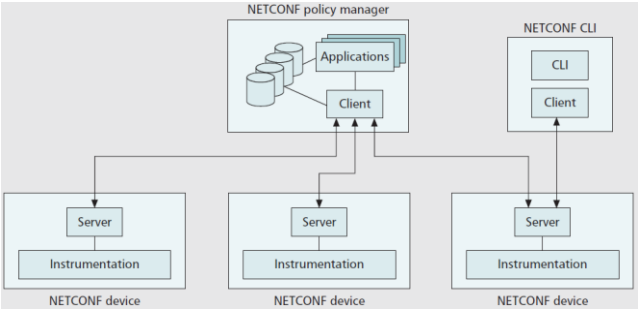


Figure 3. NetConf Deployment Scenario [8]

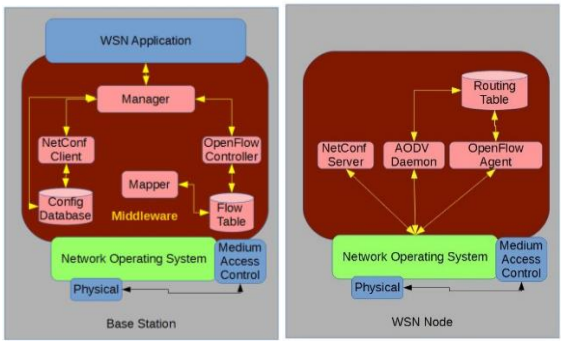
3. WSN Management Framework based on OpenFlow and NetConf

In the previous section, we have already describe the concept and mechanism of OpenFlow and NetConf protocol. Each protocol has their own function. OpenFlow Protocol focus on how to manage the network, meanwhile NetConf Protocol focus on managing the configuration of network node. These protocols can be adopted into the WSN Management Framework to provide comprehensive management that can manage the network and the node of WSN.

In this section, we propose our initial work of WSN Management that combine OpenFlow and NetConf protocol. We will show that our management framework has the comprehensive function to manage the network and node of WSN.

A. Management Framework based on OpenFlow and NetConf Protocol

In our proposed management framework, we defined the architecture of the WSN Base Station that act as the Controller and WSN Smart Node as shown in figure 4.



(a) (b)

Figure 4. (a) Architecture of WSN Base Station and (b) Architecture of WSN Node

WSN Base Station and WSN Node communicate each other using wireless medium. In WSN Base Station, on top of the Network Operating System, there are Middleware Layer and Application Layer. Middleware and Application Layer become our focus on WSN Base Station Architecture. In WSN Node, on top of Network Operating System, there is an application layer that running the modules that enabling WSN Node as SDN Node. Modules that running in the application layer of WSN Node such as NetConf Server, AODV Daemon, OpenFlow Agent and Routing Table.

1. Middleware of WSN Base Station
   Middleware Layer of WSN Base Station enabling the Base Station to act as SDN Controller. Inside the middleware layer, there are modules that provide the control for base station and simple interface for the Application Layer to define the flow table.
   - Manager
     This module is responsible for managing the network and the node of WSN. Since we have different protocol for network and node management, Manager Module gets supported by OpenFlow Controller and NetConf Client. From OpenFlow Controller, Manager Module will get information about network topology and state. From NetConf Client, Manager Module will get information about the configuration state of each node including resource status. That information then can be accessed by the Application when needed.
     Manager module also provides the interface for Application to interact with OpenFlow Controller and NetConf Client. Manager module responsible for translating the request from Application whether it is network management request or node management request. If the request is about network management, manager module will forward the request to OpenFlow Controller and if the request is about node configuration management, manager module will forward the request to NetConf Client. Manager module also can monitor the status of every node from config database.
     This function of the manager module allows easier deployment of new protocol and functionalities, network visualization and management, monitoring the state of the node and deployment of new node configuration.
   - OpenFlow Controller
     OpenFlow Controller is part of the OpenFlow Protocol in our management framework. OpenFlow Controller maintains the network of WSN through its Flow Table. OpenFlow Controller provides the global view of WSN Network to the Manager Module and update the flow table to change the network based on request application that's coming from manager module.
   - Flow Table
     Flow table module is also part of the OpenFlow

Protocol that responsible to store the flow tables defined through the OpenFlow Controller for every node of WSN. At the first time, Flow Table is updated by Mapper module and after that, this flow table only can be updated by the command from the OpenFlow Controller. OpenFlow Controller treats every incoming packet based on the rules in the flow table.

- NetConf Client
  The NetConf Client module is part of the NetConf protocol that responsible to manage the configuration of the node. NetConf Client monitors the configuration status of every node through config database. If there are configuration change request come from the manager module, NetConf Client will set the appropriate node configuration and send it to every node through NetConf Protocol.
- Config Database
  The Config Database module is part of the NetConf Protocol that responsible to store the configuration status of every node in WSN. This Config Database only can be updated by the NetConf Client. However, it can be accessed by the manager to provide the status information of every node to the manager module.
- Mapper
  Mapper module responsible in the initial step when the WSN is deployed in the first time. Mapper implements routing protocol of WSN to find out a path from base station to every node. After finding out the path for every node in the WSN, the mapper will update the Flow Table module. From this information, Flow Table will contain the current path or rules for every node and monitor these rules for maintaining the network.

2. Application Layer of WSN Base Station
   Application layer was designed for network administrator to specify the functionality of the WSN. With application layer, network administrator can monitor and update the network of WSN and also configure the node base on the needs.
3. Application Layer of WSN Node
   Application Layer of WSN Node was designed to respond any command or request that comes from WSN Base Station. In Application Layer of WSN Node, there are module such as OpenFlow Agent, AODV Daemon, NetConf Server and Routing Table. The function of each module as follows.
   - OpenFlow Agent
     OpenFlow Agent in every node of WSN is needed to enable communication between WSN Node to the WSN Base Station. This module is part of OpenFlow Protocol. OpenFlow Agent responsible to forward the packet based on rules in its routing table or update the rules based on instruction coming from the OpenFlow Controller of WSN Base Station.
   - Routing Table
     The Routing Table module in WSN Node is responsible to stores the rules related to the node only. This routing table is used to define the forwarding path that out of the node. This routing table only can be updated by the OpenFlow Agent based on instruction from the OpenFlow Controller of WSN Base Station.
   - AODV Daemon
     AODV daemon is the module that responsible to implement a routing protocol at the initial step when the WSN just deployed. This daemon implements an AODV routing algorithm to find out the path from the node to the WSN Base Node. After finding out the path from the node to the WSN Base Node, this module will update its own routing table and maintain it.
   - NetConf Server
     NetConf Server is part of the NetConf protocol that responsibility to serve the request that comes from the NetConf Client module of the WSN Base Station. NetConf server is the execution part of NetConf Protocol. If the NetConf Client module of the WSN Base Station request for any configuration state of the node, NetConf Server will collect the configuration state of the node and send it to the NetConf Client of WSN Base Station. If there is any request from the NetConf Client of WSN Base Station to change the configuration of the node, NetConf Server will respond it, accept the changes from NetConf Client and execute it to update the node configuration.

B. Scenario of WSN Management Framework

Figure 5. shows the scenario of our WSN Management Framework.

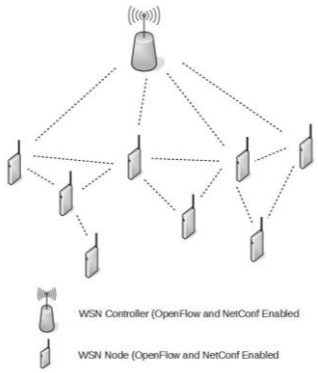


Figure 5. Scenario of WSN Management Framework

As depicted in figure 5, our scenario consists of the WSN Base Station that act as WSN Controller and WSN Node. At the first deployment of the WSN, each WSN Node will implement an AODV routing algorithm to find out the path to the WSN Base Station. This action is running by AODV Agent module for each WSN Node and Mapper in WSN Base Station. After the path from every WSN node to WSN Base Station is established, each routing table of every node is updated to stores the rules of each node. Flow Table in WSN Base Station is updated to get the global view of the WSN Network.

Then the NetConf Server of every node sends the configuration state of each node to update the Configuration Database in WSN Base Station. Now Configuration Database in WSN Base Station has the information about initial configuration for every node in the WSN. After that, manager module of the WSN Base Station takes over the jobs to monitor and maintain the network and the node of the WSN.

4. Conclusion and Future Works

This paper proposed our initial effort to develop a comprehensive WSN Management Framework that manages the network and the node of WSN. In our framework, we combine OpenFlow Protocol and NetConf Protocol from the SDN. With support by OpenFlow Protocol, our framework can manage the network of SDN. Utilize NetConf Protocol, our framework can manage the configuration management of every node in the WSN. We also provide the architecture of the WSN Base Station and WSN Node to support our management framework.

Our initial effort has shown that the combination of OpenFlow Protocol and NetConf Protocol has the possibility to provide a comprehensive IoT Management Framework that did not only focus on the Network Management but also Node Configuration Management. However, the efficiency and the impact of this combination into the overall performance of the WSN needs further research and investigation. This is our future work to develop the real test bed for the proposed framework to evaluate the efficiency and the impact to the performance of the WSN.

5. References

[1] A. Lara, "Simplifying network management using Software Defined Networking and OpenFlow," *Adv. Networks ...*, pp. 24–29, 2012.

[2] A. S. Yuan, H.-T. Fang, and Q. Wu, "OpenFlow Based Hybrid Routing in Wireless Sensor Networks," *Intell. Sensors, Sens. Networks Inf. Process. (ISSNIP), 2014 IEEE Ninth Int. Conf.*, vol. 1, no. April, pp. 21–24, 2014.

[3] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *2014 27th Bienn. Symp. Commun.*, pp. 71–75, 2014.

[4] L. Ruiz, J. M. Nogueira, and A. A. F. Loureiro, "MANNA: a management architecture for wireless sensor networks," *IEEE Commun. Mag.*, vol. 41, no. 2, pp. 116–125, 2003.

[5] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE : Design , prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *2015 IEEE Conf. Comput. Commun.*, pp. 513–521, 2015.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69, 2008.

[7] R. Enss, M. Bjorklund, J. Schoenwaelder, and A. Biearman, "Network Configuration Protocol (NETCONF)," no. 1, pp. 1–5, 2011.

[8] J. Schonwalder, M. Bjorklund, and P. Shafer, "Network configuration management using NETCONF and YANG," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 166–173, 2010.

정형 검증과 SDN 을 활용한 네트워크 데이터 플레인 검증 기술 소개

손정석, 문수복

카이스트 전산학부

{jeongseok, sbmoon}@an.kaist.ac.kr

요 약

컴퓨터 네트워크가 발전하고 클라우드 컴퓨팅이 도입되면서 점점 더 많은 서비스들이 네트워크를 통해서 사용자에게 전달되고 있다. 이에 따라 네트워크의 규모는 커지고 복잡도는 증가하고 있다. 하지만 네트워크의 오류를 찾아내는 검증 과정은 여전히 네트워크 관리자의 경험에 기반해서 수동적으로 이뤄지고 있다. 이에 대한 해결 방법으로 네트워크의 데이터 플레인을 검증해서 관리자가 설정한 규칙들을 만족하는지 자동으로 알려주는 방법들이 고안되었다. 이 논문에서는 최근에 나온 주요 데이터 플레인 검증 연구들을 핵심 아이디어에 따라서 정형 검증을 이용한 연구와 SDN 을 활용한 연구로 나누어 소개한다. 그리고 데이터 플레인 검증 분야에서 아직 해결되지 않은 문제들을 살펴보고 검증의 다음 단계에 있는 연구 분야로 네트워크 합성 분야를 제시한다.

1. 서론

오늘날 대부분의 서비스들은 컴퓨터 네트워크를 통해 사용자에게 제공된다. 최근 클라우드 컴퓨팅의 발전으로 인해 점점 더 많은 서비스들이 네트워크를 통해 연결된 서버로 옮겨가고 있다. 따라서 네트워크의 안정성이 서비스를 운영하는데 있어서 아주 중요한 요소가 되었다. 하지만 네트워크의 오류를 진단하는 작업은 여전히 네트워크 관리자의 경험과 직관에 의존하고 있다. 또한 대부분의 검사 과정이 문제가 발생한 이후에 이루어지고 있으며 문제의 원인을 찾기 위해 사용하는 도구들도 Ping, Traceroute 같이 몇 가지 기본적인 것들로 제한되어 있다. [2]

이런 문제점을 해결하기 위해서 라우팅 프로토콜과 네트워크 장비의 설정 파일을 검증하는 연구가 진행되어왔다. 그러나 설정 파일 검증으로는 찾아낼 수 있는 오류의 범위가 제한적이며 설정 문법에 의존적이라 다양한 장비로 구성된 네트워크 전체를 검증하기에 적합하지 않다. 때문에 설정 파일의 검증만으로는 실제 네트워크의 안정성을 보장하기는 어려웠다.

그러다 SDN(Software-defined Networking)의 등장으로 네트워크의 컨트롤 플레인과 데이터 플레인을 명확하게 구분하기 시작하면서 컨트롤 플레인이 아니라 데이터 플레인을 검증하는 연구들이 시작되었다. 최근 몇년 간 네트워크 검증분야에서는 프로그래밍 언어 분야의 정형 검증(Formal Verification)을 도입하거나 SDN 의 장점을 활용해 데이터 플레인을 검증하는 연구들이 활발하게 진행이 되었다. 그 중 몇 가지는 학계를 넘어 산업계의 실제 네트워크에 이미 적용이 되고 있을 정도로 데이터 플레인 검증은 네트워크를 관리하고 운영하는 방식을 빠르게 바꿔나가고 있다.

이 논문은 네트워크 데이터 플레인 검증이 무엇인지 자세하게 소개한다. 그리고 주요 데이터 플레인 검증 논문들의 핵심 아이디어를 분석하고 이를 분류해서 최근 네트워크 검증 분야의 큰 흐름을 보여준다. 또한 아직 기존의 논문들이 해결하지 못한 중요한 문제들을 짚어보고 데이터 플레인 검증의 다음 단계로 진행되고 있는 연구 분야들을 통해서 앞으로의 연구 방향을 제시하고자 한다.

2. 네트워크 데이터 플레인 검증 소개

네트워크에서 데이터 플레인은 각각의 스위치에서 패킷이 실제로 어떻게 포워딩 될지를 결정하는 부분을 말한다. 레이어 2 의 스위칭과 레이어 3 의 IP 포워딩이 대표적으로 데이터 플레인에 속한다.

2.1 데이터 플레인 검증이란

데이터 플레인 검증은 데이터 플레인 정보를 특정 시점에 스냅샷을 찍어서 이를 정적으로 분석해서 오류를 검증하는 방식을 말한다. 대표적인 데이터 플레인 정보로는 IP 포워딩 테이블, MAC 주소 테이블, ARP 테이블, LLDP 정보와 방화벽에서 트래픽을 필터링하기 위해 사용하는 ACL(Access Control List)이 있다. 그림 1 은 일반적인 데이터 플레인 검

증 과정을 나타낸다.

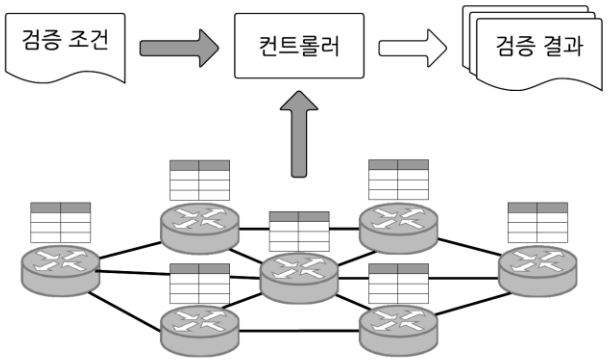


그림 1. 데이터 플레인 검증 과정

데이터 플레인을 검증하기 위해서 데이터 플레인 정보의 스냅샷을 먼저 얻는다. 스냅샷을 얻기 위해서 일반적인 네트워크에서는 스위치에서 지원하는 콘솔 도구에 원격으로 접속해서 포워딩 테이블이나 ACL 을 출력하는 명령어를 실행하는 방법을 주로 사용한다. SDN 처럼 컨트롤러가 존재하는 네트워크라면 컨트롤러에서 API 를 이용해 스위치로부터 데이터 플레인 정보를 쉽게 얻을 수 있다.

관리자는 데이터 플레인 검증 도구를 이용해서 네트워크에서 반드시 만족해야 할 조건들을 설정할 수 있다. 대부분의 검증 도구들이 지원하는 주요한 검증 조건들은 아래와 같다.

- 도달가능성(Reachability): 어떤 호스트 A 에서 보낸 패킷이 특정한 호스트 B 에 도달할 수 있는가? 혹은 도달할 수 없도록 되어있는가?
- 순환 포워딩 존재 여부(Forwarding Loop): 네트워크에 들어왔을 때 순환하게 되는 패킷 집합이 있는가?
- 네트워크의 분리(Isolation): 네트워크의 특정 부분이 다른 부분과 완전하게 분리되어 있는가?

도달가능성은 네트워크에서 항상 서로 통신이 가능해야 하거나 혹은 항상 가능하지 않아야 하는 노드의 조건을 설정하는데 사용할 수 있다. 예를 들어 네트워크를 모니터하는 서버는 다른 네트워크 장비들에 항상 접근 가능해야 할 것이다. 순환 포워딩은 네트워크의 성능을 저하시키고 DDoS(Distributed Denial of Service) 공격의 증폭기(Amplifier)로 사용되어 보안 취약점이 될 수 있으므로 반드시 발견해 없애줘야 한다.[3] 네트워크의 분리는 VLAN 이나 다중 고객(Multi-tenant) 환경에서 사용되는 VXLAN 세그먼트들이 서로 완전히 분리가 되어있는지를 검증할 때 사용한다.

2.2 컨트롤 플레인 검증과 비교

데이터 플레인 검증 연구 이전에는 주로 컨트롤 플레인에서 네트워크 오류를 잡아내는 연구가 대부분이었다. BGP 프로토콜 설정 파일 검증[4]과 특정 네트워크 장비의 설정을 검증[5]하는 방식이 컨트롤 플레인 검증에 속한다.

데이터 플레인 검증은 스위치의 라우팅 프로토콜 프로세스가 결과값으로 내놓은 포워딩 정보를 검증하는 방식인 반면에 컨트롤 플레인 검증은 라우팅 프로세스의 입력으로 들어가는 설정값을 검사한다. 컨트롤 플레인 검증은 유저가 입력한 설정을 검사해서 오류의 원인이 되는 잘못된 설정 부분을 정확히 알려줄 수 있다는 장점이 있다. 하지만 각각의 검증법이 라우팅 프로토콜이나 장비의 설정 파일 문법에 종속적이고 스위치 소프트웨어 자체에 버그가 있어서 데이터 플레인에 문제가 생기는 경우를 잡아낼 수 없다. 반면 데이터 플레인 검증은 컨트롤 플레인보다 실제 스위칭이 일어나는 하드웨어에 더 가까운 레이어에서 검증을 하므로 스위치 소프트웨어의의 버그로 인해 데이터 플레인에 오류가 생긴 경우도 발견할 수 있다. 또한 데이터 플레인 정보로 쓰이는 포워딩 테이블이나 ACL 은 라우팅 프로토콜과 장비의 모델이나 회사에 상관없이 비교적 동일한 형태를 갖는다. 네트워크는 일반적으로 서로 다른 회사에서 만든 여러 종류의 장비들로 이루어져 있기에 네트워크 전체를 검사하는데에는 데이터 플레인 검증이 더 적합하다.[6]

3. 데이터 플레인 검증 기술 분석

최근에 제안된 데이터 플레인 검증 기술들은 프로그래밍 언어의 정형 검증 기술을 네트워크 검증에 도입한 연구들과 SDN 의 장점을 활용한 연구들로 크게 두 가지로 분류 할 수 있다.

3.1. 정형 검증을 이용한 네트워크 검증

프로그래밍 언어 분야에서의 정형 검증은 검증하고자 하는 시스템과 그 시스템에서 항상 만족해야 하는 조건들을 논리식과 같은 수학적인 방법으로 기술한 후에 시스템의 모든 상태에 대해서 조건들이 항상 만족하는지를 검사하는 방법이다. 일반적인 범용 소프트웨어나 하드웨어 회로를 검증하는데 사용하는 정형 검증 기법을 네트워크 데이터 플레인을 검증하는데 처음 도입한 논문이 Anteater[7] 이다.

Anteater 는 제한된 모델 검사(Bounded Model Checking)를 통해 네트워크를 검증한다. 네트워크 데이터 플레인 검증을 논리식을 만족하는 해를 찾는 과정으로 치환한 것이 논문의 핵심 아이디어이다. Anteater 는 네트워크의 포워딩 규칙들과 네트워크가 만족해야 하는 조건들을 논리식으로 표현한 후에 SAT(Boolean satisfiability) Solver 를 이용해 이 조건들을 만족할 수 있는 해가 존재하는지 검사한다.

A의 포워딩 테이블

도착 IP 주소	전달 동작
1.1.1.0/24	B로 전달
2.2.2.0/24	B로 전달
2.2.2.0/30	전달하지 않음

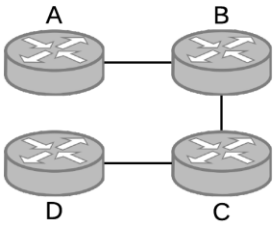


그림 2. 라우터 4 개로 구성된 네트워크 예시

Anteater 에서는 한 스위치에서 연결되어 있는 다른 스위치로 포워딩할 수 있는 패킷의 조건을 포워딩 정책 함수로 정의한다. 그림 2 의 네트워크의 라우터 A 에서 B 로 보낼 수 있는 패킷을 정의하는 정책 함수 $\mathcal{P}$ 는 다음과 같이 논리식으로 표현할 수 있다.

$$\mathcal{P}(\mathrm{A},\mathrm{B}) = (dst_{ip} \in 1.1.1.0/24 \vee dst_{ip} \in 2.2.2.0/24\ ) \wedge dst_{ip} \notin\ 2.2.2.0/30$$

이와 같이 인접한 스위치 사이의 포워딩 규칙을 함수로 정의해놓고 이를 네트워크가 만족해야하는 조건을 검증하는데 사용한다. 그림 2 에서 라우터 A 에서 라우터 D 로 패킷을 보낼 수 있는지 도달가능성을 검증하려면 다음과 같은 논리식을 만족하도록 하는 해가 있는지 검사해서 검증 할 수 있다.

$$\mathcal{P}(\mathrm{A},\mathrm{B}) \wedge \mathcal{P}(\mathrm{B},\mathrm{C}) \wedge \mathcal{P}(\mathrm{C},\mathrm{D})$$

Anteater 는 이 조건을 만족하는 해를 찾을 때 이미 상용으로 나와 있는 SAT Solver[1]를 활용한다. SAT Solver 를 이용해 논리식으로 변환된 조건을 만족하는 해를 찾은 경우 해당 조건이 네트워크에서 성립함을 확인할 수 있다. Anteater 에서는 도달가능성 뿐만 아니라 순환 포워딩 경로가 존재하는지, 포워딩 블랙홀이 존재하는지, 임의의 두 패킷 집합이 동일한 네트워크 장비들을 거치는지를 SAT 논리식으로 표현해서 검사하는 방법을 제안했다. 그리고 178 개의 라우터로 구성된 캠퍼스 네트워크에 적용해서 23 개의 오류를 찾아냈다. 같은 네트워크에서 순환 포워딩 존재 여부를 검증하는데 340 초 정도가 걸렸다.

정형 검증과 관련된 또 다른 대표적인 연구는 기호 실행(Symbolic Execution)을 네트워크 검증에 적용한 Header Space Analysis (HSA)[8] 이다. HSA 는 프로토콜에 따른 패킷 헤더의 필드 구분은 무시하고 단순한 0 과 1 의 조합으로 취급한다. 그리고 각각의 스위치는 이 헤더를 입력값으로 받아 포워딩 규칙에 따라서 적절한 연산을 헤더에 적용해 결과 헤더 값을 내놓는 함수로 모델링한다. 예를 들어 터널링을 할 때 사용하는 패킷 캡슐화(Encapsulation)는 헤더에 대한 이동(Shift) 연산으로 정의할 수 있다. 기하학적으로 보면 패킷 하나를 헤더 값에 따라 공간상의 점으로 표현할 수 있고 스위치는 이 점을 다른 점으로 변환하는 함수(Transfer function) T로 다음과 같이 정의할 수 있다.

$$T(h,p)\colon (h,p) \rightarrow \{(h1,p1),(h2,p2),\dots\}$$

h는 패킷 헤더의 값이고 p는 패킷이 도달한 스위치의 포트 번호이다. 결과 집합의 $h1, h2$ 은 스위치에서 적용하는 연산에 따른 결과 헤더 값이며 스위치가 헤더를 수정하지 않는 경우는 입력 헤더h와 같다. $p1, p2$와 같은 결과 집합의 포트 번호는 포워딩되는 패킷의 출력 포트 번호이다. 여기서 결과 값이 여러 개의$(header, port)$를 포함하는 집합인 이유는 멀티캐스팅을 모델링하기 위해서이다.

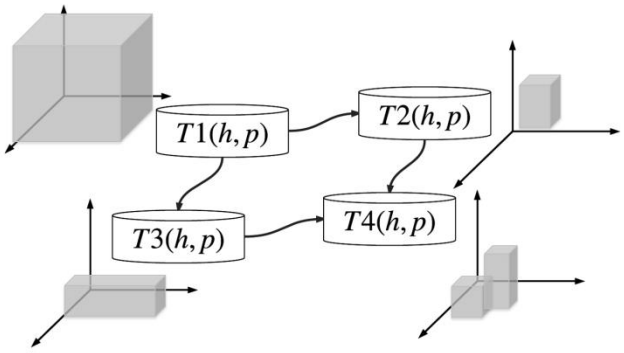


그림 3. Header Space Analysis 를 통한 도달 가능성 분석

네트워크를 기하학적인 모델로 표현한 결과 도달가능성, 순환 포워딩 존재 여부 등을 프로토콜에 상관 없이 검증할 수 있다. 그림 3 은 HSA 로 네트워크에 연결된 단말 노드들 간에 주고 받을 수 있는 패킷 집합이 있는지 검증하는 방법을 보여준다. 호스트에서 보낼 수 있는 모든 패킷을 고려하기 위해서 시작 호스트가 연결된 스위치의 변환 함수 T에는 헤더 공간 전체를 입력 값으로 넣는다. 그러면 함수 T의 결과는 스위치의 연산에 따라 변경된 헤더 값과 출력 포트 값의 쌍으로 나오게 된다. 이를 다시 링크로 연결된 인접한 스위치의 변환 함수에 넣어 단말 호스트에 도달할 때 까지 반복한다. 이 과정을 반복해 도달하는 포트에 연결된 단말 호스트들이 도달 가능한 단말 호스트 집합이 된다. 이 과정을 수식으로 표현하면 아래와 같다.

$$T_n(T_{n-1}(T_{n-2}(\dots(T_1(h,p))\dots)))$$

HSA 는 도달가능성, 순환 포워딩이 존재하는지 여부와 네트워크가 올바르게 분리되어 있는지를 검증하는 방법을 제안했다. 이를 20 여개의 스위치로 구성된 캠퍼스 네트워크 데이터 플레인의 스냅샷을 얻어서 분석했으며 10 분 이내에 모든 순환 포워딩을 찾아냈다.

3.2. SDN 을 적용한 네트워크의 검증

SDN 네트워크에서 데이터 플레인 검증을 하는 경우에는 다음과 같은 장점을 활용할 수 있다. 첫번째로 OpenFlow[9]와 같이 잘 정의된 인터페이스를 이용해 네트워크 장비 회사에 상관 없이 동일한 API 로 같은 형태의 데이터 플레인 정보를 얻을 수 있다. 또한 컨트롤러에서 데이터 플레인에 대한 업데이트를 모두 관찰할 수 있어 데이터 플레인의 변화된 부분을 쉽게 추적할 수 있다.

VeriFlow[10] 는 이런 SDN 의 특성을 데이터 플레인 검증에 활용한 대표적인 연구이다. VeriFlow 는 실시간으로 데이터 플레인을 검증하기 위해서 개별적인 포워딩 규칙 업데이트 이벤트를 관찰해서 변경된 규칙에 의해 영향을 받는 패킷만 검사를 한다. 이를 통해서 검사해야 하는 패킷 집합의 개수를 최소화해서 검증에 걸리는 시간을 크게 줄였다.

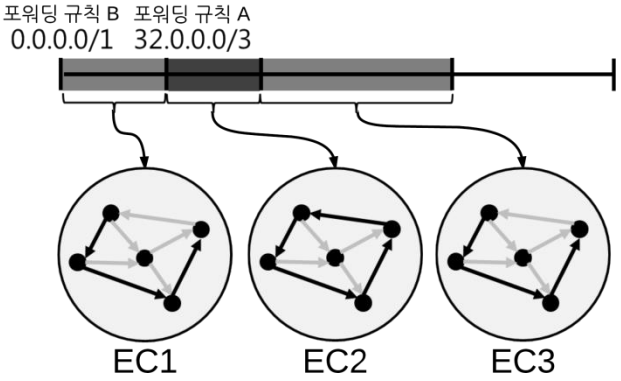


그림 4. VeriFlow 의 Equivalent Class 와 포워딩 그래프

VeriFlow 는 네트워크의 전체 포워딩 규칙 정보를 트리 구조에 저장해놓는다. 이를 이용해서 포워딩 룰 업데이트 정보가 주어졌을 때 업데이트에 영향을 받는 패킷 집합을 빠르게 계산할 수 있다. 그림 4 은 네트워크에 새로운 포워딩 규칙이 추가될 때 VeriFlow 가 어떻게 영향을 받는 패킷을 찾아 내는지 보여준다. 패킷 헤더의 도착 IP 주소가 32.0.0.0/3 과 일치하면 이를 포워딩하는 규칙이 있는 네트워크에 0.0.0.0/1 과 일치하면 포워딩하는 새로운 규칙이 삽입되는 상황을 가정하자. VeriFlow 는 컨트롤러에서 이벤트를 감지해서 이 업데이트에 의해 영향을 받는 패킷을 업데이트 정보로 알 수 있다. 그리고 트리를 탐색해서 저장해 놓은 규칙 중에서 새로운 규칙과 겹치는 영역이 있는 규칙을 찾아낸다. 그림 4 의 경우 기존의 32.0.0.3/3 대역의 패킷을 포워딩하는 규칙이 겹치는 규칙이 된다.

여기서 VeriFlow 는 네트워크에서 동일한 포워딩 경로를 따르는 패킷 집합을 빠르게 분류해내는 방법을 제안했다. 그림 4 에서 겹치는 두 포워딩 규칙의 상한과 하한에 따라서 IP 영역을 구분하면 총 3 개의 영역이 나오게 된다. 각각의 구분된 영역에 해당하는 모든 패킷들은 정확히 동일한 포워딩 규칙을 적용받는다는 사실을 알 수 있다. 논문에서는 각각의 패킷 집합을 EC(Equivalence Class)로 정의했다. 예를 들어 그림 4 의 EC1 과 EC3 에 해당하는 패킷들은 모두 포워딩 규칙 B 만 적용을 받는다. EC2 에 속하는 모든 패킷은 두 포워딩 규칙이 다른 장비에 삽입되었다면 규칙 A, B 둘 다 적용받게 된다. 이렇게 각각의 EC 들이 적용받는 포워딩 규칙을 모두 찾아낸 후에는 규칙들이 삽입된 네트워크 장비를 노드로 하고 포워딩 규칙의 다음 노드(Next hop)를 잇는 간선을 그려 포워딩 그래프를 구성한다. VeriFlow 는 이 그래프를 이용해서 루프가 있는지, 한 노드에서 다른 특정 노드에 도달할 수 있는 경로가 있는지 등을 그래프 탐색 알고리즘으로 빠르게 검사할 수 있다. VeriFlow 를 172 개의 라우터로 구성된 네트워크를 시뮬레이션해서 실험을 한 결과 업데이트 이벤트를 검증하는데 평균적으로 0.38ms 가 걸렸고 97.8%의 업데이트를 1ms 이내에 검증해냈다.

4. 해결해야 할 문제들과 미래 연구 방향

데이터 플레인 검증 기술들은 최근 몇년간 학계를 넘어 이미 산업계의 네트워크를 안정적으로 운용하는데 활용을 하고 있을 정도로 활발하게 연구가 이루어졌다. 하지만 기존의 연구들이 아직 제대로 해결하지 못한 중요한 문제들이 남아있다.

첫번째로 상태 정보를 활용하는(Stateful) 네트워크를 검증하는 문제가 있다. 방화벽이나 NAT(Network Address Translation)과 같은 네트워크 장비들은 이전에 처리한 패킷에 따라 현재 패킷에 대한 처리 결과가 달라진다. 이런 장비가 포함된 네트워크는 패킷을 처리하면서 포워딩 규칙이 실시간으로 변하게 되는데 데이터 플레인의 스냅샷을 분석하는 기존의 연구들은 이를 반영해서 처리하기 어려운 점이 있다.[11]

두번째로 큰 규모의 네트워크를 검증할 때 생기는 확장성(Scalability) 문제가 있다. 정형 검증은 시스템의 모든 상태에 대해서 완전한 검증을 할 수 있다는 장점이 있지만 큰 규모의 시스템에서는 검증에 걸리는 시간이 기하 급수적으로 늘어나서 사용하기 어렵다. 따라서 이를 활용한 네트워크 검증 기술들도 데이터 센터 네트워크 같은 대규모의 네트워크를 검증하는데에는 적절하지 않다.

아직은 이러한 문제들을 해결하는 초기 단계에 있으며 올해에 들어서야 연구 결과들이 나오기 시작한 상황이다. 상태 정보를 활용하는 네트워크를 검증하기 위해 FSM(Finite-state Machine)을 이용한 연구[12]가 2016 년 NSDI 학회에 출판되었고 대규모의 데이터 센터 네트워크를 검증하기 위해 데이터 센터 네트워크의 대칭적인 특성을 활용한 연구[13]가 올해 초 프로그래밍 언어 이론에 관련된 학회인 POPL 에 발표됐다.

네트워크의 데이터 플레인에 대한 검증의 다음 단계로 네트워크 합성(Network Synthesis)에 대한 연구가 활발하게 이루어지고 있다. 검증은 설정한 조건을 위반하는 데이터 플레인의 오류들을 수동적으

로 잡아내는 방법이지만 네트워크 합성은 사용자가 정한 고 수준의 정책에 맞춰서 네트워크의 데이터 플레인을 자동으로 설정해주는 접근법이다. 사용자가 고급 언어(High-level language)로 기술한 네트워크 정책을 데이터 플레인의 포워딩 규칙으로 변환하고 유지하기 위해서 컴파일러 기술을 도입한 논문[14]이 발표됐고 SDN 을 활용해 네트워크 정책을 입력받아 적절한 스위치들에 포워딩 규칙을 삽입하는 방법을 제안한 연구[15]도 몇년 전 발표됐다. 하지만 여전히 사용자가 네트워크 정책을 자유롭게 표현하면서도 합성에 걸리는 시간을 줄이기 위해 추가적인 연구가 필요한 상황이다. 또한 정책에 따라서 여러 대의 스위치를 동시에 업데이트 하면서 발생하는 분산 시스템에서의 일관성(Consistency) 문제를 해결하기 위해서도 더 많은 연구가 진행되어야 한다. 이러한 문제들을 해결하기 위해 다른 분야의 기술들을 네트워크 분야에 도입하는 것은 흥미로운 연구 주제가 될 것이다. 예를 들어 성능의 최적화를 위해서 컴파일러의 최적화 기술을 도입하고[16] 일관성 문제를 해결하기 위해 분산 시스템에서 사용되는 방법을 도입하는 연구[17, 18]들이 최근 출판되고 있다.

5. 결론

최근 네트워크 검증 분야에서는 패킷을 포워딩하는 데이터 플레인을 검증하는 여러가지 방법들이 제안되었다. 데이터 플레인 검증 분야의 주요 연구들은 크게 정형 검증을 도입한 연구와 SDN 을 활용한 연구로 나눌 수 있다. Anteater 와 Header Space Analysis 는 각각 정형 검증 기법인 제한된 모델 검사와 기호 실행을 도입해 데이터 플레인을 수학적으로 검증했다. VeriFlow 는 SDN 네트워크의 컨트롤러와 데이터 플레인에 접근할 수 있는 인터페이스를 활용해서 실시간으로 데이터 플레인을 검증해냈다. 많은 연구가 진행됐지만 상태 정보를 활용하는 네트워크와 데이터 센터와 같은 거대한 규모의 네트워크를 어떻게 빠르게 검증할 것인지는 해결되지 않은 문제로 있다. 검증의 다음 단계는 네트워크의 정책을 데이터 플레인에 자동으로 적용해주는 네트워크 합성이 될 것이다. 이 분야에는 프로그래밍 언어, 컴파일러, 분산 시스템의 기술들을 네트워크 분야로 가져와서 풀어볼만한 여러 문제들이 있다. 네트워크 합성 분야는 연구자들에게 네트워크와 다른 분야를 융합하는 새로운 연구를 해볼 수 있는 기회가 될 것이다.

6. 참고 문헌

[1] Boolector, http://fmv.jku.at/boolector/

[2] H. Zeng, P. Kazemian, G. Varghese and N. McKeown, “Automatic Test Packet Generation”, *in Proc. ACM CoNEXT*, pp. 241-252, 2012.

[3] J. Xia, L. Gao and T. Fei, “Flooding Attacks by Exploiting Persistent Forwarding Loops”, *in Proc. ACM/USENIX IMC*, pp. 385-390, 2005.

[4] N. Feamster and H. Balakrishnan, “Detecting BGP Configuration Faults with Static Analysis”, *in Proc. USENIX NSDI*, pp.43-56, 2005.

[5] L. Yuan, J. Mai, Z. Su, H. Chen, CN. Chuah and P. Mohapatra, “FIREMAN: A Toolkit for FIREwall Modeling and Analysis”, *in Proc. IEEE S&P*, pp. 199-213, 2006.

[6] A. Feldmann and J. Rexford, “IP Network Configuration for Intradomain Traffic Engineering”, *IEEE Network*, vol. 15 issue. 5, pp.46-57, Sep. 2001.

[7] H. Mai, A. Kurshid, R. Agarwal, M. Caesar, P. B. Godfrey and S. T. King, “Debugging the Data Plane with Anteater”, *in Proc. ACM SIGCOMM*, pp. 290-301, 2011.

[8] P. Kazemian, G. Varghese and N. McKeown, “Header Space Analysis: Static Checking for Networks”, *in Proc. USENIX NSDI*, pp. 113-126, 2012.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, *ACM SIGCOMM CCR*, vol. 38 issue. 2, pp. 69-74, Apr. 2008.

[10] A. Kurshid, X. Zou, W. Zhou, M. Caesar and P. B. Godfrey, “VeriFlow: Verifying Network-Wide Invariants in Real Time”, *in Proc. USENIX NSDI*, pp. 15-27, 2013.

[11] A. Panda, K. Argyraki, M. Sagiv, M. Schapira and S. Shenker, “New Directions for Network Verification”, *in Proc. SNAPL*, pp. 209-220, 2015.

[12] S. K. Fayaz, T. Yu, Y. Tobioka, S. Chaki and Vyas Sekar, “BUZZ: Testing Context-Dependent Policies in Stateful Networks”, *in Proc. USENIX NSDI*, pp. 275-289, 2016.

[13] G. D. Plotkin, N. Bjorner, N. P. Lopes, A. Rybalchenko and G. Varghese, “Scaling Network Verification using Symmetry and Surgery”, *in Proc. ACM SIGPLAN-SIGACT POPL*, pp. 69-83, 2016.

[14] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese and D. Walker, “P4: Programming Protocol-independent Packet Processors”, *ACM SIGCOMM CCR*, vol. 44 issue. 3, pp. 87-95, Jul. 2014.

[15] N. Kang, Z. Liu, J. Rexford and D. Walker, “Optimizing the "One Big Switch" Abstraction in Software-Defined Networks”, *in Proc. ACM CoNEXT*, pp. 13-24, 2013.

[16] V. Heorhiadi, M. K. Reiter, V. Sekar, “Simplifying Software-Defined Network Optimization Using SOL”, *in Proc. USENIX NSDI*, pp. 223-236, 2016.

[17] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger and D. Walker, “Abstractions for Network Update”, *in Proc. ACM SIGCOMM*, pp. 323-334, 2012.

[18] W. Zhou, D. Jin, J. Croft, M. Caesar and P. B. Godfrey, “Enforcing Customizable Consistency Properties in Software-Defined Networks”, *in Proc. USENIX NSDI*, pp. 73-85, 2015.

완전 자동화 페이로드 시그니쳐 업데이트 시스템

Fully Automatic Payload Signature Update System

심규석, 구영훈, 이성호, 김명섭

고려대학교 컴퓨터정보학과

{kusuk007, gyh0808, gaek5}@korea.ac.kr,

요 약

오늘날 네트워크 자원을 사용하는 응용이 증대되면서 네트워크 관리를 위한 트래픽 분석에서 현재 연구 단계의 한계가 드러나고 그런 한계를 해결하기 위한 다양한 연구가 진행되고 있다. 그 중 대표적인 연구인 시그니쳐 자동 생성 연구는 응용 트래픽을 입력으로 트래픽의 공통된 패턴을 찾아 출력하는 과정이 자동화된 연구이다. 그러나 시그니쳐 자동 생성 연구는 트래픽을 사용자가 수집해야하는 반자동 시스템이기 때문에 트래픽 수집 단계에서 문제가 발생하고, 생성된 시그니쳐의 정확도를 신뢰할 수 없는 한계가 있다. 본 논문에서는 시그니쳐 자동 생성 시스템의 한계를 극복하기 위해 트래픽 수집, 시그니쳐 생성/검증/관리까지 모든 과정이 자동으로 이루어지는 시스템을 제안한다. 제안하는 방법을 학내 망의 실제 트래픽에 적용하여 추출한 시그니쳐는 분석률은 유지하며, 오탐률을 0 로 만드는 효과를 보였다.

Keyword: Automatic, Signature, Update, Identifier, Verifier, Generation

1. 서론

네트워크 관리 분야에서 트래픽 분석은 점점 더 중요시 되어가고 있다. 네트워크 환경은 증대되고 있고, 그 속에서 통신하는 트래픽은 매우 다양한 형태를 가진다. 다양한 종류의 트래픽을 정리하고, 조절하여 네트워크 자원을 최대한 효율적으로 사용하고, 네트워크 사용자에게 서비스를 원활하게 제공하는 것이 네트워크 관리의 목적이다[6].

네트워크 관리에 있어 가장 중요한 분야는 네트워크 모니터링이다. 네트워크 모니터링은 네트워크에서 특정 응용 및 서비스가 발생량을 알아내어, 그에 맞는 관리 정책을 수립하는 것을 의미한다. 네트워크 모니터링을 위한 트래픽 분석은 사용자에게는 질 높은 네트워크 서비스를 제공받도록 하고, 제공자에게는 최소한의 네트워크 자원으로 최대한에 질 높은 서비스를 제공할 수 있는 기반이 된다.

트래픽 분석에서 필수적으로 사용되는 것은 각 응용 별로 트래픽을 분류할 수 있는 시그니쳐 이다. 시그니쳐는 트래픽의 특징 별로 다양한 종류가 존재한다. 시그니쳐를 빠르고 정확하게 생성하기 위해 시그니쳐 자동 생성 연구는 활발하게 진행되고 있다. 시그니쳐 자동 생성 연구는 패킷의 페이로드 내용을 기반으로 공통 문자열을 자동으로 추출하여 시그니쳐화 하는 방법이다. 그러나 현재 단계의 시그니쳐 자동 생성 방법은 한계가 존재한다. 먼저, 시그니쳐를 만들기 위한 최소 조건으로 추출하고자 하는 응용의 트래픽을 수집 해야하는데 이 단계는 사용자가 직접 트래픽 수집 도구[4,5]를 이용하여 수집해야한다. 본 단계에서 트래픽을 잘 못 수집하면 추출된 시그니쳐는 잘못된 시그니쳐이다. 두번째는 수집된 트래픽이 단기간의 트래픽이기 때문에 시그니쳐 또한 응용의 대표적인 시그니쳐가 아닐 확률이 높다. 세번째는 시그니쳐 검증단계를 포함하지 않기 때문에 추출된 시그니쳐가 다른 응용을 분석할 확률이 높다. 마지막으로 항상 최신 시그니쳐를 유지할 수 없는 단점이 있다. 트래픽 패턴이 변화되더라도 인지하는 것은 여전히 사람이기 때문에 즉각적인 대응이 불가하다.

따라서 본 논문에서는 이러한 한계를 극복하기 위해 완전 자동화 시그니쳐 업데이트 시스템을 제안한다. 제안하는 시스템은 트래픽 수집, 시그니쳐 생성, 생성된 시그니쳐 검증 그리고 시그니쳐 관리까지 모든 과정이 자동으로 이루어 지는 시스템이다. 또한 지속적으로 시스템을 수행하기 때문에 항상 최신 시그니쳐를 유지할 수 있고, 트래픽 패턴 변화에 대해 즉각적인 대응이 가능하다.

본 논문은 1 장 서론에 이어, 2 장에서 시그니쳐 자동 생성 시스템에 대한 관련 연구에 대해 언급하고, 3 장에서 제안하는 시스템을 제안한다. 4 장에서는 제안한 시스템의 성능을 평가하고 마지막 5 장에서 결론 및 향후 연구에 대해 서술하고 논문을 마친다.

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2015R1D1A3A01018057)

2. 관련 연구

트래픽 분석을 위한 시그니쳐는 서론에서 언급했듯이 트래픽 특성에 따라 다양한 형태로 존재한다. 트래픽의 포트번호를 이용하여 분석하는 포트 기반 시그니쳐와 트래픽의 크기, 위치, 시간 등 통계적인 정보를 이용한 통계 기반 시그니쳐, 패킷의 데이터 부분인 페이로드 정보를 이용한 페이로드 기반 시그니쳐가 대표적이다.

포트 기반 시그니쳐는 IANA 에서 지정한 포트 정보를 시그니쳐로 사용한다. 본 방법은 적은 메모리 사용으로 매우 빠르게 트래픽 분석이 가능하지만, 현재 많은 응용들은 방화벽 및 IPS 장비를 통과하기 위해 임의 포트 번호를 사용하기 때문에 더 이상 포트 기반 시그니쳐는 무의미하다.

통계 기반 시그니쳐는 플로우 내의 패킷의 크기, 위치, 방향, 지속시간등을 시그니쳐로 사용한다. 본 방법은 암호화된 트래픽을 분석할 수 있고, 분석 속도가 빠른 장점이 있지만, 시그니쳐를 생성하는 것이 어렵고 응용이 한정되어 있을 뿐만 아니라 정확성을 기대하기 힘들다는 단점이 존재한다.

따라서 본 연구에서는 페이로드 기반 시그니쳐를 다룬다. 페이로드 기반 시그니쳐는 패킷의 데이터 부분인 페이로드 내의 공통된 문자열을 의미한다. 가장 정확도가 높은 시그니쳐이지만 시그니쳐 추출이 어렵고, 추출 과정에서 인적, 시간적 소비가 크다는 단점이 있다.

이러한 단점을 해결하기 위해 시그니쳐 자동 생성 방법이 연구되고 있다. 시그니쳐 자동 생성을 위해 다양한 알고리즘이 사용되고 있는데, 대표적으로 LCS (Longest Common String)알고리즘, Smith-Waterman 알고리즘과 가장 최근 연구에서 순차 패턴 알고리즘의 한 종류인 AprioriAll 알고리즘을 이용한 시그니쳐 자동 생성 연구가 있다.

LCS 알고리즘을 응용 트래픽 시그니쳐 추출 목적에 맞게 변형한 대표적인 방법은 LASER(LCS-based Application Signature ExtRaction)이다[1]. 본 방법은 두 개의 스트링을 비교하는 Matrix 에서 Backtracking 을 이용하여 연속된 공통 문자열을 찾는 방법이다. 따라서 두 개의 스트링을 계속 비교하여야 하기 때문에 추출 과정의 시간이 오래 걸리는 단점이 있다.

Smith-Waterman 은 본래 DNA 의 유사도를 판단하는 목적에 발표된 알고리즘이다[2]. 본 알고리즘을 사용한 응용 시그니쳐 자동 생성 방법도 발표되었는데 본 방법은 LCS 와 매우 유사하다. 하지만 Backtracking 방법에서 LCS 알고리즘은 연속된 공통 문자열을 찾을 수 있지만, Smith-Waterman 알고리즘은 연속된 공통 문자열의 집합을 찾을 수 있는 차이가 있다. 그러나 본 방법 또한 두 개의 스트링을 비교하는 것에 차이가 없기 때문에 추출 과정에 많은 시간이 소비된다.

가장 최근 연구인 AprioriAll 알고리즘을 이용한 시그니쳐 자동 생성 방법은 위의 단점을 해결할 수 있는 방법이다[3]. 위의 방법들은 특정 두 문자열을 비교하여 실제 트래픽에 적용하기 위해 트래픽의 순서를 정하거나, 그룹화시키는 전처리 과정과 생성된 부분 문자열을 하나의 규칙으로 통합시키는 후처리 과정이 필요하다면, 본 방법은 모든 문자열을 후보로 길이 1 부터 증가시키며 시그니쳐가 될 수 있는 가능성이 높은 문자열만 취하기 때문에 추출 과정에 많은 시간이 소비되지 않고, 전처리 과정과 후처리 과정이 필요 없는 장점이 있다.

3. 문제 정의

트래픽 분석이 다양한 이유로 매우 어려워지고 있다. 그 중 두가지의 가장 큰 이유가 있는데, 첫 번째로 지속적으로 변하는 트래픽 패턴이다. 트래픽은 일정한 패턴을 가지고 통신을 하게 된다. 하지만 서비스를 제공하는 업체에서 보안 위험에 예방하여 트래픽 패턴을 변화시킬 수 있다. 트래픽 패턴이 변화되면, 네트워크 관리자는 다시 트래픽 패턴을 분석하고, 그렇지 않다면 해당 서비스를 관리하지 못하게 된다. 두 번째는 새로운 응용이 급격하게 생성되고 많이 사용되고 있다. 모바일 앱 다운로드는 전 세계 시장에서 5 년 사이에 약 10 배정도 증가한 것으로 나타나고, 지속적으로 증가할 것이 예상된다. 네트워크 관리자는 이러한 새로운 응용에 대해 모두 다 파악하기 힘들고 파악한다 하더라도 모든 응용들을 분석하기는 한계가 있다.

이러한 한계를 극복하기 위해 시그니쳐 자동 생성의 연구는 활발히 이루어 지고 있다. 그러나 현재 단계의 시그니쳐 자동 생성은 트래픽 수집과 추출된 시그니쳐에 대한 검증, 그리고 시그니쳐 관리 방법에 대한 한계가 존재한다. 최적의 방법으로 시그니쳐를 자동 생성할 수 있지만 입력된 트래픽에서 문제가 발생하면 잘못된 시그니쳐가 추출될 수 밖에 없다. 또한 추출된 시그니쳐는 해당 응용만을 분석할 수 있다는 신뢰가 없고, 계속해서 추출되는 시그니쳐와 사용되지 않는 시그니쳐를 관리해야하는 한계는 존재한다. 따라서 본 논문에서는 이러한 과정을 완전 자동화 할 수 있는 완전 자동화 페이로드 시그니쳐 업데이트 시스템을 제안한다.

4. 완전 자동화 시그니쳐 업데이트 시스템

본 장에서는 완전 자동화 페이로드 시그니쳐 업데이트 시스템을 제안한다. 제안하는 시스템은 트래픽 수집, 시그니쳐 생성, 시그니쳐 검증, 그리고 시그니쳐 관리의 모든 과정이 자동으로 수행된다. 따라서 본 장에서 각 과정의 수행 과정에 대한 방법론을 설명한다. 그림 1 은 완전 자동화 페이로드 시그니쳐 업데이트 시스템의 수행 과정이다.

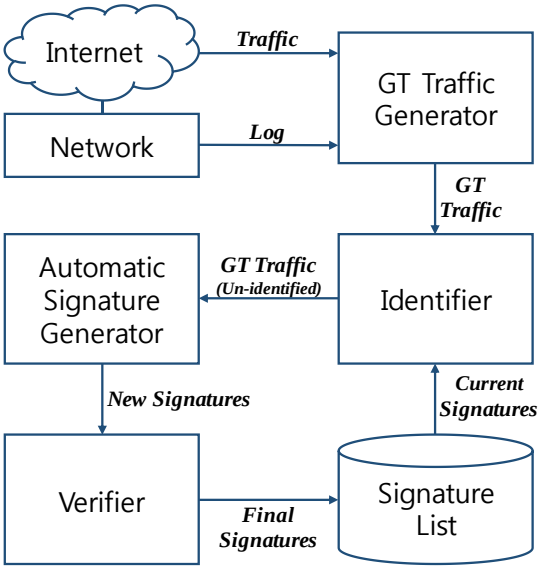


그림 1. 완전 자동화 시그니쳐 업데이트 시스템

그림 1 에서 GT Traffic Generator 에서는 트래픽 데이터와 호스트에서 발생하는 Log 데이터를 이용하여 응용 별 정답지 트래픽을 생성한다. Identifier 에서는 응용 별 정답지 트래픽과 기존 시그니쳐를 이용해 분석되지 않은 응용 별 정답지 트래픽을 출력한다. Automatic Signature Generator 에서는 분석되지 않은 응용 별 정답지 트래픽을 입력으로 새로운 시그니쳐를 출력한다. 새로운 시그니쳐는 Verifier 를 통해 검증된 시그니쳐를 분별하여 다시 Signature List 로 입력된다. 이러한 과정이 지속적으로 이어지면서 검증된 Final Signature 는 다음 수행과정 때 Current Signature 로 지속적인 검증 단계를 거쳐 항상 최신 시그니쳐를 유지할 수 있다.

4-1. GT Traffic Generator: 자동 트래픽 수집 및 정답지 트래픽 생성

기존 연구가 가진 가장 큰 한계는 트래픽을 사용자가 직접 수집해야하는 것이다. 이러한 과정에서 잘못된 트래픽이 수집될 수 있기 때문에 제안하는 시스템의 GT Traffic Generator 에서는 트래픽을 자동으로 수집하고, 각 응용 별로 정답지 트래픽을 생성한다. 정답지 트래픽을 생성하기 위해 본 시스템에서는 TMA(Traffic Measurement Agent)를 사용한다. TMA 는 각 호스트에서 실행되며 로그데이터를 남긴다. 표 1 은 TMA 에서 발생하는 로그데이터가 포함하는 정보이다.

표 1. TMA 정보

Process name
IP address (local, remote)
Port number (local, remote)
State (start, continue, end, server)
Protocol
Path

표 1 과 같은 정보들을 TMA 는 각 호스트에서 시간대별로 TMS(Traffic Measurement Server)로 전송한다. TMS 는 각 호스트에서 받아온 정보를 통합한다. 본 시스템에서는 TMS 로부터 통합된 정보와 트래픽의 정보를 매칭한다. 같은 시간, 같은 5-tuple(srcIP/Port, dstIP/Port, Protocol)를 가진 정보를 매칭하여 트래픽을 Process name 별로 저장하며 응용별 정답지 트래픽을 생성한다.

4-2. Identifier: 트래픽 분석 및 시그니쳐 관리

자동으로 수집된 정답지 트래픽을 이용하여 시그니쳐를 바로 생성할 수 있다. 하지만 같은 응용에 대해 지속적으로 동일한 시그니쳐가 추출될 수 있기 때문에 시스템에 불필요한 부하와 시간이 소비될 수 있다. 또한 기존 시그니쳐에서 트래픽 패턴 변화로 인해 사용되지 않는 시그니쳐에 대한 관리 방법이 필요하다. 본 시스템의 Identifier 에서는 기존 시그니쳐로 1 차 트래픽을 분석하여 분석되지 않는 트래픽을 분류하고, 사용되지 않는 시그니쳐를 삭제하며 시그니쳐를 관리한다.

최신 시그니쳐를 유지 하기 위해서는 새로운 시그니쳐를 생성할 뿐만 아니라 사용되지 않는 시그니쳐를 삭제해야한다. 이러한 과정이 생략된다면 시그니쳐는 지속적으로 축적되고, 축적된 시그니쳐는 트래픽 분석에 있어 시스템 부하 및 과도한 시간 소비의 원인이 된다. 따라서 본 시스템에서는 사용되지 않는 시그니쳐를 삭제한다. 수식(1)은 시그니쳐의 구성을 나타낸다.

$$\text{Signature} = \{\text{Header}, \text{Contents}, \text{Weight}, \text{Score}\} \quad (1)$$

다음과 같이 시그니쳐는 응용 서버의 정보인 IP address, Portnumber, 그리고 Protocol 로 이루어진 Header, 트래픽의 고유한 패턴인 Contents, 시그니쳐 삭제를 위한 가중치 값인 Weight, Weight 를 계산하기 위한 Score 값으로 구성된다. Score 는 알고리즘 1 과 같이 계산된다.

Procedure: *Calculation of Score*
Input: *Current Signatures, total GT traffic*
Output: *Cumulative Score by Signatures*

```
1: foreach signature S in OldSignatureSet do
2:   foreach flow F in GTtrafficSet do
3:     if ( identified(S,F) == 1 ) then
4:       S.CumulativeScore = 0; break;
5:     end
6:   end
7:   S.CumulativeScore ++;
8: end
```

알고리즘 1. Score 계산 알고리즘

다음과 같이 Score 는 분석에 사용되지 않은 횟수를 나타낸다. 그러나 Score 가 누적되더라도 분석에 사용된 시그니쳐는 다시 Score 값이 0 으로 초기화된다. 수식(2)는 Score 를 이용한 Weight 계산방법이다.

$$\mathrm{Weight_t}(S) = Weight_{t-p}(S) + C(S) - U_t(S) \quad (2)$$
$$(\mathrm{t = current\ time, p = period, C = completeness})$$

$$\mathrm{U_t}(S) = \left\lceil \frac{Weight_{(t-p)}(S)}{10} \right\rceil \times Score \quad (3)$$
$$\left(\mathrm{Weight_0} = 0, Weight \le 0, \left\lceil \frac{Weight(S)}{10} \right\rceil \ge 1\right)$$

시그니쳐에 Weight 를 구성한 것은 과거 분석률에 상당한 영향을 미친 시그니쳐이거나, 오랜 기간 분석에 사용되었던 시그니쳐가 삭제되기까지 시간을 주기 위함이다. 반면 기존 사용자가 잘못된 시그니쳐를 가지고 있었다면 Weight 값이 적어서 신속하게 삭제된다.

4-3. Automatic Signature Generator: 시그니쳐 자동생성

시그니쳐 자동 생성은 시그니쳐를 추출하기 위해 순차 패턴 알고리즘 중 Aprioriall 알고리즘을 수정하여 사용한다. 시그니쳐를 자동으로 생성하는 과정은 먼저 트래픽의 페이로드를 추출하여 시퀀스를 생성한다. 생성된 시퀀스들의 집합에서 길이 1 의 콘텐츠를 생성한다. 길이 1 콘텐츠는 최소 지지도 검사를 통해, 길이 2 로 생성된 후보자 콘텐츠와 삭제될 콘텐츠로 구분된다. 더 이상 길이가 증가되지 않을 때까지 이러한 과정을 반복하여 공통 문자열을 추출한다.

생성되는 시그니쳐는 그림 2 와 같이 총 3 가지 타입이 존재한다. 첫 번째 타입은 공통적으로 발생하는 연속된 문자열을 의미하는 콘텐츠 시그니쳐이다. 두 번째 타입은 동일한 패킷에서 발생하는 콘텐츠 시그니쳐의 조합을 의미하는 패킷 시그니쳐이다. 세 번째 타입은 동일한 플로우에서 발생하는 패킷 시그니쳐의 조합을 의미하는 플로우 시그니쳐이다.

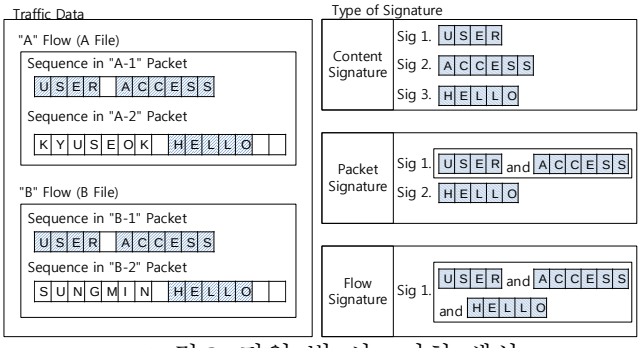


그림 2. 타입 별 시그니쳐 생성

4-4: Verifier: 시그니쳐 검증

제안하는 시스템은 새롭게 추출된 시그니쳐에 대해 검증 단계를 포함하고 있다. 검증단계에서는 "A" 응용 트래픽에 의해 추출된 시그니쳐가 "A" 응용이 아닌 "B","C"등 다른 응용 트래픽을 분석하는 것을 방지한다. 다른 응용 트래픽을 분석하는 시그니쳐는 시그니쳐로써의 의미를 잃어버리기 때문에 삭제해 주지만 아주 소량의 다른 응용 트래픽을 분석하고, 해당 응용 트래픽 분석에 큰 비중을 차지하고 있는 시그니쳐를 남기는 방법론을 제안한다.

시그니쳐를 검증은 다른 응용을 분석하는 시그니쳐 즉, False-Positive 가 있는 시그니쳐를 대상으로 한다. False-Positive 가 없는 시그니쳐는 최종 시그니쳐에 포함된다. 다음과 같이 검증과정에서 False-Positive 수치를 사용하게 된다. 표 2 는 데이터 판별 용어에 대해 설명한 것이다.

TP(True-Positive)는 A 응용 시그니쳐가 A 응용 트래픽을 분석한 수치이다. FN(False-Negative)는 A 응용 시그니쳐가 A 응용 트래픽을 분석하지 못한 수치이다. FP(False-Positive)는 A 응용 시그니쳐가 A 응용 트래픽을 제외한 나머지 트래픽을 분석한 수치이다. TN(True-Negative)는 A 응용 시그니쳐가 A 응용 트래픽을 제외한 나머지 트래픽을 분석하지 않은 수치이다.

표 2. 데이터 판별

	Relevant Traffic	Others Traffic
Analyze	TP(True-Positive)	FP(False-Positive)
Not Analyze	FN(False-Negative)	TN(True-Negative)

본 시스템에서 가장 중점적으로 FP(False-Positive) 수치를 다룬다. 특정 응용에 대한 시그니쳐 전체를 분석할 경우 TP(True-Positive)도 중요하지만, 현재 단계에서 각 시그니쳐 하나의 수치를 계산하기 때문에 FP 를 중점으로 다룬다. 각 시그니쳐 별 FP 를 계산하여 FP 가 0 인 시그니쳐는 최종 시그니쳐로 저장되고, FP 가 0 을 초과하는 시그니쳐에 한해 검증 단계에 입력된다. 시그니쳐의 검증을 위해 본 논문은 Precision, Recall, 그리고 F-measure 값을 사용한다. Precision 은 시그니쳐로 분석된 트래픽 중에 정확히 분석한 비율을 나타내고, Recall 은 특정 응용 트래픽 중 시그니쳐로 정확히 분석한 비율을 나타낸다. 다음과 같이 수식(4), 수식(5)이 표현한다.

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (4)$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad (5)$$

F-measure 는 Precision 과 Recall 을 이용하여 가중치를 주어 값을 측정하는 공식이다. 본 논문에서는 F-measure 를 사용하여 Precision 에 가중치를 두기 위해 β값을 사용하는데 1 을 기준으로 1 보다 작으면 Precision 에 민감한 수식이 되고, 1 보다 크면 Recall 값에 민감한 수식이 된다. 만약 동일한 가중치를 주어야한다면 β를 1 로 고정하면 된다. 본 논문에서는 F-measure 를 사용하여 Precision 에 가중치를 두기 위해 β값을 0.1 로 고정하여 사용한다. 수식(6)은 F-

measure 표현식이다.

$$\mathrm{F-measure} = \frac{(\beta^2+1)\times Precision\times Recall}{\beta^2\times Precision\times Recall} \quad (6)$$

F-measure 의 최대값은 1 이고 최소값은 0 이다. 1 이 나오는 경우는 해당 시그니쳐가 해당 정답지 트래픽에 있는 모든 트래픽을 분석하고, 그 외 트래픽을 하나도 분석하지 못했을 때이고, 0 이 나오는 경우는 TP=0 가 되면 된다. 따라서 본 논문에서는 F-measure 가 최소 0.95 이상의 정확도를 가진 시그니쳐만을 최종 시그니쳐로 저장한다.

제안하는 방법을 통해 관리되는 네트워크에서 호스트가 사용하는 응용에 대한 시그니쳐는 지속적으로 업데이트가 되며, 사용되지 않은 시그니쳐는 삭제되고, 새로운 시그니쳐는 추출된다. 새로운 시그니쳐는 검증과정을 거치면서 정확도 높은 시그니쳐를 유지한다.

5. 실험

본 장에서는 완전 자동화 페이로드 시그니쳐 업데이트 시스템에 의해 생성되는 시그니쳐와 기존 시그니쳐 자동 생성 시스템에 의해 생성되는 시그니쳐의 분석률과 오탐률을 비교실험한다. 분석률은 TP(True-Positive)로써 해당 응용 트래픽 중 분석한 트래픽의 양으로 표현하고, 오탐률은 FP(False-Positive)로써 해당 응용을 제외한 트래픽 중 분석한 트래픽의 양으로 표현한다. 표 3 은 호스트에서 자주 사용하는 5 가지 응용을 선정하여 비교 실험한 결과이다.

본 실험결과에서 모든 응용에 대해 FP 수치를 0 으로 감소시켰음에도 불구하고, TP 수치는 많이 감소하지 않았다. 특히, Dropbox 의 경우 FP 수치가 0 으로 감소하였지만 TP 수치는 유지하는 효과를 나타냈다. 따라서 본 시스템을 통해 시그니쳐의 정확도를 향상시키고 최신 시그니쳐로 유지할 수 있었다.

표 3. 시그니쳐 정확도 비교 실험 결과

	미 검증 시그니쳐			검증 시그니쳐		
	TP	FP	개수	TP	FP	개수
Naver	2,068 /2,128	904 /3,582	542	1,929 /2,128	0 /3,582	508
Youtube	642 /645	2,458 /5,065	112	417 /645	0 /5,065	100
uTorrent	2,138 /2,613	2,591 /3,097	631	2,108 /2,613	0 /3,097	608
Dropbox	31 /51	5 /5,659	5	31 /51	0 /5,659	4
Facebook	273 /273	1,344 /5,437	37	212 /273	0 /5,437	22

다음 실험결과는 각 응용 시그니쳐들의 분석률(TP) 및 오탐률(FP)가 지속적으로 업데이트 됨에 따라 변화량을 표현한다. 기존 연구와 비교하기 위해 각 응용의 초기 시그니쳐는 시그니쳐 자동 생성 시스템에 의해 추출된 시그니쳐를 사용한다. 실험결과에서와 같이 초기 시그니쳐로 각 응용을 분석했을 때, FP 의 수치가 높은 것을 확인할 수 있지만 본 시스템인 시그니쳐 업데이트 시스템에 의해 추출된 시그니쳐로 분석한 결과인 두번째부터 FP 의 수치가 급격히 줄어든 것을 확인할 수 있었다. 그림 3 은 Naver 응용의 시도 횟수에 따른 분석률 및 오탐률이다.

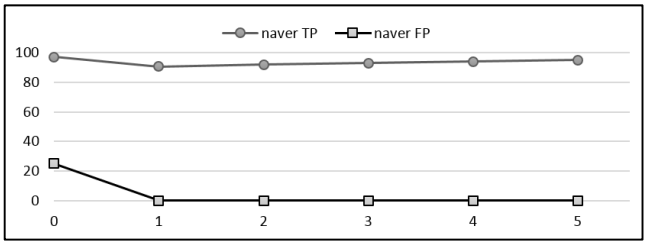


그림 3. 시그니쳐 업데이트 시도 횟수에 따른 naver 응용의 분석률 및 오탐률

6. 결론 및 향후 연구

본 논문에서 제안한 완전 자동화 페이로드 시그니쳐 업데이트 시스템은 기존 시그니쳐 생성 방법의 한계를 극복하기 위해 시그니쳐를 추출하기 위한 모든 과정인 트래픽 수집, 시그니쳐 생성, 시그니쳐 검증, 시그니쳐 관리까지 일련의 과정을 자동화한다. 제안된 시스템은 실험결과로 성능 향상이 증명되었다.

향후 본 시스템을 실시간에 실행하기 위한 성능 향상에 대한 연구가 필요하다. 또한 시그니쳐 자동 생성 과정에서 최적화된 순차 패턴 알고리즘을 선정해야한다.

참고문헌

[1] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, 2008, pp. 160-167.

[2] X. Feng, X. Huang, X. Tian, and Y. Ma, "Automatic traffic signature extraction based on Smith-waterman algorithm for traffic classification," *in Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, 2010, pp. 154-158..

[3]심규석, 윤성호, 이수강, 김성민, 정우석, 김명섭, "네트워크 트래픽 분석을 위한 snort content 규칙 자동 생성", 통신학회 논문지 Vol.40 No.04, pp.666-677, April. 2014.

[4]https://www.wireshark.org

[5]https://www.tcpdump.org

[6]Y. Wang, Y. Xiang, W. L. Zhou, and S. Z. Yu, “Generating regular expression signatures for network traffic classification in trusted network management,” *J. Netw. Comput. Appl.*, vol. 35, pp. 992-1000, May 2012.

Carrier-Grade Network Address Translation 설계 및 구현

이문상°, 이치영, 김우태, 이영우

kt 융합기술원 Infra 연구소

{moonsang.lee, chiyng.lee, utae.kim, young-woo.lee}@kt.com

요 약

최근 들어, 유무선망의 종단 사이에 위치하여 다양한 네트워크 기능을 제공하는 미들박스 서비스가 일반화되고 있다. 특히, 클라우드 컴퓨팅 분야의 가상화 기술이 네트워크 분야에 적용되면서 네트워크 가상화가 빠르게 진행되고, 가상 네트워크 장비들을 유연하게 연결하여 민첩한 네트워크 서비스를 제공하는 플랫폼들이 연구되고 있다. 본 논문에서는 캐리어급 미들박스 서비스를 제공하기 위한 필수 요소들을 살펴보고, 범용 서버에서 캐리어급 네트워크 주소변환 서비스를 제공하기 위한 프레임워크의 설계와 구현에 대해 기술한다.

1. 캐리어급 주소변환

미들박스는 통신에 참여하는 양 종단 사이에 위치하여 특정 기능을 수행하는 네트워크 장치를 통칭한다. 예를 들어, 기업의 사내외 망을 연결하여 보안을 제공하는 방화벽, 클라우드의 전단에 위치하여 백엔드 노드들의 부하를 분산시키는 로드밸런서, 연결 경로상의 중간에서 파일 및 컨텐츠의 접근 속도를 향상시키는 캐쉬/프락시, 사설망과 공용망 사이에 위치하여 사설 IP 주소를 공용 IP 주소로 변환하는 네트워크 주소 변환기 등을 포함한다. RFC 3234[1]에서는 22 종의 미들박스 서비스들을 정의하고 각 미들박스 종류의 특징을 기술하고 있다. 본 논문에서는 초고속 네트워크에서 캐리어급 주소변환(CGNAT: Carrier-Grade Network Address Translation)을 제공하기 위한 프레임워크에 대해 기술한다.

네트워크 주소변환(NAT: Network Address Translation)은 사설 IP 주소를 공용 IP 주소로 변환하는 기능이다. 네트워크 단말의 증가로 인해 부족해진 공용 IPv4 주소 공간을 효율적으로 확장하거나 네트워크 주소공간을 분리하여 보안성을 향상시키기 위한 목적으로 사용되며, 통상적으로 가입자 사이트의 공유기 또는 Wi-Fi AP(Access Point)에서 수행되어 왔다. 캐리어급 주소변환 서비스는 통신사에서 운용하는 장비에서 모든 가입자 패킷에 대한 주소변환을 수행하며, RFC 6888[2]에서는 캐리어급 주소변환을 위한 기본 요구사항들을 기술하고 있다. 기본적으로 가입자 사이트와 통신사 주소변환 장비 사이는 L2 계층으로 연결되어 있다고 가정하고, 통신사는 IEEE 802.1ad[3] 표준 태그를 이용하여 가입자를 구분하거나 가입자간 간섭을 제거할 수 있다.

일반적으로, 캐리어급 주소변환은 수많은 가입자의 모든 패킷에 대해 수행되어야 하기 때문에 높은 패킷 처리율이 요구된다. 전통적인 리눅스 서버의 네트워크 처리 성능으로는 이요구를 만족할 수 없기 때문에 리눅스 서버의 네트워크 성능을 극대화하는 것이 필요하다. 본 논문에서는 캐리어급 주소변환 서비스를 제공하기 위한 CGNAT 프레임워크를 설계하고, x86 기반의 범용 서버에서 실행 가능한 프로토타입을 구현하여 성능 측정한 결과를 기술한다. 10Gbps 링크에서 단방향 패킷에 대한 주소변환 처리율(Throughput)을 측정한 본 논문의 실험에 따르면, 리눅스 커널(iptables)을 사용하여 주소변환을 수행한 경우에 비해 본 논문의 CGNAT 프로토타입 처리율이 패킷 크기 전구간에 대해 2~15 배 향상되었다.

2. 데이터 평면 가속 기술

리눅스 커널의 TCP/IP 스택에 기반한 패킷 송수신은 인터럽트 처리, 문맥교환 및 스케쥴링, 빈번한 시스템 콜 및 데이터 복사로 인한 부하가 발생한다. 이를 개선하기 위해 UIO(Userspace IO), NAPI(New API), 폴링(Polling)을 사용한 고속 패킷 처리 기술들이 연구되어 왔으며, 오픈소스 프로젝트로 진행중인 DPDK(Data Plane Development Kit)[4]가 대표적인 예이다. DPDK 는 어플리케이션 프로세스가 하드웨어에 직접 접근할 수 있는 API 라이브러리와 프레임워크를 제공함으로써 커널을 통해 통신할 때 발생하는 부하를 줄이고, 폴링 방식을 사용하여 인터럽트 발생을 방지한다.

네트워크 가상화를 위한 오픈소스 플랫폼인 OPNFV[5]는 통신사의 캐리어급 네트워크 서비스를 제공하기 위한 통합 플랫폼으로서 그림 1 과 같은 플랫폼 스택(현재 버전: Brahmaputra)을 제공한다. ETSI NFV 표준명세에 따라 작성된 네트워크 단위 기능들은 OPNFV 상에서 가상화되고 상호 연결됨으로써 복잡한 네트워크 서비스를 유연하게 제공할 수 있다. 전통적인 TCP/IP 스택으로 다수 가입자로부터 발생하는 수많은 패킷들을 서비스 할 경우 심

각한 성능 저하에 직면할 수 있기 때문에 DPDK 또는 ODP(Open Data Plane)[5]와 같은 데이터 평면 가속화 계층을 통해 고속 처리 성능을 제공하고, 성능에 직접적인 영향이 없는 제어 평면의 경우 기존의 TCP/IP 스택을 사용하여 통신할 수 있다.

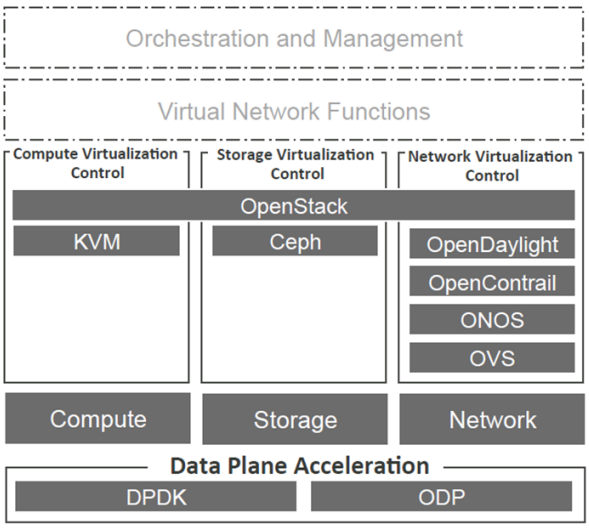


그림 1. OPNFV Brahmaputra 플랫폼

또 다른 데이터 평면 가속 기술로, 리눅스 커널의 TCP/IP 스택을 최적화하고 각 패킷의 처리 작업(라우팅, 주소변환, 필터링)을 별도의 프로세싱 유닛으로 오프로딩하는 방식이 있다. RouteBricks[6]는 소프트웨어 라우터의 성능을 향상시키기 위해 4 대의 서버를 클러스터로 구성하여 하나의 라우터로 동작하게 한다. 이는 다수의 서버에서 병렬적으로 여러 패킷의 경로 탐색을 동시에 수행함으로써 데이터 평면 전체의 성능을 향상시킨다. PacketShader[7]는 패킷 라우팅과 IPsec 처리를 GPGPU(General Purpose Graphic Processing Unit)에서 병렬적으로 수행한다. GPGPU 는 수백개의 프로세싱 코어를 가지고 있기 때문에 이들을 이용하여 많은 수의 패킷을 한꺼번에 처리할 수 있다. 하지만, 이 방식들은 네트워크의 부하가 높아짐에 따라 추가적인 프로세싱 유닛을 필요로 하기 때문에 한 서버에서의 패킷 송수신 능력을 최대화한 DPDK 에 비해 유연성이 떨어지는 단점이 있다.

3. CGNAT 프로토타입 구현

3.1. CGNAT 구성 및 동작

CGNAT 서버는 접속 단말별로 사설 IP 주소를 할당하고, 해당 단말들이 외부망을 접속할 때 사용할 공용 IP 주소를 외부망의 DHCD 서버에게 요청하여 할당 받아야 하며, 사설 IP 주소들과 공용 IP 주소간 매핑을 관리해야 한다. 특정 가입자 사이트의 단말들이 사용하는 다수의 사설 IP 주소들은 공용 IP 주소 하나를 공유하지만, 가입자 사이트별 공용 IP 주소들은 서로 구분되어야 한다.

그림 2 는 본 논문에서 구현한 CGNAT 서버의 주요 소프트웨어 구성을 나타내고, 그림 3 은 CGNAT 서버의 전체적인 제어 흐름을 상세하게 나타낸다. DHCP 동작을 위해서 ISC(Internet Systems Consortium)에서 오픈소스로 배포하는 DHCP 참조 구현(Reference Implementation)을 사용하였다. Private DHCPD 는 가입자 단말에게 사설 IP 주소를 할당하기 위한 DHCP 서버이고, Public DHCP Agent 는 가입자 단말이 외부망에서 인식될 공용 IP 주소를 할당받기 위한 DHCP 클라이언트이다. NAT Daemon 은 사설 IP 주소와 공용 IP 주소간 매핑을 관리하고, 공용 IP 주소의 요청(Request), 갱신(Renew), 및 해제(Release)를 제어한다. CGNAT 서버의 모든 패킷은 DPDK 를 통해 송수신되며, 패킷의 목적지가 CGNAT 서버일 경우에는 DPDK KNI(Kernel NIC Interface)를 통해 CGNAT 서버의 리눅스 커널에게 전달한다.

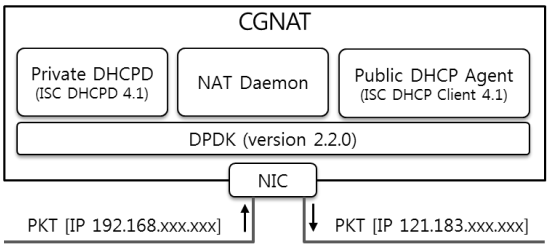


그림 2. CGNAT 서버의 소프트웨어 구성도

NAT Daemon 은 {사설 IP 주소, 단말 MAC 주소} → {공용 IP 주소, 가입자 사이트별 가상 MAC 주소}로 변환하기 위한 매핑과 세션 유지를 위한 5 튜플(src IP, dst IP, src Port, dst Port, Protocol Number)별 연결 관리를 제공한다. 가입자 사이트별 가상 MAC 주소는 SER(Service Edge Router)에서 가입자별 트래픽 관리에 사용되고, 5 튜플별 연결 관리는 의도하지 않은 패킷이 외부망으로부터 유입되는 것을 차단하기 위한 용도(Address and Port Dependent Filtering)로 사용된다.

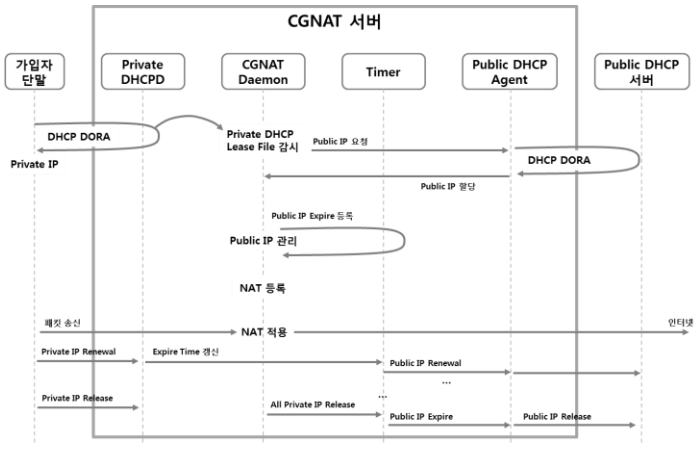


그림 3. CGNAT 서버의 제어 흐름

네트워크 주소변환을 수행하는데 있어서, 내부망에서 외부망으로 나가는 패킷의 경우 L2 헤더의 출발지 MAC 주소와 L3 헤더의 출발지 IP 주소가 변경되고, 외부망에서 내부망으로 들어오는 패킷의 경우 L2 헤더의 목적지 MAC 주소와 L3 헤더의 목적지 IP 주소가 변경된다. 또한, L4 계층의 포트 충돌이 발생할 경우 출발지 포트 번호가 변경될 수 있다. 또한, 주소변환 과정에서 L2 헤더와 L3 헤더의 변경이 발생하므로 각 헤더의 CRC 값을 갱신해야 하며, L4 헤더의 CRC 값이 출발지와 목적지 IP 주소에 의존성이 있으므로 UDP/TCP 같은 L4 프로토콜 종류에 따라 L4 헤더의 CRC 값도 갱신이 필요하다.

3.2. CGNAT 구현

본 논문에서는 Intel Xeon CPU 를 사용하는 리눅스 서버에서 CGNAT 서비스를 구현 하였고, 고속 패킷 처리를 위해 데이터 평면으로 DPDK 를 사용하였다. DPDK 에서 제시하는 두 가지 구현 모델인 순차적 처리 모델(Run-to-Completion Model)과 파이프라인 모델(Pipeline Framework)에 따라 각각 구현을 진행하여 구현 방법에 따른 성능 비교 대상으로 사용하였다.

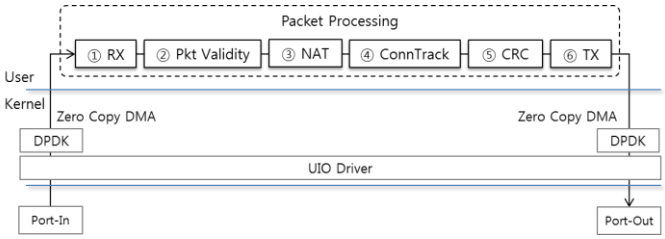


그림 4. 순차적 패킷 처리

CGNAT 서버는 네트워크 주소변환을 위해 그림 4 에 표시된 6 가지 동작을 순차적으로 수행한다. 주소 변환에 필요한 사설 IP 주소와 공용 IP 주소간 매핑과 실제 MAC 주소와 가상 MAC 주소간 매핑 관계는 3 장에서 기술한 DHCP 서버를 통해 사전에 NAT 테이블에 등록된다. CGNAT 서버로 패킷이 수신되면 패킷 종류에 따라 유효성 검사를 먼저 진행하여 유효한 패킷만 주소변환을 수행하고, CRC 값 오류가 있는 패킷들은 유효하지 않은 것으로 간주하여 드랍 처리된다. 주소가 변환된 패킷은 세션 관리를 위해 ConnTrack 테이블에 5 튜플을 등록하여 응답 패킷의 목적지 주소가 공용 IP 주소에서 사설 IP 주소로 변환될 수 있도록 준비하고, L2~L4 각 계층의 헤더에 CRC 값을 갱신한 후 아웃포트로 발신한다.

패킷의 수신과 발신은 DPDK 에서 제공하는 API 를 사용함으로써 리눅스 커널을 우회하여 네트워크 인터페이스 메모리와 CGNAT 프로세스의 메모리 사이에서 DMA 로 직접 이동되기 때문에 패킷 버퍼 복사가 발생하지 않는다. 또한, 패킷 송수신을 폴링 방식으로 수행하기 때문에 인터럽트가 발생하지 않고, UIO 를 통해 NIC 의 레지스터를 직접 접근하므로 패킷 송수신 명령어 전달이나 실행 상태 확인을 위한 시스템 콜이 필요하지 않다.

순차적 패킷 처리는 6 단계의 동작들이 완료되어야 다음 패킷의 처리를 시작하기 때문에 다수의 패킷들이 단기간내 수신될 경우 패킷들이 수신 버퍼에서 대기하는 시간이 증가하고 수신 버퍼에 가용한 공간이 없는 경우 패킷 드랍이 발생할 수 있다. 따라서, 6 단계를 분할하여 패킷 수신만 수행하는 단계, 수신 버퍼에 있는 패킷에 대해 주소변환만 수행하는 단계, 주소변환이 완료된 패킷을 발신만 하는 단계를 병렬적으로 수행되도록 하는 것이 성능 향상에 도움이 될 수 있다.

그림 5 는 6 단계 처리 과정을 2 단계씩 분할하여 파이프라인으로 처리하는 과정을 도식화한 것이다. 코어 A 는 ①, ② 동작만 반복적으로 수행하며 패킷을 수신하여 수신 버퍼를 채우고, 코어 B 는 수신 버퍼에 존재하는 패킷들에 대해 ③, ④ 동작만 반복적으로 수행하며 주소를 변환하며, 코어 C 는 ⑤, ⑥ 동작만 반복적으로 수행하며 주소변환이 완료된 패킷들을 발신한다. 파이프라인 패킷 처리는 순차적 패킷 처리에 비해 병렬성이 증가하여 특정 시점에 패킷 수신, 주소 변환, 패킷 발신이 동시에 처리될 수 있다.

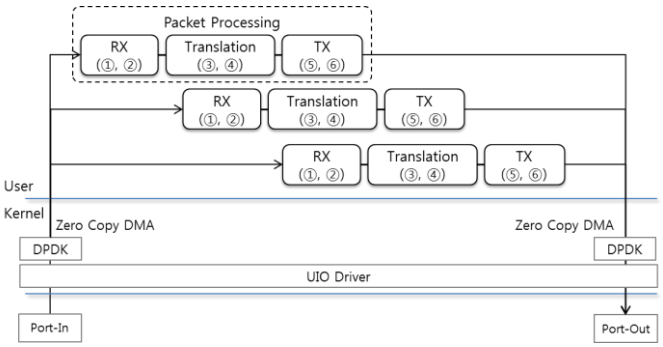


그림 5. 파이프라인 패킷 처리

파이프라인 패킷 처리는 각 파이프라인 단계 사이를 연결하는 패킷 큐를 거쳐야 하기 때문에 순차적 패킷 처리에 비해 메모리 접근이 증가할 수 있지만, 병렬적인 패킷 처리를 통해 패킷의 대기 시간을 단축할 수 있으므로 전체 지연시간이 감소하는 효과가 있다. 특히, 각 파이프라인 단계의 부하에 따라 ①~⑥을 적절히 분할함으로써 유연한 성능 최적화가 가능하다.

4. 성능 평가

4.1 실험 환경

순차적 패킷 처리 CGNAT 와 파이프라인 패킷 처리 CGNAT 의 성능 평가를 위해 표 1 과 같은 실

험 환경을 그림 6 과 같이 구성하였다. 패킷 생성기 N2X 의 P0 포트에서 발신된 패킷은 CGNAT 서버에서 주소변환이 완료된 후 P1 포트를 통해 발신되며, N2X 의 포트 감시 기능을 사용하여 N2X P1 포트에서 수신되는 패킷들을 수집하였다.

주요 구성	상세 명세
Processor	Intel Xeon E5520 2.27GHz (Quad Core)
NIC	Intel 82599EB 10G Ethernet Controller
PCI Bus	x8 PCI Express 2
Data Plane	DPDK v2.2.0
CGNAT OS	CentOS7 (Kernel 3.10.0)
Traffic Gen.	Agilent N2X

표 1. 성능 평가 환경

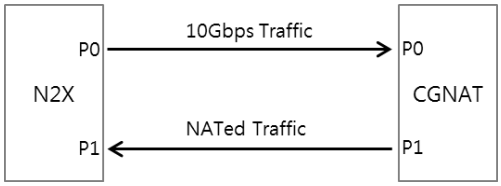


그림 6. 성능 평가 환경

4.2 주소변환 처리율

성능 평가를 위해 리눅스 커널(iptables)을 이용한 방법(Kernel), 순차적 패킷 처리를 구현한 방법(uNAT), 파이프라인 패킷 처리를 구현한 방법(uNATp), 단순히 L2 계층 포워딩을 구현한 DPDK 예제코드(L2FWD)를 대상으로 다양한 패킷 크기에 대한 주소변환 처리율을 측정하였다. L2 프레임의 프리엠블(Preamble)과 프레임간 간격(Inter-Frame Gap)을 고려하면, 64 바이트 패킷에 대한 이론상 최대 대역폭은 7.61Gbps 이지만, 프로세서, 메모리, PCI 버스에서 지연이 발생할 수 있으므로 실험에 사용된 하드웨어 사양에 따라 이론상 최대 대역폭보다 낮은 값이 측정될 수 있다. 그림 7 은 성능 측정한 결과를 도식화한 그래프이다. L2FWD 은 실험에서 사용한 하드웨어 사양으로 처리할 수 있는 최대 처리율의 상한선(Upper Bound)을 의미하고, Kernel 은 DPDK 기반 주소변환 처리율의 하한선(Lower Bound)을 의미한다.

본 논문에서 수행한 실험 결과에 따르면 패킷 크키가 증가할수록 주소변환 처리율은 증가한다. 동일한 10Gbps 대역폭 내에서 패킷 크기가 작을수록 패킷 개수가 많아지기 때문에 CGNAT 에서 처리해야 하는 주소변환의 부하가 증가하며, 패킷 크기가 증가할수록 주소변환의 부하가 감소하여 링크 대역폭에 근접한 성능을 나타낸다. 파이프라인 패킷 처리를 구현한 uNATp 의 경우, 512 바이트 이상에서는 10Gbps 링크의 전체 대역폭에 근접한 성능을 나타낸다. 64 바이트 패킷의 경우, uNATp 는 리눅스 커널을 이용한 주소변환에 비해 15.5 배 높은 처리율을 나타낸다.

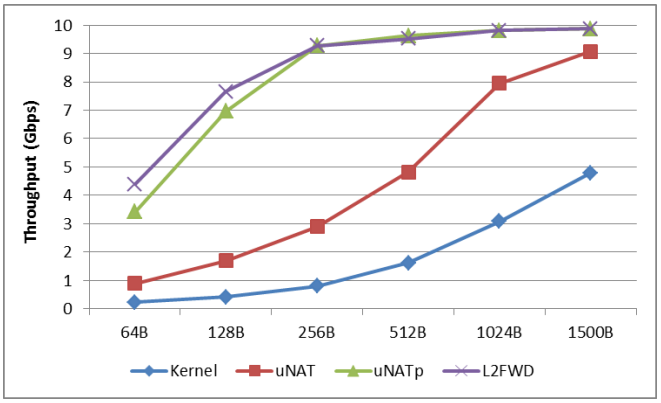


그림 7. 패킷 크기별 주소변환 처리율

5. 결론 및 향후 계획

본 논문에서는 캐리어급 주소변환을 위한 프레임워크를 설계하고 프로토타입 구현을 통해 성능 평가를 수행하였다. 전통적인 리눅스 커널(iptables)을 사용하는 경우에 비해 최대 15.5 배의 패킷 처리율 향상이 관찰되었으며, 향후에는 네트워크 가상화 기술을 적용하여 가상머신(Virtual Machin)상에서 실행되는 CGNAT 서버의 성능을 평가하고 하이퍼바이저 계층에서 제공할 수 있는 성능 개선 방법에 관한 연구를 진행할 계획이다.

3. 참고 문헌

[1] B. Carpenter, Middleboxes: Taxonomy and Issues, RFC 3234, 2002

[2] S. Perreault, Ed., Common Requirements for Carrier-Grade NATs (CGNs), RFC 6888, 2013

[3] Tony Jeffree, Provider Bridges (IEEE 802.1ad), 2005

[4] DPDK, http://dpdk.org

[5] OPNFV, https://www.opnfv.org

[5] ODP, http://www.opendataplane.org

[6] M. Dobrescu, N. Egi, K. Argyraki, B.G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, Routebricks: exploiting parallelism to scale software routers. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pages 15–28. ACM, 2009

[7] S. Han, K. Jang, K. Park, and S. Moon, Packetshader: a gpuaccelerated software router. ACM SIGCOMM Computer Communication Review, 40(4):195–206, 2010.

버퍼링 제어를 통한 효율적인 멀티미디어 스트리밍 중계 기법

권동우 º, 제희광, 김현우, 안동혁, 주홍택

계명대학교 컴퓨터공학과

{dwkwon, heegoang0426, hwkim84, donghyeokan, juht}@kmu.ac.kr

요 약

모바일 스마트 기기의 급격한 보급 및 확산으로 인해 모바일 스마트 기기 간의 멀티미디어 콘텐츠를 공유하는 기술이 확산되고 있다. 이를 위해 계층적으로 구성된 이종 무선 네트워크 상에서 멀티미디어 스트리밍 중계 시 중계 기기와 멤버 기기들 사이의 과도한 네트워크 자원 경쟁으로 인해 재생 화면 정지 문제가 발생한다. 본 논문에서는 이러한 문제를 해결하기 위해 중계 기기와 멤버 기기의 스트림 데이터 버퍼링을 제어함으로써 과도한 네트워크 점유를 방지하고 버퍼 고갈로 인한 재생 화면 정지 문제를 개선하여 스트리밍 품질을 향상시킨다.

1. 서론

최근 모바일 스마트 기기의 보급과 확산으로 개인용 컴퓨터 외의 다양한 모바일 기기에서 인터넷 서비스 제공이 가능하게 되고, 클라우드 서비스의 보급과 더불어 다수의 기기 간에 멀티미디어 콘텐츠를 공유하는 기술이 제공되고 있다[1]. 하지만 단일 무선 랜 환경에서 이 기술을 적용할 경우, 네트워크 대역폭 제약으로 인하여 규모 확장성이 낮아지는 문제가 발생한다. 이 문제는 기기들이 무선 네트워크 자원을 공유함에 따라 다수의 기기가 필요한 네트워크 대역폭을 충분히 확보하지 못하기 때문에 발생한다.

이를 해결하기 위해서 이종 무선 네트워크를 계층적으로 구성하여 멀티미디어 스트리밍 서비스를 제공하는 방법이 제안되었다[2,3]. 이 방법은 스트리밍을 위해 구성된 네트워크에 공통으로 연결된 중계 기기를 통하여 이종 네트워크 간 멀티미디어 데이터를 전달한다. 계층적 이종 무선 네트워크를 구성하여 이용하는 방법은 다중 무선 채널을 사용하여 단일 채널에 비해 대역폭을 향상되는 효과를 보였다.

하지만 이와 같은 방법으로 멀티미디어 스트리밍 중계를 진행할 경우, 중계 기기에 다수의 멤버 기기가 연결되면 멤버 기기 별 일시적 버퍼 고갈로 인한 재생 화면 정지 문제가 발생한다. 재생 화면 정지는 멤버 기기 간 과도한 경쟁으로 인해 순간적으로 해당 채널의 대역폭 사용의 공평성이 낮아지기 때문에 발생한다.

중계 기기와 멤버 기기 사이의 대역폭 공평성 문제의 해결 방법으로는 멀티미디어를 전달할 노드들에 가중치를 두어 그 가중치에 따라 서버에서 미디어를 전송하는 방식이 있다[4,5]. 하지만 이러한 방식을 사용할 경우 서버에서 미디어 전송을 관리하기 때문에 서버에 대한 의존성이 높아지고, 서비스의 참여 및 탈퇴가 상대적으로 빈번한 모바일 기기의 특성상 멤버 기기들에 대한 서버의 관리 비용이 증가한다.

이러한 문제를 해결하기 위해 멤버 기기에서 미디어의 요청을 제어하여 무선 네트워크에서 특정 기기의 과도한 데이터 전송 점유를 지양하고, 과도한 경쟁과 순간적인 네트워크 공평성 저하로 인한 버퍼 고갈을 방지하고자 한다. 본 논문에서는 멤버 기기 상에서 스트림 데이터 버퍼링 제어를 통한 효율적인 멀티미디어 스트리밍 중계 기법을 제안한다. 제안하는 기법의 평가를 위하여 재생 화면 정지 횟수와 화면 정지 시간을 측정하여 성능을 분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 멀티미디어 중계와 미디어 스트리밍 시 대역폭 공평성 제어와 관련된 연구를 소개한다. 3 장에서는 기존 중계 방법에서 발생하는 화면 정지 문제에 대해 알아보고, 버퍼링 제어를 이용한 스트리밍 중계 방법을 제안한다. 4 장에서는 제안한 중계 방법을 이용하여 성능 실험을 수행하고 그 결과를 분석하며, 마지막으로 5 장에서 결론과 향후 연구에 관해 논의한다.

2. 관련 연구

Je 등[6]은 무선 네트워크 환경과 Android 기기에 적합한 미디어 스트림 중계 엔진의 구조와 동작을 설계하였으며, Lin 등[7]은 메모리 복사 문제를 해결하기 위한 페이로드(payload) 공유, 다중 스레드와 Quality of Service (QoS) 스케줄링을 이용하는 One-to-Many Streaming Splicing (OMSS)을 제안하였다. OMSS를 적용한 결과, CPU 이용률 감소와 미디어 구독자의 최대 수가 증가함을 보였다. 하지만 제안한 방식은 운영체제 커널 상에서 동작을 제어해야 하기 때문에 시스템 호환성이 부족하다.

Hwang 등[8]은 Home-to-Home 콘텐츠 분배를 위해 Enhanced Adaptive Fast Replica (EAFR)를 기반으로

하는 Digital Living Network Alliance (DLNA) 프록시 구조를 제안하였다. 미디어 서버를 포함하는 홈 네트워크는 DLNA 프록시의 Content Distributor 를 통해 미디어를 전달하며, 미디어의 수신을 원하는 홈 네트워크는 DLNA 프록시의 Content Collector 를 통해 미디어를 전달받는다. 미디어를 전달받은 각 홈 네트워크들은 홈 네트워크 상의 노드들에게 전달받은 미디어를 멀티캐스트를 이용하여 전달한다. 하지만 멀티캐스트를 이용한 방식은 모바일 환경의 특성과 일부 모바일 기기의 무선 칩셋 및 드라이버의 지원 문제로 인한 제약으로 다양한 환경에 적용하기 어렵다.

무선 네트워크 상에서 멀티미디어 스트리밍 대역폭의 공평성을 제어하기 위해 수행된 연구로, Wei 와 Zhu[4]는 이종 네트워크를 통한 서로 다른 종류의 비디오 서비스를 수행할 때의 효율성을 보장하기 위해 콘벡스(convex) 최적화를 이용한 공정 대역폭 할당 전략을 제안하였다. Liu 등[5]은 내시 협상을 기반으로 하는 게임 이론 프레임워크를 통해 P2P 스트리밍 시스템의 네트워크 이용률과 공평성을 높이고, 대역폭 할당의 인센티브 문제를 해결하고자 하였고 공평성과 성능은 트레이드오프(trade-off) 관계였다. 이 연구들은 미디어 서버에서 각 노드의 가중치를 통해 미디어를 전달하는 노드를 선택 및 제어하기 때문에 미디어 서버에 대한 의존도가 높다.

재생 중지 현상을 개선하기 위한 연구로 서동은 등[9]은 멀티미디어 스트리밍 사용자의 체감 품질을 향상시키기 위해 QoE-enhanced Adaptation Algorithm over DASH (QAAD)를 제안하였다. 이는 가용 네트워크 대역폭을 지속적으로 측정하고, 사용자의 체감 품질을 감소시키는 요인들을 고려하여 다운로드 받을 비디오 데이터의 다음 세그먼트 비율을 결정한다. 이 때, 이전에 요청한 비디오 세그먼트의 품질과 가장 가까운 품질의 비디오 데이터를 요청하여 사용자의 체감 품질 저하를 방지하며 재생 끊김 방지를 위해 일정한 버퍼 용량을 항상 유지하도록 하였다. 이 방법은 가변적인 네트워크 상태를 가진 단일 기기에서 안정적인 비디오 재생이 가능함을 보였다.

본 논문에서 제안하는 계층적 이종 무선 네트워크의 미디어 스트림 중계 기법은 링크 계층 또는 운영체제의 커널 수정이 필요한 기존의 연구들과 달리 응용 계층에서 이루어지기 때문에 적용가능성이 높으며, 클라이언트 기기 측에서 대역폭 사용을 제어하기 때문에 미디어 서버 기기에 대한 의존성이 낮고 확장성이 높다.

3. 멀티미디어 스트리밍 중계 기법

3.1 미디어 스트림 데이터 버퍼 고갈 문제

이종 무선 네트워크 간의 멀티미디어 스트리밍 중계는 서로 다른 네트워크에 연결된 소스 기기와

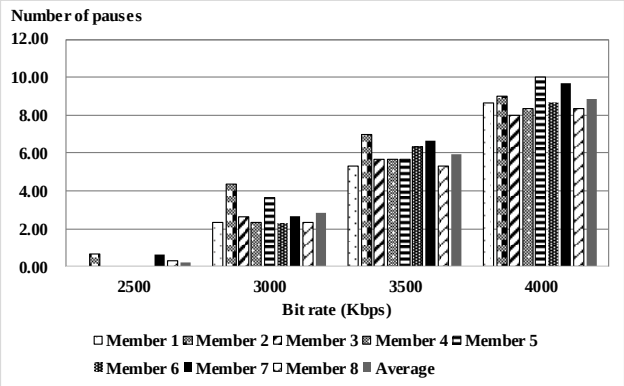


그림 1. 버퍼 고갈로 인한 재생화면 정지횟수

멤버 기기 간 미디어 스트림을 중계하기 위해 두 네트워크에 공통으로 연결된 중계 기기를 이용하여 소스 기기에서 멤버 기기에 미디어 스트림을 전달한다. 기존의 중계 방법은 원본 미디어를 보유하고 있는 소스 기기에 중계 기기가 미디어를 요청하여 다운로드하고, 이와 동시에 스트리밍을 진행하면서 멤버 기기들이 중계 기기로 미디어를 요청하여 스트리밍을 진행한다. 하지만 이 방식을 이용하여 스트리밍 중계를 진행하면 멤버 기기들이 동시에 중계 기기에 미디어를 요청하고 재생하기 때문에 대역폭 경쟁으로 인한 일부 멤버 기기에서 미디어 스트림을 수신하지 못하는 버퍼 고갈 문제가 발생한다. 결국 미디어 스트림 데이터를 받아 버퍼에 채우는 작업인 버퍼링 작업이 빈번하게 발생하게 되고, 이로 인해 미디어 재생이 원활히 진행되지 못하여 화면 정지 현상이 발생한다.

그림 1 은 이종 무선 네트워크상에서 미디어 스트리밍 중계를 진행하였을 때, 버퍼 고갈로 인한 멤버 기기별 화면 정지 횟수를 나타낸 그래프이다. 실험을 위해 하나의 중계 기기 노드에 총 8 개의 멤버 기기가 스트리밍 중계를 요청하였다. 그림 1 의 가로축은 동영상 화질을 결정하는 요소 중 하나인 비트전송률(bitrate) 값을 나타내고 있으며, 세로축은 화면 정지 횟수를 표현하고 있다. 실험 측정은 비트전송률 별로 총 세 번 측정하여 평균하였다. 이 그래프를 살펴보면 비트전송률이 2500Kbps 인 미디어 중계 시 화면 정지 횟수가 0~1 회로 거의 일어나지 않고 있으나 비트전송률이 높아짐에 따라 화면 정지 횟수가 증가하고 있다. 또한 같은 비트전송률을 가진 미디어를 재생하더라도 멤버 기기 별로 서로 다른 화면 정지 횟수가 발생하였음을 알 수 있다.

이와 같은 문제의 발생 원인은 기기 간 과도한 네트워크 자원 경쟁으로 인해 대역폭 공평성 저하가 발생하고, 그 결과 스트리밍에 필요한 대역폭이 멤버 기기들에 균등하게 할당이 되지 않아서이다. 따라서 본 논문에서는 이 문제를 해결하기 위해 멤버 기기에서 미디어 버퍼링을 제어하여 미디어 스트림을 중계하는 기법을 제안한다.

3.2 버퍼링 제어를 통한 스트리밍 중계 개념

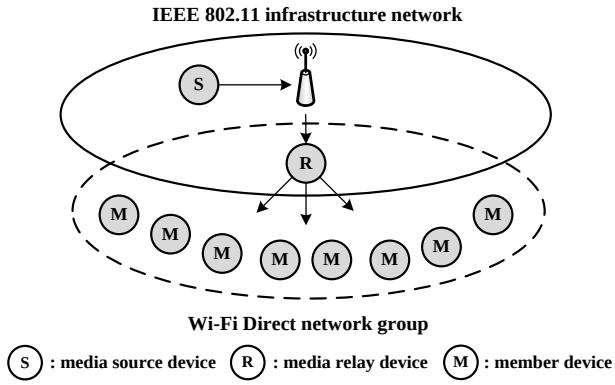


그림 2. 미디어 스트림 데이터 중계

이 절에서는 제안하는 스트리밍 중계 기법의 개념에 대해 설명한다. 그림 2 는 IEEE 802.11 infrastructure 네트워크와 Wi-Fi Direct 네트워크를 계층적으로 구성한 네트워크에서 멀티미디어 스트리밍 중계 과정의 흐름을 표현한 그림이다. 소스 기기와 멤버 기기는 각각 서로 다른 네트워크로 연결되어 있고 중계 기기는 소스 기기가 연결된 네트워크와 멤버 기기가 연결된 네트워크에 공통으로 연결되어 있다. 중계 과정을 살펴보면, (1) 먼저 소스 기기에 중계 기기가 미디어를 요청하면, (2) 소스 기기는 중계 기기에 미디어를 스트림 방식으로 전달하며 중계 기기는 이를 파일 형태로 저장한다. (3) 파일 형태로 저장된 미디어를 멤버 기기가 요청하면, (4) 중계 기기가 스트림 형식으로 미디어를 전달하여 멤버 기기가 재생할 수 있도록 한다.

기존 중계 방식은 미디어 재생을 위해 미디어를 요청하여 다운로드할 때, 해당 기기의 정해진 잔여 버퍼의 양만큼 요청하고 미디어를 다운로드하여 재생을 시작하도록 하며 점진적(progressive) 다운로드 방법을 사용한다. 이 과정에서 당장 필요하지 않은 과도한 양의 미디어 데이터를 요청하게 되고, 네트워크 사용량이 증가하여 네트워크 경쟁이 증가한다. 또한, 여러 멤버 기기에서 동시에 많은 데이터를 요청하게 되면 네트워크의 혼잡이 발생하여 멤버 기기들에 잦은 재생 화면 정지가 발생한다.

이러한 문제를 해결하기 위해 멤버 기기에서 미디어 요청과 버퍼링을 제어함으로써 네트워크에서의 과도한 경쟁 상황을 방지한다. 멤버 기기가 중계 기기로 미디어를 요청하면 우선 일정 부분을 다운로드하여 파일로 저장하고 미디어 재생을 진행한다. 그리고 버퍼 양이 일정 이하로 줄어들면 다시 미디어 파일의 다음 부분을 요청하여 다운로드한다. 미디어 재생은 미디어 파일을 다운로드한 시점부터 사용자가 정지시키지 않는 한 계속해서 재생되며, 버퍼에 저장된 스트림 데이터의 양이 충분하지 않으면 다시 미디어를 요청하여 재생이 중단되지 않도록 한다. 이러한 방법을 사용하여 당장 필요하지 않은 미디어 스트림 요청을 감소시켜 네트워크의 혼잡을 최소화한다. 이를 통해 기기 간의 과도한 대역폭 경쟁 및 공평성 저하를 방지하고 빈번한 재생

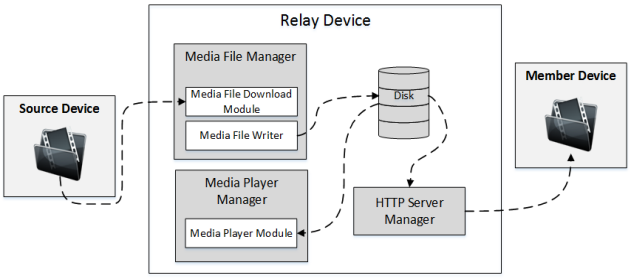


그림 3. 스트리밍 중계 엔진 구조: 중계 기기

화면 정지 문제를 해결할 수 있다.

3.3 스트리밍 중계 엔진 구조 및 중계 기법

그림 3 은 중계 기기의 스트리밍 중계 엔진의 구조를 나타낸 그림이다. 중계 기기의 스트리밍 중계 엔진은 Media File Manager (MFM), Media Player Manager (MPM), HTTP Server Manager (HSM)로 구성되어 있다. MFM 는 Media File Download Module (MFD Module)과 Media File Writer (MFW)로 구성된다. MFD Module 은 소스 기기에 미디어 파일을 요청하여 다운로드하도록 하며, MFW 는 소스 기기로부터 다운로드한 미디어 파일을 저장소에 저장하는 역할을 한다. MPM 은 Media Player Module (MP Module)로 구성되어 있으며, 이는 저장소에 저장된 미디어 파일을 읽어와 재생할 수 있도록 한다. HSM 은 멤버 기기로부터 미디어 파일 요청이 들어왔을 때 저장소에서 미디어 파일을 읽어와 멤버 기기로 전송하는 역할을 한다.

그림 4 는 멤버 기기의 스트리밍 중계 엔진의 구조를 나타낸 그림이다. 멤버 기기의 스트리밍 중계 엔진은 MFM 과 MPM 으로 구성되어 있다. MFM 은 중계 엔진과 동일하며, MPM 은 MP Module 과 함께 Buffer Management Module (BM Module)로 구성된다. MFD Module 은 중계 기기에 미디어 파일의 크기 및 전체 재생 시간에 따라 정해진 크기만큼의 미디어 스트림을 요청하여 다운로드하고, MFW 에 의해 다운로드한 스트림 데이터를 저장소에 저장하게 된다. MP Module 은 저장소에 저장된 미디어 파일을 재생하고, BM Module 은 미디어 재생기의 버퍼를 계속 감시하여 스트림 데이터의 양이 일정 수준 이하이면 MFD Module 에게 파일 다운로드를 요청하여 다음 미디어 스트림을 수신하도록 한다.

다음 식 (1)은 멤버 기기가 중계 기기에 미디어 스트림 데이터를 요청할 때 요청할 스트림 데이터의 크기를 구하는 식이다.

$$S_{download} = \frac{S_{total}}{D_{total}} \times D_{range} \times \alpha \quad (1)$$

식 (1)에서 S_{total} 은 원본 미디어의 전체 크기를 의미하고, D_{total} 은 원본 미디어의 전체 재생 시간을 의미한다. S_{total} 을 D_{total} 로 나누면 대략 1 초 동안 미

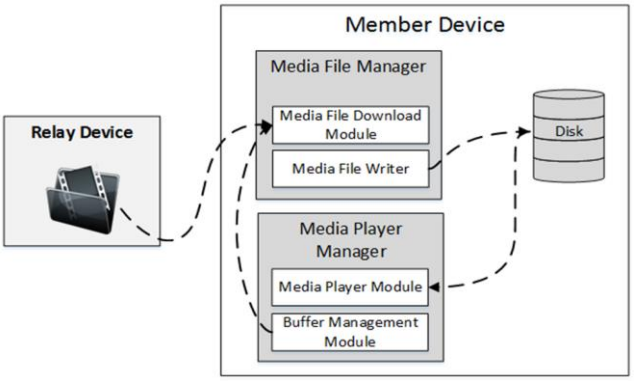


그림 4. 스트리밍 중계 엔진 구조: 멤버 기기

디어를 재생할 때 필요한 스트림 데이터의 크기를 추정할 수 있다. D_{range} 값은 멤버 기기가 요청하는 스트림 데이터의 최소 재생 시간을 의미한다. 예를 들어, 미디어 재생을 위해 최소 10 초 단위의 스트림 데이터 조각을 요청하여 다운로드하고자 하면 이 값을 10 으로 주어 대략 10 초 동안 재생할 수 있는 미디어 스트림 데이터를 요청할 수 있다.

최근 대부분의 멀티미디어는 가변 비트전송률(Variable Bitrate, VBR)을 가지므로 특정 시점의 비트전송률이 미디어 전체의 평균 비트전송률보다 높을 수 있으며 이 경우에 상대적으로 더 큰 크기의 $S_{download}$ 값이 요구된다. 이와 같이 가변 비트전송률에 따른 $S_{download}$ 값의 보정을 위해서 α 값을 사용한다. 만약 $S_{download}$ 값이 보정되지 않을 경우 다음 스트림 데이터를 요청하여 다운로드하는 시점이 오기 전에 스트림 데이터의 부족으로 인하여 미디어 재생이 중단되는 오류가 발생할 수 있다. 가변 비트전송률을 가진 멀티미디어에서 D_{range} 만큼의 재생 시간을 보장하기 위한 스트림 데이터 양인 $S_{download}$ 값은 실험을 통해 α 값을 2 로 설정하는 것이 적절함을 확인하였기 때문에 본 논문에서는 α 값을 2 로 설정하여 스트리밍 중계 실험을 수행하였다. 즉, 식 (1)의 $S_{download}$ 값은 멤버 기기가 중계 기기에게 미디어를 요청하였을 때, D_{range} 시간 동안 미디어를 재생하기 위하여 주기적으로 요청할 미디어 스트림 데이터의 용량을 나타낸다.

4. 실험방법 및 실험결과

4.1 실험 환경 및 실험방법

제안하는 멀티미디어 중계 기법의 성능을 평가하기 위해 실험에서 사용한 네트워크 구성은 그림 2 와 같다. 실험 네트워크는 소스 기기 1 대, 중계 기기 1 대, 멤버 기기 8 대로 구성된다. 원본 미디어를 가지고 있는 소스 기기와 멤버 기기로 미디어를 전달하는 중계 기기는 동일한 무선 접속점(Access Point, AP) 장치에 연결하여 IEEE 802.11n infrastructure 네트워크로 구성하였고, 원본 미디어를 전달받고자 하는 멤버 기기들과 중계 기기는 Wi-Fi Direct 네트워크 그룹으로 구성하였다. Wi-Fi Direct 네트워크의 그룹 소유자는 중계 기기로 설정하였다. 표 1 은 실험에 사용된 모바일 스마트 기기의 하드웨어 사양을 나타낸 것이다.

표 1. 실험에 사용된 모바일 스마트 기기 사양

Model	CPU	OS	RAM
Galaxy Note 10.1 2014 Edition (relay)	2.3GHz Quad-Core	Android 4.4.2 KitKat	3GB
Galaxy Tab S 8.4 (source/member)	1.9GHz/1.3GHz Octa-Core	Android 5.0.2 Lollipop	3GB

실험에 사용된 동영상은 3 분 47 초의 재생 시간을 가지는 MPEG-4 형식의 동영상이며, 동영상의 비트전송률은 각각 2500Kbps, 3000Kbps, 3500Kbps, 4000Kbps 를 가진다. 비트전송률이 높을수록 미디어의 크기가 증가하고 그 크기로 인해 네트워크 점유율이 증가한다. 이에 따라 멤버 기기에서 미디어 스트림 데이터에 대한 기아 현상이 발생하는 확률이 높아지므로 다양한 비트전송률을 가진 동영상을 실험에 사용하여 비트전송률에 따른 스트리밍 중계 성능을 비교한다.

스트림 데이터에 대한 버퍼링을 제어하지 않는 기존 중계 방법의 성능과 본 논문에서 제안한 스트리밍 중계 방법의 성능을 비교하기 위해 각 중계 방법 별로 멤버 기기의 평균 화면 정지 횟수, 평균 화면 정지 시간, 화면 정지 시간의 최소/최대 시간을 3 회씩 측정하여 평균 계산한다.

4.2 실험결과 및 분석

그림 5 는 버퍼링 제어를 하지 않는 기존 중계 방법과 본 논문에서 제안한 버퍼링 제어를 통한 중계 방법의 멤버 기기별 평균 화면 정지 횟수를 비트전송률별로 측정하여 나타낸 그래프이다. 이 그래프를 통해 제안하는 중계 방법이 기존의 중계 방법보다 더 적은 횟수의 재생 화면 정지가 발생하였음을 알 수 있다. 특히 가장 높은 비트전송률인 4000Kbps 의 미디어 재생 시, 기존 중계 방법은 8.83 회의 화면 정지가 발생한 반면에 제안한 중계 방법은 2.45 회로 매우 적은 횟수의 화면 정지가 발생하였다.

그림 6 은 기존 중계 방법과 본 논문에서 제안한 버퍼링 제어를 통한 중계 방법의 멤버 기기별 화면 정지 1 회 시의 최소/최대/평균 시간을 비트전송률별로 평균하여 그래프로 나타낸 것이다. 이 그래프를 통해 제안하는 중계 방법이 기존의 중계 방법에 비해 화면 정지가 발생하였을 때 버퍼링이 완료되어 다시 재생이 시작되기까지 더 적은 시간이 소요됨을 알 수 있다. 특히 4000Kbps 비트전송률을 가진 미디어 중계 시 제안된 중계 방법이 약 5.52 초 빠르게 재생이 회복되었다.

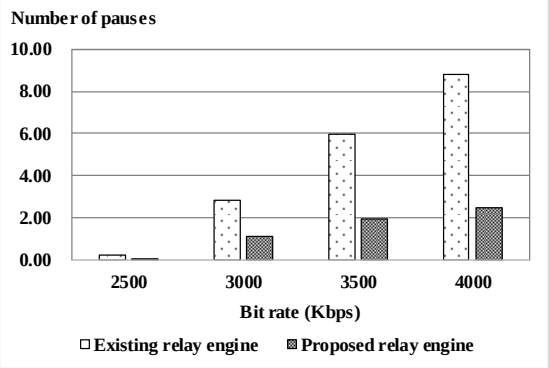


그림 5. 비트전송률별 평균 재생화면 정지횟수

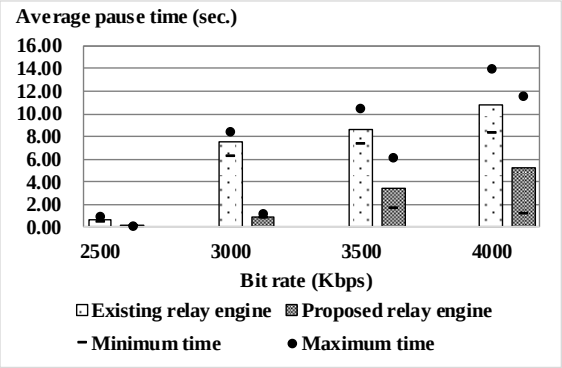


그림 6. 비트전송률별 재생화면 정지 평균 및 최소/최대 시간

실험 결과를 통해 기존 중계 방법보다 제안된 중계 방법의 재생 화면 정지 횟수와 화면 정지 시간이 현저하게 감소하였음을 알 수 있다. 이는 멤버 기기에서 미디어 스트림 데이터에 대한 버퍼링을 제어하여 멤버 기기 간 발생하는 과도한 대역폭 경쟁을 방지하고 네트워크 대역폭 사용에 대한 공평성 저하를 방지하기 때문이다. 이에 따라, 멤버 기기들의 재생 화면 정지 횟수 및 화면 정지 시간을 감소시켜 서비스 사용자가 체감하는 멀티미디어 스트리밍의 품질을 향상시킬 수 있다.

5. 결론 및 향후 연구

본 논문에서는 이종 무선 네트워크 상에서 다수의 기기를 이용한 멀티미디어 스트리밍 중계 시 발생하는 버퍼 고갈 문제를 해결하기 위해서 중계 기기에 대한 멤버 기기의 스트림 데이터 버퍼링을 제어하는 방법을 제안하였다. 그리고 제안한 중계 방법의 성능을 측정하기 위해 스트리밍 중계 시 발생하는 재생 화면 정지 횟수와 화면 정지 시간을 측정하는 실험을 수행하였다. 실험 결과, 기존 중계 방법보다 화면 정지 횟수 및 화면 정지 시간이 현저하게 감소하였고 이를 통해 제안된 중계 방법이 멀티미디어 스트리밍 서비스의 품질을 향상시킬 수 있음을 확인하였다.

향후 연구로 무선 네트워크의 상태에 따라 단일 채널의 멤버 기기 수를 제어하고, 새로운 네트워크 그룹의 참여 유도를 통해 스트리밍 중계가 가능하도록 하여 최대 중계 기기 수 제약 문제를 개선할 수 있도록 한다.

6. 참고 문헌

[1] C. Yoon, H. Lee, and W. Ryu, “Classification of N-Screen Services, Scenarios and its Standardization,” *ICACT Transactions on Advanced Communication Technology*, vol. 2, no. 3, pp. 214-222, May 2013.

[2] H. Je, D. Kwon, H. Kim, and H. Ju, "Mobile network configuration for large-scale multimedia delivery on a single WLAN," in *Proc. Asia-Pacific Network Operations and Management Symposium 2014*, pp. 1-6, Taiwan, Sep. 2014.

[3] D. Kwon, H. Je, H. Kim, and H. Ju, “An Autonomous Ad hoc Network Configuration Method for Scalable Media Streaming between Mobile Smart Devices,” *The Journal of the Korean Institute of Communication Sciences*, vol. 40, no. 3, pp. 516-528, Mar. 2015.

[4] S. Wei and Q. Zhu, “Efficient and Fair Bandwidth Allocation for Multiuser Multimedia Communication over Heterogeneous Networks,” in *Proc. of 6th International Congress on Image and Signal Processing*, vol. 1, pp. 16-20, China, Dec. 2013.

[5] L. Liu, J. Zou, and H. Xiong, “Bandwidth Allocation for Video Streaming over Peer-to-Peer Networks with Nash Bargaining,” in *Proc. of International Conference on Internet Multimedia Computing and Service*, pp. 363-367, China, Jul. 2014.

[6] H. Je, D. Kwon, and H. Ju, "Design of a Media Stream Relay Engine on Android OS," in *Proc. Asia-Pacific Network Operations and Management Symposium 2015*, pp. 1-4, Republic of Korea, Aug. 2015.

[7] Y. Lin, C. Ku, Y. Lai, and C. Hung, “In-Kernel Relay for Scalable One-to-Many Streaming,” *IEEE Multimedia*, vol. 20, no. 1, pp. 69-79, Feb. 2013.

[8] T. Hwang, H. Park, E. Paik, and J. Chung, “EAFR-Based DLAN Proxy for High-Quality Video Distribution in Extended Home Space,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, pp. 120-125, Feb. 2011.

[9] D. Suh, I. Jang, and S. Pack, “A Video Bitrate Adaptation Algorithm for DASH-Based Multimedia Streaming Services to Enhance User QoE,” *The Journal of the Korean Institute of Communication Sciences*, vol. 39, no. 6, pp. 341-349, Jun. 2014.

7. 감사의 글

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업이며(2015R1D1A1A01059786), 2016 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(R0126-16-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발).

페이로드 시그니쳐 품질 평가를 통한 고효율 응용 시그니쳐 탐색

Finding the Highly Efficient Application Signature through Payload Signature Quality Evaluation

이성호, 구영훈, Baraka D. Sija, 김명섭

고려대학교 컴퓨터정보학과

{gaek5, gyh0808, sijabarakajia25, tmskim}@korea.ac.kr

요 약

인터넷 속도의 증가와 다양한 응용의 개발로 인해 인터넷 사용자와 이들이 발생시키는 인터넷 트래픽의 양이 급격히 증가하고 있다. 트래픽 분석에 있어서 트래픽 응용 식별 방법은 페이로드 시그니쳐에 의존적이기 때문에 시그니쳐의 구성이나 개수에 따라 높은 부하와 처리 속도가 느린 단점을 갖는다. 따라서 본 논문에서는 응용 식별을 위한 페이로드 시그니쳐의 중요도를 평가하는 방법과 이를 바탕으로 높은 효율의 시그니쳐를 탐색하는 방법을 제안한다. 각 시그니쳐 별로 3 가지 기준을 바탕으로 가중치를 계산하고 계산된 가중치와 시그니쳐 맵을 통해 고효율의 시그니쳐 세트를 탐색한다. 제안하는 방법을 실제 트래픽에 적용했을 때 기존 대비 약 4 배의 응용 식별 능력을 가진 높은 효율의 시그니쳐들을 정의할 수 있었다.

Keyword : Application Traffic Identification, Application Signature, Signature Quality Evaluation, S-MAP

1. 서론

네트워크의 고속화와 더불어 다양한 서비스와 응용프로그램이 개발됨에 따라 개인 또는 기업은 인터넷으로 대표되는 네트워크에 대한 의존이 상당히 커져가고 있다. 이와 같은 현실 속에서 네트워크의 효율적 운용과 관리를 위한 응용 레벨의 트래픽의 모니터링과 분석은 네트워크 사용현황 파악과 확장 계획 수립 등의 다양한 분야에서 필요성이 증가하였다. 따라서 다양한 종류의 응용 레벨 트래픽을 정확하게 분류할 수 있는 방법과 고속 링크에서 발생하는 대용량의 트래픽을 실시간으로 처리하는 방법이 요구된다.

응용 레벨 트래픽 분류 방법에 있어 페이로드 시그니쳐 기반 분석 방법은 다른 분석 방법들에 비해 상대적으로 높은 분류 정확성과 식별률을 보였지만 낮은 처리 속도는 여전히 문제점으로 남아있었다.[1,2] 응용의 사용이 증가하고 있는 추세를 고려했을 때 페이로드 기반 분석 방법의 처리 속도 문제는 반드시 해결되어야 하는 과제이다.

본 논문에서는 트래픽 응용 식별을 위한 시그니쳐 생성 시스템에서 생성된 페이로드 시그니쳐의 중요도를 3 가지 기준으로 평가한다. 그리고 평가는 내용을 수치화해 응용 식별 측면에서 상대적으로 높은 효율을 갖는 시그니쳐를 탐색하는 방법을 제안한다.

제안하는 방법의 검증을 위하여 토렌트 응용 트래픽을 통해 실험을 진행하였다. 실험을 위해 응용 시그니쳐 자동 생성 시스템을 통해 시그니쳐들을 추출했다. 추출된 시그니쳐에 제안하는 시그니쳐 품질 평가 방법을 적용시켰고 이를 통해 탐색된 고효율의 시그니쳐들은 이전의 시그니쳐들 보다 평균적으로 4 배의 응용 식별 효율을 보였다. 품질 평가 방법은 시그니쳐의 활용성, 고유성 그리고 매칭 속도 3 가지 측면에서 이루어진다. 3 가지 기준으로 Quality Weight 값을 계산하고 시그니쳐 맵인 S-MAP 을 구성한다. 최종적으로 S-MAP 을 통해 본 논문에서 제안하는 고효율의 응용 시그니쳐를 탐색할 수 있다. 응용 식별률은 기존의 모든 시그니쳐들과 비교했을 때 약 3% 미만의 차이를 보였기 때문에 제안하는 시그니쳐 품질 평가 방법은 불필요한 시그니쳐는 제외하고 가치가 높은 시그니쳐만을 탐색할 수 있는 효율적인 방법이라고 판단할 수 있다.

본 논문의 구성은 다음과 같다. 본 장의 서론에 이어, 2 장에서는 관련 연구에 대해 기술하고, 3 장에서는 해결하고자 하는 문제에 대해 정의한다. 4 장에서는 제안하는 방법의 핵심이 되는 시그니쳐 품질 평가 방법과 탐색 방법에 대해 설명한다. 5 장에서는 제안하는 방법을 통해 정의된 고효율 시그니쳐를 트래픽 응용 식별 시스템에 적용해 실험해보고 그 결과를 통해 제안하는 방법의 타당성을 증명한다. 마지막으로 6 장에서는 결론 및 향후 연구에 대해 기술한다.

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2015R1D1A3A01018057)

2. 관련 연구

응용 프로그램 서비스 제공자는 방화벽을 우회하여 사용자에게 원활한 서비스를 제공하기 위해 복잡한 구조의 응용 레벨 프로토콜 구성하기 때문에 시그니쳐 또한 복잡하고 다양한 형태로 나타난다. 또한 인터넷에 기반한 응용의 증가로 인해 시그니쳐의 개수가 증가하고 그 가치 또한 높아지고 있다. 시그니쳐의 복잡도가 커지고, 개수가 증가하면서, 페이로드 시그니쳐 기반 응용 식별 시스템의 처리 속도는 전체적인 트래픽 분석 시스템의 성능을 결정하는 중요한 요소로 작용하게 되었다. 따라서 본 논문에서 제안하는 시그니쳐의 중요도 평가 방법과 이를 통한 고효율 시그니쳐 탐색 방법을 통해 앞에서 정의한 시그니쳐의 비효율을 개선하고 트래픽 응용 식별 시스템의 근본적인 성능 향상 효과를 기대할 수 있다. 이러한 방법과 관련해 기존에도 많은 연구들이 진행되어 왔다.

응용 식별 시스템의 속도 향상을 위한 다양한 패턴 매칭 알고리즘들이 제안되었다. 하지만 패턴 매칭 알고리즘의 성능은 입력 데이터의 구성에 의존적이며, 제한적인 성능 향상을 나타낸다[5]. 또한 매칭 단계에서 시그니쳐별로 갖는 오프셋이나 패턴에 대한 고려를 하지 않기 때문에 시그니쳐의 구성이나 구조에 종속적이라는 한계점이 있었다.

따라서 기존의 매칭 알고리즘 성능 개선의 한계적 문제점을 해결하기 위해 응용 레벨 트래픽의 발생 패턴을 분석 시스템에 반영하여 시그니쳐의 탐색 공간을 최소화하는 방법이 제안되었다[6]. 트래픽의 발생 패턴이나 특징을 반영해 응용 식별 효율을 높인다는 점에서 본 논문에서 제안하는 방법과 유사하다. 그러나 기존의 연구에서는 각 시그니쳐의 HC(Hit Count)만을 고려했고 중요도를 수치나 값을 통해 평가 하진 못하였다. 결과적으로 탐색 공간은 최소화 했지만 방법을 통해 효율적인 시그니쳐를 탐색하고 분류하지 못했다. 결과적으로 시그니쳐의 비효율성을 근본적으로 해결하지 못했고 방법 역시 제한적이었다.

시그니쳐 패턴 매칭 알고리즘이나 시그니쳐의 구성에 의존하지 않고 트래픽 간의 지역적인 연관성을 이용한 분석 방법도 제시되었다[3,4,7]. 하지만 CDN(Content Delivery Network)트래픽과 같은 상호 연관성은 존재하지만, 서로 다른 응용을 나타내는 트래픽을 식별하는 데에는 그 한계점이 존재 하고 또한 시그니쳐를 사용하지 않고 트래픽 간 연관성을 활용한 응용 식별 방법이기 때문에 그 정확도가 낮다. 따라서 전체적인 트래픽 분석 시스템의 구성에 있어서 한계점이 존재한다.

기존의 연구는 페이로드 시그니쳐 기반 트래픽 분류 시스템의 처리 속도 향상을 위해서 패턴 매칭 기법을 소프트웨어 또는 하드웨어적으로 개선하려는 노력과 트래픽의 특징을 정의하고 그룹화해서 시그니쳐 탐색 범위를 최소화 하는 방법이 주를 이루었다. 하지만 이러한 방법은 모두 앞에서 정의한 시그니쳐의 비효율성을 판단하고 수치화할 수 있는 방법이 없었다. 때문에 비효율적인 시그니쳐들은 계속 활용되어왔고 그 결과 응용 트래픽 식별 방법이나 전체 트래픽 분석 시스템의 성능 향상에 있어 많은 개선을 이루어낼 수 없었다.

3. 문제 정의

현재의 응용 트래픽 시그니쳐 생성 시스템에서 생성된 페이로드 시그니쳐는 단순히 분석을 목표로 하는 응용의 트래픽 파일에서 공통적으로 나오는 문자열을 찾아 나열하는 단계에 지나지 않았다. 따라서 생성된 시그니쳐들 중에는 응용 식별 측면에서 의미를 갖는 높은 수준의 시그니쳐도 있지만 단순히 의미 없는 공통된 문자들이 나열된 낮은 수준의 시그니쳐들도 존재하게 된다.

생성된 시그니쳐의 정확성과 그 의미를 평가할 명확한 지표와 기준이 없었다. 그 결과 불필요한 시그니쳐들 또한 분석 과정에 포함될 수 밖에 없었고 응용 식별률 측면에서는 일정 수준 이상을 만족했지만, 각 시그니쳐의 식별 효율과 그 중요성을 알 수 없었기 때문에 어떤 시그니쳐가 중요한 시그니쳐인지 판단할 수 없었다.

본 논문에서는 이러한 현상을 시그니쳐의 비효율로 정의하고 각각의 시그니쳐 별로 식별 효율과 시그니쳐로서의 중요도를 평가하는 방법과 이를 구체적으로 수치화해 고효율의 시그니쳐를 탐색하는 방법을 제안한다.

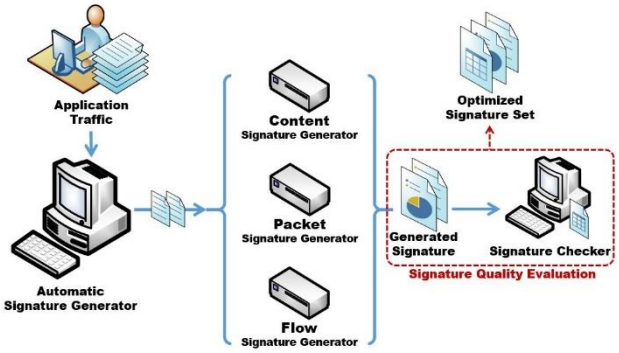


그림 1. 제안하는 방법의 개념도

그림 1 은 현재의 응용 트래픽 페이로드 시그니쳐 생성 시스템을 나타낸다. 시그니쳐 생성을 목표로 하는 응용에 대한 트래픽 파일들을 Automatic Signature Generator 에 넣게 되면 각 트래픽 파일들에서 공통적으로 나오는 문자열을 바탕으로 해당 응용에 대한 Flow, Packet, Content 시그니쳐가 생성되게 된다. 본 논문에서 제안하는 방법의 핵심은 그림 1 의 Signature Quality Evaluation 부분이다. Quality Evaluation 방법은 앞서 Signature Generator 에서 생성된 시그니쳐들을 받아와 각 시그니쳐들을 3 가지 기준을 바탕으로 평가하고 수치화해 가장 최적의 시그니쳐을 탐색한다. 이를 통해

높은 응용 식별 효율과 시그니쳐로서 특징을 갖는 Optimized Signature Set 을 탐색할 수 있다.

4. 제안하는 시그니쳐 품질 평가 방법

Quality Evaluation 은 Signature Generator 에서 생성된 시그니쳐의 응용 식별 효율, 중요도 그리고 의미를 평가하는 방법으로 그림 2 와 같이 크게 3 가지 기준으로 나뉜다.

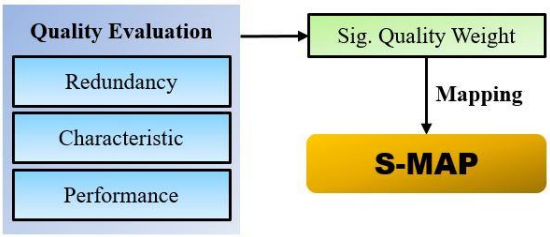


그림 2. 시그니쳐 중요도 평가 방법

첫번째 기준은 시그니쳐의 활용성(Redundancy)이다. 만약 어떤 응용 트래픽의 Flow, Packet 등이 다수의 시그니쳐에 의해 매칭 되었다면 해당 Flow, Packet 들을 분석하는 시그니쳐들은 같은 Flow 를 중복적으로 여러 번 매칭하게 된다. 따라서 시그니쳐로서의 가치가 상대적으로 낮다고 볼 수 있다. 이와 반대 개념으로 특정 시그니쳐가 여러 개의 Flow 나 Packet 에 매칭된 다면 해당 시그니쳐는 활용성과 효율이 높다고 판단할 수 있다. Quality Evaluation 에서는 이러한 시그니쳐의 활용 정도를 '시그니쳐 활용성' 이라는 기준으로 삼아 생성된 모든 시그니쳐들을 수식(1)과 같이 계산 한다.

$$\text{Redundancy Value} = |\{\text{Flows} \mid \text{Identified by } Sig_x\}| \quad (1)$$

두번째 기준은 시그니쳐의 고유성(Characteristic)으로 생성된 시그니쳐 전체 문장에서 의미를 갖는 글자와 숫자의 비율을 고려한다. 실제로 생성된 시그니쳐들 중에는 응용 트래픽 식별의 측면에서 시그니쳐로써 의미나 고유성을 갖지않는 Padding bits 나 Random String 또한 포함되어 있다. 따라서 보다 시그니쳐로써의 특징과 고유성을 갖는 시그니쳐를 찾기 위한 방법이 필요하다. Quality Evaluation 에서는 시그니쳐 문자열에서 표현가능한 Character 와 Numeric 의 비율 및 응용 이름의 유무 등을 평가한다. 그리고 이러한 기준을 '시그니쳐의 고유성' 이라 정의하고 수식 2 와 같이 계산한다.

$$\text{Characteristic Value} = \frac{|character| + |numberic|}{Sig_x TotalLength} \quad (2)$$

(if application name is in the Sig, CV is fixed to 1)

세번째 기준은 시그니쳐의 매칭 속도(Performance)이다. 시그니쳐의 매칭 속도를 판단하기 위해서는 매칭 오프셋을 고려해야 한다. 시그니쳐의 매칭 오프셋이란 시그니쳐가 나타나는 응용 트래픽 패킷의 페이로드 위치를 의미한다. 시그니쳐의 매칭 속도를 기준으로 설정한 이유는 응용 트래픽 식별이나 전체 트래픽 분석 시스템의 성능을 고려할 때, 시그니쳐가 존재하는 오프셋이 앞쪽에 고정되어 있다면 보다 빠른 응용 식별이 가능하고 이는 트래픽 분석 시스템의 성능 향상과 밀접한 연관이 있다. 따라서 시그니쳐의 매칭 속도가 빠를수록 좋은 시그니쳐이다. 따라서 '시그니쳐의 매칭 속도' 를 세번째 기준으로 정의하고 수식 3 과 같이 계산한다.

$$\text{Performance Value} = \frac{L - Sig_x MatchedOffset}{L} \quad (3)$$

(L=Max Payload Length)

결과적으로 본 논문에서 제안하는 Quality Evaluation 은 정의한 시그니쳐의 활용성(Redundancy), 시그니쳐의 고유성(Characteristic), 시그니쳐 매칭 속도(Performance)를 바탕으로 수행한다.

$$\text{Quality Weight} = PV \times CV \times \log(RV) \quad (4)$$

(PV=Performance Value, CV=Characteristic Value, RV=Redundacy Value)

각 시그니쳐 별로 3 가지 정의에 대한 내용들을 수치화 하고 계산해 최종적으로 Quality Evaluation 을 위한 최종 값인 Quality Weight 을 계산한다. Quality Weight 의 계산 방법은 수식 4 과 같다. 수식 4 와 같이 구성한 이유는 각 시그니쳐 간의 Quality Weight 값의 편차를 낮추고 정규화 하기 위해서이다.

Sig ID	QV Weight	Identified Flow Sequence
1	0.9	3,5,7,10,11,12
2	2.5	2,5,9,11
3	2.9	3,5
4	3.0	10,12
5	2.0	1,3,5,7,9,10
6	3.4	11
7	3.8	6
8	1.5	1,2,5,8,9,12
9	4.0	5
10	1.7	2,4,5,6,9
11	4.5	4
12	4.7	1

표 1. S-MAP Table(Example)

그림 2 와 같이 Quality Weight 값은 이후 고효율 시그니쳐 탐색 방법인 S-MAP 을 구성하기 위해 사용된다. 그림 2 에서 정의한 S-MAP 은 고효율의 시그니쳐 세트를 탐색하기 위한 방법으로 앞서 정의한 Quliaty Weight 값을 바탕으로 구성된다.

표 1 과 그림 3 은 S-MAP 을 구성한 예이다. 각

시그니쳐가 분석하는 Flow 들과 Quality Weight 를 사용해 표 1 과 같은 S-MAP Table 을 구성하고 2 차원 맵을 구성해 각 Flow 를 분석하는 시그니쳐를 매핑한다. 그리고 각 포인트마다 Quality Weight 를 참조해 모든 X 축 Flow 포인트를 연결하는 선을 그린다.

그림 3 과 같이 X 축에 매핑 된 모든 Flow 들을 연결하는 선은 각 포인트의 최대 Quality Weight 값들을 연결을 통해 구성된다.

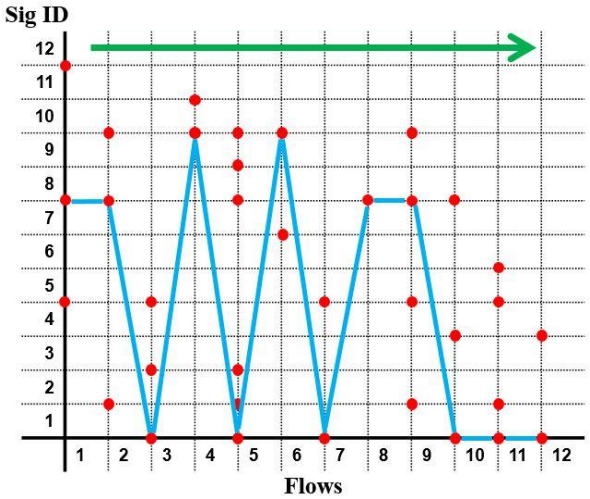


그림 3. S-MAP(Example)

그림 3 의 S-MAP 예를 참고했을 때, 각 포인트의 Quality Weight 값을 참조한 선이 그려지고 이 때 사용된 시그니쳐 세트(Y 축)은 {1, 8, 10}번 시그니쳐이다. 따라서 고효율의 시그니쳐는 1, 8 10 번 시그니쳐이고 이때 해당 시그니쳐의 효율은 식 5 의 Signature Average Efficiency 를 통해 구한다.

SAE 의 의미는 단위 시그니쳐가 분석한 Flow 의 개수이다. 이전보다 시그니쳐의 개수가 1/4 로 압축

되었으므로 SAE 는 4 배 커지게 된다. 따라서 S-MAP 을 통해 탐색된 시그니쳐 세트는 이전의 시그니쳐들 보다 평균 4 배 높은 시그니쳐 효율을 갖는다고 판단할 수 있다.

$$SAE = \frac{\sum_{x=1}^{|SigSet|} |\{Flows \mid IdentifiedbySig_x\}|}{|SigSet|} \quad (5)$$

(SAE=Signature Avergae Efficiency)

5. 실험 및 결과

본 장에서는 3 장에서 제안한 시그니쳐 Quality Evaluation 방법을 실제 응용 트래픽에 적용시켜보고 그 결과를 분석한다.

성능 평가를 위해 사용한 트래픽은 표 2 와 같다. 6 개의 트레이스 모두 토렌트 응용의 트래픽으로 각각 동영상 파일을 다운로드하며 수집했다.

Trace ID	Size(MB)	Flow	Pkt
1	113	2500	114,705
2	110	405	104,785
3	46	1452	48,867
4	34	1503	52,069
5	55	501	52,302
6	54	646	52,048

표 2. Traffic Trace

표 1 의 트레이스들을 바탕으로 그림 1 의 시스템을 적용해 Quality Evaluation 평가를 진행했고, 그 결과 그림 4,5 와 같은 결과를 얻을 수 있었다. 평가를 위해 적용한 시그니쳐들은 그림 1 의 Content Siganture 로 가장 낮은 단위의 시그니쳐이다. Content Signature 를 사용한 이유는 가장 낮은 단위의 시그니쳐에서 Quality Evaluation 방법의 효율성이 검증되면 이후 Packet 이나 Flow 시그니쳐에도 쉽게 적용 가능하기 때문이다.

그림 4 는 3 장에서 정의한 방법을 통해 구현된 S-MAP 이다. 6 개의 트레이스에서 총 139 개의 응용 Content Signature 와 약 14,000 개의 패킷 시퀀스가 24 개의 시그니쳐로 압축되었다.

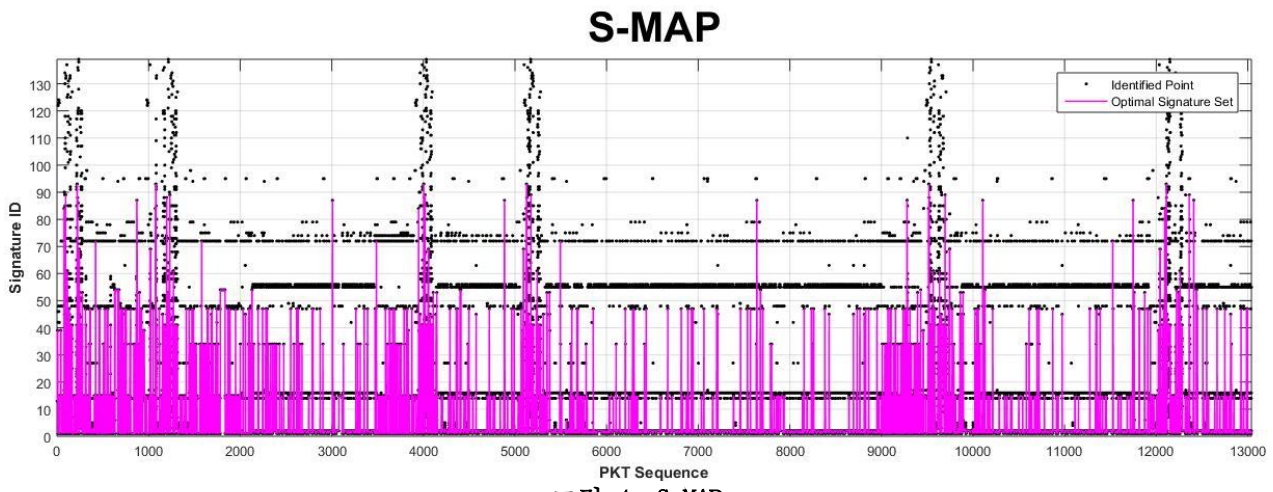


그림 4. S-MAP

그 결과 앞서 정의한 Signature Average Efficiency 의 측면에서 Quality Evaluation 을 적용하지 않은 시그니쳐 보다 평균 5 배 정도 높은 효율을 갖는 시그니쳐들을 탐색할 수 있었다. 그림 4 의 S-MAP 에서 각 점은 시그니쳐로 식별된 패킷 시퀀스를 의미한다. S-MAP 을 통해 확인하였을 때에 Y 축의 139 개의 시그니쳐 중 실제 연결된 점들의 수준이 24 개로 일정한 것을 확인 할 수 있다.

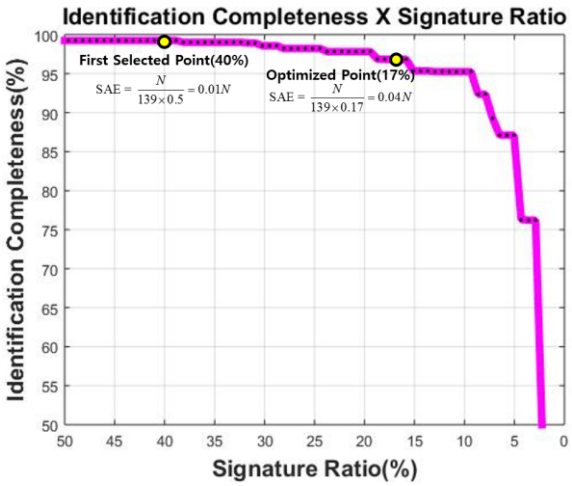


그림 5. 응용 식별률에 따른 시그니쳐 비율

그림 5 는 시그니쳐 비율에 따른 식별률을 나타낸다. 생성된 139 개의 시그니쳐를 100%로 정의했을 때 각 시그니쳐의 Quality Weight 가 낮은 시그니쳐부터 순차적으로 제외시키면서 시그니쳐 세트를 압축한다.

그림 5 를 참고했을 때 시그니쳐 세트의 압축률이 약 30%가 된 시점부터 식별률이 점진적으로 감소하는 것을 확인할 수 있다. 139 개의 시그니쳐를 같은 식별률을 유지하며 전체 시그니쳐의 35%인 40 개의 시그니쳐로 압축할 수 있었다. 하지만 본 논문에서 제안하는 Quality Evaluation 방법을 사용했을 때 압축된 24 개의 시그니쳐는 전체 시그니쳐의 약 17%로 1 차적으로 추려진 40 개의 시그니쳐 보다 18% 낮은 시그니쳐 비율을 갖는 동시에 응용 식별률 측면에서는 3% 미만의 차이를 보이고 있다.

시그니쳐 비율이 17% 보다 낮아지면서 본 논문에서 제안한 Quality Evaluation 을 통해 정의된 높은 Quality Weight 값을 갖는 시그니쳐들이 시그니쳐 세트에서 제외되고있다. 그리고 그 결과 시그니쳐 비율이 약 10%가 되는 시점부터 응용 식별률이 급격하게 낮아지는 것을 확인할 수 있다. 앞서 정의한 Signature Average Efficiency 를 바탕으로 계산했을 때 Identified Flow 의 개수를 N 으로 가정하고 그림 5 의 First Selected Point 에서는 0.01N 의 SAE 값을 갖는다. 반면 Optimized Point 에서는 0.04N 의 SAE 값을 갖는다. 따라서 본 논문에서 제안하는 QE 를 통해 이전에 비해 약 4~5 배의 응용 식별 효율을 갖는 시그니쳐를 탐색할 수 있었다.

6. 결론 및 향후 연구

결과적으로 본 논문에서 제안한 Quality Evaluation 방법을 통해 탐색된 시그니쳐 세트는 기존의 시그니쳐들 보다 높은 효율성을 갖는 동시에 응용 식별률 측면에서 큰 차이 없는 것을 확인할 수 있다. 또한 그림 5 의 그래프를 참고하였을 때, 정의된 높은 Quality Weight 의 시그니쳐들은 매우 고효율의 시그니쳐인 것을 알 수 있다.

본 논문에서 제안한 응용 트래픽 식별을 위한 페이로드 시그니쳐의 Quality Evaluation 방법은 결과적으로 그 의미와 효율성 측면에서 매우 긍정적인 것을 확인할 수 있다. 따라서 제안하는 방법을 이후 연구할 실시간 응용 시그니쳐 자동 생성 시스템에 적용해 실시간으로 생성된 시그니쳐의 중요도와 가치를 평가해볼 계획이다.

참고 문헌

[1] J. S. Park, J. W. Park, S. H. Yoon, Y. S. Oh, M. S. Kim, "Development of signature Generation system and Verification Network for Application Level Traffic Classification", in Proc. KIPS conf. Apr. 23-24, 2009, pp. 1288-1291, PuSan, Korea.

[2] S. H. Yoon, H. G. Roh, M. S. Kim, "Internet Application Traffic Classification using Traffic Measurement Agent ", in Proc. KICS Jul. 2-4, 2008, pp.618. Jeju Island, Korea.

[3] Fnag Yu, Zhifeng Chen, Yanlei Dino, T. V. Lakshman, Randy H. Katz, “Fast and memory Efficient Regular Expression Matching for Deep Packet Inspection” ANCS 2006, December , 2006, San jose, California USA.

[4] Christopher L. Hayes , Yan Luo, “DPICO: a high speed deep packet inspection engine using compact finite automata”, ACM/IEEE Symposium on Architecture for networking and communications systems, December 03-04, 2007, Orlando, Florida, USA

[5] C. L. Hayes and Y. Luo, “DPICO: A high speed deep packet inspection engine using compact finite automata,” in *Proc. ACM/IEEE Symp. Architecture Netw. Commun Syst. (ANCS ‘07)*, pp. 195-203, Orlando, USA, Dec. 2007.

[7] J. S. Park and M. S. Kim, “Performance improvement of application-level traffic classification system using application traffic pattern,” in *Proc. KICS Summer Conf.*, pp. 3-7, Jeju, Korea, Jun. 2011.

[8] J.-S. Park, S.-H. Yoon, and M.-S. Kim, “Performance improvement of the payload signature based traffic classification system using application traffic locality,” *J. KICS*, vol. 38B, no. 7, pp. 519-525, Jul. 2013.

Unitary Matrix 를 이용한 저복잡 네트워크 코딩 및 디코딩 방법

권정민 O, 박형곤

이화여자대학교 전자공학과

jungmin.kwon@ewhain.net, hyunggon.park@ewha.ac.kr

요 약

본 논문에서는 배터리 기반의 단순 단말 기기로 구성되어 있는 사물인터넷 환경에서 네트워크 코딩을 이용한 데이터 전송 시 요구되는 연산 복잡도를 최소화 하기 위한 연구이다. 제안된 시스템에서는 Unitary Matrix 를 코딩 계수로 사용하여 디코딩 복잡도를 최소화하였고 이를 모의 실험을 통하여 제안한 방법이 기존의 랜덤 선형 네트워크 코딩과 비교하여 낮은 복잡도를 요구함을 확인하였다.

1. 서론

최근 스마트 폰과 태블릿 PC 의 대중화로 인하여 데이터 트래픽이 빠르게 증가하고 있다. 특히, 사물 인터넷 (Internet of Things) 기술의 도입은 수 많은 단말 기기가 생성하는 데이터를 실시간으로 전송 가능하게 함으로써 트래픽 성장을 더욱 가속화 할 것으로 예측되고 있다 [1]. 이에 따라 트래픽 증가로 발생하는 네트워크 포화 현상을 해결하기 위한 연구의 중요성이 높아지고 있다. 그러나 현존하는 네트워크는 망 포화 현상으로 전송량 (Throughput)이 한계에 다다르고 있어, 멀티미디어와 같은 지연에 민감한 데이터를 전송할 경우 서비스 이용에 제한이 발생하는 문제점이 있다. 따라서 이러한 문제를 해결하기 위하여 기존의 통신 방법을 뛰어넘는 차세대 네트워크 통신 기법인 네트워크 코딩(Network Coding) [2]에 관한 많은 연구가 이루어지고 있다.

네트워크 코딩은 네트워크의 max-flow min-cut 한계를 향상시키고 지연 현상을 줄일 수 있다고 알려져 있으며, 랜덤선형 네트워크 코딩(Random Linear Network Coding) 알고리즘이 인코딩 기법으로 널리 사용되고 있다 [3]. 랜덤 선형 네트워크 코딩 알고리즘은 데이터를 전송하고자 하는 송신 단에서 임의로 생성된 코딩 계수와 데이터 사이의 선형 결합을 수행함으로써 인코딩된 데이터를 얻는 기법이다. 이러한 방법으로 생성된 데이터는 수신 단말기에서 원본 데이터로 복구되기 위하여 디코딩 과정을 거치는데, 이때 일반적으로 가우스 소거법을 디코딩 과정으로 사용하게 된다.

한편 사물 인터넷 환경에서 배터리를 기반으로 동작하며 저사양 센서 기능을 수행하는 단순 단말 기기의 경우, 가우스 소거법과 같이 $\boldsymbol{O}(n^3)$의 높은 복잡도를 요구하는 디코딩 기법은 단말 기기의 전력 소모를 높일 뿐만 아니라 지연 현상을 발생시키게 되는 문제점이 있다. 이에 따라 단순 단말 기기에 적합한 저복잡도의 디코딩 기법을 수행하는 네트워크 코딩 기법에 연구가 필요하다. 따라서 본 논문에서는 사물 인터넷과 같이 단순 단말 기기로 구성된 네트워크에서 네트워크 코딩을 기반으로 데이터를 전송할 때 Unitary Matrix 를 이용한 저복잡 네트워크 코딩 및 디코딩 방법을 제안한다.

2. 본론

2.1 제안된 네트워크 모델 및 데이터 구조

본 논문에서는 사물 인터넷 환경에서 단순 단말 기기와 서버로 구성되어 있는 서버-클라이언트 네트워크 모델을 고려한다. 제안된 네트워크는 N 개의 단말 기기 $t_1, t_2, \cdots, t_N$가 서버 s로 데이터를 요청하며, 서버 s에서 단말 기기로 데이터를 전송할 시 네트워크 코딩 기법을 이용하여 데이터를 전파한다. 본 논문에서는 채널에서의 패킷 손실은 발생하지 않는다고 가정한다.

서로 다른 N 개의 클라이언트 단말 기기는 각각 서로 다른 데이터 $\boldsymbol{X} = [\boldsymbol{x}_1^T; \boldsymbol{x}_2^T; \cdots; \boldsymbol{x}_N^T]$ 를 요청하며, 이때 $\boldsymbol{x}_i = [x_i(1), x_i(2), \cdots, x_i(L)]^T$ 는 단말 기기 t_i가 요청한 데이터를 의미하고 $x_i(j)$는 L 길이의 데이터 $\boldsymbol{x}_i$ 중 j 번째 원소를 의미한다. 본 논문에서는 $\boldsymbol{X}$ 는 실수 체계의 데이터라고 가정하며 각 데이터는 서로 선형 독립한다고 가정한다.

2.2 Unitary Matrix 를 이용한 데이터 인코딩 전략

네트워크 코딩 기법을 이용하여 데이터를 인코딩하는 서버 s 는 코딩 계수를 구하기 위하여 랭크 N의 바이너리 사각 행렬을 임의로 생성한 후 고유값 분해 (Eigen decomposition) 원리를 바탕으로 Unitary matrix $\boldsymbol{Q}(\in \mathbb{R}^{N\times N})$를 구한다. 본 논문에서는 이와 같이 생성된 Unitary matrix $\boldsymbol{Q}$ 를 코딩 계수 $\boldsymbol{C}$로 정의한다 (즉, $\boldsymbol{C}$=$\boldsymbol{Q}$). 이후 서버 s 는 다음과 같은 과정을 통하여 인코딩 된 데이터 $\boldsymbol{Y}$를 구한다.

$$\boldsymbol{y}(k)=\sum_{i=1}^{N} c_i(k) \times \boldsymbol{x}_i \quad (1)$$

여기서 $\boldsymbol{c}_i = [c_i(1), c_2(2), \cdots, c_N(N)]^T$는 코딩 계수 행렬의 i번째 열 벡터를 나타내며 $c_i(k)$는 코딩 계수 행렬의 (i,j) 원소를 나타낸다. 또한, $\boldsymbol{y}(k)$는 인코딩 된 데이터 $\boldsymbol{Y}$의 k 번째 데이터를 의미하며 $\times$ 연산자는 실수 체계에서의 행렬 곱을 의미한다. 한편, 현재 채널에서 패킷 손실은 발생하지 않는다고 가정하였으므로 인코딩 된 데이터의 개수 k는 N과 동일하다. 최종적으로 서버에서는 제안된 방식을 통하여 생성된 코딩 계수 $\boldsymbol{C}$ 와 네트워크 코딩된 데이터 $\boldsymbol{Y}$를 N 개의 클라이언트 단말 기기 $t_1, t_2, \cdots, t_N$에 전송한다.

2.3 Unitary Matrix 를 이용한 저복잡 디코딩 전략

클라이언트 단말 기기는 2.2 장에서 제안한 기법으로 네트워크 코딩 된 데이터를 복구하기 위하여 코딩 계수의 역 행렬을 구한 후 네트워크 코딩 된 데이터 $\boldsymbol{Y}$와 선형 결합을 수행한다. 한편, 본 논문에서 제안한 기법은 코딩 계수가 Unitary matrix 이므로 역행렬이 전치 행렬과 동일한 특성을 갖는다 (즉, $\boldsymbol{C}^{-1} = \boldsymbol{C}^T$). 따라서 단말 기기는 코딩 계수의 전치행렬을 구한 후 아래와 같은 방식을 통하여 디코딩 된 데이터 $\hat{\boldsymbol{X}}$를 구한다.

$$\hat{\boldsymbol{X}} = \boldsymbol{C}^T \times \boldsymbol{Y} \quad (2)$$

이를 통하여 배터리 기반의 단순 단말 기기는 기존의 높은 복잡도를 요구하는 가우스 소거법 대신, 행렬의 전치화와 곱 연산을 통하여 낮은 복잡도로 데이터를 복구하게 된다. 다음 장에서는 모의 실험을 통하여 제안된 전송 방법의 성능평가를 다룬다.

3. 시뮬레이션 결과

본 장에서는 사물 인터넷의 서버-클라이언트 모델 구조에서 제안한 저복잡 네트워크 코딩 및 디코딩 기법의 성능을 모의실험을 통하여 확인하였다. 클라이언트 단말 기기의 수가 N인 네트워크에서 데이터의 길이 L 이 1024 일 때, 서버에서 제안된 기법과 기존의 랜덤 선형 네트워크 기법을 이용하였을 때 요구되는 연산 복잡도를 비교하였다. 본 실험에서의 연산 복잡도는 Intel Core i7 3.40GHz CPU, 8.00GB RAM, Windows 7 환경의 MATLAB 에서 요구하는 실행 시간으로 측정하였다. 한편, 본 논문에서는 패킷 손실로 발생하는 전송 지연은 고려하지 않았다.

그림 2 를 통하여 확인할 수 있듯이, 본 논문에서 제안한 네트워크 코딩 및 디코딩 기법을 이용할 경우 실행 시간이 감소함에 따라 복잡도가 현저히 줄어듦을 확인할 수 있다. 또한 클라이언트 단말 기기가 증가함에 따라 제안된 기법을 통하여 얻는 이득이 증가함을 확인할 수 있다. 이는 많은 수의 사물로 구성되어 사물 인터넷 환경에 적합한 조건으로 전체 네트워크의 전송량을 크게 향상시킬 수 있을 것으로 기대된다.

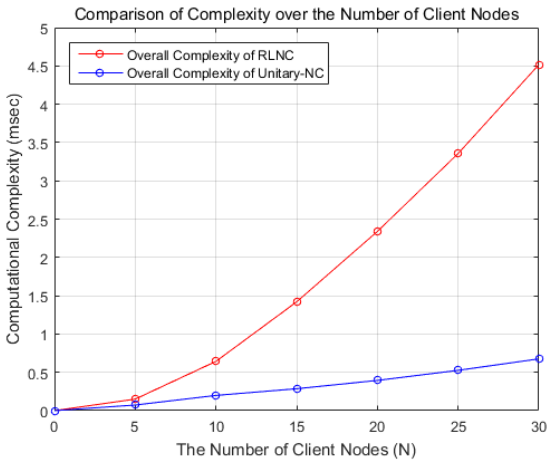


그림 2. 클라이언트 수에 따라 요구되는 연산 복잡도

4. 결론

본 논문에서는 배터리 기반의 단순 단말 기기로 구성되어 있는 사물 인터넷 환경에서 네트워크 코딩 기법을 이용하여 각 단말 기기로 데이터를 전송할 때, Unitary matrix 를 이용한 저복잡도의 네트워크 코딩 및 디코딩 기법에 대하여 제안하였다. 모의 실험을 통하여 본 논문에서 제안하는 기법이 기존의 랜덤 선형 네트워크 코딩에서 요구하는 연산 복잡도와 비교하여 현저히 감소함을 확인할 수 있었다. 뿐만 아니라, 클라이언트 단말 기기가 증가할수록 전체 네트워크에서 요구하는 복잡도가 더욱 효과적으로 감소함을 확인하였다.

5. 참고 문헌

[1] 표철식, “사물인터넷 기술 동향” 한국전자파 학회지 제 25 권 제 4 호, pp. 49-58, 2014 년 7 월.

[2] 권정민, 김순영, 이민지, 조예슬, 권민혜, 박형곤, “네트워크 코딩을 이용한 저전력 IoT 네트워크 망 관리 전략 연구”, 한국통신학회 종합 학술 발표회 논문집 (하계), pp. 944-945, 2014 년 6 월.

[3] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi and B. Leong, “A random linear network coding approach to multicast. Information Theory,” *IEEE Transactions on*, vol. 52, no. 10, pp. 4413-4430, 2016.

ACKNOWLEDGMENT

본 연구는 2016 년 정부재원(미래창조과학부 여대학(원)생 공학연구팀제 사업)으로 한국연구재단과 한국여성과학기술인지원센터의 지원을 받아 연구되었고 미래창조과학부 및 정보통신기술진흥센터 대학 ICT 연구센터 육성지원사업(IITP-2016-H8501-16-1007)의 지원을 받아 수행되었으며, 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원(No. NRF-2014R1A2A1A11051257)을 받아 수행된 연구임.

TOSCA 기반의 NFV 정보 모델링

최수길 ○, 최미정

강원대학교 컴퓨터과학과

{sgchoi89, mjchoi}@kangwon.ac.kr

요 약

NFV 는 네트워킹에 필요한 모든 유형의 자원들을 추상화하고, 소프트웨어적으로 관리 및 제어가 가능케 하는 기술을 의미한다. NFV 를 구현하여 효과적으로 운용하기 위해서는 정보 모델링이 필수적이다. VNF 간의 라이플 사이클 관리, 다양한 서비스에 대한 정의, 각 서비스 간의 관계 등을 모델링하여 효과적으로 운용 및 관리가 이루어져야 한다. 본 논문에서는 NFV 관리 정보 모델링을 위하여 OASIS 의 표준 정보 모델링 언어인 TOSCA 에 대하여 분석하고 또 다른 모델링 언어인 Yang 과의 비교 분석을 실시하였다. 또한 정보 모델링을 하기 위한 항목과 절차를 기술하고 정의된 정보 모델링에 대한 검증 방법을 서술한다.

1. 서론

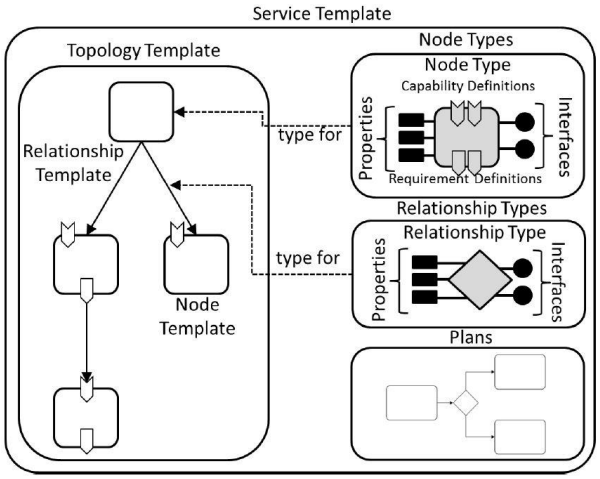
최근 각광받고 있는 기술인 NFV(Network Functions Virtualization)는 주로 미들박스와 같은 네트워크 서비스 장비내의 네트워크 기능들을 하드웨어 전용 장비로부터 고성능 범용 서버 등으로 분리시켜 소프트웨어적으로 유연하게 제어 및 관리가 가능하도록 가상화시키는 기술을 말한다. NFV 의 가장 중요한 요소는 네트워크 서비스 기능을 표준화된 형태로 어떻게 가상화를 지원하느냐에 달려있다. 일반적으로 NFV 의 구현 및 동작 방법은 기존의 하드웨어 내에 구현된 네트워크 기능을 컴퓨팅/네트워크 자원의 클라우드 형태로부터 소프트웨어 형태로 가상화하여 이를 VNF(Virtualized Network Function)로 만들고, 이를 서비스 체인(Service Chain) 형태로 연결하여 지원하도록 하는 형태이다. NFV 를 구현하는 방법은 다양하며, 이를 표준화하기 위하여 유럽표준화기구인 ETSI 에서는 통신사업자 중심으로 2012 년 NFV ISG(Industry Specification Group)를 구성하여 활발한 표준개발 작업을 진행하고 있다.

NFV 를 구현하여 효과적으로 운용하기 위해서는 정보 모델링이 필요하다. NFV 에서의 정보 모델링이란 VNF 간의 라이프 사이클 관리나, 다양한 서비스에 대한 정의, 각 서비스간의 관계 등에 대한 정의를 명시하는 것이다. 정보 모델링이 이루어지게 되면 효과적인 운용 및 관리뿐만 아니라 확장성을 제공할 수 있다. 정확한 정보 모델링에 대한 표준화가 이루어지고 검증이 완료되면 어떠한 오케스트레이터에 대해서도 모델링에 대한 포맷만 만족한다면 다양한 인프라에서 자기만의 서비스 시나리오가 적용될 수 있으므로 확장성을 제공할 수 있다. 본 논문에서는 NFV 관리 정보 모델링을 위하여 OASIS 의 표준 정보 모델링 언어인 TOSCA(Topology and Orchestration Specification for Cloud Applications)에 대하여 분석하고, 네트워크 장비에 대한 구성관리 정보 모델링 언어인 Yang 데이터 모델링과의 비교를 실시한다. 실제 정보 모델링을 하기 위하여 필요한 절차와 항목들에 대해서 나열하고 정의된 정보 모델링에 대한 검증방안을 제안하고자 한다.

2. TOSCA

TOSCA 는 클라우드의 토폴로지에 대한 설명을 명세하기 위한 OASIS(Organization for the Advancement of Structured Information Standards)에서 제정한 표준 언어이다. TOSCA 는 서비스 컴포넌트들에 대한 정의와 각 컴포넌트들에 대한 관계를 Service Topology 를 사용하여 제공하며, 서비스 생성 및 수정에 대한 관리의 처리과정을 Orchestration 을 사용하여 설명[1]할 수 있다. TOSCA 는 IT 서비스를 정의하기 위한 메타모델을 정의한다. 메타 모델은 서비스들을 관리하는 방법뿐만 아니라 서비스의 구조에 관한 것들도 정의할 수 있다. TOSCA Service Template 구조에서 Topology Template 는 서비스의 구조를 정의하며, Plane 은 전체 생명주기 동안에 서비스의 관리와 생성, 삭제 등에 사용되는 프로세스 모델을 정의한다. Topology Template 는 Node Template 와 Relationship Template 로 구성되어 있으며 각각은 방향 그래프로서 서비스의 토폴로지 모델을 정의한다. Node Template 는 그래프의 노드를 표현하며 각 서비스 컴포넌트의 Node Types 에 대하여 명세하고 있다. Node Type 은 컴포넌트들의 성질을 정의하며, 컴포넌트를 조작하기 위하여 필요한 오퍼레이션들을 정의하는데 사용된다. Relationship Template 는 노드들의 관계를 명세하는데 사용되며 각각의 Relationship Template 는 관계의 성질이나 의미를 정의하는데 사용되는 Relationship Types 를 나타낸다. 다음 그림 1 은 TOSCA Service Template 의

구조를 나타낸 것이다.



(그림 1) TOSCA Service Template 구조

3. TOSCA 와 Yang 비교

Yang 은 NetConf 네트워크 환경설정 프로토콜을 위한 데이터 모델링 언어[6]이다. 즉, Yang 은 NetConf 의 오퍼레이션과 내용 계층(content layer)을 모델링하는데 사용된다. Yang 데이터 모델링 언어는 IETF 소속의 NETMOD(Netconf Data Modeling Language) WG 이 개발했으며 2010 년 10 월에 RFC 6020 로 정의되었다. 최근 Yang 모델링 언어는 NetConf 의 전송 내용을 위해 표준화되었으나 NetConf 프로토콜에 종속되지 않고 다양한 모델링에 적용되고 있다.

TOSCA 의 장점은 Orchestration 이다. 즉, TOSCA 는 복잡한 어플리케이션 환경에서의 다양한 서비스나 관계에 대해서 설명[2]할 수 있다. Yang 의 장점은 네트워크 장비에 대한 구성관리 측면이다. 예를 들면 Yang 으로 모델링 된 라우터를 NETCONF 를 통해 구성할 수 있다. 구성관리에 대한 정보는 누구나 쉽게 확인할 수 있고, 다른 구성관리와 비교하여 쉽게 이식할 수 있어 확장성을 갖는다. TOSCA 는 전체 시스템의 토폴로지 선언이나, 서비스에 대한 생성, 수정 등에 대한 선언에 사용되고, Yang 은 실제 서비스 내의 장비나 서비스의 정보 설정에 사용될 수 있다. 이것은 TOSCA 와 Yang 이 서로 경쟁관계가 아니라 서로 상호 의존적으로 사용될 수 있다는 것을 나타낸다.

4. TOSCA 기반의 NFV 정보 모델링 방안

본 장에서는 앞서 살펴보았던 TOSCA 를 이용하여 NFV 에 대한 정보 모델링을 정의하기 위해 필요한 항목들을 나열한다. 먼저 TOSCA 를 이용한 정보 모델링에 앞서 TOSCA 템플릿에 대한 연구 및 분석이 이루어 져야 한다. TOSCA 에 대한 컨셉과 유즈케이스, 요구사항 등을 분석하고 TOSCA 언어에 대한 정책과 문법에 대하여 분석한다. 두번째는 ETSI NFV 에서 제시하는 TOSCA 템플릿의 포맷을 준용하기 위하여 TOSCA Simple Profile for (NFV) Version 1.0 Specification[3]를 분석한다. NFV 에서 TOSCA 템플릿 Deployment 에 대해서 분석하고 VNF, VNFD, VLD, VNFFGD, NSD 에 대한 TOSCA 템플릿을 분석한다. 세번째는 앞서 진행했던 분석을 바탕으로 정보 모델링에 대한 정의를 진행한다. VNF 간의 라이플 사이클 관리, Deployment, 서비스의 다양한 요소 등을 정의한다. 특히 TOSCA 의 서브셋인 YAML 을 분석하여 YAML 포맷[4]을 이용한 정보 모델링에 대한 서비스 템플릿이 개발되어야 한다. 마지막으로 정의된 정보 모델링을 검증하기 위하여 오픈 소스 플랫폼인 Cloudify[5]를 이용하여 네트워크 관리 툴을 개발하고, 개발된 관리 툴에 정의된 서비스 템플릿을 적용하여 효율적인 관리가 이루어 지는 것을 검증한다. 최종적으로 정의된 서비스 템플릿을 다양한 환경의 오케스트레이터에 적용하여 모델링의 효율성 및 서로 다른 환경에서의 확장성을 검증한다.

5. 결론

본 논문에서는 최근 각광받는 기술인 NFV 을 효율적으로 관리하기 위한 방법으로 TOSCA 정보 모델링을 서술하였다. TOSCA 는 시스템의 다양한 서비스에 대한 정의, 서비스간의 관계 등을 모델링 하는데 사용된다. 또한 TOSCA 와 네트워크 장비 구성관리 관련된 정보 모델링언어인 Yang 과의 비교 분석도 실시하였다. 마지막으로 TOSCA 기반의 NFV 정보 모델링을 위하여 필요한 절차와 항목들에 대해 나열하고 각각의 항목들에 필요한 내용들을 서술하였다.

향후 계획으로는 클라우드 오픈 소스 플랫폼인 Cloudify 를 이용하여 네트워크 관리 툴을 개발하고 정의된 정보 모델링을 적용하여 효율성을 검증하고자 한다.

6. 참고 문헌

[1]Derek Palma, Thomas Spatzier, *Topology and Orchestration Specification for Cloud Applications Version 1.0, OASIS*, 2013.

[2]Matt Rutkowski, *TOSCA - An Open Standard for Cloud Application Portability, OASIS, 2014*

[3]Shitao Li, *TOSCA Simple Profile for Network Functions Virtualization(NFV) Version 1.0, OASIS*, 2015.

[4]Derek Palma, Matt Rutkowski, Thomas Spatzier, *TOSCA Simple Profile in YAML Version 1.0, OASIS*, 2014.

[5]Cloudify DOCS 3.4.0. (2016) [Online]. Available http://docs.getcloudify.org/3.4.0/intro/what-is-cloudify/

[6] RFC 6020: Yang Base Specification

미래네트워크선도시험망(KOREN) 현황 및 향후 발전전략

김형우 ○, 양종한

케이티 기업사업수행본부, 한국정보화진흥원(NIA) ICT 융합본부

hyoungwoo.kim@kt.com, yjh@nia.or.kr

요 약

이 논문은 미래네트워크선도시험망(KOREN) 구축 운영에 관한 것으로서 2015 년도의 주요 수행결과를 소개한 논문이다. 내용은 크게 국내외 연구시험망 구축 운영, 국내외 공동 실증시험 및 협력, 네트워크 시험검증, 미래네트워크선도시험망(KOREN) 운영센터 환경 구축 및 운영, 그리고 주요성과로 구성되어 있다. 이 논문을 통하여 미래네트워크선도시험망(KOREN)이 무엇이며, 어떻게 산학연 연구개발분야에 활용될 수 있는 지를 제시하고, 향후 이용 활성화에 보탬이 되기를 기대한다.

1. 개요

미래네트워크선도시험망(KOREN, Korea advanced REsearch Network)은 미래창조과학부와 한국정보화진흥원에서 운영하는 비영리 선도시험망으로 상용망에 적용하기 어려운 미래 네트워크 기술에 대한 시험검증과 실증시험을 산·학·연에 지원해 연구역량 강화에 나서고 있으며, 현재 전국 8 개 대도시 지역(서울, 수원, 대전, 광주, 대구, 부산, 제주, 강원)과 미국, 유럽, 아시아 등 해외 65 개국 연구망과 연동되어 있다. 미래네트워크선도시험망(KOREN)은 선도성, 비영리성 등의 성격을 지니므로 민간사업자는 투자하기 어려운 정부 주도형 국가전략사업의 망으로서 1995 년 선도시험망을 시작으로 초고속선도망, 광대역통합 연구개발망을 거쳐 현재 미래네트워크선도시험망(KOREN)으로 지속적으로 점진적으로 고도화가 추진되고 있다. 미래네트워크선도시험망(KOREN)은 초고속국가망, 초고속인터넷망, 광대역통합망, IPTV, IPv6, 전자정부망 등 국가적 추진이 필요한 사업을 수요자와 연구개발자 관점에서 실증하고 사업화 하기 위한 인프라를 제공하고 있으며, 또한, 최근에는 인터넷 한계를 극복하기 위한 SDN/NFV 차세대네트워크 기술과 서비스를 객관적이고 안정적으로 시험하는 분야에 활용되고 있다.

한편, 미래네트워크선도시험망(KOREN)은 국제연구망을 중심으로 글로벌 연구협력체계를 구축하고 이를 통해 일본, 유럽, 중국 및 미국 등과의 국제공동연구 발굴 및 기술개발, 표준화 활동에 참여할 수 있는 기반을 마련하고 있다. 학계, 연구소, 중소기업 등에 미래 네트워크 관련 기술 및 응용서비스 분야의 R&D 시험 검증을 지원하고, 국내 선도시험망(KOREN) 및 국제 선도시험망(APII, TEIN)을 안정적으로 운영함으로써 이용기관에게 세계 최고 수준의 네트워크 테스트베드 인프라를 제공하고 있다.

미래네트워크선도시험망(KOREN)을 기반으로 연구 결과물의 시험 검증이 가능한 체계를 마련하여 연구개발 결과가 상용화 단계로 이어질 수 있는 선순환 단계의 연결고리를 제공하고, 최근 Big Data, 사물인터넷(IoT), Cloud, 고신뢰 네트워크 등 국정과제에 대한 시험 검증 테스트 인프라를 제공하여 글로벌 시장에서 선도적 지위를 확보하는데 이바지하고 있다. 이 논문에서는 크게 (1) 국내외 연구시험망 구축 운영, (2) 국내외 공동 실증시험 및 협력, (3) 네트워크 시험검증, (4) 미래네트워크선도시험망(KOREN) 운영센터 환경 구축, 그리고 (5) 주요성과로 나누어 각각에 대하여 간략히 소개하고자 한다.

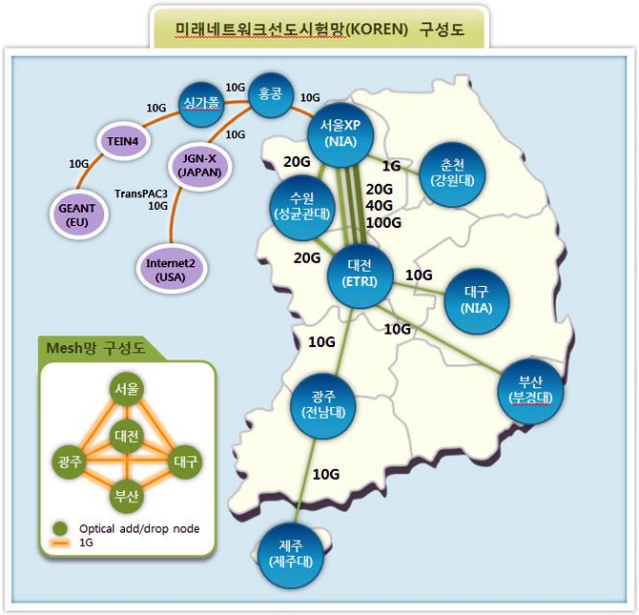
2. 국내외 연구시험망 구축운영

가. 국내 선도시험망 구축운영

ICT 신기술 및 SDN 등 미래네트워크 분야에 대한 연구시험 및 독립적 테스트에 대한 필요성이 요구됨에 따라, 산업계, 학계 및 연구소 등이 미래네트워크 관련 최근 기술들을 다양하게 시험, 검증할 수 있는 백본 전송망을 제공하였다. 정부의 미래네트워크선도시험망(KOREN) 구축 사업 추진 정책에 따라 전국 8 개 지역접속점을 중심으로 백본 전송망을 구축, 운영하였다. (그림 1)에 제시되어 있는 바와 같이 서울 – 대전 구간은 160G 로 연동되어 있으며, 서울 – 수원 – 대전 구간은 10G*2 회선이 연동되어 있다. 백본 전송망은 전국 6 개 지역접속점(POP)간 ROADM 을 기반으로 구성되어 있으며, 보호절체(Protection) 기능을 포함하여 50ms 이내 우회절체가 가능하도록 되어 있다. 전국은 3 개의 Multi-Ring(수도권링, 영남링, 호남링)형 Topology 로 구성되어 있으며, 전송망 장애 시 이중화된 다른 루트로 자동 절체가 가능하도록 되어 있다.

미래네트워크선도시험망(KOREN)망의 개방형 연

구과제 활성화 및 상용망을 통한 외부 선도시험망과의 연계 시험 강화를 위해 국내 최대 상용 인터넷망인 KORNET 망과 연동되어 있다. 물리적으로 총 20Gbps 속도(10Gbps*2 회선)를 제공하고 있다. 한편, 수백 기가급의 연구데이터 전송, 미래인터넷 신규장비 시험을 위하여 미래네트워크 테스트베드가 필요하며, 백본망과 분리된 별도의 시험망에 대한 요구가 증가함에 따라 서울 – 대전 구간에 2core, 그리고 서울 – 수원 구간에는 4core 광코어(Dark Fiber)를 각각 제공하고 있다.



(그림 1) 미래네트워크선도시험망(KOREN) 망 구성도

SDN 등 미래네트워크 연구과제 실증시험과 대용량 트래픽 전송 연구 등 이용기관의 연구를 위하여 1Gbps~10Gbps 까지 다양한 대역폭의 가입자회선을 제공하였는데, 2015 년 12 월 현재 총 62 개 이용기관이 미래네트워크선도시험망(KOREN) 전용회선을 이용하고 있다(표 1 참조).

<표 1> 미래네트워크선도시험망(KOREN) 이용기관 현황

속도	이용기관
10G	고려대학교, 광주과학기술원(GIST), 분당서울대병원, 연세의료원, 스맥(SMEC), 아토리서치코리아(AttoResearch), 경기과학기술진흥원, 경북대학교 (8개)
	대구경북과학기술원(DGIST), 에프아이시스(Fisys) (2 개)
	한국과학기술정보연구원(KISTI)(홍릉, KREONET), 한국전자통신연구원(대전 ETRI), 한국전자통신연구원(광주 ETRI) (3개)
1G	건국대학교, 경희대학교, 고려대 세종캠퍼스, 국립암센터, 기상청, 서울대학교, 서울대학교 의대, 서울시립대학교, 서울아산병원, 순천향대학병원, 안양대학교, 연세대학교, 우주측지관측센터, 인하대학교, ㈜크레블, ㈜퓨쳐시스템, 충북대학교, 코위버㈜, 한국과학기술연구원(KIST), 한국과학기술원(KAIST-서울캠퍼스), 계명대, 라이트웍스㈜, 포항공대, 한양대학교병원, 엘에스웨어㈜, 전북대학교병원, 삼성서울병원, 한국인터넷진흥원_창조(KISA-서초), 강남세브란스병원, 아이엔소프트(INSoft), 동국대학교 실감미디어 성과확산 사업단, 쿨클라우드, 나임네트웍스 (33 개)
	FAU(Friedrich-Alexander Universitat Erlangen-Nurnberg) 부산캠퍼스, 개방형융합미디어산업협회, 디오넷, 경북대학교상주캠퍼스 (4 개)
	엠투소프트, 욱성미디어, 위스캔, 유프리즘, 새하컴즈, 네이블 커뮤니케이션즈, 부경대학교, 성균관대학교, 전남대학교, 제주대학교, 재단법인 테인협력센터, 한국네트워크산업협회 (KANI) (12 개)

나. 국제 선도시험망 구축운영

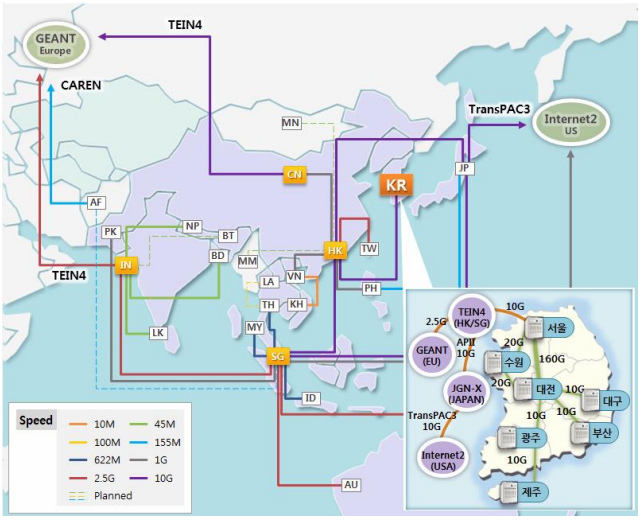
(1) APII 회선 구축 운영

한국 – 일본 간 연결되어 있는 국제회선인 APII (Asia Pacific Information Infrastructure)회선은 APEC TEL 협력사업으로부터 시작되어, 아태지역 선도시험망을 연동하여 IT 교류협력 및 응용서비스 개발에 활용되고 있다. 특히, 고에너지 물리분야, 의료, 미래네트워크 분야의 여러 가지 다양한 연구가 진행되는 가운데 한일 간을 연결하는 DCN 및 Openflow 등의 테스트베드 연계 활동이 지속적으로 추진되고 있다

지난 2015 년 3 월 한일 관계기관 간 협력미팅을 통해 2016 년 3 월 이후 APII 국제회선을 끊기로 결정함에 따라 APII 회선은 2016 년 4 월부터 회선서비스를 제공하지 않게 되었다. 하지만 APII 사업은 홍콩을 통한 우회루트를 통해 커넥션을 유지시켜 국제연구협력활동을 지속적으로 지원 할 예정이다.

(2) TEIN 회선 구축 운영

한국 – 유럽간 백본 회선인 TEIN (Trans-Eurasia Information Network)은 2007 년 1 월부터 유럽의 GEANT 와 아시아 23 개국으로 연결되어 아시아와 유럽간 IT 및 첨단 응용기술의 활용연구를 위한 망 인프라로 활용되고 있으며, 관련 국제협력을 수행하고 있다(그림 2 참조). TEIN 과 관련하여 한국정보화진흥원(NIA)는 2009 년부터 한국 – 홍콩, 홍콩 – 싱가폴 구간의 국제 회선을 직접 투자하여 TEIN 사업 자체에 참여하고 있다. 또한, 망 운영과 관련해서는 2009 년초에 TEIN NOC 와의 망 운영 협력을 통해 국제선도시험망 백본 운영에 참여하고 있다.



(그림 2) TEIN4 회선 구성도

TEIN 백본 환경을 기반으로 한국 – 아시아 – 유럽 전 구간에 걸쳐 대용량 대역폭을 필요로 하는 다양한 시험검증 및 응용분야 연구의 원활한 국제 공동협력을 기대하고 있다. 특히, 최근 활성화되고 있는 원격의료, 기상, 고에너지 물리 등 응용분야 인프라로 활용될 수 있으며, 또한 우리나라의 우수한 ICT 기술을 동남아에 전달할 수 있는 교두보가 될 것으로 기대된다.

3. 네트워크 시험검증

연구개발의 선순환 체계인 기술개발 → 시험검증 → 실증시험 → 상용화에 있서 시험검증 단계 강화를 위해 미래네트워크선도시험망(KOREN)을 활용하였다. 2015 년도에는 총 13 개 시험검증 사업을 수행하였으며, 대상사업 중 12 개 과제는 ETRI 에서, 1 개 과제는 TTA 에서 시험검증을 추진하였다(표 2 참조).

<표 2> 2015 년도 시험검증 사업

No.	과제명
1	Non-Stop Active Routing 을 지원하는 고가용성 네트워크 운영체제 기술 개발
2	초소형 100G 클라이언트 트랜시버용 TOSA 개발
3	하향 40Gbps / 상향 10Gbps 지원 차세대 수동형 광가입자망 시스템 기술 개발
4	SDN/Openflow 기반의 기업용 네트워크 컨트롤러 기술 개발
5	Optical SDN 구현을 위한 Metro 용 Port-agnostic 광송수신 기술 개발
6	NFV 개념의 멀티서비스 맞춤형 스위칭 시스템 및 운영체제 개발
7	상하향 1Gbps 가입자망 폰스틱 개발
8	개방제어 기반 분산구조 모바일 코어 네트워크 기술 개발
9	SDN 기반 동적 네트워크 은닉 핵심 기술개발
10	SDN/NFV 기술을 이용한 혁신적 IoT 서비스 인프라 상용화 개발
11	이기종 고용량 RAN 을 지원하는 프론트홀 및 백홀 통합기술 개발
12	테라급 코어 라우터 상용화 개발
13	SDN/NFV 기반 IoT GW 원격관리시스템

(그림 3) 차세대네트워크 시험검증센터 전경

미래네트워크선도시험망(KOREN) 네트워크 기반 시험검증 및 실증시험테스트베드는 현재 ETRI 내의 융합기술생산연구센터 내 구축되어 운용 중에 있으며(그림 3 참조), 전송 및 네트워크 장비의 상용화 촉진에 힘쓰고 있다. 이와 함께 국내 중소업체를 대상으로 고가의 시험계측장비를 이용할 수 있게 함으로써 산업체의 애로기술 해소에도 기여하고 있다.

4. 운영센터 환경구축

한국정보화진흥원(NIA)와 경기과학기술진흥원(GSTEP)은 2015 년 창조경제 거점인 판교테크노밸리 내에 미래네트워크센터(FNC: Future Network Center)를 구축하였다(그림 4 참조). 미래네트워크센터(FNC)는 네트워크 장비 및 관련 서비스(SW, 컨텐츠, 게임 등)의 개발부터 상용화까지 전 단계를 체계적으로 지원하기 위한 국내 유일의 개방형 플랫폼 센터로서 단계별 기술지원 및 신뢰성 시험 등의 환경을 제공하는 센터이다. 한국정보화진흥원(NIA)과 경기과학기술진흥원(GSTEP)은 이를 위해 지난해 말 판교테크노밸리 경기창조경제혁신센터로 미래네트워크선도시험망(KOREN) 운영센터(NOC) 이전을 성공적으로 마쳤으며, 향후 이를 통해 미래네트워크센터(FNC) 중심의 전국지역 거점의 기업지원을 추진해 나갈 예정이다.

미래네트워크센터(FNC)를 통해 네트워크 장비 제조사와 관련기업들은 그 동안 해외에서 수행해오던 실증환경 테스트를 국내에서 무료로 할 수 있어 연간 100 억원 이상의 비용 절감 효과가 기대된다. 미래네트워크센터(FNC)에서는 개방형 고속대용량 통신 테스트 인프라를 제공하여 ICT 기업의 연구개발, 신뢰성, 사업화 등 기업경쟁력 강화를 지원하며, 스타트업/중소· 중견기업의 연구개발, 사업화 지원 체계를 구축하고, 또한 공공안전 관련 제품 및 서비스의 실증테스트가 가능하도록 지원할 계획이다.

향후 미래네트워크선도시험망(KOREN)은 판교 미래네트워크센터를 중심으로는 산 · 학 · 연 중심의 미래 ICT 기술의 사업화/상용화를 지원하고 이를 확대하여 스타트업 / 1 인기업 등 기술은 있으나 통신 기반시설이 부족한 기업을 적극 지원하여 국내 미래네트워크 기술 기업의 차세대 네트워크 · 컴퓨팅 장비나 소프트웨어 솔루션분야 연구개발의 전주기

지원체계 환경이 조성될 수 있도록 노력할 예정이다.

(a) 사무실 전경

(b) 전산실 전경

(그림 4) 미래네트워크 운영센터

5. 미래네트워크선도시험망(KOREN) 주요성과

미래네트워크선도시험망(KOREN)은 산업체, 학계, 연구기관 등이 무료로 이용할 수 있는 광대역, 고품질의 국내외 선도시험망으로서 미래네트워크 관련 기술의 시험검증과 첨단 ICT 국제공동 협력기반을 조성하고 있다. 이와 같은 네트워크 환경 하에 2015 년도에도 여러 가지 국내외 실증시험이 수행되었으며, 대표적인 성과를 요약해보면 다음과 같다.

- 글로벌 실증시험 분야: 분당서울대학교병원 주관 국제공동의료연구, 개방형융합미디어산업진흥협회 주관 베트남 호치민 한류산업 진출
- 국내 실증시험 분야: 경희대학교 재난대응 테스트베드, 전남대학교 SDN/NFV 환경 공격탐지/방어 실증, 광주과학기술원 OF@KOREN 검증, ETRI 차세대 광네트워크 장비(POTN) 성능 검증

가. 분당서울대학교병원 국제공동 의료연구

분당서울대병원은 의료 ICT 분야에 있어 선진대열에 있는 병원으로서 차별화된 국제 연수 프로그램을 보유하고 있다. 원격 의학교육 분야를 헬스케어 전략 사업으로 추진하고 있으며, 또한 의료정보 시스템에서 축적한 다양한 경험을 국산 ICT 기술과 융합하여 국가 간 의료 소통에 활용하고 있다. 2015 년도 주요 사업내용은 다음과 같다.

- 말레이시아 케방산대학병원 간 고화질 원격의료교육 테스트베드환경 성능시험
- 10G live-Surgery 복강경 수술 4K 전송 실험(그림 5)
- NREN/TEIN 활용 의료교육망 연동시험
- MTC(Medical Tele Collaboration) 워크숍 개최

(그림 5) 제 12 차 아시아태평양 내시경 복강경 학술대회

나. 한류 컨텐츠 베트남 진출

베트남 국립 호치민사범교육대학교 내에서 한국어를 배우는 학생들이 TV 와 스마트폰에서 한국어 교육방송을 시청할 수 있도록 무선 네트워크를 구축하고, KOREN – TEIN – VinaREN 네트워크를 기반으로 한류 콘텐츠 방송시스템을 구축하였다. 베트남의 경우 수십 km 의 떨어진 섬과 섬을 네트워크로 연결해야 하는데 국내 회사가 개발한 1 쌍의 안테나인 무선브리지를 활용하여 광대역 고속통신이 가능하게 하였다. 이와 같은 인프라를 통하여 대역폭이 작고 속도가 느린 베트남에서도 한국어교육방송을 깨끗하게 시청할 수 있게 되었으며, 한류 컨텐츠가 해외에 진출할 수 있는 계기가 마련되었다.

다. APII/TEIN 망을 이용한 재난대응 SDN 테스트베드 구축 / 실증

경희대학교에서는 APII/TEIN 망을 이용하여 재난대응 SDN 테스트베드를 구축하여 실증을 하였으며, 연구과제 주요 목표는 다음과 같다.

- KOREN-APII/TEIN 망을 이용한 SDN 기반 재난망 테스트베드 구축
- 공공 재난정보, 환경센서 정보 기반 네트워크 장비 신뢰도 평가기술 개발
- 재난 복구를 위한 SDN Controller Application 제어 모듈 개발
- 재난 데이터 공유 및 빅데이터 분석 플랫폼 개발
- 국제선도시험망 기반 재난대응 기술 실증시험

KOREN-APII/TEIN 망을 이용한 SDN 기반 재난망 테스트베드 구축 분야에 있어 SDN Controller 인 Floodlight 를 이용하여 SDN 기반 제어환경을 구축하였으며, Quagga 와 Open vSwitch 를 이용하여 사이트별 노드를 연결하고, Open HW 인 Raspberry Pi 2 와 OpenWrt 를 이용하여 Wi-Fi 기반의 WLAN 환경을 구축하였다.

공공재난정보 및 환경센서정보 등 외부 재난정보 기반 네트워크 장비 신뢰도 평가기술 개발 분야에 있어서 USGS(U.S. Geological Survey)에서 제공하는 지진데이터 등 공공재난 데이터를 수집하는 플랫폼과 이를 기반으로 네트워크 장비 신뢰도를 평

가하는 시스템을 개발하였다. 또한 재난 복구 네트워크를 위한 SDN Controller Application 재난 복구 제어모듈을 개발하였으며 sFlow 기반으로 Flow 를 실시간으로 감시하고 관찰하는 시스템을 개발하여 네트워크상의 트래픽을 모니터링 할 수 있게 하였다.

재난 데이터 공유 및 빅데이터 분석 플랫폼 개발 분야에 있어 사물인터넷(IoT) 기반의 재난 데이터를 수집하고 빅데이터 분석 플랫폼과 연동할 수 있는 Thin-Gateway 를 개발하였으며, 공공 재난 데이터를 수집, 분석하고 이를 공유할 수 있는 빅데이터 분석 플랫폼을 개발하였다.

한편, 국제선도시험망 기반 재난대응 기술 실증 시험 분야에 있어서 일본 규슈대, 경희대, 전남대, 경남대, 제주대를 연결하는 SDN 기반의 재난 망 테스트베드를 구축하고 USGS(U.S. Geological Survey) 및 University of California, Irvine 의 재난 데이터를 수집하여 분석할 수 있는 환경을 구축하였다.

라. SDN/NFV 미래네트워크 환경에서의 공격탐지 및 방어 실증

전남대학교에서는 SDN/NFV 등 미래네트워크 환경에서의 공격탐지 및 방어에 관한 실증시험을 수행하였으며, 주요 연구내용은 다음과 같다.

- ❍ Snort 와 sFlow 설치 및 SDN 연결을 통한 KOREN 테스트베드 구축
- ❍ KOREN 테스트베드상에서 제안된 네트워크 공격 탐지/방어 시스템 검증
- ❍ 시스템 부하 및 방어 능력 분석
- ❍ 네트워크 공격 탐지/방어 시스템 개선안 제안

구현한 공격 탐지/방어 시스템을 실제 스위치 네트워크에 적용하고 그 동작을 검증하기 위해, 실험실 내 로컬 테스트베드 및 전남대와 제주대를 연결하는 테스트베드를 구축하였다. 이 연구를 통해, SDN 환경에서 다양한 네트워크 플러딩 공격을 샘플링 기반 네트워크 공격탐지 및 방어 시스템으로 자동으로 방어 할 수 있음을 실증하였다.

마. SmartX 플랫폼의 OF@KOREN Playground 적용 및 검증

광주과학기술원에서는 IoT-SDN-Cloud 대응형 오픈 SmartX 플랫폼의 OF@KOREN Playground 적용 및 검증을 수행하였다. 국내외 14 개 사이트에 분포한 오픈소스 소프트웨어 OpenFlow/OpenStack 기반 SDN-Cloud 대응 실증개발 환경 구축 및 운용경험을 활용하여 연구가 진행되었으며, 주요 결과는 다음과 같다.

- ❍ SmartX Automation 자동화 소프트웨어 개발/개선을 통한 IoT-FastData-Cloud 통합 인프라 제어/관리 자동화
- ❍ 오픈소스 SDN 제어기를 활용한 라우팅 기반의 L3 네트워크 연결성 확보
- ❍ OF@KOREN Playground 사용자를 위한 가상 놀이터 환경 제공
- ❍ IoT-FastData-Cloud 인프라/서비스 지원 참조모델 실증

바. 차세대 광네트워크 장비(POTN) 성능 검증

국내에서 개발된 차세대 광네트워크 장비가 미래네트워크선도시험망(KOREN)에 적용되어 그 기능과 성능을 검증 받았는데, ETRI, 텔레필드, 우리넷 등이 공동개발한 광· 회선· 패킷 통합스위치시스템(POTN)을 ETRI 주관 하에 한 달 동안 서울 – 수원, 서울 – 대전 구간의 광케이블(Dark Fiber)에 설치하여 현장적용시험을 진행하였다. 시험에 적용한 장비 성능은 스위치 용량이 3.2 테라급이며, 400 기가급 중용량 통합 스위칭 시스템이다. 이 장비는 각 서비스 특성에 맞게 광, 회선, 패킷의 다양한 전송망 자원을 유연하게 할당해 전송할 수 있어 기존 장비보다 소비전력과 비용을 60% 가량 절감할 수 있다는 장점을 갖고 있다.

6. 향후 발전전략 및 제언

지금까지 미래네트워크선도시험망(KOREN)에 대한 개요 및 2015 년도 주요 성과에 대하여 간략히 살펴보았다. 향후에는 네트워크 관리의 효율화를 도모하고 신규 서비스의 창출가능성을 검증하며, 네트워크와 산업 분야가 융합된 선도시험망 오픈랩을 구축할 예정이다. 이렇게 구축된 오픈랩을 기반으로 중소벤처 기업이 네트워크와 산업 분야가 융합된 고부가 서비스 및 비즈니스 모델을 발굴하고 이를 시장환경에서 실증할 수 있도록 지원할 예정이다. 뿐만 아니라 네트워크 신기술· 장비를 선도시험망을 통해 민관이 협력하여 검증하도록 하는 등 선도시험망의 R&D 기반 인프라로서의 활용성을 극대화할 계획으로 있다.

학계, 산업계, 연구계에 지속적으로 고품질의 광대역 회선을 제공하고 SDN 가상망 제공 플랫폼, 컴퓨팅/스토리지 환경, 5G 재난안전통신망의 실증을 지원하는 등 새로운 구조의 네트워크 기술을 시험할 수 있도록 현재의 네트워크를 고도화할 예정이다. 아울러 국제적인 협력을 통해서 시험 환경을 연계 구축하고, 국내 연구 개발물을 해외 현장에 적용하여 현지 진출의 교두보가 될 수 있도록 노력을 경주할 예정으로 있다.

미래네트워크선도시험망(KOREN)에 대한 자세한 내용 및 이용안내는 홈페이지(www.koren.kr)에 제시되어 있으며, 광대역, 고품질 네트워크를 통하여 연구개발 및 실증시험이 더욱 확대되기를 기대한다.

7. 참고 문헌

[1] 홈페이지: http://www.koren.kr/koren/kor/index.html

[2] 디지털타임스_2016 년 2 월 17 일 11 면: ETRI 차세대 광네트워크 장비 성능검증.

[3] 한국정보화진흥원, 2015 년 미래네트워크선도시험망 구축·운영에 관한 연구, NIA III-RER-B-15013, 미래창조과학부, 2015.12.31.

ONOS 클러스터의 성능 분석

한 솔, 장석원, 서동은, 백상헌*

고려대학교 전기전자공학과

{hs1087, imsoboy2, fever1989, shpack}@korea.ac.kr

요 약

SDN(Software Defined Networking)의 적용 범위가 초기 작은 규모의 데이터 센터 네트워크에서 큰 규모의 캐리어급 네트워크로 확장됨에 따라 SDN 제어평면의 고 가용성 및 확장성이 가장 중요한 이슈로 떠오르고 있다. 이에 따라 여러 대의 컨트롤러들을 클러스터링하여 논리적으로 하나의 컨트롤러로 동작하게 하여 가용성과 확장성을 보장하는 방법의 연구가 진행 되고 있다. 본 논문에서는 오픈소스 기반의 SDN 컨트롤러인 ONOS 에서 제공하는 컨트롤러 클러스터링을 네트워크 상태 데이터베이스 파티션/동기화 및 컨트롤러 장애극복/부하 분산 메커니즘의 관점에서 분석한다. 또한, ONOS 클러스터의 플로우 룰 설치 처리량 및 컨트롤러 장애 발생시 장애 극복 시간에 대한 실험 결과를 보인다.

1. 서론

소프트웨어 정의 네트워킹(SDN: Software Defined Networking)은 기존 네트워크 인프라의 한계를 극복하기 위해 떠오르는 패러다임 중 하나이다. SDN 은 네트워크 장비의 제어기능을 중앙의 컨트롤러로 집중화하고 컨트롤러와 네트워크 장비 사이의 인터페이스를 정의하여 네트워크를 소프트웨어 기반으로 제어하는 기술을 말한다. 중앙 집중화된 SDN 컨트롤러는 쉽게 전체 네트워크에 대한 정보를 얻을 수 있고, 이를 통하여 최적의 네트워크 서비스를 제공할 수 있게 된다. 다시 말해서, SDN 은 효율적인 네트워크 제어와 네트워크 관리비 절감 및 빠른 서비스 제공 등의 많은 장점을 갖고 있다.

한편, SDN 기술의 네트워크로의 적용 범위는 초창기 작은 규모의 데이터 센터 네트워크 또는 캠퍼스 수준의 네트워크에서 큰 규모의 캐리어급 네트워크로 확장되었다. 캐리어급 네트워크는 그 규모가 크고 주어진 임무에 필수적으로 반응해야 한다는 특징을 갖는데, 이러한 특징들을 만족하기 위해서 SDN 의 제어평면이 반드시 높은 가용성과 확장성을 보장하도록 설계 되어야 한다. 따라서, SDN 의 제어평면을 여러 SDN 컨트롤러들로 클러스터링 하여 설계하고 관리하게 된다. 이에 따라, SDN 컨트롤러들의 데이터베이스 파티셔닝과 동기화 및 SDN 컨트롤러간의 부하분산 등의 추가적인 기술적인 이슈가 생기게 되었다. 이러한 이슈사항을 해결하기 위해, 몇몇의 연구가 진행되고 있는데, 특히, 오픈소스 기반의 커뮤니티들이 적극적으로 높은 가용성과 확장성을 갖는 SDN 제어평면을 개발하고 있다.

본 논문에서는 SDN 의 고 가용성 및 확장성 이슈와 관련하여 분석하고 관련 연구에 대해서 소개한다. 그 후에, ONOS 의 분산 데이터 베이스 파티셔닝/동기화 및 장애극복/부하분산 메커니즘을 알아 본다. 또한 ONOS 클러스터의 클러스터 크기에 따른 플로우 룰 설치 처리량 및 컨트롤러 장애 발생시 장애 극복 시간에 대한 실험 결과를 제시한다. 본 논문의 구성은 다음과 같다. 2 장에서는 앞서 말한 이슈와 관련하여 일반적인 접근 방법과 관련 연구를 요약하고, 3 장에서는 ONOS 가 이러한 이슈들에 대해서 어떻게 접근하고 있는지에 대해 분석하며 4 장에서 성능에 대한 실험 결과를 제시한다. 5 장에서는 본 논문의 결론을 맺는다.

2. SDN 에서의 고 가용성과 확장성 이슈

앞서 말했듯이, 높은 가용성과 확장성을 보장하는 SDN 제어평면을 구축하기 위해서는 여러 개의 컨트롤러들을 클러스터로 관리하는 것이 필수적이다. 그림 1 은 3 개의 SDN 컨트롤러들이 동기화되어 있는 일반적인 클러스터링 구조를 나타낸다. 이 구조에서는 forwarding service, topology service 등의 다양한 네트워크 서비스들이 여러 컨트롤러들에서 동작하게 되며, 이때, 서비스들의 네트워크 상태 정보는 네트워크 상태 데이터베이스에 저장되게 된다. 그림 1 을 보면, 높은 확장성을 얻기 위해 네트워크 상태 데이터베이스는 P1, P2, P3 의 파티션으로 나누어지게 되고, 네트워크의 상태 정보는 P1, P2, P3 파티션에 나뉘어 매핑된다. 또한, 높은 가용성을 얻기 위해서 각 파티션은 파티션을 복제하여 replica 집합을 2 개씩 유지하게 된다. 이때, replica 들은 3 개의 컨트롤러에 균일하게 분배되어 관리된다. 각

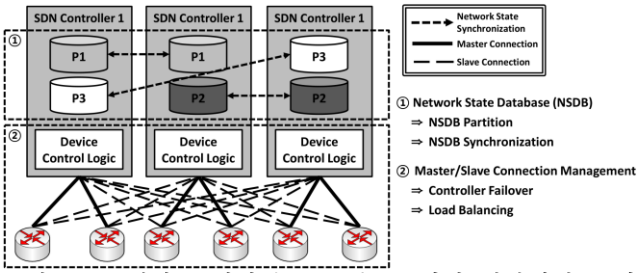


그림 1. 3 개의 동기화된 SDN 컨트롤러의 일반적인 클러스터링 구조

replica 들간의 동기화 또한 지원되기 때문에, 클러스터내의 컨트롤러들은 일관적인 네트워크 상태 정보를 제공해줄 수 있게 된다. 한편, 전달 평면의 스위치는 Master/Slave 형식으로 제어 평면과 다중 연결을 맺음으로써 Master 컨트롤러의 장애 발생시나 Master 연결을 재조정하여 부하 분산을 할때, Slave 상태의 컨트롤러와 새로운 Master 연결을 맺음으로써 Master 연결을 복구하게 된다.

이러한 환경에서 고 가용성과 확장성에 대한 기술적인 이슈는 다음과 같이 1) 어떤 방식으로 네트워크 상태 데이터베이스를 나누고, 나누어진 데이터베이스 파티션의 replica 들을 어떻게 분배 해야 할 것인가, 2)같은 네트워크 상태 정보를 저장하는 replica 들 간의 동기화를 어떻게 맞출 것인가, 3)컨트롤러 장애 발생시 어떻게 장애를 극복할것인가, 4)전달 평면의 네트워크 디바이스들에게 어떤 형식으로 Master/Slave 연결을 지정 할 것인가. 위 4 가지로 정의 될 수 있다

이와 관련하여 몇몇의 연구가 진행 되고 있다. Krishnamurthy *et al.*[2]는 애플리케이션의 상태 정보 파티셔닝과 네트워크 디바이스간의 상관관계에 대해 연구하였다. [2]에서는 IDS(침입 감지 시스템) 애플리케이션을 예로 상태 정보 파티셔닝 전략을 세웠으며, 컨트롤러간 통신 오버헤드를 최소화 하기 위한 최적의 SDN 스위치 할당과 상태정보 파티셔닝 방법을 제시하였다. Dixit *et al.*[3]은 네트워크 부하에 동적 적응능력 결핍을 초래하게 되는 컨트롤러와 네트워크 디바이스들 간의 고정적인 매핑 문제에 대해 소개하고, 이를 해결하기 위해 네트워크 부하 상태에 따라서 동적으로 클러스터의 크기를 조절하는 동적 클러스터링과 클러스터 크기 변경시 스위치의 Master 연결을 지속적으로 유지하도록 하기 위한 스위치 이동 프로토콜을 제안하였다.

다음 장에서는 잘 알려진 오픈소스 커뮤니티인 ONOS 에서 위의 4 가지 이슈에 대해서 어떤 방식으로 접근하는지에 대해 알아보도록 하겠다.

3. ONOS 에서의 고 가용성과 확장성

고 가용성과 확장성은 ONOS 에서 중요한 두가지 이슈이다. 이 장에서는 ONOS 클러스터링에 대해 간략하게 설명하고 앞에서 언급한 두가지 이슈를 ONOS 에서 어떻게 처리하는 지 살펴본다.

A. ONOS 클러스터링 개요

ONOS 는 Intent 서비스, Topology 서비스 등의 다양한 서비스를 제공하며 이러한 서비스들을 Subsystem 이라 한다. 다수의 ONOS 컨트롤러들은 클러스터를 형성하며 각 컨트롤러는 개별적으로 동일한 Subsystem 을 제공한다. 각 Subsystem 은 상태 정보를 각자의 Store 에 저장한다. 한편, Store 은 서로 다른 파티셔닝 및 동기화 기법을 갖는 ConsistentMap 과 EventuallyConsistencyMap 의 두가지 네트워크 상태 데이터베이스들 중 하나로 구현될 수 있다. 또한 ONOS 에서는 각 스위치가 다수의 ONOS 컨트롤러들에게 연결을 맺도록 허용하며, 부하 분산과 장애 극복을 위해 Master/Slave 연결 관리 기능을 제공한다.

컨트롤러들의 장애를 검출하기 위해, ONOS 에서는 ϕ-accrual failure detector[4] 기반의 장애 검출을 수행한다. 각 컨트롤러들은 자신의 상태 정보를 포함하는 heart-beat 메시지를 주기적으로 다른 컨트롤러들과 교환하며 이에 따라 각 컨트롤러마다의 ϕ 값을 계산한다. ϕ 는 $\phi = -\log10(1 - F(t))$로 계산되고, 이 때, F(t)는 heart-beat 메시지들의 수신 시간 간격 t 의 히스토리로부터 계산한 평균과 표준편차를 갖는 정규분포의 누적분포함수이다. 만약 특정 컨트롤러의 ϕ 값이 미리 정한 문턱 값 Φ보다 크다면 ONOS 클러스터는 해당 컨트롤러의 장애를 감지하게 된다.

B. 네트워크 상태 데이터베이스

EventuallyConsistentMap 기반 Store 는 모든 Map entry 를 하나의 파티션에 포함한다. 따라서 파티션의 총 개수는 Store 의 총 개수와 같다. EventuallyConsistentMap 기반 Store 들은 클러스터에 참여하는 모든 ONOS 컨트롤러에 복제되고, 따라서, EventuallyConsistentMap 기반의 Store 에 저장될 수 있는 네트워크 상태 정보의 양은 클러스터 내의 컨트롤러의 수에 비례하지 않는다. 또한 EventuallyConsistentMap 의 동기화는 Store 단위로 수행되므로, 오버헤드는 EventuallyConsistentMap 기반으로 구현된 Store 수에 비례한다.

ConsistencyMap 기반 Store 의 각 Map entry 는 N 개의 파티션 중 하나에 매핑될 수 있고, 매핑은 해쉬 범위 [1,N]에서 Map entry 의 해쉬 키값에 의해 결정된다. 여기서 N 은 클러스터 내의 컨트롤러의 수이다. ConsistentMap 의 각 파티션은 R 개의 replica 를 유지하고, R 은 컨트롤러 관리자가 결정할 수 있다. 이때, 각 파티션의 replica 는 가장 적은 수의 replica 를 가진 컨트롤러에 할당된다. 그렇기 때문에, R 이 클러스터에 참여하는 컨트롤러 수 보다 적게 설정되면, ConsistentMap 에 의해서 처리될수 있는 네트워크 상태정보의 양은 클러스터의 컨트롤러 수가 늘어남에 따라 증가하게 된다. 한편, ConsistentMap 의 동기화는 partition 단위로 수행되기 때문에 그 오버헤드는 N 에 비례한다.

C. 네트워크 상태 데이터베이스 동기화

EventuallyConsistentMap 파티션의 replica 들은 anti-

entropy 프로토콜을 기반으로 동기화되며, 우수한 읽기/쓰기 성능을 제공하는 반면, weak consistency 만을 제공한다. EventuallyConsistentMap 의 파티션의 모든 replica 는 primary replica 이다. 따라서 모든 읽기 동작은 로컬 primary replica 에서 시행되는 반면 쓰기 동작은 로컬 primary replica 에서 시행이 되고난 후 다른 replica 들로 전달된다.

replica 들의 의견수렴을 위해, EventuallyConsistentMap 의 업데이트 이벤트를 받자마자 각 replica 들은 업데이트 이벤트에 대한 논리적인 timestamp 를 정한다. 그 후, 업데이트 이벤트는 EventuallyConsistentMap 엔트리로 commit 되고 동시에 자신을 제외한 다른 replica 들로 timestamp 와 함께 브로드캐스트 된다. 브로드캐스트된 업데이트 이벤트를 받자마자 이벤트를 받은 replica 들은 받은 이벤트가 최신인지 확인하고, 만약 받은 이벤트의 timestamp 가 더 이전의 것이라면 그 이벤트는 처리하지 않는다. 반면 받은 이벤트의 timestamp 가 최신의 것이라면 EventuallyConsistentMap 엔트리로 commit 된다. 이렇게 함으로써, 모든 replica 의 시스템 상태 정보는 알맞은 상태 정보로 수렴하게 된다.

또한, 새로 추가된 컨트롤러나 재시작된 컨트롤러의 replica 를 정해진 시간 안에 지체 없이 동기화하기 위해서 각 replica 들은 자신을 제외한 다른 replica 중 하나를 무작위로 골라 상태정보 동기화를 수행하고, 만약 두 replica 의 timestamp 가 다르다면 최신의 EventuallyConsistentMap 엔트리로 업데이트 한다.

ConsistentMap 의 파티션의 replica 들은 strong consistency 를 제공하는 Raft 프로토콜[5]을 기반으로 동기화되며, 상대적으로 낮은 읽기/쓰기 성능을 제공하는 반면, strong consistency 만을 제공한다. Raft 프로토콜에서 읽기/쓰기 동작은 프로토콜을 통해 선출된 primary replica 에서만 허용된다. 또한 쓰기 요청을 commit 하기 위해는 과반수의 backup replica 들의 동의가 필요하다. ONOS 에서는 primary replica 와 backup replica 를 각각 leader replica, follower replica 라 한다.

반면, 한 파티션의 leader replica 를 갖고 있는 ONOS 컨트롤러에 고장이 발생하면, 그 파티션 leader 의 부재로 인해 데이터 접근이 금지된다. leader replica 가 동작 중인지 파악할 수 있도록 leader replica 는 주기적으로 Raft heart-beat 메시지를 follower replica 에게 보낸다. 만약, 사전에 정의된 선출 시간동안 follower replica 중 하나가 leader replica 로부터 어떠한 응답도 받지 못하면, leader replica 에 장애가 발생했다 인지하고, 새로운 leader 선출을 요청한다. 이 때, 가장 많은 표를 받은 replica 가 새로운 leader replica 로 선출된다.

그림 2 는 ONOS 에서의 토폴로지 상태 동기화의 예를 보여준다. ONOS 에서는 동기화에 anti-entropy 프로토콜을 사용하는 Topology Store 에 토폴로지 상태가 저장된다. 그림 2 에서 각 ONOS 컨트롤러는

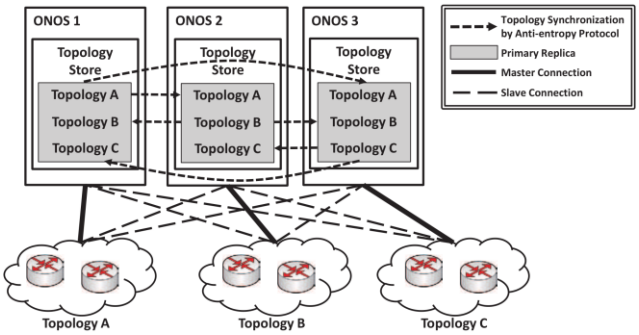


그림 2. ONOS 의 토폴로지 상태 동기화 예

전체 토폴로지의 일부를 관리하고, 토폴로지 상태에 대한 읽기/쓰기 요청은 로컬 replica 에서 처리된다. 예를 들어, ONOS 1 컨트롤러가 Topology A, Topology B 또는 Topology C 에 대한 토폴로지 상태 읽기 요청을 받으면 컨트롤러는 요청을 로컬 replica 에서 처리한다. 또한, ONOS 1 이 Topology A 로부터 토폴로지 업데이트 이벤트를 받으면, 컨트롤러는 업데이트를 로컬 replica 로 commit 하고, 클러스터에 참여하는 다른 replica 들과 업데이트를 동기화한다. 하지만 ONOS 에서는 토폴로지 상태 replica 들 간의 불일치가 발생할 수 있다. 예를 들면, commit 과 동기화를 진행하는 동안 동기화되지 않은 replica 가 읽기 요청을 처리할 수 있고, 읽기 요청은 동기화 이전의 토폴로지 상태를 읽게 된다.

D. 컨트롤러 장애극복

ONOS 컨트롤러의 장애 발생시 클러스터내의 다른 ONOS 컨트롤러들은 ϕ failure detector[4]를 통해 해당 컨트롤러의 장애를 탐지하고 해당 컨트롤러가 Master 로써 관리하던 스위치들에게 새로운 Master 컨트롤러를 할당하게 된다. 새 Master 컨트롤러로 결정된 컨트롤러는 해당 스위치에 ROLE_REQUEST 를 보내 자신이 새로운 Master 컨트롤러임을 알리게 되고, 이에 대한 응답으로 스위치는 ROLE_REPLY 메시지를 새로운 Master 컨트롤러에 전송한다. 결과적으로, 장애가 발생한 컨트롤러가 Master 컨트롤러로 설정되어있던 모든 스위치들은 새로운 Master 컨트롤러들을 할당받게 된다.

E. 부하 분산

새로운 스위치가 다중 ONOS 컨트롤러에 연결될 때, 가장 적은 Master 연결을 가진 컨트롤러가 해당 스위치의 Master 컨트롤러가 되고, 각 컨트롤러가 제어하는 스위치의 수가 균형을 이루게 된다.

4. 실험 결과

이번 장에서는 플로우 룰 설치 처리량과 컨트롤러 장애극복에 관한 실험결과를 살펴본다.

두 개의 실험을 위해서 각 컨트롤러를 6GB RAM 과 2 CPU core 의 가상머신에 ONOS-1.4 (Emu) distribution 버전으로 동작시킨 후 컨트롤러들이 클러스터링 되도록 설정하였다

플로우 룰 설치 처리량을 측정하기 위하여 C 개의 컨트롤러들을 동작시키고 하나의 컨트롤러를

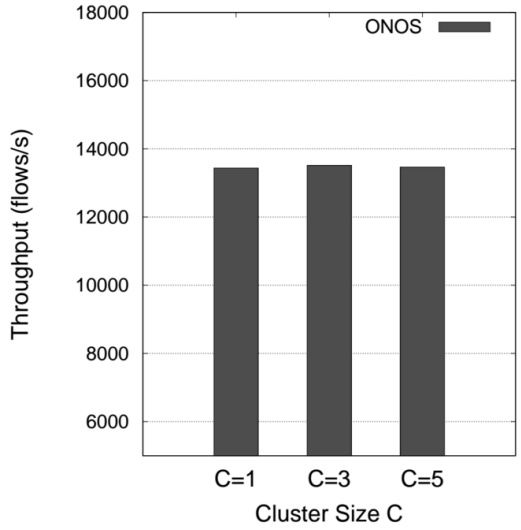


그림 3. 클러스터 크기에 따른 플로우 룰 설치 처리량 측정

Master 컨트롤러로 지정하여 9 개의 스위치를 연결하였고, Master 컨트롤러에서 플로우 룰 설치 어플리케이션을 설치하여 각 스위치마다 500 개의 플로우 룰을 설치하도록 하며 처리량을 측정하였다. 또한, 플로우 룰 상태를 갖은 파티션은 모든 컨트롤러에 완전히 복제된 상태에서 실험을 진행하였다. 플로우 룰 설치 요청에는 ONOS 에 내장되어 있는 플로우 룰 설치 요청 도구를 사용하였다. 플로우 룰 설치 요청 도구는 Master 컨트롤러에서 동작하는 플로우 처리량 테스트 어플리케이션에 플로우 룰 설치를 요청한다. 그 후, 테스트 어플리케이션은 플로우 룰을 무작위로 생성하여 로컬 FlowRule Store 에 쓰고, ONOS 의 OpenFlow 컨트롤러 서비스는 위 플로우 룰을 OpenFlow 스위치에 설치한다. 한편 ONOS 에서 플로우 룰은 FlowRule Store 에 저장되며 이는 EventuallyConsistentMap 을 기반으로 구현되었다.

그림 3 은 클러스터에 참여하는 컨트롤러 개수 C 에 따른 플로우 룰 설치 처리량을 보여준다. 그림 3 에서 볼 수 있듯이 ONOS 의 FlowRule Store 는 EventuallyConsistentMap 을 사용하기 때문에, 클러스터의 크기와 상관없이 일정한 처리량을 보이는 것을 확인할 수 있다.

컨트롤러 장애극복 실험에서는 3 개의 ONOS 컨트롤러를 사용하였고, 그 중 하나를 Master 컨트롤러로 지정하였다. 그 후, Master 컨트롤러에 D 개의 스위치를 할당한 뒤 실험하였다. 먼저, 해당 Master 컨트롤러에 장애를 발생시킨 뒤 해당 컨트롤러에 연결되어 있던 D 개의 스위치가 모두 새로운 Master 연결을 맺을 때까지의 시간을 측정하여 장애극복 시간을 파악하였다. 이때, **Φ**는 8.0 으로 설정하고 실험하였다. 그림 4 에서 볼 수 있듯이, 스위치 개수가 늘어남에 따라서 장애극복시간이 증가하는 것을 알 수 있다.

5. 결론

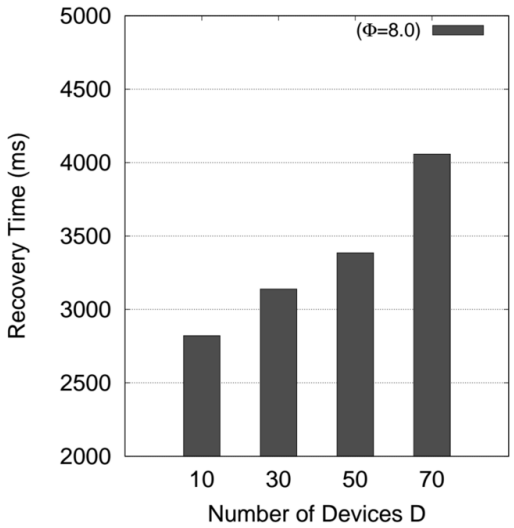


그림 4 스위치 개수에 따른 컨트롤러 장애극복 시간

본 논문에서는 높은 가용성과 확장성을 보장하기 위한 일반적인 SDN 제어평면 구조에 대해서 알아보았다. 또한, ONOS 의 고 가용성 및 확장성 이슈 들에 대한 접근 방법을 분석하였다. 실험 결과를 통해서 1) ONOS 에서의 플로우 룰 설치 처리량은 클러스터 크기에 영향을 받지 않는 것을 알 수 있고, 2) 컨트롤러의 장애극복 시간은 스위치의 개수에 비례한다는 것을 알 수 있었다. 향후에는 SDN 어플리케이션 별로 네트워크 상태 동기화 방식에 대한 성능 분석에 대한 연구를 진행할 예정이다.

ACKNOWLEDGMENT

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개 소프트웨어 핵심 모듈/기능 개발)

5. 참고문헌

[1] ONOS controller, http://onosproject.org

[2] A. Krishnamurthy, S. Chandrabose, and A. Gember-Jacobson, “Pratyaastha: An Efficient Elastic Distributed SDN Control Plane,” in *Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN) 2014*, Chicago, IL, August 2014.

[3] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman and R. Kompella, “Towards an Elastic Distributed SDN Controller,” in *Proc. ACM Workshop on Hot Topics in Software-Defined Networking (HotSDN) 2013*, Hongkong, August 2013.

[4] N. Hayashibara, X. Defago, R. Yared, and T. Katayama, “The ϕ accrual failure detector,” in *Proc. IEEE SRDS 2004*, Florianpolis, Brazil, October 2004.

[5] D. Ongaro and J. Ousterhout, “In Search of an Understandable Consensus Algorithm,” in *Proc. USENIX ATC 2014*, Philadelphia, USA, June 2014.

오픈소스하드웨어를 이용한 농업ICT 플랫폼

조원익0, 강석원, 김혜원, 송왕철
제주대학교 전기전자통신컴퓨터공학부
{twer4774, dudtntdud, coope0357, kingiron}@gmail.com

요약

본 논문은 Raspberrypi와 Arduino 시스템을 기반으로 과실수를 키우기 위한 시스템을 개발한 것으로, 인터넷 상의 Web 서비스와 연결 된 오픈소스하드웨어를 통해 원격으로 농업기계를 제어하고, 오픈소스하드웨어에 부착된 센서를 이용하여 데이터 값을 받아들여 저장된 데이터를 이용하여 센서 및 장치들을 제어하는 시스템이다. 센서 데이터 값에 따라 농업기계가 자동으로 통제되며 Web Browser상에서 실시간으로 데이터 값의 변화를 차트로 확인 가능하다. 나무의 생장과정을 확인할 수 있도록 Web에서 제어 가능한 카메라를 통해 화상을 통한 모니터링이 가능하도록 하였다. 통합된 시스템이 확립되면 누적된 데이터의 분석을 통하여 다양한 농업분야에서 활용될 수 있을 것으로 기대된다.

1. 서론

정보화 사회로의 산업구조의 변화로 1차 산업인 농업의 성장세가 둔화되고 있다. 이를 극복하기 위해서는 정보화 시대에 발맞추어 농업 또한 스마트하게 발전되어야 한다. 최근 각광받고 있는 IoT 기술을 활용한 농업ICT 분야가 농업분야에서 화두가 되고 있다. 창조농업 실현을 위한 정부의 지원과 산업통상자원부를 비롯한 기관에서 농업ICT를 지원하고 있으며 이 같은 지원 사업으로 1차 산업인 농업이 IoT를 만나 더욱 발전한 고차원 산업으로 발전할 수 있을 것이다.

본 논문은 농업ICT의 기초 플랫폼에 대한 것으로 과실수를 키우기 위한 플랫폼으로 초점을 맞추었다. 농업ICT를 위해서는 식물의 생장에 대한 표본데이터가 필요하며 데이터에 맞추어 생장환경을 조절할 수 있어야 한다. 따라서, 이 플랫폼은 IoT기술을 이용해 식물이 생장하는 동안 생장환경의 온·습도, 조도, 바람, 토양습도를 데이터베이스에 저장시켜 표본데이터를 추출하고, 표본데이터 값에 따라 사물간의 통신을 통해 스프링클러, 영양제, 전구를 조절하여 식물의 생장환경을 최적으로 만드는데 목표가 있다.

그림1에서와 같이 IoT기술을 손쉽게 사용하기 위하여 오픈소스하드웨어인 Raspberrypi와 Arduino보드를 이용하였으며, 사람의 손을 거치지 않고 사물간의 통신으로 생장환경을 자동으로 조절 할 수 있

(그림 1) 플랫폼 구성도

음은 물론, 필요에 따라 서버 컴퓨터에 설치된 Web 서버와 연결된 Web Browser 서비스를 이용해 원격

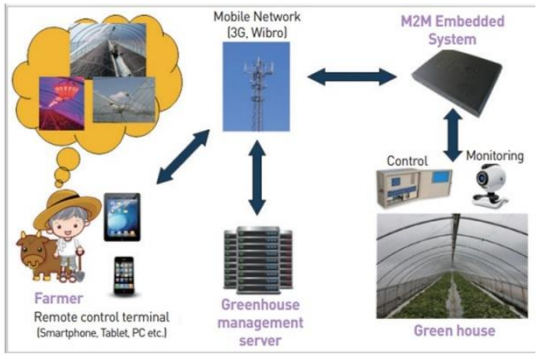
으로 오픈소스하드웨어를 제어할 수 있다. Arduino 보드와 연결된 모터와 전구를 제어하여 식물의 생장환경을 임의적으로 조절할 수 있으며, Arduino보드에 부착된 센서들을 통해 센서 데이터 값을 받아들여 Web Browser상에 차트로 표현해준다. 식물의 생장을 눈으로 확인하기 위하여 Arduino보드에 카메라를 부착하여 실시간으로 영상을 볼 수 있도록 한다.

본 논문은 2장에서 관련 연구를 서술하고 현재 서비스 중인 사례를 소개한다. 3장에서는 플랫폼 설계에 대해 설명하였고, 4장에서는 실제 플랫폼 구현에 대한 부분을 기술하고 5장을 마지막으로 결론을 맺는다.

2. 관련연구

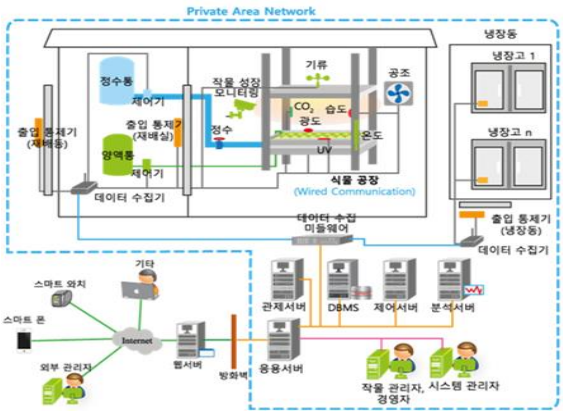
2.1 서비스 사례

농업ICT 플랫폼의 서비스를 몇 가지 찾을 수 있어서 본 논문을 작성하는데 많은 도움이 되어 소개하고자 한다. 첫 번째로 SKT에서 서비스하고 있는 SmartFarm 서비스이다.[1] SmartFarm의 서비스는 비닐하우스에 대한 서비스를 실시하고 있으며 그림 2와 같은 플랫폼 구성도를 가지고 있다. SmartFarm의 기능으로는 장비들의 작동여부를 확인하는 역할의 CCTV와 하우스 내 온·습도를 실시간으로 점검하는 기능, 스프링클러, 분무, 급 배수 기능, 농약과 비료를 공급하는 기능이 구현되어 있다.

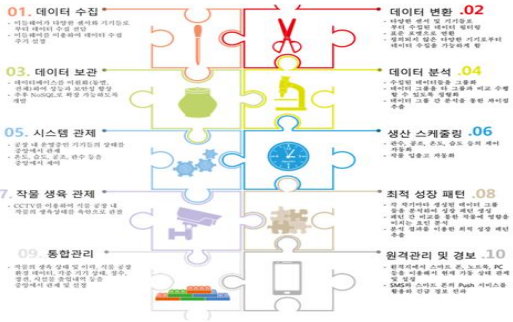


(그림 2) SKT의 스마트팜 플랫폼 구성도[1]

두 번째 사례로, Agronics에서 서비스하고 있는 스마트팜팩토리사업이다.[2] 그림3은 플랫폼 구성도이며 그림4은 플랫폼 서비스로 인해 얻을 수 있는 솔루션기능이다.

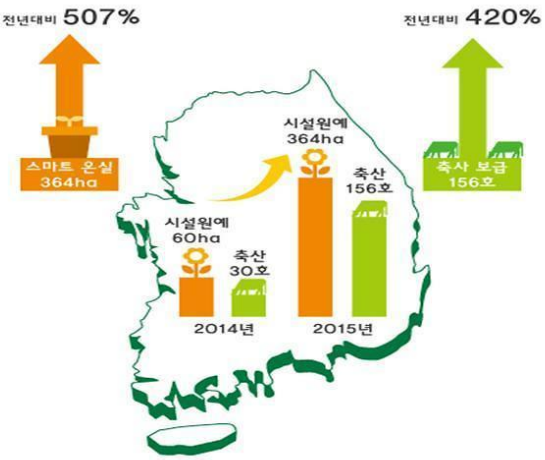


(그림 3) 플랫폼 구성도[2]



(그림 4) 솔루션 기능

스마트팜 도입으로 농가 생산량의 25% 수입은 31% 증가하였으며[3], 정부와 이동통신사 등 많은 기관에서 스마트팜을 지원하고 있다.



(그림 5) 스마트팜 도입효과

2.2 오픈소스하드웨어(Open Source Hardware)

오픈소스하드웨어란 해당 제품과 똑같은 모양 및 기능을 가진 제품을 만드는 데 필요한 모든 것(회

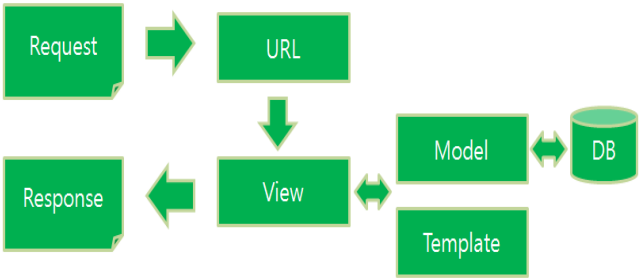
로도, 자재 명세서, 인쇄 회로 기판 도면 등)을 대중에게 공개한 전자제품[4]을 말하며 IoT를 쉽게 농업ICT분야에 접목 시킬 수 있는 방안으로 오픈소스하드웨어를 택하게 되었다. 오픈소스하드웨어를 활용하여 ICT를 구현하기 위해서 Web서버와 I/O Adapter를 연결하는 서비스를 활용하는 설계 사례를 찾아 볼 수 있었다.[5]

본 논문에서는 Gateway 역할을 하는 Raspberrypi는 서버 컴퓨터에 구현된 Django를 통해 URL이 변하면 Arduino보드를 통제해주는 역할을 하고 있으며, 또한 Arduino보드 센서에서 값을 받아 들여 서버 컴퓨터의 데이터베이스에 센서 데이터 값을 저장시키는 역할을 담당하고 있다.

2.2.1 Django

Django는 Python언어로 작성된 Web Framework로 HttpResponse로 동작하여 빠르고 깔끔하게 Web 사이트를 개발할 수 있으며, 오픈소스이다. Python 언어의 특징을 닮아 다른 언어로 구성된 Framewrok을 기반으로 응용개발이 용이한 장점을 가지고 있다.

그림6은 Django의 구조를 보여준다. Django는 MVC 패턴과 유사한 MVT 패턴을 이용하여 웹 사이트를 구성하며 M은 Model을 말하며, 데이터베이스에 저장되는 데이터의 양식을 설정하고 데이터베이스와 실제적으로 연동되는 부분이다. V는 View를 말하며, MVC패턴에서 C(Controler)와 유사한 기능으로 URL에 따라 Model과 Template을 제어한다. T는 Template으로 실제로 보여지는 View의 역할을 한다.

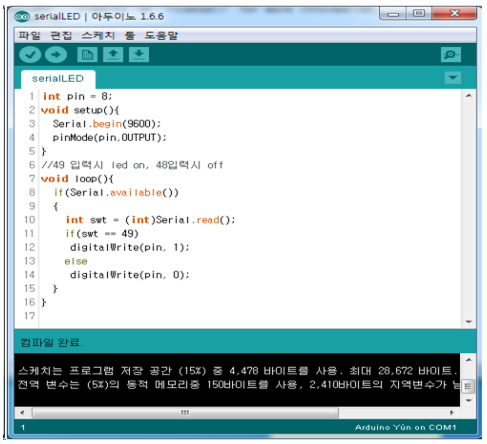


(그림 6) Django 구조

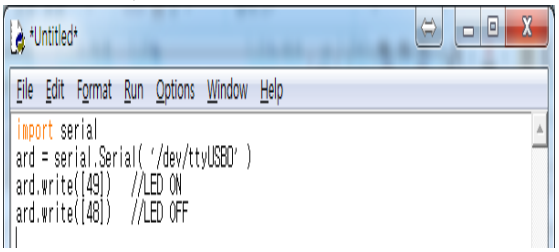
본 논문에서는 Django를 서버 컴퓨터에 설치하여 Web서버를 구축하였으며, Web Browser를 통해 제어된 신호가 Raspberrypi를 거쳐 Arduino보드로 전달되어 장치 제어가 가능하다. Django를 이용해 Arduino보드를 제어하는 사례가 있어 참고하였다.[6]

2.2.2 PySerial

Arduino보드와 Raspberrypi의 빠른 통신을 위하여 Serial 통신을 택하였고, 그에 따라 Raspberrypi에 PySerial을 설치하였다. 그림7은 Arduino보드에 업로드 된 코드로 특정 값이 입력되면 LED ON/FF기능을 하며, 그림8은 Serial통신을 이용하여 LED를 제어하는 Python 코드이다.



(그림 7) Arduino 업로드 코드



(그림 8)
Serial통신을 이용한 LED제어 Python코드

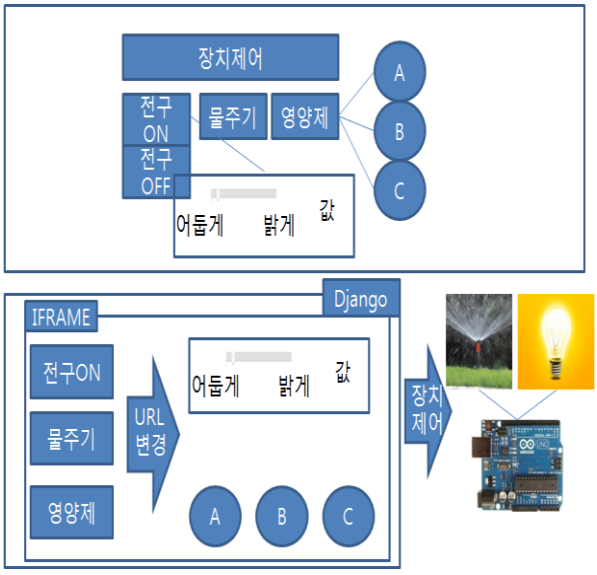
3. 설계

본 논문에서는 그림1과 같이 Arduino보드에 부착된 센서들의 값을 서버 컴퓨터의 데이터베이스에 저장시키고, 서버 컴퓨터는 Django를 이용하여 그림9와 같은 웹 페이지를 구현한다.

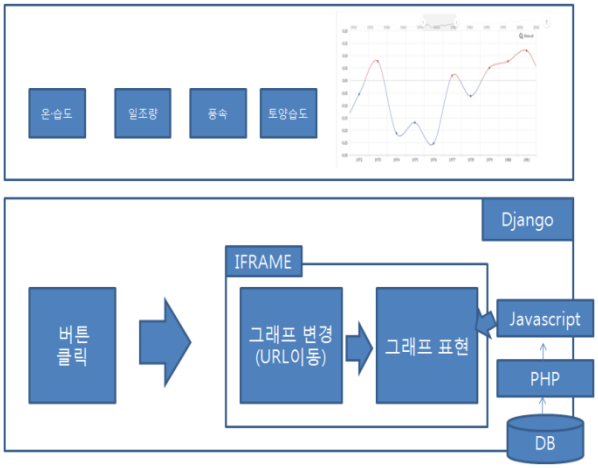
(그림 9) 웹페이지 구성도

웹 페이지의 기능은 크게 카메라제어, 장치제어, 센서 값에 따른 차트 확인으로 나눌 수 있다. 카메라는 Arduino보드에 장착되어 있어 버튼 클릭 시 제어한다. 장치제어의 경우 버튼 클릭 시 Django에서 구현된 페이지의 URL이 변경되며, 변경된 URL이 Gateway 역할을 하는 Raspberrypi를 거치면서 제어신호를 Arduino보드로 전송하여 장치를 제어하도록 한다. 차트 확인의 경우 온·습도, 일조량, 풍속, 토양습도 버튼 클릭 시 장치제어와 마찬가지로 Django로 구현된 페이지의 URL이 바뀌면서 php로 작성된 MySQL쿼리문으로 데이터베이스에서 수치를 가지고 와 JavaScrip언어로 구현한 AMChart를 보여준다.

아래 그림10과 그림11은 각각 장치제어와 차트 표현에 대한 실행 모식도이다.



(그림 10) 장치제어 실행 모식도



(그림 11) 차트표현 실행 모식도

4. 구현

4.1 Arduino보드

Arduino보드는 센서와 장치를 연결하여 센서의 데이터 값을 읽어 들여 Raspberrypi로 전송하고, 센서 데이터 값에 따라 장치를 자동으로 제어하며 사용자의 인위적인 요청으로 장치를 제어할 수 있는 코드 또한 함께 업로드되어 있다.

(그림 12) Arduino

(그림 13) Arduino 부착 센서
조도센서, 온·습도센서, 토양습도센서, 바람센서

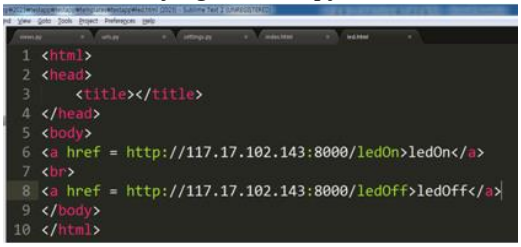
(그림 14) Arduino 부착 장치 -전구, DC모터

그림13,14는 Arudino보드에 부착된 센서와 장치들을 보여주며, 그림15,16,17은 LED전구를 Django를 이용하여 제어하기 위한 코드이다. 그림17의 Template이 Django에 표현되면 ledOn과 ledOff라는 URL이동 문자가 생기고 클릭시 URL이 변경되면서 Urls.py에서 URL을 매핑하고, URL에 맞는 Views.py은 신호를 Arduino에 전송하여 장치를 제어한다.

```
from django.http import HttpResponse
from django.template.loader import get_template
from django.shortcuts import render
import serial

ard = serial.Serial('COM12')

def hello(request):
    return HttpResponse("Hello")
def ledOn(request):
    on="<html><body>ON</body></html>"
    return HttpResponse(on, ard.write([49]))
def ledOff(request):
    off="<html><body>OFF</body></html>"
    return HttpResponse(off, ard.write([48]))
def index(request):
    return render(request, 'testapp/led.html')
```

```
from django.conf.urls import url, include
from testDjango.views import index, ledOn, ledOff

urlpatterns = [
    url(r'^$', index),
    url(r'^ledOn/$', ledOn),
    url(r'^ledOff/$', ledOff),
]
```

(그림 15) (좌) Django의 Views.py
(그림 16) (우) Django의 Urls.py



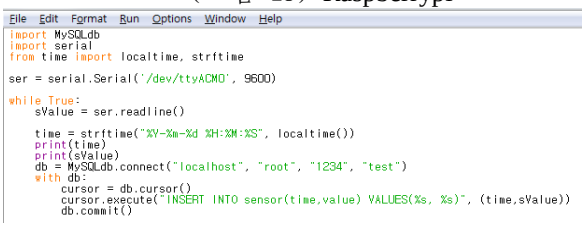
```
<html>
<head>
    <title></title>
</head>
<body>
<a href = http://117.17.102.143:8000/ledOn>ledOn</a>
<br>
<a href = http://117.17.102.143:8000/ledOff>ledOff</a>
</body>
</html>
```

(그림 17) Django의 Template -led.html

4.2 Raspberrypi

Raspberrypi는 Arduino보드와 서버 컴퓨터의 중간에서 Gateway역할을 담당하며 Arduino의 센서 데이터 값을 Serial통신으로 받아들여 Python코드로 작성된 MySQL문으로 서버 컴퓨터의 데이터베이스 센서 데이터 값을 저장하는 역할을 한다.

(그림 18) Raspberrypi

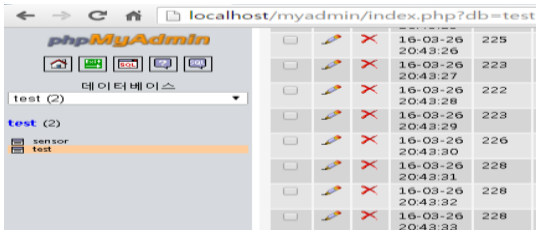


```
import MySQLdb
import serial
from time import localtime, strftime

ser = serial.Serial('/dev/ttyACM0', 9600)

while True:
    sValue = ser.readline()

    time = strftime("%Y-%m-%d %H:%M:%S", localtime())
    print(time)
    print(sValue)
    db = MySQLdb.connect("localhost", "root", "1234", "test")
    with db:
        cursor = db.cursor()
        cursor.execute("INSERT INTO sensor(time,value) VALUES(%s, %s)", (time,sValue))
        db.commit()
```

(그림 19)Python코드로 작성된 MySQL, Serial통신

4.3 서버 컴퓨터

서버 컴퓨터는 실행속도가 비교적 느리고 저장 공간에 제약이 있는 Rasberrypi의 단점 최대한 보완하고자 데이터베이스를 서버 컴퓨터에 생성하고 Django를 서버 컴퓨터에서 실행함으로써 플랫폼 전체의 실행속도와 성능 향상을 가져왔다. 서버 컴퓨터에서 Django로 구현한 웹 페이지를 통해 오픈소스하드웨어를 제어하고, 실시간으로 센서 데이터 값의 변화를 차트로 갱신하는 역할을 맡고 있다.



(그림 20) 서버 컴퓨터에 센서데이터 값이 저장된 모습

테이블명	Sensor					
테이블설명	Sensor 데이터 값의 저장					
번호	컬럼명	데이터타입	길이	NULL여부	기본값	KEY
1	Num	INT	11	NOT		PK
2	Time	VARCHAR	20	NOT		
3	Illum	INT	11			
4	wind	INT	11			
5	Temp_Hum	INT	11			
6	Soil_Hum	INT	11			

(표 1) Sensor 테이블의 데이터구조

표1에서 보는바와 같이 테이블의 구조가 정의되어 있으며 Autoincrement 하는 Num을 기본키로 두어 데이터들을 구분할 수 있도록 설계했다.

5. 결론

본 논문은 농업ICT를 구현하기 위한 기초 플랫폼에 대한 것으로, 사람의 관여 없이 식물의 생장환경을 최적화 시켜준다. 센서 데이터 값에 따라 장치들을 제어하여 조도를 맞추고 물을 주며, 영양제를 투입해주는 기능을 한다. 센서 데이터 값을 데이터베이스에 저장함으로써 식물 생장의 표본데이터 값을 추출할 수 있고, 이를 토대로 다양한 분야의 농업ICT에 적용할 수 있다. 식물의 생장 과정을 농업지역에 설치된 카메라를 통해 확인할 수 신뢰도가 높고, 센서 데이터 값을 차트로 표현하여 사용자의 편의를 제공하였다. 또한 사용자의 필요에 따라 Web Browser를 통해 장치를 제어하여 식물의 생장과정에 관여할 수 있다.

농업 지역에 직접 방문하지 않더라도 원격으로, 자동으로 식물을 기를 수 있어 농업인의 편의뿐만 아니라 지역적인 제한으로 식물을 기르는 일을 하지 못했던 이들에게 서비스를 유용하게 제공할 것으로 보여진다.

사사

이 논문은 제주대학교 스마트그리드와 청정에너지 융복합산업 인력양성사업단(CK-I)의 지원을 받아 수행되었습니다.

참고문헌

[1] SKT의 스마트팜, 비닐하우스 관리 시스템
http://blog.naver.com/jellyinc/220037307865
[2] ICT 스마트팜팩토리 통합 운영 플랫폼
http://agronics.co.kr/wp/연구개발/개발/ict기반-식물공장-통합-운영-플랫폼-개발/
[3] 스마트팜 도입효과
http://www.ajunews.com/view/20160428074017061
[4] 오픈소스하드웨어
https://ko.wikipedia.org/wiki/오픈_소스_하드웨어
[5] Smart_IoT 설계 시스템
http://www.huins.com/new/sub/goods_view.php?it_id=1429161608&ca_id=40&ca_id2=40&n=1&sn=1
[6] Arduino with Django and Python
https://alejandrok5.wordpress.com/2011/06/27/arduino-with-django-and-python/

Secured Network Architecture and Solution for IoT Services

김우태 O, 이세희, 이영우

KT 융합기술원 Infra 연구소

{utae.kim, sehuilee, young-woo.lee}@kt.com

요 약

최신 IT 기술과의 접목을 통해 다양한 산업분야에서 각양각색의 사물 인터넷(Internet of Things) 제품 및 서비스 확대와 함께 모든 사물이 네트워크로 연결되는 사물 인터넷 시대가 도래 되었다. 그렇지만, 사물 인터넷이 활용되는 분야는 우리 실생활의 모든 사물에 '직접 접목' 되는 특성을 갖기 때문에, 기존 사이버 공간의 위험이 현실 세계로 전이·확대될 수 있으며, 특히 보안위협은 오동작·정지 등 사람의 생명을 위태롭게 할 수 있으므로, 보안의 중요성이 점점 증대되고 있는 추세이다.[1][2] 그리고, 홈 IoT 분야에서는 개인의 사생활과 직접 연관되는 관계로 보안의 중요성이 더욱 강조되고 있다. 본 논문에서는 사물인터넷 서비스의 보안성 확보를 위한, Secured Network Architecture 및 서비스 제공방안을 제안한다. 본 논문에서 제안한 방식은, 사물 인터넷 단말에 사설 IP 를 제공하고, 인터넷 사업자의 에지 노드의 VR 및 터널링 기능을 이용하여 SDR(Service Distribution Router)로 모든 Secured 트래픽이 전달되도록 하고, SDR 에 적용된 제어 정책을 통한 양 종단간 통신으로 마치 전용회선 수준의 Secured Connectivity 를 제공하는 방법이다.

1. 사물인터넷(Internet of Things)

지금까지의 인터넷은 사용자가 컴퓨터 또는 스마트폰을 통해 인터넷에 접속하며 정보를 제공받고 서비스를 이용하고 있었다. 하지만, 사물인터넷은 세상의 모든 사물이 인터넷 네트워크에 연결되어, 사람과 사물, 사물과 사물간 통신을 통해 새로운 정보나 서비스를 이용할 수 있게 되었다.

하지만, 개방형 모델에서 출발한 인터넷 구조로, 인터넷 네트워크에 연결되어 서비스가 제공되는 사물인터넷 역시 개방형 모델에 노출되어 모든 사물들이 해킹 대상이 될 수 있으며, 이에 사물인터넷 보안 위협이 대두가 되고 있다. 그리고, 최근 전세계 CCTV 73,000 여개가 해킹되어 인터넷에 노출이 되었으며, 우리나라도 약 6,000 여개의 CCTV 가 해킹 사고가 발생되어 이슈가 되었다. 특히, 홈 CCTV 와 같은 IoT 단말이 해킹이 되는 경우, 개인의 사생활이 전세계에 노출이 될 수 있어 보안 이슈해소가 필수적이다.

이러한 사물인터넷 보안이슈를 해소하기 위해 보안모듈 탑재 등의 표준작업이 활발히 진행 중이지만, 가장 기본적으로 네트워크 계층에서의 보안성 제공을 통해 원천적인 이슈해소를 하는 것이 필수적이다. 본 논문에서는, 이러한 사물인터넷 보안성 확보를 통한 이슈 해소를 위해, 네트워크 계층에서의 보안성 제공 구조에 대해 제안하고자 한다.

2. 인터넷 구조

인터넷은 전세계에 위치한 각 단말에 대하여, IP 주소를 할당하고, 라우팅을 통해 통신이 가능하도록 되어 있다. 이 때 사용하는 IP 주소는 누구에게나 공개되어 있는 공인 IP 주소로서, 어느 단말이든지 인터넷을 이용하기 위해서는 공인 IP 주소를 할당 받아야 한다.[3] 그러나, 전세계에서 사용하는 공인 IP 주소는 IPv4 주소체계를 준수해야 하며, 전세계 단말들이 인터넷을 이용하기 위하여 주소를 할당 받기에는 공인 IP 주소의 부족이 예상되고, 기업이나 개인적으로 인터넷상에 단말의 직접적으로 공개되기를 원하지 않고 특히 방화벽 등을 통해 자신의 단말을 보호하기 위한 목적으로 사설 IP 주소 대역이 정해져 있으며, 이를 통해 복수의 단말들이 하나의 공인 IP 주소를 가지고 사설 IP 주소를 할당 받은 여러 대의 단말이 인터넷을 이용할 수 있게 하였다. 사설 IP 주소는 공인 IP 주소와는 별도의 IP 주소 블록으로 IETF 에서 할당하였다.[4]

일반적으로 인터넷사업자는 인터넷을 이용하도록 하기 위해 공인 IP 주소를 할당하지만, 사설 IP 주소는 사용자가 복수 단말을 이용 또는 특정 목적에 맞도록 정책적으로 이용하기 위한 방법이다. 이러한 목적으로 많이 활성화 되어 있는 장치가 공유기 이다. 공유기의 동작은 인터넷 사업자로부터 공인 IP 주소를 할당 받고, 공유기에 연결된 여러 단말들은 사설 IP 주소를 할당 받아 인터넷을 사용하도록 하는 방법이다.[5][6]

사물인터넷 서비스도 상기와 같은 인터넷 구조를 이용하여 서비스가 제공되어야 하며, 인터넷의 개방형 모델에서는 항상 보안에 노출이 될 수 있는 문제점이 있다.

본 논문에서는, 사물인터넷이 범용적인 인터넷 네트워크 접속과 달리 특수한 목적을 위한 서버 또는 사업자와의 접속을 한다는 점에 착안하여, 네트워크 계층에서 사설 IP 네트워킹 기술을 이용하여 원천적인 보안을 제공할 수 있는 방안에 대해 제안하고자 한다.

3. Secured Network Architecture

3.1 네트워크 서비스 모델

사물인터넷 서비스 보안 이슈의 원천적 해소를 위해, 인터넷의 기본 구조인 공인 IP 주소를 이용한 개방형 모델의 개념에서 벗어나, 사설 IP 주소를 이용한 End-to-End Connectivity 를 통해, Secured Network Architecture 를 구성하는 것이다. 네트워크 기반 사설 IP 주소의 적용모델은 그림 1 에서 보는 바와 같이, 일반 인터넷 제공 모델과의 논리적 구분을 위해 인터넷 사업자의 에지 노드에서 VR(Virtual Router, 이하 VR) 기능을 이용하는 방법이다. VR 은 하나의 물리적인 라우터 내에 논리적인 라우터로 구분하여 각각이 별도의 라우팅 모드로 동작하도록 하는 기술을 의미한다. 즉, 라우터 내 일반 인터넷 라우팅을 위한 공인 VR 과 사물인터넷 Secured 라우팅을 위한 사설 VR 로 분리하여 관리하고, 인터넷 회선에서는 논리적으로 구분하여 공인 및 사설 IP 주소를 단말 별로 할당하도록 한다. 이후, 공인 IP 주소를 할당 받은 단말은 공인 VR 을 통해 일반 인터넷 서비스를 이용하며, 사설 IP 주소를 할당 받은 사물인터넷 단말은 사설 VR 을 통하여 서비스 분배 시스템인 SDR(Service Distribution Router, 이하 SDR)로 트래픽을 터널링 하도록 한다. 터널링 하는 목적은 사설 IP 트래픽에 대해 일반 공중망을 유통하기 위해 사용하는 방식으로, GRE 등 다양한 터널링 기법을 사용할 수 있다. SDR 은 해당 사설 IP 단말이 해당되는 사물인터넷 사업자와 연결하여 통신이 되도록 중재하는 기능을 담당한다. 사물인터넷 단말과 사업자간에는 네트워크 모드로 사설 IP 통신이 가능하게 된다. 아울러, 인터넷 상의 라우팅 프로토콜을 이용하지 않고 라우터에 설정된 정책 기반으로 사설 트래픽을 터널링 하므로 사물인터넷의 End-to-End Secured Connectivity 구성이 가능하게 된다.

그리고, 사업자의 에지 노드는 공인 및 사설 VR 간 별도의 라우팅 정책을 통해 제어됨으로 상호간의 트래픽이 전달되는 것을 방지할 수 있고, 사물인터넷 단말에서 전달된 트래픽은 모두 SDR 을 통해 통신이 되고, 사설 IP 주소를 사용하므로 네트워크 계층에서 사물인터넷 해킹을 원천적으로 예방할 수 있다. SDR 은 사물인터넷 단말과 사업자의 사설 IP 주소를 본 후 이를 포워딩 또는 필터링할지 정할 수 있으며, 다른 사물인터넷 사업자로의 트래픽이 전달되지 않도록 제어할 수 있다.

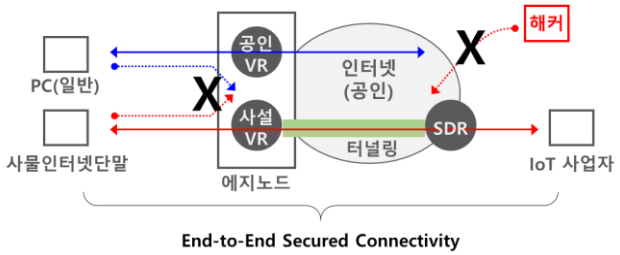


그림 1. 네트워크 서비스 모델

그림 1 과 같은 End-to-End Secured Connectivity 를 구성함으로써, 폭발적인 증가가 예상되는 사물인터넷 단말에 사설 IP 주소를 제공함으로 현재 부족한 공인 IP 주소 사용을 절약할 수 있으며, 인터넷 상에서 별도의 보안 프로그램 및 장치가 없어도 원천적으로 보안이 유지되는 장점이 있다. 다만, 이러한 서비스는 앞에서 언급한 바와 같이 범용적인 일반 인터넷 접속을 요하지 않은 사물인터넷과 같이 특수한 목적을 위한 네트워크 제공 서비스 모델이다. 이러한 서비스 모델은 인터넷 상에서 기존 전용서비스와 같은 솔루션을 대체할 수 있으며, CCTV, 환경·교통감시, 신용카드 결재 등과 같이 보안이 필수적인 사물인터넷 서비스에 매우 적합하면서도 경제적으로 제공할 수 있게 될 것이다.

3.2. 서비스 적용 방안

앞 절에서 설명한 서비스 모델을 적용하는 것이 기존 공유기 등에서 제공하는 방식과의 차이점은 사설 IP 네트워크 도메인이라 볼 수 있다. 기존 공유기가 사설 IP 주소를 할당하는 것은 동일한 서브 네트워크 내에 공인 IP 주소 영역이 존재하여야 한다. 즉, 인터넷 IP 주소 할당 원칙에 따른 라우팅이 되기 위해서는 같은 서브 네트워크 도메인 내에 공인 IP 주소가 존재하여야만 인터넷 상에서 포워딩이 가능하다. 그러나, 본 논문의 서비스 모델에서 사물인터넷 단말은 전국적으로 분포되어 있으며, 이들은 에지 노드의 라우팅 정책에 따라 사설 IP 주소가 할당이 되며, 사업자와 End-to-End 사설 네트워크 도메인으로 구성된다.

각 에지 노드에서 터널링되어 SDR 로 전달된 사설 트래픽은 각 사설 IP 주소 블록 별로 어느 사업자와 연결되어야 하는지 관리가 되어야 한다. 그리고, 사업자로부터 전달된 사설 트래픽이 어느 에지 노드로 터널링할지 결정하는 매커니즘이 있어야 한다. 본 논문에서는 이 매커니즘에 대해서는 생략하고, 어떻게 관리 및 제어가 되어야 하는지에 대해서만 언급하고자 한다.

그림 2 에서 보는 바와 같이, 본 솔루션이 적용되기 위해서는 사전에 사물인터넷 단말과 사업자에 대한 사설 IP 주소 정보 및 종단간 연결이 필요하다

는 합의가 되어야 한다. 이러한 방법은 기업간 사설 IP 주소를 이용한 인터넷 연결 시에 NAT 서버에게 종단간의 NAT 테이블을 고정적으로 설정하는 것과 유사하다고 볼 수 있다.

정책제어 시스템은 양 종단간의 연결을 위하여 수집된 정보를 가지고, 사전 합의된 종단간의 통신이 가능하도록 정책 제공을 위해 정책 적용 매핑 테이블이 설정되어야 한다. 설정된 매핑 테이블에 따라, SDR 에 정책을 적용하여야 한다. 여기서의 매핑 테이블은 기존 NAT 와 같이 시스템에 주소 변환을 하기 위한 것이 아니라, 해당 사설 트래픽이 어느 경로를 타고 전달되도록 할 것인지 결정하기 위해 사용된다. 경로에 대한 가입자 구간 또는 사업자로부터 사설 트래픽이 전달되면 SDR 은 정책제어 시스템에 의해 전달된 정책에 따라, 해당 경로로 트래픽을 전달하여 종단간 통신이 가능하도록 한다.

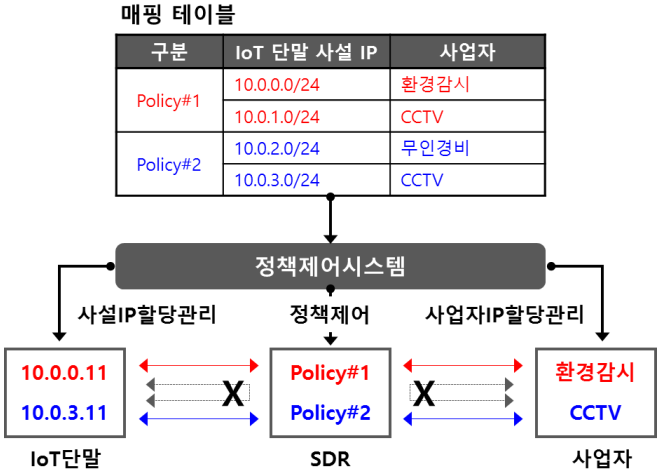


그림 2. SDR 트래픽 제어 방법

3.3 기존 네트워크 서비스 방법과의 차이

본 논문에서 제안한 방식은, 사물인터넷 단말에 사설 IP 를 제공하고, 인터넷 사업자의 에지 노드의 VR 및 터널링 기능을 이용하여 SDR 로 모든 Secured 트래픽이 전달되도록 하고, SDR 에 적용된 제어 정책을 통한 양 종단간 통신으로 마치 전용회선 수준의 Secured Connectivity 를 제공하는 방법이다. 인터넷 사업자에서 이와 유사한 서비스로 전용회선 서비스와 VPN 서비스가 존재한다. 참고로, VPN 서비스는 UTM 또는 공유기 등 별도의 접속장비를 이용하여 기업고객에게 보안 등의 차별화된 서비스를 제공하는 방식이다.

표 1 에서 보는 바와 같이, 본 제안 모델 및 기존 네트워크 서비스 모두 네트워크 계층에서 안전한 네트워크 서비스를 이용할 수 있게 해주는 솔루션들이다.

그러나, 각 솔루션들은 사용 목적 및 망 구성 방법, 이용 요금 및 적용 서비스 대상에 따른 차이가 있다.

본 논문의 제안 모델은 인터넷 사용을 직접 할 수 없다는 제약사항이 있지만, 환경감시, 무인경비 등과 같이 특수한 목적을 가지는 사물인터넷 서비스의 보안 이슈해소를 위해서는 매우 효율적인 솔루션이라 할 수 있다.

4. 결론

본 논문에서는 일반적으로 인터넷 사용자가 여러 대의 단말을 사용하기 위해 공유기를 설치하고 각 단말에 사설 IP 주소를 사용하는 방법을 응용하여, 인터넷 사업자가 사설 IP 주소를 이용하여 서비스를 제공하는 방법에 관한 새로운 구조와 방법을 제시하며, 특히 보안성 확보가 매우 중요한 사물인터넷 서비스의 End-to-End Secured Connectivity 제공 모델을 제안하고 있다. 본 논문에서 제안하는 네트워크 서비스 모델은 일반 인터넷 사용자에게 별도의 회선을 제공하지 않고, 기존 인터넷 접속회선을 이용하여 논리적으로 공인 IP 주소와 사설 IP 주소를 에지 노드에서 분리하여 라우팅 처리를 하도록 하고, SDR 을 이용하여 각 사업자들을 연결함으로써, 전용회선과 유사한 수준의 보안 서비스가 가능하다.

표 1. 제안모델과 유사 네트워크 서비스 비교

구 분	제안모델	VPN	전용회선
목 적	• 사물인터넷과 같은 특화 단말 중심의 회선임대 서비스	• 기업 본·지사간 연결을 위한 네트워크 서비스	• 회사 및 지점간 Point-to-Point 연결
공통점	• End-to-End Secured Connectivity 제공이 가능함		
차이점	• 인터넷 이용 불가 • 단말에 사설IP 사용 (환경에 따라 고정 or 유동방식) • 사설 트래픽 공중망 유통을 위한 터널링 방식 사용	• 별도 VPN 망 구성으로, 인터넷 이용 가능 • 본사/지사간 연결을 위해 MPLS를 통한 L2/L3 VPN 이용	• 별도 전용망으로 구성 • 기업, 관공서 등이 주 고객
가입자 시설	• 필요 없음	• 일부 VPN GW(or UTM) 등 구축	• 일부 전용회선 종단장치 필요
비용수준	• Low	• Medium	• High
용도	• 무인경비, 환경감시 등 사물인터넷 서비스	• 기업정보 유통 및 문서작업 등	• 본사 및 지사간 통신

아울러, 기존 공유기와는 달리 전국적으로 할당된 사설 IP 단말들을 SDR 을 통하여 분리하고 이를 사업자와 연결을 시키는 방법에 대해서도 고찰하였다.

이러한 서비스 모델을 이용하여 네트워크 사업자는 비용 효율적으로 기존 전용회선 또는 일부 VPN 서비스를 대체할 수 있는 기회를 갖게 될 것이며, 고객은 기존의 전용회선보다 저렴하지만 대등한 보안 수준으로, 인터넷 기가 속도 제공과 함께 다양한 서비스를 제공받을 수 있는 혜택을 가질 것으로 기대된다.

5. 참고 문헌

[1] 사물인터넷 보안 위협 동향, Internet & Security Bimonthly, 2014
[2] 사물인터넷(IoT) 정보보호 로드맵, 2014
[3] RFC791, “Internet Protocol”
[4] RFC1918, “Address Allocation for Private Internets”
[5] RFC3022, “Traditional IP Network Address Translator (Traditional NAT)”
[6] S.Shieh, F.Ho, Y.Huang and J.Luo, “Network Address Translators : Effects on Security Protocols and Applications in the TCP/IP Stack”, IEEE Internet Computing, Vol.4, No6, Nov./Dec 2000
[7] http://biz.olleh.com

IoTivity 기반의 출입 관리 방법

이응기 ○, 김현우, 주홍택

계명대학교 컴퓨터공학과

{leggod1004, hwkim84, juht}@kmu.ac.kr

요 약

본 논문에서는 다양한 IoT 플랫폼들을 연결해주는 중계기를 개발하기 위하여 IoT 플랫폼으로 IoTivity 를 선택하여 출입 관리 시스템을 개발하였다. 기존의 출입 관리 시스템들의 경우 각자가 정의한 프로토콜에 의해 개발되어 확장성이나 유연성이 떨어지기에 IoTivity 를 사용하여 확장성 있고 유연한 시스템을 개발하고자 하였다. 본 논문에서는 아두이노 RFID 모듈과 IP Camera 를 IoTivity 를 통해 서로 메시지를 주고받으며 출입자의 권한을 확인하는 방법을 제안하고 있다.

1. 서론

인터넷 기술의 발전과 사물인터넷(Internet of Things, IoT)에 대한 관심이 늘어남에 따라 IoTivity, Mobius, IoTMakers, AllJoyn 등과 같은 수많은 IoT 플랫폼들이 개발되고 있다[1]. IoT 플랫폼은 개발자들이 손쉽게 디바이스들을 제어하고 관리할 수 있도록 도와줌으로써, IoT 서비스 개발에 있어서 시간과 인건비 절약에 직접적인 도움이 된다.

그러나 무수하게 쏟아져 나오는 IoT 플랫폼들 사이에서도 아직까지 IoT 플랫폼에 대한 표준이 정해지지 않았다. 많은 기업들과 연구단체들이 각자의 프로토콜과 각자의 방식대로 IoT 플랫폼을 개발하고 있다. 이렇게 IoT 플랫폼의 표준이 확립이 되지 않았기 때문에 IoT 서비스 개발자들은 서비스를 개발하기 전에 어떤 IoT 플랫폼을 사용하여 어떤 기능을 구현해야 할지 면밀하게 검토를 해야 한다. IoT 플랫폼에 따라서 서비스의 기능이나 알고리즘이 달라질 수 있기 때문이다. 이런 서비스 개발자의 고민은 서비스 개발 초기에만 필요한 것이 아니라 추후에 서비스를 확장 하고 유지 보수할 때 더 중요하다. 추가/확장 하고자 하는 기능이 현재 사용하고 있는 IoT 플랫폼으로는 구현이 어려운 경우가 있다. 이럴 경우 서비스 개발자는 추가/확장 하려는 기능의 방향을 바꾸어야 한다. 최악의 경우에는 현재 서비스 중인 시스템을 뒤엎어 사용 중인 IoT 플랫폼을 바꿔야 할 수도 있다.

하지만, IoT 플랫폼들 사이에 IoT 중계기가 있으면 몇 가지 이점이 있다. 서비스 개발자는 추가하고자 하는 기능이 현재 이용 중인 IoT 플랫폼으로 구현이 어려울 경우 개발이 쉬운 다른 IoT 플랫폼을 선택하여 개발함으로써 두 가지 이상의 IoT 플랫폼을 중계기를 통해 하나의 서비스에서 동시에 사용할 수 있을 것이다. 또한, IoT 중계기를 사용하여 한 번이라도 개발은 해본 사용자라면, 처음 개발할 때 사용했던 IoT 플랫폼 이외의 다른 플랫폼을 사용하더라도 추가적인 학습을 할 필요가 없어지므로 개발을 하는데 소요되는 시간과 노력이 줄어들 것이다.

이러한 IoT 중계기를 개발하기 위해서는 우선 IoT 플랫폼들의 프로토콜과 구조를 이해하고 서로의 차이점을 비교하여, IoT 중계기 개발에 필요한 요소가 무엇인지 파악해야 한다. 이를 위해 본 논문에서는 IoT 디바이스를 위한 오픈소스 하드웨어와, IoT 플랫폼을 한 가지씩 선택하여 IoT 서비스 시스템을 구현하고자 한다.

우선 본 논문에서 선택한 IoT 플랫폼은 IoTivity 이다. IoTivity 는 다수의 IoT 디바이스들을 운영체제나 네트워크 프로토콜에 구애 받지 않고 쉽게 연결하기 위한 오픈소스 프레임워크이다. 현재 IoTivity 는 에디슨, 아두이노와 같은 오픈소스 하드웨어와 안드로이드, 우분투, 타이젠과 같은 소프트웨어 플랫폼도 지원하고 있으며 계속적으로 지원범위를 넓혀가고 있다. 이렇게 다양한 디바이스와 플랫폼을 지원하는 IoTivity 가 IoT 중계기를 개발하기 위한 첫 플랫폼으로 적합하다고 판단하여 선택하게 되었다[2].

IoT 디바이스는 아두이노로 선택하였다. 아두이노와 라즈베리파이를 고려한 이유는 널리 알려진 오픈소스 하드웨어로 IoT 디바이스로 개발하기에 편리하고 접근성이 용이하여 보편적으로 활용되고 있기 때문이다. 하지만 라즈베리파이의 경우 아직 IoTivity 에서 지원을 하지 않고 있어서 본 논문의 개발에는 제외되었다.

본 논문에서는 위와 같은 이유로 IoTivity 와 아두이노를 사용하여 IoT 시스템을 개발하였다. 본 논문에서 개발한 시스템은 출입 관리 시스템으로, 기존의 출입 관리 시스템들이 자체적인 프로토콜만을

사용하여 개발한 점을 개선하고자 IoT 플랫폼을 도입함으로써 유연하고 다양한 기능을 쉽게 구현이 가능하도록 하였다.

논문의 구성은 다음과 같다. 2 장에서는 관련 연구에 대해서 소개한다. 3 장에서는 출입 관리 시스템의 개발 요구사항에 대해 설명한다. 4 장에서는 출입 관리 시스템의 환경과 구조에 대해 설명한다. 5 장에서는 출입 관리 시스템의 시나리오에 대한 세부적인 동작 방법에 대해서 설명한다. 6 장에서는 본 논문에서 개발한 출입 관리 시스템에 대한 고찰과 함께 결론 및 향후 연구에 대해 설명한다.

2. 관련 연구

Y.-T. Park 외 2 명은 가정 자동화 기술을 출입관리장치인 도어락과 연동하여 가정에 출입 시 가정 내부의 센서들로부터 데이터를 얻어와 가정 내부의 상태를 출입자에게 피드백 해주는 방법을 제안하고 있다[3].

강성철 외 5 명은 특정 구역의 안전을 위하여 RFID 를 이용하여 미들웨어 기반의 출입문 제어 에이전트를 설계 및 구현하는 방법을 제안하고 있다[4].

W. Ping 외 4 명은 지문인식 기술을 이용하여 출입자의 출입을 관리하고, 무선통신 기술을 이용하여 원격으로 도어락 시스템을 관리하는 방법을 제안하고 있다[5].

위 연구들은 특정 구역의 출입자를 여러 가지 방법으로 제한하는 방법을 소개하고 있다. 하지만 각자가 정의한 프로토콜에만 의존한 시스템의 한계로 인하여, 외부 다른 서비스와 연동하거나 새로운 기능을 추가하기가 어렵다. 따라서 본 논문에서는 출입 관리 시스템을 IoT 플랫폼을 사용하여 확장성과 유연성을 높이고자 한다.

3. 개발 요구 사항

본 논문에서 제안하는 방법은 기존의 출입 관리 시스템을 IoT 플랫폼인 IoTivity 를 사용하여 구현하는 방법이다. 따라서 본 논문에서 제안하는 출입 관리 시스템은 IoT 서버와 클라이언트 간에 주고 받는 탐색, 제어 통신을 IoTivity 에서 지원해주는 기능을 통하여 구현해야 한다.

또한 기존의 단일 센서에 의존하는 출입 관리 시스템과는 달리 IoTivity 를 이용하여 한 대 이상의 디바이스를 제어함으로써 IoT 플랫폼이 가진 개발상의 유연함과 확장성에 대한 이점을 보여줄 수 있어야 한다. 더 나아가 출입 관리 시스템의 기능에 있어서도 두 가지 이상의 센서로부터 데이터를 취합함으로써 위장출입자의 출입을 방지하고, 출입 권한 질의의 결과에 대한 신뢰도를 높여야 한다.

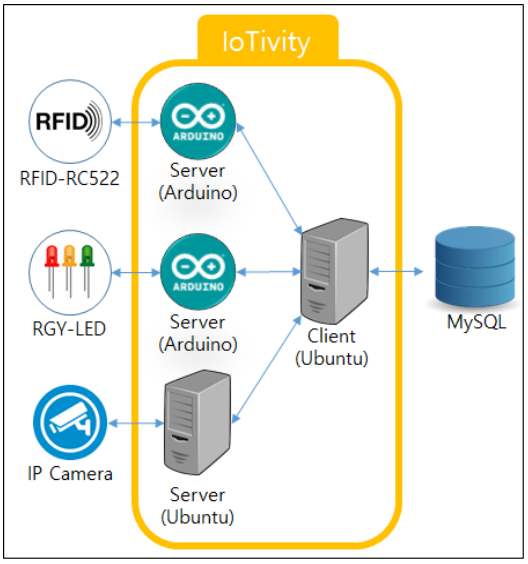


그림 1. 출입 관리 시스템 구조

4. 시스템 환경 및 구조

4.1. 시스템 하드웨어 사양

출입 관리 시스템의 IoTivity 클라이언트는 운영체제로 Ubuntu 14.04 LTS 64bit 버전을 사용하였고, CPU 는 Intel Core2 Duo CPU E8400 @ 3.00GHz*2, 메모리는 2 GB 이다. IoTivity 서버는 Arduino ATmega2560 두 대와 PC 한 대에 설치하였다. Server PC 의 경우 Client PC 와 같은 사양의 PC 를 사용하였다. Arduino ATmega2560 의 메모리는 256KB, SRAM 은 8KB 이다. ATmega2560 두 대중 한 대에는 RFID-RC522 센서모듈을 사용 하였고, 다른 한대에는 시각적인 피드백을 위해 Actuator 를 모터 모듈 대신하여 LED 모듈을 사용하였다.

4.2. 시스템 구조

그림 1 과 같이 출입 관리 시스템은 크게 IoTivity 서버와 IoTivity 클라이언트 두 가지로 나뉜다. IoTivity 클라이언트는 로컬네트워크 내에 있는 디바이스를 탐색하고 디바이스의 상태를 서버에게 요청할 수 있다. IoTivity 서버는 클라이언트의 요청(GET / POST / PUT / OBSERVE)에 따라 자신의 상태를 클라이언트에게 알려준다. 각 서버들은 클라이언트로부터 요청 메시지가 오면 각각의 현재 상태를 응답 메시지를 통해 전달하는데, 출입 관리 시스템의 각 서버들이 보내는 응답메시지는 다음과 같다.

RFID-RC522 모듈과 연결된 아두이노 서버는 클라이언트로부터 OBSERVE 요청이 오면 요청 직전에 RFID 리더에 읽힌 태그의 ID 값을 전달한다. IoTivity 에서 OBSERVE 메시지는 클라이언트가 서버에게 한번 요청하게 되면 서버가 디바이스의 상태를 담은 응답메시지를 일정 간격으로 계속해서 클라이언트에게 보낸다. IoTivity 에서는 디바이스의 변

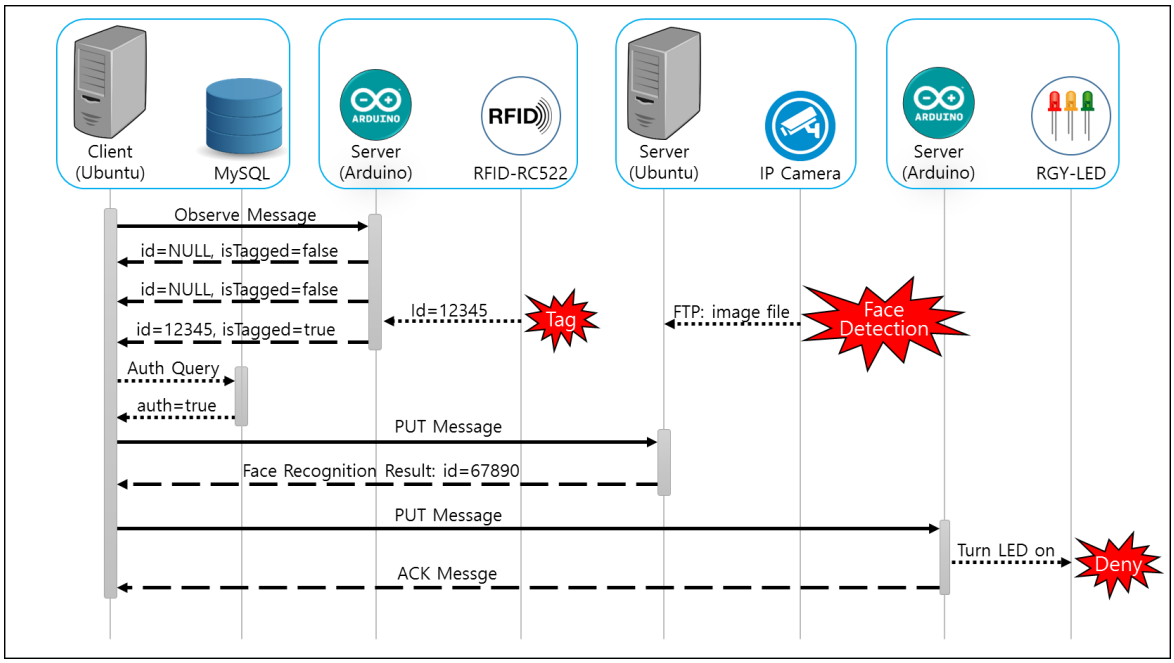


그림 2. 출입 관리 시스템 출입 거부 시나리오 시퀀스 다이어그램

화를 알리기 위하여 디바이스가 GET 또는 POST 메시지를 요청하는 것이 아니라, 클라이언트에서 OBSERVE 메시지로 디바이스의 상태 변화를 요청한다. 그래서 RFID 태그의 값을 읽어오기 위하여 OBSERVE 메시지를 이용하였다.

RGY-LED 와 연결된 아두이노 서버는 클라이언트로부터 출입자에 대한 질의 결과가 넘어오면 그 허가/불허가와 같은 경우에 따라 해당하는 LED 를 점등한다.

IP Camera 와 연결된 Ubuntu 서버는 클라이언트부터 요청이 오면 그 즉시 가장 최근에 얼굴감지로 인해 FTP 서버에 저장된 이미지의 출입자 얼굴을 인식하여 출입자의 정보를 클라이언트로 전송한다.

5. 출입 관리 시스템 구현

그림 2 는 본 논문에서 제안한 출입 관리 시스템이 타인의 RFID 를 도용하여 출입을 시도하는 출입자의 출입을 거부하는 상황의 시퀀스 다이어그램이다. 시퀀스 다이어그램에서 실선과 굵은 점선은 IoTivity 의 기능을 사용하여 전달하는 메시지이다. 작은 점선은 IoTivity 의 기능이 아닌 프로토콜을 사용하여 전달하는 메시지이다.

Ubuntu Client 에서 RFID-RC522 모듈이 연결된 Arduino Server 로 Observe Message 를 보내면 Arduino Server 에서는 일정주기로 Ubuntu Client 로 ACK Message 를 계속해서 보낸다. ACK Message 에는 RFID 리더에 태그 id 가 넘어오기 전까지는 계속해서 id=NULL(태그 된 id 의 string 변수), isTagged=false(태그가 되었음을 알리는 bool 변수)값이 전송된다. 그러다 RFID 리더로부터 id=12345 가 넘어오면 Arduino Server 가 가지고 있는 id=12345, isTagged=true 값이 함께 전달된다. Arduino Server 는 id 를 한번 전달 한 이후에는 다음 태그가 인식 될 때까지 다시 계속해서 ACK Message 에 id=NULL, isTagged=false 를 전송한다.

Ubuntu Client 에서 id 를 전달받으면 해당 id 의 권한을 DB 에 질의한다. 동시에 Ubuntu Server 로 PUT Message 를 전달한다. Ubuntu Server 는 PUT Message 를 받으면 IP Camera 가 출입자의 얼굴을 감지하여 FTP 를 통해 전송한 사진 중 가장 최근 사진을 인식하여 출입자의 신원을 파악한다. 신원이 파악된 출입자의 id=67890 는 Ubuntu Client 로 ACK Message 를 통해 전달된다.

Ubuntu Client 는 RFID 리더로부터 인식된 출입자가 출입 관리 시스템이 설치된 구역에 출입 권한이 있음에도 불구하고, IP Camera 로부터 인식된 출입자와 서로 다른 인물이라는 결과에 의하여 RGY-LED 모듈이 연결된 Arduino Server 로 PUT Message 를 전송한다. PUT Message 를 받은 Arduino Server 는 RGY-LED 에 Deny 를 나타내는 LED 를 점등하고 Client Ubuntu 로 ACK Message 를 전송한다.

이와 같이 RFID 와 IP Camera 두 가지 센서의 정보를 혼합하여, 출입자가 타인의 RFID 태그를 도용하여 출입을 시도한다는 사실을 판단하고 출입을 제지하였다.

이와 같이 단일 센서가 아닌 다중센서에 의한 출입 권한 질의로 출입 관리 시스템의 결과에 대한 신뢰도가 높아졌다.

6. 결론 및 향후 연구

본 논문에서는 일반적인 출입 관리 시스템을 IoT 플랫폼을 사용하여 개발함으로써 기존 시스템보

다 유연하고 확장성 있는 시스템을 개발하고자 하였다. 제안된 출입 관리 시스템은 IoT 플랫폼 중 오픈소스 플랫폼인 IoTivity 를 사용하였고, IoTivity 의 기능을 활용하여 세 가지의 IoTivity 서버 모듈을 연동하였다. IoTivity 서버와 클라이언트가 주고받는 메시지들은 모두 IoTivity 를 통하여 전달하였다. 또한, RFID 와 IP Camera 서버 에서 수집한 데이터를 바탕으로 출입자의 신원과 출입공간에 대한 권한을 동시에 확인 함으로써, 단일 센서에 의존하던 기존의 출입 관리시스템과는 달리 RFID 카드 도용에 의한 출입자의 위장출입을 방지하고 권한 질의에 대한 결과에 신뢰도가 높아졌다.

향후에는 신뢰성, 확장성 측면에서 출입 관리 시스템의 성능을 분석할 것이다. 또한, IoTivity 이외의 IoT 플랫폼과 연동이 가능한 IoT 플랫폼 중계기를 개발 할 것이다.

7. 참고 문헌

[1] 김재호, 윤재석, 최성찬, 류민우, “IoT 플랫폼 개발 동향 및 발전방향”, 한국통신학회지 (정보와 통신), 제 30 권 제 8 호, pp. 29-39, 2013 년 7 월.

[2] 이원석, 차홍기, 전종홍, “사물인터넷 오픈소스 기술 – IoTivity”, 정보와 통신 열린강좌, 제 32 권 제 12 호(별책 2 호), pp. 27-35, 2015 년 11 월.

[3] Y.-T. Park, P. Sthapit, and J.-Y. Pyun, “Smart digital door lock for the home automation,” in *Proc. TENCON 2009 - 2009 IEEE Region 10 Conference*, pp. 1-6, Jan. 2009.

[4] 강성철, 강민성, 김형찬, 도양희, 이광만, 김도현, “RFID 미들웨어 기반의 출입문 제어 에이전트 설계 및 구현”, 2006 년도 추계학술발표논문집, pp. 650-653, 2006 년 11 월.

[5] W. Ping, W. Guichu, X. Wenbin, L. Jianguo, L. Peng, “Remote Monitoring Intelligent System Based on Fingerprint Door Lock”, in *Proc. International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 2, pp. 1012-1014, May. 2010.

8. 기타

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

An Architecture for Managing IoT Device Object in the IMS

왕기초 O, 양우석, 김정호, 이재오

한국기술교육대학교 전기전자통신공학과

Qicha0117@gmail.com, {oos13, jungho32, jolee}@koreatech.ac.kr

요 약

With the fast developing growth of information technology, more and more network functions connecting each others are more popular. In order to efficiently manage emerging concept of the network which allows any object in the real world is connected together at anywhere and anytime through the integration of object identification, interaction, and information gathering technologies. Besides, IP Multimedia Subsystem (IMS) is an architectural framework for delivering IP-based information to users. Thus, Internet of things and IP Multimedia Subsystem (IoT-IMS) communication platform can provide a convenient way to efficiently deploy a novel application service to users. In this paper, an IoT information gateway is proposed to effectively collect and manage sensor data in IoT-IMS communication platform. Thus, IoT-IMS communication platform can provide a convenient way to efficiently deploy a novel application service to users. Through the IoT Application Service Scenario, people can be easily to observe and manage gathering devices information in a unified mode.

1. Introduction

With the rapid development of information and technologies, objects and things, actuators and things, are more and more equipped with communicating modules to connect the various types of wireless, wired, and mobile networks.

Internet of things (IoT) allows objects in the real world can be connected together at anywhere and anytime through the integration of object identification and interaction technologies [1]. For providing a convenient approach to access thing information, a novel and unified application service is needed. The framework of IP Multimedia Subsystem (IMS)-based Application Service (AS) can be fit the design requirement. In our previous background, we have developed IoT-enabled IMS-based ASs which is considered to integrate with identification and interaction among things [3, 4].

IoT Information Gateway (IIG) can collect the raw data from devices such as IoT devices, produce abstract-level meaningful IoT information from the raw data of the devices, and even make a reasonable decision to do some actions among the collected raw data information.

The user's applications can easily to retrieve the meaningful raw data information from IIG via an IoT-enable AS in IoT-IMS communication platform.

2. Related Works and Research Back Grounds

As for service discovery, the Constrained RESTful Environments (CoRE WG) has defined a mechanism denoted as Resource Directory (RD), to be adopted in IoT applications. The use of RD is necessary because of the impracticality of a direct resource discovery, due to the presence of duty-cycled nodes and unstable links in LINs. The registration of a resource in the RD is performed a HTTP/ POST request to the RD, while the discovery can be accomplished by issuing a GET request to the well-known/core URI. This discovery mechanism is totally self-contained in Constrained Application Protocol (CoAP) as it uses only CoAP messages.

The adoption of the Session Initiation Protocol (SIP) provides an alternative mechanism to register resources on a RD, which may be also called Constrained of the Session Initiation Protocol (CoSIP) Registrar Service. The advantage of using a CoSIP based registration mechanism is that it might be possible to register resources other than those reachable through Constrained Application Protocol (CoAP), thus providing a scalable and generic mechanism for service discovery in constrained applications with a higher degree of expressiveness, such as setting an expiration time for the registration[9].

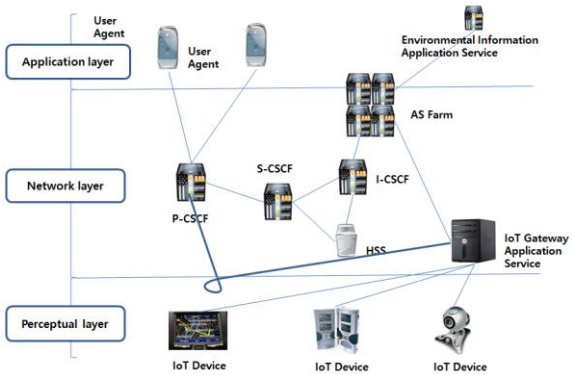


Figure 1.An AS acting as IoT gateway in IoT-IMS Communication Platform

The IoT-IMS communication platform although the perceptual layer can deliver the sensing information obtaining from sensors to related IoT applications via the network layer, the collected sensing information may be transmitted individually [7]. It will lead the transmission overhead of delivering sensing information. Figure 1 show an IoT gateway can efficiently collect and unite the sensing information to throttle the traffic flow of sensing information among IoT devices and corresponding applications [8]. The gateway is a key component for collecting, recording and forwarding sensing data obtained from the devices.

The IoT gateway also acts as a proxy perception layer and network layer towards the IoT things that are connected to it. The common features of IoT gateway should be included multiple interfaces, protocol conversion, and manageability [5]. Therefore, a well-designed IoT gateway is programmable for efficiently aggregating raw data information dissemination from perception layer to the network layer [2].

3. The framework of IIG acting as an IoT-IMS AS

In this section, the IoT-enable application service technology will be adopted to design the IoT information gateway (IIG) collecting and managing the sensing data among IoT things in order to reduce the traffic flow of delivering sensing information in the IoT-IMS communication platform. The proposed IIG framework is depicted in Figure 2.

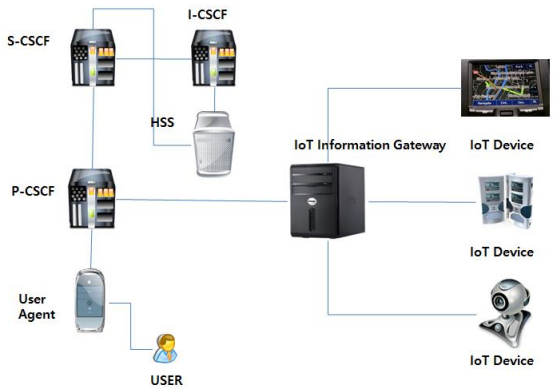


Figure 2. The framework of IIG acting as an IoT-IMS AS

The functional modules of proposed IoT-enable of IIG are shown in Figure 3. The majority of IIG are four modules: IoT SIP Register Service, Sensing Data Collection, Information Aggregation, and Information modules. For control message forwarding to the IoT device, The remainder modules of IIG are used to context-aware computing through things interaction to achieve the thing intelligence in IoT-IMS communication platforms.

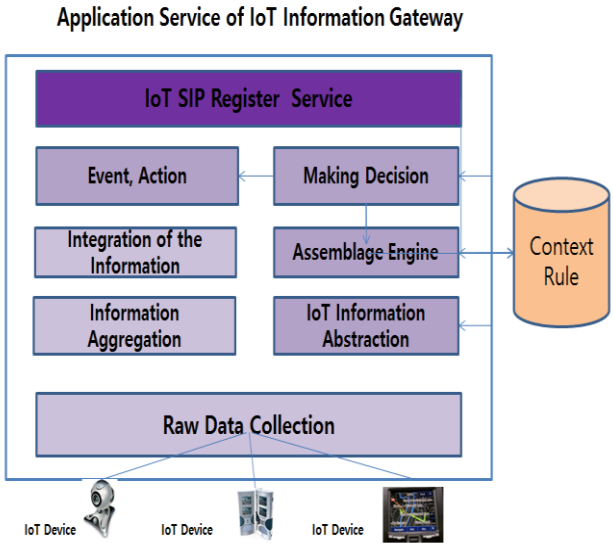


Figure 3. Functional modules of IoT-enable Applic at-ion Service of IIG

Raw Data Collection module is responsible for collecting the sensing raw data from the devices in the perception layer. Information Aggregation module aggregates various types of raw data into a compact mode. In order to encapsulate aggregated information in a readable fashion. The IoT Information Abstraction module is responsible for converting the IoT sensing. Raw data is used to the meaningful IoT information for conveniently examining and realizing the meaning of IoT raw data information. The Assemblage Engine module analyzes the information to summarize the suggesting results based on context rules. The Making Decision module will refer the suggesting results to make some decision to trigger some event occurrences or due to some actions through the Action and Event module. The IoT SIP Register Service module is though the SIP header and identification of IoT information.

The IIG retains the feature of context-awareness [6]. It can easily integrate the ubiquitous and pervasive computing to realize the smart intelligence in IoT networks.

Figure 4 shows a request of sensing information from user carried by a SIP MESSAGE. To carry the aggregated sensing raw data between devices and IIG, it adopts the SIP "MESSAGE" and "200 OK" messages to encapsulate the requests and responses of sensing information respectively. A specific tag header, called "EVENT", is also embedded into the SIP "MESSAGE" and "200 OK" messages to identify and extract the sensing information from SIP messages based on the "EVENT" tag.

```
MESSAGE sip:iot-devices SIP/2.0
Via: SIP/2.0/UDP 220.68.65.162:5060;branch=z9h
From:<sip:devices@iot-ims.korearech>;tag=31415
To: <sip:iot-ims-as>
Call-ID: apb304a94sslfaser2le93aj22
Cseq: 3584 MESSAGE
...
Contact: iot-device id
Address of record: iot-device id<d1,d2,d3>
Route:<sip:orig@s-cscf.iot-ims.koreatech:5060;lr>
Event: iot-device
...
<?xml version="1.0"encoding="UTF-8"?>
<IoT_Information_Gateway type="request"id="1"time="4294967295">
<iot-device id="d1">
<iot-device id="d2">
<iot-device id="d3">
...
</IoT_Information_Gateway>
```

Figure 4. A request of sensing information from user carried by a SIP MESSAGE

Figure 5 shows a complete service registration and IoT Application Scenarios by CoSIP. A network element, denoted as "IoT Information Gateway", which includes also a HTTP/CoAP proxy, which can be used by nodes residing outside the constrained network to access CoAP services. CoSIP allows smart objects to register the services they provide to populate an IoT SIP Register Service, which serves as a RD. The terms "IoT SIP Register Service" and "Resource Directory" are here interchangeable. Upon receiving the registration request, the Registrar Server stores the Address of Record (AoR) to Contact Address mapping in a Location Database and then sends a 200 OK response.

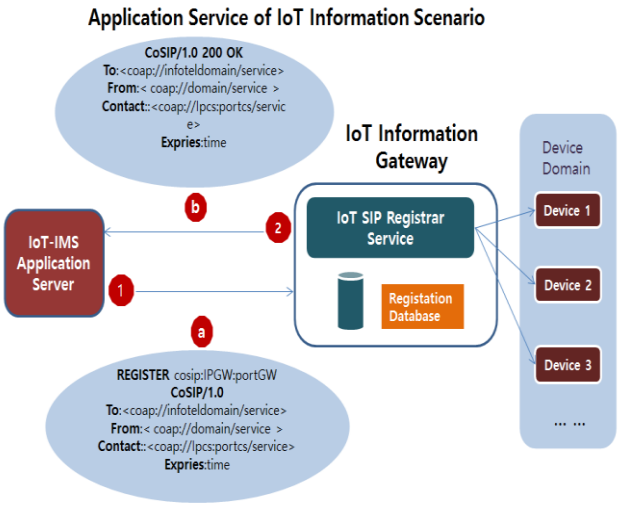


Figure 5. IoT Application Service Scenario

The depicted Application Service of Information scenarios consider as several Subscribe/Notify interactions: the notifications can be either send by an IoT-IMS Application Server and an IoT information Gateway, subscribers can be mobile users. Let's assume that the notifiers have registered with their IoT CoSIP Registrar Service. (This step is also denoted as the Publishing phase in a typical Subscribe/Notify scenario). The standard subscription/notification procedure is the following:

1. The subscriber sends a request to the user, and specifying the service events.
2. The user stores the subscriber's and event inform ation and sends a 200 OK response to the subscr iber.
3. Whenever the user' s state changes, it sends a m essage to the subscriber.
4. The subscriber sends a 200 OK response back to the user.

4. Conclusion

IoT Information Gateway (IIG) is also developed to collect the changes of sensing information obtaining from the IoT devices and the changes of environmental statuses. It also set some triggering events according the changes of IoT sensing information located in the environment for sen -sing the effects of context-awareness. Therefore, it can easily to organize a scenario environment to service the things interaction and observe the effects of the IIG AS in IoT-IMS communication fashion. We believe that the next step in the field of Internet of Things (IoT) is to realize a virtual computing platform that provides access to heterogeneous group of device resources present in our environments.

In this paper, an IoT information gateway is proposed to effectively collect and manage data in IoT-IMS communication platform. Thus, IoT-IMS communication platform can provide a convenient way to efficiently deploy a novel application service to the device users. Through the IIG, mobile users can easily observe and manage gathering information in a unified mode. . Through the IoT Application Service Scenario, people can easily observe and manage gathering devices information in a unified mode.

5. References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, Vol. 54, No. 15, pp. 2787-2805, 2010.

[2] L. Tan and N. Wang, "Future internet: The Internet of Things," in *Proceedings of the 3rd International Conference* on Advanced Computer Theory and Engineering (ICACTE 2010), Vol. 5, pp. V5-376 - V5-380, 2010.

[3] J. M. Hsu and C. H. Hsieh,"Deployment of IMS-based Voice over IP Service with IOT-enable Interaction Mechanism on Smart Phones," in *Proceedings of 2012 International Conference* on Business and Information (BAI2012), pp. H-1125-H-1130, 2012.

[4] C. Y. Chen, H. C. Chao, T. Y. Wu, C. I. Fan, J. L. Chen, Y. S. Chen, and J. M. Hsu, "IoT-IMS Communication Platform for Future Internet," *International Journal of Adaptive, Resilient and Autonomic Systems*, Vol. 2, No. 4, pp. 74-94, 2011.

[5] H. Chen, X. Q. Jia, and H. Li, “A brief introduction to IoT gateway,” in *Proccedings of IET International Conference* on Communication Technology and Application (ICCTA 2011), pp. 601-613, 2011.

[6] T. Gong, Z. Hu ; H. F. Liu, F. Lin, D. Zhou, and H. Tian, “A contextaware computing mediated dynamic service composition and reconfiguration for ubiquitous environment,” in *Proceedings of 2012 3rd International Conference* on the Internet of Things (IOT), pp. 16-23, 2013.

[7] S. Li, G. Oikonomou, T. Tryfonas, T. Chen, and L. Xu, “A Distributed Consensus Algorithm for Decision-making in service-oriented Internet of Things,” *IEEE Transactions on Industrial Informatic s*, Vol. pp.1461-1468, 2014.

[8] Jenq-Muh Hsu,Ching-Yo Chen,” A Sensor Information Gateway based on Thing Interaction in IoT-IMS Communication Platform,” *Tenth International Conference* on Intelligent Information Hiding and Multimedia Signal Processing, pp.835-838,2014

[9]Simone Cirani, Marco Picone, and Luca Veltri,” CoSIP: a Constrained Session Initiation Protocol for the Intern et of Things,” *ESOCC Workshops, volume 393 of Com munications* in Computer and Information Science, pp. 13-24. Springer, 2013.

ICBMS 플랫폼 연동 및 매쉬업 서비스 개발을 위한 Smart Mediator 연구

이도영 º, 정세연, 정태열, 홍원기

포항공과대학교 컴퓨터공학과

{dylee90, jsy0906, dreamerty, jwkhong}@postech.ac.kr

요 약

인터넷의 발달과 데이터를 공개하여 새로운 가치를 창출하고자 하는 오픈데이터화 움직임에 힘입어 이를 이용해 새로운 매쉬업 서비스를 개발하고자 하는 시도가 활발히 진행되고 있다. 새로운 매쉬업 서비스를 개발하기 위해서는 매쉬업 서비스에 필요한 데이터 및 서비스를 제공하는 여러 플랫폼들을 연동해야한다. 따라서 좋은 매쉬업 서비스를 만들기 위해서는 매쉬업 서비스개발에 필요한 플랫폼들의 유기적인 연동과 다양한 방법으로 제공되는 데이터를 효과적으로 처리하는 기능들이 필수적이다. 하지만 지금까지의 매쉬업 서비스 개발 환경은 개발자들이 매쉬업 서비스에 필요한 플랫폼들의 인터페이스를 이해하고 직접 모든 플랫폼들을 연동해야 하는 어려움이 존재했다. 본 논문에서는 이러한 어려움을 해결하기 위해 ICBMS 플랫폼 연동 및 스마트 중재 기술의 개념을 제안 및 구현하였으며 이를 활용하여 실제 매쉬업 서비스를 생성함으로써 제안하는 ICBMS Smart Mediator 를 검증하였다.

1. 서론

인터넷의 발전은 사용자로 하여금 방대한 데이터에 손쉽게 접근하고 이용할 수 있는 환경을 제공해왔다. 더불어 근래에는 공공기관과 기업 등이 자신들이 보유한 데이터를 공개하고 이를 누구나 이용할 수 있게 제공하여 새로운 가치 창출을 추구하고 있다. 이렇게 공개된 오픈데이터들을 이용하여 개발자들은 새로운 서비스들을 개발하여 제공했는데 이처럼 다양한 출처의 데이터와 서비스를 연동하여 새롭게 생성된 서비스를 매쉬업 서비스라고 부른다.

오픈데이터는 그 데이터를 공개하는 기관이 제공하는 OpenAPI 를 통해서 접근 및 활용이 가능하다. 하지만 각 기관에서 제공하는 OpenAPI 의 형태는 통일되어 있지 않고, 그 OpenAPI 를 사용하기 위한 토큰 획득과 인증 문제 등 번거로운 절차를 거쳐야 한다는 어려움이 존재한다. 또한, 개발자가 다양한 기관에서 제공하는 데이터를 이용하고자 하는 경우 그 데이터 출처에 맞는 OpenAPI 를 이용하고 연동 인터페이스를 구축해야한다는 단점이 존재한다. 때문에 매쉬업 서비스에 더 많은 플랫폼을 연동할 수록 개발 부담은 증가하게 되고 이는 매쉬업 서비스 구축을 어렵게 만드는 원인이다.

이러한 어려움을 극복하기 위해서 본 논문에서는 ICBMS SM (IoT, Cloud, Big Data, Mobile, Security Smart Mediator)라고 명명된 새로운 매쉬업 서비스 개발을 가능하게 하는 ICBMS 플랫폼 연동 및 스마트 중재 기술을 소개한다. ICBMS SM 의 가장 직접적인 용도는 현재 서비스가 제공되는 IoT, Cloud, Big Data, Mobile, Security 플랫폼을 연동하여 새로운 매쉬업 서비스를 보다 쉽게 개발 가능하도록 만드는 것이다. 앞으로 개발될 매쉬업 서비스는 다양한 플랫폼에서 제공되는 데이터와 서비스를 이용해야 할 것이며 이러한 요구사항을 만족시키기 위해 플랫폼들을 효과적으로 지원하는 방안이 필요하다. 결과적으로 ICBMS SM 은 매쉬업 서비스를 간단하고 쉽게 개발할 수 있는 환경을 제공함으로써 유용한 새로운 매쉬업 서비스 개발을 촉진시킬 수 있다. 이를 위해 ICBMS SM 은 다양한 플랫폼 접속 기능을 제공함과 동시에 플랫폼 간 데이터 연동 및 중계 기능을 제공한다. 뿐만 아니라 ICBMS 플랫폼에서 생성된 데이터에 의미를 부여하고 이를 공개하여 매쉬업 서비스에서 활용할 수 있도록 제공한다.

2. 관련 연구

다양한 플랫폼들을 융합하여 서비스를 제공하는 사례는 기존에도 국내외에서 다수 존재해왔다. 국내 사례로는 [1]에서 이종 기기 간에 연결 및 데이터 처리의 어려움을 극복하고 디바이스 제조사, 이용자 등 다양한 종류의 매쉬업 요구를 만족시키는 클라우드 기반의 IoT 매쉬업 서비스 모델 (IoTMaaS) 를 제안하였다. [2]에서는 다양한 모바일 매쉬업 서비스의 늦은 로딩 시간과 과도한 트래픽 병목 현상 등의 기존 모바일 매쉬업 문제를 해결하기 위한 방법

으로 모바일에서는 디스플레이나 간단한 로직만 처리하고 실제 서비스는 중앙 집중 서버에서 수행되는 OpenAPI 를 활용한 클라우드 기반 모바일 매쉬업을 개발하였다. 또한 [3]에서는 Google, Amazon, Naver 등 다양한 기업에서 제공하는 Web 2.0 OpenAPI 와 SaaS (Software as a Service) 플랫폼은 서로 호환성을 제공하지 않는 단점을 해결하기 위해 SaaS(Software as a Service) 플랫폼과 매쉬업 툴의 연동을 지원하는 매쉬업 매니저를 설계하였다. [4]에서는 선행 연구인 통합 SNS 게이트웨이를 개선하여 IoT 와 SNS 의 매쉬업 서비스를 제안하였다. 통합 SNS 게이트웨이는 IoT 어댑터와 SNS 어댑터를 통해 IoT, SNS 와 통신을 수행하고, Analysis Engine 을 통해서 SNS 플랫폼에서 IoT 플랫폼을 제어할 수 있는 이종 플랫폼간 매쉬업 서비스를 제공한다.

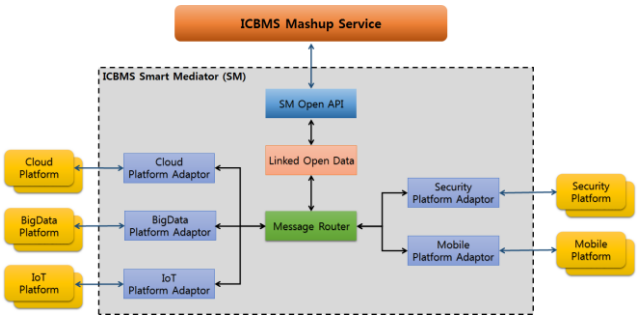
국외 사례로는, [5]에서 이종의 다양한 IoT 데이터에 효율적으로 접근하고 가치 있는 데이터를 만들기 위해 Data Mashup 프레임워크를 제안하였다. 또한, [6]에서는 Mobile 에 Cloud Computing 을 접목하여 모바일 웹 콘텐츠를 제공하는 LAMEC (Lightweight architecture for mobile web content access over enterprise cloud mashup) 방법을 제안하였으며 모바일과 클라우드를 매쉬업하여 모바일 웹 콘텐츠 사용시 전송되는 데이터 사이즈를 최소화하는 방법을 고안하였다.

위에서 소개된 관련 연구 [1-6]는 ICBMS 플랫폼을 연동하여 매쉬업 서비스를 구축하거나 그 과정에서 발생하는 이슈들을 구조적 관점에서 해결한다는 성과를 보이지만, 사용되는 ICBMS 플랫폼의 종류와 구축되는 매쉬업 서비스의 목적이 제한적이라는 한계가 있다. 반면, 본 논문에서 제안하는 SM 은 매쉬업 서비스 개발자들에게 보다 포괄적인 성격의 매쉬업 서비스 구축 환경을 제공하는 것을 목표로 하며, 다양한 ICBMS 플랫폼 간의 연동 기술에 초점을 맞춘다.

3. ICBMS SM 구조

매쉬업 서비스를 효과적으로 개발하기 위해서는 연동해야 할 플랫폼들의 종류와 개수를 결정하는 것이 선행되어야 한다. 이를 위해 본 연구에서는 매쉬업 서비스에 이용되는 플랫폼들의 영역을 크게 다섯 가지 (IoT, Cloud, Big Data, Mobile, Security)로 나누었다. 플랫폼들간에 제공되는 서비스 및 데이터의 연동을 위해서는 중간에서 각 플랫폼들을 잇는 중계기가 필요하다. 또한 실제 플랫폼들 마다 앞에서 언급한 중계기에 연결되기 위한 연동 인터페이스가 구축되어야 한다. 이러한 요구사항들을 만족시키기 위해 본 논문에서는 중계기 역할을 하는 Smart Mediator (SM)를 구현하여 각 플랫폼 간에 이루어지는 서비스 연동 및 데이터 처리를 관장한다.

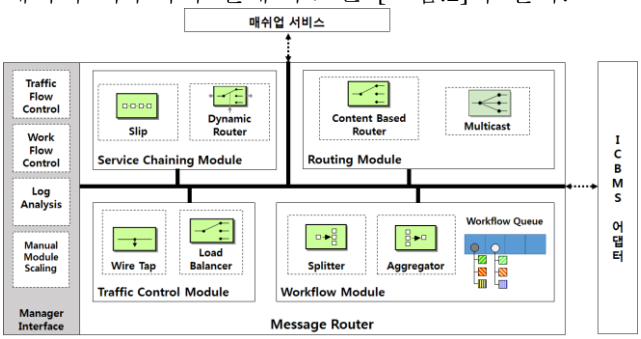
본 논문에서 제안하는 매쉬업 서비스를 위한 ICBMS 플랫폼 연동 및 스마트 중재 기술을 위한 프레임워크인 SM 의 구성요소는 크게 1) 메시지 라우터, 2) ICBMS 어댑터들, 3) Linked Open Data, 4) OpenAPI, 5) 개발된 매쉬업 서비스이다 [그림.1].



[그림.1] ICBMS SM 구조도

3.1. 메시지 라우터

SM 의 가장 중요한 특징은 다양한 산업별 ICBMS 플랫폼에서 제공하는 정보 및 서비스를 연동하여 매쉬업 서비스를 구축하기 위한 프레임워크를 제공하는 것이다. 이를 위해 사용자의 요청을 수신 및 분석하여 이에 대응하는 여러 ICBMS 플랫폼에 대한 서브 요청으로 분할, 전달하며 플랫폼에서 반환된 응답을 종합하여 사용자에게 전달하는 역할을 수행하는 중계기가 필요하다. 메시지 라우터는 기능적인 측면에서 이러한 역할을 수행하는 동시에 관리적인 측면에서 해당 프레임워크 자체 또는 그 위에 구축된 매쉬업 서비스에 대한 트래픽 제어, 스케일링, 플랫폼 관리자 화면 등의 기능을 제공한다. 메시지 라우터의 전체 구조는 [그림.2]와 같다.



[그림.2] 메시지 라우터 구조

각 모듈에 대한 소개에 앞서 메시지 라우터는 일종의 EIP (Enterprise Integration Pattern)기반의 EAI (Enterprise Application Integration) 프레임워크로 볼 수 있다 [10]. EAI 프레임워크는 메시지 라우터와 다양한 ICBMS 플랫폼의 어댑터 간에 메시지 기반의 커뮤니케이션을 통한 연동 및 통합을 지원한다. EIP 는 EAI 를 이용한 어플리케이션의 통합에 있어서 자주 사용되는 어플리케이션 간 메시지 처리 패턴, 용어 등을 기술하고 있으며, 이를 통해 보다 효과적이고 체계적인 방식으로 어플리케이션간 통합에 접근할 수 있다. 메시지 라우터를 구성하는 각각의 모듈들은 EIP 를 기반으로 내부 모듈 간 메시지 라우팅 및 메시지 라우터와 ICBMS 어댑터 간의 라우팅

방법을 설계한다.

- **라우팅 모듈**

사용자로부터 매쉬업 서비스 요청을 수신 받아 내용을 분석하고 서비스 실현을 위한 라우팅 경로를 정의한다. 라우팅 경로의 목적지는 메시지 라우터의 다른 내부 모듈이거나 ICBMS 플랫폼 어댑터, 또는 여러 목적지의 조합이 될 수 있다. 사용자로부터의 서비스 요청은 라우팅 정보를 포함하는 메시지 형태로 재가공되어 다음 목적지로 전달된다. 서비스 요청 메시지 분석에 따른 라우팅 정보 추출을 위해 Content Based router EIP 를 적용한다.

- **서비스 체이닝 모듈**

라우팅 모듈의 매쉬업 서비스 요청에 대한 분석 과정에서 특정 서비스의 경우 여러 개의 목적지를 순차적으로 거치면서 서비스 또는 데이터를 제공받는 형태로 실현된다. 예를 들어, 클라우드 내부에 저장된 파일을 빅데이터 분석 요청하기 위한 서비스의 경우, 서비스 요청 → ICBMS 플랫폼 통합인증 → Cloud 플랫폼 접근 → Big Data 플랫폼 접근의 서비스 체인을 가진다. 서비스 체이닝 모듈은 서비스 요청 메시지가 서비스 체인을 따라 라우팅 될 수 있도록 Slip 과 Dynamic Router EIP 를 이용하여 설계된다.

- **워크플로우 모듈**

SM 에서의 워크플로우는 시간 측면에서의 작업의 흐름으로 나타낸다. (그냥 스케쥴링이라고 해) 여러 IoT 디바이스 스트리밍 데이터를 실시간 분석하는 매쉬업 서비스의 경우, 여러 IoT 플랫폼에 병렬적으로 쿼리를 전달하고 결과를 취합하여 Big Data 플랫폼에 전달, 실시간으로 분석 결과를 제공받을 것을 요구한다. 워크플로우 모듈은 서비스 요청 메시지의 분할(전달) 및 취합 기능과, 실시간 서비스 요청, 배치 프로세싱 등 시간 측면에서의 요구 기능을 지원한다. 이를 위해 Splitter 와 Aggregator EIP 기반으로 설계된다.

- **트래픽 제어 모듈**

메시지 라우터는 다양한 매쉬업 서비스와 여러 종류의 ICBMS 플랫폼을 연동 및 중계하기 때문에 많은 양의 네트워크 트래픽을 처리해야 한다. 또한, 특정 매쉬업 서비스가 지나치게 많은 네트워크 자원을 사용하는 경우 다른 매쉬업 서비스와의 Fairness 문제를 야기하므로 이러한 서비스를 감시 및 식별하고 제어할 수 있는 도구가 필요하다. 메시지 라우터 관리자는 트래픽 제어 모듈을 통해 매쉬업 서비스 수준에서 발생하는 트래픽을 확인하고 네트워크 관리 정책을 적용할 수 있어야 한다. 또한, 자율적인 모듈 단위 스케일링 및 트래픽 로드밸런싱을 통해서 SM 의 안정적인 매쉬업 서비스 구축 환경과 확장성을 제공해야 한다. 이를 위해 Wire Tap 과 Load Balancer EIP 를 적용하여 설계 된다.

3.2. ICBMS 어댑터

ICBMS 플랫폼의 종류만큼이나 서비스를 제공하는 방식도 각 플랫폼 별로 다양하다. 클라우드 서비스 기반의 ICBMS 플랫폼의 경우 IaaS, PaaS (Platform as a Service), SaaS (Software as a Service) 등의 형태로 서비스를 제공하고 있다. 또한 각 ICBMS 플랫폼 별로 전달되는 데이터의 종류 또한 다양하다(예: 스트리밍 데이터, LOD). ICBMS 어댑터는 각 ICBMS 플랫폼과 메시지 라우터 사이의 인터페이스 역할을 수행하는데, 이를 통해 매쉬업 서비스의 다양한 ICBMS 플랫폼 요구사항을 지원한다.

- **IoT 어댑터**

IoT 어댑터는 다양한 IoT 디바이스의 허브 역할을 수행하는 IoT 플랫폼과 SM 의 연동을 지원한다. 대표적인 IoT 플랫폼으로는 Mobius, IoTMakers 등이 있다. IoT 플랫폼이 제공하는 기능 및 서비스를 매쉬업 서비스 구축에 활용하기 위해, 플랫폼에 대한 장기적인 연결성 보장, 데이터의 실시간/비실시간성 보장, 디바이스 프로파일 관리 지원, 스트리밍 데이터 처리 지원, 사용자 디바이스 접근 권한 제어 등의 기능이 지원되어야 한다.

- **Big Data 어댑터**

Big Data 어댑터는 Google BigQuery 와 같은 PaaS 형태, Amazon EMR (Elastic MapReduce)과 같은 IaaS 형태 또는 Apache Spark 와 같이 사용자의 개인 환경에 구축된 Big Data 플랫폼과 SM 의 연동을 지원한다. Big Data 플랫폼이 제공하는 기능 및 서비스를 매쉬업 서비스 구축에 활용하기 위해, 다양한 데이터소스간 연동을 통한 분석 기능, 입력된 빅데이터를 학습하고 데이터 모델을 구축하여 데이터를 분석하는 기능 (머신러닝), 분석된 데이터에 대한 다양한 시각화 등의 기능이 지원되어야 한다.

- **Cloud 어댑터**

Cloud 어댑터는 IaaS, PaaS, SaaS 형태의 다양한 Cloud 플랫폼에서 제공하는 서비스를 SM 에 연동시키는 기능을 수행하며, 대표적인 Cloud 플랫폼으로는 Amazon EC2, Google Compute Engine 등이 있다. . Cloud 플랫폼이 제공하는 기능 및 서비스를 매쉬업 서비스 구축에 활용하기 위해, Cloud 서비스를 생성하고 설정하여 사용하기 위한 접속 기능, Cloud 플랫폼이 제공하는 자원에 대한 상태 감시 및 사용 설정에 관한 기능, Cloud 플랫폼 자원에 대한 명시적 표현 및 기술 방안 등이 제공되어야 한다.

- **Mobile 어댑터**

Mobile 어댑터는 다양한 Mobile 플랫폼(예, 카카오톡, Twitter, Android 어플리케이션 등)을 SM 에 수용하는 기능을 제공하며, 다른 ICBMS 플랫폼에서 제공되는 정보 및 서비스를 Mobile 플랫폼 사용자에게 전달하거나, SNS 에서 수집된 정보를 활용하는 매쉬업 서비스에 사용된다. 이를 위해, 다양한 Mobile 플랫폼에 대한 통합 인증, 주기적인 폴링 혹은 비주기적인 푸시 방식의 통신, 멀티캐스트, 브로드캐스트 방식의 통신 기능이 지원되어야 한다.

- **Security 어댑터**

ICBM 플랫폼 자체적으로나 이를 연동하는 과정

에서 수많은 보안 위협에 노출되기 때문에, SM 에 대한 통합적인 보안 체계가 필요하다. Security 어댑터를 통해 Security 플랫폼(예, 방화벽, IDS 등)을 연동하여 매쉬업 서비스를 구축하는데 있어서의 보안 관련 요구를 효과적으로 제공할 수 있어야 하며, ICBM 플랫폼 데이터에 대한 암호화 및 익명화 기능, 보안 처리 요청 및 응답에 대한 스케줄링 기능 등이 지원되어야 한다.

3.3. Linked Open Data

Linked Open Data 는 기계가 이해할 수 있게 시맨틱 웹 기술을 사용하여 데이터에 의미를 부여하고 하이퍼링크를 사용하여 서로 연계시킨 데이터 공유, 연계, 재이용 기술을 말한다. SM 은 플랫폼들에서 발생하는 데이터를 Linked Open Data 형태로 변환하여 매쉬업 서비스에 필요한 데이터들을 쉽게 재사용 할 수 있도록 제공한다.

3.4. OpenAPI

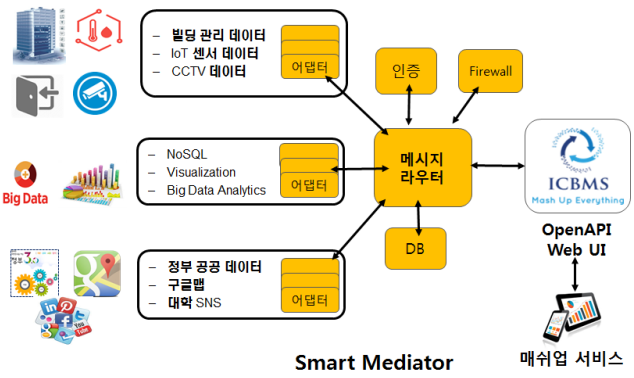
SM 은 OpenAPI 를 통해 기능들을 제공한다. 개발자는 OpenAPI 를 통해 매쉬업 서비스에 필요한 플랫폼들의 데이터에 접근할 수 있다. 또한 플랫폼들이 제공하는 기능 활용 및 제어도 OpenAPI 를 통해 가능하다. OpenAPI 는 별도의 OpenAPI 포털을 통해 개발자에게 제공된다

4. 구현

본 논문에서는 제안한 SM 및 SM 을 이용한 매쉬업 서비스 개발의 유효성을 검증하기 위한 하나의 예시로 빌딩 종합 관리 매쉬업 서비스의 프로토타입을 구현하였다.

4.1. SM

SM 은 크게 SM 구성요소간에 연동 및 메시지 전달을 담당하는 메시지 라우터와, ICBMS 플랫폼을 SM 에 연동하기 위한 인터페이스 역할의 어댑터들로 구성된다. 추가적으로 SM 의 보안 및 안전성을 위해 인증 및 방화벽 기능을 포함한다. 이러한 구성요소들은 각각 Docker [8]의 Container 단위로 구현되어 Kubernetes [11]를 통해 통합 관리되며 SM 과부하 및 장애 시에 Scale-in/out, Auto-healing 등의 기능을 제공한다 [그림.3].



[그림.3] Smart Mediator 구현

4.2. OpenAPI 포털

OpenAPI 포탈은 매쉬업 서비스 개발자가 SM 을 통해 새로운 매쉬업 서비스를 보다 편리하게 개발할 수 있도록 웹 기반의 인터페이스를 제공한다. 개발자는 원하는 매쉬업 서비스의 특성에 따라 연동할 플랫폼, 데이터소스, job 스케쥴링, 시각화, 분석 등의 기능을 단계별로 설정할 수 있다. 이러한 설정 및 요청 항목들은 각 항목들에 대해 미리 정의된 OpenAPI 형태로 변환되어 SM 으로 전달된다. 이후 각종 ICBMS 플랫폼과 연동되어 처리되며 그에 대한 결과로 웹 페이지 기반의 매쉬업 서비스가 생성된다 [그림.4].

4.3. 매쉬업 서비스

SM 을 통해 구현된 빌딩 종합 관리 매쉬업 서비스는 주로 IoT 플랫폼과 Big Data 플랫폼의 서비스를 이용하며 이를 위해 각각의 어댑터를 필요로 한다. IoT 어댑터는 빌딩 내부의 데이터를 수집하며, Big Data 어댑터는 수집되어 DB 에 누적된 데이터를 분석하여 스마트한 빌딩 관리를 위한 의미 있는 정보를 제공한다.

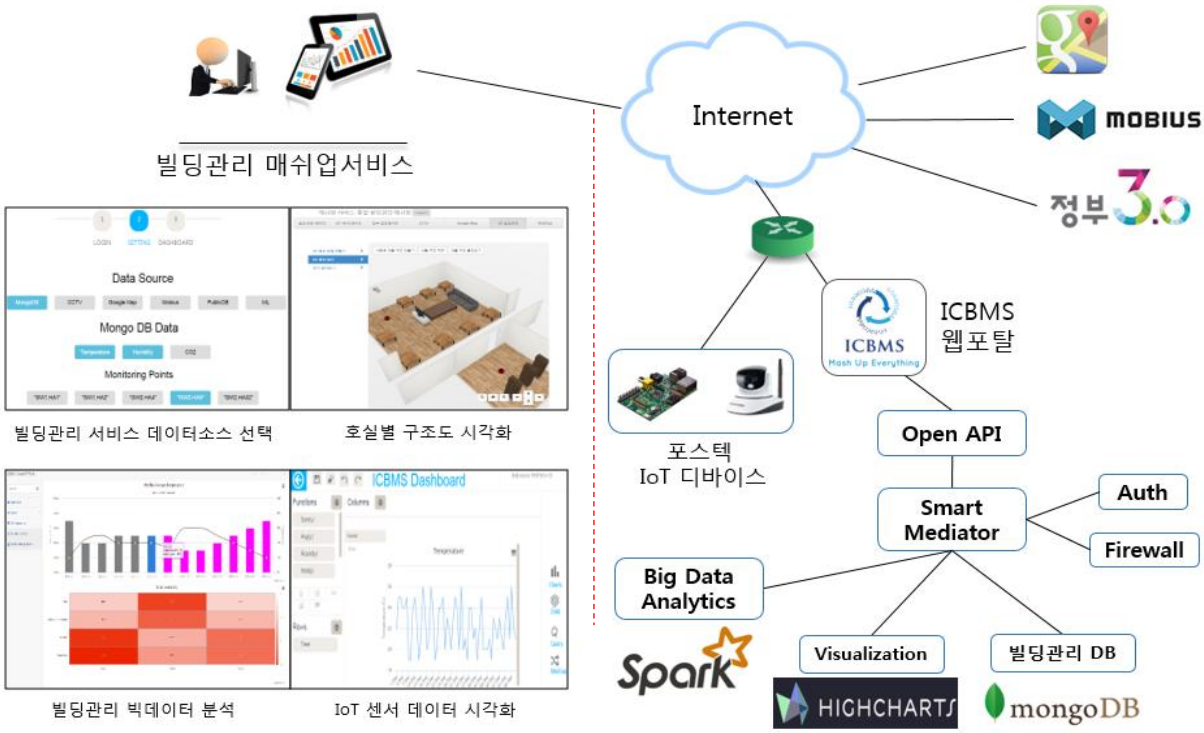
구현된 IoT 어댑터는 KT 에서 제공하는 IoTMakers 플랫폼과 SKT 와 KETI 에서 제공하는 Mobius 플랫폼을 SM 에 연동하는 인터페이스를 구현한다. 해당 IoT 플랫폼들은 포항공과대학교 정보통신연구소에 설치한 IoT 센서들로부터 온도, 습도, 조도 값을 지속적으로 수집한다. 수집된 데이터는 IoT 어댑터를 통해 SM 내부의 MySQL 관계형 데이터베이스에 저장되며 D2RQ [7]를 통해 LOD 로 변환된다. 이를 통해 수집된 IoT 데이터는 매쉬업 서비스에서 설정된 시간 단위로 그래프나 차트 등으로 시각화 되며 LOD 의 온톨로지 정보를 활용해 호실 또는 센서 단위의 연관관계를 가진다.

Big Data 어댑터는 SM 에서 MongoDB 에 누적된 기존의 빌딩 관리 자체 데이터를 이용하기 위해 NoSQL 플랫폼과 연동된다. 또한 누적된 대용량 데이터를 분석하기 위해 Apache Spark [9]를 사용하고 SM 과의 연동 인터페이스를 구현하였다.

이와 같이 구현된 SM 기반 빌딩 종합 관리 매쉬업 서비스는 지도 상의 위치를 포함한 빌딩 정보, IoT 센서 및 CCTV 데이터를 호실 별로 시각화(3D, 차트 등)된 모니터링 환경으로 제공한다. 또한 실내 온도와 정부 3.0 날씨 데이터(중기예보조회, 불쾌지수 등)를 연동하여 미래의 에어컨 사용량 예측 및 불쾌지수를 고려한 냉방 지표를 제공하여 매쉬업 기반의 보다 스마트한 빌딩 관리 환경을 제공한다.

5. 결론 및 향후 연구

본 논문에서는 다양한 플랫폼들을 효과적으로 연동하여 매쉬업 서비스 개발 환경을 제공하는 ICBMS SM 의 개념과 구조를 제안하였다. 또한 제안한 ICBMS SM 을 구현하고 이를 활용하여 빌딩 종



[그림.4] 빌딩 관리 매쉬업 서비스 구현

합 관리 매쉬업 서비스를 개발함으로써 ICBMS SM 의 기능을 검증하였다. ICBMS SM 은 개발자가 매쉬업 서비스를 쉽게 개발할 수 있는 환경을 제공하며 매쉬업 서비스에 필요한 플랫폼들을 효과적으로 연동한다. 또한 SM 은 플랫폼들에서 생성된 데이터에 의미를 부여하고 저장 및 공개해 매쉬업 서비스에서 활용할 수 있도록 제공한다. 마지막으로 SM 은 OpenAPI 포털을 통해 SM 의 기능을 제공하여 개발자가 효율적으로 매쉬업 서비스를 개발할 수 있도록 돕는다.

향후 계획으로는 다양한 시나리오를 고려하여 실제 SM 에 연동되는 플랫폼의 종류를 늘리고 이러한 플랫폼 연동을 위한 인터페이스를 체계적으로 정립하는 것을 목표로 한다. 또한 효과적인 매쉬업 서비스 기능 제공을 위한 효율적인 서비스 체이닝 및 워크 플로우 제어 기술 연구 등을 진행 하며, 각 플랫폼을 통해 얻어진 데이터를 LOD 형태로 변환하여 매쉬업 서비스를 구축하는데 의미 있게 사용할 수 있는 방법을 고안할 계획이다.

6. 감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-15-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

7. 참고 문헌

[1] D. J. Im, S. Kim, and D. Kim, “IoT Mashup as a Service: Cloud-based Mashup Service for the Internet of Things,” in Proc. 10th International Conference on Services Computing (SCC), pp. 462-469, June, 2013.
[2] 이용주, “Open API 를 활용한 클라우드 기반 모바일 매쉬업 개발”, 한국정보기술학회논문지, 제 12 권, 제 3 호, pp. 155-161, March, 2014.
[3] 임예준, 이승룡, “이기종 SaaS 플랫폼과 매쉬업 툴의 연동을 지원하는 매쉬업 매니저 설계”, 한국통신학회 하계종합학술발표회논문집, pp. 933-934, June, 2010.
[4] 김태영, 주홍택, “사물인터넷 환경을 위한 SNS 통합 게이트웨이 구조 개선”, 한국통신학회 동계종합학술 발표회, pp. 831-832, January, 2015.
[5] Du Zhiquan, Yu Nan, Cheng Bo, and Chen Junliang, “Data Mashup in the Internet of Things,” in Proc. International Conference on Computer Science and Network Technology (ICCSNT), pp. 948-952, December, 2011.
[6] Shawkat K. Guirguis, Adel A. El-Zoghabi, and Mohamed A. Hassan, “Lightweight Architecture for Mobile Web Content Access over Enterprise Cloud Mashup,” in Proc. World Symposium on Computer Applications & Research (WSCAR), pp. 1-8, January, 2014.
[7] Bizer, Christian, and Andy Seaborne. "D2RQ-treating non-RDF databases as virtual RDF graphs." Proceedings of the 3rd international semantic web conference (ISWC2004). Vol. 2004. Hiroshima: Citeseer, 2004.
[8] Docker, https://www.docker.com/
[9] Apache Spark, http://spark.apache.org/
[10] Enterprise Integration Patterns http://www.enterpriseintegrationpatterns.com
[11] Kubernetes, http://kubernetes.io/

스파크 클러스터의 SLA 기반 오토스케일링

오유리 ○, 최지은, 김윤희

숙명여자대학교 컴퓨터과학과

{yoori0203, jechoi1205, yulan}@sookmyung.ac.kr

요 약

인터넷 및 모바일 기기의 발달로 인해 전세계적인 사용자들이 생성하는 데이터 양이 기하급수적으로 증가하고 있다. 이러한 빅데이터 시대를 반영하여 대량의 데이터를 분석하는 일은 필수적이다. 스파크는 분산처리 프레임워크로 실시간 스트리밍 데이터를 처리가능하며, 디스크 I/O 가 아닌 인메모리 컴퓨팅을 지원하여 기존에 많이 사용되었던 하둡보다 빠른 실행이 가능해졌다. 또한 클라우드 기술은 언제 어디서나 원하는 시점에 원하는 만큼의 자원을 사용할 수 있는 환경이 제공한다. 스파크를 이용한 빅데이터 분석 처리 시, 자원 요구량을 예측하기 어렵기 때문에 유연한 자원사용이 가능한 클라우드 기술을 접목한다면 한층 더 효율적인 작업 실행이 가능하다. 이때, SLA 기반의 적절한 자원 오토스케일링은 자원을 낭비하지 않고 능률적인 실행을 가능하도록 하며 가장 잘 구성되어야 하는 부분이다. 본 연구에서는 스파크 클러스터의 SLA 기반 오토스케일링을 제안한다. 이는 어플리케이션의 SLA 를 만족하는 실행을 보장하기 위하여 실시간 모니터링을 통해 오토스케일링을 실행한다. 특정 어플리케이션이 SLA 를 만족하지 못하는 상황이 예측되었을 때, 다른 어플리케이션의 실행상황을 분석하여 SLA 를 만족하는 선 내에서 자원을 협상(negotiate)한다. 이로써 모든 어플리케이션이 SLA 를 충족하는 효율적인 자원 활용이 가능하도록 한다.

1. 서론

인터넷과 모바일 기기의 발달로 인터넷에서 생성되는 데이터의 양이 기하급수적으로 증가하고 있는 추세이다. 이에 따라 초기에 등장한 빅데이터 분산처리 프레임워크인 하둡[1]은 맵리듀스 알고리즘을 이용하여 단일 머신으로 처리가 불가능한 작업을 수 십대의 머신에서 나누어 처리하는 기능을 지원하였다. 그러나 하둡[1]은 데이터를 디스크에 읽고 쓰기 때문에 I/O 병목현상, 실시간 처리가 불가능하다는 단점이 드러났다. 이러한 상황을 반영하여 최근에 등장한 분산프레임워크인 스파크[2]는 인메모리 컴퓨팅을 이용하며, 실시간 스트리밍 데이터 처리를 지원한다. 빅데이터 분석은 일괄 처리로 끝나기 어렵고, 연속적인 여러 단계를 거쳐 장시간 동안 처리되는 경우가 많다. 이는 다양한 작업이 실시간으로 실행되는 시스템에서는 동적인 자원 관리가 필수적이다.

필요로 하는 자원이 런타임에 동적으로 달라짐에 따라, 대용량의 자원을 원하는 시점에 필요한 만큼 빌려 쓸 수 있는 클라우드 기술은 이러한 문제점에 해결책으로 사용될 수 있다. 클라우드 기술은 가상화를 통해 자원의 물리적인 제약을 받지 않고 유연한 자원의 사용이 가능하므로 스파크의 유동적인 자원 요구량을 충족시킬 수 있는 환경을 제공한다. 이 때, 다양한 SLA(Service Level Agreement; e.g. 비용, 데드라인) 및 자원 요구사항과 예측할 수 없는 작업 제출 시점 등 예측하기 어려운 상황에 대해 적절한 자원 오토스케일링은 필수적이다.

본 논문에서는 클라우드 컴퓨팅 환경에서 스파크 클러스터의 자원을 효율적으로 활용하기 위한 오토스케일링을 제안한다. 제안하는 오토스케일링은 데드라인 SLA 를 만족시키는 것을 목표로 한다. 따라서 데드라인을 만족하는 범위에서 효율적인 오토스케일링이 가능하다.

본 논문의 구성은 다음과 같다. 1 장의 서론에 이어 2 장에서는 관련 연구들을 살펴보고, 3 장에서는 본 논문에서 제안하는 SLA 기반 오토스케일링 기법을 적용한 프레임워크 구조에 대하여 설명한다. 4 장에서는 SLA 기반 오토스케일링 기법에 대해 설명하고 마지막으로 5 장에서 결론을 맺는다.

2. 관련연구

1) 스파크

기존에 분산 프레임워크로 활발하게 연구되었던 하둡[1] 프레임워크에 이어 스파크[2] 프레임워크가 등장하였다. 스파크[2]는 하둡과 유사하게 작업을 여러 노드에 분산처리하며 큰 특징으로 인메모리 컴퓨팅을 지원한다. 데이터 처리를 위해 디스크 I/O 를 실행하였던 하둡[1]과 달리 새로운 개념의 데이터 구조인 RDD(Resilient Distributed Dataset)를 사용한

메모리에서의 컴퓨팅을 지원한다. 이는 반복하는 실행 또는 스트리밍 데이터 처리에 용이하여 결과적으로 더욱 빠른 실행결과를 도출할 수 있다.

이러한 스파크의 효율적인 실행을 위한 연구로는 [3-5]가 있다. [3]은 온라인 병렬 분석 서비스를 제공시, 효율적이고 신뢰성 있는 서비스를 제공하기 위한 엔트로피 개념 기반의 스케쥴링 기법을 제안한다. 엔트로피 개념을 시스템의 무질서도를 측정하는데 사용되며 이는 신뢰성을 판단하는 기준이 된다. 또한 동적인 코어 성능을 측정하여 자원을 스케쥴링하는데 이용한다. [4]는 빅데이터를 처리할 때, 병렬접근의 최적화에 관한 연구이다. 이를 위하여 병렬데이터 요청을 저장 서버와 매칭하는 방법을 제시하며, 대화형 데이터 접근 방식에서 성능 향상을 위한 계획을 제시한다. [5]는 스파크를 이용한 날씨 이벤트 감지 및 추적 시스템을 제안한다. 이를 위하여 새로운 데이터 구조인 sRDD(scientific RDD)를 이용하여 분석을 진행하고, 단계간의 데이터 재사용을 특징으로 언급한다.

2) 오토스케일링 기법

사용자의 요구사항을 충족시키며, 효율적인 자원 사용을 위하여 자원에 대한 오토스케일링에 관한 연구는 [6-8]가 있다. [6]은 하이브리드 클라우드 환경에서의 오토스케일링 기법을 제안한다. 과학응용을 실행하기 위하여 사설 및 공용 클라우드 환경에서 데드라인을 만족하는 비용 효율적인 스케쥴링 방법을 제안한다. [7]은 응용의 패턴을 고려한 하이브리드 클라우드 환경에서 오토스케일링 기법을 제안한다. 응용의 특성 및 작업의 데드라인을 고려하여 효율적인 자원 활용을 증명하였다. [8]은 하둡 클러스터의 오토스케일링 기법을 제안한다. 실행시간에 대한 일반적인 모델을 제시하고, 각 워크로드에 맞게 튜닝 후, 예상 실행시간을 공식화 하였다. 이를 기반으로 오토스케일링 기법을 제안하였다.

스파크 또는 오토스케일링 기법에 관한 여러 관련연구들이 있었으나, 스파크 클러스터에 대한 오토스케일링을 진행한 연구는 없었다. 또한 스파크의 런타임 도중 다양한 자원 및 데드라인의 요구사항과 예측할 수 없는 작업 제출 시점 등의 상황에 대한 클라우드 환경의 오토스케일링은 필수적이다. 따라서 본 연구에서는 스파크 클러스터의 SLA 기반의 오토스케일링을 제안한다.

3. 오토스케일링 프레임워크 구조

본 논문에서 제안하는 오토스케일링 프레임워크 구조는 그림 1 과 같다. 사용자가 SLA(데드라인, 비용)를 제출하고 스파크 어플리케이션을 수행하고자 하는 경우, 스파크 어플리케이션은 오픈스택 자원을 활용하여 작업을 실행한다. 이때, 자원 및 작업에 대한 관리는 'YARN scheduler' 가 담당한다. 'Yarn scheduler'는 동적인 자원관리를 위해 VM 모니터링과 제어를 담당한다. 또한 작업실행을 위해 작업 모니터링과 실행기능을 갖는다.

'Auto-Scaling Service'는 오토스케일링 프레임워크의 핵심적인 서비스로 런타임 동안 모니터링을 통해 오토스케일링을 진행한다. 사용자가 제출한 SLA를 충족시키기 위하여 실시간 작업에 대한 모니터링을 하며, 오토스케일링이 필요한 시점에 자원에 대한 스케쥴링을 진행한다. 'Metadata Management Service'는 작업, 자원, VM 에 대한 요약된 정보를 가지고 있으며 이는 작업 스케쥴링 시, SLA 만족여부를 확인하는 용도로 사용된다.

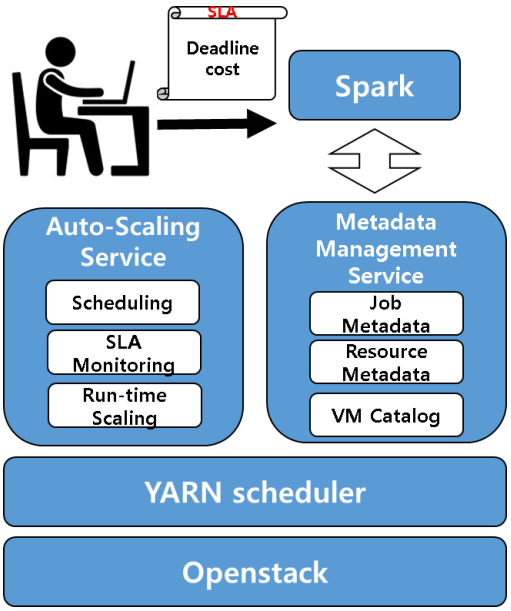


그림 1. 오토스케일링 프레임워크 구조

4. SLA 기반 오토스케일링

본 논문에서는 SLA 기반의 오토스케일링 기법을 제안한다. 스파크 어플리케이션과 같은 빅데이터 분석 응용은 동적으로 작업의 부하가 변화하여 작업 수행에 대한 예측 및 자원 사용에 대한 예측이 어렵다. 따라서 주기적인 모니터링을 통해 실시간으로 변하는 환경을 확인하고 사용자의 요구사항을 충족시키는 것이 중요하다.

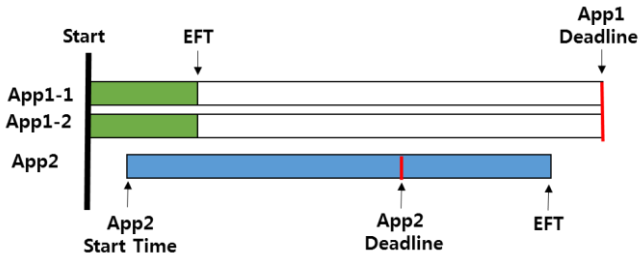


그림 2. 초기 스케쥴링 결과

그림 2 는 초기 스케쥴링 결과이다. 어플리케이션

1 이 제출되고 실행되는 도중 어플리케이션 2 가 제출되어 실행되는 상황이다. 각 어플리케이션의 색깔 막대는 예상 수행시간을 의미하며 EFT(Estimated Finish Time) 은 예상 종료시간을 의미한다. 또한 빨간 선은 사용자가 요구하는 데드라인을 의미한다.

어플리케이션 1 이 제출되었을 때에는 모든 자원을 사용 가능한 상태이다. 따라서, 어플리케이션 1 에 제출된 데드라인보다 상당히 빠른 시간 내에 작업을 끝낼 수 있는 상황이며, 이는 빠른 예상종료시간을 통해 알 수 있다. 어플리케이션 1 이 수행되는 도중에 어플리케이션 2 가 제출되었다. 시스템은 어플리케이션 2 가 제출될 것을 예상하지 못했기 때문에 어플리케이션 2 를 위한 충분한 자원을 할당하지 못했다. 어플리케이션 2 의 데드라인보다 더 많은 시간 작업을 해야 작업이 끝나는 상황이 되어, 이는 시간상으로 데드라인보다 예상종료시간이 뒤에 있음을 통해 확인할 수 있다. 이러한 경우, 자원의 균등한 분배가 일어나지 않아 먼저 수행을 시작한 어플리케이션이 과도하게 빠른 시간 내에 수행되고, 나중에 제출된 작업은 데드라인을 만족시킬 수 없는 상황이 일어난다.

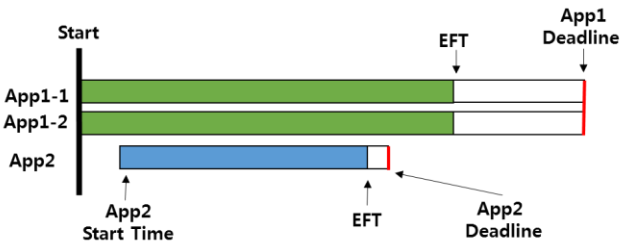


그림 3. 스케쥴링 결과

앞서 초기 스케쥴링을 이용하여 작업을 수행하는 경우, 오토스케일링을 통해 어플리케이션 1 과 어플리케이션 2 모두의 각각 데드라인을 만족하는 작업수행이 가능하다. 주기적인 모니터링을 통해 모든 어플리케이션이 사용자가 요구한 데드라인을 만족하는지 확인하고 데드라인을 만족하지 못하는 어플리케이션이 있는 경우, 해당 어플리케이션을 위하여 다른 어플리케이션의 실행 상황을 분석하여 자원을 여유 있게 사용하는 어플리케이션의 자원을 수거하여 데드라인을 만족하지 못하는 어플리케이션에 배정한다. 위의 예에서 어플리케이션 2 의 예상종료시간과 데드라인을 비교하였을 때, 데드라인을 만족하지 못하는 상황을 인지하고 오토스케일링을 실행한다. 어플리케이션 1 이 데드라인에 비해 예상종료시간이 빨리 끝나므로, 어플리케이션 1 의 자원을 일부 수거하여 어플리케이션 2 에 배정한다. 어플리케이션 1 은 활용하는 자원이 줄어들었기 때문에 이전보다 예상수행시간은 늘어났지만 여전히 예상종료시간은 데드라인을 만족한다. 어플리케이션 2 의 경우는 추가적인 자원을 할당 받았기 때문에 예상종료시간이 이전보다 줄어들어 데드라인 내에 작업을 종료할 수 있게 되었다. 어플리케이션 1 은 일부 수거된 자원으로 인해 예상종료시간이 늘어났지만 여전히 데드라인을 만족하는 실행이 가능하다. 위 오토스케일링 시나리오를 통해 어플리케이션 1 과 어플리케이션 2 은 효율적인 자원 협상으로 인해 모두 데드라인을 만족하는 작업수행이 가능함을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 스파크 클러스터에서 효율적인 자원 활용을 위한 SLA 기반 오토스케일링을 제안하였다. 기하급수적으로 늘어난 데이터로 인해 스트림 데이터 등 빅데이터 분석에 적합한 스파크 클러스터를 활용하는 추세이며 제안하는 오토스케일링 프레임워크에서 스파크 클러스터는 클라우드 컴퓨팅의 가상화 기술을 이용하여 동적으로 원하는 시점에 원하는 만큼의 자원활용이 가능하도록 한다. 또한 오토스케일링 기법 적용 시나리오를 제시한다. 시나리오에서는 기존 실행하고 있던 어플리케이션이 SLA 를 만족할 수 있도록 실행되고 있는 상태에서 또 다른 새로운 어플리케이션이 제출되었을 때, 이후에 제출된 어플리케이션이 충분한 자원을 할당받지 못해 SLA(데드라인)을 만족하지 못하는 경우에 대해 언급한다. 이때, 기존에 실행하던 어플리케이션의 자원의 일부를 수거하고 이후에 제출된 어플리케이션을 위해 재할당한다면, 두 어플리케이션 모두 데드라인을 만족하는 실행이 가능함을 보였다. 이는 효율적인 자원활용을 위하여 오토스케일링의 필요성을 나타내었다.

향후에는 SLA 정책에 따른 효율적인 오토스케일링 알고리즘을 제안하고 실험을 통하여 스파크 클러스터에서 오토스케일링 기법의 효과를 증명할 예정이다.

6. 참고문헌

[1] Apache hadoop. [Online]. Available: http://hadoop.apache.org/.

[2] Apache Spark: Lightning-fast cluster computing, “Apache spark,” 2015. [Online]. Available: https://spark.apache.org/

[3] H. Chen and F. Z. Wang, "Spark on entropy: A reliable & efficient scheduler for low-latency parallel jobs in heterogeneous cloud," Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th, Clearwater Beach, FL, 2015, pp. 708-713.

[4] J. Yin and J. Wang, "Optimize Parallel Data Access in Big Data Processing," Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, Shenzhen, 2015, pp. 721-724.

[5] R. Palamuttam et al., "SciSpark: Applying in-memory distributed computing to weather event detection and tracking," Big Data (Big Data), 2015 IEEE International Conference on, Santa Clara, CA, 2015, pp. 2020-2026.

[6] 강혜정, 고정인, 김윤희, "과학 계산 응용 실행을 위한 하이브리드 클라우드에서의 SLA 기반 VM 오토-스케일링 기법", 정보과학회논문지: 시스템 및 이론 제 40 권 제 6 호, pp. 266-273, 2013 년 12 월
[7] 안윤선, 정솔, 김윤희, "하이브리드 클라우드상의 과학응용을 위한 가상 자원 오토스케일링 기법", 정보과학회논문지: 시스템 및 이론 제 41 권 제 4 호, pp. 266-273, 2014 년 8 월
[8] Gandhi, Anshul, et al. "Autoscaling for Hadoop Clusters."

7. 기타

"이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보・컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(2015M 3C 4A7065646)."

SDN/SFA 기반 미래 인터넷 연구자원 플랫폼 연동 구조

SDN/SFA-based Federation Architecture of Future Internet Testbeds

김성민 [1], 이수강 [1], 석우진 [2], 김명섭 [1]

고려대학교 컴퓨터정보학과 [1]

KISTI [2]

{gogumiking, sukanglee, tmskim}@korea.ac.kr[1], wjseok@kisti.re.kr[2]

요 약

오늘날 전세계적으로 네트워크의 규모가 커지고 네트워크를 활용한 실험의 중요도가 높아짐에 따라 안정적이며, 고성능의 컴퓨팅 네트워크 환경이 요구되고 있다. 하지만 그러한 실험 환경을 개인 또는 소규모 연구진이 구축하기에는 여러가지 측면에서 제약사항이 발생하게 된다. 이러한 제약사항을 해결하기 위하여 세계의 다양한 연구 기관에서 연구자원 플랫폼을 구축하였으며, 대규모 실험 환경을 위하여 각 연구자원 플랫폼들을 연동하는 연구가 진행이 되었다. 하지만 현재 연구자원 플랫폼의 연동은 사용자의 요구에 맞춰 네트워크 환경을 동적으로 설정하지 못하는 한계점이 여전히 존재한다. 따라서 본 논문에서는 다수의 연구자원 플랫폼과 SDN 기술을 연동하여 연구자에게 동적인 네트워크 자원을 제공할 수 있는 시스템의 구조를 제안한다.

Keyword : Future Internet, Testbeds Federation, SFA, SDN

1. 서론

오늘날 전세계적으로 네트워크의 규모가 커지고 네트워크를 활용한 실험의 중요도가 높아짐에 따라 안정적이며 고성능의 컴퓨팅 네트워크 환경이 요구되고 있다. 하지만 그러한 환경을 개인 또는 소규모 연구진에서 구축하기에는 비용과 공간적인 측면에서 많은 제약사항이 발생하게 된다. 또한 타 연구진과의 공동연구를 진행함에 있어서 지리적인 차이로 인하여 연구를 원활하게 진행하지 못하는 어려움 역시 발생한다.

이러한 제약사항들을 해결하기 위하여 세계의 다양한 연구 기관에서 원격 접속이 가능한 실험 환경을 구축하여 연구자들에게 제공을 하고 있다. 그 대표적인 테스트베드로는 유럽에서 운영중인 FIRE[1, 2](Future Internet Research and Experimentation), PlanetLab[3, 4]과 미국에서 운영중인 GENI[5, 6](Global Environment for Network Innovations), Emulab[7] 등이 있으며, 국내에서는 KISTI 의 KreonetEmulab[8]이 있다. 이렇게 다양한 기관에서 연구자원 플랫폼을 운영하고 있지만 대규모의 실험 환경을 구축하여 실험을 진행하지 못하는 한계가 있어 각 연구자원 플랫폼들의 연동 필요성이 증가하고 있다.

미래인터넷 테스트베드들의 연동을 위한 연구도 활발히 진행이 되었는데 서로 다른 연구자원 플랫폼간 커뮤니케이션의 괴리를 극복하기 위해 표준연동규약 SFA[9](Slice-based Federation Architecture)를 정의하여 서로 다른 테스트베드들의 컴퓨팅 자원을 Slice 단위로 사용할 수 있다. 대표적으로 프랑스 Sorbonne 대학교의 UPMC(the Universite Pierre et Marie Curie)의 연구지인 UPMC 에서 개발하여 서비스 하고 있는 MySlice[10]가 있다. 하지만 현재의 연구자원 플랫폼의 연동은 각 연구자원 플랫폼이 운영중인 다수의 컴퓨팅 자원을 빌려 제공하는 것은 가능 하지만, 네트워크를 사용자의 요구에 맞게 설정하지 못하여 동적인 실험 환경을 구축하지 못한다.

따라서 본 논문에서는 연구자원 플랫폼 연동시스템의 컴퓨팅 자원을 이용하여 사용자의 요구에 맞춰 동적인 네트워크 환경을 구축하는 방법과 구조를 제안한다. 미래 인터넷 연구자원 플랫폼의 동적인 네트워크 자원 연동을 위해서는 SDN[11](Software Defined Networking)기술과의 연동이 필수적이다.

본 논문의 2 장에서는 연구자원 플랫폼과 표준연동규약 SFA 및 연구자원 플랫폼 연동 시스템에 대하여 기술하며, 3 장에서는 현재 연구자원 플랫폼 분야의 문제점을 정의한다. 또한 4 장에서는 본 논문에서 제안하는 SDN 및 SFA 기반 연구자원 플랫폼 연동 시스템의 구조에 대하여 기술하며, 마지막으로 5 장에서는 결론과 향후 연구 계획에 대하여 기술한다.

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단(No.2015R1D1A3A01018057) 및 한국과학기술정보연구원의 "첨단연구망기반 협업플랫폼 서비스 및 글로벌 연동"과제(K-16-L01-C02-S03)의 지원을 받았음.

2. 관련연구

본 장에서는 연구자원 플랫폼의 네트워크 자원을 연동하기 이전에 활발히 진행되어온 연구 및 실제 연구자들에게 서비스를 제공하고 있는 시스템에 대하여 기술한다. 연구자들에게 안정적인 실험 환경을 제공하기 위하여 세계의 다양한 연구기관에서는 원격 접속이 가능한 연구자원 플랫폼을 구축하여 연구자들에게 제공하였으며, 대규모 실험 환경의 수요에 의해 각 연구자원 플랫폼들 간의 연동을 위해 표준연동규약 SFA 를 정의하여 연구자원 플랫폼들의 연동을 이루었다. 이에 대한 내용은 이어서 기술한다.

2.1 연구자원 플랫폼

소규모 연구진에서 대규모의 컴퓨팅 실험환경을 구축하기에는 비용적, 공간적인 측면에서 많은 제약사항이 발생한다. 때문에 세계의 다양한 연구기관에서는 연구자원 플랫폼을 구축하여 연구자들에게 연구자원 이용 권한을 주어 컴퓨팅 자원을 제공하고 있다.

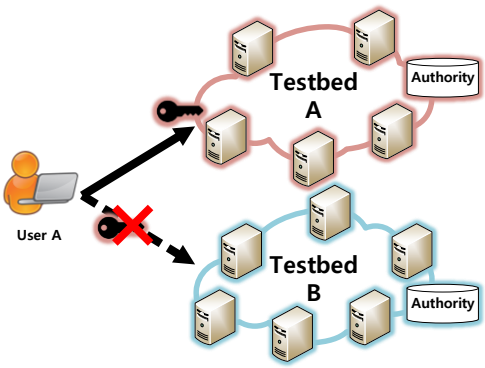


그림 1. 연구자원 플랫폼의 구조

그림 1 은 현재 다양하게 구축되어 있는 연구자원 플랫폼의 가장 기본적인 구조로 각 연구자원 플랫폼은 독립적으로 운영이 되며, 사용자는 이용하고자 하는 연구자원 플랫폼에 계정을 등록하여 권한을 받은 후에 연구자원을 이용할 수 있다. 만약 계정을 등록하지 않은 연구자원 플랫폼의 컴퓨팅 자원을 이용하기 위해서는 새로 계정을 생성해야 하며, 서로 다른 연구자원 플랫폼은 사용자의 계정을 독립적으로 관리하기 때문에 연동은 이루어질 수 없다. 또한 각 연구자원 플랫폼의 컴퓨팅 자원을 이용하기 위해서는 별도의 프로그램을 이용해야 하므로 접근이 용이하지 않다.

미래 인터넷 연구자원 플랫폼에 관한 연구는 미국과 유럽이 주축이 되어 연구를 진행하고 있는데, 미국 영역의 연구자원 플랫폼으로 Emulab[7]과 GENI[5, 6]등이 있으며, 유럽 영역의 연구자원 플랫폼으로는 FIRE[1, 2]와 PlanetLab[3, 4]등이 있다. Emulab 은 미국 Utah 대학에서 개발된 연구자원 플레임워크로 연구자에게 시스템을 개발, 디버그와 평가를 할 수 있는 광범위한 환경을 사용자가 요구하는 네트워크 구조로 구성하여 제공하는 연구자원 플랫폼이다. Emulab 은 PlanetLab 과 비교되는 대표적인 미래인터넷 연구자원 플랫폼으로 사용자가 요구하는 가상의 네트워크 토폴로지를 생성하는 것이 가능하여 현재까지도 Emulab 을 이용한 연구가 활발히 진행중에 있다. 그리고 GENI 는 대표적인 미래인터넷 연구자원 플랫폼 프로젝트로 Emulab 을 확장하여 망 가상화를 통해 응용 계층 및 물리/제어계층 실험이 가능하다. 또한 FIRE 는 유럽의 미래 인터넷 관련 연구를 지원하는 프로젝트인 FP7[12](Future Internet in Framework Programme 7) 내에서 미래인터넷을 위한 기술개발 및 다양한 실험 인프라를 구축하는 프로젝트로 유럽의 다양한 국가에서 운영하고 있는 연구용 컴퓨팅 자원을 통합, 연동하는 연구를 진행중에 있다.

2.2 SFA 및 SFA 기반 연구자원 연동 시스템

세계의 다양한 연구기관에서 제공하는 연구자원 플랫폼은 수 많은 연구자들의 실험 환경 구축에 대한 부담을 어느정도 해소 시켜 주었지만 대규모의 실험을 진행하는 데에는 여전히 한계점이 존재하기 때문에 이를 해결하기 위해서 대부분 Stand-alone 형태로 서비스를 제공하는 연구자원 플랫폼 연동의 필요성이 증대되었다. 서로 다른 기관에서 운영하는 연구자원 플랫폼은 각각 다양한 구조와 기능을 가지기 때문에 효과적인 연동 방안이 필수적인데, 그 방안으로 표준연동규약 SFA[9]가 정의되었다.

SFA 는 Princeton 대학의 Larry Peterson 을 주축으로 미국영역의 GENI 와 유럽영역의 FIRE 가 함께 정의하고 개발하였다. SFA 는 연동하고자 하는 연구자원 플랫폼들의 컴퓨팅 자원을 추상화 하여 연구자들에게 제공될 수 있도록 하는 역할을 하는데 이때 사용자는 다수의 컴퓨팅 자원을 Slice 단위로 묶어 제공 받을 수 있다.

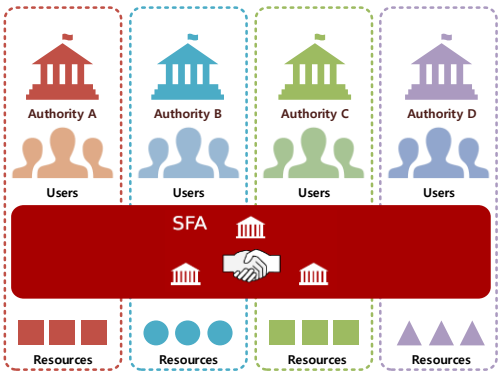


그림 2. SFA 의 역할

그림 2 는 다수의 연구자원 플랫폼에서 SFA 의 역할을 나타낸 것으로 SFA 는 서로 다른 기관에

서 구축한 연구자원 플랫폼들 간의 괴리를 한가지로 통일시켜 연동이 가능하게 해준다. 연동되지 않은 연구자원 플랫폼들은 각자의 정책과 Database 로 사용자를 관리하기 때문에 사용자는 다른 기관의 연구자원 플랫폼에 접근하기 위해서는 새로운 계정을 생성하는 것이 불가피 하다. 또한 각 연구자원 플랫폼이 소유하고 있는 자원의 종류 및 이들과 통신하는 방법이 서로 상이하기 때문에 이들을 통합해줄 수 있는 하나의 규약이 필요하며, 이를 극복하기 위한 방법이 표준연동규약 SFA 이다.

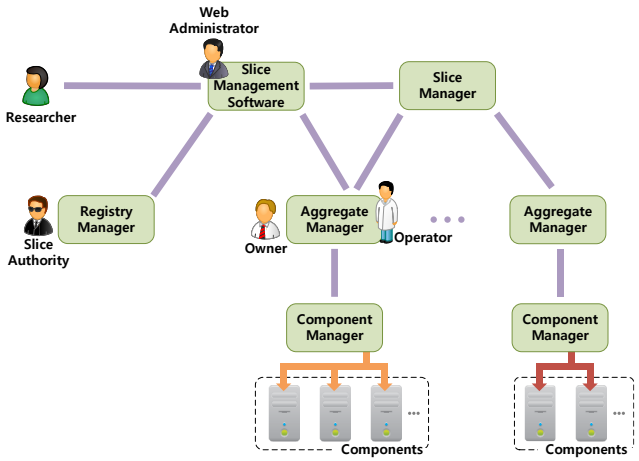


그림 3. SFA 기반 연구자원 플랫폼 연동구조

그림 3 은 SFA 를 통하여 연동을 한 연구자원 플랫폼의 구조를 나타낸 것으로 SFA 기반 연구자원 플랫폼 연동 시스템은 크게 네 가지로 구성요소를 나눌 수가 있는데 CM(Component Manager), AM(Aggregate Manager), SM(Slice Manager) 그리고 Registry 로 나뉜다. CM 은 연구자원 플랫폼의 HW 자원 정보를 수집, 분리 및 Sliver 기반으로 제어하며, 자원에 관련된 정보를 AM 에 전달한다. AM 은 다수의 CM 을 관리하며, 각 CM 으로부터 Sliver 로 정의된 인프라 자원을 수집한다. 또한 전달받은 Sliver 를 사용자 조건 별로 분리하여 생성한 Slice 를 SM 에 전달한다. SM 은 AM 에서 생성된 Slice 를 초기화 및 공급 제어 등의 관리를 하며 웹 UI 를 통하여 Slice 모니터링 환경을 제공한다. 마지막으로 Registry 는 사용자 계정 및 인증 정보인 Certificate 와 Slice 생성 및 관리 정보인 Credential, Slice 에 대한 정보 역시 DB 또는 파일의 형태로 유지하고 해당 정보를 SM 으로 전달하여 Slice 제어가 가능하게 한다.

그림 4 는 SFA 기반의 연구자원 연동 시스템의 대표적인 예시와 연동 방법을 나타낸 것으로 SFI 는 유럽 영역에서 사용되는 연구자원 플랫폼 연동 시스템이다. MySlice[10]는 프랑스 Sorbonne 대학 UPMC 의 연구팀인 OneLab 에서 개발한 연구자원 플랫폼 연동 시스템으로 CLI 기반으로 사용하는 SFI 를 발전시켜 웹 UI 로 사용자 입장에서 쉽게 이용 할 수 있다. 또한 OMNI 는 미국 영역에서 사용하는 CLI 기반 연구자원 연동 시스템으로 이를 발전 시킨 것이 Flack 이다. 각 연구자원 플랫폼은 고유의 문법(RSpec)으로 사용자의 요구를 전달하거나 연구자원의 현황 정보를 전달 하지만 SFA API 는 이를 통일된 포맷으로 변환시켜 연동이 가능케 한다. 또한 유럽과 미국 두 영역의 연구자원 플랫폼 연동 시스템은 모두 SFA 를 통하여 연동을 하기 때문에 한가지 시스템으로 서로 다른 영역의 연구자원 플랫폼 연동이 가능하다.

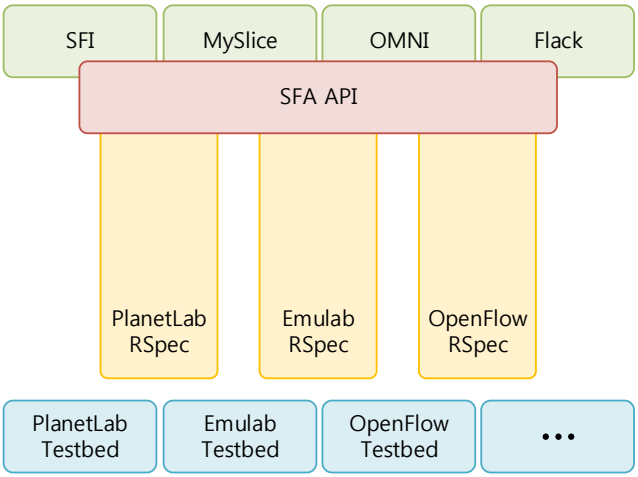


그림 4. SFA 기반 연구자원 플랫폼 연동 시스템

3. 문제 정의

연구자원 플랫폼을 연동함에 있어서 가장 중요한 요구사항은 지리적으로 분산되어 구축된 연구자원 플랫폼의 자원을 자신의 자원처럼 사용할 수 있어야 하는 점이다. 이에 대해서 사용자의 위치에 따른 제약이 없어야 하며, 사용자가 요구하는 실험환경 규모에 제약이 없어야 한다. 또한 사용자가 요구하는 실험환경의 구조적 제약이 없어야 한다. 이와 같은 요구사항을 충족시키기 위하여 원격지에서 접속하여 자신의 컴퓨터처럼 사용가능한 연구자원 플랫폼, 다양한 연구자원 플랫폼을 통합하여 대규모 실험을 진행 할 수 있도록 하는 연구자원 플랫폼 연동 시스템 등 연구자들에게 양질의 실험 환경을 제공하기 위한 다양한 연구가 진행되어 왔지만, 여전히 모든 연구자들의 요구하는 실험환경을 제공하지 못하는 한계점이 몇가지 존재 한다.

첫째, 현재의 연구자원 플랫폼 연동 시스템을 통해서는 통합된 연구환경을 생성할 수 없다. 다수의 컴퓨팅 자원을 Slice 로 생성할 수는 있지만 그들은 네트워크 상에서 독립적인 자원으로 인식이 되기 때문에 다수의 컴퓨팅 자원들을 이용한 네트워크 실험을 진행하지 못한다. 사용자가 다수의

컴퓨팅 자원을 동시에 요구하는 것은 그 만한 컴퓨팅 파워가 필요하다는 것이며, 독립적인 자원 하나가 그에 합당한 작업을 수행 하지 못한다면 연구자원 플랫폼을 연동하는 의미가 없다.

둘째, 사용자의 요구에 맞는 네트워크 환경을 구성하지 못한다. 사용자는 네트워크 실험을 진행할 때 필요한 토폴로지를 구성하여 실제로 구현하기 힘들기 때문에 가상으로 구현해야만 한다. 이를 위해서는 각 연구자원 플랫폼의 스위치의 플로우 테이블 설정을 변경하여 네트워크 트래픽의 경로를 설정할 수 있어야 한다.

셋째, 실험을 진행함에 있어서 높은 QoS 를 보장하지 못한다. 현재의 서로 다른 연구자원 플랫폼은 서로 직접적인 네트워크 선로가 연결이 되어 있지 않기 때문에 안정적인 QoS 를 보장하지 못함은 물론 정확한 실험 결과를 도출하기 어렵다. 이러한 문제를 해결 하기 위해서는 각 연구자원 플랫폼의 스위치 장비를 통합하여 제어할 수 있는 기술이 필요하다.

마지막으로 네트워크에 문제 발생 시 발빠른 대응이 어렵다. 원거리로 떨어져 있는 연구자원 플랫폼의 자원 상태를 모니터링하는 것은 쉬운 일이 아니며, 문제 발생 시 이를 원격으로 복구하는 것 또한 불가능에 가깝다.

따라서 미래인터넷 연구자원 플랫폼의 네트워크 자원의 연동을 위해서는 실제 구성되어 있는 네트워크 구조와 관계 없이 사용자의 요구에 맞춰 동적으로 트래픽의 흐름을 변경하여 가상의 네트워크 구조를 생성해야 한다. 이를 위해서는 라우터, 스위치 등 네트워크 장비의 역할을 가상화 하여 소프트웨어로 정의할 수 있는 SDN[11]기술과의 연동이 필수적이다.

4. 연구자원 플랫폼의 네트워크 자원 연동 구조

본 장에서는 연구자원 플랫폼의 컴퓨팅 자원은 물론 네트워크 자원까지 연동할 수 있는 구조에 대하여 설명 한다. 컴퓨팅 자원의 연동은 원격지에서 다수의 CPU, RAM, Storage 등 컴퓨팅 자원을 하나의 서버랙에 있는 것처럼 사용이 가능 하도록 하는 것으로 사용자의 위치와 실험 환경의 규모에 대한 제약을 해결 할 수 있다. 하지만 실험 환경을 사용자의 요구에 따라 설정하지 못하는 구조적 제약이 존재한다. 반면에 네트워크 자원 연동이란 사용자의 요구에 따른 동적 네트워크 구조 설정이 가능 하도록 하여 다양한 네트워크 구조를 갖는 실험 환경을 구성할 수 있다.

그림 5 는 연구자원 플랫폼과 SDN 기술을 연동하는 구조를 나타낸 것으로 기존 컴퓨팅 자원의 연동이 완료된 연구자원 플랫폼 연동 시스템과 SDN 컨트롤러, 각 스위치가 연결된 형태이다. 이 때, SDN 컨트롤러는 연구자원 연동 시스템 운영 주체 측에서 관리 해야 하며, 연동이 약속된 모든 연구자원 플랫폼과 연결이 되어 있어야 한다. 각 연구기관에서 관리를 하게 되면 각 연구자원 플랫폼 간의 명령 체계의 괴리가 생겨 연동이 불확실 또는 불안정 하기 때문이다.

이와 같이 연구자원 플랫폼의 연동이 이루어 지면 각 연구자원 플랫폼은 통상적인 인터넷 망과는 별개로 독립적인 물리망으로 직접 연결이 되기 때문에 다른 트래픽 흐름의 영향을 받지 않아 안정적인 QoS 를 보장받을 수 있다. 그리고 각 연구자원 플랫폼의 스위치 장비는 통합된 SDN 컨트롤러에 의해 Bandwidth, 플로우 테이블 등이 제어 되기 때문에 사용자가 요구하는 토폴로지를 SDN 컨트롤러에서 입력 받아 Slice 의 네트워크 환경을 구성할 수 있다. 또한 네트워크 장비에 문제가 발생하면, SDN 컨트롤러에서 동적으로 새로운 네트워크 경로를 설정 할 수 있기 때문에 네트워크 문제에 따른 실험 진행이 불가능한 상황에 대하여 발빠른 대응이 가능하다.

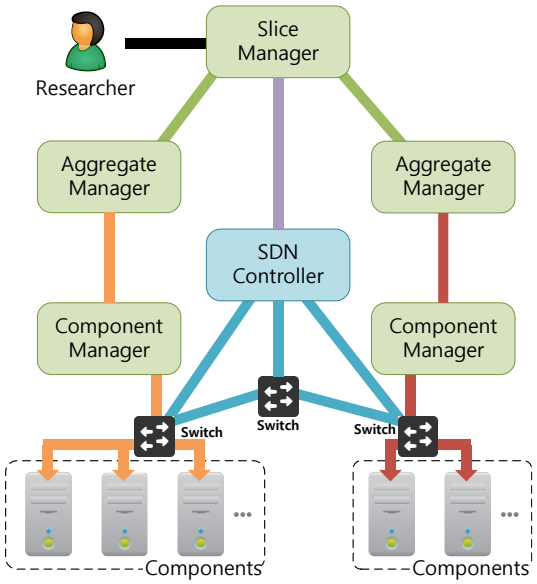


그림 5. 연구자원 플랫폼과 SDN 기술 연동 구조

본 논문에서 제안하는 방법은 기존 연구자원 플랫폼 연동 시스템과 마찬가지로 표준연동규약 SFA 를 통하여 다양한 연구자원 플랫폼의 컴퓨팅 자원을 Slice 로써 생성한다. 이어서 사용자의 요구에 맞는 네트워크 토폴로지를 입력 받아 SDN 컨트롤러의 플로우 테이블을 수정하여 Slice 에 포함 되어있는 컴퓨팅 자원들 간의 트래픽 흐름을 제어하여 가상의 토폴로지를 생성함으로 네트워크 자원 연동을 이룬다. 이때 컴퓨팅 자원을 연동하는 것과 같이 네트워크 자원을 연동함에 있어서 SFA 를 통해 사용자의 요구를 전달하는 것이 아닌 SDN 컨트롤러에 명령을 전달하는 별도의 API 가 필요하다. 대표적으로 Restful API 는 ONOS[13] SDN 컨트롤러의 Northbound 와 연동이 가능하며, 실제로 연구자원 플랫폼 연동 시스템을 위의 방법을 통하여 구축할 계획이다.

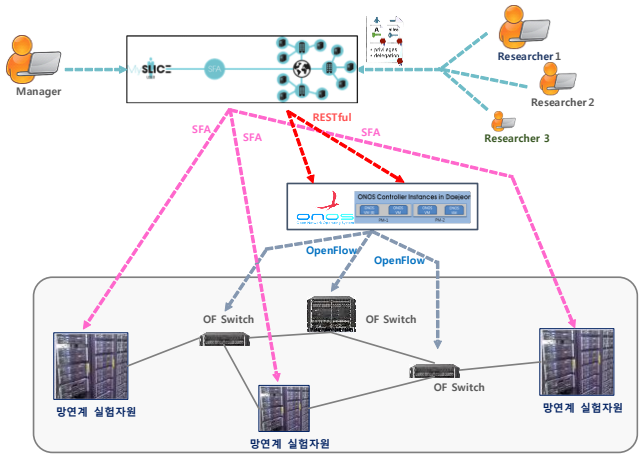


그림 6. 계획중인 연구자원 플랫폼 연동 시스템 구성도

그림 6 은 실제로 구축을 계획하고 있는 연구자원 플랫폼 연동 시스템의 구성도로 대표적인 오픈소스 연구자원 플랫폼 연동 시스템인 MySlice 와 ONOS SDN 컨트롤러의 연동 방안을 연구 중에 있다. ONOS SDN 컨트롤러는 Northbound API 를 통해 외부의 인터페이스와 통신이 가능하며, Restful API 는 ONOS 에 명령을 전달하는 기능을 지원한다. 따라서 향후 Restful API 를 이용하여 MySlice 와 ONOS SDN 컨트롤러를 연결해주는 인터페이스를 개발하여 연구자원 플랫폼의 완벽한 연동을 계획 중 이다.

또한 웹 UI 를 통하여 서비스를 제공하는 MySlice 를 활용하여 연동 중인 연구자원 플랫폼의 컴퓨팅 자원은 물론 네트워크 장비의 현황을 실시간으로 제공할 수 있어야 하며, 이를 토대로 사용자는 실험 환경을 손쉽게 설정 할 수 있어야 한다. 이를 위한 MySlice 내부에서 동작하는 실험 환경 설정 플러그인 개발을 계획 중이다.

5. 결론 및 향후 연구

본 논문에서는 연구자에게 실험환경을 제공하는 현재의 연구자원 플랫폼의 한계점을 극복하기 위하여 연구자원 플랫폼 연동 시스템과 SDN 기술을 연동하는 구조를 제안하였다. 현재의 연구자원 플랫폼은 대규모 실험을 진행하는데 있어서 비용적, 공간적인 제약사항이 많으며, 타 연구기관의 사용자에게 폐쇄적인 단점이 있다. 이를 해결하기 위해 SFA 표준연동규약을 통하여 각 연구자원 플랫폼 간의 통신 문법과 사용자 인증 체계를 통일시켰다. 하지만 사용자의 요구에 따른 다양한 네트워크 실험 환경을 구성하지 못하는 구조적 한계점이 존재하며, 실험결과에 큰 영향을 미치는 안정적인 QoS 를 보장하지 못하기 때문에 이를 극복하기 위해 연구자원 플랫폼 연동 시스템과 하드웨어로 구축되어 있는 네트워크를 소프트웨어로 정의하는 SDN 기술의 연동이 필요하다. 연구자원 플랫폼과 SDN 기술은 연동하기 위한 방법으로 SFA 표준연동규약을 통하여 빌린 컴퓨팅 자원을 SDN 컨트롤러를 이용하여 네트워크 환경을 설정하는 구조를 제안하였다.

향후 본 논문에서 제안하는 구조를 실제 망에 적용시켜 그림 6 과 같이 연구자원 플랫폼 연동 시스템인 MySlice 와 SDN 컨트롤러인 ONOS 를 연동하는 연구를 진행 하여 연동 가능성을 검증할 계획이다. 이를 위해 ONOS SDN 컨트롤러와 MySlice 간 명령을 주고 받는 인터페이스를 Restful API 를 통해 구현해야 한다.

참고 문헌

[1] Gavras, Anastasius, et al. "Future internet research and experimentation: the FIRE initiative." *ACM SIGCOMM Computer Communication Review* 37.3 (2007): 89-92.

[2] Schwerdel, Dennis, et al. "Future Internet research and experimentation: The G-Lab approach." Computer Networks 61 (2014): 102-117.

[3] Chun, Brent, et al. "Planetlab: an overlay testbed for broad-coverage services."ACM SIGCOMM Computer Communication Review 33.3 (2003): 3-12.

[4] Lo, Shihmin, Ellen Zegura, and Marwan Fayed. "Virtual network migration on real infrastructure: A planetlab case study." Networking Conference, 2014 IFIP. IEEE, 2014.

[5] Elliott, Chip. "GENI-global environment for network innovations." LCN. 2008.

[6] Berman, Mark, et al. "GENI: A federated testbed for innovative network experiments." Computer Networks 61 (2014): 5-23.

[7] Hibler, Mike, et al. "Large-scale Virtualization in the Emulab Network Testbed."USENIX Annual Technical Conference. 2008.

[8] Lee, Minsun, Woojin Seok, and Kwan-Jong Yoo. "Challenges and Opportunities in the KREONET-Emulab Network Testbed." 한국콘텐츠학회 ICCC 논문집 (2014): 213-214.

[9] Fdida, Serge, Timur Friedman, and Thierry Parmentelat. "OneLab: An open federated facility for experimentally driven future internet research." New Network Architectures. Springer Berlin Heidelberg, 2010. 141-152.

[10] Baron, Loic, et al. "OneLab: Major computer networking testbeds open to the IEEE INFOCOM community." Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on. IEEE, 2015.

[11] Nunes, Bruno AA, et al. "A survey of software-defined networking: Past, present, and future of programmable networks." Communications Surveys & Tutorials, IEEE 16.3 (2014): 1617-1634.

[12] Stuckmann, Peter, and Rainer Zimmermann. "Toward ubiquitous and unlimited-capacity communication networks: European research in framework programme 7." Communications Magazine, IEEE 45.5 (2007): 148-157.

[13] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS."Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.

차량 클라우드 보안 요구사항 및 연구 이슈

김명수 º, 장인선, 주석진, 백상헌

고려대학교 전기전자공학과

{myeongsukim, zerantoss, choo0128, shpack}@korea.ac.kr

요 약

최근 ICT 기술 발전과 함께 자동차 산업에도 ICT 기술이 융합되어 다양한 형태의 통신 기반 서비스를 제공할 수 있는 커넥티드 카 (Connected Car) 기술이 주목받고 있다. 하지만 차량이 소유하고 있는 한정된 자원 제약으로 인해 고품질의 서비스를 수행하는데 한계점이 존재한다. 이를 해결하기 위해 고성능의 컴퓨팅 및 통신 자원을 소유하고 있는 이웃 차량들을 클라우드 형태로 그룹화하는 차량 클라우드 (Vehicular Cloud) 기술이 제안되었다. 하지만 차량 클라우드에서의 다양한 보안 위협으로 인해 차량뿐만 아니라 운전자에게도 영향을 미칠 수 있다. 따라서 본 논문에서는 차량 클라우드에서의 보안 위협과 요구사항에 대해서 살펴보고 이에 따른 향후 연구 이슈를 기술한다.

1. 서론

최근 ICT 기술 발전으로 인한 파급효과가 자동차 산업에까지 영향을 미치면서 다양한 형태의 통신 기반 서비스를 지원하는 커넥티드 카 (Connected Car) 기술이 주목을 받고 있다. 커넥티드 카 기술은 차량과 차량간 (V2V: Vehicle-to-Vehicle) 통신 혹은 차량과 인프라간 (V2X: Vehicle-to-Infrastructure) 통신을 통해 다양한 서비스를 제공할 수 있다. 하지만 차량이 소유하고 있는 한정된 자원 제약 (예, 낮은 프로세싱 성능, 작은 메모리 크기 등) 과 높은 이동성으로 인한 통신 성능 저하로 인해 음성인식, 스트리밍, 대용량 컨텐츠 다운로드, 완전 자율주행 등과 같은 고품질의 서비스를 수행하는데 많은 한계점이 존재한다.

이러한 한계점을 해결하기 위해 차량들은 고성능의 컴퓨팅 및 통신 자원을 소유하고 있는 이웃 차량들과 협력적인 네트워킹을 통해 차량 클라우드 (Vehicular Cloud) [1-3]를 형성한다. 즉, 차량 클라우드 기술은 높은 컴퓨팅 자원과 통신 자원을 소유하고 있는 차량들을 클라우드 형태로 그룹화 하여 가상의 클라우드 컴퓨팅 인프라로 활용하는 기술이다. 차량 클라우드 기술을 통해 이웃 차량의 컴퓨팅 자원을 제공하는 Computing as a Service (CaaS), 이웃 차량의 저장 공간을 공유하여 활용하는 Storage as a Service (SaaS), 기지국에 근접한 차량이 멀티 홉 무선 통신을 통해 통신 영역 밖의 차량들에게 네트워크 연결성을 보장하는 Network as a Service (NaaS) 가 가능하다. 다음 그림 1 은 중앙 집중형 차량 클라우드 구조를 나타낸다 [4]. 차량 클라우드 구조에서 중앙의 차량 클라우드 컨트롤러는 차량들의 위치 정보 및 가용 가능한 자원에 대한 정보를 유지하며,

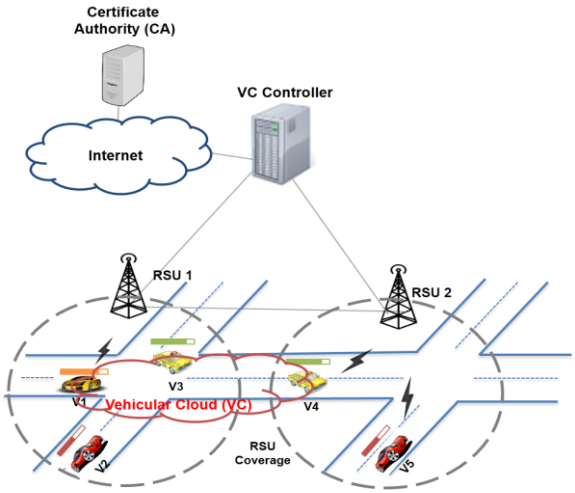


그림 1. 차량 클라우드 구조

이를 기반으로 차량 클라우드를 형성 및 관리하는 역할을 수행한다. 이와 유사하게 차량 클라우드를 활용한 연구가 최근 활발히 이루어지고 있다. 하지만 차량 클라우드에서 다양한 보안 위협으로 차량뿐만 아니라 운전자에게도 영향을 미칠 수 있기 때문에 보안에 관한 연구가 필요하다. 따라서 본 논문에서는 차량 클라우드에서의 위협과 보안 요구사항에 대해서 살펴보고 이에 따른 향후 연구 이슈에 대해 기술한다. 본 논문의 구성은 다음과 같다. 2 장에서는 차량 네트워크 환경인 VANET (Vehicular Ad hoc Network) 에서 보안 관련 기존 연구에 대해서 살펴본다. 3 장에서는 차량 클라우드에서 보안 위협과 이에 따른 보안 요구사항에 대해서 살펴보고, 4 장 에서는 차량 클라우드의 보안 연구 이슈에 대해서 기술한다. 마지막으로 5 장에서 결론을 맺는다.

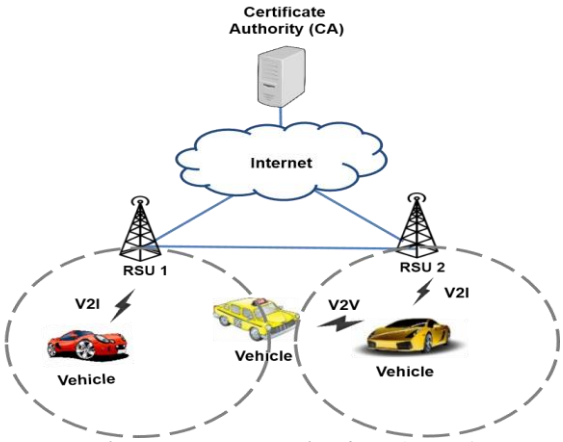


그림 2. VANET 보안 시스템 모델

2. VANET 보안 기존 연구

본 장에서는 차량 네트워크 환경인 VANET 에서 보안과 관련된 기존 연구에 대해서 기술한다. 먼저, VANET 환경에서 차량간 안전한 통신을 위한 기본적인 시스템 모델에 대해서 살펴본 뒤, 차량간 안전한 통신을 보장하기 위한 차량 인증 과정과 메시지 암호화에 대해서 자세하게 기술한다.

다음 그림 2 는 VANET 환경에서 차량간 안전한 통신을 위한 시스템 모델을 나타낸다. 시스템 모델을 살펴보면 차량, 노변 장치 (RSU: Road-side Unit), 인증기관 (Certificate Authority) 으로 구성되어 있다. 먼저, 차량 내부에는 차량 주행에 관련된 정보를 감지하는 센서와, 이를 제어 및 처리하기 위한 다수의 제어 유닛 (Control Unit)이 존재한다. 이러한 제어 유닛에는 대표적으로 차량의 엔진, 변속기 등의 내부 장치들의 동작을 제어하기 위한 ECU (Electronic Control Unit) 가 있다. 또한 차량 내부 혹은 차량 외부와의 통신을 담당하는 CCU (Communication Control Unit) 가 존재한다. 따라서 차량은 CCU 를 통해 V2V 혹은 V2I 통신을 수행한다. 그 다음으로 RSU 는 도로 주변에 설치되어 차량과 통신 가능한 장치이다. 또한 RSU 는 인터넷 망 혹은 차량 네트워크와 연동되어 차량에게 다양한 서비스를 제공할 수 있다. 마지막으로 인증기관은 차량의 인증을 위해 차량에게 인증서 (Certificate) 를 발급해주는 역할을 수행한다.

다음으로 시스템 모델을 기반한 VANET 환경에서 차량간 안전한 통신을 위한 인증과 메시지 암호화에 대해서 기술한다. 그림 3 은 차량간 인증을 수행하는 과정을 나타내며, 차량 V1 은 차량 V2 에게 메시지를 전송하는 상황을 가정한다. 먼저, 차량들은 차량간 인증을 위해 신뢰 가능한 제 3 자의 인증기관으로부터 인증서와 개인키 (Private key) 를 발급받는다. 그림 3 의 1 번 과정은 차량 V1 이 인증기관으로부터 차량 V1 의 인증서와 개인키를 발급 받는 과정을 나타낸다. 인증기관으로부터 발급 받은 인증서에는 다음과 같은 정보가 담겨 있다.

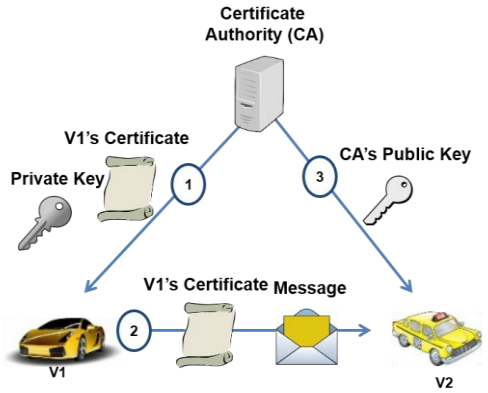


그림 3. 인증서 기반의 차량 통신 과정

- 인증서 ID
- 공개키 (Public Key)
- 인증서 유효기간
- 인증기관의 디지털 서명 (Digital Signature)

인증기관의 디지털 서명은 인증기관이 소유하고 있는 개인키로 디지털 서명을 생성한 것을 의미한다. 다음으로 2 번 과정은 인증서를 발급 받은 차량 V1 이 차량 V2 에게 메시지를 전송하는 과정을 나타낸다. 차량 V1 은 전송하고자 하는 메시지를 차량 V1 의 개인키를 이용하여 디지털 서명을 생성한다. 그리고 차량 V1 은 전송하고자 하는 메시지와, 디지털 서명, 인증서를 수신 차량 V2 에게 전송한다. 만약, 차량 V1 이 보다 안전하게 메시지를 전송하고자 한다면 차량은 메시지 암호화를 수행하여 전송한다. 일반적으로 차량 통신에서 메시지 암호화는 대칭키 암호화 알고리즘을 사용한다. 또한 차량 통신에서 사용되는 대칭키 암호화 알고리즘은 128 비트의 블록 암호를 사용하며, 대표적인 예로는 AES, ARIA, LEA 등이 있다. 반면, 대칭키 암호화 알고리즘 연산에 사용되는 암호화 키는 사전에 공개키 암호화 알고리즘을 통해 암호화 한 뒤 차량간에 공유한다.

마지막으로 3 번 과정은 차량 V2 가 차량 V1 으으부터 전송 받은 메시지를 검증하는 과정을 나타낸다. 차량 V2 는 전달 받은 메시지를 검증하기 위해 인증기관에게 인증기관의 공개키를 요청한다. 그리고 차량 V2 는 인증기관의 공개키를 이용하여 차량 V1 이 전송한 인증서 내에 저장되어 있는 인증기관의 디지털 서명을 검증한다. 이를 통해, 차량 V1 의 인증서의 유효성을 검증할 수 있다. 또한 차량 V1 의 인증서로부터 획득한 차량 V1 의 공개키를 이용하여 차량 V1 의 디지털 서명을 검증한다. 이를 통해, 차량 V2 는 전송 받은 메시지가 차량 V1 으로부터 전송 되었는지 검증할 수 있으며, 메시지의 위조 혹은 변조 여부를 검증할 수 있다. 만약. 차량 V1 이 메시지를 암호화 하여 전송하였다면, 사전에 차량간 공유하고 있는 대칭키를 이용하여 메시지를 복호화 한다.

3. 차량 클라우드 보안 위협 및 요구사항

본 장에서는 차량 클라우드에서 발생할 수 있는 보안 위협에 대해서 살펴보고, 이에 따른 차량 클라우드 보안 요구사항에 대해서 기술한다.

먼저, 차량 클라우드에서 발생할 수 있는 보안 위협에 대해서 살펴본다. 다음 그림 4 는 차량 클라우드에서 발생할 수 있는 보안 위협을 나타낸다. 그림과 같이 차량 클라우드 환경에서 발생할 수 있는 공격은 물리적으로 외부에서 전파를 교란시키는 공격과 인터넷 망 혹은 차량 네트워크를 통한 공격이 존재한다. 먼저, 외부에서 물리적으로 전파를 교란시키는 공격에는 전파 방해 (Jamming)가 있다. 즉, 전파 방해 공격이란 일정 구역 내에서 차량 혹은 RSU 가 정상적인 통신을 수행할 수 없도록 장애를 일으키는 전파를 발생시키는 공격이다. 이는 물리적 계층 혹은 네트워크 계층에서의 공격을 의미한다.

다음으로 전파를 통해 교란시키는 공격이 아닌 인터넷 망 혹은 차량 네트워크를 통해 발생할 수 있는 보안 위협에 대해서 살펴보면 다음과 같다.

첫 번째 보안 위협은 차량 인증과 메시지 무결성 (Integrity) 에 대한 공격이다. 차량 인증이란 차량이 부여 받은 ID 의 정당한 소유자임을 밝히는 것을 의미하며, 메시지 무결성이란 송수신되는 메시지가 위조 혹은 변조 되지 않음을 의미한다. 차량 인증과 메시지 무결성에 대한 공격을 살펴보면 다음과 같다.

- 차량 위장 (Impersonation): 공격자는 차량 네트워크 상의 다른 차량으로 위장할 수 있다. 즉, 다른 차량의 인증서 혹은 임시의 ID (예, Pseudonym) 정보를 획득하여 해당 차량으로 위장할 수 있다. 이와 같이 차량이 위장을 하여 특정 차량에게 전송될 메시지를 수신하거나, 특정 차량이 전송해야 하는 메시지를 공격자가 거짓으로 전송하는 것이 가능하다.

- Sybil 공격: 공격자는 다른 차량의 차량의 인증서 혹은 임시의 ID 정보를 바탕으로 Sybil 공격을 수행할 수 있다. Sybil 공격이란 하나의 공격자가 다수의 차량 인증 정보를 도용하여 공격하는 방식이다. 그리고 공격자는 다른 차량의 인증 정보를 바탕으로 악의적인 공격을 수행하여 인증센터, 차량 클라우드 컨트롤러, RSU, 다른 차량들로부터 다양한 서비스를 제공 받지 못하게 한다.

- 차량의 정보 변조 공격: 공격자는 차량 자신의 속도, 방향, 위치 (GPS)와 같은 이동성 정보를 조작하여 전송한다. 차량의 이동성 정보를 조작하여 전송한다면 최적의 차량 클라우드를 형성하는데 어려움이 존재한다.

두 번째 보안 위협은 기밀성 (Confidentiality) 에 대한 공격이다. 기밀성이란 차량 통신에서 송수신되는 메시지가 인가되지 않은 차량에 대해서 비밀성을 유지하는 것을 의미한다. 따라서 인가되지 않은 차량이 메시지를 수신하더라도 해석할 수 없어야 한다. 기밀성에 대한 공격을 살펴보면 다음과 같다.

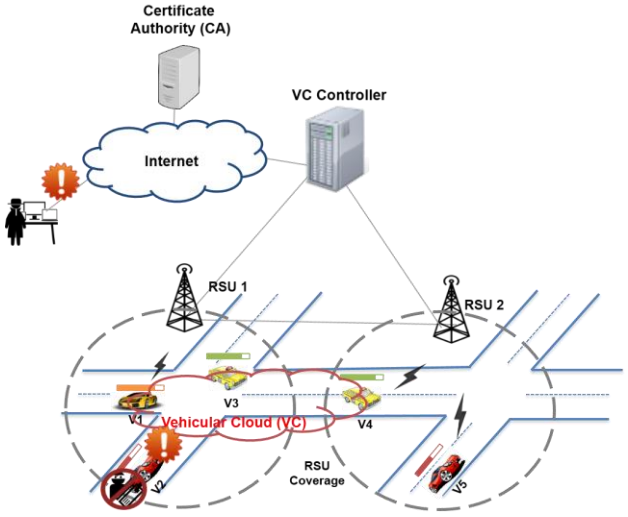


그림 4. 차량 클라우드 보안 위협

- 차량 통신 메시지 도청: 공격자는 차량 근접 거리 혹은 도로 상에서 RSU 의 통신 메시지를 도청하는 공격이다. 이와 같이 도청한 메시지를 분석하여 다양한 정보를 습득할 수 있다.

세 번째 보안 위협으로는 부인봉쇄 (Non-repudiation) 에 대한 공격이다. 일반적으로 차량 통신에서 부인 봉쇄를 제공하는 보안 메커니즘은 디지털 서명이다. 따라서 부인 봉쇄에 대한 공격은 디지털 서명에 연관된 공격 이다. 즉, 디지털 서명 생성을 위한 인증서 및 개인키를 획득하여 디지털 서명을 공격을 수행한다.

네 번째 보안 위협으로는 가용성 (Availability) 에 대한 공격이다. 가용성에 대한 공격은 공격자가 통신 채널을 점유하여 차량간 메시지 송수신이 불가능하게 하는 공격이다. 가용성에 대한 공격을 살펴보면 다음과 같다.

- 분산 서비스 거부 공격 (DDoS): 분산 서비스 거부 공격은 차량 혹은 RSU 에 악성코드를 삽입하고, 다수의 차량 혹은 RSU 가 통신 채널을 점유하여 무한대의 메시지를 전송하는 공격이다. 또한 불특정 다수의 차량들이 특정 차량에게 많은 양의 데이터와 많은 연산이 필요한 메시지를 다량으로 전송하는 공격이다.

마지막 보안 위협으로는 프라이버시 (Privacy) 에 대한 공격이다. 프라이버시란 차량의 이동성 정보가 다른 개체로부터 보호받을 수 있어야 하는 것을 의미한다. 프라이버시에 대한 공격을 살펴보면 다음과 같다.

- 차량의 이동성 정보 수집 공격: 공격자가 차량 통신 메시지를 수신 및 분석하여 해당 차량에 대한 소유자를 추정하고, 차량에 대한 이동성 정보를 추적한다. 이를 통해 차량의 출발지, 경유지, 목적지와 같은 정보를 알아낼 수 있는 공격이다.

다음으로 차량 클라우드에서 보안 요구사항에 대해서 살펴본다. 차량 통신 환경에서 차량들은 서로 다른 방향 및 속도를 가지고 이동하기 때문에 차량의 위치가 급변하는 특징을 가지고 있다. 따라서 차량 클라우드를 형성하더라도 차량의 위치가 빈번하게 바뀌기 때문에 지속적으로 차량 인증을 수행할 수 있어야 하며, 차량 클라우드 내에서 구성원들 간에 안전한 메시지 송수신이 보장될 수 있어야 한다. 또한 차량 클라우드를 형성하더라도 차량 클라우드의 그룹 유지를 위해 새로운 차량이 차량 클라우드의 구성원으로 추가될 수 있으며, 차량의 높은 이동성으로 인해 차량 클라우드의 구성원 차량이 그룹을 탈퇴하는 경우가 발생할 수 있다. 이와 같이 차량들의 급변한 위치 변화에도 차량 인증과 안전한 메시지 송수신을 위한 보안 연구가 필요로 하다. 따라서 차량 클라우드에서 보안 요구사항에 대해서 살펴보면 다음과 같다.

첫 번째 보안 요구사항은 차량 인증이다. 차량 인증이란 차량 클라우드 구성원 차량들이 차량 클라우드 그룹의 구성원임을 밝힐 수 있어야 한다.

두 번째는 메시지 무결성이다. 메시지 무결성이란 차량 클라우드 구성원들간 통신에서 송수신 되는 메시지가 위조 혹은 변조 되지 않아야 하는 것을 의미한다.

세 번째는 기밀성이다. 기밀성이란 차량 클라우드 구성원들간 통신에서 송수신 되는 메시지가 인가되지 않는 차량에 대해서 비밀성을 유지하는 것을 의미한다.

네 번째는 부인 봉쇄이다. 부인 봉쇄란 차량 클라으두 그룹 내 구성원들간 통신에서 메시지를 송신한 차량은 메시지 송신 사실을 부인할 수 없어야 하는 것을 의미한다.

다섯 번째는 가용성이다. 가용성이란 차량 클라우드 그룹 내 구성원들간 통신에서 송신된 메시지는 적절한 시간 내 수신 차량에게 도착할 수 있어야 하는 것을 의미한다.

마지막으로 프라이버시이다. 프라이버시란 차량 클라우드 그룹 내 구성원들의 주행 정보를 추적할 수 없어야 하는 것을 의미한다.

4. 차량 클라우드 보안 연구 이슈

본 장에서는 차량 클라우드에서 보안 요구사항을 바탕으로 연구 이슈에 대해서 살펴본다 [5].

첫 번째 연구 이슈는 차량 클라우드 그룹내 차량 인증이다. 일반적으로 VANET 환경에서 차량들은 인증을 위해 인증기관으로부터 인증서를 발급받으며, 송신하고자 하는 메시지에 인증서를 함께 전달하여 인증을 수행한다. 하지만 차량의 개체정보가 담긴 인증서를 매번 전달하는 경우 정보가 유출될 수 있는 문제점이 존재한다. 이를 해결하기 위해 특정 그룹 내에서 인증을 위한 그룹기반의 디지털 서명 (Group-based Digital signature) 기법이 제안되었다. 하지만 디지털 서명을 위한 그룹 공개키 (Group Public Key)와 개인키를 생성 및 관리하는 효율적인 방법에 관한 연구가 필요하다.

두 번째는 차량 클라우드 그룹 내에서 인증 받은 차량들간의 기밀성을 유지해야 한다. 이를 위해 차량 클라우드 그룹 내 차량들은 비밀키 (Secrete key) 로 메시지를 암호화 하여 송수신 한다. 하지만 차량 클라우드 그룹 구성원을 위한 대칭키를 생성 및 관리하는 방법에 관한 연구가 필요하다.

마지막으로 차량 클라우드 그룹 내 차량들의 프라이버시를 보장해야 한다. 이를 위해 차량 클라우드 그룹 내 차량들은 임시의 ID (예, Pseudonym) 를 이용하고, 이를 주기적으로 변경하여 차량의 프라이버시를 보장한다. 하지만 임시의 ID 를 발급 및 관리하는 방법에 관한 연구가 필요하다.

5. 결론

본 논문에서는 차량 통신 환경에서 보안 관련 기존 연구에 대해서 살펴보았으며, 차량 클라우드에서 발생할 수 있는 보안위협과 이에 따른 보안 요구사항에 대해 살펴보았다. 또한 차량 클라우드 보안과 관련된 연구 이슈에 대해서 기술하였으며, 향후 연구에서는 차량 클라우드 환경에서 보안 요구사항을 모두 만족할 수 있는 보안 프레임워크를 제안할 예정이다.

6. Acknowledgment

본 연구는 산업통상자원부 및 한국산업기술평가관리원의 산업핵심기술개발사업의 일환으로 수행하였음 [10051306, 안전한 차량 IoT 서비스를 위한 차량 클라우드 기반의 동적 보안 프레임워크 개발]

7. 참고 문헌

[1] S. Bitam, A. Mellouk, and S. Zeadally, “VANET-Cloud: A Generic Cloud Computing Model for Vehicular Ad Hoc Networks,” *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96-102, Feb. 2015.

[2] E. Lee, E. Lee, and M. Gerla, “Vehicular Cloud Networking: Architecture and Design Principles,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 148-155, Feb. 2014.

[3] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, “A Survey on Vehicular Cloud Computing,” *Journal of Network and Computer Applications*, vol. 40, pp. 325-344, Apr. 2014.

[4] S. Choo, I. Jang, M. Kim, and S. Pack, “Software Defined Vehicular Cloud Architecture for Resource Sharing,” in *Proc. ICDCN*, Jan. 2016.

[5] F. Qu, Z. Wu, F. Wang, and W. Cho, “A Security and Privacy Review of VANETs,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, Dec. 2015.

SDN 네트워크에서 무선 랜 이동성 지원 방안

김성실°, 윤준호, 박정재, 권혁명, 석승준

경남대학교 컴퓨터공학과

tjdtlf1159@gmail.com, yjh4408@naver.com, jungjae326@gmail.com, fruits_97@naver.com, sjseok@kyungnam.ac.kr

요 약

본 논문에서는 SDN 네트워크에서 Wi-Fi 무선 랜 사용자 단말 이동 시 단절 없는 서비스를 제공하기 위한 방안을 제안한다. 기존 Wi-Fi 무선 랜에서 단말이 연결된 AP 변경하는 동안 패킷 손실과 패킷 순서 변경으로 등으로 인한 문제가 발생한다. 본 논문에서는 무선 랜 서비스 성능 향상을 위해 중앙집중식으로 네트워크를 제어할 수 있는 SDN 환경을 이용하여 Wi-Fi 단말의 이동성 지원 방안을 제안한다. 제안하는 방안에서는 SDN 컨트롤러는 단말의 이동 여부를 미리 파악한 후 주위의 적절한 AP 를 검색하고 Flow 의 경로를 소프트 핸드오버 한다.

1. 서론

기존의 Wi-Fi 무선 랜 네트워크에서는 단말 이동 시 현재 연결된 AP 를 벗어나게 되면 데이터 단절이 일어나게 되고 다른 AP 로의 연결 시간 지연으로 인하여 데이터 끊김, 패킷 손실 등과 같은 서비스 품질 저하 문제가 발생한다. 스마트폰, 태블릿 등과 같은 휴대용 스마트 기기의 사용이 급격히 증가하면서 이동성에 대한 요구도 상당히 늘어나고 있지만 기존의 Wi-Fi 이동성 지원 서비스는 여러 제약사항들로 인하여 폭 넓게 제공하고 있지 못하는 상황이다. 이러한 문제를 해결하기 위해 최근에 미래 인터넷 기술이라고 불리우는 SDN('Software Defined Networking')을 기반으로 한 Wi-Fi 이동성 지원 서비스 연구가 늘어나고 있다.

최근 SDN 을 기반으로 한 Wi-Fi 이동성 지원 서비스 연구들은 대부분 단말기의 연결 AP 가 변경될 때 새로운 AP 로의 연결 인증/인가 과정에서 delay 가 발생하게 되는데 SDN 네트워크 내에 연결되는 단말의 인증 정보를 모든 AP 에 미리 보내 놓아 추가적인 인증 과정을 없이 신속한 핸드 오버가 이루어 질 수 있도록 하여 delay 를 줄이는 방식을 사용하였다. 하지만 이 방법도 delay 를 줄여줄 뿐이지 완벽하게 이동성을 지원하지는 못하고 있다. 제안하는 시스템에서는 SDN 의 중앙집중식으로 네트워크 전체를 관장할 수 있는 원리를 기반으로 하여 단말에 설치된 소프트웨어를 통해 단말의 이동을 미리 예측하고, SDN 컨트롤러에서 데이터 패킷의 경로를 제어하여 패킷을 끊김 없이 받을 수 있도록 하여 Wi-Fi 이동성을 실험을 통해 성능 검증 하기로 한다. 2 장에서는 제안하는 이동성 서비스의 이론을 설명하고, 3 장에서 단말의 이동성 지원 과정을 설명하며, 4 장에서 구현 기술에 대한 설명 후 5 장의 실험을 통하여 증명한다.

2. 제안하는 Wi-Fi 이동성 지원 서비스

본 논문에서는 Wi-Fi 네트워크 환경을 SDN 기술을 기반으로 구축함으로써 단말의 이동성을 지원한다. 단말과 AP 사이의 정보를 SDN Testbed 를 통해 컨트롤러에 공유하여 이를 처리할 수 있도록 함으로써 효과적이고 체계적인 단말의 이동 지원 서비스 제공이 가능하도록 한다.

SDN 기술을 접목하게 된 것은 기존의 Wi-Fi 환경에서 단말이 다른 AP 로 연결이 변경될 때 단말 인증으로 인하여 delay 가 생기기도 하지만 경로도 함께 변경 되고 변경된 AP 의 경로까지 데이터가 내려오는데 시간이 걸리면서 패킷이 손실되어 데이터 단절이 생긴다. 이러한 문제를 해결하기 위해 컨트롤러에서 중앙집중적으로 네트워크 전체를 제어할 수 있는 SDN 을 접목시켜 단말을 통해 움직임을 인식하고, 컨트롤러는 움직임 정보를 받아 변경될 AP 로의 경로를 계산 후 해당 경로를 미리 열어 단말에 연결된 AP 가 변경 되자마자 데이터를 곧바로 이어 받을 수 있도록 하였다. 이를 통해 단말이 손실 없이 데이터를 받을 수 있도록 하고자 이 방법을 제안한다.

그림 1 는 제안하는 SDN 기반 무선 랜 환경을

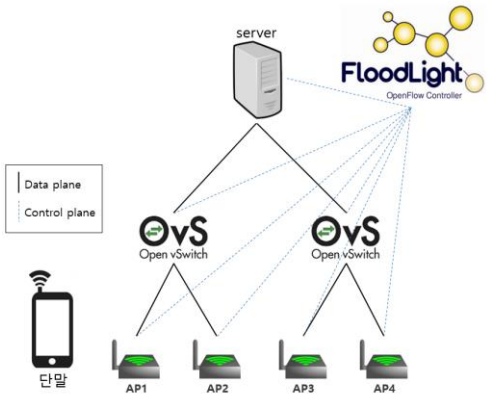


그림 1. 제안하는 서비스 구축 환경

나타낸다. 단말이 서버로부터 데이터를 내려 받아 패킷을 측정하기 위한 데이터플레인 영역을 2 단 트리 형태로 구축하고, 전체 네트워크를 제어를 위해 컨트롤러에 연결하여 컨트롤 플레인 영역을 구축하였다.

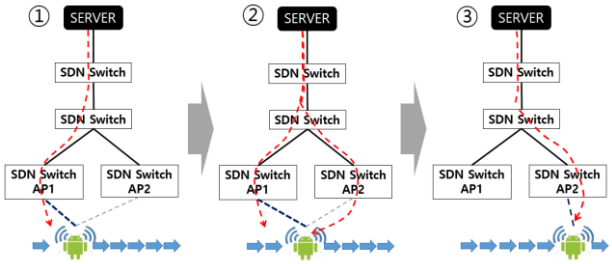


그림 2. 제안하는 이동성 지원 방법

그림 2 은 그림 1 의 SDN 환경에서 이동성을 지원하는 방법을 보여준다.

① 단말은 현재 AP1 에 연결되어 있는 상태로 서버로부터 내려오는 경로는 AP1 의 경로만 열린 상태로 AP1 의 경로에서 단말은 데이터를 내려 받고 있다.

② 단말이 AP2 쪽으로 이동하면서 AP 변경을 감지하여 현재 연결된 AP1 과 연결이 바뀔 AP2 의 경로 모두 열린 상태로 데이터가 내려오고 있다.

③ 단말은 AP2 에 완전 가까워지면서 연결이 변경되고, AP 의 경로를 통해 데이터를 이어받게 되어 이전에 연결되어 있던 AP1 의 경로는 삭제된다.

3. Wi-Fi 이동성 지원 과정

제안하는 Wi-Fi 이동성 지원 방안에서는 단말은 AP 로부터 신호를 측정하여 접속할 AP 를 계산하여 SDN 컨트롤러 Wi-Fi 이동 서비스 모듈에 전송한다. Wi-Fi 이동서비스 모듈은 모바일 단말의 AP 변경을 예측하여 새로운 AP 로의 Flow Path 경로를 SDN 스위치들에 미리 설정한다. 따라서 모바일 단말이 이동 시 중단 없는 데이터 수신이 가능하도록 한다.

그림 3 은 단말이 AP1(current)에서 AP2(future)로 이동할 때 이동성 지원 단계이다. 단말이 받는 데이터의 손실 패킷을 측정하기 위해 UDP SERVER 로부터 데이터를 내려 받고 단말의 애플리케이션으로 측정한 신호 세기를 컨트롤러에 Rest API 형식으로 보내주기 위한 PHP 서버가 존재한다. current 는 현재 연결되어 있는 AP, future 는 다음 연결될 것으로 예측되는 AP 를 나타낸다. 단말은 AP1 에서 AP2 방향으로 이동하고 있고 그림의 빨간 점선은 단말이 이동하는 것에 따라 바뀌는 경로를 보여준다.

① Current AP1 에 연결되어 있는 단말은 4 초 주기로 신호 세기를 측정하여 주위 AP 들의 신호 세기와 증가율을 고려하여 다음 연결

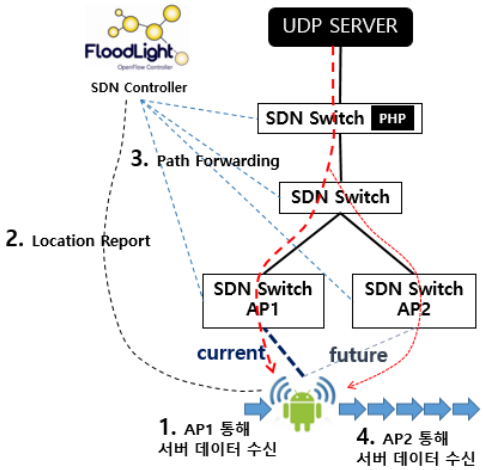


그림 3. AP 변경 시 이동성 지원 과정

될 가능성이 높은 AP 를 선정한다.

② 단말은 현재 연결된 current AP1 과 다음 연결 가능성이 높은 future AP2 정보를 SDN 스위치를 통해 컨트롤러에 전송한다.

③ 컨트롤러는 받은 정보에 따라 future AP2 의 경로에 해당되는 스위치들의 flow table 항목을 새로 만들어 future AP2 가 연결되기 전에 미리 경로를 열어 Packet 을 보내 놓는다.

④ 단말은 future AP2 로 연결이 바뀌어 지속적으로 데이터를 수신하게 되고 future AP2 는 current AP2 가 된다.

4. 구현

4.1 컨트롤러 모듈 구축

컨트롤러 내의 Wi-Fi 이동서비스 모듈은 세 모듈을 통해 동작하는데 첫째로, Host 등록여부를 결정하기 위한 Host(단말) 등록 모듈, 둘째로, Host 로부터 들어온 Rest API 정보를 분석, 비교하여 서버로부터 데이터를 내려 받기 위한 경로를 계산하는 경로 계산 모듈, 마지막으로, 계산된 경로를 스위치들에게 할당해 주는 경로 할당 모듈로 구현하였다.

Host 로부터 current AP SSID, future AP SSID, HOST(단말) MAC address 를 포함한 정보가 Rest API 형식을 사용하여 Host 등록 모듈을 거쳐서 최초로 controller 에 들어온다. Host 등록 모듈에서는 Rest API 를 분석하여 controller 에 Host 의 등록 여부를 결정한다. 결정 과정을 통해 controller 에 Host 가 등록이 된 후에 해당 Host 는 현재 연결된 current AP 를 통해 경로 계산 모듈에게 Packet In 메시지를 보내고, 경로 계산 모듈은 Packet In 메시지가 들어오면 current AP 를 통한 Host 까지의 초기 경로를 열어준다. 초기 경로가 열리고 나면 경로 계산 모듈에서는 Host 로부터 주기적으로 들어오는 Rest API 정보의 current AP, future AP 와 해당 Host 의 이전에 저장되어있는 current AP, future AP 정보와 비교하게 된다. 새로 들어온 정보가 이전에 등록되어 있는 정보와

다르다면 다시 계산을 통해 새로운 경로를 만들어 경로 할당 모듈을 통하여 계산된 경로를 스위치들에게 할당하고 이전의 경로는 삭제한다.

4.2 스마트 폰 백그라운드 서비스 구축

모바일 단말에 설치된 백그라운드 애플리케이션은 3 가지의 단계로 구현하였다. 첫 번째로는 SDN 네트워크 내에 있는 Host 주변 AP 들의 신호 세기를 4 초 주기로 수집한다. 두 번째로는 future AP 를 선정하는 단계로 앞에서 수집한 신호 세기 정보를 이용하여 각 AP 들의 신호 세기와 변화 량을 합산하여 점수를 부여하고 가장 높은 점수의 AP 를 future AP 로 선정한다. 선정된 future AP 가 없을 경우에는 'NULL' 을, 선정된 future AP 가 있을 경우에는 future AP SSID, current AP SSID, Host MAC address 정보를 Rest API 형식을 사용하여 controller 에 전달하게 된다. 마지막으로, 선정된 future AP 의 점수가 정해놓은 임계 치를 넘어가게 되면 Wi-Fi 변경을 요청한다.

5. 실험 및 결과

5.1 실험 환경

- SDN Controller : floodlight
- SDN Switch : OpenVswitch (Raspberry pi2)
- AP : Raspbian (Raspberry pi2 + iptime 동글)

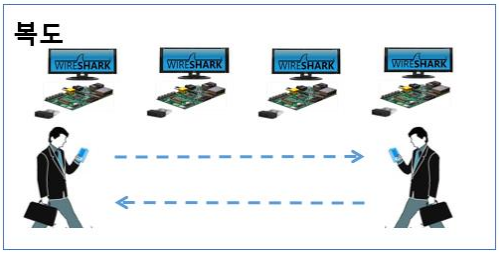


그림 4. 실험 환경

- SERVER : UDP 패킷 0.1 초 마다 단말 전송
- HOST : Android (Android Smart Phone)
  - RSSI 측정, 위치 계산, AP 자동 변경, 손실패킷측정

그림 4 의 환경에서 약 50 미터 길이의 복도에서 AP 4 개를 일정한 간격으로 놓고 각 AP 마다 서버로부터 받는 데이터의 흐름을 모니터링 하기 위한 Wire Shark 를 설치하여 사용자는 복도에 표시된 화살표를 따라 단말을 들고 이동하면서 실험을 진행한다.

정확한 실험을 위해 먼저 세 가지의 테스트를 실시하였다.

- 단말 Wi-Fi 변경 테스트 : 단말이 Wi-Fi 변경을 시도하는 적절한 임계 값을 찾는다.
- 컨트롤러와 단말 간 AP 신호처리 테스트 : 단말에서 컨트롤러로 보내는 Rest API 신호 전송이 잘 되는지 확인한다.
- 경로 할당 테스트 : 단말에서 받은 AP 정보를 확인하여 적절하게 경로를 할당하는지 확인한다.

단말 어플리케이션에서 서버로부터 Host 가 받는 UDP 패킷의 수를 측정하여 증명하고자 한다. 실험은 두 개의 단말을 사용하여 진행하였다. 하나의 단말은 기존에 AP 가 변경이 되고 난 후에 서버로부터 데이터를 내려 받는 경로가 바뀌게 되고, 다른 하나의 단말은 제안하는 방법으로써 변경될 경로를 미리 열어 패킷을 더 빨리 이어 받을 수 있도록 하여 두 가지의 손실 패킷 수를 비교하였다.

5.2 실험 결과 및 분석

그림 6 은 손실 패킷 수를 측정한 결과이다. 기존 네트워크보다 제안하는 네트워크에서의 손실 패킷 수가 더 적게 나타나 성능이 향상되었지만 여전히 손실 패킷은 존재하는 것을 그래프로 나타내었다.

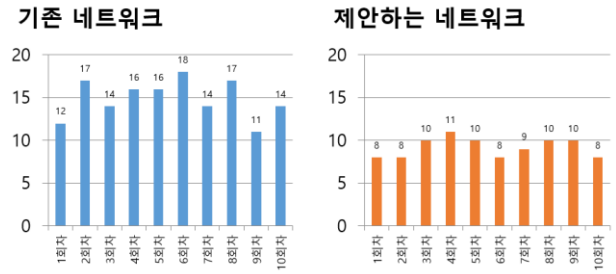


그림 6. 손실 패킷 수 비교

6. 결론

본 논문에서는 SDN 기반의 Wi-Fi 네트워크 환경에서 단말이 hand over 를 시도할 때 끊김 없는 통신 서비스를 제공할 수 있는지 이동성을 테스트 하였다. 단말의 이동경로를 미리 예측하여 SDN 컨트롤러 어플리케이션을 통해 SDN 스위치의 flow table 을 이동경로에 따라 업데이트 하여 단말이 이어받을 데이터를 미리 보내 놓는 과정을 통하여 안정적인 서비스를 지원할 수 있는지 실험하였다. 경로 제어는 잘 구현 하였지만 완벽한 이동성을 지원하기엔 부족하였다. 하지만 손실 패킷 수를 통해 기존의 서비스보다 성능이 향상 된 것을 확인할 수 있었다.

7. 참고 문헌

[1] N. McKeown, T. Amderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: Engineering innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[2] 유재형, 김우성, 윤찬현, "SDN/OpenFlow 기술 동향 및 전망", KNOM Review, Vol. 15, No. 2, 2012

네트워크 가상화 기술 연구 동향

현종환, 홍원기

포항공과대학교 컴퓨터공학과

{noraki, jwkhong}@postech.ac.kr

요 약

네트워크 가상화 기술은 물리 네트워크 상에 여러 개의 가상 네트워크를 생성하여 멀티 테넌시(multi-tenancy)를 제공한다. 또한 SDN/NFV 와 함께 사용될 수 있으며, 네트워크에 유연성 및 확장성을 제공하고, CAPEX 와 OPEX 절감 등의 장점을 바탕으로 클라우드 및 데이터센터 등 많은 곳에서 네트워크 가상화 기술을 도입하였거나 도입을 검토하고 있다. 본 논문에서는 네트워크 가상화 기술을 슬라이스(slice) 기반 가상화 기술과 오버레이 기반 가상화 기술로 나누어 설명하고, 각 가상화 기술의 연구 동향 및 솔루션, 앞으로의 연구 방향에 대해 정리한다.

1. 서론

네트워크 가상화 기술은 최근 활발한 연구가 이루어지고 있는 분야 중 하나로, 물리 네트워크 위에 가상의 노드와 가상의 링크로 구성된 가상 네트워크를 생성할 수 있고, 하나의 물리 네트워크 위에 여러 개의 격리된 가상 네트워크를 생성할 수 있게 해 준다. 또한 각 가상 네트워크를 테넌트에게 할당해 줌으로써 멀티 테넌시(multi-tenancy)를 제공할 수 있다는 특징이 있다. 또한, 네트워크 운영에 있어 유연성과 확장성을 제공하고, CAPEX(Capital Expenditure) 및 OPEX(Operational Expenditure)를 줄일 수 있는 장점이 있다[1].

네트워크 가상화 기술은 슬라이스 (slice) 기반과 오버레이(overlay) 기반의 두 가지 방식으로 분류할 수 있다. 슬라이스 기반 가상화 기법은 OpenFlow 와 같은 프로토콜을 사용하여 네트워크 페브릭을 직접 프로그래밍하는 방식이다. 오버레이 기반 가상화 기법은 가상 네트워크에 속한 엣지 스위치들끼리 VXLAN, NVGRE 등의 터널링 기법을 사용하여 물리 네트워크 위에서 터널을 구성하여 패킷을 주고받는 방식이다.

본 논문에서는 두 가지 방식의 네트워크 가상화 기술에 대해 소개하고, 대표적인 연구 및 솔루션에 대해 비교 분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 네트워크 가상화 기술의 개요 및 슬라이스 기법과 오버레이 기법에 대해 살펴보고, 3 장에서는 각 기술별 연구 동향 및 주요 연구 내용에 대해 정리한다. 4 장에서는 연구 동향 요약 및 향후 연구 이슈에 대해 기술한다.

2. 네트워크 가상화 기술

네트워크 가상화 기술은 크게 슬라이스 기반 가상화 기술과 오버레이 기반 가상화 기술로 분류된다. 본 장에서는 두 가지 가상화 기술에 대한 설명과 장단점에 대해 살펴본다.

2.1. 슬라이스 기반 네트워크 가상화

슬라이스 기반 네트워크 가상화 기술은 네트워크 페브릭을 직접 프로그래밍하는 방식으로, OpenFlow 프로토콜을 주로 사용하여 구현한다. 이 방식에서는 SDN 의 제어 평면과 데이터 평면 사이에 가상화 계층을 추가하며, 가상화 계층에서는 네트워크 물리 자원들을 추상화하고 네트워크 슬라이스를 생성하는 역할을 수행한다.

네트워크 슬라이스는 추상화된 네트워크 물리 자원들의 부분집합으로 이루어지며, 네트워크 토폴로지, 링크 별 대역폭, 스위치 CPU, 포워딩 테이블(forwarding table), Flowspace 등이 포함된다. Flowspace 는 물리 계층 (Layer 1) 에서 전송 계층 (Layer 4)에 해당하는 모든 패킷 헤더의 부분집합으로 정의된다. 또한 하나 이상의 Flowspace 가 하나의 슬라이스에 할당될 수 있다. 가상화 계층은 각 슬라이스 별로 물리 자원과 이에 대응되는 가상 자원의 매핑을 관리한다. 테넌트 컨트롤러에서 발생된 메시지는 가상화 계층에서 해당 슬라이스의 물리 자원에만 적용되도록 수정되며, 스위치로부터 발생된 메시지는 해당 물리 자원이 속해있는 테넌트의 컨트롤러로만 전달된다.

슬라이스 기반 네트워크 가상화 기술의 장점은 오버레이 기반 방식과 달리 별도의 패킷 인캡슐레이션 과정이 필요하지 않다는 점이다. 또한, hop-by-hop 라우팅과 QoS, SLA 관리가 가능하다는 장점이 있다. 하지만, 대부분의 슬라이스 기반 가상화 기술이 OpenFlow 프로토콜을 사용하기 때문에 전체 네트워크 장비를 OpenFlow 를 지원하는 장비로 교체하여야 하는 단점이 있다.

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신 · 방송 연구개발사업의 일환으로 수행하였음. [B0190-15-2011, 한 · 미 SDN/NFV WAN 네트워크 안전성 기술연구 및 Test bed 구축]

2.2. 오버레이 기반 네트워크 가상화

오버레이 기반 네트워크 가상화 기술은 L3 터널링 기법을 사용하여 물리 네트워크 위에 가상의 토폴로지를 생성하는 방식이다. 터널링 기법은 주로 VXLAN 과 NVGRE 가 사용된다. 이 방식에서는 엣지 스위치에서 터널 생성 및 다른 엣지 스위치와의 연결, 패킷 인캡슐레이션 및 디캡슐레이션을 수행한다.

오버레이 기반 네트워크 가상화 기술은 주로 데이터센터 네트워크에 적용된다. 이는 각 호스트가 가상 머신들과 함께 가상 스위치를 구동하고, 가상 스위치가 엣지 스위치의 역할을 수행할 수 있기 때문이다.

이 방식을 도입한 솔루션으로는 VMWare NSX **오류! 참조 원본을 찾을 수 없습니다.**, Microsoft Hyper-V **오류! 참조 원본을 찾을 수 없습니다.**, Juniper OpenContrail **오류! 참조 원본을 찾을 수 없습니다.** 등이 있으며, 대부분의 솔루션은 가상화된 네트워크와 가상화되지 않은 외부 네트워크와의 연결을 지원하기 위해 게이트웨이 기능을 포함하고 있다. 또한 방화벽, 로드밸런싱, IDS/IPS 등의 NFV 기능과 이를 활용한 서비스 체이닝 등의 기능도 지원한다.

이 기술의 장점은 기존에 구축된 IP 네트워크 인프라 상에 바로 배포가 가능하기 때문에 추가적인 네트워크 장비의 구입이 필요하지 않다는 점이다. 하지만 터널링 기법의 사용으로 인한 오버헤드가 발생한다.

3. 네트워크 가상화 기술 연구 동향

슬라이스 기반 네트워크 가상화 방식에는 FlowVisor[2], FlowN[3], VeRTIGO[4], OpenVirteX [5]등의 연구가 진행되었으며, 표 1 에 각 기술 별 비교 분석 내용을 정리하였다.

FlowVisor 는 슬라이스로 격리된 가상 네트워크를 구현하였으나 IP 주소 가상화 및 토폴로지 가상화 등이 지원되지 않는다는 단점이 있다. FlowN 은 IP 주소 가상화 및 토폴로지 가상화를 구현하였으며, 관계형 데이터베이스를 사용하여 시스템의 확장성을 높였다. 하지만 자원 격리를 지원하지 않는다. VeRTIGO 는 임의의 가상 토폴로지를 생성할 수 있는 것이 특징이다. OpenVirteX 에서도 물리 네트워크에 의존적이지 않은 가상 네트워크를 생성할 수 있으며, IP 주소 가상화 기능도 제공한다.

표 1 슬라이스 기반 가상화 기술 비교

슬라이스 기반 가상화 기술	FlowVisor (Stanford, 2009)	FlowN (Princeton, 2012)	VeRTIGO (CREATE-NET, 2012)	OpenVirteX (ON.Lab, 2014)
IP주소 가상화	X	O (VLAN Tunneling)	X	O (IP Rewriting)
토폴로지 가상화	X	O	O	O
제어평면 인스턴스 격리	X	O (Threads)	X	X
데이터 평면 자원 격리	O (Flow Table, CPU, BW)	X	X	O (Flow Table, CPU, BW)
오픈소스 여부	O	X	X	O

표 2 오버레이 기반 가상화 기술 비교

오버레이 기반 가상화 기술	OpenDOVE (Open DayLight)	Hyper-V (Microsoft)	NSX (VMWare)	OpenContrail (Juniper Networks)
오버레이 기법	VXLAN	NVGRE	VXLAN	MPLS over GRE/UDP, VXLAN
Underlay 네트워크	IP Transport Network			
엣지 스위치 종류	OpenDOVE vSwitch	Hyper-V Virtual Switch	Open vSwitch, vSphere Distributed Switch	OpenContrail vRouter
오픈소스 여부	O	X	X	O

오버레이 기반 네트워크 가상화 방식에는 OpenDOVE[6], Microsoft Hyper-V **오류! 참조 원본을 찾을 수 없습니다.**, VMWare NSX **오류! 참조 원본을 찾을 수 없습니다.**, Juniper OpenContrail 등의 솔루션이 있으며, 각 기술 별 비교 분석 내용을 **오류! 참조 원본을 찾을 수 없습니다.**에 정리하였다.

오버레이 기법으로는 VXLAN 과 NVGRE 를 주로 사용하며, 엣지 스위치는 각 솔루션 별로 구현하여 사용하며, NSX 의 경우 Open vSwitch 도 지원한다. 또한, 모두 IP 네트워크 상에서 동작한다.

4. 결론 및 향후 연구

본 논문에서는 네트워크 가상화 기법에 대해 살펴보고, 슬라이스와 오버레이의 두 가지 접근법 및 장단점에 대해 정리하였다. 그리고 각 접근법 별로 연구 동향 및 솔루션에 대해 비교 분석하였다.

향후 연구 이슈로는 네트워크 하드웨어 자원 격리 및 추상화 기법 연구, 가상화로 인해 발생하는 오버헤드 감소, 가상 네트워크를 명세할 수 있는 언어 개발 등이 있다.

5. 참고 문헌

[1] “2015 Special Report: Network Virtualization in the Data Center,” SDxCentral, 2015

[2] R. Sherwood, G. Gibb, K. K. Yap, G. Appenzeller, M. Casado, N. Mckeown, and G. Parulkar, “FlowVisor: A network virtualization layer,” OpenFlow Consortium, Tech. Rep., 2009

[3] D. Drutskoy, E. Keller, and J. Rexford, “Scalable network virtualization in software-defined networks,” IEEE Internet Computing, vol. 17, no. 2, pp. 20–27, 2013.

[4] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, VeRTIGO: Network virtualization and beyond," in Proc. Eu. Workshop on Software Defined Networks (EWSDN), pp. 24–29, 2012.
[5] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, "OpenVirteX: a network hypervisor," in Proc. Open Networking Summit (ONS), Santa Clara, CA, Mar. 2014.
[6] K. Barabash, R. Cohen, D. Hadas, V. Jain, R. Recio, and B. Rochwerger, "A Case for Overlays in DCN Virtualization," in Proc. 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching, pp. 30-37, 2011.

머신 러닝 기법을 활용한 SDN 트래픽 엔지니어링 기법 개선 방안

최준묵*o, 한윤선**, 현종환*, 홍원기*

포항공과대학교 *컴퓨터공학과, **정보전자융합공학부

{juk909090, seon054, noraki, jwkhong}@postech.ac.kr

요 약

IT 기술의 발전으로 인해 데이터 센터 트래픽이 급증하고 있다. 따라서 네트워크 트래픽을 관리하는 트래픽 엔지니어링의 중요성이 커지고 있다. 트래픽 엔지니어링은 트래픽을 분석하고 조절하여 전체 네트워크의 성능을 최적화하는 기법이다. 그러나 기존의 제어 평면과 데이터 평면이 고정 되어 있는 네트워크에 기반하는 트래픽 엔지니어링 기법들은 효율적이지 못 하다. 그 이유는 다중 경로를 효율적으로 활용하지 못 하거나 네트워크의 전역 상황을 활용하지 못 하며, 트래픽 간의 조율이 불가능하기 때문에 네트워크를 효율적으로 활용하지 못 하기 때문이다. 따라서 최근 화제가 되고 있는 Software Defined Networking (SDN) 기술을 활용하여 이러한 문제를 해결하고자 하는 트래픽 엔지니어링 기법들이 등장하였다. 본 논문에서는 SDN 을 활용하는 트래픽 엔지니어링 기법들을 소개하고 이러한 기법들의 성능을 저하 시킬 수 있는 문제점을 분석한다. 그리고 분석한 문제점을 머신 러닝 기법 알고리즘을 활용하여 개선 시킬 수 있는 방향을 제시하고자 한다.

1. 서론

오늘날 SNS 서비스의 발달, 동영상 스트리밍 기술, 클라우드 컴퓨팅 기술 등과 같은 급진적인 IT 기술의 발전으로 인해 네트워크 트래픽이 급증하고 있다. 이로 인해 네트워크의 트래픽을 효율적으로 분산시키거나 적합한 경로를 할당하여 관리하는 트래픽 엔지니어링의 중요성이 커지고 있다. 트래픽 엔지니어링은 트래픽을 분석하고 조절하여 전체 네트워크의 성능을 최적화하는 기법이다 [1]. 기존의 트래픽 엔지니어링 기법들은 다중 경로를 효율적으로 활용하지 못 하거나 네트워크의 전역 상황을 활용하지 못 하며, 트래픽 간의 조율이 불가능하기 때문에 네트워크를 효율적으로 활용하지 못 한다는 문제점을 가지고 있다.

이러한 문제점을 극복하기 위해 최근 Software Defined Networking (SDN) 기술을 활용하는 트래픽 엔지니어링 기법들이 제시되었다 [1][2]. SDN 은 네트워크의 제어 평면과 데이터 평면을 분리하고 제어 평면에 대한 유연한 제어를 가능하게 하여 네트워크를 효율적으로 관리 할 수 있게끔 해주는 기술이다. 이러한 기술의 도입으로 인해 SDN 에 기반한 트래픽 엔지니어링 기법들은 몇 가지 이점을 활용할 수 있게 되었다. 먼저 트래픽 엔지니어링에 활용할 수 있는 네트워크의 전역 상태를 집중형 (centralized) 컨트롤러에서 다룰 수 있으며 이를 통해 네트워크를 최적의 상태로 빠르게 수렴시킬 수 있으며 높은 활용도 실현이 가능하다. 또한 플로우 테이블을 통해 유연한 트래픽 엔지니어링 구현이 가능하다.

본 논문에서는 SDN 을 활용하는 트래픽 엔지니어링 기법들을 분석하고 이러한 기법들의 성능을 저하 시킬 수 있는 문제점을 분석한다. 그리고 분석한 문제점을 머신 러닝 기법 알고리즘을 활용하여 개선 시킬 수 있는 방향을 제시하고자 한다.

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

2. 관련 연구

2.1. SDN 기반 트래픽 엔지니어링 기법

SDN 에 기반하는 트래픽 엔지니어링 기법들은 트래픽에 효율적인 경로와 대역폭을 할당하여 네트워크의 활용도나 고장 허용 한계(Fault tolerance) 성능을 향상시키고자 한다. 이를 위해 별도의 모니터링 서버를 사용하거나 스위치로부터 트래픽 정보를 수집하여 알고리즘의 효율성을 위해 정보를 집적한다. 그리고 할당 알고리즘을 통해 트래픽마다 적절한 경로와 대역폭을 할당한다.

MicroTE [3]는 현재 트래픽의 값과 트래픽의 평균값의 차이가 일정한 범위 내에 있는 지 확인하여 트래픽의 예측 가능성을 판단하여 이를 트래픽 엔지니어링에 활용한다. 할당 알고리즘은 트래픽 통계정보를 사용하여 예측 가능한 트래픽에 대해 선형계획법(Linear programming)이나 Bin-packing heuristic 을 적용하여 먼저 경로를 할당한다. 그리고 예측 불가능한 트래픽에 대해서 이미 할당된 대역폭을 가중치로 반영하여 가중치가 존재하는 ECMP 알고리

즘을 적용한다.

Hedera [4]는 트래픽의 크기 변화에 따라 능동적으로 대응 할 수 없는 Equal Cost Multipath(ECMP)의 단점을 해결하기 위해 트래픽의 크기에 따라 서로 다른 트래픽 엔지니어링 기법을 사용한다. 트래픽의 크기가 작을 때에는 ECMP 방식을 사용하며 트래픽이 커질 경우 SDN 컨트롤러에서 적절한 경로를 할당해주는 방식이다. 특정 트래픽을 선택적으로 스케쥴링하는 방식을 통해 Hedera 는 알고리즘으로 인한 부하를 최소화하면서도 네트워크 활용도를 높일 수 있다.

2.2 머신 러닝 기법을 활용한 트래픽 관련 연구

머신 러닝 기법을 통해 컴퓨터에게 자료의 특징을 스스로 분석하고 예측 할 수 있는 능력을 부여할 수 있다. 이러한 머신 러닝 기법에는 K-means clustering, Regression 등 다양한 기법들이 있으며 이를 통해 자료를 분류하거나 예측 할 수 있다. 따라서 트래픽 패턴 특징에 따라 트래픽을 분류하거나 트래픽의 크기를 예측하여 이를 트래픽 엔지니어링에 활용 할 수 있다. 이러한 연구에는 TCP 처리량 예측 [5], 트래픽 분류 [6]가 있다.

3. 본론

MicroTE 는 트래픽의 predictability 와 같은 간단한 트래픽 패턴을 활용하고자 하였지만 이는 트래픽 크기 변화가 거의 없는 패턴에 대해서만 의미가 있다. 또한 트래픽을 할당 알고리즘에서 사용하기 위해 서버 랙(Rack) 단위로 집적함으로써 많은 정보를 잃어버리게 된다는 단점을 가지고 있다. Hedera 또한 일정한 한계에 따라 스케쥴링할 트래픽을 선택하기 때문에 트래픽의 패턴에 따라 효율적인 트래픽 엔지니어링이 불가능하다는 단점을 가지고 있다. 즉 개별적인 트래픽에 대한 특징을 최대한 활용하면서도 확장성을 위해 어느 정도의 집적이 필요하다는 문제와 트래픽 패턴에 대한 예측 문제를 해결하는 것이 이러한 트래픽 엔지니어링을 좀 더 개선 할 수 있는 방안이 될 것이다.

3.1 집적 수준을 위한 머신 러닝 기법

K-means 클러스터링 알고리즘은 Unsupervised learning 기법의 대표적인 방식으로 주어진 데이터들을 거리를 이용하여 임의의 K 개의 클러스터로 묶는 알고리즘이다. 따라서 트래픽의 크기, 크기 변화와 같은 트래픽 패턴에 관한 정보와 포트 번호, 목적지 IP 주소와 같이 어플리케이션과 관련된 정보를 벡터로 정의하여 이들의 유클리디안 거리를 사용하는 K-means 클러스터링 알고리즘을 생각 해 볼 수 있다. 이러한 클러스터링 알고리즘을 사용하게 된다면 유사한 특징을 가지는 트래픽끼리 집적 할 수 있게 될 것이며 이를 효율적인 트래픽 엔지니어링 구현에 사용 할 수 있을 것이다.

3.2 트래픽 패턴 예측을 위한 머신 러닝 기법

Unsupervised learning 기법과 달리 Supervised learning 기법은 데이터로부터 어떠한 관계식을 유추 해내기 위한 기법이다. 따라서 특정 시간대나 특정한 서버에서의 트래픽을 예측하는 데에 이러한 기법이 사용될 수 있다. 대표적인 기법으로는 회귀분석이 있으며 예측된 트래픽에 따라 다른 트래픽 경로를 미리 변경 해두는 등의 활용이 가능할 것이다.

4. 결론

기존의 트래픽 엔지니어링 기법들은 트래픽의 경로 할당을 위한 최적화 문제를 풀 때, ToR(Top of Rack) 단위와 같은 형태로 트래픽 정보를 집적하여 사용하였다. 이러한 집적은 알고리즘의 계산 복잡도를 줄이는 데에 도움이 되지만 특정 트래픽이 가지는 트래픽 패턴 같은 정보를 잃어버리게 되는 단점이 있다. 따라서 트래픽 패턴에 대한 정보를 활용할 수 있되 가급적 그 수를 줄일 수 있는 집적 알고리즘이 필요하다. 이에 K-means 알고리즘과 같은 클러스터링 알고리즘을 적용하는 집적 알고리즘을 고안하고자 한다. 또한 머신 러닝 기법을 활용하여 트래픽의 미시적인 특성뿐만 아니라 거시적인 관점에서 트래픽을 예측 할 수 있다면 트래픽 변화에 효율적으로 적응하는 알고리즘을 고안 할 수 있을 것으로 기대된다.

5. 참고 문헌

[1] I.F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow networks,” *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 71, pp. 1-30, Oct. 2014.

[2] 서신석, 리건, 현종환, 유재형, 홍원기, 백성복, 황찬규, 이영우, “소프트웨어 정의 네트워킹 기술을 활용한 데이터 센터 네트워크 트래픽 엔지니어링,” *KNOM Review*, Vol. 16, No. 2, pp. 12-25, Dec. 2013.

[3] T. Benson, A. Anand, A. Akella and M. Zhang, “MicroTE: Fine Grained Traffic Engineering for Data Centers,” in *Proc. 7th COnference on emerging Networking EXperiments and Technologies(CoNEXT ’11)*, 2010.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghaven, N. Huang and A. Vahdat, “Hedera: dynamic flow scheduling for data center networks,” in *Proc. 7th USENIX Conference on Networked Systems Design and Implementation(NSDI ’10)*, 2010.

[5] M. Mirza, J. Sommers, P. Barford and X. Zhu, “A machine learning approach to TCP throughput prediction,” in *Proc. ACM SIGMETRICS(2007)*, Vol. 35(1), pp. 97-108, June. 2007.

[6] T. T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE Communications Surveys & Tutorials*, Vol. 10(4), pp. 56-76, 2008.

Dynamic ACL 모델을 사용한 SCADA 시스템 내 FTP 서비스의 화이트리스트 표현에 관한 연구

Study on the Whitelist Representation of FTP Service in SCADA System by using Dynamic ACL Model

정우석, 김성민, 구영훈, 김명섭

고려대학교 컴퓨터정보학과

{ hary5832, gogumiking, gyh0808, tmskim }@korea.ac.kr

요 약

SCADA 시스템은 최근 비즈니스 시스템과의 통합으로 인해 개방형 시스템으로 전환됨에 따라 보안상 취약점들이 증가하고 있다. SCADA 시스템의 보안문제 해결을 위해 화이트리스트를 기반으로 한 제어시스템의 보안 기법이 각광받고 있다. 현재 대부분의 화이트리스트 보안 기법에서 사용하는 Static ACL 모델의 경우 표현의 한계점을 지니고 있다. 본 논문에서는 FTP 서비스의 특징을 추출하여 Dynamic ACL 모델로 표현하는 방법을 제안한다.

Keywords : Industrial Control System, SCADA, Whitelist, Dynamic ACL, FTP

1. 서론

제어 시스템은 특정 산업현장 전체 또는 산업 단지를 전반적으로 감시하고 제어하기 위하여 다양한 기간 시설과 산업에서 사용되고 있는 컴퓨터 기반의 시스템이다. 미국 ICS-CERT 는 미국내 기반시설에 대한 사이버공격 2015 년 한해 동안 공식 보고된 침해사고 수만 294 개에 달한다고 발표했다. 안전이 증명된 것만을 허용하는 화이트리스트 보안 기법은 제어시스템 환경에서 보안을 담보할 수 있는 효율적 방안으로 주목 받고 있다.

현재의 화이트리스트를 사용한 제어시스템의 보안 기법은 Static ACL 모델을 사용하고 있다. 하지만 Static ACL 모델은 표현상의 한계점을 지닌다.

본 논문에서는 기존 Static ACL 모델이 가지는 한계를 극복하기 위해 FTP 서비스를 대상으로 정의 된 순서를 가지는 ACL 의 집합인 Dynamic ACL 모델을 사용하여 표현하는 방법을 제시한다.

2 장에서는 Static ACL 모델이 가지는 한계점을 서술하고, 3 장에서는 제안하는 FTP Dynamic ACL 모델에 대해 서술한다. 4 장에서는 결론 및 향후 연구에 대해 서술한다.

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단(No.2015R1D1A3A01018057) 및 한국과학기술정보연구원의 "첨단연구망기반 협업플랫폼 서비스 및 글로벌 연동"과제(K-16-L01-C02-S03)의 지원을 받았음.

2. Static ACL 의 한계점

SCADA 시스템에서 FTP 나 OPC 와 같이 Dynamic 서버 포트를 사용하는 통신을 Static ACL 모델로 표현하는 방법은 ANY-ANY 규칙을 사용하는 것이 유일하다. 하지만 ANY-ANY 규칙을 허용하게 되면 해당 IP 간의 모든 연결에 대하여 모든 포트를 오픈을 해야한다. 이는 기존 해당 IP 간에 생성된 다른 ACL 규칙들이 무의미 해지게 되는 것을 의미하며, 특히 FTP 를 사용하는 서버는 해당 서버에 접근하는 모두에게 오픈 될 가능성이 있다.

두번째는 빈도에 상관 없이 Static ACL 모델로 작성된 모든 규칙은 항시 오픈 된다는 점이다. 1 년에 몇 회 사용되지 않는 규칙도 불필요하게 항시 열어두어야한다.

3. 제안하는 Dynamic ACL 모델

ACL(Access Control List)은 사용자들이 특정 시스템에 접근할 수 있는 권한을 설정해 놓은 리스트이다. ACL 은 표현 할 수 있는 범위에 따라 Static ACL 모델과 Dynamic ACL 모델로 나누어진다. Static ACL 모델이 단순한 네트워크 및 시스템에 대한 접근 권한 리스트라면, Dynamic ACL 모델은 ‘A 가 접근한 이후에 B 가 접근 가능하다.’ 또는 ‘C 는 월요일에만 접근 가능하다.’ 라는 ACL 의 순서나 조건까지 표현이 가능한 모델이다.

Figure 1 은 Passive FTP 를 사용할 때, ANY-ANY 규

칙이 생성되는 과정을 도식화 한 것이다. 서버의 21 번 포트를 사용하여 세션이 유지되는 사이 서버와 클라이언트에서 랜덤 포트로 데이터를 교환한다. 데이터 전송을 위해 서버와 클라이언트 모두 랜덤 포트를 사용하기 때문에 ANY-ANY 이 생성된다.

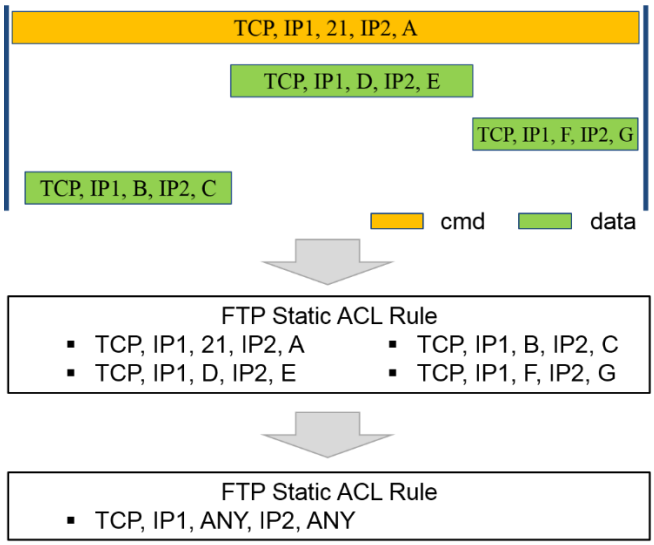


Figure 1. ANY-ANY 규칙이 생성되는 과정

Figure 2 는 일반적으로 Static ACL 모델을 사용하여 화이트리스트를 표현한 예시이다. 5-tuple 정보를 포함하고 있지만 ACL 간의 순서나 조건을 표현 하지 못한다는 한계점을 가지고 있다.

{TCP, (IP1, ANY <-> IP2, ANY)}

Figure 2. 일반적인 Static ACL 모델

Figure 3 은 FTP 를 본 논문에서 제안하는 Dynamic ACL 모델을 통해 표현한 것이다. IP1 과 IP2 사이에 서버 21 번 포트를 사용한 세션이 열려 있는 경우 ANY-ANY 규칙을 각각 n 시간만큼 오픈한다는 의미를 내포하고있다. 또한 "<<"를 사용하여 서버를 구분함으로서 더 확실한 제어가 가능하다.

{TCP, (IP1, 21 <<-> IP2, ANY),
+, ((IP1, ANY <<-> ((IP2, ANY), n)}

Figure 3. 제안하는 Dynamic ACL 모델

Algorithm 1 은 FTP 를 Dynamic ACL 모델로 표현하기 위한 규칙 생성 알고리즘이다. 알고리즘의 입력은 플로우의 5-tuple 정보와 플로우의 지속 시간인 duration 이고, 알고리즘의 결과물은 FTP 규칙이다.

Input : Flows
Output : FTPRules

```
1:  For each Flow in Flows do
2:      makeFTPList(protocol, dIP, dPort,
                    sIP, sPort, duration)
3:  end for
4:
5:  For each Flow in Flows do
6:      FTPRuleGeneration(protocol, dIP, dPort,
                          sIP, sPort, duration)
7:  End for
8:
9:  makeFTPList:
10:     if dPort == 21
11:         addFTPList(sIP, dIP)
12:
13: FTPRuleGeneration:
14:     if sIP and dIP ∈ FTPList
15:         AddFTPDurationDic(sIP+dIP: duration)
16:
17:     For sIP+dIP in FTPDurationDic
18:         Limittime(sIP+dIP) = MAX(duration)
19:
20:     For dIP in FTPList
21:         AddFTPRule(protocol, dIP, ANY,
                       sIP, ANY, Limittime+ α )r
22:     End for
```

Algorithm 1. FTP Rule 추출

4. 결론 및 향후 연구

본 논문에서는 SCADA 시스템의 보안에 사용되는 화이트리스트 보안 기법의 표현을 위해 사용하는 static ACL 모델이 가지는 한계점을 서술하고, 이를 극복하기 위한 방법으로 FTP 서비스를 Dynamic ACL 모델로 표현 할 수 있는 방법을 제안하였다. 제안한 모델은 Static ACL 모델로는 표현하기 어려운 Passive FTP 의 통신 특성과 순서를 반영하여 Dynamic ACL 모델로 표현하였다.

향후 연구에서는 FTP 뿐만 아닌 Static ACL 모델로 표현하기 어려웠던 Dynamic 서버 포트를 사용하는 모든 프로토콜에 대하여 Dynamic ACL 모델로 표현하고 이를 실제 환경에 적용하는 방법에 대해 연구할 계획이다.

[1] Yun, Jeong-Han, et al. "Burst-based anomaly detection on the DNP3 protocol." International Journal of Control and Automation 6.2 (2013): 313-324.

[2] Choi, Seungoh, et al. "Traffic-Locality-Based Creation of Flow Whitelists for SCADA Networks." Critical Infrastructure Protection IX. Springer International Publishing, 2015. 87-102.

[3] 유형욱, 윤정한, 손태식. "제어시스템 보안을 위한 whitelist 기반 이상징후 탐지 기법." 한국통신학회논문지 38.8 (2013): 641-653

[4] Schneider, Johannes, Sebastian Obermeier, and Roman Schlegel. "Cyber security maintenance for SCADA systems." Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research. British Computer Society, 2015.

응용 트래픽 분류에 정교한 페이로드 시그니쳐 구조

Payload Signature Structure for Elaborate Application Traffic Classification

구영훈, 이성호, 김성민, 이수강, 김명섭

고려대학교 컴퓨터정보학과

{gyh0808, gaek5, gogumiking, sukanglee, tmskim}@korea.ac.kr

요 약

오늘날 네트워크 고속화와 함께 인터넷 연결이 가능한 다양한 디바이스의 보편화로 인해 네트워크 기능을 사용하는 응용이 급속도로 생성되고 있다. 이에 따라 한정된 네트워크 자원을 효율적으로 사용하고 안정적인 서비스를 제공할 수 있도록 다양한 종류의 응용에 대한 정확한 트래픽 분류가 필수적이다. 다양한 페이로드 시그니쳐 추출 방법론 가운데, 대부분의 페이로드 시그니쳐 구조는 단순히 트래픽의 페이로드 내 출현 빈도가 높은 공통 부분 문자열이다. 따라서, 특정 응용프로그램에 고유하지 않고 다른 응용프로그램과 중복되는 페이로드 시그니쳐가 추출될 문제가 있다. 이에 본 논문에서는 다른 응용프로그램과 중복되지 않고 특정 응용프로그램에 특화된 보다 정교한 시그니쳐의 구조를 제안한다. 본 시그니쳐의 구조는 3 단계로 공통 부분 문자열인 Content 시그니쳐, 동일한 Packet 내 Content 시그니쳐의 집합인 Packet 시그니쳐, 동일한 Flow 내 Packet 시그니쳐의 집합인 Flow 시그니쳐로 구성된다. 본 시그니쳐 구조와 기존의 페이로드 시그니쳐 구조를 실제 응용 트래픽 분류에 적용하여 그 실효성을 증명한다.

Keyword: Signature structure, Content signature, Packet signature, Flow signature, Completeness, False positive

1. 서론

오늘날 네트워크 고속화와 더불어 인터넷 연결이 가능한 스마트 기기들의 보편화에 따라 인터넷에 기반한 응용프로그램의 사용이 급격하게 증가하고 있다. 이에 따라 한정된 네트워크 자원을 효율적으로 사용하고, 사용자에게 안정적인 서비스를 제공하기 위한 많은 연구가 수행되고 있다. 이를 위해서는 다양한 종류의 응용 레벨 트래픽을 정확하게 분류할 수 있는 방법이 필요하다.

트래픽의 분류를 위한 다양한 방법론이 존재하지만 분석률과 정확도 측면에서 가장 성능이 높은 방법론은 페이로드 시그니쳐 기반 분석 방법이다[1,2,7,8]. 이는 페이로드를 추출한 응용의 트래픽에 대해서는 정확한 분석이 가능하기 때문이다. 하지만 오늘날 급증하는 응용과 함께 확인되지 않은 트래픽에 대하여 적용하였을 시 시그니쳐의 중복성에 의해 잘못 분류할 가능성이 존재한다.

기존에 연구된 대부분의 페이로드 시그니쳐의 구조는 단순히 페이로드 내의 공통 부분 문자열이다[3,5,10]. 따라서 특정 응용프로그램에 고유하지 않고 다른 응용프로그램과 중복되는 페이로드 시그니쳐가 추출될 가능성이 있다. 이는 네트워크 관리에 있어 신뢰도를 떨어뜨리며 네트워크 정책이나 고장 진단, 용량 계획 등을 정확히 수행할 수 없다. 특히 네트워크 보안 분야에서는 악성 트래픽을 오탐(False Positive)하거나 미탐(False Negative)할 경우 엄청난 손실을 초래할 수 있다.

급격하게 증가하는 응용 트래픽에서 네트워크 관리자가 페이로드를 일일히 확인하여 검증된 시그니쳐를 수작업으로 추출하는 과정은 많은 시간과 인력을 낭비한다. 이를 해결하기 위한 다양한 자동 페이로드 시그니쳐 추출 방법 연구가 진행되고 있지만, 자동으로 추출된 페이로드 시그니쳐의 경우 각 응용 및 서비스를 오탐이나 미탐이 없이 그 응용만을 정확하게 탐지할 수 있는 시그니쳐인지 보장할 수 없다. 따라서 좀 더 특정 응용 프로그램에 고유하고 다른 응용프로그램의 시그니쳐와 확실히 구분되는 정교한 페이로드 시그니쳐 구조가 필요하다.

본 논문에서는 특정 응용프로그램에 특화된 보다 정교한 페이로드 시그니쳐 구조를 제안한다. 본 시그니쳐의 구조는 총 3 단계로 페이로드 내 공통 부분 문자열인 Content 시그니쳐, 동일한 패킷 내 Content 시그니쳐의 집합인 Packet 시그니쳐, 동일한 Flow 내 Packet 시그니쳐의 집합인 Flow 시그니쳐로

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2015R1D1A3A01018057).

구성된다. 본 시그니쳐 구조와 선행 연구의 시그니쳐 구조를 실제 응용 트래픽 분류에 적용하여 비교한 결과 응용프로그램의 분석률은 유지하면서 오탐률을 감소시켜 정확한 분류가 이루어질 수 있도록 하였다.

본 논문은 다음과 같은 순서로 기술한다. 본 장의 서론에 이어 2 장에서는 관련 연구에 대해 살펴보고, 3 장에서는 선행 연구의 한계를 설명한다. 4 장에서는 제안하는 페이로드 시그니쳐의 정의와 구조에 대해 설명한다. 5 장에서는 실제 응용 트래픽 분류에 제안한 페이로드 시그니쳐 구조를 적용하여 그 타당성을 증명하기 위한 실험 결과를 기술한다. 마지막으로 6 장에서는 결론 및 향후연구를 기술한다.

2. 관련 연구

페이로드 시그니쳐 분석 방법은 Packet 의 페이로드를 확인하여 다른 응용 프로그램과 구분 지을 수 있는 특징을 추출하고 응용을 식별할 수 있는 시그니쳐로 정의한 후 분석할 트래픽 Packet 의 페이로드 내에 응용의 시그니쳐 포함 여부를 판단하여 응용을 분석하는 방법이다. 페이로드 시그니쳐 추출 방법론은 다양한 방법으로 연구되고 있다.

Kim 의 연구[3]에서는 웜 시그니쳐 생성을 위한 방법으로 COPP(Content-based Payload Partitioning)를 사용하여 페이로드 내 연속적으로 발생하는 공통 부분 문자열을 breakmark 로 사용하였다. Cheng 의 연구[10]에서는 페이로드를 3 종류의 비트(1bit, 4bit, 8bit) 단위로 자른 후 같은 오프셋에서 두 비트 시퀀스 간의 공통 비트 시퀀스를 찾아 시그니쳐로 활용하였다. Mingjiang 의 연구[11]에서는 Shingle 이라는 단위의 서브 스트링을 정의하고 Shingle 의 발생 빈도수 및 임계값을 이용하여 최종 공통 문자열을 페이로드 시그니쳐로 사용하였다. Park 의 연구[5]에서는 바이오 인포매틱스의 유전자 분석 알고리즘 기법의 하나인 LCS (Longest Common String) 알고리즘을 응용 트래픽 시그니쳐 추출 목적에 맞게 변형한 LASER(LCS-based Application Signature ExtRaction) 알고리즘을 사용하여 비교할 두 페이로드 내에서의 최장 길이 공통 문자열을 시그니쳐로 활용하였다. Newsome 의 연구[4]와 Feng 의 연구[6]에서는 Smith-Waterman 알고리즘을 이용하여 두 페이로드 내의 공통적인 부분 문자열의 집합을 시그니쳐로 사용하였다.

앞서 언급한 연구들의 페이로드 시그니쳐 구조는 단순히 페이로드 내 공통 부분 문자열이므로 다른 응용과 중복이 되는 시그니쳐가 추출될 문제가 있다. 이에 정확한 트래픽 분류를 위하여 특정 응용에 더 정교한 페이로드 시그니쳐 구조가 필요하다.

3. 선행 연구의 페이로드 시그니쳐 구조의 한계

본 장에서는 선행 연구의 페이로드 시그니쳐 구조에 대한 한계를 설명한다. 위에서 언급한 논문들의 페이로드 시그니쳐 구조는 Smith-Waterman 알고리즘을 제외하고 단순히 여러 페이로드에서 공통적으로 발견되는 부분 문자열이다. 이와 같은 페이로드 시그니쳐 구조는 특정 응용에 대한 정답지 트래픽을 수집하여 추출한 시그니쳐라 할지라도 이 시그니쳐가 목적하고 있는 특정 응용에 대해 고유한지 다른 응용의 트래픽에서는 전혀 발견이 되지 않는지 보장할 수 없다.

표 1 과 표 2 는 추출된 시그니쳐 중 다른 응용과 중복이 될 가능성이 큰 시그니쳐의 예이다.

Example
HTTP /1.1 Accept Referrer GET / 201 User-Agent Content-Type

표 1. 특정 프로토콜의 메타데이터로 쓰이는 문자열

표 1 과 같이 특정 프로토콜에 메타 데이터로 쓰이는 문자열이 시그니쳐로 추출될 경우 이 프로토콜을 쓰는 응용에 대해서는 모두 같은 시그니쳐가 추출이 될 수 있다. 예를 들어 여러 Packet 에서 발견되는 HTTP Request, Response 의 명령어나 HTTP header Field 의 경우 페이로드 시그니쳐로 추출될 가능성이 크고, HTTP 프로토콜은 다양한 응용에서 사용되는 프로토콜이므로 특정 응용의 식별에 큰 효용 가치가 없는 시그니쳐이다.

또한 사전적 의미를 가지는 문자열이 특정 응용의 페이로드 시그니쳐로 추출되는 경우 이는 다른 응용의 트래픽에서도 발견될 가능성이 매우 크다. 표 2 는 사전적 의미를 가지는 단어가 특정 응용의 페이로드 시그니쳐로 추출되는 경우이다.

Example
Album Image Data : Server : Music

표 2. 사전적 의미를 가지는 문자열

예를 들어 Album 과 같은 문자열은 웹 캠 응용의 트래픽에서도 나올 수 있고 음악 관련 응용의 트래픽에서도 나올 수 있다. Music 과 같은 문자열의 경우에도 같은 목적의 음악 관련 응용이라 할지라도 다양한 음악 서비스를 제공하는 응용 중 정확히 어

떤 음악 응용 프로그램인지 구분할 수 없다.

그림 1 은 Smith-Waterman 알고리즘을 이용하여 추출한 페이로드 시그니쳐의 구조로 단순히 공통 부분 문자열이 아닌 두 페이로드 내의 공통 부분 문자열의 집합이다. 이와 같은 페이로드 시그니쳐 구조의 경우 앞서 언급한 논문들의 페이로드 시그니쳐 구조보다는 응용에 대한 고유성이 높아질 수 있다. 두 개의 Packet 페이로드를 입력으로 하여 공통 부분 문자열의 집합을 찾기 때문에 이 시그니쳐는 한 Packet 단위의 시그니쳐라 할 수 있다. 이는 앞서 언급한 공통 부분 문자열 시그니쳐보다 정교하다.

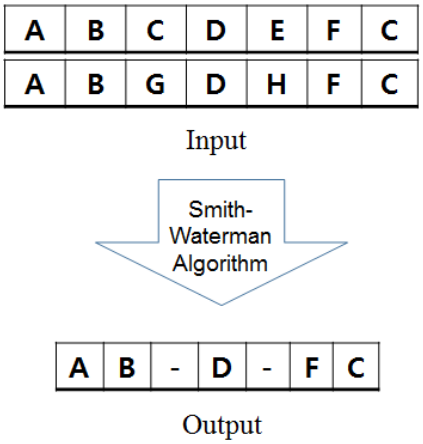


그림 1. Smith-Waterman 알고리즘

그러나 Smith-Waterman 알고리즘의 경우 시간복잡도와 계산 복잡도가 크며 입력으로 항상 2 개의 packet 페이로드를 사용하여 하나의 공통 문자열 집합을 생성하기 때문에 모든 Packet 내의 공통 부분 문자열 집합을 찾기 위해서는 Packet 개수의 제곱만큼 Smith-Waterman 알고리즘 과정을 수행해야 한다. 결과물로 나온 공통 부분 문자열 집합 중 시그니쳐로 추출할 공통 부분 문자열 집합을 선정하기 위하여 일정 빈도수나 임계값을 계산하고 이를 넘는 공통 부분 문자열 집합을 탐색하기 위해서는 더 많은 계산 과정이 필요하다. 또한 그림 1 에서 LASER 알고리즘과 같은 선행 연구의 방법에서는 시그니쳐로 추출될 수 있었던 공통 부분 문자열 AB, D, FC 는 Smith-Waterman 알고리즘을 사용시 추출이 되지 않아 AB, D, FC 가 분석할 수 있었던 트래픽을 분류할 수 없게 되어 분석률이 감소할 수 밖에 없다. 그림 2 에서는 AB 와 D 와 FC 가 동시에 존재하는 Packet 만을 분석하는 Smith-Waterman 알고리즘 분석률의 한계를 도식화한 것이다.

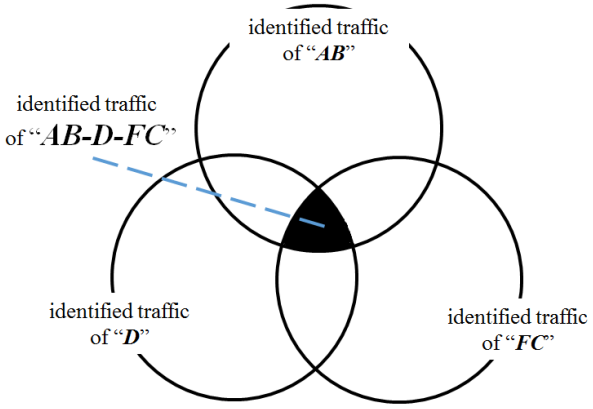


그림 2. Smith-Waterman 알고리즘의 분석률의 한계

다음 장에서는 다른 응용과 중복되지 않고 특정 응용에 정교한 페이로드 시그니쳐의 구조를 설명한다.

4. 제안하는 페이로드 시그니쳐의 구조

본 논문에서는 3 단계로 구성된 페이로드 시그니쳐 구조를 제안한다. 먼저 1 차적으로 페이로드 내 일정 빈도 수를 만족하는 공통 부분 문자열을 시그니쳐로 추출하고 이를 Content 시그니쳐라 명명한다. 이는 선행연구의 대부분의 페이로드 시그니쳐의 구조와 같다. 2 차적으로 동일한 Packet 내에서 추출되는 Content 시그니쳐의 집합 중 일정 빈도수를 만족하는 집합을 시그니쳐로 추출하고 이를 Packet 시그니쳐라 명명한다. Packet 시그니쳐는 하나의 Packet 에서 여러 개의 Content 시그니쳐가 매칭되어야 하기 때문에 오탐률이 적어 정확성이 향상된다. 따라서 Content 시그니쳐보다 응용에 정교한 시그니쳐이다. 3 차적으로 동일한 Flow 내에서 추출되는 Packet 시그니쳐의 집합 중 일정 빈도 수를 만족하는 집합을 Flow 시그니쳐라 명명한다. Flow 시그니쳐는 하나의 Flow 내에서 여러 개의 Packet 시그니쳐가 매칭되어야 하기 때문에 Packet 시그니쳐보다 더 응용에 정교하고 오탐률이 적다.

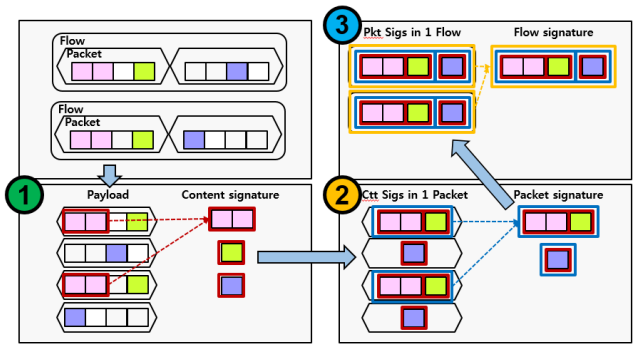


그림 3. 제안하는 시그니쳐 구조의 추출 과정

수식 1 은 제안하는 시그니쳐의 구조이다. C 는 Content 시그니쳐, P 는 Packet 시그니쳐, F 는 Flow

시그니쳐를 의미한다.

C={c | c is single substring in a payload}

P={p | p is a subset of C, p appears in a packet}

F={f | f is subset of P, f appears in a flow}

수식 1. 제안하는 시그니쳐의 구조

한 응용에 대한 공통 부분 문자열 중 일정 빈도수가 넘는 9 개의 Content 시그니쳐를 추출하였을 때 $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$와 같이 표현할 수 있다. 이 중 동일한 Packet 내에 존재하는 Content 시그니쳐의 집합 중 일정 빈도 수가 넘는 Packet 시그니쳐가 4 개가 추출되었을 때 $P = \{p_1, p_2, p_3, p_4\}$이며 이는 $P = \{p_1, p_2, p_3, p_4\} = \{\{c_1, c_3, c_9\}, \{c_2, c_5, c_9\}, \{c_8, c_4\}, \{c_6\}\}$과 같다. 또한 Flow 시그니쳐가 2 개 추출이 되었을 때는 $F = \{f_1, f_2\}$라 표현하며 $F = \{f_1, f_2\} = \{\{p_1, p_3\}, \{p_4\}\} = \{\{\{c_1, c_3, c_9\}, \{c_8, c_4\}\}, \{\{c_6\}\}\}$과 같은 의미이다.

표 3, 표 4, 표 5 는 제안한 시그니쳐의 구조로 추출한 Content 시그니쳐, Packet 시그니쳐, Flow 시그니쳐의 예시이다.

$Signature_{id}$	Content Signature
1	<GET />
2	<HTTP /1.1>
3	<MUSIC>
4	<CACHE-CONTROL>
5	<NO>
6	<FACEBOOK>
7	<MUSIC>
8	<ALBUM>
9	<32>

표 3. Content 시그니쳐의 예시

$Signature_{id}$	Packet Signature
1	< <GET /> <MUSIC> <32> >
2	< <HTTP /1.1> <NO> <32> >
3	< <ALBUM> <CASHE-CONTROL> >
4	< <FACEBOOK> >

표 4. Packet 시그니쳐의 예시

$Signature_{id}$	Flow Signature
1	< < <GET /> <MUSIC> <32> > < <ALBUM> <CASHE-CONTROL> > >
2	<<<FACEBOOK>>>

표 5. Flow 시그니쳐의 예시

표 3 의 Content 시그니쳐 6 과, Packet 시그니쳐 4 와 표 5 의 Flow 시그니쳐 2 는 모두 “FACEBOOK” 이라는 문자열을 나타내는 시그니쳐이다. 응용에 대한 분석률은 모두 같을지라도 Content 시그니쳐 6 보다 Packet 시그니쳐 4 가, Packet 시그니쳐 4 보다 Flow 시그니쳐 2 가 더 응용에 정교하다고 할 수 있다.

한편, Packet 시그니쳐를 사용하거나 Flow 시그니쳐를 사용할 시 그림 2 에서 표현한 분석률의 한계가 존재할 수 있다. 하나의 Packet 시그니쳐가 매칭되기 위해서는 그 Packet 시그니쳐를 생성하기 위해 사용된 Content 시그니쳐도 모두 매칭되어야 하기 때문이다. 일반적으로 분석률은 수식 2 와 같이 계산할 수 있다. 트래픽의 양으로는 Flow 의 수 혹은 Packet 의 수나 용량(Bytes)를 사용할 수 있다.

$$\text{분석률} = \frac{\text{시그니쳐가 매칭된 트래픽의 양}}{\text{전체 트래픽의 양}} \times 100$$

수식 2. 일반적인 분석률의 계산식

그러나 Packet 시그니쳐 혹은 Flow 시그니쳐의 분석률은 수식 2 와는 다른 방법으로 계산되어야 한다. 상위 단계 시그니쳐는 하위 단계 시그니쳐의 집합으로 생성되었기 때문에 하위 단계 시그니쳐가 분석한 트래픽은 상위 단계의 시그니쳐도 분석을 할 수 있다 해도 무방하다. 예를 들어, 3 개의 Content 시그니쳐 A, B, C 가 하나의 Packet 시그니쳐를 생성하는데 사용되었다면 이 Packet 시그니쳐의 분석률은 A, B, C 가 매칭된 트래픽의 양을 단순히 더하는 것이 아닌 분석한 트래픽들의 합집합의 양을 전체 트래픽의 양으로 나눈 것의 백분율로 계산할 수 있다. A 와 B 와 C 가 각각 분석한 트래픽이 서로 중복될 수 있기 때문이다.

$$\text{분석률} = \frac{A \cup B \cup C\text{의 양}}{\text{전체 트래픽의 양}} \times 100$$

수식 3. Packet 시그니쳐의 계산식의 예

따라서, 응용 트래픽 분석에 있어 상위 단계의 시그니쳐를 사용할 시 하위 단계 시그니쳐의 분석률을 그대로 유지하면서 오탐률은 낮추어 특정 응용을 더 정확히 분석할 수 있다.

상위 단계의 시그니쳐를 다수 보유할 시 대용량의 트래픽을 효율적으로 분석할 수 있다. 대용량의 트래픽을 발생하는 네트워크의 효율적인 분석을 위해서는 Flow 단위의 트래픽 수집과 분석이 필요하다[9]. 상위 단계의 시그니쳐일수록 특정 응용에 정교하므로 생성되기 위한 조건이 까다롭다. 동일한 단위(Packet 또는 Flow) 내에 하위 단계 시그니쳐들이 모두 매칭되어야 하며 이로써 만들어진 하위 단계 시그니쳐들의 집합이 일정 이상의 빈도 수를 만족해야 시그니쳐로 생성되기 때문이다. 따라서, 상위 단계의 시그니쳐 구조를 사용할 시 시그니쳐의 개수를 줄이면서 특정 응용에 더 정교한 시그니쳐가 생성된다.

5. 실험 및 결과

본 장에서는 제안한 페이로드 시그니쳐의 구조와 선행 연구의 페이로드 시그니쳐 구조를 특정 응용 트래픽에 적용하여 그 실효성을 증명한다.

실험 환경은 다음과 같다. 먼저 수집된 트래픽을 TMA(Traffic Measurement Agent)를 이용하여 2 가지 응용에 대한 정답지 트래픽을 생성한다. 본 실험에서는 2 가지 응용을 Naver 와 Yahoo 로 선정하였다. 2 가지 응용의 정답지 트래픽을 바탕으로 각각 응용들의 Content 시그니쳐와 Packet 시그니쳐를 추출한다. 추출한 2 가지 응용의 각 시그니쳐를 2 가지 응용에 대한 정답지 트래픽에 반대로 적용하여 표 6 의 (2)와 (3)을 구한다. (1)과 (4)는 2 가지 응용의 시그니쳐로 각 응용에 해당하는 정답지 트래픽에 적용한 분석률이다. 최종적으로 (1), (2), (3), (4)의 값을 가지고 Content 시그니쳐와 Packet 시그니쳐의 분석률과 오탐률을 비교 분석한다.

	Naver.Sig	**Yahoo.Sig**
Naver.GTT	(1) TP_{Naver} $= TN_{Yahoo}$	(2) FN_{Naver} $= FP_{Yahoo}$
Yahoo.GTT	(3) FP_{Naver} $= FN_{Yahoo}$	(4) TN_{Naver} $= TP_{Yahoo}$

표 6. 실험 환경

표 6 에서 G.T.T 는 정답지 트래픽을 의미하고, Sig 는 시그니쳐를 의미한다. (1)은 Naver 의 시그니쳐를 Naver 의 정답지 트래픽에 적용한 분석률을 나타내며 이는 Naver 응용 분석에 대한 TP(True Positive)이다. (2)는 Yahoo 의 시그니쳐를 Naver 의 정답지 트래픽에 적용한 분석률을 나타내며 Yahoo 응용 분석에 대한 FP(False Positive), 즉 오탐률이다. (3)은 Naver 의 시그니쳐를 Yahoo 의 정답지 트래픽에 적용한 분석률을 나타내며 Naver 응용 분석에 대한 FP 이고 (4)는 Yahoo 의 시그니쳐로 Yahoo 의 정답지 트래픽을 분석한 분석률을 나타내며 Yahoo 응용에 대한 TP 이다.

	Content Signature		Packet Signature	
	TP	FP	TP	FP
Naver	5,567 /5,739 **97%**	403 /3,847 **10%**	5,459 /5,739 **95%**	205 /3,847 **5%**
Yahoo	3,774 /3,847 **98%**	635 /5,739 **11%**	3,740 /3,847 **97%**	309 /5,739 **5%**

표 7. 실험 결과

표 7 은 실험 결과로 모든 값은 트래픽의 Flow 단위이다. 본 실험 결과에서 Naver 와 Yahoo 의 응용에 대해 모두 Content 시그니쳐의 오탐률보다 Packet 시그니쳐의 오탐률이 더 낮았으며 Content 시그니쳐에 비해 Packet 시그니쳐의 분석률은 변화가 적었다.

6. 결론 및 향후 연구

본 논문에서는 다른 응용프로그램과 중복되지 않도록 특정 응용프로그램에 정교한 페이로드 시그니쳐의 구조를 제안하였다. 이를 통해 응용별 시그니쳐 추출시 시그니쳐 중복을 방지하고 응용 트래픽 분석에서 오탐률을 감소시킴으로 인해 정확도를 향상시킬 수 있다. 따라서 분석 결과에 신뢰성을 부여하여 효율적인 네트워크 관리가 가능하다. 실험을 통해 제안한 시그니쳐 구조와 선행 연구의 시그니쳐 구조의 TP(True Positive) 및 FP(False Positive)의 수치를 비교 분석한 결과, TP(True Positive)는 유지시키고 FP(False Positive)의 수치는 감소시킴으로써 본 페이로드 시그니쳐 구조의 실효성을 증명하였다.

향후 연구로써는 제안한 페이로드 시그니쳐의 구조를 실시간 트래픽 분류에 활용할 수 있도록 자동 페이로드 생성 시스템에 적용하고 이를 관리 및 검증할 수 있는 방안에 대한 연구를 통해 본 시그니쳐의 실용성을 확보하고자 한다. 또한 본 페이로드 시그니쳐 구조를 사용하여 추출한 시그니쳐들을 최적화할 수 있는 적절한 임계값 산출 방법을 연구하고자 한다.

참고 문헌

[1] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, “Lightweight, Payload-Based Traffic Classification An Experimental Evaluation,” IEEE International Conference on Communications, Beijing, China, May. 19-23, pp. 5869-5875, 2008.

[2] Liu, Hui Feng, Wenfeng Huang, Yongfeng Li, Xing “Accurate Traffic Classification”, Networking, Architecture, and Storage, NAS 2007. International Conference

[3] H.-A. Kim and B. Karp, “Autograph: Toward automated, distributed worm signature detection,” in *USENIX Security Symp.*, vol. 286, 2004.

[4] J. Newsome, B. Karp, and D. Song, “Polygraph: Automatically generating signatures for polymorphic worms,” *IEEE Symp. Security and Privacy*, pp. 226-241, 2005.

[5] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, “Towards automated application signature generation for traffic identification,” *IEEE Network Operations and Management Symp. (NOMS 2008)*, pp. 160-167, 2008.

[6] X. Feng, X. Huang, X. Tian, and Y. Ma, “Automatic traffic signature extraction based on Smith-waterman algorithm for traffic classification,” *IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, pp. 154-158, 2010.

[7] 박준상, 윤성호, 박진완, 이현신, 이상우, 김명섭, "페이로드 시그니쳐 기반 응용 레벨 트래픽 분류 시스템 성능 향상에 관한 연구," KNOM

Review, Vol. 12, No. 2, Dec. 2009, pp. 12-21.
[8] S.-H. Lee, J.-S. Park, S.-H. Yoon, and M.-S. Kim , "High performance payload signature-based Internet traffic classification system ," *Proc. of the Asia-Pacific Network Operations and Management Symposium (APNOMS) 2015, Busan, Korea*, Aug. 19-21, 2015, pp.491-494.
[9] 윤성호, 박준상, 김명섭, "멀티 코어 환경에서 실시간 트래픽 분석 시스템 처리속도 향상", 한국통신학회 논문지 '12-05 Vol.37B No.5, pp. 348-356, May 2012.
[10] C. MU, X.-h. HUANG, X. TIAN, Y. MA, and J.-l. Qi, "Automatic traffic signature extraction based on fixed bit offset algorithm for traffic classification," *The J. China Universities of Posts and Telecommun.*, vol. 18, pp. 79-85, 2011.
[11] M. Ye, K. Xu, J. Wu, and H. Po, "AutoSig-Automatically Generating Signatures for Applications", *Proc. Of IEEE 9th International Conference on Computer and Information Technology (ICTI)*, Xiamen, China, October 11 – 14 , 2009 .

연속된 플로우 정보를 이용한 CDN 서비스 트래픽 분류 방법 연구
Study on CDN Traffic Classification using Cascade Flow Information

이수강 O, 심규석, 정우석, 김명섭

고려대학교 컴퓨터정보학과

{sukanglee O, kusuk007, hary5832, tmskim}@korea.ac.kr

요 약

인터넷 속도의 증가에 힘입어 인터넷을 이용하는 다양한 웹 서비스가 개발됨에 따라 이들이 발생시키는 인터넷 트래픽의 양이 급격히 증가하고 있다. 이러한 웹 서비스의 트래픽에서 제공하는 컨텐츠는 대부분 CDN(Contents Delivery Network)를 통해 전송된다. 기존 트래픽 분류 방법은 단일 플로우를 대상으로 트래픽을 분류하기 때문에 CDN 트래픽을 정확히 분류할 수 없다. 본 논문에서는 CDN 트래픽 분류의 한계를 해결하기 위해 연속된 플로우 연관관계를 이용하여 트래픽을 분류하는 방법을 제안한다. 제안하는 방법은 특정 서비스를 사용할 때 발생하는 연속된 플로우들 간 나타나는 연관관계를 추출하여 시그니쳐로 사용하는 것이다. 제안하는 방법을 이용하여 시그니쳐를 만들고, 실제 트래픽에 적용한 결과 기존 트래픽 분류 방법으로는 unknown 트래픽으로 분류되던 CDN 트래픽이 각 웹 서비스로 정확히 분류 할 수 있었다.

Keyword : Cascade flow information, Traffic association rule mining, CDN traffic classification

1. 서론

인터넷 속도의 증가에 힘입어 인터넷을 이용하는 다양한 웹 서비스가 개발됨에 따라 서비스를 이용하는 사용자와 이들이 발생시키는 인터넷 트래픽의 양이 급격히 증가하고 있다. 이에 따라 2000 년대 초반부터 급격히 늘어나는 트래픽에 대응하기 위한 트래픽 분산 기술이 연구되어 왔으며, CDN (Content Delivery Network)은 트래픽 분산 기술 중 가장 보편적으로 사용되고 있는 기술이다.

CDN[1]은 기존의 컨텐츠 전송 과정에서 빈번하게 발생하는 트래픽 집중 및 병목현상, 지연, 끊김과 같은 문제를 해결하기 위해 등장한 개념으로 사진, 동영상과 같은 대용량의 컨텐츠를 사용자 근처에 미리 옮겨놓고 그곳에서 콘텐츠를 사용자들에게 신속하게 배달하는 것이다. 국내에서는 KT, SK Broadband, LG U+와 같은 ISP 사업자들이 CDN 서비스를 제공하고 있다. 실제 국내 포털 사이트들이 제공하는 대부분의 컨텐츠는 컨텐츠를 제공하는 주체와 물리적으로 다른 곳에 위치한 CDN 서버에서 전송되고 있다.

CDN 트래픽은 그림 1 과 같이 발생한다. 사용자는 웹 포털에서 동영상과 같은 특정 컨텐츠를 요청하면 해당 포털에서 사용하는 CDN 업체의 서버에서 컨텐츠를 제공받는다. 이러한 상황에서 네트워크 관리자는 사용자가 컨텐츠를 요청할 때 발생하는 플로우(A)와 실제 컨텐츠가 사용자에게 제공될 때 발생하는 플로우(B)를 모두 분류할 수 있어야 한다. 기존의 트래픽 분류 방법으로는 플로우(A)가 어떤 포털인지 구분할 수 있지만 플로우(B)가 어떤 포털의 어떤 컨텐츠를 가지고 있는지를 정확하게 알 수 없기 때문에 특정 응용이나 서비스로 분류하기가 까다롭다. 따라서 플로우(A)와 플로우(B)가 같은 응용이나 서비스에 의해 발생한 트래픽이지만 두 플로우를 같은 응용이나 서비스로 분류할 수 없다.

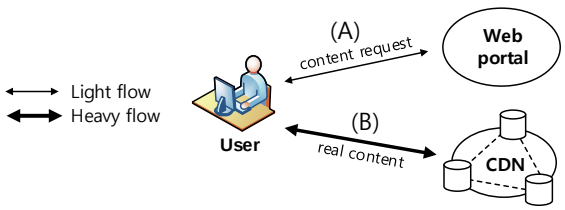


그림 1. CDN 트래픽 발생 예

본 논문에서는 연속되어 발생하는 플로우의 연관관계를 분석하여 CDN 트래픽이 어떤 컨텐츠 제공자에 의해 발생하는지를 분석하고, 이를 규칙화 하여 해당 컨텐츠 제공자가 제공하는 서비스와 함께 분류하는 방법을 제안한다.

본 논문은 다음과 같은 순서로 기술한다. 2 장에

이 논문은 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2015R1D1-A3A01018057).

서는 기존에 제시된 트래픽 분류 방법 설명하고 3 장에서는 CDN 트래픽 분류 방법을 제안한다. 4 장에서는 본 논문의 결론과 향후 연구를 제시한다.

2. 관련 연구

인터넷 트래픽 분류는 그 중요성이 증가함에 따라 다양한 분류방법이 제시되고 있다. 가장 원시적인 포트 기반 분석은 Internet Assigned Number Authority(IANA)[2] 에 등록된 포트 정보를 이용하여 트래픽을 분류한다. 초기 인터넷에서는 고정적인 포트 번호와 대응하는 서비스(HTTP(80), SSH(22), FTP(20, 21), e-mail(25,110))가 트래픽 대부분을 차지하였기 때문에 이를 기준으로 정확한 트래픽 분석이 가능하였다. 하지만 시간이 지남에 따라 인터넷을 사용하는 서비스가 다양해지고, 사용되는 포트 번호가 임의의 포트가 사용되므로 포트번호로는 정확한 트래픽 분류를 할 수 없게 된다. 포트 기반 분석의 한계점을 극복하기 위해 제시된 시그니쳐 기반의 인터넷 트래픽 분류 방법은 현재까지 보안, 응용 트래픽 분류 등의 다양한 분야에서 활용되고 있다.

시그니쳐 기반 트래픽 분류 방법은 패킷 또는 플로우의 특정 위치에서 응용을 식별하기 위한 고유한 정보를 추출하고 이를 기반으로 분류하는 방법이다. 시그니쳐는 시그니쳐 추출에 사용되는 정보에 따라 Header[3], Payload[4,5,6], Statistic[7,8], Behavior[9,10] 시그니쳐로 구분할 수 있다.

3. CDN 트래픽 분류 방법

지금까지 연구되었던 트래픽 분류 방법들은 단일 플로우를 대상으로 트래픽 분류를 하기 때문에 CDN 서비스를 이용할 때 발생하는 트래픽을 분류하기 위해서는 플로우 간 연관관계를 고려하여 트래픽을 분류해야 한다. 이를 위해 본 장에서는 연속된 플로우 정보를 이용한 트래픽 분류 방법을 제안한다.

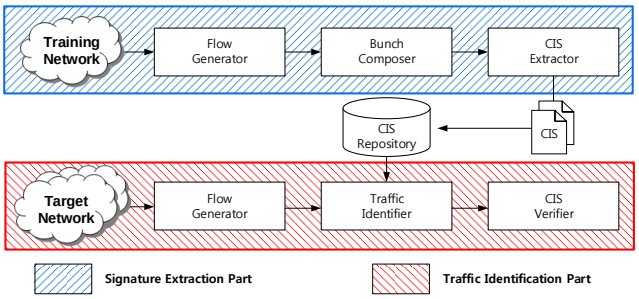


그림 2. Entire structure of identification system

그림 2 는 본 논문에서 제안하는 연속된 플로우 정보를 이용한 트래픽 분류 시스템의 구조를 나타낸다. 해당 시스템은 크게 2 개의 파트로 구성되며, 시그니쳐 추출부(Signature Extraction Part)에서는 트래픽 데이터를 플로우 데이터 형태로 변환하고, 플로우를 Bunch 로 구성한다. Bunch 로 구성한 후 Bunch 별 헤더 정보(IP, Port, Protocol)를 추출하여 CIS(Cascade Information Signature) 형태로 시그니쳐를 생성한다. 트래픽 분석부(Traffic Identification Part)에서는 트래픽 데이터를 플로우 데이터 형태로 변환하고, 생성된 CIS 와 Flow 데이터를 매칭하여 트래픽을 분류한다. 최종적으로 본 논문에서 제안하는 분류 방법의 결과를 기존의 트래픽 분류방법으로 분류한 결과와 비교하게 된다.

CIS(Cascade Information Signature) 추출 단계에서는 플로우의 헤더 정보를 사용한다. 플로우의 헤더 정보로 목적지 IP, 목적지 포트 번호, 프로토콜 번호를 사용한다.

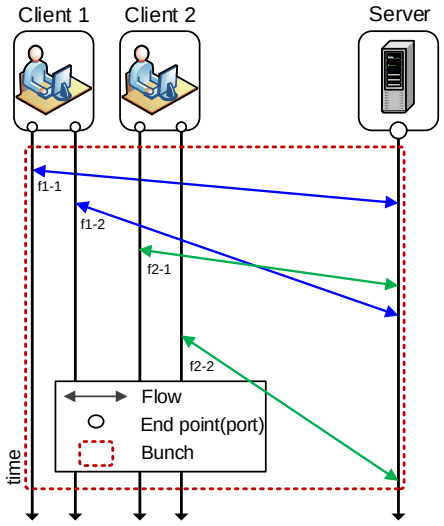


그림 3. Bunch 의 구조

일반적으로 사용자가 특정 서버에 접속하여 인터넷을 사용할 때, 단일 호스트와 단일 서버사이에서 발생하는 트래픽에서 나타나는 포트 번호와 여러 호스트와 서버와의 관계를 N:1 관계로 표현할 수 있다. 즉 여러 호스트가 여러 개의 포트번호를 사용하여 고정된 서버 포트에 접속한 다는 것이다. 이러한 서버와 다수의 호스트에서 발생하는 여러 플로우를 서버 정보가 동일한 것들을 모아서 Bunch 라는 플로우 집합을 정의하였다. Bunch 는 목적지(서버) IP, 포트 번호, 프로토콜이 같은 플로우들로 구성된다.

그림 3 은 Bunch 의 구조를 설명하기 위한 그림으로 Client1 과 Client2 가 특정 서버(IP, 포트, 프로토콜 번호)로 접근할 때 발생하는 플로우(f1-1, f1-2, f2-1, f2-2)는 각각 파란색(Client1), 녹색(Client2)으로 표현하였다. 그림 2 의 경우 Client1, 2 가 발생시킨 플로우들은 하나의 Bunch 로 구성되며, 이것은 목적지(서버)의 IP, 포트 번호, 프로토콜이 서로 같다는 것이다.

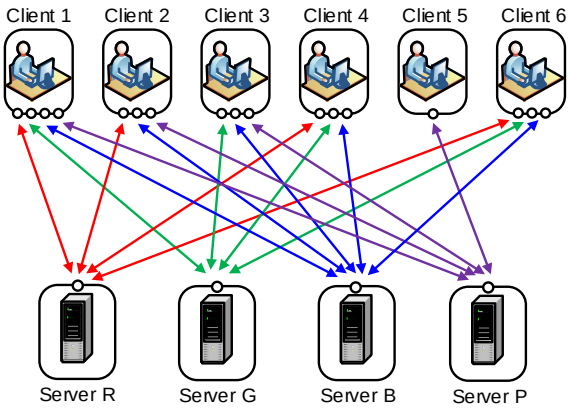


그림 4. 특정 사이트의 서비스에서 발생하는 플로우 예

그림 4 는 특정 사이트의 서비스 이용 시 클라이언트(1 ~ 6)과 서버(R, G, B, P) 사이에서 발생하는 플로우를 나타낸 그림이다. 발생한 플로우는 화살표로 표현되어 있다. 그림 4 의 경우 각 플로우들은 서버 기준으로 Bunch 가 구성되며 CIS 생성단계에서는 발생한 플로우를 바탕으로 구성된 Bunch 간 연관관계를 분석하여 규칙을 찾아낸다. Bunch 간 연관관계 분석을 통해 각 Bunch 의 Support(지지도), Confidence(신뢰도), 그리고 최종적으로 Lift(향상도) 값을 계산하여 특정 서비스를 사용할 때 발생하는 연속적인 플로우를 찾아낼 수 있다. 아래 수식은 Support, Confidence, Lift 를 계산하는 수식이다.

$$Support(A \Rightarrow B) = P(A \cap B) \quad (1)$$

$$Confidence(A \Rightarrow B) = P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (2)$$

$$Lift(R \Rightarrow B) = \frac{P(B|R)}{P(B)} = \frac{P(R \cap B)}{P(B) \cdot P(R)} \quad (3)$$

Support(R⇒B)는 전체 트래픽에서 서버 R 의 플로우와 서버 B 의 플로우가 얼마나 같이 나타나는지를 의미하며 수식 (1)로 정의된다. Confidence(R⇒B)는 전체 트래픽에서 서버 R 의 플로우가 나타났을 때 해당 트레이스에서 서버 B 의 플로우도 나타나는 조건부 확률을 의미하며 수식(2)로 정의된다. Lift(R⇒B)는 서버 R 의 플로우 출현에 상관없이 서버 B 의 플로우가 나타나는 확률에 비해 서버 R 의 플로우가 나타날 경우 서버 B 의 플로우가 나타나는 확률의 증가 비율이며 수식(3)으로 정의된다. 즉, 두 서버의 플로우가 출현하는 것이 서로 관련이 없는 경우에는 Lift ≈1 이며, Lift > 1이면 두 서버 플로우의 출현은 서로 연관관계가 크다는 것이다. Lift < 1 인 경우 두 서버 플로우의 출현은 서로 반대의 관계가 있다는 것을 의미한다.

표 1 은 그림 4 의 플로우를 발생현황을 나타낸 것이다. 서버 R 과 B 의 Support 는 0.66 이며 Confidence 는 1 이 된다. 최종적으로 두 서버 R 과 B 의 Lift 는 1.2 가 되며 Lift > 1이므로 두 서버 플로우의 출현은 서로 연관관계가 크다는 것이다. 이와 같이 각 Bunch 의 모든 경우의 수를 계산하고 난 후 Lift > 1 을 만족하는 경우는 (R⇒B), (R⇒G), (B⇒G), ({R,B}⇒G)이다. Lift > 1 을 만족하는 경우를 시그니쳐로 표현하기 위해 DAG(Directed Acyclic Graph)를 사용하며, 위상정렬 알고리즘을 이용하여 연관관계 규칙 추출 결과를 그래프 형태의 그림 5 처럼 나타낼 수 있다.

표 1. 그림 4 의 플로우 발생 현황

Server	R	B	G	P
Client 1	O	O	O	O
Client 2	O	O		O
Client 3		O	O	O
Client 4	O	O	O	
Client 5				O
Client 6	O	O	O	

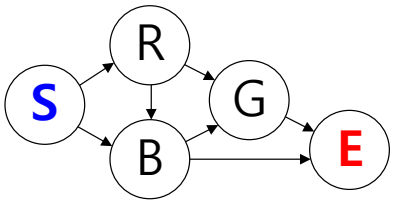


그림 5. 추출된 연관관계 규칙

표 2 는 그림 5 와 같이 추출된 연관관계 규칙을 DAG 형태의 시그니쳐로 나타내기 위한 위상정렬 알고리즘의 코드이다.

표 2. CIS 생성을 위한 위상정렬 알고리즘

```
1:  topologicalSort(R)
2:  Input : R is rule about cascade flow information
3:  For(i=1; i=R.count(); i++)
4:      Find node its indegree is zero → b
5:      B[i] = b
6:      Delete "b" node and out-coming edge of "b"
7:  End For;
8:  Return B[];
```

표 2 에서는 추출된 연관규칙을 입력으로 사용하며, 연관규칙의 모든 노드를 검사하여 DAG 형태의 그래프를 생성한다. 표 2 의 4 번째 줄은 해당 노드가 진입간선의 개수(incoming edge count, indegree)가 0 인 검색하고 선택하는 부분이며, 5 번째줄은 선택된 노드 b 를 B 에 저장하는 것이다. 6 번째 줄은 선택된 노드 b 의 모든 진출간선(outcoming edge)과 노드 b 를 삭제하는 것이다. 이와 같은 방법으로 모든 노드를 탐사하여 위상정렬이 완료된 그래프가 생성되며 이를 CIS 로 트래픽 분류에 사용한다.

5. 실험 및 결과

본 장에서는 제안하는 방법의 타당성 및 성능을 검증하기 위해 본 논문에서 제안하는 방법으로 CDN 서비스를 이용하는 각 웹서비스들의 CIS 를 생성하였다. 그리고 생성된 CIS 를 실제 학내 망에서 발생하는 트래픽 분류에 적용해보았다. 실험 대상으로 선정한 4 개 서비스는 Naver Sports, pooq, Daum Sports, Daum TV 이다. 최종적으로 CIS 의 CDN 트래픽 분류 성능을 확인하기 위해 CIS 를 이용한 트래픽 분류 결과와 페이로드 시그니쳐를 이용한 트래픽 분류 결과를 비교한다.

표 3. CIS 생성에 사용된 트래픽 정보

	Packet	Flow	Byte(MB)	CIS
Naver Sports	29,564,624	8,529	800	246
pooq	4,298,174	3,574	340	128
Daum Sports	6,438,894	5,321	533	157
Daum TV	5,165,127	4,268	412	201

표 3 은 각 웹서비스의 시그니쳐를 생성하기 위한 트래픽의 정량적 수치와 생성된 CIS 의 개수이다. 해당 트래픽을 이용하여 시그니쳐를 생성한 결과 각 응용별 246, 128, 157, 201 개의 연관관계 규칙이 추출되었고, 이를 이용하여 CIS 를 생성하였다.

생성된 CIS 시그니쳐와 페이로드 시그니쳐를 실제 학내망 트래픽 분류에 적용한 결과 각 웹 서비스에서 발생하는 CDN 서비스 트래픽 탐지에서 많은 차이가 있었다. 그림 6 은 CIS 와 페이로드 시그니쳐의 분류 결과를 그래프로 나타낸 것이다. 실험 결과 기존 페이로드 시그니쳐로 트래픽을 분류할 때, 각 웹 서비스에서 발생된 CDN 트래픽이 대부분 Unknown 트래픽으로 분류되었다. 그러나 본 논문에서 제안한 CIS 를 이용하면 Unknown 으로 분류된 트래픽이 각각 웹 서비스 별로 분류가 된다. 결론적으로 CDN 서비스를 이용하는 웹 서비스의 트래픽에서 실제 컨텐츠를 전송하는 CDN 트래픽을 각 웹 서비스 별로 분류할 수 있다는 것이다.

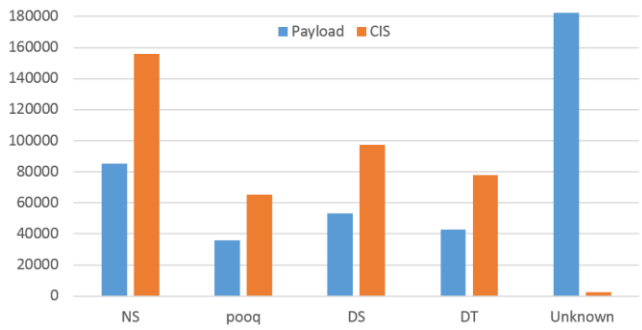


그림 7. CDN 트래픽 분석 결과 비교

6. 결론 및 향후 연구

본 논문에서는 CDN 트래픽을 분류할 때 기존의 트래픽 분류 방법의 한계점을 극복하기 연속된 플로우의 특징을 이용한 트래픽 분류 방법을 제안하였다. 연속되어 나타나는 플로우들의 특징을 시그니쳐로 추출하기 위한 방법을 연구하였으며 해당 시그니쳐를 CIS 로 정의하였다. 본 논문에서 제안한 CIS 를 사용한 결과 기존 방법으로는 Unknown 으로 분류된 대부분의 CDN 트래픽이 각 웹 서비스로 분류되었다. 결론적으로 본 논문에서 제안한 연속된 플로우 정보를 이용한 트래픽 분류 방법은 그동안 분류하지 못했던 CDN 트래픽을 각 웹 서비스 별로 분류 할 수 있다는 것이다.

향후 연구 계획으로는 CIS 를 이용한 트래픽 분류에 정확도 측면까지 고려한 방법을 연구할 계획이다.

참고 문헌

[1] https://en.wikipedia.org/wiki/Content_delivery_network

[2] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, “Internet traffic classification demystified: myths, caveats, and the best practices,” in Proceedings of the 2008 ACM CoNEXT conference, 2008, p.11

[3] Sung-Ho Yoon, Jun-Sang Park, and Myung-Sup Kim, "Signature Maintenance for Internet Application Traffic Identification using Header Signatures," Proc. of the 4th IEEE/IFIP International Workshop of the Management of the Future Internet (ManFI 2012), Hwaii, USA, Apr. 16, 2012.

[4] R. Antonello, S. Fernandes, D. Sadok, J. Kelner, “Characterizing Signature Sets for Testing DPI Systems”, Proc. IEEE GLOBECOM Management of Emerging Networks and Services Workshop, Houston, TX, USA, pp. 678-683, Dec. 2011.

[5] Y. Wang, Y. Xiang, W. L. Zhou, and S. Z. Yu, "Generating regular expression signatures for network traffic classification in trusted network management," Journal of Network and Computer Applications, vol. 35, pp. 992-1000, May 2012.

[6] N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, "Application traffic classification at the early stage by characterizing application rounds," Information Sciences, vol. 232, pp. 130-142, May 2013.

[7] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, “A modular machine learning system for flow-level traffic classification in large networks,” ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 1, pp. 1-34, March, 2012.

[8] An, H. M., Lee, S. K., Ham, J. H., & Kim, M. S. (2015). Traffic Identification Based on Applications using Statistical Signature Free from Abnormal TCP Behavior. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING,31(5), 1669-1692.

[9] Sung-Ho Yoon, Myung-Sup Kim, "Behavior Signature for Big Data Traffic Identification" Proc. of the International Conference on Big Data and Smart Computing (BigComp) 2014, Bangkok, Thailand, Jan. 15-17, 2014, pp. 261-266.

[10] Sung-Ho Yoon, Jun-Sang Park, and Myung-Sup Kim, "Behavior Signature for Fine-grained Traffic Identification," Applied Mathematics & Information Sciences Vol. 9, No. 2L, , Apr. 2015, pp. 523-534.

RBAC 기반의 캠퍼스 출입자 권한 제어방법

임이삭°, 김현우, 주홍택

계명대학교 컴퓨터공학과

{islim2254, hwkim84, juht}@kmu.ac.kr

요 약

역할기반 접근통제(Role-Based Access Control)란 사용자의 시스템 자원에 대한 접근을 역할을 기반으로 하여 통제하는 방법을 말한다. 데이터가 늘고 네트워크 시스템이 발전함에 따라 자원에 접근하는 사용자의 통제를 위해서는 접근통제의 기본 모델이 필요하다. 정의된 접근통제 모델 중 RBAC 를 제안하고 사용하여 기존의 단점의 보안과 효율적인 서비스를 제공하고 외부의 사용자 계정 관리 서비스를 연동하여 JSON 형식의 메시지를 주고받는 RBAC 기반의 캠퍼스 출입자 권한 제어방법을 제시한다. 본 논문에서는 RBAC 의 정의를 기반으로 출입자 권한 제어방법을 제시하고 출입자 모델을 정의 함으로써 사용자 계정관리 서비스 API 를 통해 캠퍼스 출입자 권한 제어방법을 구현하였다.

1. 서론

데이터들이 늘어나고 네트워크 시스템이 발전함에 따라 데이터에 접근하는 사용자들이 늘어나고 있다. 점점 복잡해지는 데이터 들의 관계 속에서 데이터를 사용자들을 통제하기 위해 이해하기 쉽고 관리자에게 편리한 관리기법이 제공된다.

접근통제정책에는 규칙기반 접근통제정책의 강제적 접근통제(MAC)과 신분기반 정책의 임의적 접근통제(DAC)에 비하여 정교함과 유연성을 제공하며 기존의 단점들을 보완한 RBAC (Role-Based Access Control)가 등장하였고 이를 기반으로 사용자들의 접근통제를 관리하는 기관들이 늘고 있다[1].

최근 들어 캠퍼스 내의 출입은 모든 이들로부터 자유롭게 개방되어있으며 내부의 공간에 관하여 특별한 제제 없이 누구든 출입이 가능하다. 이러한 외부의 방문자들의 출입이 증가하면서 캠퍼스 내 연구실과 같은 출입이 통제된 구역에서의 사건 사고들이 발생이 늘어나고 있다. 이런 일들을 방지하기 위해서 캠퍼스 내에서 출입 통제 관리 시스템을 적용시켜 쉽고 안정적인 RBAC 기반 시스템을 설계한다.

본 논문에서는 외부 사용자 계정 관리 API 서비스를 접목하여 효과적인 관리와 외부 사용자 계정관리 API 서비스를 통하여 RBAC 기반의 캠퍼스 출입자 권한 제어 방법을 구현한다.

본 논문의 구성은 다음과 같다. 2 장에서는 RBAC 에 대하여 관련된 논문에 대해 기술한다. 3 장에서는 본 논문에서 제안하는 RBAC 기반 캠퍼스 출입자 모델 정의에 대하여 설명한다. 4 장에서는 RBAC 기반 출입자 권한 제어 방법에 대하여 설명한다. 5 장에서는 RBAC 의 모듈을 활용하여 캠퍼스의 출입자 권한 제어 시스템을 구현하고자 한다. 마지막으로 6 장에서는 결론 및 향후 연구에 대해 논의한다.

2. 관련 연구

Sandhu 외 2 명은 RBAC 표준 참조 모델은 기본 모델인 Flat RBAC, Hierarchical RBAC, Constrained RBAC, Symmetric RBC 와 같이 4 단계의 모델들을 정의하였다[2].

노승민 외 4 명은 RBAC 에 기반하여 현재의 의료 및 질환 정보 관리에 적용시켜 각 정보 개체들과 사용자 간의 효율적인 역할 분담과 정보보호를 위한 시스템을 설계 및 구현하여 실제 시스템에 적용하였다[3].

김지헌은 헬스 케어 시스템의 의료정보 유출과 불법 수정으로 인해 발생할 수 있는 피해를 줄이기 위해 적합한 확장된 RBAC 구현을 제안한다[4].

김진식 외 4 명은 다른 어플리케이션에 쉽게 추가되거나 독자적으로 인증 기능을 가지는 계층적 RBAC 서버에 사용될 수 있는 RBAC 서버 API 를 데이터베이스를 이용하여 설계 및 구현 하였다[5].

3. RBAC 기반 출입자 제어 모델 정의

RBAC 은 사용자(Users)와 역할(Roles), 허가(Permissions)로 구성되어 있다. 사용자는 시스템을 통하여 시스템내의 정보를 사용하는 객체로서 한 사용자는 한 명의 사람에 대응된다. 역할은 접근제어 정책을 구현하는 중요한 의미적 구조로서, 조직 내의 직급을 나타내며 고유의 권한과 의무를 가지고 사용자와 권한 사이에서 중재역할을 하면서 인증되지 않은 사용자가 특정 객체에 접근하는 것을 통제한다. 허가는 시스템의 하나 또는 그 이상의 객체에 대한 특정 접근의 승인을 나타낸다. RBAC 기반의 출입자 제어 모델은 다음과 같이 정의된다.

USERS = {고유 ID}
ROLES = {교수, 학생, 교직원..}
PERMISSIONS = {강의실, 도서관, 전산실..

RBAC 기반 출입자 제어 모델에서 USERS 는 사용자의 고유 ID 를 의미한다. ROLES 은 교수, 학생과 같은 사용자의 역할을 정의하며. 이러한 역할들은 역할 고유의 PERMISSIONS 을 가진다. 예를 들면 사용자 "S0001"이 학생일 경우 사용자는 "학생"의 역할에 속하게 된다. 역할이 포함한 PERMISSIONS 은 역할을 가진 사용자가 권한을 가진 공간 "강의실", "도서관"을 나타내며 해당 공간의 출입이 가능하다.

4. RBAC 기반 출입자 권한 제어 방법

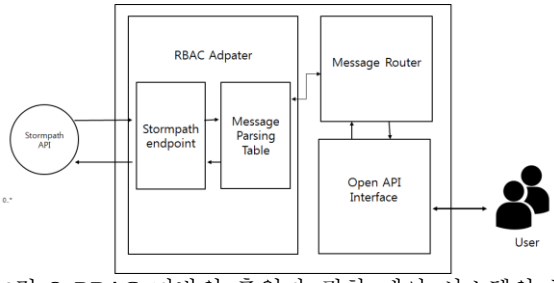
RBAC 은 단순히 출입자 권한 제어 설계 만이 아닌 외부 사용자 관리 서비스 API Stormpath 를 사용하였고 데이터 모델의 구조에 의해서 관리, 제어된다. Stormpath 는 Application, Directories, Groups, Accounts 네 가지의 컨테이너 자원으로 구성되어 있다. Application 은 Stormpath 와 REST API 를 통해 통신되는 실제 소프트웨어에 대한 정보를 나타낸다. Directory 는 Stormpath 내 에서 최상위 컨테이너 자원이며 하나의 Application 안에 포함되고 RBAC 이 접근하기 위한 Resource 에 해당한다. Account 는 RBAC 의 사용자 요소에 해당하며 모든 Account 는 반드시 하나의 Directory 에 포함되어야 한다.

출입자 제어 모델에 해당하는 USERS 는 Account 에 매칭되며 ROLES 은 Directory, PERMISSIONS 은 Group 각각에 매칭된다.

5. 캠퍼스 출입자 권한 제어 구현

RBAC 기반 캠퍼스 출입자 제어는 JSON 형태의 메시지를 사용하며 Stormpath 의 데이터 모델을 기반으로 이루어 졌다. RBAC 의 구조 설계를 통하여 사용자와 Stormpath 간의 메시지 전달이 이루어지고 사용자의 출입제한을 할 수 있게 된다.

그림 3 은 RBAC 기반의 출입자 권한 제어 시스템의 구조이다. Open API Interface 는 다양한 서비스를 사용자가 쉽게 접근하고 생성 할 수 있도록 해주는 인터페이스이다. 사용자에게 호출된 API 정보를 Message Router 에게 전달한다. Message Router 는 호출된 API 를 통해 전달된 메시지의 유효성을 검사하고 라우팅 엔진을 통하여 생성된 경로로 메시지를 전달한다[6]. 전달된 메시지는 RBAC Adapter 내의 Message Parsing Table 를 통하여 사용자의 정보들을 Stormpath 에게 전달하여 출입 통제를 하게 된다. RBAC 모델을 적용 사용함으로써 관리의 효율성을 높이고 사용자가 가지는 역할에 따라 캠퍼스의 출입자 권한 제어를 할 수 있다.



<그림 3. RBAC 기반의 출입자 권한 제어 시스템의 구조>

6. 결론 및 향후 연구

본 논문에서는 역할기반 접근제어를 기반으로 사용자 계정관리 서비스를 통하여 캠퍼스 출입자 권한 제어방법을 구현하였다. 앞서 제안한 RBAC 기법을 기반으로 구현된 캠퍼스 출입자 권한제어는 RBAC 모델을 적용 사용함으로써 관리의 효율성을 높였다.

향후 연구에서는 상속관계를 포함한 RBAC 모델을 추가하고 JSON 메시지를 통한 메시지 전달 방식을 활용하여 출입 제어관리를 다른 시스템과 매쉬업 하여 사용할 수 있도록 연구 할 것이다.

7. 참고 문헌

[1] D. F. Ferraiolo and D. Richard Kuhn, "Role-based access control," *in Proc. 15th NIST-NCSC National Computer Security Conference*, pp 554-563, Baltimore, MD, October 13-16, 1992.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control," *IEEE.... Computer*, pp. 38-47, Feb. 1996.

[3] 노승민, 이수철, 황인준, 박상진, 김현주, "의료정보 보호를 위한 역할기반 접근 제어 모델 설계", 한국정보기술학회, 제 31 권, 제 1 호, pp358-360, 2004 년 4 월.

[4] 김지헌, 박동규 "헬스 케어 시스템에 적합한 확장된 RBAC 의 구현", 한국정보기술학회, pp. 211-215, 2005 년 7 월.

[5] 김진식, 김민영, 이상원 "데이터베이스를 이용한 RBAC(역할기반 접근제어) 서버 API 구현", 한국정보과학회, 제 32 권 제 2 호, pp. 199-201, 2005 년 11 월.

[6] T. Kim, H. Kim, D. Kim, K. Ok, I. Im, E. Lee, H. Je, N. Rakhmatov, D. An, and H. Ju, "Design of an Effective Framework for Mash-up Service Development," *in Proc. International Conference on Information Networking (ICOIN)*, pp. 357-359, January. 2016.

8. Acknowledgement

이 논문은 2016 년도 정부의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

출입 관리 서비스를 위한
Enterprise Integration Patterns 기반 매쉬업 프레임워크 설계

김현우 ○, 권동우, 주홍택

계명대학교 컴퓨터공학과

{hwkim84, dwkwon, juht}@kmu.ac.kr

요 약

본 논문에서는 출입 관리 서비스를 위해 EIP (Enterprise Integration Patterns)에 따라 설계된 매쉬업 프레임워크를 제안한다. 제안된 프레임워크는 각각의 어댑터를 통해 사물인터넷 플랫폼과 클라우드, 빅데이터, 보안 플랫폼과 연동하며, 이를 기반으로 여러 플랫폼을 활용하는 출입 관리 서비스의 기능을 제공한다.

1. 서론

매쉬업은 두 개 이상의 기존 서비스를 혼합하여 새로운 서비스를 만드는 것을 말한다[1]. 매쉬업을 통해 기존의 서비스를 활용함으로써 개발자가 새로운 서비스를 구축하는데 소요되는 개발 기간과 비용 부담을 줄일 수 있는 장점이 있기 때문에 매쉬업이 효과적인 구현 방법으로 활용되고 있다[2]. 그러나 아직 몇몇 부분에서 고려해야할 문제점이 있다. 먼저, 대부분의 서비스 제공자들은 자신들의 서비스에 대한 Open API 를 제공하고 있지만, 매쉬업을 하기 위한 모든 과정을 개발자가 수행해야 하기 때문에 어렵고 많은 시간이 소비된다. 두 번째는 SaaS (Software as a Service) 와 같이 서비스 기반 매쉬업 외에 다양한 이기종 플랫폼 간에도 매쉬업 할 수 있는 기술에 대한 연구가 부족하다. 세 번째는 이러한 매쉬업 개발에 대한 중간 과정을 자동으로 생성해주는 기술이 요구되고 있다[3]. 따라서, 이와 같은 문제점을 보완하여 새로운 매쉬업 서비스를 효율적으로 개발할 수 있는 프레임워크가 필요하다.

우리는 이전 연구에서 새로운 매쉬업 서비스를 효율적으로 개발할 수 있는 프레임워크 구조를 제안하였다[1]. 본 논문에서는 기존에 제안된 매쉬업 프레임워크 구조를 활용하여 서비스 개발이 가능한지 실증하고자 출입 관리 서비스를 고려하였으며, 이 서비스를 제공하기 위한 매쉬업 프레임워크를 EIP (Enterprise Integration Patterns)에 따라 설계하였다.

2. 관련 연구

T. Kim 외 9 명은 복잡하고 다양한 서비스들의 요구사항을 수용하면서 사용자들이 매쉬업 서비스를 효율적으로 개발할 수 있도록 지원하는 매쉬업 프레임워크를 설계하였다[1].

D. Zhiquan 외 3 명은 여러 종류의 사물인터넷 데이터에 효율적으로 접속하고, 새로운 부가가치 서비스에 대해 의미 있는 데이터 생성이 가능한 데이터 매쉬업 프레임워크를 제안하였다[4].

S.K. Guirguis 외 2 명은 클라우드 서비스와 모바일 디바이스 간 전송되는 데이터의 양을 줄이고 프로세싱 오버헤드를 최소화하기 위한 프레임워크를 제안하였다[5].

기존 연구[4,5,6]들은 2 개의 특정 서비스 또는 플랫폼 간 매쉬업 하는 프레임워크를 제안한 것이며, 우리가 제안하는 프레임워크는 2 개 이상의 플랫폼과 연동하여 이를 기반으로 새로운 서비스를 제공하는 것이 핵심이다.

3. 출입 관리 서비스의 요구사항

출입 관리 서비스는 특정 공간으로 출입하는 사람을 각종 센서를 이용하여 인식하고 출입 권한 정책에 따라 출입을 통제하는 서비스를 말한다. 이 서비스의 목적은 허가되지 않은 외부인의 출입을 사전에 감시함으로써 보안성과 안정성을 강화하는 것이다. 이 서비스의 가장 기본적인 기능으로는 먼저 각종 센서를 통해 감지된 출입자 정보를 분석해야 하며, 출입자에 대한 출입 권한 여부에 따라 출입 허가를 결정해야 한다. 이 기능의 핵심은 출입 통제에 대한 100% 정확성을 제공하는 것이다.

4. EIP 기반 매쉬업 프레임워크 설계

EIP (Enterprise Integration Patterns)는 기업에서 분산된 애플리케이션들을 통합하기 위한 방법론으로서, 여러 시스템들의 통합에 활용될 수 있는 다양한 패턴들을 제공한다[7]. 그림 1 에서 우리는 외부 서비스의 프로토콜에 맞게 메시지를 생성하는 각 플랫폼 어댑터와 최종 사용자로부터 매쉬업 Open API

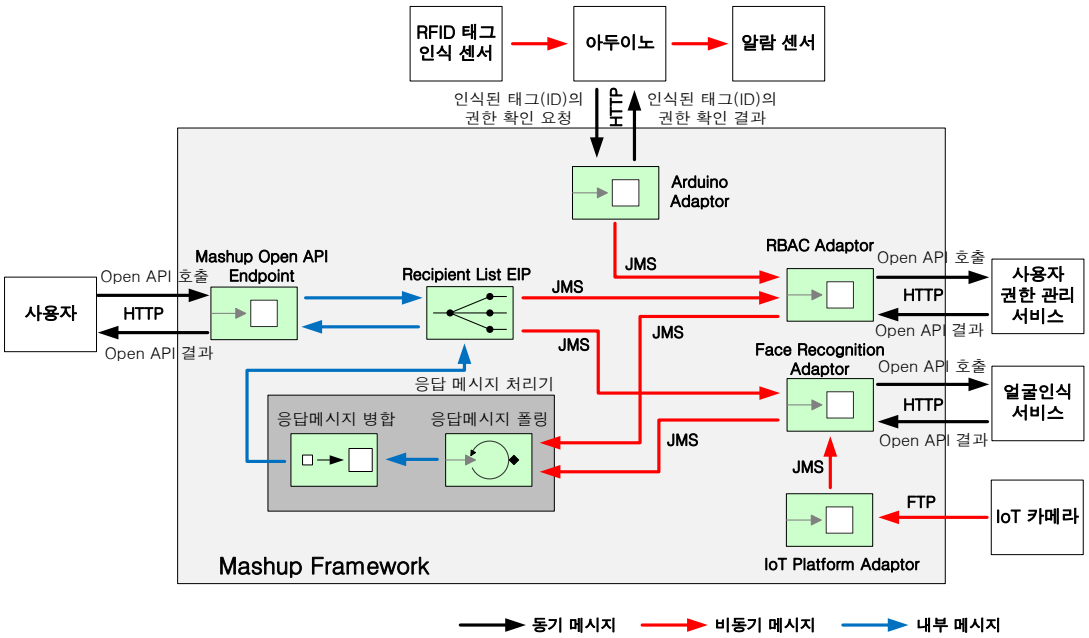


그림 1. EIP 기반 매쉬업 프레임워크 설계 및 메시지 통신 방식

가 호출 되는 인터페이스를 엔드포인트로 표현하였다. 각 엔드포인트로 전달되는 메시지는 비동기 방식으로 전송함으로써 분산 처리와 느슨한 결합(loose coupling)을 제공한다. 사용자로부터 호출되는 Open API 는 각각의 엔드포인트로 메시지를 전달해야 하는데, Recipient List EIP 패턴을 메시지 라우팅 기법으로 적용하였다. Recipient List EIP 패턴은 각각 메시지에 대해 보내고자 하는 엔드포인트를 동적으로 결정할 수 있고, 비동기 전송이 가능하기 때문에 적합하다고 판단하였다.

응답 메시지 처리기는 각 엔드포인트로부터 응답 메시지를 폴링(polling)하여 수신된 메시지를 수집한다. 그리고 응답 메시지의 헤더 정보를 확인하여 동일한 Open API 의 결과인 경우 해당하는 응답 메시지를 병합하여 Recipient List EIP 로 전달한다.

5. 결론 및 향후 연구

본 논문은 외부의 다양한 플랫폼을 활용하여 기본적인 출입 관리 서비스를 제공하기 위해 기존 연구에서 제안된 매쉬업 프레임워크를 EIP 기반으로 설계하였다.

우리는 출입 관리 서비스만을 고려한 매쉬업 프레임워크의 구조를 EIP 기반으로 설계하였으나, 향후에는 매쉬업 프레임워크가 다양한 플랫폼 또는 서비스와 연동 가능하도록 개선할 것이다. 또한 기존에 제안된 프레임워크 구조에서 트래픽 제어 및 워크플로우 관리 기능을 고려하여 다양한 측면에서 매쉬업 프레임워크의 성능 평가를 수행할 것이다.

6. 참고 문헌

[1] T. Kim, H. Kim, D. Kim, K. Ok, I. Im, E. Lee, H. Je, N. Rakhmatov, D. An, and H. Ju, “Design of an Effective Framework for Mash-up Service Development,” in *Proc. International Conference on Information Networking (ICOIN)*, pp. 357-359, January. 2016.

[2] 이용주, “RESTful 웹 서비스를 위한 리소스 매치메이킹”, *한국정보기술학회논문지*, 제 11 권, 제 8 호, pp. 135-143, 2013 년 8 월.

[3] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, “Mashup Advisor: A Recommendation Tool for Mashup Development,” in *Proc. IEEE International Conference on Web Services (ICWS)*, pp. 337-344, September, 2008.

[4] D. Zhiquan, Y. Nan, C. Bo, and C. Junliang, “Data Mashup in the Internet of Things,” in *Proc. International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 948-952, December, 2011.

[5] S.K. Guirguis, A.A. El-Zoghabi, and M.A. Hassan, “Lightweight Architecture for Mobile Web Content Access over Enterprise Cloud Mashup,” in *Proc. World Symposium on Computer Applications & Research (WSCAR)*, pp. 1-8, January, 2014.

[6] J. Im, S. Kim, and D. Kim, “IoT Mashup as a Service: Cloud-based Mashup Service for the Internet of Things,” in Proc. International Conference on Services Computing (SCC), pp. 462-469, June, 2013.

[7] Enterprise Integration Patterns, http://www.enterpriseintegrationpatterns.com/

7. Acknowledgement

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

무선 네트워크에서의 다양한 TCP 프로토콜의 성능 평가

정진화, 안동혁

계명대학교 컴퓨터공학전공

blackv33@naver.com, donghyeokan@kmu.ac.kr

요 약

인터넷에서 TCP 는 가장 많이 사용되고 있는 프로토콜이며, 네트워크 대역폭의 사용률에 중요한 역할을 한다. 본 논문은 사용자들이 실제로 사용하는 무선 네트워크에서 TCP 프로토콜의 성능을 평가하고자 한다.

1. 서론

인터넷(Internet)이 사용되기 시작하면서, 신뢰성 있는 전송 서비스의 필요성은 꾸준히 제기되어 왔고, TCP 프로토콜이 개발되어 사용되어 왔다. 이와 더불어 통신 기술이 발전함에 따라, 인터넷의 속도 향상을 위해 TCP 프로토콜은 네트워크 대역폭(bandwidth)를 충분히 활용하기 위해 다양한 TCP 프로토콜 개발 연구가 인터넷 초기 단계부터 최근까지 진행되고 있다.

TCP Reno 는 패킷 손실로 인한 급격한 성능 감소를 방지함으로써 TCP 성능을 개선하였다 [1]. 유선 네트워크를 기반으로 설계된 TCP 프로토콜을 수정 없이 무선 네트워크에서 사용이 가능하나, 무선 네트워크는 유선 네트워크에 비해서 비트 오류가 높아 패킷 손실이 더 빈번하게 발생해 TCP 성능이 예상과 다르게 저하된다. 이에 무선 네트워크 특성을 반영한 TCP 프로토콜도 제안되었다 [2]. 최근 들어, 유선 및 무선 네트워크의 대역폭이 인터넷 초기의 대역폭에 비해 매우 크게 증가하였고, 이에 대처하고자 TCP Cubic 프로토콜이 개발되었다 [3]. 3G 와 LTE 와 같은 셀룰러 네트워크 (cellular network)에서는 짧은 시간 동안 대역폭의 변동이 빠르게 발생하며, 이로 인한 성능 저하를 방지하기 위해서 위해서 TCP Verus 가 제안되었다 [4].

최근까지 네트워크 대역폭 성능 향상을 위해서 많은 TCP 프로토콜이 개발되었고, 여러 TCP 프로토콜들이 구현되어 사용되고 있다. 이에 통신 환경에 따라 TCP 프로토콜의 성능이 다양하게 나타나므로, 실제 무선 네트워크의 성능평가가 필요하다. [5]에서 여러 TCP 프로토콜 성능 평가가 이루어졌으나, 이는 LTE 네트워크에서만 성능 평가를 하였다. 이에 본 논문에서는 3G, LTE, WLAN 과 같이 사용자들이 실제로 사용한 무선 네트워크에서 TCP Reno 와 TCP Cubic 과 같은 대표적인 TCP 프로토콜 성능을 측정 및 분석하고자 한다.

2. 관련연구

본 논문에서는 무선 네트워크에서 TCP Reno 와 TCP Cubic 프로토콜을 기반으로 TCP 프로토콜의 성능을 측정한다. TCP Tahoe 는 패킷 손실을 감지한 후 Slow Start 부터 다시 시작함으로써 패킷 손실 시 전송률이 급격하게 저하되는 문제점이 발생하였다. 이를 극복하기 위해서 TCP Reno 에서는 Fast Recovery 단계를 추가하여 혼잡 윈도우가 패킷 손실 전의 절반부터 시작하도록 수정하였다. 이를 통해 TCP Reno 는 TCP Tahoe 에 비해서 증가된 throughput 을 보이고 있다.

유무선 통신 기술의 발전으로 인해 네트워크 대역폭은 매우 큰 폭으로 증가하였으나, 기존의 혼잡 회피 알고리즘은 네트워크 대역폭을 최대한으로 사용하지 못하였다. 이를 해결하기 위해서, TCP Cubic 은 높은 대역폭을 최대한 활용하기 위해서 혼잡 윈도우를 급격하게 증가시킨다. 패킷 손실이 발생하면, 혼잡 윈도우를 threshold 로 설정한 후 threshold 에 근접하였을 때 혼잡 윈도우 증가 속도를 늦추어 추가적인 패킷 손실을 방지한다. 혼잡 윈도우가 threshold 를 어느 정도 넘어서면, 다시 혼잡 윈도 증가 속도를 높여 새로운 네트워크 대역폭 threshold 를 탐색한다. 이를 통해 TCP Cubic 은 높은 네트워크 대역폭을 충분히 활용해 데이터를 전송한다.

3. 성능 평가

본 논문에서는 무선 네트워크에서 실제로 사용하는 스마트 단말에서의 TCP Reno 와 TCP Cubic 프로토콜의 성능을 측정하였다. 성능 측정에 사용한 스마트 단말은 Android 6.0.1 OS 를 사용하는 Nexus 5X 이다. 해당 기기의 대략적인 사양은 1.8GHz 헥사코어, 2G RAM, GSM, LTE, 802.11 a/b/g/n/ac 이다. 3G 와 LTE 의 성능 평가를 위해서 본 논문에서는 KT 의 이동 통신망을 사용하였다. 무선랜은 802.11n 까지 지원이 가능한 일반적인 무선 AP 를 사용하였으며, 무선 AP 는 계명대학교 공학관에 설치하였다.

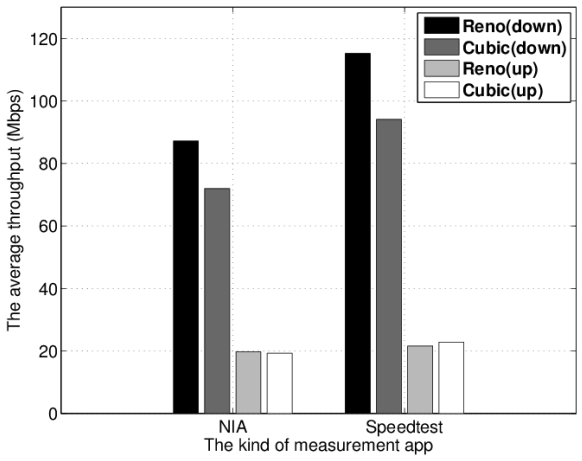


그림 1. LTE 망에서의 성능 측정

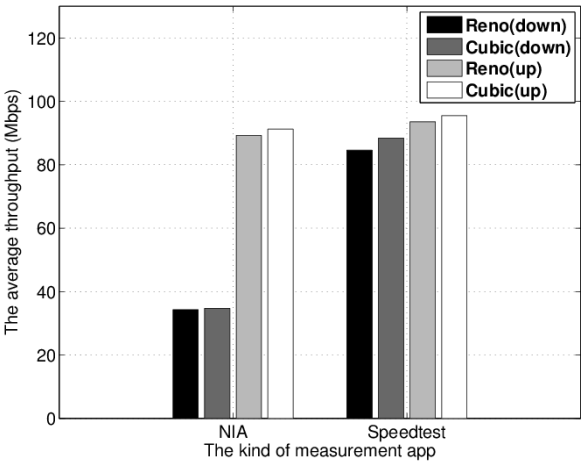


그림 2. WiFi 망에서의 성능 측정

본 성능 평가는 LTE, 무선랜, 3G 환경에서 각각 10 번씩 측정하여 평균값을 계산하였다. 순간적인 네트워크 상태 변화로 인한 오류를 줄이기 위해서 성능 평가를 2 분 간격으로 5 번 측정 후 TCP 프로토콜을 변경하여 측정을 진행하였다. 성능 측정은 구현 방법에 따라 성능 차이가 발생할 수 있으므로, 두 가지 어플을 활용해 성능 평가를 진행하였다. 하나는 한국정보화진흥원에서 제작한 "무선인터넷 속도측정" 어플과 "Speedtest.net" 어플이다.

그림 1 은 LTE 망에서의 성능 측정 결과를 나타낸 것이다. NIA 는 "무선인터넷 속도측정" 어플을 의미하며, LTE 망에서는 다운로드가 업로드보다 높은 성능을 보이고 있다. 결과에서 TCP Reno 가 TCP Cubic 보다 다운로드 성능이 좋다는 것이 주목할만한 결과이다.

무선랜과 3G 망에서의 TCP 프로토콜 성능 측정 결과는 그림 2 와 그림 3 에서 보여주고 있다. 두 결과에서 각 어플마다 Reno 와 Cubic 의 성능 측정 결과가 비슷하게 나오는 것을 확인할 수 있다. 무선랜의 측정 결과에서 다운로드가 업로드보다 낮게 나타나는데 이는 다운로드의 수요가 높아 경쟁으로 인해 성능이 업로드보다 낮게 측정된 것이다.

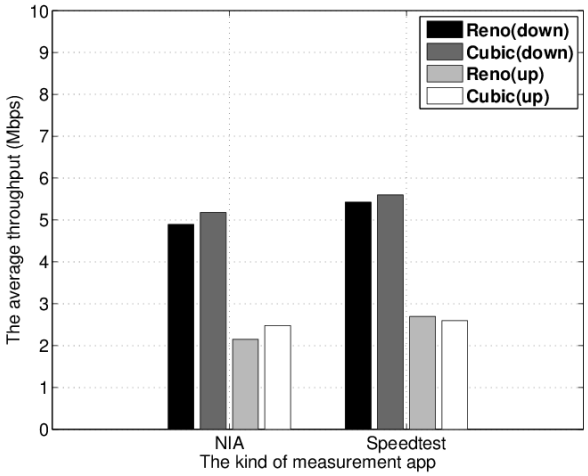


그림 3. 3G 망에서의 성능 측정

4. 결론

본 논문에서는 LTE, 3G, 무선랜과 같이 실제 사용되고 있는 무선 네트워크에서 TCP Reno 와 Cubic 의 성능 측정을 수행하였다. LTE 망에서 두 프로토콜의 성능이 다름을 확인하였다. 추후 두 프로토콜의 성능 차이가 발생하는 원인을 분석하고, throughput 외에도 전송지연시간(transmission dely) 등 추가적으로 성능 측정을 수행할 계획이다. 추가적으로 정적인 환경 외에도 이동성이 있는 환경 및 통신 환경이 좋지 못할 때의 성능 측정도 진행할 계획이다.

5. ACKNOWLEDGMENT

이 성과는 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2015R1C1A1A01052627).

6. 참고 문헌

[1] M. Allman, V. Paxson, and W. Richard Stevens, *TCP Congestion Control*,RFC 2581, 1999.

[2] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang, "TCP Westwood: End-to-End Congestion Control for Wired/Wiress Networks," *Wireless Networks*, vol. 8, no. 5, pp. 467-479, Sep. 2002.

[3] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64-74, July 2008.

[4] Y. Zaki, T. Potsch, J. Chen, L. Subramanian, and C. Gorg, "Adaptive Congestion Control for Unpredictable Cellular Networks," *Proc. ACM SIGCOMM 15*, pp. 509-522, Aug. 2015.

[5] 염홍주, 김인태, "LTE Network 에서의 여러 TCP 의 혼잡제어 방식의 성능평가에 관한 연구", 한국통신학회 2015 년 하계종합학술발표회, 2015 년 6 월.

얼굴인식 Open API 게이트웨이 설계

옥기수 ○, 권동우, 주홍택

계명대학교 컴퓨터공학과

{ok7323, dwkwon, juht}@kmu.ac.kr,

요 약

본 논문에서는 다양한 얼굴인식 Open API 서비스를 통합하는 얼굴인식 Open API 게이트웨이를 제안한다. 현재 제공되고 있는 얼굴인식 Open API 서비스들은 각각 다른 특징을 가지고 있기 때문에 다수의 얼굴인식 Open API 서비스를 함께 조합하여 사용하면 새로운 서비스를 개발하는데 도움이 된다. 하지만 개발자가 다양한 얼굴인식 Open API 조합하여 서비스를 개발하기에는 많은 시간이 소요가 된다. 본 논문에서는 각 Open API 의 특징, 인식요소, 메시지의 흐름을 파악하고 통합하여 다양한 얼굴인식 Open API 를 사용 가능한 게이트웨이를 설계한다. 얼굴인식 Open API 게이트웨이는 새로운 얼굴인식 서비스를 효율적으로 개발할 수 있도록 지원한다.

1. 서론

최근 얼굴인식 서비스는 원거리로부터 얼굴정보를 취득할 수 있는 편의성과 다양한 응용서비스들의 요구로 인해 얼굴인식의 중요성과 수요가 확대되고 있다[1]. 페이스북은 온라인 서비스에 얼굴인식 기능을 추가하는 등 웹 기반의 얼굴인식 서비스 제공하고 있다[2]. 얼굴인식을 응용한 새로운 서비스를 개발하고 얼굴인식 품질을 향상시키기 위해서는 다양한 얼굴인식 Open API 의 합성 및 앙상블이 요구된다.

하지만 얼굴인식 서비스들을 연동하여 사용하기에는 몇 가지 문제점이 있다. 첫 번째는 개발자가 다수의 얼굴인식 서비스의 인식 요소들을 활용하여 새로운 서비스 개발에 많은 시간이 소비된다. 두 번째는 연동되는 얼굴인식 서비스들이 서로 다른 프로토콜을 및 다른 응답형식을 사용하여 하나의 서비스처럼 연동하여 사용하기 힘들다.

본 논문에서는 다양한 얼굴인식 Open API 를 합성하여 개발 편의성을 제공하고, 얼굴인식 Open API 의 특징을 앙상블 하여 서비스 개발의 품질을 향상시키는 얼굴인식 Open API 게이트웨이를 제안한다. 얼굴인식 Open API 게이트웨이는 개발자와 얼굴인식 Open API 사이에서 데이터를 중계, 변환, 통합 등의 역할을 한다. 얼굴인식 Open API 게이트웨이는 단순히 데이터를 중계하는 것뿐만 아니라 개발자가 서비스를 개발하기에 효율적인 환경을 제공한다.

본 논문의 구성은 다음과 같다. 2 장에서는 얼굴인식 시스템과 Open API 게이트웨이에 관련된 연구들을 소개한다. 3 장에서는 얼굴인식 서비스 합성 방법을 제시한다. 4 장에서는 얼굴인식 Open API 게이트웨이와 내부 모듈들에 대해서 설명한다. 마지막으로 5 장에서는 결론 및 향후 연구에서 대해서 논의한다.

2. 관련 연구

김상일과 김화성은 REST (Representational State Transfer) 프로토콜 기반의 Open API 를 선별하고 자동으로 합성하는 기법에 대해서 제안하였다[3]. 그러나 Open API 합성방법에 필요한 기준에 대한 설명이 부족하다.

이용주는 시맨틱 기반 기술을 이용하여 다양한 Open API 들을 매쉬업 할 수 있는 도구를 제안하였다. Open API 의 프로토콜인 REST, SOAP (Simple Object Access Protocol), JavaScript, XML-RPC 등에 의해 제공되는 서비스들을 시맨틱 기법을 이용하여 자동으로 온톨로지를 구축하였다[4].

정진욱 외 4 명은 매쉬업 서비스를 위한 링크드 데이터 및 Open API 통합 연계 시스템을 설계하고 구현하였다. 통합 연계시스템은 개발자와 Open API 사이에서 데이터 전달 역할을 하고, 개발자는 Open API 에 대한 사전 지식 없이 서비스를 이용가능 하다[5]. 이 연구는 단일 주제의 Open API 만 사용하여 서비스의 활용도가 부족하다.

3. 얼굴인식 Open API 분석 및 합성

얼굴인식을 위해서는 두 가지의 기본 흐름이 있다. 첫 번째는 얼굴인식 서비스에 업로드 한 사진에서 얼굴 추출하여 개인 저장소에 저장하는 흐름이다. 두 번째는 얼굴인식 서비스는 업로드 된 사진에서 얼굴을 탐지하고, 개인저장소에 저장된 얼굴들과 비교하여 가장 유사한 얼굴을 찾는 흐름이다. 이 두 가지의 기본 흐름을 포함하는 얼굴인식 Open API 서비스에는 BetaFace, Lambda, KAIROS 등이 있다.

Lambda Open API 는 구글 글라스에 적용되었던 얼굴인식 서비스다. Lambda 는 8 개의 기준점을 이용하여 얼굴 탐색을 실시하고, 다른 얼굴인식 서비스

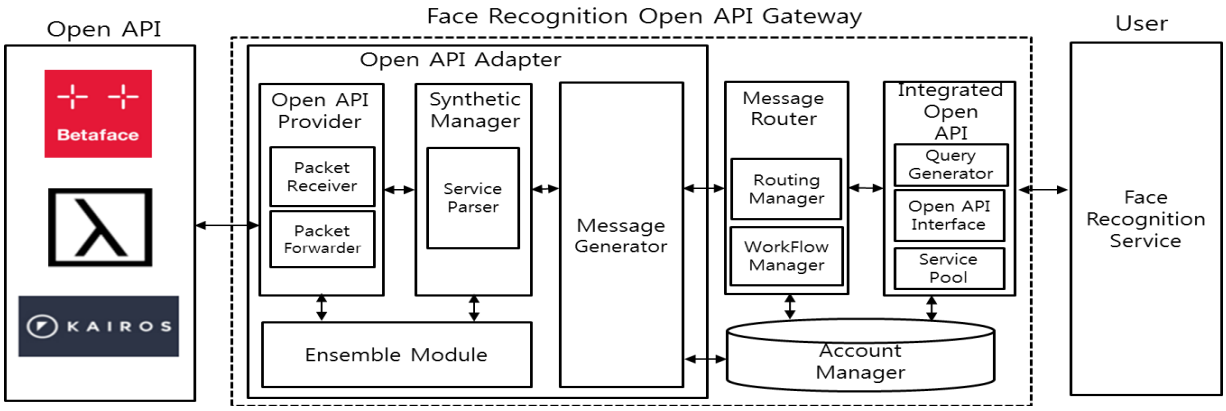


그림 1. 얼굴인식 Open API 게이트웨이 구성도

보다 빠른 속도로 인식결과를 제공한다. Betaface Open API 는 다른 얼굴인식 서비스와는 다르게 폴링 방식을 사용하여 얼굴인식 서비스를 제공한다. Betaface 에서는 propoints, classifiers, extended 와 같이 얼굴인식에 성능에 대한 옵션을 제공한다. KAIROS Open API 는 JSON 타입의 페이로드 데이터를 설정해야 얼굴인식 서비스 요청이 가능하다. KAIROS 는 얼굴인식 범위와 결과 반환에 대한 조건설정 및 개인저장소 다양하게 활용이 가능하다.

Open API 얼굴인식 서비스의 특징은 인식 기준점, 인식성능, 인식결과 조정, 개인저장소 활용, 얼굴인식 결과 표기, 서비스 방식 등이 있다. 본 논문에서 제안하는 얼굴인식 Open API 게이트웨이는 이러한 서비스의 특징을 앙상블 하는 기능을 제공한다.

4. 얼굴인식 Open API 게이트웨이 설계

그림 1 은 다양한 얼굴인식 Open API 를 통합하여 사용할 수 있게 해주는 게이트웨이이다. 게이트웨이는 개발자의 얼굴인식 요청을 받으면 얼굴인식 Open API 로 요청을 전달하고, 얼굴인식 Open API 로부터 얻은 결과 값을 개발자에게 전달해준다. 각 모듈에 대한 설명은 다음과 같다.

Integrated Open API 는 개발자 사용하려는 응용서비스에 적합한 얼굴인식 Open API 를 선택한 후 서비스를 요청 할 수 있다. 메시지는 Query Generator 에 의해 생성 후 Message Router 로 전달된다. Query Generator 에서 생성된 메시지 혹은 Open API Adapter 에서 전달받은 메시지의 경로 설정 기능을 한다. Account Manager 는 얼굴인식 Open API 게이트웨이의 개발자 계정, Open API ID, 키 등 인증에 대한 정보를 관리한다. Message Splitter 는 얼굴인식을 요청하는 메시지를 각각 서비스에 대한 쿼리로 분리시키는 역할을 한다. Synthetic Manager 는 연결된 Open API 정보를 취합하여 하나의 서비스로 합성한다. Open API Provider 는 Open API Adapter 와 Open API 서비스 사이에서 통신과 인증을 담당한다. Ensemble Moule 은 얼굴인식 Open API 에 앙상블 알고리즘을 적용하여 서비스 품질을 향상시키는 가능을 한다.

5. 결론 및 향후 연구

본 논문에서는 다수의 얼굴인식 Open API 서비스를 통합하여 사용 가능한 얼굴인식 Open API 게이트웨이를 설계하였다. 제안된 게이트웨이는 통합되는 얼굴인식 Open API 서비스의 개념, 인식요소, 특징들을 파악하여 얼굴인식 서비스를 통합하였다.

향후 연구에서는 얼굴인식 Open API 의 인식결과를 이용하여 얼굴인식의 성능을 개선할 수 있는 방법을 연구할 것이다.

6. 참고 문헌

[1] R. Jafri and H. R. Arabnia, "A Survey of Face Recognition Techniques," *Journal of Information Processing Systems*, Vol. 5, No. 2, pp. 41-68, Jun. 2009.

[2] B. C. Becker and E. G. Ortiz, "Evaluation of face recognition techniques for application to facebook," *in Proc. IEEE Automatic Face & Gesture Recognition (FG)*, pp. 1-6, Sep. 2008.

[3] 김상일, 김화성, "REST 프로토콜 기반의 API 선별 기법 및 Open API 자동 합성 방안", *한국통신학회논문지*, 제 38 권 제 7 호, pp. 587-594, 2013 년 7 월.

[4] 이용주, "다양한 Open API 타입들을 지원하는 시맨틱 기반 매쉬업 개발 툴", *인터넷정보학회논문지*, 제 13 권 제 7 호, pp. 115-126, 2012 년 6 월.

[5] 정진욱, 임동혁, 이경민, 종난수, 김홍기, "시맨틱 웹 매쉬업 서비스를 위한 링크드 데이터 및 Open API 통합 연계 시스템의 설계 및 구현", *한국정보과학회 학술발표논문집*, 제 39 권 제 1A 호, pp. 71-73, 2012 년 6 월.

7. 감사의 글

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1009, ICBMS 플랫폼 간 정보모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

SDN 기반 드론 무충돌 경로 제어

백성복, 이영우

KT 인프라연구소

{sbbaik, young-woo.lee@kt.com}@kt.com

요 약

본 논문은 SDN 기술의 적용 분야를 네트워킹 분야 외의 다른 분야로 확장할 수 있을지, 또한 그것이 네트워킹 분야에서 보여준 장점을 동일하게 보일 수 있을 지를 알아보기 위한 초기 수준의 시도를 기술한 것이다. 최근 괄목할만한 발전을 보이고 있는 드론 분야로의 적용을 시도했으며, 특히 도심 지역과 같이 다수의 드론이 혼잡 비행할 것으로 예상되는 환경에서 SDN 기술을 이용한 무충돌 경로 산출 방안을 제시한다.

제안된 방식은 SDN 컨트롤러를 중앙의 비행 경로 관리 서버로 사용하여 비행 구역 내 모든 드론의 비행 경로를 설정하도록 구현함으로써 드론 자체에 장착된 기능에만 의존하여 충돌을 회피하는 전통적인 방식에 비해 전체 비행 시스템의 구조를 단순화 및 효율화 시킬 수 있을 것이라는 가능성을 보인다.

1. 서론

2008 년 OpenFlow 프로토콜의 발표를 기점으로 지속적인 발전을 보여온 SDN (Software Defined Networking) 기술은 현재까지 Data Center 를 위한 네트워킹 분야를 중심으로 기술적인 성숙과 시장 가시화를 이루어낸 것으로 평가 받고 있다[1].

SDN 기술은 제어평면(Control Plane)과 데이터 전달 평면(Data Forwarding Plane)의 분리를 통해 양쪽 평면의 자유도를 높이고, 전체 아키텍처의 복잡성을 낮출 뿐 아니라, 관리의 편의성과 직관성을 높이는 등 여러 가지 혁신적인 장점을 가지고 있다.

이러한 SDN 기술의 장점은 SDN 기술을 네트워킹 분야 이외의 다른 분야에도 적용함으로써 여러 가지 이점을 얻을 수 있을 것이라는 가정을 해 볼 수 있게 한다.

적용 가능한 다양한 분야가 존재 하겠지만 경로를 미리 설정해 주고 그 경로에 따라 물리적인 사물을 이동시키는 일반적인 메커니즘을 갖는 분야로의 적용이 SDN 기술 자체의 변형 없이 그대로 적용할 수 있는 최적의 분야일 것으로 판단된다.

본 논문에서는 최근 새롭게 떠오르는 드론의 비행 경로 설정 분야에 SDN 기술을 적용하는 방법을 제시하고자 한다.

2. 배경

2013 년 미국의 온라인 쇼핑몰 회사인 아마존에서 드론을 이용한 택배 서비스의 개발을 공표한 이래 3 년이 지나고 있지만 상용화 사례에 대해서는 아직 보고된 바가 없다. 2015 년에는 중국의 쇼핑몰 회사인 타오바오도 중국 주요 도시 고객들을 대상으로 가벼운 물품의 시험 배달을 성공적으로 수행한 바 있으나 시험 서비스 이 후의 진척 상황이 보고되지 않고 있어서 이 분야의 상용화는 생각보다는 느리게 진행되고 있는 것으로 관측된다.

드론의 상용화, 특히 물류 분야에서의 성공적 적용을 가로 막는 기술적인 요인은 충돌 회피 기술의 부족이다[2]. 드론이 상용화 되면 필연적으로 도심 상공과 같이 드론들이 혼잡 비행을 해야 하는 상황을 만나게 될 것이며 이때 드론 간의 충돌의 문제가 심각하게 대두될 것이다.

최근까지 드론의 충돌 회피 방법에 관한 연구는 주로 드론 자체에 충돌 회피 장치를 구현하여 장착하는 방식을 채택하고 있으며, 이미지 프로세싱이나 각종 센서 기술을 통해 비행 경로상의 장애물을 감지하고 회피하는 연구가 주류를 이룬다. 비행체의 각속도와 선속도를 계산 및 수정하여 예정된 경로상에 존재하는 장애물을 회피하는 방법[3]도 이 부류에 속한다고 할 수 있다.

한편 드론의 사용 목적 및 운용방식 상의 특징과 드론 자체의 기동 방식 상의 차이점을 이용한 충돌 회피 방법도 연구되고 있는데, [4]에서는 드론 간 통신을 이용한 충돌 회피 방법을 제시하면서 충돌 회피를 위한 드론 택배 시스템의 통신망 토폴로지 및 성능평가에 대해 연구한 바 있다.

도심지의 물류, 감시, 측량 등 다양한 용도로 드론이 활용될 것으로 예상하고, 도심에서 드론을 운용할 경우 원거리의 안전한 통신선 확보와 비행체의 정확한 출처 인증이 중요한 문제임을 제기하여 이에 대한 해결책으로 3G/LTE 등 기간통신사업자의 망에 접속하도록 하는 방안을 제시한 연구 사례도 있다[5].

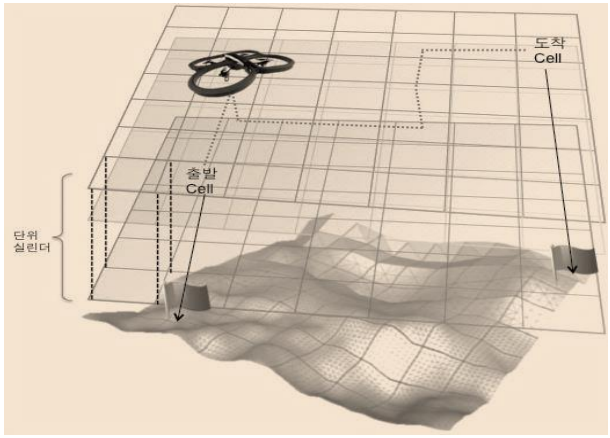
위의 여러 가지 연구사례와 같이 드론 자체에 충돌 회피 장치를 장착하고 고도화/정밀화 하는 방식은 드론을 소프트웨어적/하드웨어적으로 무겁게 만드는 요인이 되며, 이미지나 센서 정보 처리기의 처리속도에 의해 비행체의 운행 속도가 제한 받을

수 있다는 문제점을 안고 있다.

3. SDN 기반 경로 제어

본 논문은 SDN 컨트롤러의 형식으로 중앙에 구현된 제어기가 각 드론의 비행 경로를 미리 설정해 줌으로써 드론간 충돌의 가능성이 원천 차단된 상태에서 여러 개의 드론들이 고속으로 목표 지점까지 안전하게 이동할 수 있게 해주는 방안을 제시한다.

그림 1 에서와 같이 동시에 많은 수의 드론이 비행할 것으로 예상되는 특정 지역 상공의 특정 고도 구간에 대해 비행 구역을 확보하고, 이 비행 구역 전체를 작은 실린더들로 구분하고, 실린더를 다시 여러 계층으로 나누어 작은 셀로 구성한 다음, 인접한 셀들을 연결하여 비행 경로를 설정 및 제공함으로써 드론들이 설정된 경로를 따라 안전하게 이동하도록 유도하는 것이 본 연구의 핵심이다.

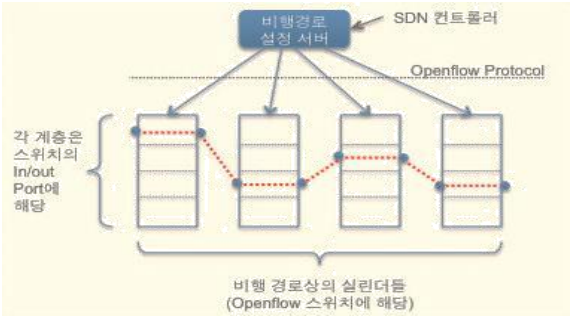


<그림 1. 작은 셀로 분할된 비행 구역 내에서 설정된 드론 경로>

그림 1 에 나타난 바와 같이 세로로 쪼개진 실린더는 여러 층의 셀로 나뉘어져 있기 때문에 이론상 층 수만큼의 드론이 동시에 같은 실린더를 지날 수 있도록 운영함으로써 비행 공간을 효율적으로 사용할 수 있도록 구성되어 있다.

셀의 점유상태에 대한 모니터링과 경로 계산 및 할당 작업은 중앙 관제소에 위치한 비행 경로 관리 서버에서 수행된다. 비행 경로 관리 서버는 여러 개의 무충돌 경로를 미리 계획해 줌으로써 각 경로를 따라 이동하는 비행체들의 무충돌을 보장할 수 있다. 이때 전제 원칙은 특정 시간에 특정 셀 내를 비행하는 비행체는 단 1 개만 존재 하도록 구성한다는 것이다.

여기서 각각의 실린더가 수행하는 역할을 관찰해 보면 네트워크 상에서 스위치가 하는 일과 매우 유사하다는 것을 알 수 있다. 즉, 각 실린더는 특정 층의 셀로 유입되는 드론을 받아서 비행 경로상에 존재하는 다음 실린더의 빈 셀로 내 보내는 역할을 하게 되는데, 이때 각 층은 스위치의 포트 번호에 해당하며 인접한 두 실린더는 네트워크 상의 이웃하는 노드에 해당한다고 볼 수 있는 것이다.



<그림 2. SDN 개념을 적용한 비행경로 설정 구조>

그림 2 에서와 같이 각 실린더를 OpenFlow 스위치로 대치하고, 중앙의 비행 경로 관리 서버를 OpenFlow 컨트롤러로 놓는다면 현재의 OpenFlow 프로토콜과 SDN 아키텍처를 그대로 적용하여 앞서 설정한 비행 공간 전체를 완벽하게 제어할 수 있고 이 공간을 비행하는 드론들에 대해 무충돌 경로를 제공할 수 있게 된다.

4. 결론

본 논문은 SDN 기술을 네트워킹 분야에 국한하지 않고 다른 분야로 적용할 수 있다는 가능성을 제시했다는 점에서 의의를 갖는다. 새롭게 부각되고 있는 드론 분야에 SDN 기술을 접목함으로써 드론 분야의 전통 방식으로는 해결이 어려웠던 문제점들을 간단히 해결할 수 있음을 보였다.

이 방법은 비행 금지 또는 제한 지역에서 무조건 해당 구역을 관할하는 중앙 서버(SDN 컨트롤러)의 제어를 받도록 강제하는 시스템에도 사용될 수도 있을 것이다.

추가 연구가 필요한 분야로는 실린더와 셀의 크기 최적화의 문제, 에너지 효율적인 경로 구성 방법, 셀간 이동 시 속도 저하 없이 이동이 가능한 이동거리의 계산 방법, 등을 들 수 있다.

5. 참고 문헌

[1] J. Chen, X. Zheng, and C. Rong, “Survey on Software-Defined Networking,” in *link.springer.com*, vol. 9106, no. 9, Cham: Springer International Publishing, 2016, pp. 115–124.

[2] A. Lotz, “Drones in Logistics: A Feasible Future or a waste of effort.,” *Honors Project at Bowling Green State University*, pp. 1–41, Nov. 2015.

[3] 윤영훈, 박진배, 최윤호, “충돌 회피와 실시간으로 변하는 경로를 추종하는 무인비행기 제어 알고리즘,” presented at the 대한전기학회 학술대회 논문집 SN - VL - IS -, 2010, pp. 1772–1773.

[4] J.-M. Jo, “Communication Network Topology and Performance Evaluation of the Drone Delivery System for Collision Avoidance,” 한국전자통신학회 논문지, vol. 10, no. 8, pp. 915–920, 2015.

[5] 김은, 김종민, 김성윤, 이윤석, “USIM 을 탑재한 드론 시스템에 관한 연구,” 한국통신학회 학술대회논문집, 2015.

Evolution of Identity Management in OpenStack based Clouds

Ramneek[1, 2], Wonjun Choi[1, 2], Woojin Seok[1, 2], Yoonjoo Kwon[2]

Korea University of Science and Technology (UST), Daejeon, Korea[1]

Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea[2]

ramneek@kisti.re.kr

Abstract

Cloud Computing enables on-demand access to a number of computing and storage resources, provisioned as a service to the end users. Cloud eco-system has become a popular choice for scientific research collaborations, where cloud services can be deployed by different service providers, at distributed locations, using different middleware software stacks, thereby creating a heterogeneous eco-system. Apart from the highly powerful computational resources and efficient network infrastructure needed to support these data intensive application, there is a need for a robust federated identity management system for seamless communication and resource sharing between the heterogeneous cloud systems. In the present work, we aim at exploring the features and challenges associated with the identity federation support in the OpenStack software platform. We will discuss in brief about the evolution of keystone, the identity management module of OpenStack, from a purely backend based service to a more flexible version, capable of supporting external authentication. In addition, we will discuss some of the shortcomings in the existing module and discuss some possible solutions.

1. Introduction

Cloud computing offers an on-demand ubiquitous access to a set of hardware and software resources, including servers, storage, networks, applications, services, etc., that can be accessed remotely with minimum management overhead. Through the use of virtualization and resource sharing, it allows a large number of users to be served using a small set of physical resources, located remotely at centralized or distributed locations [1]. It is being widely used in academia as well as industry.

Cloud eco-system has become a popular choice for scientific research collaborations, where cloud services can be deployed by different service providers, at distributed locations, using different middleware software stacks, thereby creating a heterogeneous eco-system. Scientific research collaboration is very important for large scale research applications such as astronomy, medical science, bioinformatics, nuclear physics, computational science, and so on. These applications are highly resource intensive and need high computation power as well as high performance networks as infrastructure. Processing large volumes of scientific data might require high performance supercomputers, which have a huge hardware and maintenance cost associated with them. Also, there is a need of suitably skilled man-power to operate these resources. In addition to the computational hardware, there is need of high speed reliable network for sharing data among the peer groups for the purpose of active collaboration. As the researchers from different parts of the world might generally be located at different geographical location, this can lead to additional complexities. Apart from the highly powerful computational resources and efficient network infrastructure needed to support these data intensive application, there is a need for a robust Federated Identity Management (FIM) system for seamless communication and resource sharing between the heterogeneous cloud systems.

In the existing studies related to the optimization of cloud infrastructure for supporting scientific collaboration, people have mostly focused on the performance related issues. However, in a distributed environment where different resource providers might be using different hardware and software stacks, identity management is a challenging issue. Federated Identity Management is related to the authentication and authorization of resources and defining access policies. Identity federation plays a key role in a scientific collaborative environment as all the other services including resource provisioning, accounting and brokering etc will be dependent on it.

In the current work, we will focus on the OpenStack cloud software and discuss in brief about its identity module- Keystone. We will provide a brief overview of the currently supported services and the existing challenges that need to be addressed for seamless federation between heterogeneous cloud systems. The rest of paper is organized as follows: Section 2 provides a brief overview of the keystone architecture; Section 3 will describe the keystone external authentication and the related work; Section 4 includes the overview of the existing issues in Openstack identity federation and a brief overview of the proposed solutions based on the concept of Virtual Organizations (VO).

2. Openstack Identity Service – Keystone

Openstack is an open source cloud software platform that focuses on providing Infrastructure as a Service (IaaS) through a set of services: Horizon (the dashboard), Nova (compute service), Neutron (networking module), Swift

본 연구는 KISTI 연구과제(첨단연구망 기반 협업플랫폼 서비스 및 글로벌연동: K-16-L01-C02-S03) 지원으로 수행되었습니다.
본 연구는 미래부 ICT R&D 과제(서비스/단말/네트워크 다양성 지원을 위한 미래형 다차원 네트워크 시스템 아키텍처 연구: R-20160229-002917) 지원으로 수행되었습니다

(object storage), Cinder (block storage), Keystone (identity service), Glance (image service), Ceilometer (telemetry), Heat (orchestration) and Trove (database service), etc [2].

Keystone (Openstack Identity Service)

Keystone is the identity service of OpenStack that provides identity management and serves as an entry point for all the other services. It is organized as a group of internal services (as shown in Figure 1) exposed on one or more endpoints. The identity service performs the credential validation and provides metadata about the users, groups, etc. The token service is responsible for validation and management of tokens for authentication requests after the credentials have been verified by the identity service. The Catalog service provides endpoint registry for discovery of other OpenStack services. The Policy service provides an engine for rule-based authorization and the associated interface for rule management [3].

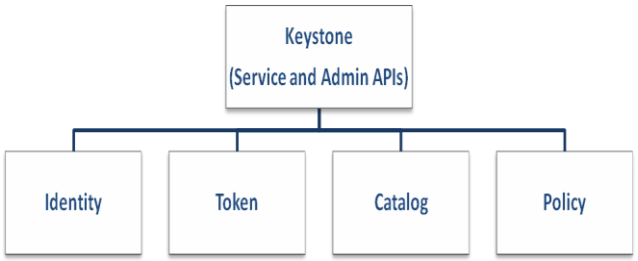


Figure1: Keystone Internal Services

When a user wants to access any Openstack service, he will provide the username and password to the Keystone, along with the request for the resources he wants to access. If the credentials are valid, then a scoped token is generated based on the project that the users wants to access. If the user does not specify the project he wants to access, then an un-scoped token will be generated, along with the catalog of services to choose from. This unscoped token can then be exchanged for a scoped token when the user specifies a particular project. After a scoped token is generated, the user can choose the service endpoint and forward the request along with the scoped token. Then the token is validated based on the trust relationship between the keystone and other Openstack services and if valid, the user request will be granted (as shown in Figure 2).

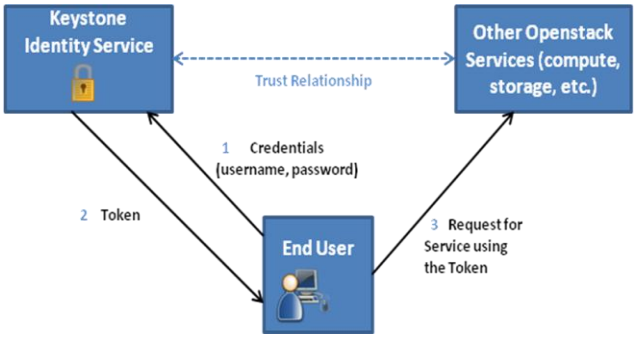


Figure 2: Keystone Authentication and Authorization

3. External Authentication Using Keystone

The traditional keystone implementation is centralized and has a number of limitations. Every time a new user enters the system, he has to be manually enrolled into the database by the Openstack admin, using the Horizon or the command line interface. This task can be tedious in case of a large number of users. Hence the traditional identity service is inefficient. In addition, unlike the traditional cloud systems, a federated cloud environment involves inter-cloud interactions, and hence there is a need of Federated Identity Management to enable authentication and authorization across the whole federated environment using a single set of credentials.

In order to address the above issue, a number of methods were proposed [4][5] etc. In case of [4], the authors proposed a framework for adding protocol independent identity management to Openstack so that existing Federated Identity Management protocols such as SAML could be integrated with the Keystone. One drawback of this solution was that relied on the SAML libraries to implement their own SAML functionalities and did not re-use the existing middleware to handle Shibboleth requests and assertions. Another solution, proposed by the researchers from CERN and IBM was based on WebSSO [5], a protocol independent federation module that works well with SAML, OpenID and other identity federation protocols. The external authentication support was standardized and available in OpenStack from the Juno release onwards [6]. It basically consisted of three entities: The Service Provider (SP), the Identity Provider (IdP) and the assertions, containing information about the users, as provided by the IdPs, as shown in Figure 3.

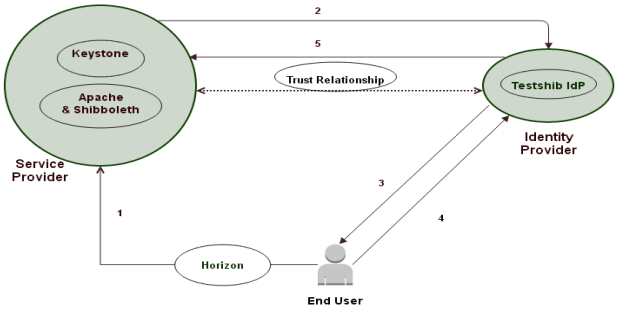


Figure 3: External Authentication in Keystone

The user request to login (1) is redirected to external IdP (2). The user is prompted to enter the credentials (3) and after the successful login (4), the user is redirected back to the SP (5), along with the attributes that are asserted by the IdP. These attributes are then mapped to the keystone internal users and groups to provide access to the Openstack services. In this case, the keystone is run under Apache HTTPD and external protocols implementations such as Shibboleth, Mellon, OpenID connect, etc can be installed and configured to be used for external authentication. Later on, the Keystone-to-keystone federation was also included, where one of the keystones was configured as the identity provider and the other as the service provider.

4. Identity Federation using the concept of VOMS

In the case of external authentication support in Openstack, there are a number of limitations that need to be addressed in the future. Firstly, in the current implementation, the user access rights are decided based on the group that they are mapped to, depending on the attributes asserted by the external IdPs. There is a single mapping defined per protocol for each IdP. However, not all the users from a single IdP may need same access right and different users from different IdPs may need same access rights. The standard authentication protocols such as SAML provide a set of attributes based on a single organization or institute, and do not provide the group information for assigning same access rights to users from different organizations, but belonging to a single collaboration. In addition, in the case of heterogeneous cloud environment, where different resource providers use different middleware software stack such as OpenStack, Cloudstack, OpenNebula, etc., there is a need for a robust federation management that can provide authentication and authorize access to services across different cloud systems in collaboration.

To address the above issues, a number of solutions have been proposed based on the concept of Virtual Organizations (VOs) and Virtual Organization Management Systems (VOMS) [7][8][9]. The concept of VOMS has been adopted from the grid environment, where the concept of VO was used in the multi-organization scientific collaborations to decide the user membership information based on an abstract group and not bounded by any particular organization [10]. In [7], Chadwick used the concept of federation attribute mapping to form VO groups and roles. Since the VO admin may not know what unique attributes will be asserted by the IdPs for each VO member, they proposed the use of VO role registration, where the VOs are generated by the admin and the group password is sent to the member of VO, out of the band. Hence, after the users have successfully authenticated through their respective IdPs, they can request to join a VO by providing the correct password. This solution is based on the set to set mapping of the attributes, which is complex to implement and requires a specialized API to be integrated into the current Openstack implementation. In another solution proposed in Heder et al [8], the SP is configured to retrieve information from independent Attribute Authorities (AAs). The Shibboleth SP collects all the attributes from the IdPs and the AAs, and forwards them to the module called regisite by making the use of SessionHook capability in Shibboleth. Based on the entitlement information received from the AAs, the regisite creates new users and/or tenants or update the existing users, tenants or roles. In this way, it creates and updated the user information in the backend, instead of creating ephemeral users as in the case of normal external authentication in Keystone. Some of the issues related to this approach include the implementation complexity, freeing up of resources when a user no longer exists, enabling command line access, etc. In [9], the authors proposed the use of WSGI filters to integrate VOMS with the keystone external authentication. In this case, the user authenticates using the VOMS proxy against the HTTPD server. On successful authentication, the mapping between the VOMS attributes (for example Distinguished Name DN, VO groups, VO roles, etc) and the local Openstack tenants is done to support the group based authorization. This solution is based on the standard VOMS, and hence can be useful in integrating heterogeneous cloud systems for collaborative research.

5. Conclusion

In this paper, we discussed various issues related to the Federated Identity Management in scientific cloud based collaborations involving heterogeneous cloud systems. The present work was focused on the evolution of identity management in the OpenStack cloud software from the traditional identity service purely based on the backend, to a more flexible external authentication support. We also discussed some of the issues in the current Federated Identity Management system of OpenStack, and discussed some of the proposed solution. The present work is aimed at providing an insight into the various design issues and ongoing work related to the OpenStack's identity management module. As a part of the future work, we aim at implementing a flexible solution in RealLab [11][12], which currently supports high performance collaborative research over the KREONET network [13].

References

[1] Peter Mell and Timothy Grance, " The NIST Definition of Cloud Computing",http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[2] "Openstack Open Source Cloud Computing Software", http://www.openstack.org/software/.

[3] " Keystone Architecture – keystone 9.0.1.dev42 documentation ", http://docs.openstack.org/developer/keystone/architecture.html

[4] David W. Chadwick, Kristy Siu, Craig Lee, Yann Fouillat and Damien Germonville, "Adding Federated Identity Management to Openstack," *Journal of Grid Computing*, vol. no. 12, pp.3-27, March 2014.

[5] "Federation of Openstack Clouds | CERN Openlab, "http://openlab.web.cern.ch/publications/technical_documents/federation-openstack-clouds".

[6] "Configuring Keystone for Federation–keystone 9.0.1.dev33 documentation",http://docs.openstack.org/developer/keystone/configure_federation.html.

[7] David W. Chadwick, Ioram S. Sette and Kristy W. Siu, "Opening up Openstack's Identity Service", https://kar.kent.ac.uk/52931/.

[8] Mihaly Heder, Szabolcs Tenczer and Andrea Biancini, "Collaboration Between SAML Federations and Openstack Clouds", https://arxiv.org/ftp/arxiv/papers/1510/1510.04017.pdf.

[9] Á L. García, E. Fernández-del-Castillo and M. Puel, "Identity Federation with VOMS in Cloud Infrastructures," *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, Bristol, 2013, pp. 42-48.

[10] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Angello, A. Frohner, A. Gianoli, K. Lorentey and F. Spataro, "VOMA, an Authorization System for Virtual Organizations", In Grid Computing, volume 2970 of Lecture Notes in Computer Science, pages 33{40. Springer, 2004.

[11] Ramneek and Yoonjoo Kwon, "RealLab: A High Performance Collaborative Research Environment based on Open Source Cloud Platform", International Conference on Future Web (ICFW 2014), Busan, 2014, pp.302-310.

[12] "Login-RealLab Dashboard", https://reallab.kreonet.net/horizon/.

[13] "Korea Research Environment Open NETwork (KREONET)", http://kreonet.net/kreonet/index.jsp

사물인터넷을 위한 다양한 응용층 프로토콜

하태영, 정종문

연세대학교 전기전자공학부

taeyoungha@yonsei.ac.kr, jmc@yonsei.ac.kr

요 약

통신기능이 탑재된 다양한 형태의 기기가 출시됨과 동시에 정보통신기술이 발전함에 따라 미래인터넷의 중요성이 강조되고 있다. 미래인터넷의 하나의 형태로 사물인터넷이 각광받고 있으며, 사물인터넷은 기존의 인터넷 서비스 또는 센서 네트워크와의 연동을 위해 상용화된 물리층 (Physical Layer), 데이터링크층 (Data Link Layer), 네트워크층 (Network Layer), 전송층 (Transport Layer) 프로토콜을 사용하는 반면에, 응용층 (Application Layer) 프로토콜들은 사물인터넷특성을 반영한 다양한 프로토콜들이 표준화 및 개발되고 있다. 본 논문에서는 사물인터넷을 위한 다양한 응용층 프로토콜의 특징을 비교해보고자 한다.

1. 서론

정보통신기술이 발전함에 따라서 사물인터넷 (Internet of Things)이 미래인터넷의 하나의 형태로 각광받고 있으며, 사물인터넷 서비스를 제공하는 개체들이 2020 년에는 전 세계적으로 2120 억개에 달할 것으로 예측된다 [1]. 뿐만 아니라 사물인터넷의 큰 부분인 사물통신 (Machine to Machine, M2M)이 2020 년에는 전체 인터넷 트래픽의 45%을 차지할 것으로 예측된다 [2]. 이러한 사물인터넷의 상용화를 위해 다양한 그룹에서 표준화를 진행하고 있으며, 특히 World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPCglobal, Institute of Electrical and Electronics Engineers (IEEE), 그리고 European Telecommunications Standards Institute (ETSI)에서 표준화를 주도하고 있다 [3].

사물인터넷은 기존에 존재하고 있는 다양한 인터넷 서비스와 센서 네트워크와의 연동을 위해 상용화된 프로토콜을 주로 사용하고 있다. 물리층 (Physical Layer)과 데이터링크층 (Data Link Layer) 프로토콜로 LTE/LTE-A, IEEE 802.11, IEEE 802.15.1, IEEE 802.15.4 등이 주로 사용되고 있으며, 네트워크층 (Network Layer) 프로토콜로는 IPv4/IPv6 또는 IPv6 over Low power WPAN (6LoWPAN)이 주로 사용되고 있다. 또한 전송층 (Transport Layer) 프로토콜 또한 상용 네트워크에서 주로 사용되는 TCP 또는 UDP 가 사용되고 있는 반면에, 응용층 (Application Layer) 프로토콜로는 사물인터넷을 위한 새로운 프로토콜들이 사용되고 있다. 본 논문에서는 사물인터넷을 위한 다양한 응용층 프로토콜 중에서 CoAP, MQTT, XMPP, 그리고 AMQP 의 특징을 비교해보고자 한다.

2. CoAP

CoAP 는 Constrained Application Protocol 의 약자로 IETF 의 Constrained RESTful Environments (CoRE) working group (WG)에 의해서 만들어진 응용층 프로토콜이다. REpresentational State Transfer (REST)는 Hypertext Transfer Protocol (HTTP)을 사용하는 서버와 클라이언트간의 데이터를 손쉽게 주고받을 수 있도록 제안된 기법이다. CoAP 는 이 REST 를 기반으로 한 응용층 프로토콜로, 비교적 오랫동안 그리고 널리 사용된 HTTP 를 기반으로 동작하기 때문에 CoAP 를 위해 새롭게 인프라를 구축하지 않아도 된다는 장점을 가지고 있다 [4].

CoAP 는 REST 와 다르게 사물인터넷 서비스에 적합하도록 UDP 를 기반으로 동작한다. 하지만 CoAP 는 REST 를 기반으로 동작 하기 때문에 네트워크 내에 REST-CoAP Proxy 를 둠으로써 REST 기반의 인터넷과 CoAP 기반의 개체들이 서로 쉽게 호환될 수 있는 장점을 가지고 있다.

3. MQTT

MQTT 는 Message Queue Telemetry Transport 의 약자로 2013 년에 OASIS 에 의해 표준화 되었다 [5]. MQTT 는 일대일, 일대다, 다대다 간의 라우팅을 제공하고 헬스케어, 센서 네트워크, Facebook 알림 등 다양한 응용서비스에 활용될 수 있어서 사물인터넷 또는 M2M 을 위한 연결 및 메시징 프로토콜로 적합하다고 여겨지고 있다 [3].

MQTT 는 TCP 를 기반으로 동작하며, 3 가지 Quality of Service (QoS) 모드를 제공한다. QoS 모드는 2 개의 bit 으로 나타낼 수 있으며, 0~2 의 값을 갖는다. QoS 값이 0 인 경우 메시지는 한번만 전달되며,

전달여부를 확인하지 않는다. QoS 값이 1 인 경우 메시지는 적어도 한번 전달된다. 이는 해당 메시지가 최소한 한번은 전송되나, 메시지가 중복 전송될 수 있음을 뜻한다. QoS 값이 2 인 경우 전달여부를 확인하여 메시지는 정확히 한번 전달된다.

4. XMPP

XMPP 는 Extensible Messaging and Presence Protocol 의 약자로 W3C 에서 개발된 Extensible Markup Language (XML)을 활용한 응용 프로파일 (Application Profile) 이다 [6]. XML 을 통해 근 실시간 (Near-Real-Time)으로 두 개 이상의 네트워크끼리 확장 가능한 구조화된 데이터를 전송할 수 있다. XMPP 는 IETF 에 의해 2002 년에 표준화 되었으며, 인스턴트 메시징을 위한 국제 표준 프로토콜이다. 처음에는 1999 년에 Jabber 라는 이름을 갖고 사용자의 운영체제에 상관없이 인터넷을 통해 메시지를 전송할 수 있도록 오픈 소스로 개발되었다.

XMPP 는 TCP 를 기반으로 동작하며, 'XML stanzs'라는 상대적으로 작은 크기의 구조화된 데이터를 주고받기 위한 프로토콜이다. XML 을 사용하는 XMPP 를 활용한 문자기반의 통신은 네트워크 오버헤드가 크다는 단점이 있다.

5. AMQP

AMQP 는 Advanced Message Queuing Protocol 의 약자로 메시지 기반의 미들웨어를 위한 오픈소스로 개발된 표준 프로토콜이다. AMQP 는 XMPP 와 같이 오프소스를 기반으로 한 프로토콜이지만 다음의 차이점을 갖는다 [7]. XMPP 는 일반 사용자간의 인스턴트 메시지 서비스와 같은 가벼운 용도로 적합한 반면에, AMQP 는 XMPP 에 비해 더 나은 시간당 처리량 (Throughput), 확장성 (Scalability)과 안정성 (Reliability)을 제공하기 때문에 상용 메시지 서비스에 적합하다. 이는 XMPP 와 AMQP 가 모두 TCP 를 기반으로 동작하지만, XMPP 는 QoS 보장을 위한 기능을 제공하지 않지만, AMQP 는 MQTT 와 동일한 3 가지 QoS 모드를 제공하기 때문이라고 할 수 있다 [3].

6. 결론

본 논문에서는 사물인터넷을 위한 4 가지 응용층 프로토콜인 CoAP, MQTT XMPP, AMQP 의 특징에 대해 간략히 알아보았다. 각각의 프로토콜의 특징은 표 1 과 같이 정리할 수 있다. 개발자들은 자신이 제공하려는 응용서비스를 개발 함에 있어 다음의 사항들을 고려하여 응용층 프로토콜을 선택해야 한다. 첫 번째로 서비스를 제공하는 개체의 하드웨어 성능을 고려하여 각각의 응용층 프로토콜이 사용하고 있는 패킷 특성 (패킷의 크기, 발생주기, 교환 순서)을 반영해서 응용층 프로토콜을 선정한다. 두 번째로 표 1 에 정리된 각각의 응용층 프로토콜 사용하는 전송층 프로토콜, 보안 프로토콜의 특성과 QoS 지원 여부를 고려하여 응용층 프로토콜을 선택해야 한다. 이와 같은 특성을 반영하여 응용층 프로토콜을 선택하여 사용한다면, 제공하는 서비스를 이용하는 사용자에게 더 나은 사용자 경험을 제공할 수 있을 것이다.

응용층 프로토콜	전송층 프로토콜	보안 프로토콜	QoS 지원 여부
CoAP	UDP	DTLS	O
MQTT	TCP	SSL	O
XMPP	TCP	SSL	X
AMQP	TCP	SSL	O

표 1 사물인터넷을 위한 응용층 프로토콜 비교

ACKNOWLEDGMENT

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1009, ICBMS 플랫폼 간 정보 모델 연동 및 서비스 매쉬업을 위한 스마트 중재 기술 개발)

7. 참고 문헌

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Anal. Future*, vol. 2007, pp. 1–16, Dec. 2012.

[2] S. Taylor, "The next generation of the Internet revolutionizing the way we work, live, play, and learn," CISCO, San Francisco, CA, USA, CISCO Point of View, 2013.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015.

[4] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny Internet nodes," *IEEE Internet Comput*., vol. 16, no. 2, pp. 62–67, Mar./Apr. 2012.

[5] D. Locke, "MQ telemetry transport (MQTT) v3. 1 protocol specification," IBM developerWorks, Markham, ON, Canada, Tech. Lib., 2010. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html

[6] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," Internet Eng. Task Force (IETF), Fremont, CA, USA, Request for Comments: 6120, 2011.

[7] S. Vinoski, "Advanced Message Queuing Protocol," *IEEE Internet Comput*., vol. 10, no. 6, pp. 87-89, Nov.-Dec. 2006.