역량 포트폴리오

박헌일

목차

■ 자기 소개 및 보유 기술

박헌일 (Heonil Park)

주니어 소프트웨어 엔지니어

heonil89@gmail.com
heonil8@nate.com

linkedin.com/in/heonil

github.com/heonilp

주요 경력

2009.3~2015.2 한국외대 정보통신공학
차세대 인터넷 연구실
2016.9~2018.8 한양대 컴퓨터소프트웨어
MIRLAB(네트워크 연구실)
2019.3~2019.6 네오위즈 블록체인 연구
2019.8~2019.11 넥슨 인텔리전스랩스

역량 : 협업 능력 + 업무 능력

여조삭비(如鳥數飛) 배움은 끊임 없이, 늘 겸손한 자세로 배움을 실천
가치를 기여하는 사람

■ 자기 소개 및 보유 기술

2011

-광고동아리 기획팀장
(Must Have item)
-입대(통신병)

2013

캡스톤 디자인
옷장 관리 프로젝트
(안드로이드 앱 개발
DB, 임베디드)

2015

학부 졸업
게임아카데미
대용량 서버프로그램 구현

2017

-한국통신학회 논문 3편 (MQTT, CoAP, HTTP)
-한국통신학회 하계학술대회 논문 1편
논문 우수상(스마트그리드)
-네이버 기술 장학생상(개발 및 연구성과 우수)

2009

-대학 입학
2009~2015
-대학교 광고동아리
애드밸리 '기획'
-대학교 봉사 동아리
참동이

2012

-제대(통신병)
-웹 프로젝트
(웹 서버)

2014

-차세대 인터넷 연구실(2014~2015)
-학부 졸업 논문 빅데이터 관련 논문
-OCP 11g 자격증(DB)
-JAVA, 빅데이터 교육
(한국생산성본부)

2016

-대학원 입학
MIRLAB(2016~2018)
-네트워크 대표조교
-연구실 대표(~18.3)
-과제 실무담당자(2건)
-프로토콜(표준화) 개발연구
(스마트그리드 IoT, SDN)

2018-2020

-석사졸업논문
-표준화 기고 1건
-학위논문우수상
-석사 졸업(18.8)
-네오위즈 블록체인팀 인턴
-넥슨 데이터인프라개발 인턴
-현재 AWS, 알고리즘 학습중

활동 : Mobile Intelligence and Routing LAB(http://mir.hanyang.ac.kr)
논문 3건, 학술대회 논문 1건, 단기강좌 2건, 워크샵 발표 1건, 정부 과제 2건, 네이버 기술 장학생상 1건, 학위 논문상 1건
JAVA 빅데이터 교육,AWS 교육, GCP스터디잼 수강, 논문 우수상 1건,사용가능 언어: C, C++, JAVA, Python, Go, PHP, MySQL, 주 개발환경 : AWS, GCP

- 학사 논문: Hadoop 및 Mongo DB 기반 대용량 로그 처리 시스템 설계 및 구현
- 석사 논문: 에너지 IoT 기반 집합 건물 환경에서 효율적인 수요반응을 위한 경량 에너지관리에이전트 프로토콜 설계 및 구현
- 논문: 스마트 에너지 IoT 를 위한 CoAP 기반 Lightweight OpenADR2.0b 프로토콜의 구현 및 분석. 한국통신학회논문지, 42(4), 904-914
- 논문: 에너지 IoT 환경에서 실시간 수요반응 서비스를 위한 CoAP Observe 기반 Push Mechanism 설계 및 분석. 한국통신학회논문지, 43(3), 529-540
- 논문: 에너지 IoT 환경을 위한 Multicast 방식의 Lightweight 수요반응 프로토콜 제안 및 분석. 한국통신학회논문지, 43(7), 1163-1175.
- 한국통신학회 하계 학술대회(우수논문상) : 스마트 에너지 IoT 환경에서 OpenADR 2.0 b 수요반응의 Push 메카니즘 필요성 연구". 한국통신학회 학술대회논문집(2017), 283-284
- 단기강좌 강의 : IoT 기반 OpenADR 단기 강좌(TIPS 최진식, 박헌일, 박현진, 이성환, 이재조박사(한국전기연구원), OSIA), 2017. 7. 3-4
- 단기강좌 강의 : SDN 단기 강좌(TIPW, 최진식, 조호준, 이양, 천세준, 박헌일, 박현진, 이성환, ATTO Research, OSIA), 2018. 2. 26-27
- 워크샵 발표 : Design and Analysis of Push mechanism based on CoAP Observe for Demand Response in Energy IoT Environment, 2017 Waseda-UKM-UMS-Hanyang IT Workshop, 2017. 12. 1-2
- 정부 과제 :스마트그리드와 연계된 스마트가전 인터페이스 시험방법 표준개발 (**2016.9.1~2018.3.31**), MIRLAB 대표, 실무담당자 참여
- 정부 과제 : IoT 구축을 위한 초고신뢰성 패킷네트워크 개발의 사실 표준화 (**2016.9.1~2018.2.28**), 실 MIRLAB 대표, 실무담당자 참여

목차

■ 네오위즈 블록체인 서버 개발 담당 업무 및 프로젝트 설명(2019), 4명 진행

내용 : Blockchain에서 RNG(Random Number Generation, 난수 생성)기반 서비스 구현

■ 블록체인 서버 개발 담당 업무 및 프로젝트 설명(2019)

담당 업무: 블록체인 스마트 컨트랙트 설계 및 개발, 기획, EOS 메인넷 경험, GCP, QA 운영
개선점 : TDD 기반 설계 및 개발(유닛테스트), Gitlab이용, 페어프로그래밍으로 알고리즘 고도화

eosio

알고리즘 구현

Unit-Test

EOS-js API

EOSIO 스마트 컨트랙트

로직 검증
블록 정보저장

1. GCP VM EOS 테스트 망

Google Cloud Platform

2. EOS 정글넷

3. EOS 메인넷

EOS

mongoDB

redis

socket.io

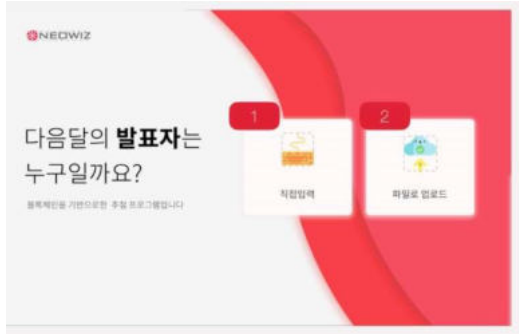
Web Back-end

WebSockets

node JS

(GCP VM node JS서버)

Google Cloud Platform

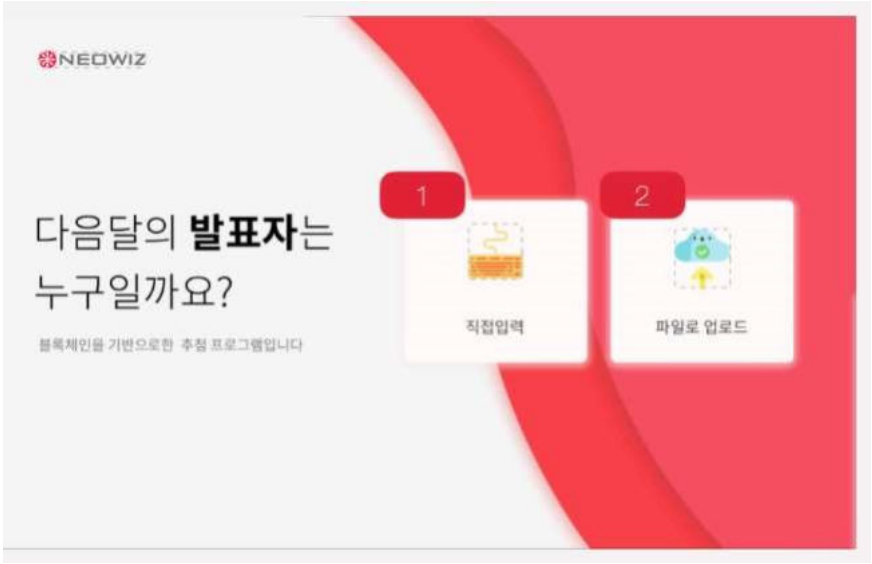


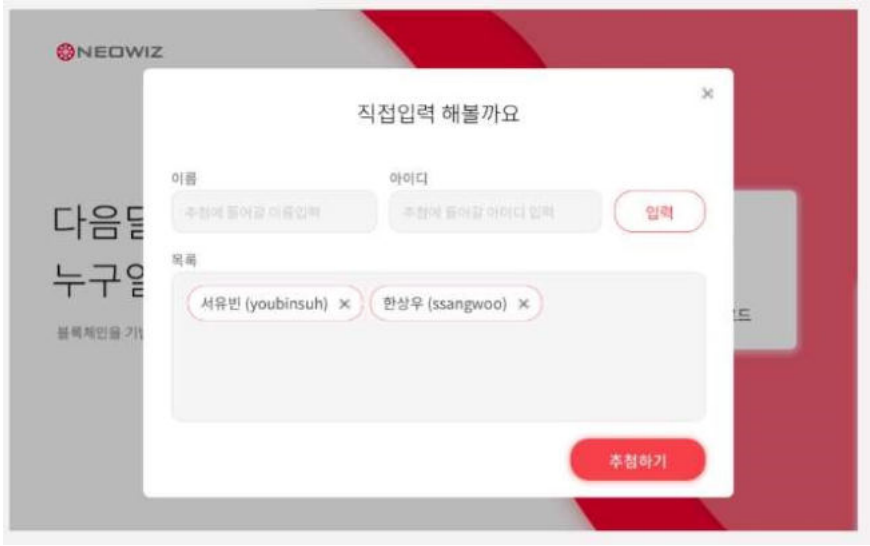
Web Front-end

React JS

iOS

■ 블록체인 기반 사내 월간 회의 발표자, 주차장 추첨 서비스 개발

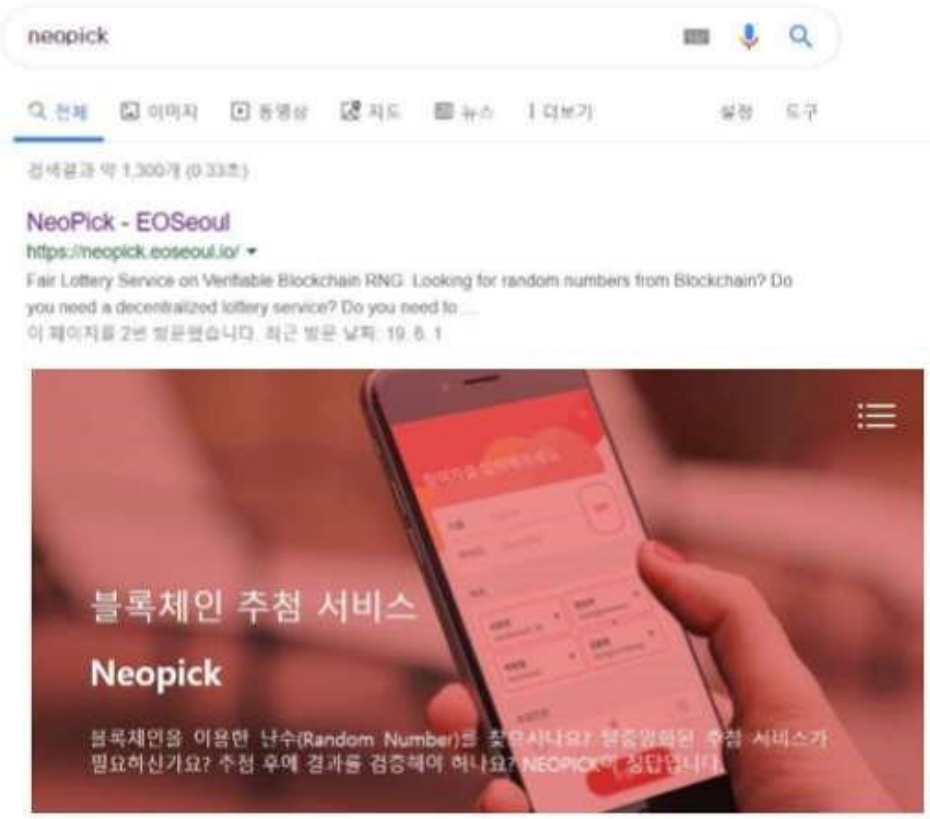




주요 개발 언어 : **C/ C++,** 자바스크립트
개발 환경 : 리눅스**, MAC OS, GCP**(구글 클라우드 플랫폼)

■ 블록체인 기반 추첨 서비스 웹앱 개발(실시간 서비스)

네오픽(Neopick) : 블록체인 기반 추첨 뽑기 서비스(4명 작업 진행)

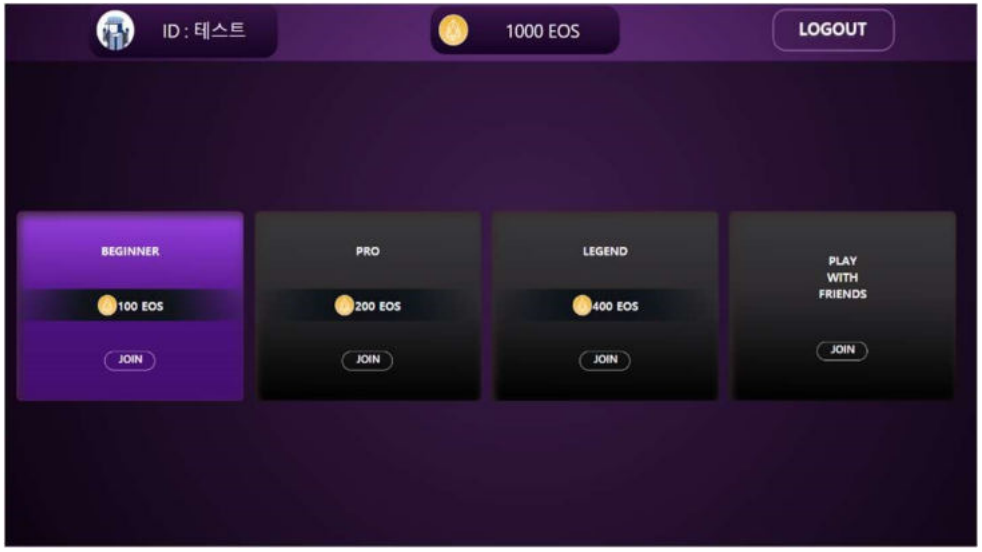


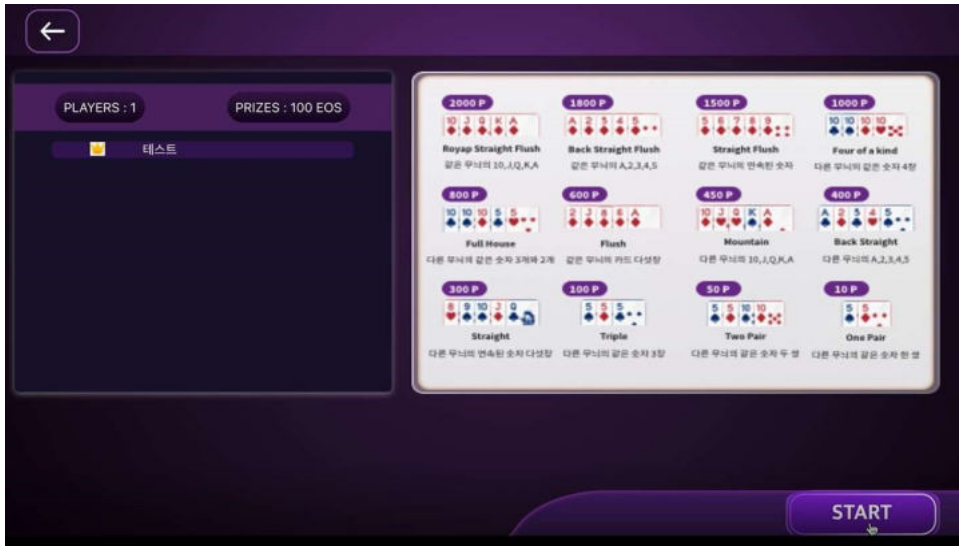
직접 사이트에 가서 나온 난수 값을 확인

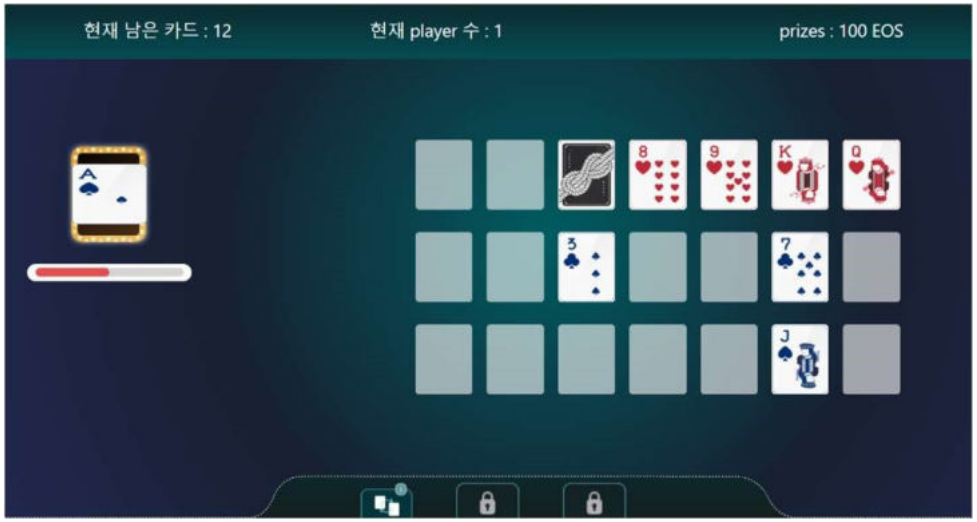
e0900775	May-29-2019, 06:11:49 PM	neopickrng33 - checkrns	**event_id:** 376 **offset:** 3 **randnums:** 0.730996,0.548065,0.415534 **rns_sig:** SIG_K1_KVA56ZGUytL6zDH66da9L5tVVDp6GhHF8Tdw6Swj2Sd3tkYbVqJYsYF3
b63c6b41	May-29-2019, 06:11:45 PM	neopickrng33 - setseed	**binfo:** **event_id:** 376 **hash_sig:** SIG_K1_KcPjcYtR1EGgyZnZFVm8DsMuN4QxRkjH46zai2gMbfBxDZumMX4hzkjc
e07418fc	May-29-2019, 06:11:41 PM	neopickrng33 - closeevent	**event_id:** 376
022b528e	May-29-2019, 06:11:38 PM	neopickrng33 - addentries	**entries:** 밥먹는다,1 밥안먹는다,2 외식,3 **event_id:** 376 **offset:** 3

주요 개발 언어 **: C/ C++,** 자바스크립트
개발 환경 **:** 리눅스**, MAC OS, GCP(**구글 클라우드 플랫폼**)**

■ 블록체인 기반(EOS)카드 빙고게임(Roper In the Card)





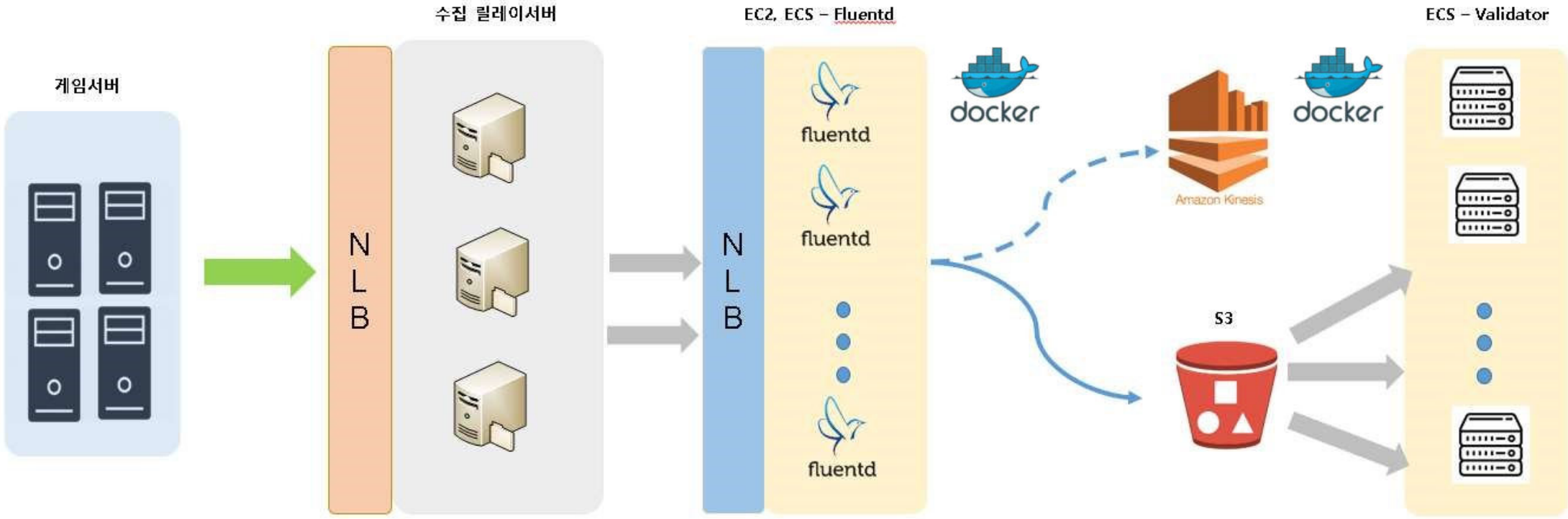


주요 개발 언어 : **C/ C++, Node.js(Express.js), EOS-js API, Socket.IO, MongoDB, Redis(+Bee-queue)**
개발 환경 : 리눅스, **MAC OS, GCP**(구글 클라우드 플랫폼)

■ 넥슨 수집, 적재 개발 데이터엔지니어 담당 프로젝트 설명(2019)

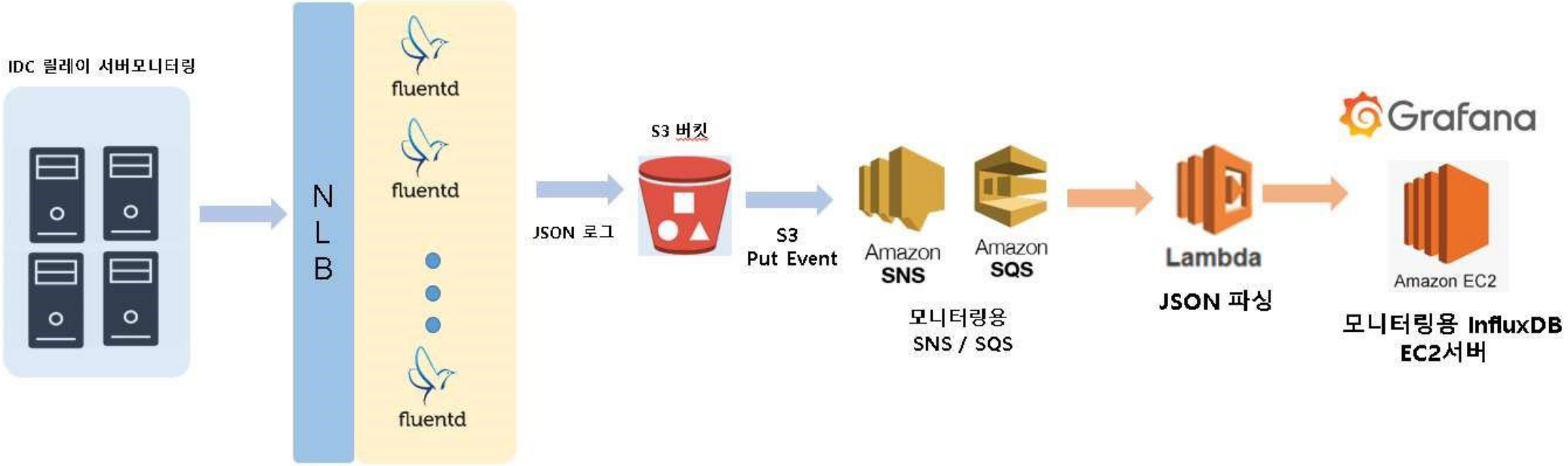
-담당업무 : 수집 릴레이서버 개발, 유지보수, Fluentd, Spark 운영, JIRA, Confluence, Gitlab, Jenkins 사용
-개발 환경 : C++, GO언어, Fluentd, Spark, AWS(SNS, SQS, Lambda, S3, EC2, ECS 등)
-데이터 : 대용량 게임 로그 및 시스템 로그 (JSON) 수집 및 적재
-개선점 : IDC 모니터링 서버 개발, 슬랙 알림 봇 제작, 릴레이 서버 유지보수(멀티 스레드 및 더미 테스트 기능)

게임 로그데이터 수집, 적재, 검사 전송 예시



■ 릴레이서버 모니터링 툴 개발 상세 내용

문제점: IDC센터에 있는 윈도우 기반 서버가 모니터링 기능이 없어 퍼블리셔들이 직접 접속해서 문제점을 파악, 보안 문제
해결 방안: 기존 망에 모니터링 툴을 개발하여서 릴레이 서버의 상태를 모니터링하는 서비스 제공(대시보드, 슬랙 알람 제공)
요구사항 : 릴레이 서버 운영을 위한 시스템 모니터링 로그를 Fluentd Forward Protocol 에 실어서 전송 및 AWS망 이용



■ 릴레이서버 부하 테스트 툴 개발 내용

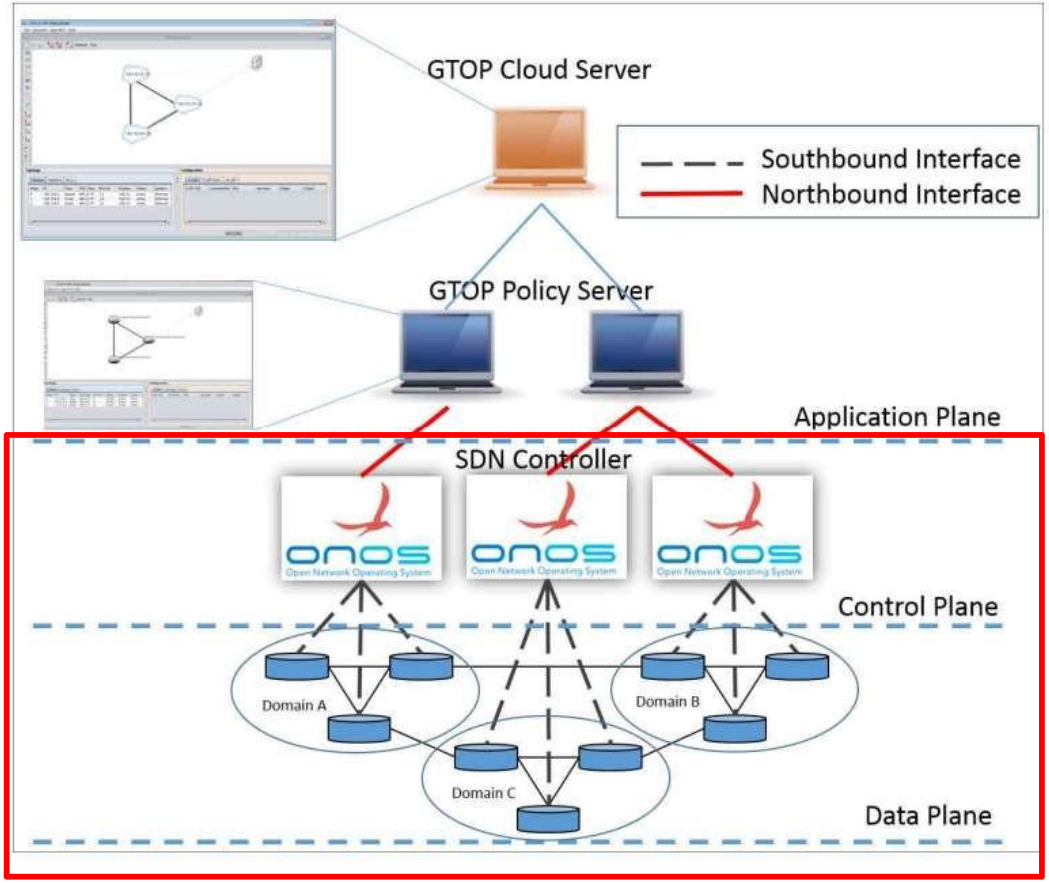
요구사항 및 개선점: 실제 사항을 대비해 릴레이서버에 대한 테스트 툴 개발 및 BPS, PPS 기반 기능 .
기술 : C++ Boost Asio, Strand 기반 대용량 서버, 클라이언트

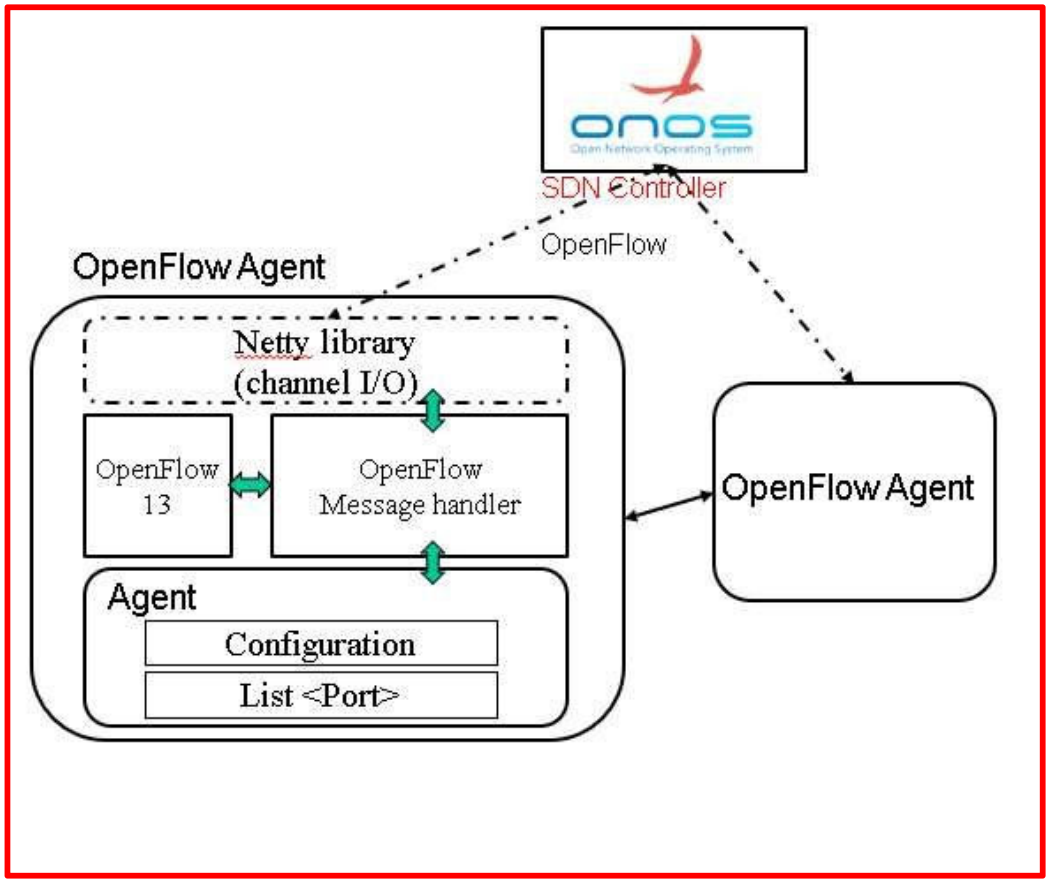
목차

■ 석사: SDN(Software Defined Network) 프로젝트(2016~2018)

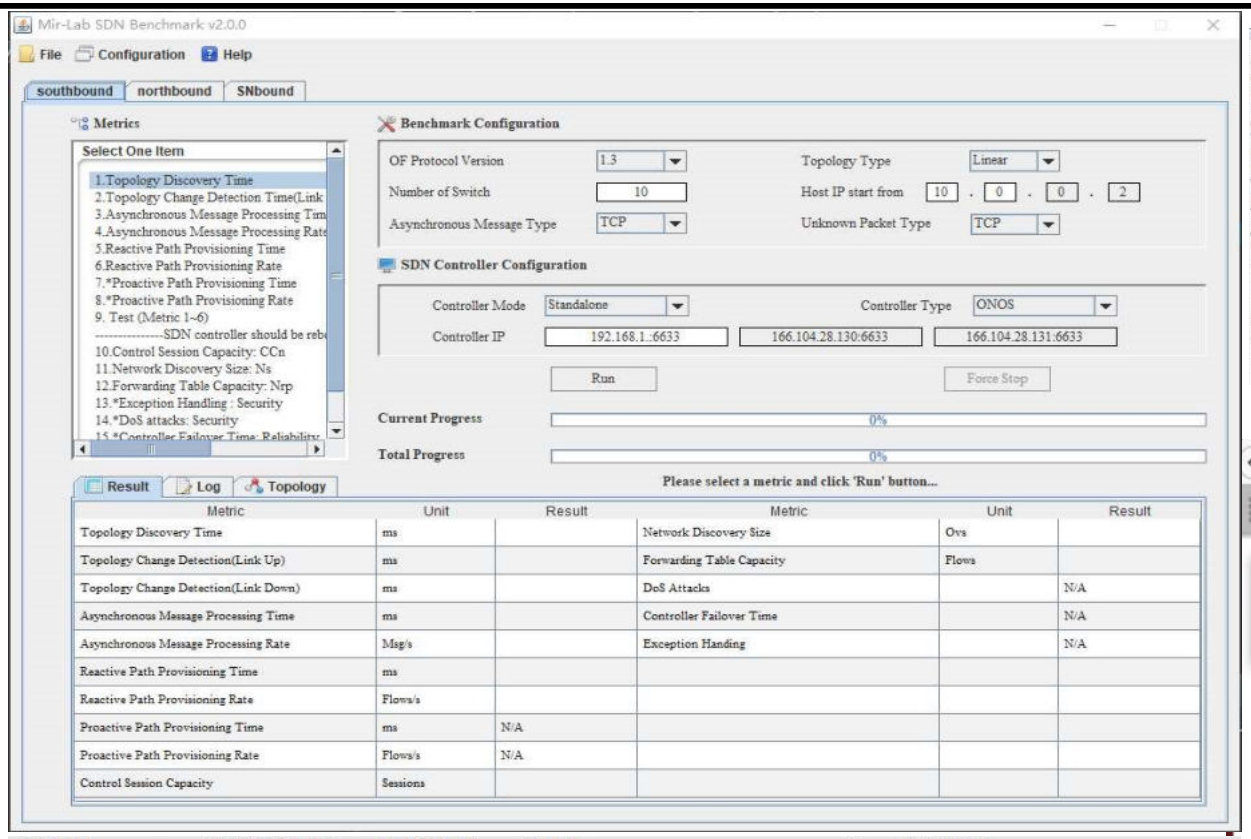
SDN(소프트웨어 정의 네트워킹)은 소프트웨어 애플리케이션을 사용하여 네트워크를 지능화하고 중앙에서 제어하거나 '프로그래밍' 할 수 있는 네트워크 아키텍처 접근법입니다. 이를 통해 사업자는 기본 네트워크 기술에 상관없이 전체 네트워크를 일관적이고 전체적으로 관리할 수 있습니다.

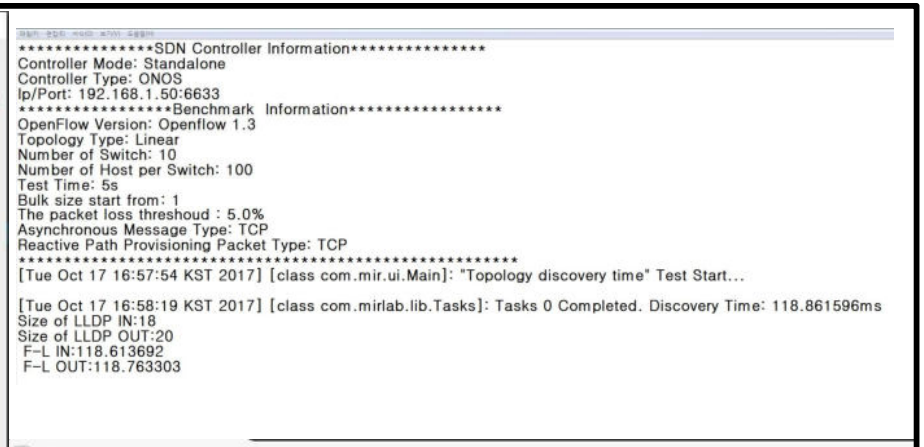
-담당 업무 : ONOS(네트워크 컨트롤러) 부하 테스트- 가상 스위치 개발
-개발 환경 : JAVA Netty 서버, 미니넷, OpenVSwitch(OpenWRT) 개발
-개선점 : 오픈소스 SDN 컨트롤러인 ONOS의 부하테스트를 하였고 벤치마킹 표준화 도움

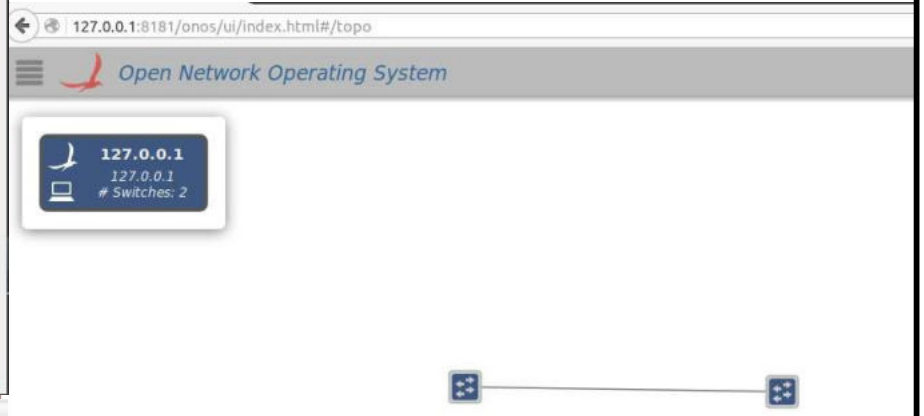




■석사: SDN(Software Defined Network) 프로젝트, 벤치마킹







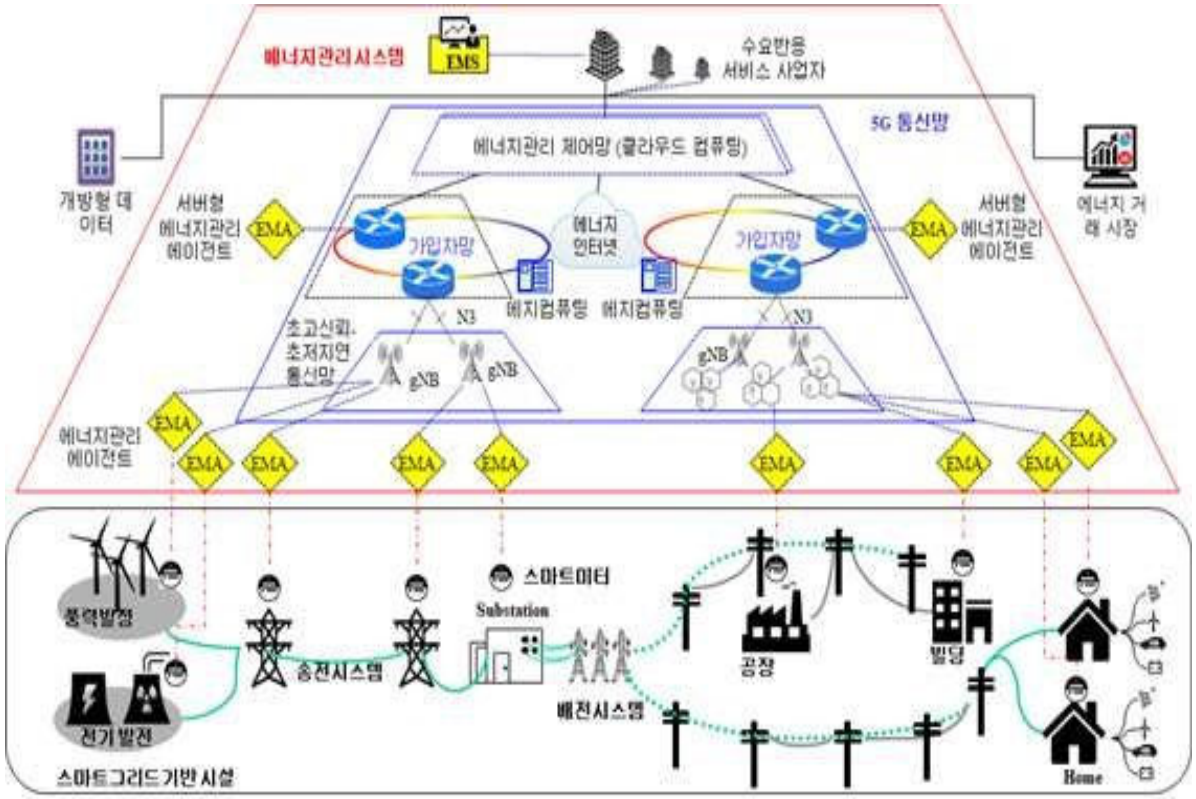

```
14 *REF*      62 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_HELLO
16 0.001206   70 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_HELLO
17 0.001486   62 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_FEATURES_REQUEST
19 0.002157   86 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_FEATURES_REPLY
20 0.002477   70 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_DESC
21 0.002900   70 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC
22 0.003292   82 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_GET_CONFIG_REQUEST
23 0.003773   62 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_BARRIER_REPLY
24 0.003839   66 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_GET_CONFIG_REPLY
26 0.005139   70 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_MULTIPART_REQUEST, OFPMP_DESC
27 0.005688 1126 192.168.1.105  192.168.1.50   OpenFlow  Type: OFPT_MULTIPART_REPLY, OFPMP_DESC
28 0.013999   78 192.168.1.50   192.168.1.105  OpenFlow  Type: OFPT_ROLE_REQUEST
```

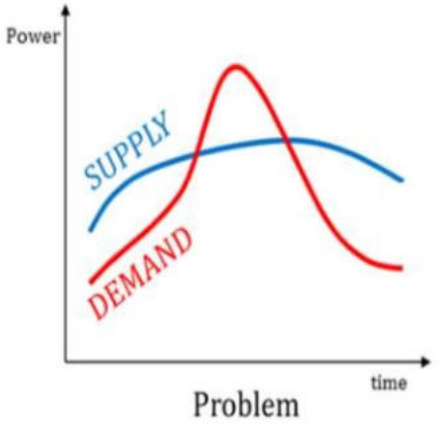
-개발 환경 : JAVA Netty 서버, 미니넷, OpenVSwitch(OpenWRT) 개발
- 리눅스**, OpenWRT,** 라즈베리파이

■ 석사: 스마트 그리드 프로젝트(2016~2018)

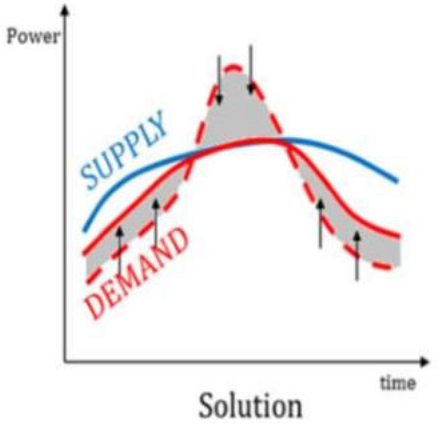
-담당 업무 : IoT 기반 에너지 프레임워크(스마트그리드) 개발, 수요반응 기반 에너지관리에이전트 개발, 분산 전원 개발
-개발 환경 : C++(임베디드, 크로스컴파일), JAVA(GUI, 서버) , 실험 환경 : 리눅스, OpenWRT, AWS, Docker
-목적: 연구, 에너지관리에이전트 관련 ISO/IEC 국제 표준화 도움(ISO/IEC 15067-3, ISO/IEC 15067-3-3)
에너지관리에이전트 프로토콜 및 에너지 프레임워크 및 구조 제안

수요반응 : 증가하는 전력 수요와 피크부하에 대응하고 안정적인 전력망을 구축하기 위한 방법
전력 소비자 측에서 전기 요금이나 공급자의 요청에 따라 전력 소비를 조절함으로써 전력계통의 안정성과 신뢰성을 확보하는 기술





(a) Demand and supply problem



(b) Solving through Demand Response

■ 석사: 에너지IoT 학위 논문 프로젝트(연구 배경 및 필요성)

에너지 IoT 기반 집합 건물 환경에서 효율적인 수요반응을 위한 경량 에너지관리에이전트(EMA) 프로토콜 설계 및 구현

(1)집합건물 환경의 에너지 관리의 필요성 : 블랙아웃(정전), 전력 수요 증가, 에너지 수요관리 대책 강구

집합 건물 환경의 지능화된 에너지 관리(수요반응)가 필요(주거공간의 에너지관리에이전트 집중화 필요)
-여름철이나 겨울철 냉난방기기 판매량과 사용량 급증에 따른 블랙아웃은 심각한 문제를 발생
-집합 건물과 일반 가정 사용량이 많게는 피크 시간대 전력수요의 60%를 차지함
-수요반응, OpenADR(Open Automated Demand Response)프로토콜 등장

(2)국민 DR를 통한 수요반응 영역 확대 및 에너지 IoT, 경량수요반응 프로토콜의 등장

(경량 OpenADR2.0b CoAP/JSON, MQTT/JSON)
-제한적인 IoT환경, 디바이스나 IoT기기, 게이트웨이 ->배터리와 제한된 리소스
-에너지관리에이전트 기술 →에너지 IoT 기술 발전(에너지 분야와 IoT 분야와 융합)
-Utility 회사 중심의 송배선 산업설비에서 에너지 관리 플랫폼인 에너지관리시스템(EMS)은 효율적 에너지 관리하기 위해 사용

(3)에너지관리 에이전트 표준화 및 EMA Protocol 표준화가 진행(ISO/IEC 15067-3, ISO/IEC 15067-3-3)

-주거지역, 집합건물, 대학캠퍼스, 빌딩 등에 Interacting EMA을 통해 에너지관리와 수요반응에 대한 표준화가 등장(다양한 구조, 계층화, 분산화)
(ISO/IEC 15067-3-3의 Model of a system of interacting Energy Management Agents)
-상호 작용하는 EMA는 주거지역, 집합건물, 대학캠퍼스, 빌딩 의 지역 사회의 에너지관리와 수요반응을 관리함.

(4)집합건물 내에 수 많은 수요관리 고객을 수용할 수 있는 수요반응 프로토콜(EMAP)이 필요함

-수요관리사업자가 모두 수요 관리해주기 위해서는 복잡성 증가(병목현상이나 서비스 지연으로 인한 블랙아웃 방지, 서버의 과부하 방지)
-다단계 구조인 가정과 집합건물들에 대한 Aggregator(중개업자)가 필요(통합적 수요반응(집중화 필요))
-기존 주거공간의 단일 에너지관리에이전트 → 다단계 서버/클라이언트 에너지관리에이전트의 개념이 필요
-다양한 수요참여고객의 증가 및 집합건물 환경에서 다양한 서비스 필요→ 에너지관리에이전트 간 경량 수요반응 프로토콜의 전송 메커니즘이 필요
-수요반응 참여고객의 보안성 문제, 서비스 요구 사항 등 발생(기존 연구는 Explicit 사용) → Implicit/Explicit 모니터링이 필요

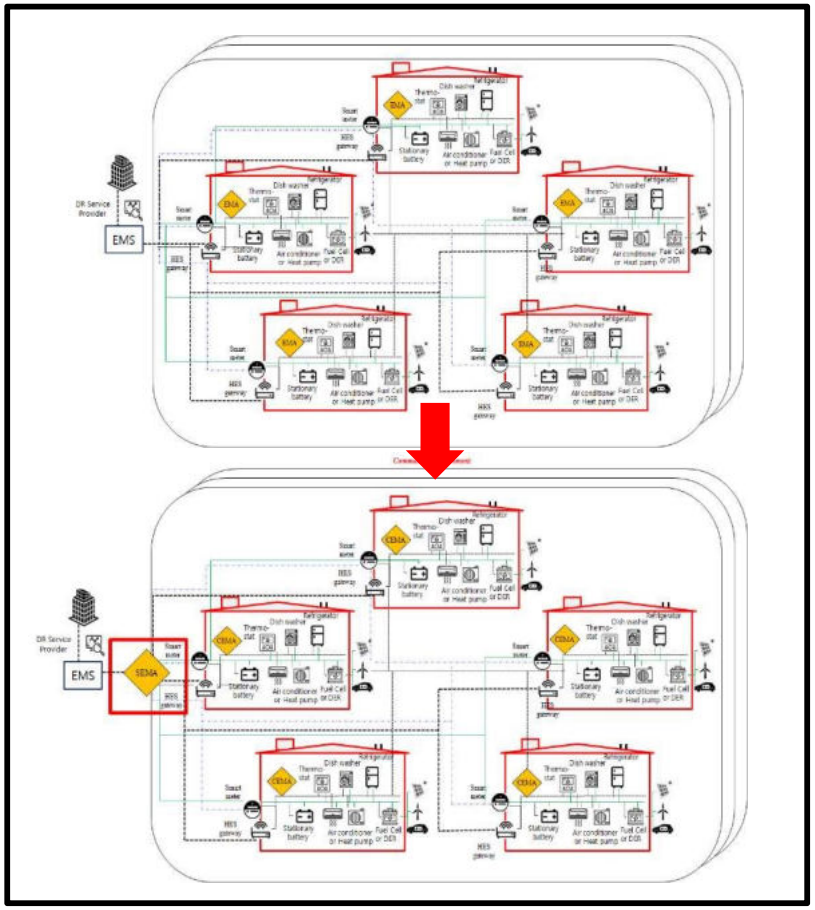
■ 석사 학위 논문 프로젝트 (논문 총 정리)

석사 학위 논문(2018.08.17)
에너지 IoT 기반 집합 건물 환경에서 효율적인 수요반응을 위한 경량 에너지관리에이전트(EMA) 프로토콜 설계 및 구현

목적: (1)집합 건물 환경으로 수요반응을 하는 목적 (수요반응 영역 확대, 계층 분산화)
(2)에너지 관리 에이전트 프로토콜(EMAP)의 목적

구현: (1)경량 수요반응 프로토콜 기반 OpenADR 2.0b와
에너지관리에이전트 프로토콜의 4가지 수요반응 서비스 구현
(2)서버 에너지관리에이전트(Aggregator)의 기능적 중계 기능 구현
(3)에너지관리에이전트의 Explicit 모니터링 문제점
-Implicit/Explicit 모니터링 통신 기능 구현

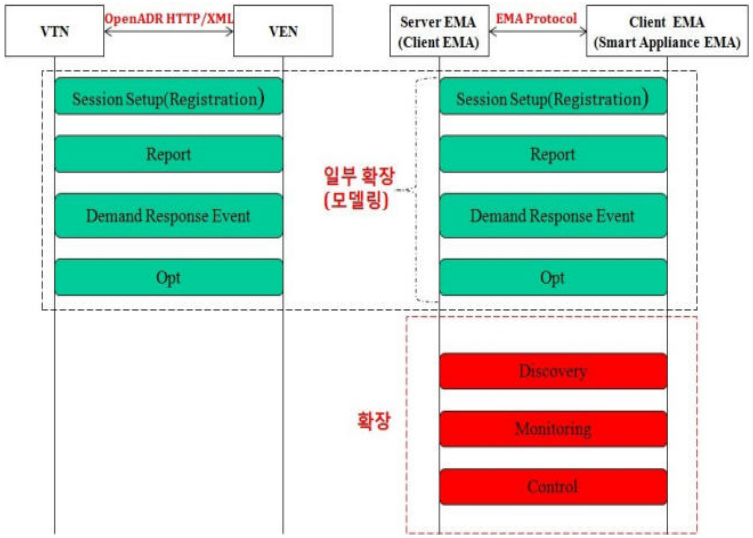
실험: (1)경량 수요반응 프로토콜의 데이터 트래픽 및 지연 시간 분석 : 경량 프로토콜 우수
(2)집합 건물 내 이벤트 수요반응 전송 메커니즘 비교 실험 : CoAP 우수
(3)에너지관리에이전트의 Implicit/Explicit 통신 비교 실험 : 병목현상↓, 보안성↑

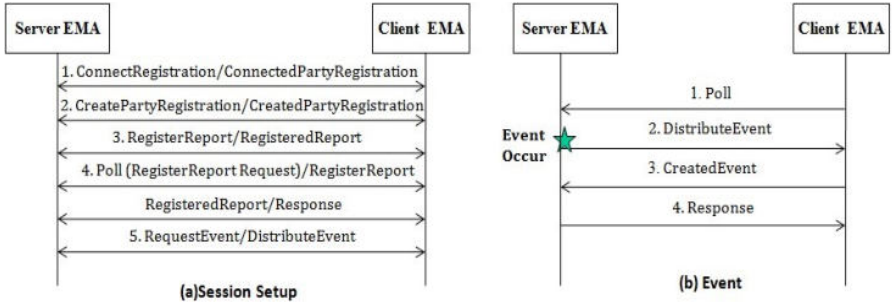


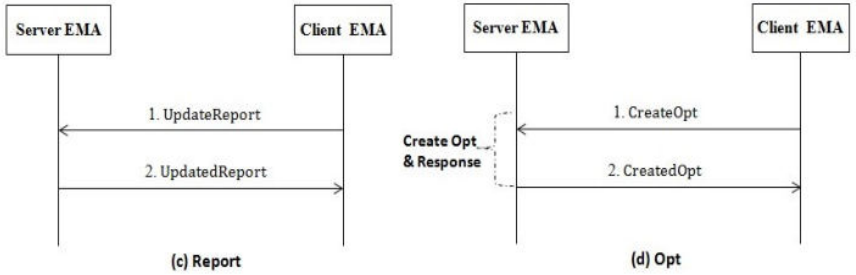
결론
(1)에너지관리에이전트을 통해 집합 건물 환경에서도 지능화된 에너지관리인 수요반응을 함
(2)에너지 관리 에이전트 프로토콜을 통해 수많은 집합건물의 수요관리 고객에 대한 실시간 수요반응을 할 수 있음(Push 실험을 통해 증명)
(3)집합 건물 환경에서 에너지관리에이전트 간 통신을 통해 수요반응 및 통합적 스케줄링을 할 수 있음
(4)집합 건물 환경 에서 EMA 프로토콜을 경량 IoT 프로토콜인 CoAP, MQTT 이용
-Heavyweight인 HTTP/XML 방식의 수요반응 프로토콜은 초소형 에너지 기기에 제한이 있음(OpenADR2.0b, SEP2.0)
-디바이스나 IoT기기, Gateway에는 제한된 리소스 및 배터리 용량이 제한, IoT(CoAP, MQTT) 프로토콜이 lightweight, 저전력 기반
(5)제한적인 IoT환경에서 데이터 트래픽, 수요반응 전송 속도를 줄이기 위해서 IoT 프로토콜의 메커니즘 비교를 통해
적합한 프로토콜인 CoAP/JSON 이용- (이유 : 집합건물의 수많은 개인지역으로 확대(수많은 노드) , 효율적 수요반응 관리가 필요)
-1. CoAP, MQTT 기반 lightweight, 2. CoAP Observe, 3. MQTT Broker 기반 Push 메커니즘, 4. MQTT 기반 멀티캐스트
(6)기존연구의 Explicit모니터링과 에너지관리에이전트 프로토콜의 Implicit/Explicit 모니터링 비교를 통해 데이터 추상화기능의 필요성을 증명

수요반응 (DR - Demand Response) -증가하는 전력 수요와 피크부하에 대응하고 안정적인 전력망을 구축하기 위한 방법

■ 논문 상세 내용(집합 건물 환경의 경량 에너지관리에이전트 프로토콜 구현)
- 에너지관리에이전트 프로토콜의 수요반응 4가지 서비스 (차이)OpenADR2.0과 비교





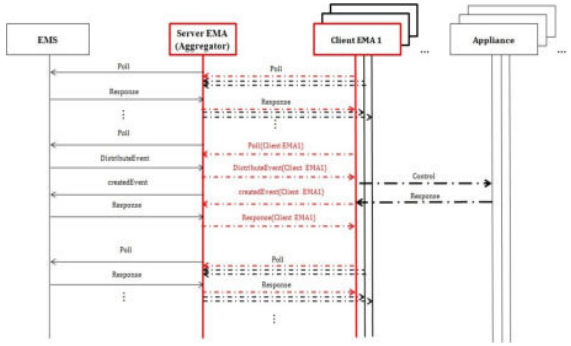
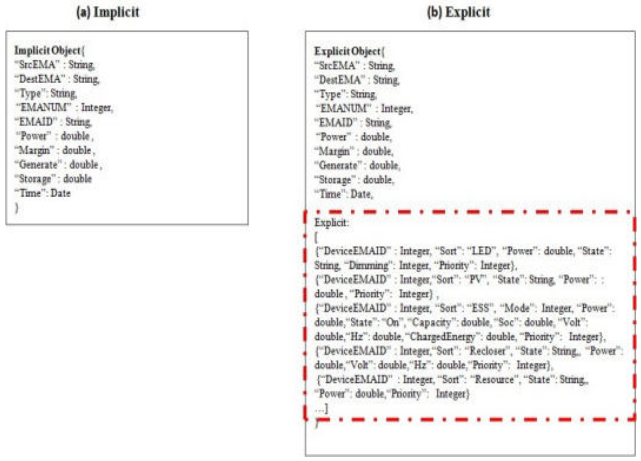


- (a)Session Setup(Registration 과정)
  - 에너지관리에이전트 간 서로 연결을 수립하고 Report을 교환할 때 **기존의 Report에 대한 에너지 가격 정보나 클라이언트 에너지관리에이전트의 에너지 관리 정보를 얻음.**
- (b)Event(Demand Response Event)
  - **에너지관리에이전트 프로토콜은 다양한 이벤트 종류가 있다.** 예로 가격기반 수요반응 이벤트에서 Initial Price, Incentive Price, Negotiation Price가 있다. 서버 에너지관리에이전트는 실시간적으로 가격에 대한 정보를 이벤트로 발생
- (c)Report(updateReport)
  - 에너지관리에이전트 간 서로 연결을 수립 할 때 **Report을 교환할 때 실시간 에너지에 대한 가격 정보나 클라이언트 에너지관리에이전트의 디바이스 정보 등을 얻음.**
- (d)Opt(수요반응 가용상태 및 스케줄링)
  - 에너지관리에이전트 프로토콜의 Opt는 클라이언트 에너지관리에이전트가 상위 서버 에너지관리에이전트에게 수요반응 이벤트의 가용상태 또는 **수요반응 이벤트 프로그램 변경, 수요반응 스케줄링을 요청을 알려주는 서비스**

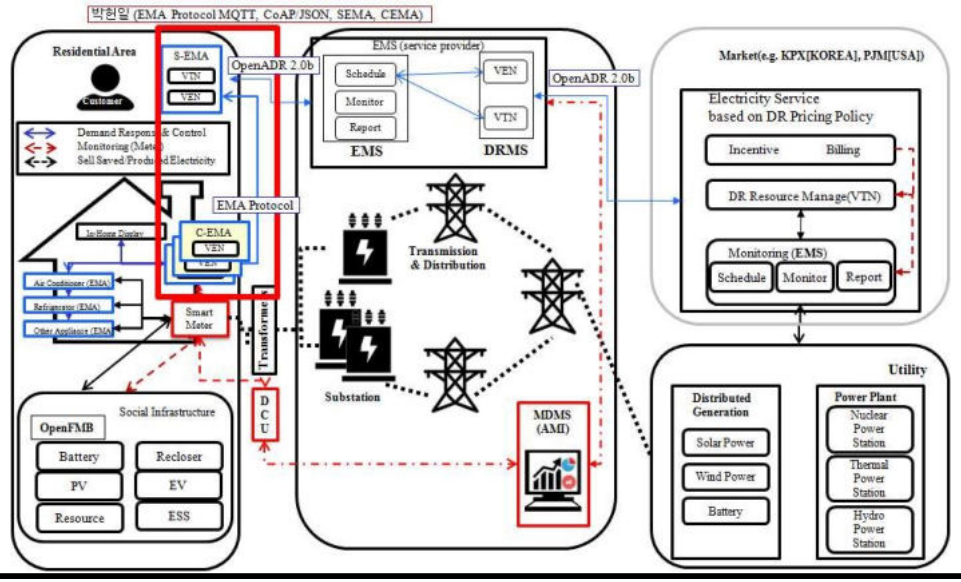
■ 논문 상세 내용(기존 문제점 및 해결책)

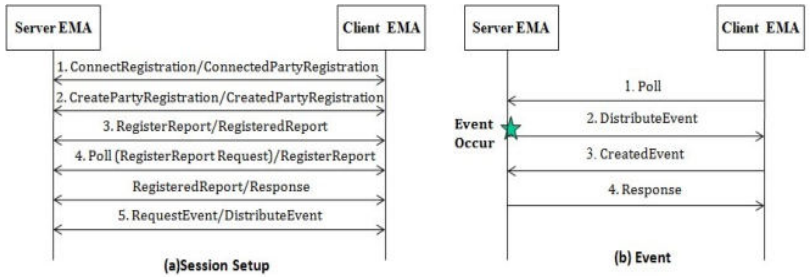
문제점 및 해결책	문제점	해결책
(1)Heavyweight 수요반응 프로토콜의 제한점 -집합 건물 환경을 위한 경량에너지관리에이전트 프로토콜	IoT 환경→저전력이고 소형화 된 네트워크 장비 HTTP 수요반응 프로토콜은 Heavyweight 해서 많은 리소스와 데이터 트래픽이 발생. XML은 많은 프로세싱 오버헤드,파싱 과정이 많음 Pull 메커니즘/네트워크 방화벽 문제	-경량 수요반응 프로토콜 및 JSON 사용 (CoAP, MQTT/JSON) -UpdateReport 실험 →CoAP, MQTT/JSON경량프로토콜 데이터 트래픽, 지연시간 감소
(2)Scalability 한계 및 수요 반응의 문제점 -통합형 에너지관리에이전트프로토콜	수요반응에 대한 효율적 수요 관리 및 분배 필요, 보안성 문제-트래픽 발생 (예 : UpdateReport) 수요반응사업자로 부터 받은 수요반응 이벤트를 주거공간의 에너지관리에이전트가 Control만 지시 IoT 디바이스들에 Control 지시 (SEP2.0도 다이렉트 Control)	서버 에너지관리에이전트의 Aggregator 역할 다단계 에너지관리에이전트 설계로 인한 데이터 트래픽 감소, 병목 현상이 감소
(3)에너지관리에이전트의 Explicit 모니터링 문제점 -Implicit/Explicit 모니터링	모든 디바이스의 정보를 받기 위해서는 응답시간이 오래 걸리고 디바이스를 개수가 증가하게 되면 데이터 트래픽이 늘어 병목현상이 발생 기존 연구는 디바이스 정보가 모두 보내기 때문에 보안성 문제	에너지관리에이전트 프로토콜의 Implicit/Explicit 기반 모니터링 -데이터 트래픽의 단점을 극복 Implicit 방식 : 보안성(프라이버시) 향상 실험을 통해 증명
(4) 주거 공간 환경에서 수요반응 Event 서비스의 제한점 -수요반응 Event 메커니즘을 이용한 에너지관리에이전트 프로토콜	제한적인 IoT환경에서 집합건물 내의 고객의 요구에 맞춘 수요 관리가 필요 (보안성 문제, 네트워크 방화벽문제, 그룹화 정보가 없어 유니캐스트 전송 문제 데이터 트래픽 문제 Topic: Group/#)	경량 프로토콜인 CoAP, MQTT을 사용 (EMAP 프로토콜 설계 및 구현) 경량 수요반응 프로토콜의 다양한 메커니즘 사용(다양한 시나리오 제시) 실험을 통해 증명

■ 논문 상세 내용(집합 건물 환경의 경량 에너지관리에이전트 프로토콜 구현)

구현 내용	그림	구현
(1)경량 수요반응 프로토콜 기반 OpenADR 2.0b와 에너지관리에이전트 프로토콜의 4가지 수요반응 서비스 구현		에너지 관리 에이전트 프로토콜(EMAP)가 다른 수요반응 프로토콜(**OpenADR2.0b SEP 2.0 의 데이터모델링**을 따랐기 때문에 **수요반응 프로토콜과 상호운용성**이 있음 OpenADR 2.0b의 수요반응 명령어들과 Smart Energy Profile 2.0 (SEP2.0)의 과금, 제어 및 상태 정보에 대한 데이터모델링을 사용
(2) 서버 에너지관리에이전트(Aggregator)의 기능적 중계 기능		1.서버 에너지관리에이전트는 에너지관리시스템으로부터 **수요반응 이벤트를 중계**해서 클라이언트 에너지관리에이전트에게 바로 전달 2. **서버 에너지관리에이전트의 자체적인 수요반응 스케줄링** 통해 클라이언트 에너지관리에이전트에게 수요반응 이벤트를 내림 (**외부 그리드와 통합적 수요반응 가능**)
(3)Implicit/Explicit 모니터링 통신 기능		-**Implicit 메시지 방식** (1)모니터링에 있어 사용자가 누군지 모르고 디바이스 **제어 같은 의사 결정이 어려움** (2)광범위한 스마트 홈 네트워크 **환경에서 사용자가 얼마나 많을지 모름** (3)**데이터 트래픽 양이 적고 보안성은 높기 때문에 공격자의 침입을 막을 수 있음** -**Explicit 메시지 방식** (1)상위 서버관리에이전트나 에너지관리시스템에서는 **모든 디바이스 데이터가 있음** - 통계적 분석과 빅데이터 분석을 할 수 있는 장점이 있음.

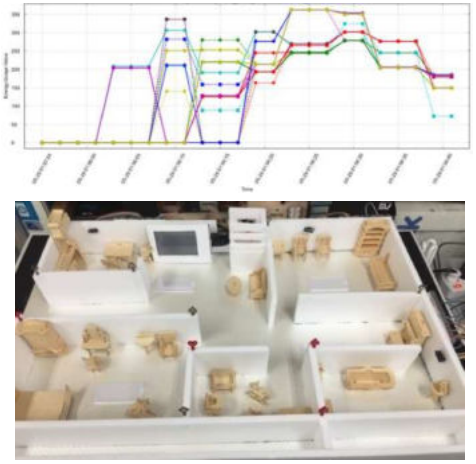
■ 논문 상세 내용(연구 배경 및 실험 환경)





VTN(EMS), Server EMA

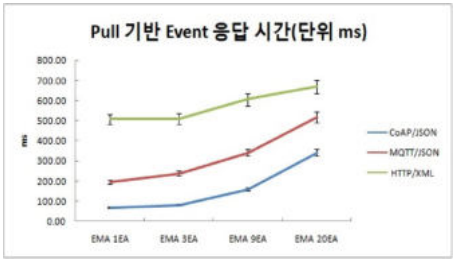
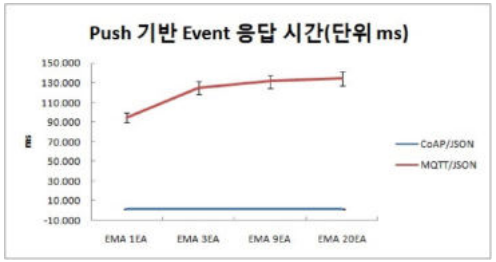
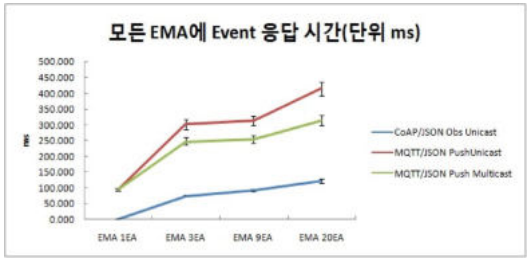
Client EMA, OpenWRT



Smart Home(Appliances)

주요 개발 언어 : **C/ C++, JAVA, ruby in rails**
개발 환경 : 리눅스, **OpenWRT,** 라즈베리파이, 아두이노

■ 논문 상세 내용(실험 및 구현 결과(대표적 실험만))

실험 내용	그래프
집합 건물 내부 이벤트 수요반응 전송 메커니즘 비교 실험	  

실험 시나리오 및 결과

EMS/ Server EMA/ VTN	Host PC	CPU	Intel Core i7-5600U
		CPU Speed	2.6GHz
		RAM	1GB
		OS	Ubuntu 14.04 LTS 32bit
		Language	JAVA, Jruby
EMA/ Client EMA/ VEN	Gateway	Model	Buffalo WZR-HP-G300NH
		CPU	Atheros AR9132 rev 2 (0xb9)
		CPU Speed	400 MHz
		RAM	64 MB
		OS	OpenWRT 15.05 Chaos Calmer
		Language	C, C++
MQTT/ CoAP	Device	Model	RaspberryPi3
		CPU	ARM Cortex-A53
		CPU Speed	1.2GHz
		RAM	1GB
		Language	C

실험 시나리오	CoAP/JSON	MQTT/JSON	실험 결과	결론
1. 평소 Pull 기반 수요반응 서비스 트래 픽 비교 (UpdateReport 실험)	Pull 기반 CoAP/JSON	Pull 기반 MQTT/JSON	(SEMA에서 트래픽측정) CoAP와 MQTT와 트래픽 양은 비슷했음	리소스가 제한적인 집합건물에 경량 수요반응 프로토콜 MQTT/JSON, CoAP/JSON 이 필요
2. 수요반응 Event 을 한대에 내릴 때(Pull)	Pull 기반 CoAP/JSON	Pull 기반 CoAP/JSON	Event 응답 시간은 HTTP/XML<MQTT/JSON <CoAP/JSON 순서로 빠름 (CoAP가 MQTT보다 1.5배 빠름)	리소스가 제한적인 집합건물에 경량 수요반응 프로토콜 MQTT/JSON, CoAP/JSON 이 필요
3. 수요반응 Event 을 한대에 내릴 때(Push)	CoAP Observe	Broker 기반 MQTT Push	20대 일때 MQTT Push Un 보다 CoAP Observe(Push) 응답속도가 81배 빠름	집단 건물 내의 많은 EMA가 있고 긴급한 실시간 수요반응 프로토콜은 CoAP/JSON이 적합.
4.수요반응 Event 을 모든 EMA에 내릴때	CoAP Observe 유니캐스트	Broker 기반 MQTT Push 유니캐스트 / Broker 기반 MQTT Push 멀티캐스트 메커니즘	20대 일때 MQTT Push Unicast보다 CoAP Observe(Push) Unicast가 응답속도가 3.4배 빠름 20대 일때 MQTT/JSON Push Muticast보다 CoAP Observe(Push) Unicast가 응답속도가 2.5배 빠름	-집단 건물 내의 많은 CEMA가 있고 여러 개의 중복 메시지를 전달 할 때 CoAP/JSON이 적합. (하지만, 집합 건물의 SEMA가 부하가 많을 때 MQTT Broker 기반의 MQTT/JSON Push Multicast을 쓰면 보내는 쪽에서 1/N(CEMA)로 트래픽이 적음)

■ 논문 상세 내용(논문의 결론 및 향후 과제/데모)

- 본 논문의 결론

(1)EMA을 통해 집합 건물 환경에서도 지능화된 에너지관리인 수요반응을 함
(2)에너지 관리 에이전트 프로토콜을 통해 수많은 집합건물의 수요관리 고객에 대한 **실시간 수요반응을 할 수 있음(Push 실험을 통해 증명)**
(3)집합 건물 환 경에서 **에너지관리에이전트 간 통신을 통해 수요반응 및 통합적 스케줄링을 할 수 있음**
(4)집합 건물 환경 에서 EMA 프로토콜을 경량 IoT 프로토콜인 CoAP, MQTT 이용
-Heavyweight인 HTTP/XML 방식의 수요반응 프로토콜은 제한이 있음(OpenADR2.0b, SEP2.0)
-디바이스나 IoT기기, Gateway들에는 제한된 리소스 및 배터리 용량이 제한, IoT(CoAP, MQTT) 프로토콜이 lightweight, 저전력 기반
(5)제한적인 IoT환경에서 데이터 트래픽, 수요반응 전송 속도를 줄이기 위해서 IoT 프로토콜의 메커니즘 비교를 통해
적합한 프로토콜인 CoAP/JSON 이용- (이유 : 집합건물의 수많은 개인지역으로 확대(수많은 노드) , 효율적 수요반응 관리가 필요)
-1. CoAP, MQTT 기반 lightweight, 2. CoAP Observe, 3. MQTT Broker 기반 Push 메커니즘, 4. MQTT 기반 멀티캐스트
-CoAP 가 실시간 수요반응의 효율성이 있음을 증명
(6)기존연구의 Explicit모니터링과 EMAP의 Implicit/Explicit 모니터링 비교를 통해 데이터 추상화 기능 의 필요성을 증명

-본 논문의 향후 과제

-에너지관리에이전트 프로토콜은 스마트 홈과 스마트 공장 등에서 **스마트 시티**로 확대 적용
(스마트시티 기반의 에너지관리에이전트를 분산전원 및 Advanced Metering Infrastructure(AMI)과 결합 하는 추가적 연구가 필요)
-다양한 IoT 기술 및 플랫폼과 결합하여 IoT 기기들과 MQTT, CoAP을 이용하여 통합 **서비스 프레임워크를 구축**
-4차 산업 혁명이 등장함에 따라 집합 건물 내의 에너지관리에이전트는 **홈 게이트웨이나**
스마트 IoT 디바이스(인공지능 스피커, 공유기 등)와 새롭게 결합하고 서로 간 서비스 결합이 활발히 발생

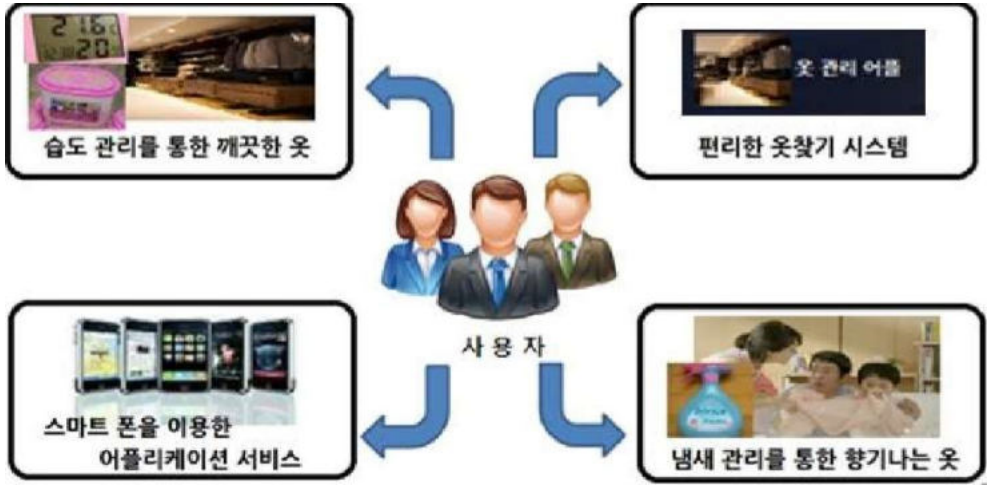
-DEMO

https://www.youtube.com/watch?v=6_pg-Md-EKk

목차

■ 캡스톤 디자인 프로젝트(2013년도), 3명 진행

옷장 관리 시스템 설계 및 구현(스마트 옷장)



(1)개발 배경 : 현대인들의 바쁜 생활로 부가적인 자기 관리를 소홀하게 됨

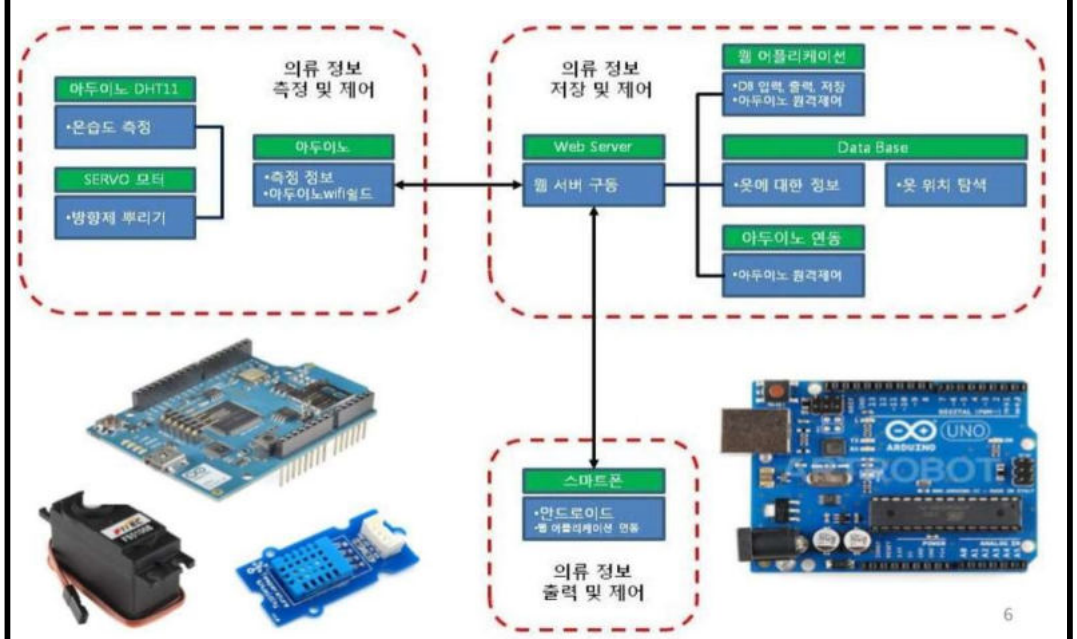
현대인들은 자신을 꾸미는 욕구가 높아짐 – **의류에 대한 관심이 많아 짐**

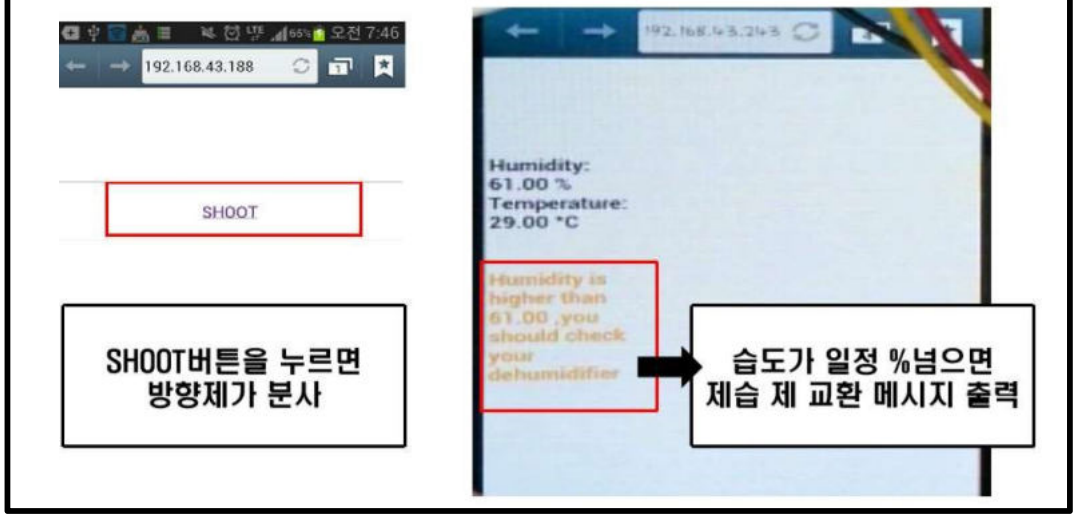
하지만, 의류에 대한 관리 소홀, 버리는 옷이 많아 짐, 한해 버려지는 옷이 **6만 톤**이 넘음

(2)스마트 홈 IoT 기술의 발전- 옷장에 대한 발전이 필요함(융합)

모바일 어플리케이션 개발 부상, 옷장에 관련된 스마트 홈 사업이 없음, 패션 의류 IT 분야가 뜨고 있음

■ 스마트 옷장 - 개발 상세 내용 / 담당: 안드로이드 앱 개발





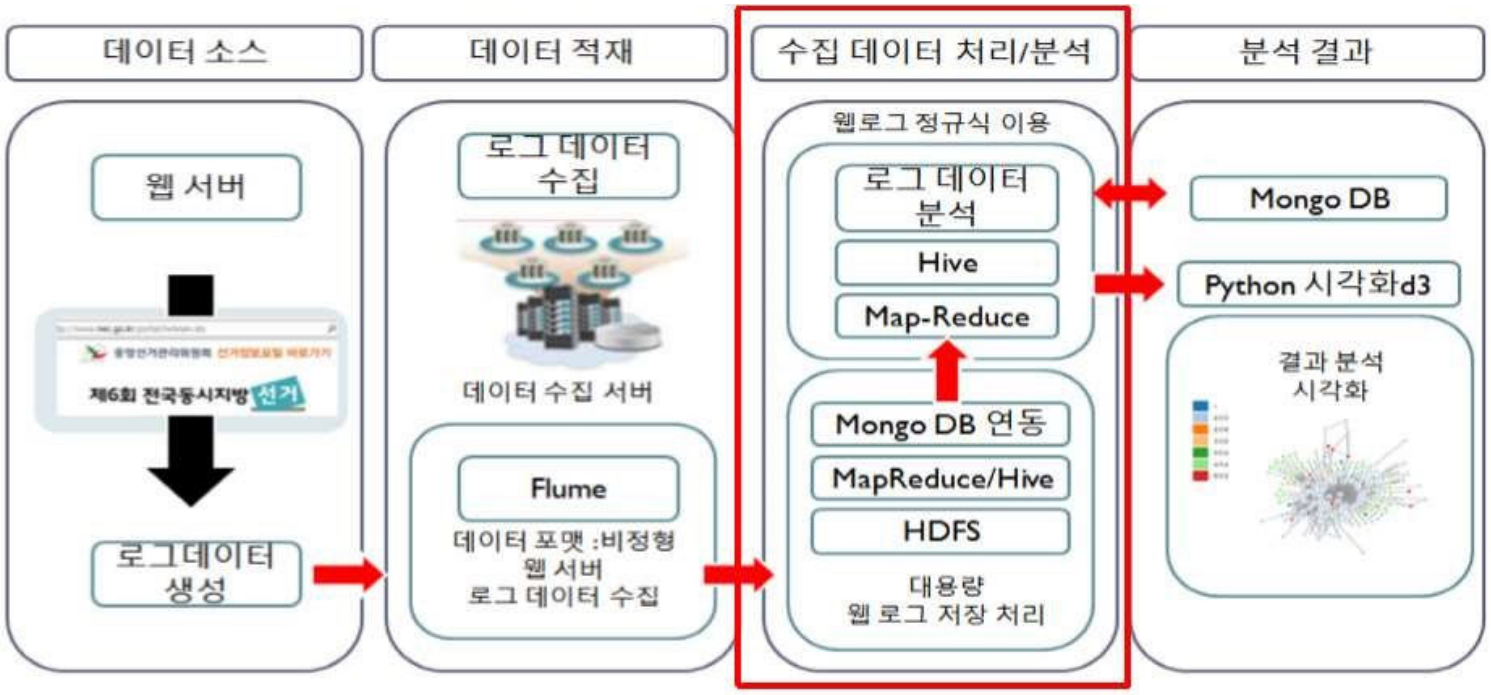
주요 개발 언어 : **C/ C++, JAVA, SQLite**(데이터베이스)
개발 환경 : 안드로이드, 리눅스, 아두이노(프로토타입으로 구현)

■ 학사 빅데이터 논문 프로젝트(2013~2014년도), 혼자 진행

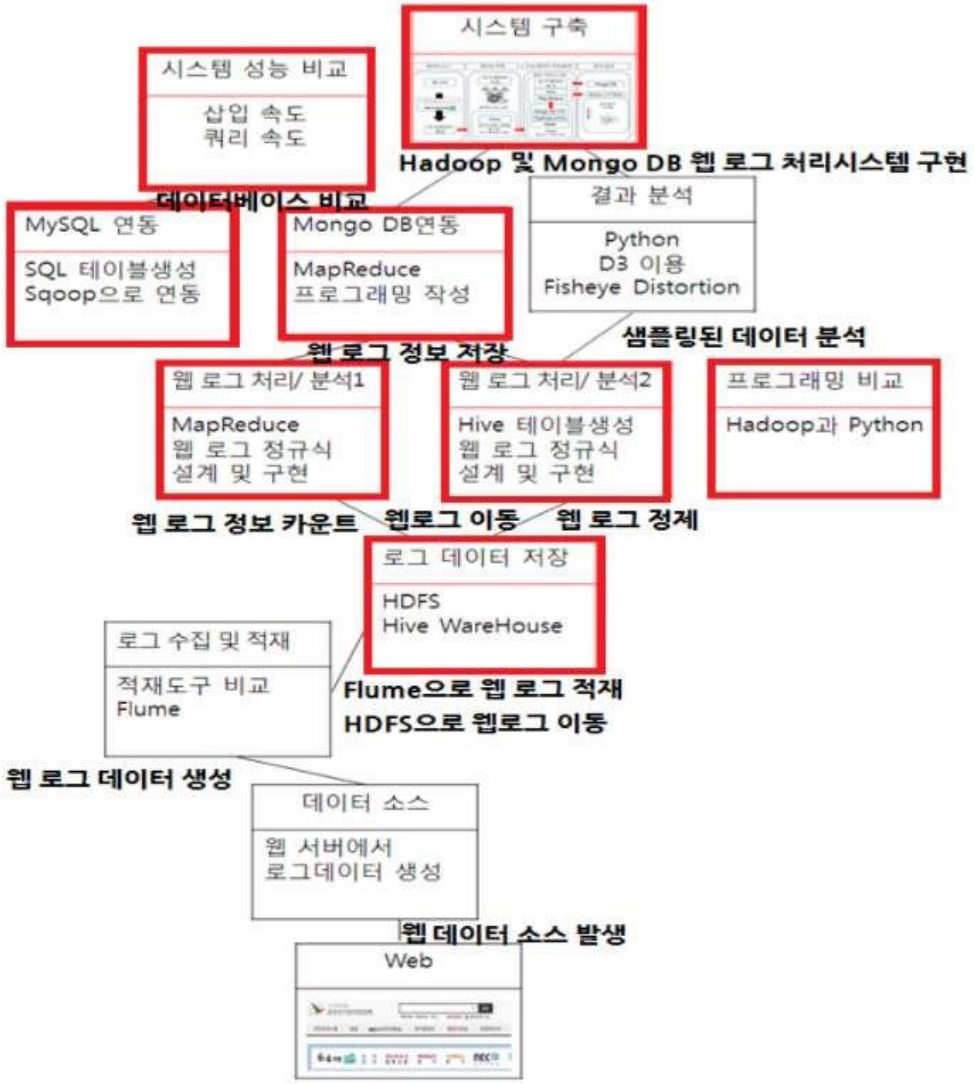
Hadoop 및 Mongo DB 기반 대용량 로그 처리 시스템 설계 및 구현

(1)개발 배경 : 대량으로 들어오는 데이터에 대한 적재, 처리가 기존 관계형 데이터베이스 시스템(RDBMS) 기반 로그 시스템에서 Hadoop 기반 빅데이터 대용량 분산 처리 플랫폼 시스템으로 넘어가고 있음
-따라서, 데이터 수집 처리하는 파이프 라인을 설계해 제안해서 시스템을 구축

(2)연구 내용 : 학과 웹 사이트의 웹 로그 처리/분석 중심적 시스템 설계 및 구현
-Hadoop, Hive, MongoDB 기반 시스템 성능 평가



■ 빅데이터 논문 개발 상세 내용

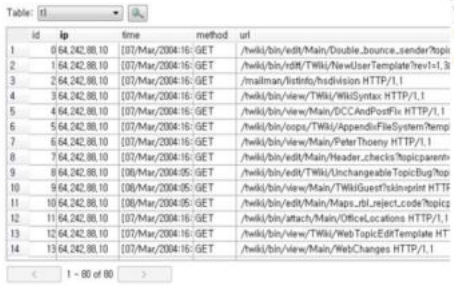


주요 개발 언어 : **JAVA, Python, SQL, nosql**
개발 환경 : 리눅스**, Hadoop echo system**

■ 빅데이터 논문 개발 상세 내용

다른 프로그래밍 (Python) - 로그 파일 입출력 형식

- 장점 : 스크립트 복잡성이 낮음
- 단점 : 대용량 측면에서 속도, 쿼리속도 데이터베이스에 삽입속도가 느림
  연계된 로그 적재 도구가 거의 없고 분석 관련 도구 지원이 미흡
  일괄적으로 작업을 하는 번거로움



```
*Python 3.4.0: access.py -
File  Edit  Format  Run  Options  Windows  Help
encoding = sys.stdout.encoding
logfile = open('access_log.txt','r',encoding="utf8")
for line in logfile:
        sys.stdout.buffer.raw.write(line.encode(encoding, 'ignore'))
for source in logfile.readlines():

        ip = re.search('^(.+) ', source)
        print ("IP : ", ip.group(0))

        remo = re.search('(\\S+) (\\S+)', source)
        print ("Info : ", remo.group(0))

        time = re.search('\\[([\\w:/]+\\s[+\\-]\\d{4})\\]', source)
        print ("Access Time : ", time.group(0))

        method = re.search('\"([^\\s]+)', source)
        print ("Method : ", method.group(0))

        page = re.search('(\s/.+\HTTP\/\d\.\d)', source)
        print ("Page : ", url.group(0))

        code = re.search('([^\\s]+) (\\s*[^\"]*)\"', source)
        print ("code : ", code.group(0))

        byte =re.search('(\\d{3}) (\\d+)', source)
        print ("Size : ", byte.group(0))
```

Map-Reduce를 이용한 웹 로그 정보 분할 및 카운트 방법

웹 로그 포맷을 파싱하기 위해 정규식 표현 식으로 패턴을 구현

```
"^(.+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"([^\\s]+)
 (1) client Ip 또는 도메인 (2)-  (3) -        (4)accessTime        (5)requestMethod
([^\\s]+)(\\s*[^\"]*)\" (\\d{3}) (\\d+) ";
 (6)requsetPage          (7)responsecode (8)bytesSent
```

```
yanafal.tele.nokia.fi - - [01/Jun/1995:00:26:31 -0600] "GET /~scottp/publish.html" 200 271
130.54.25.198 - - [01/Jun/1995:00:28:36 -0600] "GET /~macphed/finite/fe_resources/node1.html" 200 9651
```

^(.+)분석 : ^ :줄의 시작 , .+ . 1회 이상 반복 ( )는 group를 지칭 , group(1) : client Ip

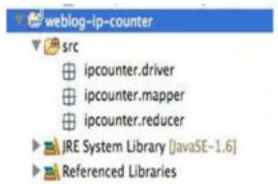
yanafal.tele.nokia.fi 130.54.25.198

```
if ( matcher.matches() ) {
        clientIp = matcher.group(1);
        accessTime = matcher.group(4);
        requestMethod = matcher.group(5);
        requestPage = matcher.group(6);
        responseCode = matcher.group(7);
        bytesSent = matcher.group(8);
```

결론 : ip 뿐만 아니라 웹로그에서 다른 정보도 group으로 묶기 때문에 카운트 할 수 있음

Map-Reduce를 이용한 웹 로그 IP카운트 및 분할

1.Eclipse로 Java 작업 후 Jar파일 생성 2. HDFS 웹 로그 저장



3.MapReduce 웹 로그 IP Count 프로그램 실행

```
heonil89@ubun:/opt/hadoop-1.0.3$ ./bin/hadoop jar weblog-ip-loader.jar
ipcounter.driver.IpCounterDriver /sample UofS_access_log/result/out1
14/05/06 23:05:10 INFO input.FileInputFormat: Total input paths to process : 1
14/05/06 23:05:10 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/05/06 23:05:10 WARN snappy.LoadSnappy: Snappy native library not loaded
14/05/06 23:05:11 INFO mapred.JobClient: Running job: job_201405062144_0019
14/05/06 23:05:12 INFO mapred.JobClient:  map 0% reduce 0%
14/05/06 23:05:41 INFO mapred.JobClient:  map 2% reduce 0%
14/05/06 23:05:45 INFO mapred.JobClient:  map 5% reduce 0%
14/05/06 23:05:48 INFO mapred.JobClient:  map 15% reduce 0%
14/05/06 23:05:51 INFO mapred.JobClient:  map 21% reduce 0%
```

Hive 테이블 데이터 생성 및 HDFS안에 Hive warehouse 생성

```
hive> CREATE TABLE weblog(
    > client_ip string,
    > access_time string,
    > request_method string,
    > request_page string,
    > response_code string,
    > bytes_sent string)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > ;
OK
Time taken: 0.41 seconds
```

Contents of directory /user/hive/warehouse

Goto : /user/hive/warehouse go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
weblog	dir				2014-05-06 23:17	rwxr-xr-x	heonil89	supergroup

Go back to DFS home

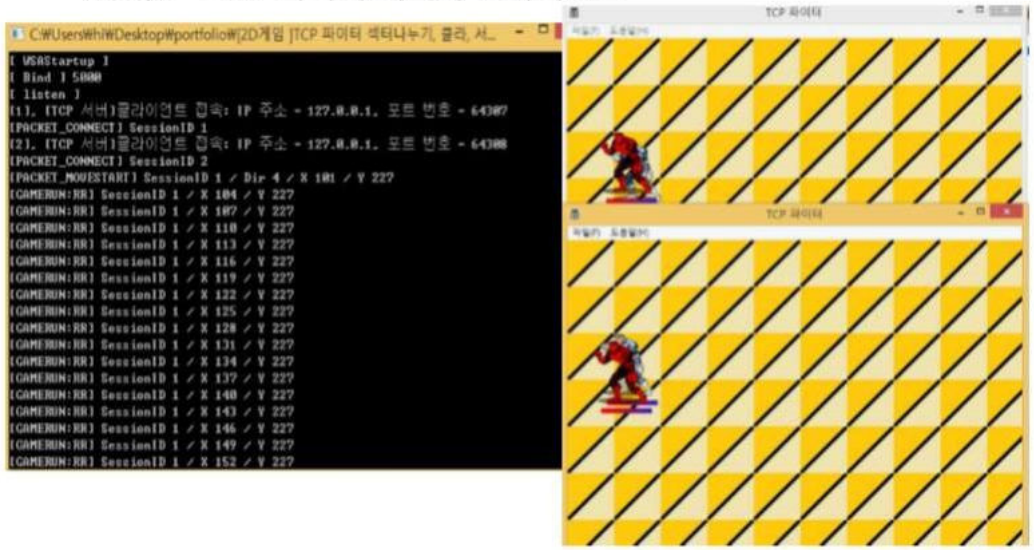
주요 개발 언어 : **JAVA, Python, SQL, nosql**
개발 환경 : 리눅스, **Hadoop echo system**

■ 대용량 게임 서버 프로젝트(2014~2015년도, 프로카데미)

대용량 처리 서버 설계 및 개발/ 모니터링 툴 개발

수강 배경 : 자료구조와 네트워크 프로그래밍에 대한 실습을 하고 싶어 프로카데미에 들어가서 게임 서버 개발 과정을 1년 동안 학습을 하였습니다. 직접 대용량 서버를 설계하고 개발하면서 인프라와 개발 과정에 대해서 많은 학습을 하였습니다.

2D게임 - TCP 파이터 서버와 클라이언트



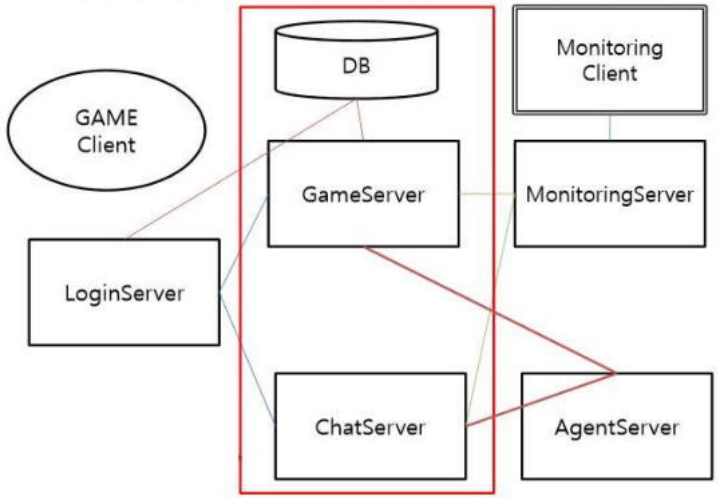
유니티 액션 게임 게임서버 , 채팅서버 IOCP 이용,DB

■ 게임 개발 상세 내용

유니티 액션 게임 게임서버 , 채팅서버 IOCP 이용,DB

MySQL DB 구조와 모니터링 클라이언트

유니티 액션 게임 게임서버 , 채팅서버 IOCP 이용,DB



주요 개발 언어 : **C/ C++, WIN API, C#, PHP, MySQL**
개발 환경 : 윈도우 서버, 유니티