OpenWRT 기반의 무선 네트워크 통합 관제 시스템

부산대학교 정보컴퓨터공학부

지도교수: 김태운

팀 명: 와신상담

팀 원: 201924657 장원석

202155508 권내현

202155547 남원정

〈목차〉

0. 용어 정의

0.1 목적

프로젝트와 관련하여 사용되는 주요 용어들에 대한 정의이다. 팀원들 간의 의사소통 및 보고서의 이해를 돕기 위해 각 용어의 의미를 간결하게 설명하여 제시한다.

0.2 정의

용어	정의
사용자	해당 서비스를 이용하는 모든 개인
관리자	최종 관리자와 일반 관리자를 통칭
최종 관리자	회사 내 라우터 및 일반 관리자를 관리하는 사용자
일반 관리자	회사 내 라우터를 관리하는 사용자
일반 사용자	핸드오프 서비스를 이용하여 최적의 와이파이에 연결하는 사용자
공유기	여러 기기가 유선,무선으로 인터넷에 접속할 수 있도록 인터넷 환경을 관리하는 기기
프로파일	'변경 조건'과 '변경 설정'으로 구성된 라우터의 사전 설정값
one-click 스크립트	제시하는 시스템에 신규 라우터를 손쉽게 등록하고 관리하기 위한 라우터 초기 설정용 쉘 스크립트
인증 서버	사용자 인증 및 권한 관리를 담당하는 스프링 서버
관리 서버	라우터 제어 및 모니터링, 프로파일 관리를 담당하는 스프링 서버
Handover (Hand-off)	네트워크 사용자가 이동할 때 통신의 질을 보장하기 위해 사용자가 연결되어 있는 라우터를 변경하는 것
메트릭	모니터링을 통해 알 수 있는 라우터의 상태, 네트워크 속도 등 라우터와 관련된 수치
트래픽	네트워크를 통해 전송 및 수신되는 데이터의 양

용어	정의
사용자	해당 서비스를 이용하는 모든 개인
관리자	최종 관리자와 일반 관리자를 통칭
최종 관리자	회사 내 라우터 및 일반 관리자를 관리하는 사용자
OpenWrt	무선랜 라우터에 설치하여 기기를 제어하는 데에 사용할 수 있는 리눅스 기반 경량형 운영체제
백오피스	기업 내 최종관리자가 일반 관리자에 대한 회원가입을 하는 등 보안 강화를 위한 기업 내부 시스템

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 과제 배경 및 필요성

1.1.1 다중 공유기 제어의 어려움

대학, 회사 등 규모가 큰 시설은 여러 대의 공유기를 제어 및 관리하는 서비스가 필요하다. 공유기를 제어하는 방식 중 하나로, 공유기에 리눅스 기반 운영체제를 통해 공유기를 제어할 수 있도록 만들어진 OpenWrt를 설치한 후 웹 GUI인 Luci를 이용하여 공유기를 조작하는 방식을 채택할 수 있다. 해당 방식은 안정적으로 라우터에 대한 전반적인 제어를 할 수 있지만, Luci를 통한 공유기 조작은 단일 공유기에 대해서만 유효하기 때문에 여러 대의 공유기 설정을 일괄적으로 변경하고 싶은 경우에는 설정을 일일이 처리해야 하므로 작업의 효율이 떨어진다.

1.1.2 최적의 네트워크 설정 및 연결 어려움

기존의 스마트폰, 태블릿, 노트북 등의 기기들은 사용자가 가장 안정적이고 빠른 연결을 선택할 수 있도록 와이파이의 목록을 신호 세기 순으로 정렬하여 표시한다. 일반적으로 신호 세기가 강할수록 네트워크의 품질이 좋고 연결 속도도 빠를 가능성이 높기 때문이다. 그러나 신호 세기가 강한 공유기라 하더라도 많은 사용자가 접속해 있을 경우, 병목 현상이 발생하여 원활한 통신을 하지 못하는 일이 잦다. 특히 축제 등 사용자가 밀집된 공간에서 많은 사람들이 하나의 라우터에

연결되어 있는 경우, 연결되어 있는 모든 사용자가 네트워크를 사용할 수 없게 되기 때문에 불편함을 겪는다.

1.2 과제 목표 및 기대효과

본 과제는 OpenWRT를 이용하여 다중 공유기를 모니터링 및 제어할 수 있는 시스템을 개발하는 것을 목표로 한다. 네트워크 관리자는 앱에 로그인하면 다중 공유기 모니터링 및 제어 서비스를 통해 공유기 모니터링 및 관리의 효율을 올릴 수 있다. 또한, 다양한 상황에 최적화된 동작을 수행할 수 있도록 미리 정의해둔 공유기 설정 프로파일을 사용함으로써 병목현상을 해결하여 네트워크 연결 품질을 개선하거나 최대한 많은 사용자에게 일정한 네트워크 연결 품질을 제공하도록 할 수 있다.

또한, 신호 세기만 고려하던 기존 시스템의 연결 가능 공유기 리스트와 달리, 현재 연결할 수 있는 공유기 중에서 신호 세기, 작업량, 트래픽 등의 다양한 메트릭을 고려하여 최적의 공유기에 접속하는 알고리즘을 개발하여 사용자 측면에서 네트워크 연결의 질을 높인다.

1.3 요구조건

1.3.1 사용자 인증

- 사용자의 아이디, 이메일, 비밀번호, 역할 등 정보를 저장 및 관리한다.
- 사용자 역할(최종 관리자/일반 관리자/일반 사용자)에 따라 접근 권한을 부여하여 보안성을 높인다.
- 최종 관리자는 한 회사 당 한 명만 등록할 수 있으며 일반 관리자의 회원가입을 승인한다.
- 일반 관리자는 최종 관리자에 의해 승인된 경우에만 서비스를 이용한
- 관리자는 앱 실행시 4자리 잠금 패스워드를 입력해야 한다.

1.3.2 다중 공유기 모니터링

- 공유기에서 자체적으로 수집한 메트릭 정보를 관리자가 모니터링한다.
- 관리자는 각각의 공유기를 따로 모니터링하거나 여러 대의 공유기를 한 번에 모니터링한다.

- 공유기에서 비정상적 연결이나 예기치 못한 문제가 감지되었을 때는 이를 모니터링 중이던 서버가 관리자에게 알림을 송신한다.
- 관리자는 공유기의 접속자 내역이나 실행되는 프로세스에 대한 로그 파일을 요청한다.

1.3.3 프로파일 설정을 통한 다중 공유기 제어

- 관리자는 공유기의 특정 프로파일을 적용시켜 공유기가 현재 환경에 최적화된 방식으로 동작하도록 설정한다.
- 관리자가 수동으로 설정하지 않아도 공유기의 모니터링 결과에 따라 자동으로 공유기 프로파일이 변경되는 설정을 선택한다.
- 신규 공유기의 쉬운 인프라 설치를 위한 one-click 스크립트를 제공한다.

1.3.4 안드로이드 어플리케이션 제공

- 관리자는 다중 라우터 모니터링(1.3.2) 및 제어 서비스(1.3.3)를 이용한다.
- 관리자는 회사 정보 및 라우터의 MAC 주소, 이름, 위치 정보 등을 등록하고, 이후 회사 내 모든 라우터의 상태 및 위치를 확인한다. 위치는 건물 도면 이미지 내 마커로 표시된다.

1.3.5 Hand-off 기능

- 일반 사용자는 현재 연결할 수 있는 와이파이 목록을 신호 세기, 트래픽, 혼잡도 등 다양한 요소를 기반으로 정렬하고, 자동 또는 수동으로 최적의 와이파이에 연결한다.
- 수동 연결 모드일 경우 사용자가 앱 내에서 와이파이에 직접 연결한다.
- 자동 연결 모드일 경우 최적의 와이파이에 자동으로 연결된다. 이는 사용자가 이동하여 와이파이가 변경되었을 때도 마찬가지로 동작한다.

1.4 제약 사항과 대책

1.4.1 제약 사항

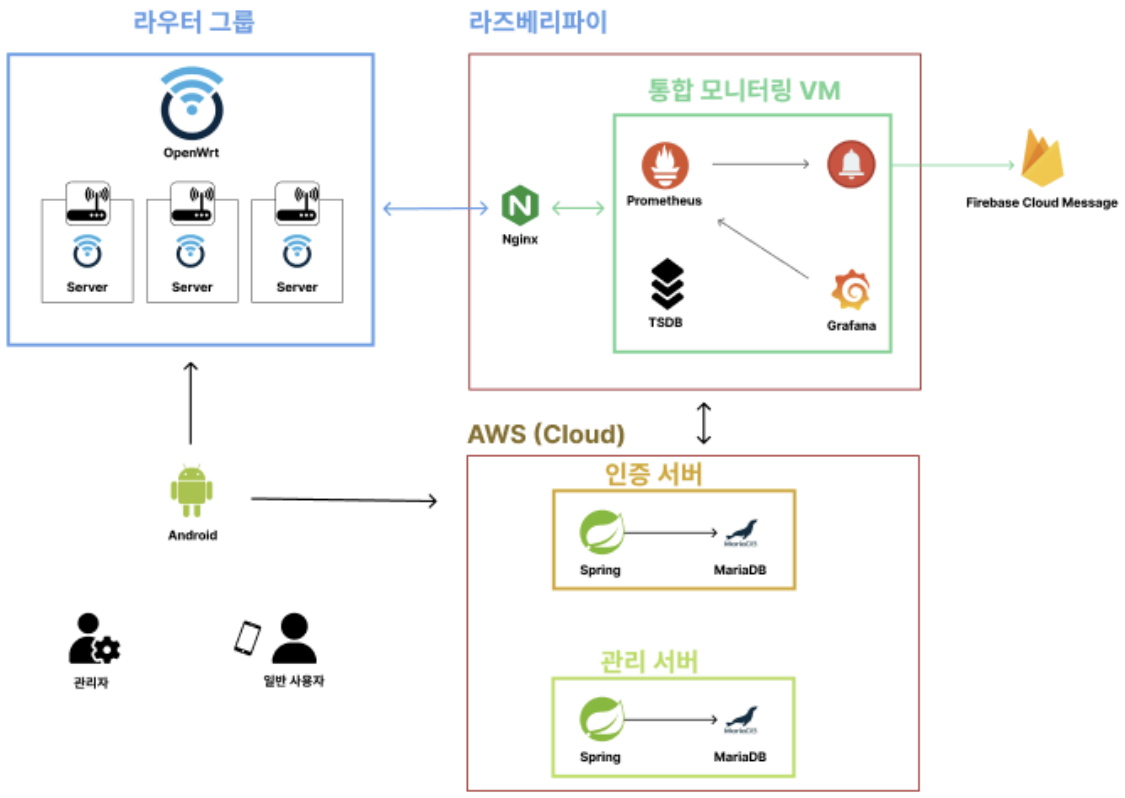
- 각 공유기 별로 메트릭을 수집 및 저장이 발생할 때 동작 주체가 공유기일 경우, 오버헤드 양이 크다.
- 원격으로 공유기에 접속하려면 공유기가 WAN에 연결되어 있어야 한다.
- 공유기가 갑자기 강제종료 되는 경우 문제가 발생했는지 확인하기 어렵다.

- 애플리케이션에서 최적의 Wifi를 찾을 때, OpenWrt가 설치된 공유기 외에도 일반 공유기의 Wifi도 Wifi 리스트에 나타날 수 있다.

1.4.2 대책

- 공유기가 주기적으로 서버에 메트릭 값을 보내주는 방식 대신 서버가 직접 공유기 내부 OpenWrt에 접속하여 메트릭 값을 읽어 오는 방식을 채택한다.
- 주기적으로 공유기의 상태를 점검하여 문제가 발생한 경우 알림을 보낸다.

2. 설계 상세화 및 변경 내역



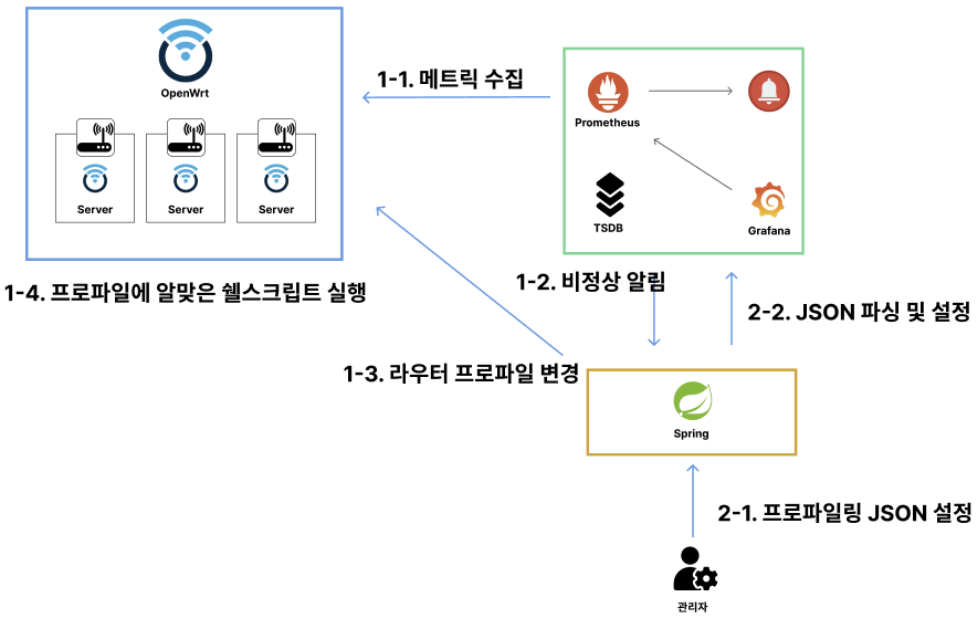
〈 서비스 전체 구조도 〉

2.1 사용자 인증 서버

본 서비스의 이용자 유형은 일반 사용자, 일반 관리자, 최종 관리자로 나뉘며 최종 관리자일수록 더 많은 권한을 가진다. 일반 사용자는 현장에서 라우터 중 하나에 접속하여 네트워크망을 활용하는 주체이고, 일반 관리자는 라우터를 그룹별로 등록하여 각자 관리 정책에 따라 라우터의

상태를 변경할 수 있는 사용자이다. 최종 관리자는 모든 라우터와 관리자를 권한을 조정하고 새로운 관리자의 등록을 승인한다. 인증 서버는 관리(중계) 서버와 구분되어 사용자의 서비스 기능을 제한하고 권한을 인증하는 기능을 수행한다. 이때 인증 서버는 인가 결과 API 접근 권한을 부여하는 Token을 발급한다. 관리(중계) 서버는 라우터 설정을 변경하고, 서비스 기능을 호출하나, 일부 기능에 대해서는 적법한 사용자의 인가를 Token 형태로 검증한다.

2.2 공유기 프로파일 설정



〈 공유기 프로파일 설정 순서도 〉

공유기 프로파일 설정 변경은 수동 변경과 자동 변경으로 나뉜다. 수동 변경은 특정 라우터 또는 라우터 그룹의 상태를 관리자가 직접 사전 정의된 모드 중 하나로 변경하는 것이며, 자동 변경은 모니터링 중인 라우터의 상태값이 특정 조건을 만족하면 모드를 변경하는 것이다. 여기서는 프로파일로 '기본 모드', '사용자 수 제한 모드', '초절전 모드', '연결 보장 모드'를 사용했다. 모드의 변경은 Server-Client 사이 통신을 통해 이루어지기에 JSON 형태로 데이터를 주고 받는다.

모드	설명
기본 모드	공유기의 설정에 별도의 제한을 두지 않는다.
사용자 수 제한 모드	공유기의 네트워크 품질이 낮으면 더이상 사용자가 추가되지 않도록 막는다.

초절전 모드	라우터의 부가 기능을 비활성화하여 전력 소모를 줄인다.
연결 보장 모드	네트워크 속도를 소폭 하락시키는 대신 모든 사용자가 네트워크 연결을 보장받을 수 있도록 한다.

〈 사전설정된 공유기 모드의 예시 〉

Server-Client 사이 통신을 통해 데이터를 주고 받을 때, 프로파일링의 조건과 임계치, 취할 행동을 담는 양식으로 YAML 형식을 사용하고자 하였으나, 관리자가 앱 어플리케이션을 통하여 서버 API를 호출하기에 프로파일링에는 통신에 적합한 JSON 형식을 사용하는 것으로 변경하였다.

2.2.1 자동 모드

자동 모드는 모니터링 서버에서 개별 공유기의 상태를 수집하다가, 특정 조건이 만족되면 관리(중계) 서버에 이를 알린다. 관리(중계) 서버는 알림을 바탕으로 해당 라우터에 취할 행동을 자체 데이터베이스로부터 검색해 결정하고, SSH로 접근하여 모드를 변경한다. 이 과정에는 관리자가 개입하지 않으며, 라우터에 사전 작성된 쉘 스크립트의 호출만으로 모드를 변경하게 된다. 첫 설계에서는 Go언어를 통해 개별 라우터가 모드 변경 API를 제공하도록 계획했으나, 라우터의 컴퓨팅 자원 한계로 인하여 UNIX 명령어만으로 동작하도록 변경하였다.

2.2.2 수동 모드

수동 모드는 관리자가 안드로이드 어플리케이션을 통해 관리(중계) 서버에 직접 지시하여 라우터의 모드를 변경하는 방식이다. 앱은 관리자가 개별 라우터의 상태를 모니터링할 수 있도록 Grafana의 Public Dashboard를 제공하고, 관리자는 이를 토대로, 또는 현장 상황을 토대로 라우터의 모드를 언제든 자유롭게 변경할 수 있다.

2.3. 공유기 상태 계산 및 핸드 오프

2.3.1 공유기 상태 계산

공유기 상태는 네트워크 품질, CPU 사용률, 현재 공유기에 인가된 트래픽의 양 등을 종합적으로로 고려하여 다음과 같은 식을 통해 0~100 사이의 점수로 변환한다.

$$avg(network\ quality)\ *\ (1\ -\ avg(node\ load)\ /\ count(cpu))$$

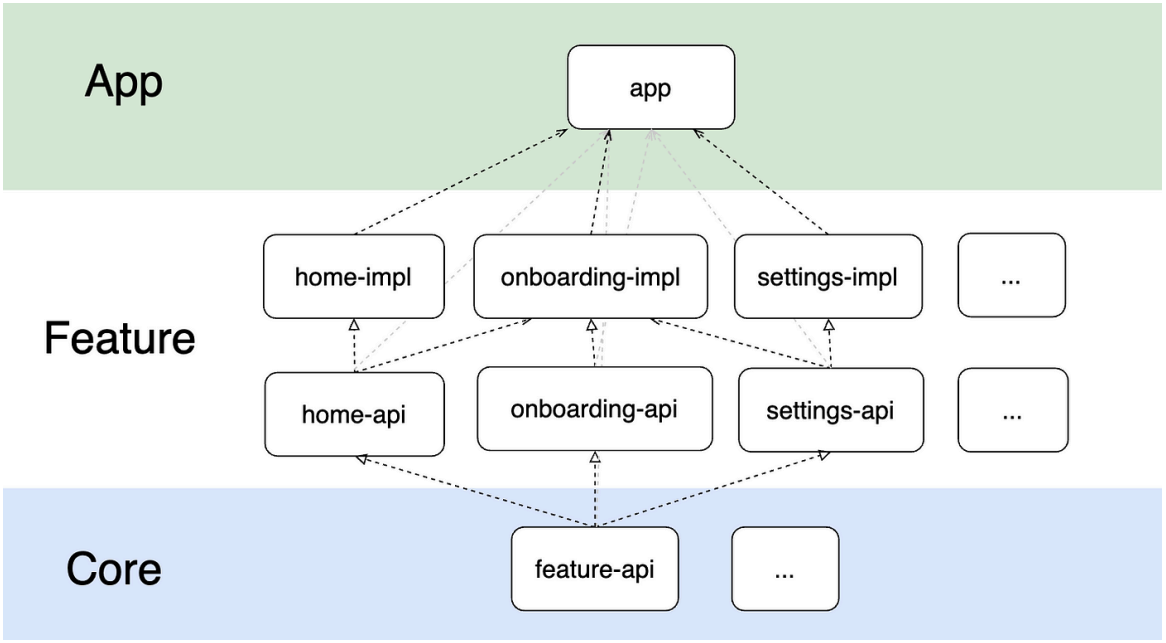
관리자의 공유기 전체 조회 화면에서 라우터의 마커 색상이 아래와 같은 색상으로 매핑되어 보여진다. 또한, 공유기가 고장 등의 이유로 정상적인 작동을 하고 있지 않을 경우, 그 공유기는 검은색으로 표시한다.

2.3.2. 핸드 오프

사용자가 수동 모드로 설정해둔 경우, 직접 연결 가능한 Wifi 리스트를 통해 연결하고자 하는 Wifi를 선택하여 비밀번호를 입력한 뒤, 사용할 수 있다. Wifi 목록은 기본적으로 네트워크 품질이 좋은 순서대로 정렬되며, 혼잡도 순 등으로 정렬할 수 있다. Wifi를 정렬할 때, 현재 시스템에서 관리하고 있지 않아 메트릭 값을 알지 못하는 Wifi의 경우, 연결 가능한 Wifi 목록들의 평균값으로 산정하거나 시스템에서 관리하는 Wifi보다 우선도를 낮게 주도록 한다.

사용자가 자동 모드로 설정해둔 경우, 앱은 백그라운드에서 5분마다 현재 연결 가능하고 비밀번호를 아는 Wifi 목록 중 와이파이 신호 세기와 2.3.1의 식을 이용하여 가장 점수가 높은, 최적의 와이파이에 자동으로 연결된다.

2.4 애플리케이션 아키텍처



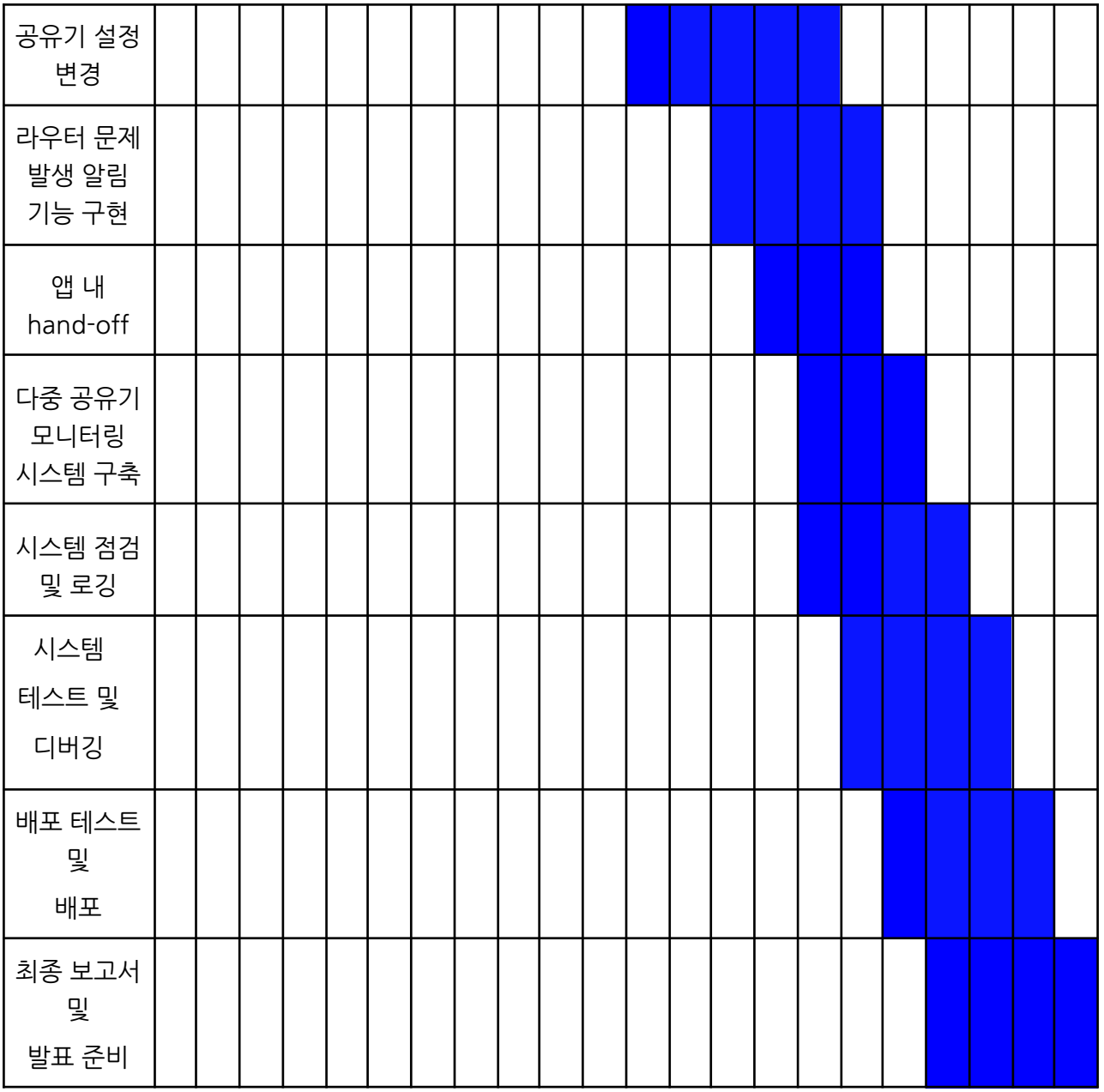
< 안드로이드 멀티 모듈 아키텍처 >

안드로이드 앱은 주로 관리자의 편의를 위해 제공되며, 라우터 그룹 등록/변경과 라우터 모드 변경, 라우터 상태 알림 등의 기능을 제공한다. 앱은 추후 추가될 다양한 기능에 대비하여 ‘Presentation - Domain - Data’ 레이어로 나뉘어진 Clean Architecture를 따르도록 하고 기능 확장을 용이하게 한다.

아울러, 사용자의 분류에 따라 제공하는 화면과 기능이 각기 다르기에 권한 별로 모듈을 분리한 ‘멀티 모듈 아키텍처’를 사용하도록 한다. 이를 통해 수정이 일어난 모듈만 재빌드하여 상시 배포를 용이하게 하고, ‘관심사 분리 원칙’을 만족하도록 모듈을 분리하여 유지보수를 용이하게 한다. 본래 멀티 모듈 아키텍처는 처음 계획한 부분이 아니었으나, 테스트 코드와 CI/CD 구축의 용이함을 위하여 이를 적용하기로 변경하였다.

3. 갱신된 과제 추진 계획

구분	작업 일정																					
	5월			6월				7월				8월					9월				10월	
		4	5	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2
환경 및 분석																						
요구사항 분석																						
시스템 환경 구축																						
공유기 데이터 수집 및 저장																						
사용자 권한 라우터 관리 API																						
UI 개발 및 API 연동																						
상황에 따른																						



4. 구성원별 진척도

팀원	진척도
권내현	**OpenWrt 기초 인프라 구성** • OpenWrt 패키지 이용 방식 수정 • 공유기 ssh 환경 원격접속 허용 • prometheus node exporter를 이용한 메트릭 수집 • prometheus target 설정을 통해 메트릭 전송 • OpenWrt를 이용해 공유기 로그파일 추출

	공유기 프로파일 작성 ● 프로파일별 발동 조건 및 설정값 정의 ● uci를 이용해 프로파일을 적용하는 프로그램 작성 ● DDoS를 대비하여 악성 사용자 패턴 정의 ● 특정 사용자를 연결 해제 시키는 기능 구현
남원정	**애플리케이션 설계** ● Figma를 이용하여 UI 설계 ● ERD를 이용한 데이터베이스 설계 ● API 문서 작성 **인증, 관리 서버 개발** ● 유저 인증, 회사 등록, 백오피스, 공유기, 프로파일 API 구현 ● 서버 테스트 코드 작성 ● 그라파나, 공공데이터 포털 Open API 등 외부 API와의 통신 테스트 **안드로이드 애플리케이션 개발** ● 유저, 회사, 백오피스, 공유기, 프로파일 UI 구현 및 API 연결
장원석	**모니터링 환경 구성** ● 라즈베리파이 배포 환경 구성 ● Prometheus Docker 배포 ● Grafana Docker 배포 ● Nginx Docker 배포 ● Docker Compose 구성 **협업 환경 구성** ● Github Repository, Project 설정

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5.1 OpenWrt 인프라 구성

5.1.1 공유기 원격접속 설정

공유기의 환경에 원격으로 접속하기 위해서는 WAN을 통해 공유기에 연결될 필요가 있다. 그러나 OpenWrt의 방화벽은 기본적으로 WAN → LAN 접속을 모두 차단하고 있기 때문에 초기에는 원격으로 공유기의 OpenWrt 환경에 접속할 수 없다. 공유기의 접속 설정을 위해서는 방화벽에 Accept에 대한 설정을 추가한 후 포트포워딩을 설정하는 과정이 필요하다.

```
config rule
        option src 'wan'
        option proto 'tcp'
        option dest_port '22'
        option target 'ACCEPT'

config rule
        option src 'wan'
        option proto 'tcp'
        option dest_port '80'
        option target 'ACCEPT'
```

ssh를 통해 로컬 네트워크에서 공유기에 접속하여 '/etc/config/firewall' 파일의 마지막 부분에 다음과 같은 구문을 추가했다. 이 구문을 통해 방화벽이 포트 번호가 22 또는 80인 TCP 연결을 허용하도록 제어할 수 있다.

또한, Luci 인터페이스에서 포트포워딩을 사진과 같이 설정해주었다. 이 포트포워딩을 통해 WAN에서 온 연결 중 8080번 포트를 통해 들어온 연결은 192.168.1.1의 80번

포트와 연결되고, 1222번 포트를 통해 들어온 연결은 192.168.1.1의 22번 포트와 연결된다.

포트포워딩 또한 방화벽을 설정하는 것처럼 CLI로 설정할 수 있으나, 서비스를 사용하는 관리자가 Luci와 같은 웹 인터페이스에서 공유기 설정을 변경할 수 있음을 확인하기 위해 포트포워딩 설정은 Luci를 통해 진행되었다.

방화벽과 포트 포워딩을 설정한 결과, OpenWrt를 사용하는 공유기의 공용 IP와 SSH전용 포트 번호를 알고 있으면 SSH를 통해 공유기에 접속할 수 있음을 확인하였다.

5.1.2 공유기 활동 메트릭 수집 및 전송

시스템을 구현하기 위해서는 서버가 공유기의 활동 메트릭을 주기적으로 수집하고, 수집한 메트릭을 기반으로 최적의 와이파이 엑세스 포인트를 찾아야 하기 때문에 공유기의 활동 메트릭을 수집하는 과정이 필수적이다. 해당 기능을 구현하기 위해 착수 보고서에는 '서버에서 SSH를 통해 공유기에 주기적으로 접속하여 원하는 값을 수집한다'고 작성했으나, 자료조사를 하던 도중 'Prometheus는 Node Exporter' 패키지를 발견했다. Prometheus Node Exporter는 하드웨어의 상태와 커널 관련 메트릭을 수집하는 일종의 메트릭 수집기이다.

Prometheus는 Node Exporter의 metrics HTTP endpoint에 접근하여 해당 메트릭을 수집하기 때문에 Prometheus Node Exporter를 이용하면 메트릭을 Prometheus와

연동하여 용이하게 수집할 수 있다. 그렇기 때문에 Prometheus Node Exporter를 이용해 공유기의 메트릭을 수집하도록 본래의 계획을 수정했다.

```
root@OpenWrt_origin:~# opkg install prometheus-node-exporter-lua \
> prometheus-node-exporter-lua-nat_traffic \
> prometheus-node-exporter-lua-netstat \
> prometheus-node-exporter-lua-openwrt \
> prometheus-node-exporter-lua-wifi \
> prometheus-node-exporter-lua-wifi_stations
```

SSH를 통해 OpenWrt 환경에 접속하여 Prometheus Node Exporter 패키지를 설치해주었다.

```
# TYPE node_scrape_collector_duration_seconds gauge
# TYPE node_scrape_collector_success gauge
# TYPE node_nf_conntrack_entries gauge
node_nf_conntrack_entries 191
# TYPE node_nf_conntrack_entries_limit gauge
node_nf_conntrack_entries_limit 31744
node_scrape_collector_duration_seconds{collector="conntrack"} 0.0016410350799561
node_scrape_collector_success{collector="conntrack"} 1
# TYPE node_boot_time_seconds gauge
node_boot_time_seconds 1720325970
# TYPE node_context_switches_total counter
node_context_switches_total 68689457
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="cpu0",mode="user"} 607.33
node_cpu_seconds_total{cpu="cpu0",mode="nice"} 23.33
node_cpu_seconds_total{cpu="cpu0",mode="system"} 916.82
node_cpu_seconds_total{cpu="cpu0",mode="idle"} 180626.56
node_cpu_seconds_total{cpu="cpu0",mode="iowait"} 0.23
node_cpu_seconds_total{cpu="cpu0",mode="irq"} 0
node_cpu_seconds_total{cpu="cpu0",mode="softirq"} 2864.32
node_cpu_seconds_total{cpu="cpu0",mode="steal"} 0
node_cpu_seconds_total{cpu="cpu0",mode="guest"} 0
node_cpu_seconds_total{cpu="cpu0",mode="guest_nice"} 0
node_cpu_seconds_total{cpu="cpu1",mode="user"} 578.38
node_cpu_seconds_total{cpu="cpu1",mode="nice"} 24.03
node_cpu_seconds_total{cpu="cpu1",mode="system"} 969.19
node_cpu_seconds_total{cpu="cpu1",mode="idle"} 182222.62
node_cpu_seconds_total{cpu="cpu1",mode="iowait"} 0.34
node_cpu_seconds_total{cpu="cpu1",mode="irq"} 0
node_cpu_seconds_total{cpu="cpu1",mode="softirq"} 1243.86
node_cpu_seconds_total{cpu="cpu1",mode="steal"} 0
node_cpu_seconds_total{cpu="cpu1",mode="guest"} 0
```

Prometheus Node Exporter를 설치하면 해당 공유기 네트워크에 연결된 상태에서 localhost에 포트번호 9100으로 연결했을 때 사진과 같이 공유기의 메트릭을 수집할 수 있다. 외부에서 이 메트릭에 접속하기 위해서는 외부에서 공유기에 원격으로 접속하는 것을 허용한 것처럼, 방화벽을 설정하고 포트 포워딩을 설정해 주어야 한다. 다음과 같은 스크립트를 작성하여 해당 메트릭에 원격으로 접속하는 것을 허용할 수 있었다.

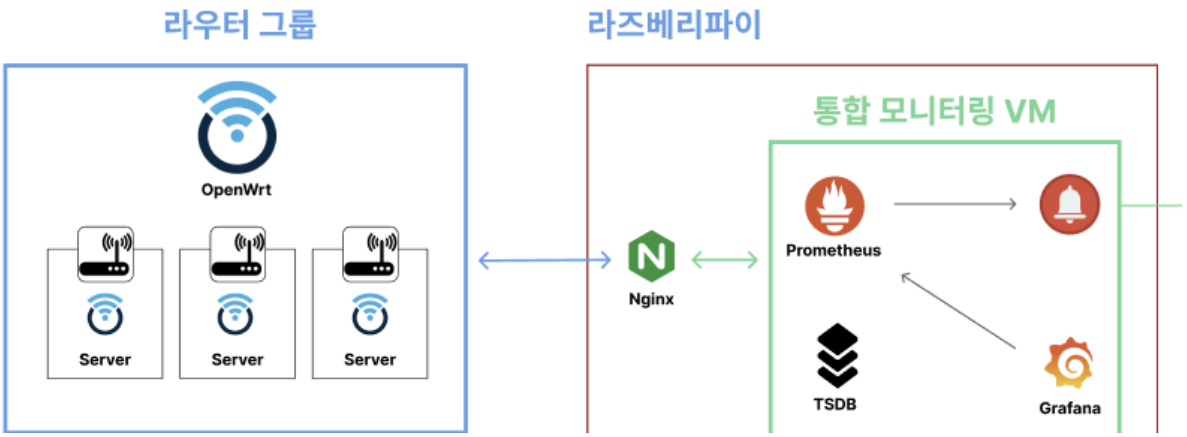
```
config prometheus-node-exporter-lua 'main'
        option listen_interface '*'
        option listen_ipv6 '0'
        option listen_port '9100'
```

```
config rule
        option name 'Allow-prometheus'
        option src 'wan'
        option proto 'tcp'
        option dest_port '9100'
        option target 'ACCEPT'
        option enabled '1'

config redirect
        option dest 'lan'
        option target 'DNAT'
        option name 'M_rule'
        option src 'wan'
        option src_dport '9100'
        option dest_ip '192.168.1.1'
        option dest_port '9100'
```

해당 설정을 적용한 결과, OpenWrt가 설치 공유기의 네트워크에 연결되어 있는 기기가 아닌 외부 네트워크에 연결된 기기도 prometheus의 타겟으로 지정한 링크에 접속하여 메트릭을 확인할 수 있었다.

5.2 모니터링 환경 구성



〈 모니터링 환경 구성도 〉

5.2.1. Prometheus

Prometheus는 개별 라우터에 대해 Pull 방식으로 라우터가 제공하는 메트릭을 주기적으로 수집하고, 자체 DB인 Time Series DB에 이를 저장한다. Docker를 통해 Prometheus 인스턴스를 실행하고, 이는 localhost의 9090 포트로 접근할 수 있다. 이때 설정 파일은 yaml 형식으로 도커 외부에서 제공하고 TSDB의 영속성을 위하여 도커 볼륨을 아래와 같이 마운트한다. 아래는 Docker Compose에 작성한 Prometheus 설정이다.

```
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yaml'
    ports:
      - 9090:9090
    restart: unless-stopped
    volumes:
      - ./prometheus:/etc/prometheus
      - prom_data:/prometheus
    networks:
      - monitoring
volumes:
  prom_data:

networks:
  monitoring:
    driver: bridge
```

5.2.2. Grafana

Grafana는 Prometheus에서 수집한 개별 라우터의 메트릭을 시각적으로 표현하는 툴로, 각종 메트릭을 PromQL을 통해 가공하여 유의미한 도표를 실시간으로 제공한다. Grafana 또한 Docker로 인스턴스를 실행하며, localhost의 3000 포트로 접근할 수 있다. 마찬가지로 설정값을 yaml 형식으로 제공하고, 대시보드의 영속성을 위해 도커 볼륨을 마운트한다. 상세한 설정값은 아래 Docker Compose와 같다.

```yaml
services:
  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - 3000:3000
    restart: unless-stopped
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=grafana
    volumes:
      - ./grafana:/etc/grafana/provisioning/datasources
    networks:
      - monitoring
```

5.2.3. Nginx

Nginx는 라즈베리파이 1대의 배포 환경에서 여러 도커 인스턴스에 대하여 트래픽을 분산하는 역할을 수행한다. 배포 중인 도메인 daily-codidie.com에 대하여, prometheus.daily-cotidie.com 서브도메인은 9090번 포트(Prometheus Docker)로 라우팅하고, grafana.daily-cotidie.com 서브도메인은 3000번 포트(Grafana Docker)로 라우팅한다. Nginx 또한 도커로 실행하며, 서브도메인에 따른 트래픽 분산을 위한 nginx 설정 파일은 아래와 같이 정의된다.

```
http {

  include mime.types;

  set_real_ip_from        0.0.0.0/0;
  real_ip_recursive       on;
  real_ip_header          X-Forward-For;
  limit_req_zone          $binary_remote_addr zone=mylimit:10m rate=10r/s;

  upstream docker-prometheus {
    server prometheus:9090;
  }

  upstream docker-grafana {
    server grafana:3000;
  }

  server {
    listen 80;
    server_name prometheus.daily-cotidie.com;
    root /proxy;
    limit_req zone=mylimit burst=70 nodelay;

    location / {
       proxy_pass http://docker-prometheus;
    }
  }

  server {
    listen 80;
    server_name grafana.daily-cotidie.com;
    root /proxy;
    limit_req zone=mylimit burst=70 nodelay;

    location / {
        proxy_set_header Host $http_host;
        proxy_pass http://docker-grafana;
    }
  }
}

events {}
```

5.3 공유기 프로파일 작성

5.3.1 CLI를 이용한 공유기 프로파일 제어 스크립트 작성

OpenWrt를 이용한 공유기 제어에서 가장 핵심적인 부분은 사전에 정의해둔 프로파일을 이용해 공유기를 현재 환경에 최적화된 방식으로 동작하도록 설정하는 데에 있다. 프로파일을 적용시키기 위해서는 먼저 프로파일이 적용되는 조건, 프로파일을 적용하였을 때의 설정값, 프로파일을 주로 사용하게 되는 상황 등을 정리할 필요가 있었다. 이번 과제 구현을 위해 작성한 프로파일은 총 4가지로, 다음의 표와 같은 특징을 가진다.

프로파일 이름	특징	발동 조건	적용 상황
default	- 네트워크 사용량에 대한 제한 없음 - 네트워크 전송 속도에 대한 제한 없음 - Wifi 사용자 수 제한 없음	- Wifi 사용자가 200명 이하 - 손실되는 패킷의 비율이 15% 이하 - 네트워크 총 전송량이 100MB 이상	특별한 조건이 붙어있지 않은 한 평상시에는 default 프로파일로 작동하도록 설정했다.
user_limit	- 네트워크 사용량에 대한 제한 없음 - 네트워크 전송 속도에 대한 제한 없음 - 최대 Wifi 사용자 수를 250명으로 제한	- Wifi 사용자가 200명 이상인 상태가 3분 이상 지속됨 - 손실되는 패킷의 비율이 15% 이상	Wifi를 사용하는 사람이 많아 네트워크 성능에 영향을 미칠 때, 적어도 일부에게는 최적의 네트워크 성능을 보장하도록 한다.
power_saving	- 네트워크 사용량을 평상시의 1/2로 제한 - 네트워크 전송 속도를 평상시 속도의 1/2로 제한	- Wifi 사용자가 4시간 이상 존재하지 않음 - 단, 자동으로 power_saving 모드가 된 상태에서 와이파이 사용자가 발생하면 일정 시간 이후 default 모드 또는 connective 모드로 돌아감	Wifi를 사용하는 사람이 없는 상태를 가정하여 심야와 같이 인적이 드문 시간대에 Wifi 전력을 최소화시키기 위해 사용한다.
connective	- 많은 사람이 연결된 상태에서도 네트워크 연결의 안정성을 보장 - 네트워크 사용량을 평상시의 1/2로 제한 - 네트워크 전송 속도를 평상시 속도의 1/2로 제한 - Wifi 사용자 수 제한 없음	- 관리자가 수동으로 지정할 수 있는 모드이며, 자동으로 connective 모드로 전환하는 것은 보장하지 않음	최대한 많은 이용자에게 네트워크 '연결을 제공'한다. Wifi에 여러 명의 사용자가 연결된 상태에서도 네트워크 안정성을 보장한다.

이렇게 작성한 프로파일 정보를 기반으로, 공유기의 설정을 제어할 때에는 UCI(Unified Configuration Interface)와 qos-scripts 패키지를 이용했다.

UCI는 가장 중요한 시스템 설정을 위한 OpenWrt의 기본 구성 사용자 인터페이스를 의미하며, 기본 네트워크 인터페이스를 구성하고 무선 네트워크 설정, 로깅 등의 기능을 지원한다. 또한, 변경된 UCI 설정을 init.d 호출을 통해 적용함으로써 공유기를 리셋할

필요 없이 변경 사항을 업타임 동안 적용시킬 수 있다. 공유기 리셋이 필요 없다는 점에서 편의성이 보장되기 때문에 프로파일을 구현할 때 UCI를 이용하였다.

qos-scripts 패키지는 OpenWrt에 설치하여 네트워크 통신 속도, 패킷의 크기, 전송 딜레이 값 등에 제한을 둘 수 있는 패키지로, 일종의 QoS 기능을 제공한다. 이 패키지를 이용하면 네트워크 사용자가 많아 네트워크 품질이 저하되었을 상황을 대비하여 OpenWrt가 설치된 공유기의 성능을 원하는 수치만큼 제한할 수 있기 때문에 프로파일 구현에 qos-scripts를 이용하였다. UCI와 QoS를 통해 작성한 코드의 기본 형식은은 다음과 같다.

```
#
# GLOBAL_VAR_FILE : present state of router's mode file
# LOGGING_FILE : location of log_file
GLOBAL_VAR_FILE="/root/test_excute/present_profile"
LOGGING_FILE="/root/log/log_profile"

# now router's profile mode
current_mode=$(cat "$GLOBAL_VAR_FILE")

# check router's mode if profile is default
if [ $current_mode -eq 1 ]; then
    echo "This router's mode is already [profile_default]."
        exit 1
fi

# start changing profile
echo "Start to change profile...(loading)"
echo 1 > "$GLOBAL_VAR_FILE"

# Part of wifi user limit setting
uci set wireless.default_radio0.maxassoc=3
uci commit wireless
/etc/init.d/network restart

# Part of QOS, Internet speed setting
uci set qos.wan.enabled=0
uci commit qos
/etc/init.d/qos start

echo "[$(date "+%Y-%m-%d %H:%M:%S")] Profile changed to default mode" >> "$LOGGING_FILE"
echo "Finished to change profile...(loading)"
```

이것은 default 모드로 전환할 때 사용하는 프로그램의 소스코드 이미지이다. 크게 네 부분으로 나눌 수 있는데, 각 부분은 다음과 같은 일을 한다.

1. GLOBAL_VAR_FILE은 현재 공유기가 채택 중인 프로파일 값을 저장한 파일의 경로를 나타낸다. 예기치 못하게 공유기가 종료되었을 경우를 대비하여 SSH를 통해 접속해서 전역변수를 통해 확인하는 방법보다 따로 설정 파일을 만들어 두는 편이 더욱 안정적이라고 판단하였다. LOGGING_FILE은 공유기 프로파일 변경 내역을 저장하는

파일의 경로를 나타내는 변수로, 해당 파일에는 사진의 맨 아래와 같은 형식으로 프로파일 변경 내역이 저장된다.

2. current_mode 변수에 현재 공유기의 프로파일 값을 읽어온다. 이후, 만약 변경하려는 프로파일이 현재 상태와 같다면 변경하지 않고 프로그램을 종료시키도록 설계했다. 변경하려는 프로파일이 현재 상태와 다르다면 GLOBAL_VAR_FILE에 저장되어 있는 현재 상태를 새로 변경할 상태로 갱신한다.

3. UCI와 qos-script 패키지를 이용해 공유기의 설정을 제어한다. default는 네트워크 사용량에 대한 제한이 없고 네트워크 전송 속도에 대한 제한이 없으므로 QoS를 이용한 네트워크 제한을 사용하지 않도록 설정했다. 앞서 설명한 것처럼, UCI를 이용하면 공유기의 전원을 리셋하지 않고도 공유기의 설정을 변경할 수 있기 때문에 restart 명령어를 이용해 공유기의 설정을 변경해주었다.

4. 모든 설정을 적용한 후, LOGGING_FILE의 경로에 로그를 기록한다. 이 때, 로그는 한 줄씩 기록이 축적되어야 하기 때문에 '>>' 기호를 사용하였다.

```
config interface 'wan'
        option classgroup 'Default'
        option upload '40000'
        option download '40000'
        option enabled '1'
```

/etc/config/qos 파일을 통해 QoS 기능 사용 여부와 다운로드 속도 및 업로드의 최대 속도를 제어할 수 있다. upload와 download는 최대 업로드 속도와 최대 다운로드 속도를 제어하는 역할을 하며, 단위로는 Kbps를 사용한다. connective 모드와 power_saving 모드에서는 기본 인터넷 속도의 1/2까지만 제공해야 한다. 현재 공유기가 연결되어 있는 네트워크의 평균 속도는 90Mbps이기 때문에 upload와 download의 값을 '40000'으로 지정해 주었다.

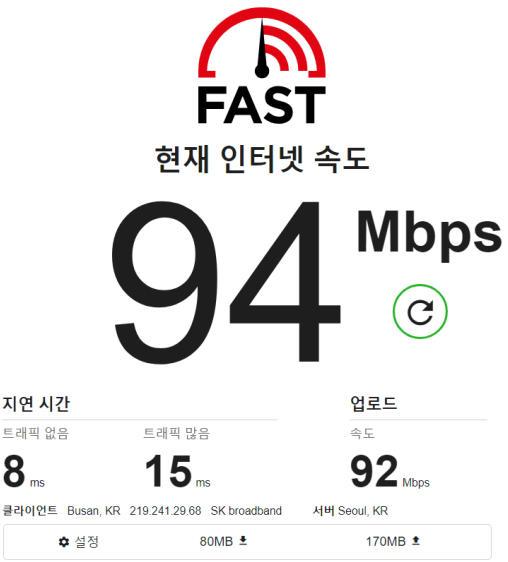
```
root@OpenWrt_origin:~/test_excute# ./profile_default
Start to change profile...(loading)
Finished to change profile...(loading)
```

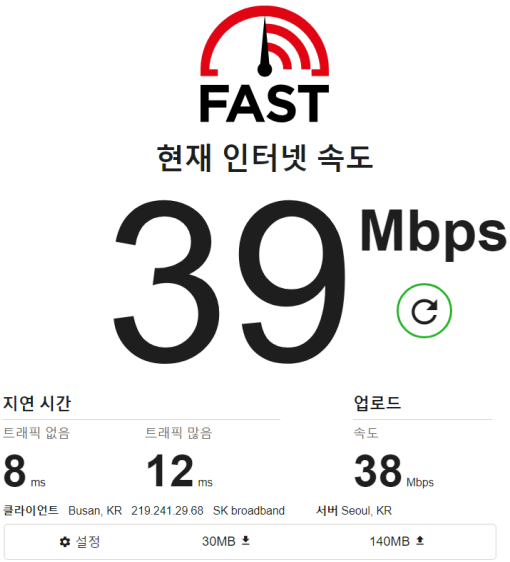
```
root@OpenWrt_origin:~/test_excute# ./profile_default
This router's mode is already [profile_default].
```

쉘 프로그램을 이용해 프로파일을 default로 변경할 수 있다. 만약 현재 공유기의 프로파일이 이미 default 모드일 경우, 변경사항이 적용되지 않고 프로그램이 바로 종료되는 것을 확인할 수 있다.

```
root@OpenWrt_origin:~/test_excute# ./profile_connective
Start to change profile...(loading)
Device for interface wan not found.
Device for interface wan not found.
Device for interface wan not found.
Device for interface wan not found.
```

공유기에 connective 프로파일을 적용하는 과정도 위와 같이 이루어진다. 인터페이스 적용 문제 때문인지 경고문이 뜨지만, 정상적으로 작동은 하였다. 현재는 해당 경고문이 뜨는 원인을 확인 중에 있다.





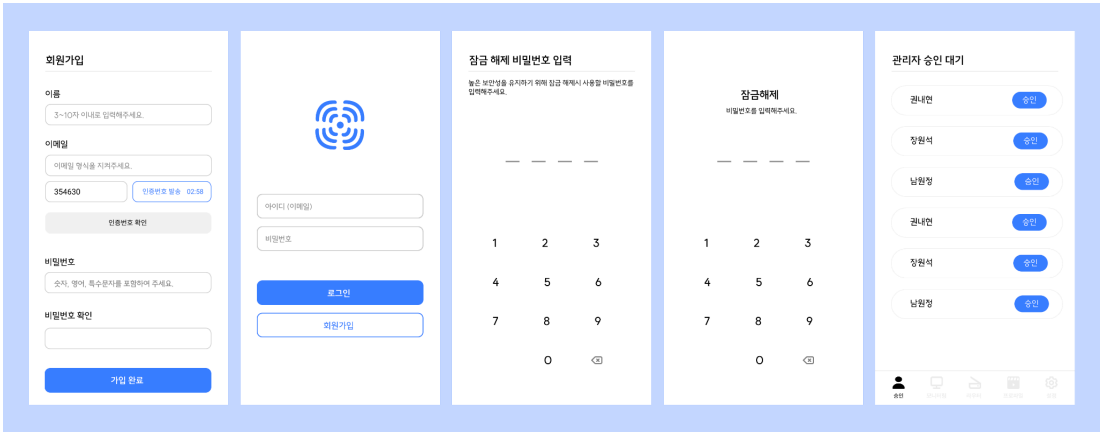
왼쪽은 default 모드를 적용시켰을 때의 네트워크 속도를 나타내며, 오른쪽은 connective 모드를 적용시켰을 때의 네트워크 속도를 나타낸다. connective 모드를 적용시켰을 때에는 QoS 설정 파일에서 지정해준 것처럼 업로드 및 다운로드 속도가 각각 40MB에 근접해있는 것을 확인할 수 있다.

예시는 default 모드를 설정하는 프로그램이지만, 다른 3가지 프로그램도 이와 비슷한 방식으로 작성하였다. 이렇게 작성한 4가지 쉘 프로그램을 원격 서버의 API와 연동한다면 관리자가 원격으로 공유기의 작동 프로파일을 바꿀 수 있어 보다 편리한 공유기 관리가 가능하다.

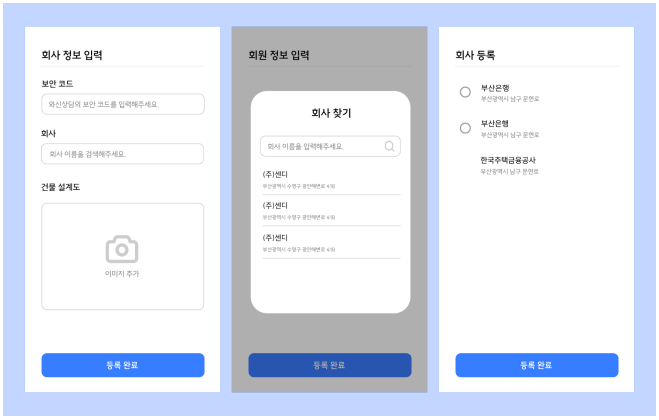
5.4 애플리케이션 설계

5.4.1 Figma를 이용하여 UI 설계

a. 유저 관련 (회원가입, 로그인, 잠금해제 패스워드 설정, 입력, 관리자 승인)

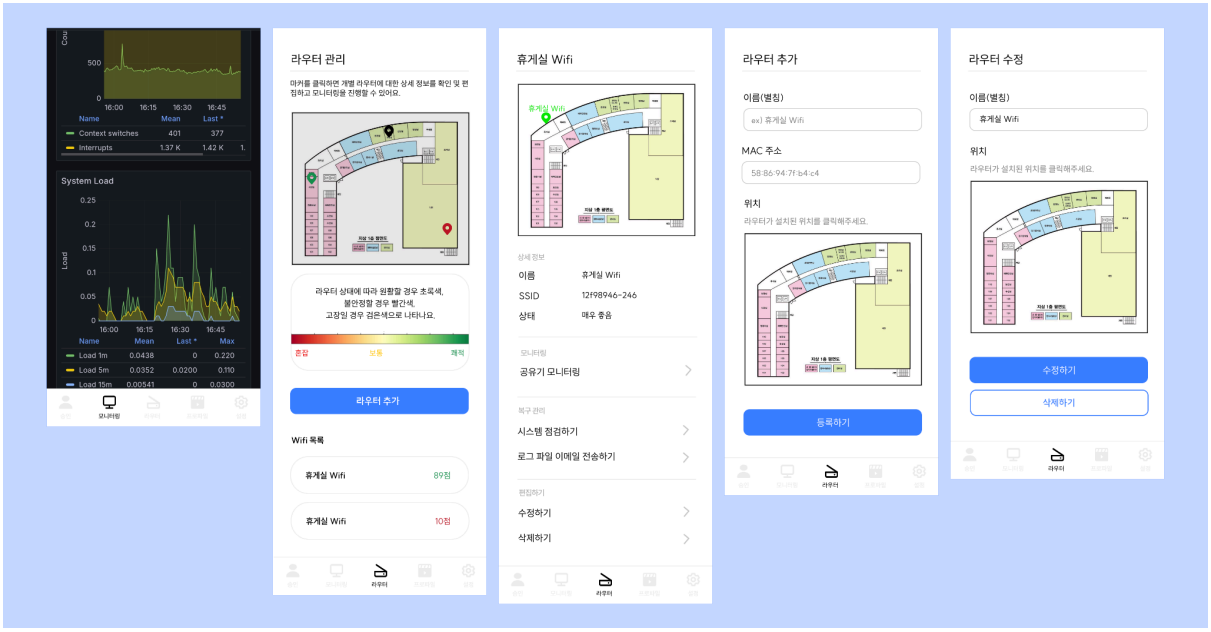


b. 회사 등록 (최종 관리자용 1, 2, 일반 관리자용)

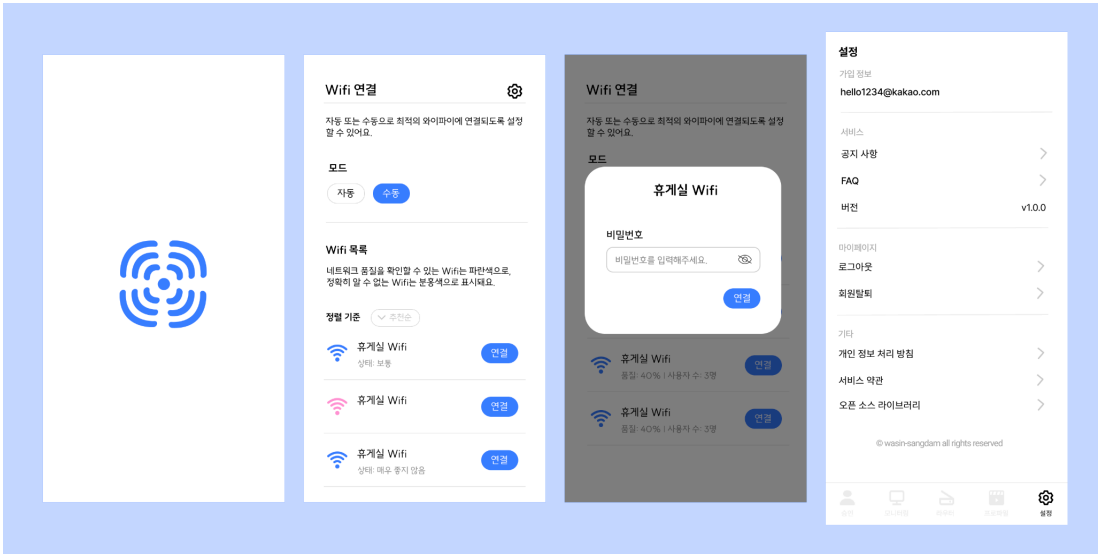


c. 프로파일 설정

d. 라우터 관련 (모니터링, 라우터 전체 조회, 라우터 개별 조회, 라우터 수정)

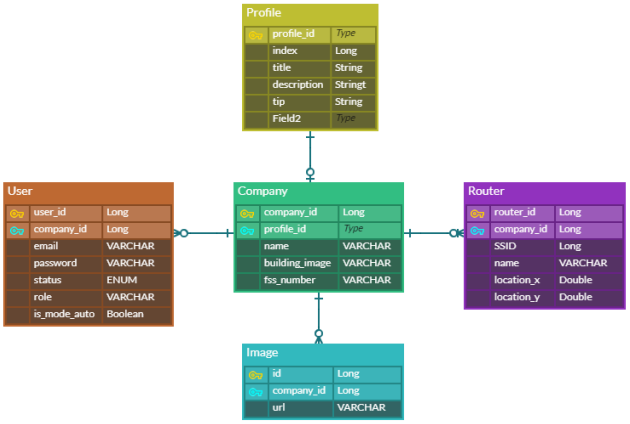


e. 기타 (스플래시, 와이파이 연결 1, 2, 설정)



5.4.2 ERD를 이용한 데이터베이스 설계

- **User:** 사용자의 이메일, 패스워드, 상태, 권한, 모드에 대한 정보를 담은 테이블
- **Profile:** 라우터 프로파일의 인덱스, 제목, 설명, 팁에 대한 정보를 담은 테이블
- **Company**: 회사의 이름, 빌딩 이미지, 고유 번호에 대한 정보를 담은 테이블
- **Router**: 라우터의 ssid, 이름, x좌표, y좌표에 대한 정보를 담은 테이블
- **Image**: 회사 건물 이미지에 대한 정보를 담은 테이블



5.4.3 API 문서 작성

▼ 유저 9

Aa API	Method	url	사용자	로그인 필요
회원가입	POST	/user/signup	공통	☐
로그인	POST	/user/login	공통	☐
로그아웃	POST	/user/logout	공통	☑
회원탈퇴	DELETE	/user/withdraw	공통	☑
4자리 잠금 패스워드 설정	POST	/user/lock	슈퍼 관리자 일반 관리자	☑
4자리 잠금 패스워드 확인	POST	/user/lock/confirm	슈퍼 관리자 일반 관리자	☑
토큰 재생성	POST	/user/refresh	공통	☐
이메일 인증 코드 전송	POST	/user/auth/email	공통	☐
이메일 인증 코드 확인	POST	/user/email/check	공통	☐

+ 새로 만들기

▼ 회사 4

Aa API	Method	url	사용자	로그인 필요
오픈 API 내 회사 목록 조회	GET	/company/open-api?name=name&page=1	슈퍼 관리자	☑
오픈 API 내 회사 정보 등록	POST	/company/open-api	슈퍼 관리자	☑
DB내 회사 목록 조회	GET	/company/db	일반 관리자	☑
DB내 회사 정보 등록	POST	/company/db	일반 관리자	☑

+ 새로 만들기

▼ 백오피스 2

Aa API	Method	url	사용자	로그인 필요
회원가입 승인	POST	/backoffice/accept	슈퍼 관리자	☑
회원가입 승인 대기자 목록	GET	/backoffice	슈퍼 관리자	☑

+ 새로 만들기

▼ 공유기 7

Aa API	Method	url	사용자	로그인 필요
관리자용 공유기 조회	GET	/router	슈퍼 관리자 일반 관리자	☑
관리자용 개별 공유기 조회	GET	/router/routerId	슈퍼 관리자 일반 관리자	☑
공유기 추가	POST	/router	슈퍼 관리자 일반 관리자	☑
공유기 수정	PUT	/router	슈퍼 관리자 일반 관리자	☑
공유기 삭제	DELETE	/router	슈퍼 관리자 일반 관리자	☑
모니터링	GET	/router/monitoring	슈퍼 관리자 일반 관리자	☑
개별 모니터링	GET	/router/monitoring/routerId	슈퍼 관리자 일반 관리자	☑

+ 새로 만들기

▼ 프로파일 4

Aa API	Method	url	사용자	로그인 필요
프로파일 조회	GET	/profile	슈퍼 관리자 일반 관리자	☑
프로파일 모드 자동으로 변경	POST	/profile/mode/auto	슈퍼 관리자 일반 관리자	☑
프로파일 모드 수동으로 변경	POST	/profile/mode/manual	슈퍼 관리자 일반 관리자	☑
프로파일 변경	POST	/profile/profileId	슈퍼 관리자 일반 관리자	☑

+ 새로 만들기

5.5 인증, 관리 서버 API 개발

5.5.1 유저 인증, 회사 등록, 백오피스, 공유기, 프로파일 API 구현

Spring Security를 이용하여 인증된 사용자만 이용이 가능하며, 각자 권한에 맞는 API만 호출할 수 있다. 중요한 정보를 담은 테이블은(User, Profile, Router, Image, Company) MySQL을 이용하여 관리되며, 만료기간이 있는 정보를 담은 내용(Email, Token)은 비관계형 데이터베이스인 레디스를 이용하여 관리함으로써 빠르게 접근한다.

유저 인증 관련 API는 모든 사용자가 이용할 수 있으며, “회원가입, 로그인, 로그아웃, 회원탈퇴, 잠금해제 패스워드 설정, 잠금해제 패스워드 확인, 토큰 리프레시, 이메일 인증 코드 전송, 이메일 인증 확인”이 있다.

백오피스 관련 API는 최종 관리자만 이용할 수 있으며 대기중인 일반 관리자 목록을 조회하고, 승인요청을 통해 일반 관리자를 활성화할 수 있다.

공유기 관련 API는 관리자만 이용할 수 있으며, “전체 조회, 개별 조회, 추가, 수정, 삭제”가 있다. 공유기를 조회할 땐 그라파나 서버의 HTTP API를 이용하여 상태를 받아온다. 공유기 상태는 다음 식을 통해 계산한다. 공유기 상태를 계산한 값은 0에서 100 사이의 실수 값을 가지며, 100에 가까울 수록 공유기가 현재 최적의 상태에서 돌아가고 있는 것을 의미한다. 계산한 공유기 상태값은 모바일 앱에서 지도 상의 공유기 상태를 표시할 때에 사용된다. 상태 값은 쾌적함, 혼잡함 두 가지 상태를 구별하기 위해 사용한다. Router_score 값이 특정 수치 이하일 경우, 해당 공유기는 혼잡한 상태인 것으로 가정한다.

다만, 그 threshold를 어떤 값으로 잡아야 할지는 구체적으로 정하지 않은 상태이며, 이후 다중 공유기 환경을 구성한 후 여러 공유기의 score 값을 종합적으로 관찰하여 구체적인 값을 산정할 계획이다.

```
avg(wifi_network_quality{instance=~"$node:$port",job=~"$job"}) *
    (1 - avg(node_load1{instance=~"219.241.29.68:9100",job=~"routers"}) /
        count(count(node_cpu_seconds_total{instance=~"219.241.29.68:9100",job=~"routers"}) by (cpu)))
```

관리자는 라우터 정보를 이용하여 추가할 수 있다. 이때, 요청받은 라우터가 그라파나 서버에 있으며, 현재 서버에 없는지 확인하여 유효한 경우에만 추가하도록 제어한다. 수정과 삭제도 할 수 있다.

회사 관련 API는 "최종 관리자용 회사 목록 조회, 등록, 일반 관리자용 회사 목록 조회, 등록"이 있다. 일반 사용자는 이용할 수 없다. 최종 관리자는 공공데이터 포털의 Open API인 금융위원회 기업기본정보를 이용하여 불러온 회사 정보를 조회하고, 그 중 한 회사를 선택하여 회사 정보와 이미지를 시스템 내 데이터베이스에 등록한다. 이때, 회사 이미지는 AWS S3를 이용하여 관리함으로써 높은 확장성과 신뢰성을 확보했다. 일반 관리자는 시스템 데이터베이스 내 저장된 회사 목록을 불러오고, 그 중 한 회사를 선택한다. 또한, 보안성을 높이기 위해 최종관리자가 기업을 등록할 땐 서비스 보안키를 입력하여 인증받는다. 이 기능은 관리자만 이용할 수 있다.

프로파일 관련 API는 관리자만 이용할 수 있으며 "조회", "모드 자동으로 변경", "모드 수동으로 변경", "프로파일 선택"이 있다. 프로파일은 서버 내 json 파일을 통해 관리되며, 서비스 구동시 DB에 해당 내용이 기록된다. 그리고, 매일 새벽 3시에 파일 내용과 DB 내용을 비교하여 만약 두 내용이 다를 경우, 파일 내용으로 변경한다. "조회"는 서버 내에서 관리하는 모든 프로파일 정보를 출력하는 기능이다. 또한 모드를 자동, 수동으로 변경할 수 있으며 프로파일을 선택할 수 있다. 이때, 프로파일 선택은 수동 모드일때만 가능하다.

5.5.2 서버 테스트 코드 작성과 CI 구축

현재는 컨트롤러에 대한 통합 테스트를 진행했으며, MySQL, Redis 등 외부 모듈을 많이 사용하기 때문에 Test Container를 이용하여 실제로 동작하는 것처럼 테스트를 진행했다. 또한 Git flow를 이용하여 브랜치를 develop, release, master로 분리하여 관리했으며, 각 브랜치로 PR를 올렸을 때, 테스트 코드를 실행하고 서버를 30초동안 구동하여 시스템의 안정성을 높였다.

5.5.3 그라파나, 공공데이터 포털 Open API 등 외부 API와의 통신 테스트

그라파나 서버로 전체 라우터 리스트 조회, 각 라우터별 상태 조회, 알림 활성화/비활성화를 테스트했으며 금융위원회 기업기본정보 API로부터 기업 정보를 조회하는 부분을 테스트했다. 또한, 그라파나 서버에서 현재 서버로 웹훅을 이용하여 알림을 보냈을 때 전송되는지 확인했다.