



국민대학교
전자정보통신대학
컴퓨터공학부


캡스톤 디자인 I

종합설계 프로젝트

| | |
|--------|----------|
| 프로젝트 명 | 작은 서버 |
| 팀 명 | 원숭이띠 미혼남 |
| 문서 제목 | 결과보고서 |

| | |
|---------|-------------|
| Version | 1.2 |
| Date | 2018-MAY-29 |

| | |
|----|----------|
| 팀원 | 강현구 (조장) |
| | 박주언 |
| | 송민석 |
| | 조경문 |
| | |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인Ⅰ 수강 학생 중 프로젝트 “작은서버”를 수행하는 팀 “원송이띠 미혼남”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “원송이띠 미혼남”의 팀원들의 서면 허락 없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


| | |
|-----------------|--------------------|
| Filename | 수행결과보고서-5조.docx |
| 원안작성자 | 강현구, 박주언, 송민석, 조경문 |
| 수정작업자 | 강현구, 박주언, 송민석, 조경문 |

| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
|------------|-------|----------|----------|-----------|
| 2018-05-20 | 강현구 | 1.0 | 최초 작성 | 초안 작성 |
| 2018-05-21 | 송민석 | 1.1 | 내용 추가 | 내용 추가 |
| 2018-05-23 | 박주언 | 1.2 | 내용 수정 | 내용 수정 |
| 2018-05-27 | 조경문 | 1.3 | 내용 수정 | 내용 수정 |
| 2018-05-29 | 강현구 | 1.4 | 내용 추가 | 테스트케이스 추가 |
| | | | | |
| | | | | |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

목 차

| | | |
|----------|---------------------|----|
| 1 | 개요 | 4 |
| 1.1 | 프로젝트 개요 | 4 |
| 1.2 | 추진 배경 및 필요성 | 5 |
| 2 | 개발 내용 및 결과물 | 5 |
| 2.1 | 목표 | 5 |
| 2.2 | 연구/개발 내용 및 결과물 | 5 |
| 2.2.1 | 연구/개발 내용 | 5 |
| 2.2.2 | 시스템 기능 및 비기능 요구사항 | 7 |
| 2.2.3 | 시스템 구조 및 설계도 | 8 |
| 2.2.4 | 활용/개발된 기술 | 11 |
| 2.2.5 | 현실적 제한 요소 및 그 해결 방안 | 12 |
| 2.2.6 | 결과물 목록 | 13 |
| 2.3 | 기대효과 및 활용방안 | 14 |
| 3 | 자기평가 | 14 |
| 4 | 참고 문헌 | 14 |
| 5 | 부록 | 16 |
| 5.1 | 사용자 매뉴얼 | 16 |
| 5.2 | 테스트 케이스 | 16 |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

1 개요

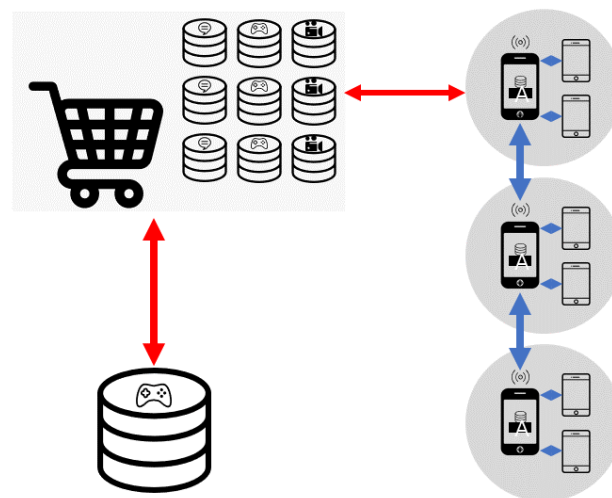
1.1 프로젝트 개요

서버 플랫폼이란 모바일 기기만으로도 간단하게 어플리케이션 서버를 여는 서비스를 제공하는 플랫폼이다. 이로써 대용량의 거대하다는 인식 속의 서버라는 통념을 깨고, 작은 모바일 기기도 서버가 될 수 있다는 생각을 제안한다.


그리고 특정 어플리케이션을 개발한 개발자에게는 이것이 서버로 구동될 수 있는 서비스를 제공하고, 이를 하나의 재화로 생각하여 서버 플랫폼을 이용하는 사용자들끼리 주고받을 수 있는 환경을 제공한다.

이렇게 만들어진 서버 플랫폼 ‘작은 서버’ 어플리케이션은 셀룰러 또는 Wi-Fi를 통해 인터넷과 연결되지 않은 상황에서도, 하나의 기기가 AP(Access Point)가 되어 근거리에서 있는 기기와 연결되어 작은 네트워크를 형성한다. AP의 기기 내부에 서버를 형성하여, 이와 연결된 다른 기기들과 함께 채팅 또는 게임과 같은 콘텐츠를 함께 실행할 수 있다.

‘작은 서버’는 이러한 기본적인 연결 구현을 바탕으로, 외부에 콘텐츠 서버를 구축하고, 어플리케이션을 서버에 업로드하여 하나의 상품가치가 되는 콘텐츠를 저장 및 관리한다. 이곳에서 다양한 콘텐츠들을 다운로드한 유저는 콘텐츠를 서버 위에서 작동하게 한다. 또한, 개발자들이 개발한 어플리케이션이 콘텐츠 서버에 하나의 상품으로 올려질 수 있도록 개발자들을 위한 개발환경, 모듈 및 라이브러리를 제공한다.



[그림 1-1] 서버 플랫폼 프로젝트 ‘작은 서버’ 구성도

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

1.2 추진 배경 및 필요성

요즘 인터넷이 없으면 아무것도 할 수 없는 세상이라고 해도 과언이 아니다. 만약 인터넷과의 연결이 되지 않는 상황이라면, 우리가 가진 수많은 디바이스들은 세상과 단절된 계산기, 메모장 또는 동영상 플레이어에 불과할 것이다. 그것을 절실히 그리고 가장 쉽게 체험할 수 있는 공간은 대표적으로 비행기 기내가 될 것이다. ‘같은 공간을 점유하고 있는 다른 승객들과 채팅을 하거나 게임을 할 수 있는 환경이 제공된다면 어떨지’와 같은 물음에서부터 연구는 시작되었다.

2 개발 내용 및 결과물

2.1 목표


일차적으로 한 기기가 Access Point가 되어, 그 기기가 서버 역할을 하며 간단한 채팅 또는 게임 기능을 제공하도록 구현하는 것을 목표로 한다. 그리고 ‘비행기 모드’에서도 AP Mode가 실행될 수 있게 하여, 두 기기사이의 연결을 가능하게 할 것이다. 이러한 기본적 구조에서 심화 및 발전시켜, 개발자들이 만든 어플리케이션을 손쉽게 서버에서 구동할 수 있는 모듈 라이브러리를 제공하고, 이렇게 제작된 콘텐츠를 외부 서버에서 다운로드하여 서버 내부에서 실행될 수 있게 한다.

그리고 최종적으로 재난 상황, 생태계, 그리고 정보에 소외된 지역에까지 사회적인 이슈에도 긴요하게 쓰일 수 있는 방안을 찾아 모색해보려 한다.

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

우리 서버 플랫폼은 네트워크가 없는 장소에서도 우리가 제공하는 서비스를 이용 할 수 있다는 것이 핵심이다. 우리 서비스는 local server 들을 이용하여 네트워크가 없는 장소나 특정 지역에서 본인이 원하는 기능들을 APP 설치 없이도 사용 할 수 있다. 유저가 Wifi 를 hotspot 을 터트려 rooting 이 된 디바이스로 AP 를 구축한다. Application 을 사용하고 있는 유저가 device 를 AP 로 사용하면 다른 유저들이 이를 wifi 로 잡아 Application 설치 없이도 유저가 사용하는 Application 의 기능들을 이용 할 수 있다. 이는 Web or raspberry pi 에서도 이용 가능하여 활용 방안이 크다. Application 의 main server 를 이용하게 되면 즉 네트워크를 사용하여 접속하면 모든 유저들이 올려놓은 여러 기능들을 다운 받아 이용 할 수 있게 되고, 유저들이 개발자가 되어 우리 서비스에 좋은 기능들을 올릴 수 있게 툴을 제공할 예정이다. 우리 서비스는 ad hoc network 를 이용하여 local server 들을 연결 시켜 서버를

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

구축한다. 오늘날 무선 네트워크의 경우, 노드와 사용자 이동성은 주로 포워딩을 통해 처리된다. 하지만 이 같은 포워딩 방식은, 네트워크 가장자리에 있는 노드(단말기)가 움직일 때만 적용되기 때문에 ad hoc 네트워크처럼 네트워크 중앙에 위치한 노드가 움직이거나 통신 기기가 라우터와 호스트 기능을 동시에 수행하는 경우에는 불가능하다. 따라서 ad hoc 네트워크의 경우, 라우팅 알고리즘이 이동성을 직접 처리한다. 만약 노드가 움직여 트래픽을 다른 쪽으로 강제로 옮기면, 라우팅 프로토콜은 노드의 라우팅 테이블에 일어난 변화를 관리한다.

■ AP로 동작하는 서버 기기와 클라이언트로서 동작하는 기기와 연결

서버를 인터넷 연결이 되어있지 않은 안드로이드 기기에서 실행하고, 테더링 기능을 활성화하게 만든다.

현재 안드로이드 환경에서는 수동으로 모바일 핫스팟을 켤 수 있는 기능이 존재하나, 해당 프로젝트의 시스템을 사용하기 위해 수동으로 핫스팟을 키는 방식은 번거로울뿐더러, 안드로이드 운영체제에서 기능적으로 에어플레인모드에서는 핫스팟을 동작할 수 없는 등의 제약 사항이 따르므로, 핫스팟 기능을 강제적이며 자동적으로 활성화하는 기능을 해당 시스템에 추가한다. 클라이언트 유저는 해당 시스템 앱을 거치지 않고 브라우저를 통해 로컬 서버에 접근할 수 있게 한다.

■ 콘텐츠 개발자의 메인서버로의 콘텐츠 업로드

콘텐츠 개발자는 자신이 만든 콘텐츠를 AWS EC2 기반의 기구축되어 있는 서버에 복잡한 절차 없이 업로드 할 수 있게 만든다.

채팅, 게임 등의 기능을 수반하는 콘텐츠를 개발한 후, 해당 시스템의 앱 내 ‘콘텐츠 병합’ 기능을 이용해 해당 콘텐츠를 서버로 동작하게 하는 module 과 병합한다.

이와 같은 module 은 개발자들의 콘텐츠가 어느 것인지와 상관없이 호환이 정상적으로 이루어져야 한다. 그 후 ‘콘텐츠 업로드’ 기능을 통해 해당 콘텐츠서버를 메인 서버에 업로드하게 된다.

■ 콘텐츠 Server를 만들기 위한 App 사용자의 콘텐츠 다운로드

App을 사용하는 사용자는 앱을 통해 메인서버로부터 원하는 콘텐츠를 다운받는다. 다운 완료된 콘텐츠는 실행 또는 삭제를 할 수 있다. 사용자의 기기가 local server로서의 기능을 하며 콘텐츠를 제공한다.

■ AP로 동작하는 서버 기기와 AP로 동작하는 서버 기기와의 연결

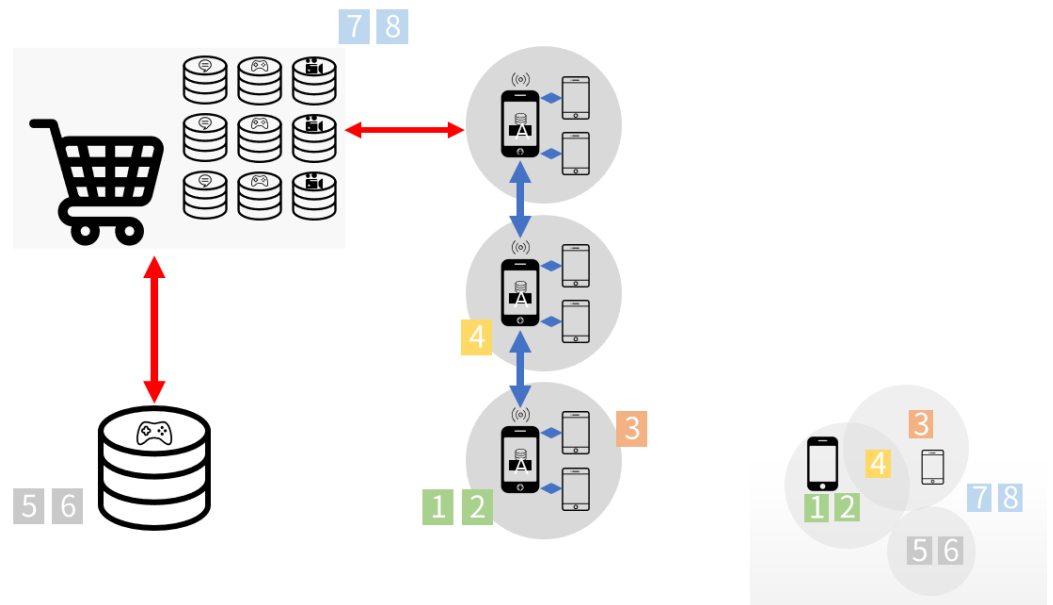
현재 안드로이드 환경에서는 핫스팟을 켜놓은 상태의 기기에서의 Wifi 연결을 지원하지 않는다. 따라서, 핫스팟 기능과 Wifi 모드를 함께 활성화하는 것을 강제할 수 있는 기능을 해당 시스템에 추가한다. 해당 기능은 앱 내에서 수동으로 선택할 수 있다.

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

서버 유저는 다른 서버에 연결되며, 이를 통해 서버들의 연쇄적인 연결이 가능하다. 작은 서버가 모여 Ad hoc 방식의 네트워크를 형성할 수 있다.


2.2.2 시스템 기능 및 비기능 요구사항

- 계획서에서 제시한 시스템 기능 요구사항



[그림 2-1] 서버 플랫폼 ‘작은 서버’ 시스템 기능 요구사항

1. Server User는 Group Owner가 되어 N 개의 기기와의 연결을 수용한다.
2. Server User는 N 개의 Group member 기기가 접근할 Server를 생성한다.
3. Client User는 Group Owner의 Group Member로서 연결될 수 있다.
4. Group Owner인 Client 또한 또다른 Group Owner의 Group Member로서 연결될 수 있다.
5. 개발자는 개발한 Application을 Contents Market Server에 Upload할 수 있는 Module을 통해 Contents로 변환한다.
6. 개발자는 변환된 Contents를 Contents Market Server에 업로드할 수 있다.
7. 모든 User는 Contents Market Server에 존재하는 Contents를 다운로드할 수 있다.
8. 모든 User는 Download한 Contents를 실행, 조회 또는 삭제할 수 있다.

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

- 시스템 기능 및 비기능 요구사항의 완료/변경/미완료 여부

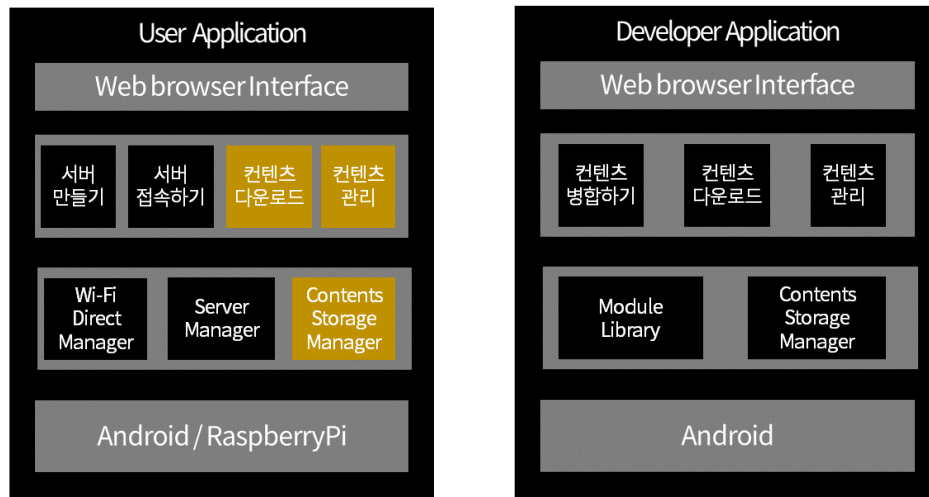
| 대분류 | 소분류 | 기능 | 내용 | 완료 여부 | 비고 |
|------------|------------|-----------------|---|-------|-------|
| 기능적 요소 | 연결 매니저 | 모바일 기기끼리의 연결 | AP Mode 를 기반으로 한 인프라스트럭처 네트워크 형성 | 완료 | 3 |
| | | | Wi-Fi Direct 를 기반으로 한 메시(애드혹) 네트워크 형성 | 변경 | 1. 4. |
| | 서버 매니저 | 서버파일 열기 | Node.js 의 node 명령을 통한 서버파일 열기 | 완료 | 2. |
| | | 서버파일 관리 | 안드로이드 내부 저장소의 서버파일 삭제 | 완료 | 8. |
| | 컨텐츠 매니저 | 개발자 도구 API | 개발자가 제작한 컨텐츠(어플리케이션)에 서버 구동을 실행하는 소스코드를 API 로써 제공 및 서버파일로 생성 | 미완료 | 5. |
| | 서버 마켓 | 서버파일 업로드 | AWS S3 서버에 개발자가 제작한 서버파일을 업로드 | 완료 | 6. |
| | | 서버파일 다운로드 | AWS S3 서버에서 개발자가 제작한 서버파일을 다운로드하여 안드로이드 내부 저장소에 저장 | 완료 | 7. |
| 비기능적 요소 | 성능 | 서버 성능 | N(서버 이용자수)의 값에 따른 서버 성능 저하 문제 해결 | 미완료 | |
| | 성능 | 기기의 배터리 성능 | 서버 부하 증가로 인한 배터리 사용량 문제 해결 | 미완료 | |

2.2.3 시스템 구조 및 설계도

- 계획서에서 제시한 아키텍처

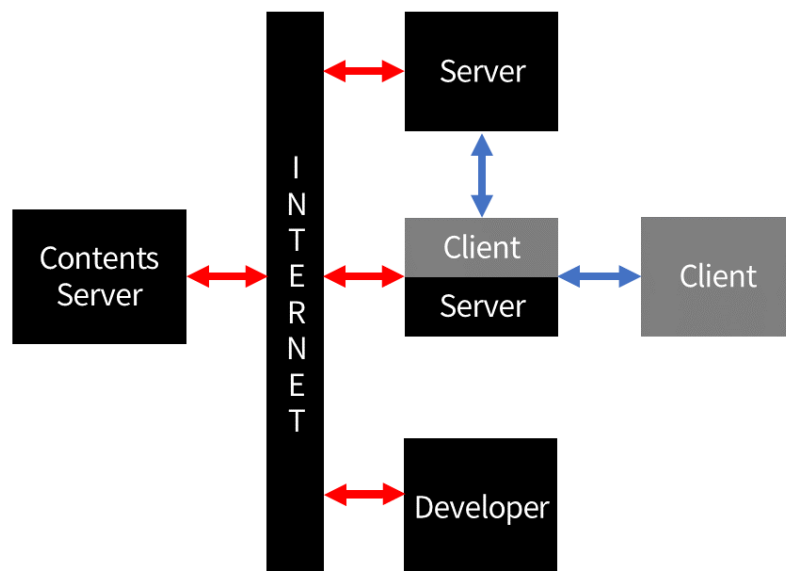
| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

■ Layer Architecture




[그림 2-2] ‘작은 서버’ Layer Architecture

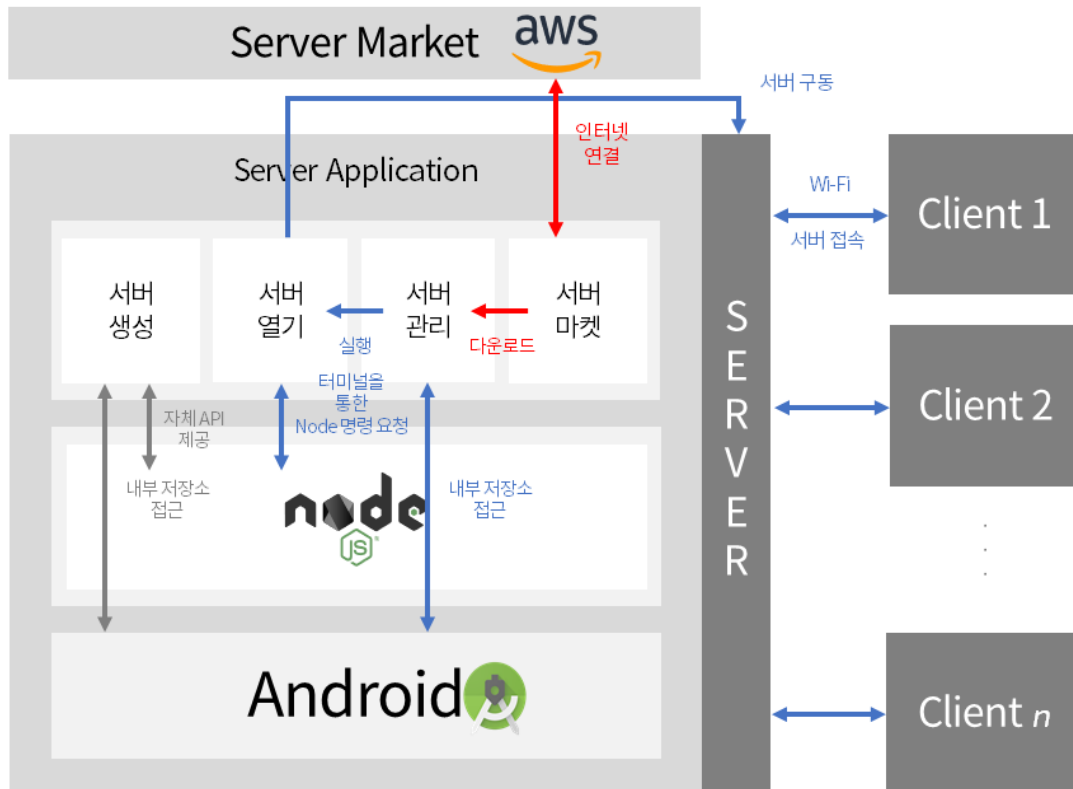
■ Client-Server Architecture



[그림 2-3] ‘작은 서버’ Client-Server Architecture(Plan A)

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

- 아키텍처 최종 버전

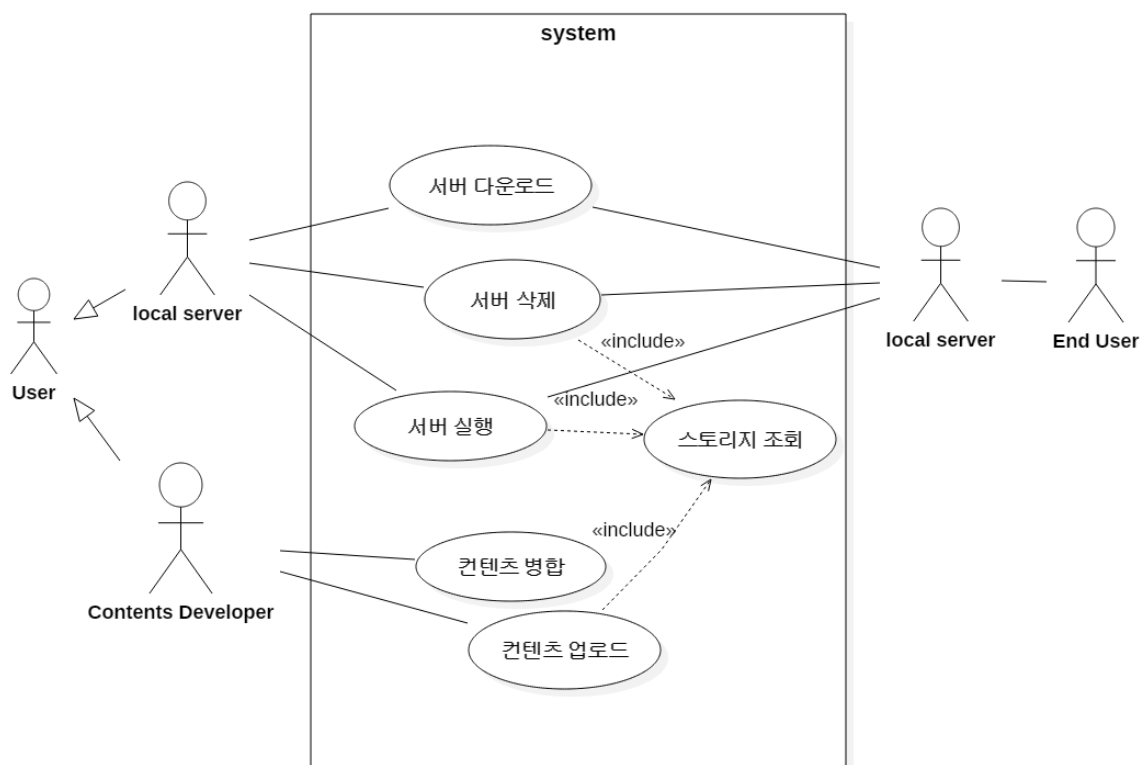


[그림 2-4] ‘작은 서버’ Client-Server Architecture(Plan A)

- 변경 사항 및 그 근거

- 1) Layer Architecture와 Client-Server Architecture를 동시에 두었음 : Layer Architecture의 경우 서버를 구동시키는 모바일 기기를 위한 안드로이드 어플리케이션에 대해서만 나타내었고, Client-Server Architecture의 경우 모바일 기기와 서버마켓 사이의 관계만 나타내었다. 일목요연하게 정리하는 것을 목적으로 수정하였음
- 2) 안드로이드 어플리케이션은 두가지로 계획하였으나(서버 기기/개발자용) 하나의 어플리케이션에서 모두 기능하도록 통합하였음
- 3) 기능에 따라 색상별로 기능 동작 시 실행흐름을 표시하였음

- Use-Case Diagram




[그림 2-5] 서버 플랫폼 프로젝트 ‘작은 서버’ Use-Case Diagram

- ✧ local Server: App을 이용해 서버를 가동하는 유저(아래 그림에서 Server User와 같다)
- ✧ Contents Developer: 콘텐츠를 만들고 App을 통해 메인서버에 업로드하는 개발자(아래 그림에서 개발자와 같다)
- ✧ End User: local Server 유저가 만든 서버와 연결되는 클라이언트 유저(아래 그림에서 Client User와 같다)

2.2.4 활용/개발된 기술

| 기술명 | 설명 |
|-----------------|--|
| Node.js Runtime | 서버 구동을 위한, Javascript 를 기반으로 하는 프레임워크 |
| AWS S3 | 서버 마켓을 제작하기 위해 사용된 서버 |
| Android Studio | Android OS 에서 우리 서비스 어플리케이션을 제작하기 위한 IDE |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

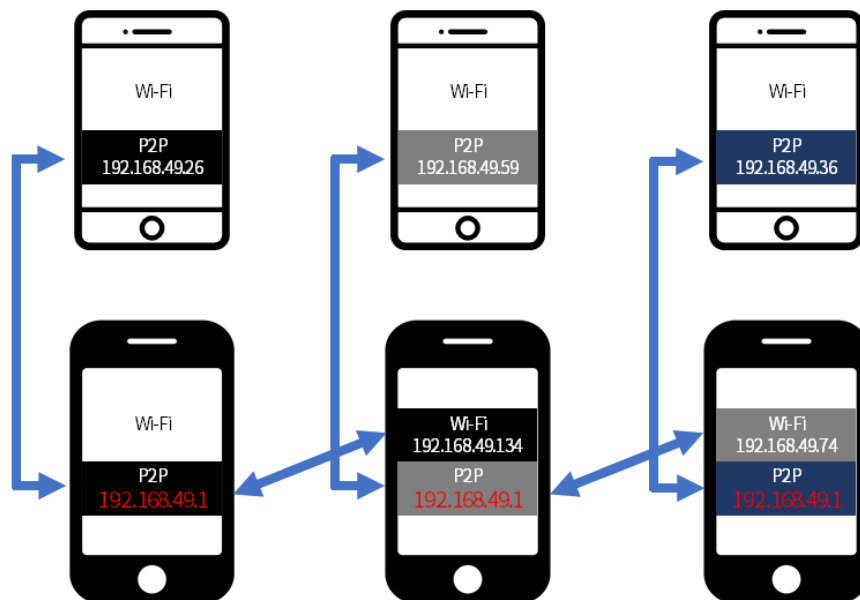
2.2.5 현실적 제한 요소 및 그 해결 방안

- 하드웨어

■ Wi-Fi Direct(WifiP2pManager):

개발 초기에는 Wi-Fi AP(Access Point) Mode, 즉 Hotspot을 이용하여 편리함을 우선으로 하였으나, Android OS에서 Wi-Fi 연결 상태에서 AP mode를 실현할 수 없다는 하드웨어상의 제약조건으로 인해 Wi-Fi Direct 기술을 사용하여 애드혹(메시) 네트워크를 구현하기로 하였다.

- 소프트웨어




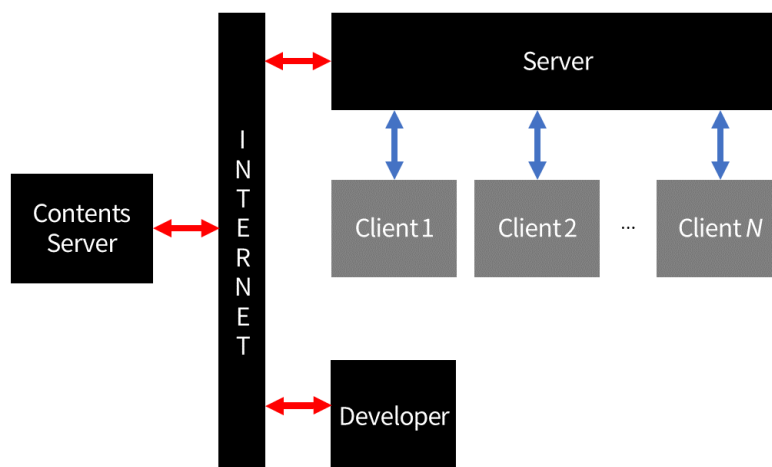
[그림 2-6] Wi-Fi Direct의 연결의 한계점(같은 IP주소 사용으로 인한 충돌)

■ Wi-Fi Direct(P2P)로 Group을 형성하는 경우,

(A,B) / (B,C) 와 같이 Wi-Fi Direct Group 두 개를 만든다고 가정하자. B는 두 Group에 동시에 속할 수 없다. Wi-Fi Alliance의 whitepaper에 따르면 이는 가능하나 선택사항으로 두었다. 그러나 Android OS에서는, 'P2P 그룹은 단일그룹 소유자와 하나 이상의 클라이언트로 구성된다'고 하였으므로, 여러 그룹 소유자가 있을 수 없다. 이는 위 [그림 2-6] 와 같이, 192.168.49.1 IP주소를 여러 기기가 가지게 되므로, 충돌이 발생하게 된다.

따라서 다음 문제점에 대해 논하는 영문 논문과, 국내 특허 발원이 된 발명 특허 논문을 통해 이를 분석하여 다음 문제에 대처해보려고 한다. 연결 문제를 먼저 해결하는 것보다 서버 플랫폼의 기본 형태를 구현하는 것이 우선순위라는 판단 하에, 결국 아래 그림과 같이 1:N 연결을 이용한 연결관계를 구현하였다. (Plan B)

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |




Plan B

[그림 2-7] ‘작은 서버’ Client-Server Architecture (Plan B, Plan A는 [그림 2-4] 참조)

2.2.6 결과물 목록

| 대분류 | 소분류 | 기능 | 기술 문서 |
|---------|--------------|-----------------------------|-------|
| 연결 매니저 | 모바일 기기끼리의 연결 | 서버를 구동하는 모바일 기기의 AP Mode 설정 | - |
| | | 클라이언트 기기의 Wi-Fi 연결 | - |
| 서버 매니저 | 서버파일 열기 | 서버를 구동하는 모바일 기기의 서버파일 열기 | - |
| | | 클라이언트 기기의 서버 접속 | - |
| | 서버파일 관리 | 서버를 구동하는 모바일 기기의 서버파일 삭제 | - |
| 컨텐츠 매니저 | 개발자 도구 API | 서버를 구동하는 모바일 기기의 서버파일 생성 | - |
| 서버 마켓 | 서버파일 업로드 | 서버를 구동하는 모바일 기기의 서버파일 업로드 | - |
| | 서버파일 다운로드 | 서버를 구동하는 모바일 기기의 서버파일 다운로드 | - |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

2.3 기대효과 및 활용방안


현재 대부분 휴대전화 사용자들은 Cellular 또는 Wi-Fi를 통한 인터넷 연결이 없으면 오프라인을 통한 개인활동(음악듣기, 다운 받아 놓은 콘텐츠 즐기기) 이외에는 휴대전화를 통해 할 수 있는 기능이 없게 된다. 하지만 누군가 자신의 모바일 기기를 이용해 AP mode를 통한 근거리 무선 네트워크를 형성한다면 인터넷 연결 없이도 주변 사람들과 연결되어 소통할 수 있게 될 것이다.

- ✓ 우리가 만든 어플리케이션이 가장 활약할 수 있는 부분은 큰 재난상황이거나 예기치 못한 일이 생겼을 때가 될 것이다. 인터넷으로의 연결은 차단된 상태로 사람들을 구출해야 할 상황일 때이다. 실종자가 되었든 구조대원이 되었든, 우리 어플리케이션을 통해 AP mode를 통한 근거리 무선 네트워크를 형성하여, 다른 실종자들이 이곳에 접속한 후 위치나 주변 정보 등을 공유한다면 인명구조를 효과적으로 할 수 있을 것으로 기대된다.
- ✓ 위같이 극단적인 상황이 아닌 경우에도, 가령 인터넷 연결이 불가능한 비행기 내에서 한 모바일 기기가 AP mode를 통한 근거리 무선 네트워크를 형성하여 서버를 구동시킨다. 그리고 이 기기에 연결된 다른 모바일 기기들이 클라이언트가 되어, 채팅 또는 게임 등의 콘텐츠를 함께 즐길 수 있을 것이다. 이를 통해 장시간 비행에서 오는 무료함을 달랠 수 있을 것이다.
- ✓ 어플리케이션 개발자의 경우 서버를 구동시키는 모듈을 생각하지 않고 어플리케이션 제작에만 집중할 수 있도록 한다. 우리 서비스가 개발자들을 위해 서버 구동을 위한 모듈 및 API를 제공한다.
- ✓ ‘작은 서버’를 활용하여 추후에 사회적 이슈를 해결해보고자 한다.
 - 멸종 위기에 처한 동물들의 생태계를 지키기 위해 ‘작은 서버’를 활용한다.
 - 정보에 소외된 계층, 지역의 사람들에게 보급하여 통신할 수 있는 기회를 제공한다.


3 자기평가

4 참고 문헌

| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
|----|-----|---|--------|------|----|----|
| 1 | 사이트 | https://stackoverflow.com/questions/35325 | 스택오버플로 | | | |

| | | | | | |
|---|-------------------------|--|-------------|--|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | | | |
| | 프로젝트 명 | | 작은 서버 | | |
| | 팀 명 | | 원송이띠 미혼남 | | |
| | Confidential Restricted | | Version 1.4 | | 2018-MAY-29 |

| | | | | | | |
|----|-----|---|--------|----------|--------|--------------------------|
| | | 454/create-a-lan-access-point-on-android-device-no-internet-needed/35327467#35327467 | | | | |
| 2 | 사이트 | https://academy.realm.io/kr/posts/difference-between-kotlin-java/ | | | | kotlin |
| 3 | 어플 | Servers Ultimate | 플레이스토어 | | | |
| 4 | 강의 | https://www.youtube.com/watch?v=7ThkvfCKKQs&list=PLuHgQVnccGMC5AYnBg8ffg5utOLwEj4fZ&index=1 | 유튜브 | | | AWS |
| 5 | 사이트 | http://mobileandlife.tistory.com/entry/ 모바일기술정리-와이파이-다이렉트-WIFI-Direct | 블로그 | | | 와이파이 |
| 6 | 서적 | 더 도커 북 | 루비페이퍼 | 2014. 10 | 제임스 턴볼 | |
| 7 | 사이트 | https://m.blog.naver.com/PostView.nhn?blogId=horusi&logNo=220337395781&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F | | | | AP 만들기 |
| 8 | 사이트 | http://comterman.tistory.com/621 | | | | 와이파이 다이렉트 |
| 9 | 사이트 | http://hsj0511.tistory.com/205 | | | | ajax |
| 10 | 사이트 | https://ruslanspivak.com/lbaws-part3/#fn-1 | | | | 웹서버 만들기 |
| 11 | 사이트 | https://blog.outsider.ne.kr/312#recentEntries | | | | http method |
| 12 | 사이트 | https://stackoverflow.com/questions/21533616/is-it-possible-to-connect-two-or-more-wifi-direct-groups | | | | |
| 13 | 사이트 | https://github.com/janeasystems/nodejs-mobile-nodejs | | | | |
| 14 | 사이트 | https://hackernoon.com/how-to-turn-an-android-device-into-a-web-server-9816b28ab199 | | | | termux를 사용한 nodejs 서버 실행 |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |


| | | | | | | |
|----|-----|---|--|--|--|-------------------------------------|
| 15 | 사이트 | https://android.googlesource.com/platform/frameworks/base+/refs/heads/master/wifi/java/android/net/wifi/p2p | | | | 안드로이드 스튜디오 내부 api 코드 페이지 node.js |
| 16 | 사이트 | https://socket.io/get-started/chat/ | | | | nodejs에서 사용하는 socket.io |

5 부록


5.1 사용자 매뉴얼

5.2 테스트 케이스

| 대분류 | 소분류 | 기능 | 테스트 방법 | 기대 결과 | 테스트 결과 |
|--------|--------------|-----------------------------|---|---|--------|
| 연결 매니저 | 모바일 기기끼리의 연결 | 서버를 구동하는 모바일 기기의 AP Mode 설정 | AP Mode를 기반으로 한 인프라스트럭처 네트워크 형성 좌측 메뉴바 버튼을 누른 후 1) ‘핫스팟 설정’ 버튼을 터치한다. 2) 안드로이드 설정의 ‘모바일 핫스팟’으로 이동한다. 3) SSID와 비밀번호를 설정한다. | 모바일 기기끼리의 연결에 필요한 AP Mode의 SSID 및 비밀번호가 설정된다. | 성공 |
| | | 클라이언트 기기의 Wi-Fi 연결 | 1) 각 클라이언트 기기의 Wi-Fi 설정 메뉴로 이동한다. 2) 위에서 설정한 SSID를 찾아 올바른 비밀번호를 입력한다. | 연결하려는 서버 모바일 기기에 Wi-Fi 연결에 성공한다. | 성공 |
| 서버 매니저 | 서버파일 열기 | 서버를 구동하는 모바일 기기의 서버파일 열기 | 1) 안드로이드 ‘작은 서버’ 서비스 어플리케이션을 실행한다. 2) 초기 화면에서 다운로드된 서버 파일을 터치하여 선택한다. | 원하는 다운로드 서버 파일을 안드로이드 내부 저장소에 | 성공 |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원송이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

| | | | | | |
|---------|------------|----------------------------|---|---|----|
| | | | 3) 2)에서 선택할 수 있는 서버파일이 없으면, 좌측 상단의 메뉴바 버튼을 터치하여 선택한다. 4) ‘서버 마켓’에 접속하여 다운로드할 서버 파일을 선택한다. | 다운로드한다. | |
| | | 클라이언트 기기의 서버 접속 | Wi-Fi 연결 이후, 웹브라우저에서 HTTP 프로토콜을 이용하는 특정 주소 및 포트번호를 입력하여 접속한다.(http://192.168.43.1:8080) | 서버 기기가 열려둔 서버에 접속하여 서버를 이용할 수 있다. | 성공 |
| | 서버파일 관리 | 서버를 구동하는 모바일 기기의 서버파일 삭제 | 1) 초기 화면에서 삭제할 서버 파일을 길게 터치하여 선택한다. 2) ‘해당 서버를 삭제하시겠습니까?’ 선택 팝업창에서 ‘삭제’를 선택한다. | 삭제하기를 원하는 서버 파일을 안드로이드 내부 저장소에서 삭제한다. | 성공 |
| 컨텐츠 매니저 | 개발자 도구 API | 서버를 구동하는 모바일 기기의 서버파일 생성 | 개발자가 제작한 콘텐츠(어플리케이션)에 서버 구동을 실현하는 소스코드를 API 로써 제공 및 서버파일로 생성 1) 초기 화면에서 좌측 상단의 메뉴바 버튼을 터치하여 선택한다. 2) 개발자 메뉴의 ‘서버 생성’을 터치하여 선택한다. 3) 서버 파일로 변환할 어플리케이션을 선택한다. | 개발자가 제작한 어플리케이션이 서버로 구동될 수 있도록 하는 서버 파일을 생성한다. | 실패 |
| 서버 마켓 | 서버파일 업로드 | 서버를 구동하는 모바일 기기의 서버 파일 업로드 | 1) 초기 화면에서 좌측 상단의 메뉴바 버튼을 터치하여 선택한다. 2) 개발자 메뉴의 ‘서버 업로드’를 터치하여 선택한다. 3) 업로드할 서버 파일을 선택하여 ‘업로드’ 버튼을 터치한다. | 업로드를 원하는 서버 파일을 외부 인터넷망의 ‘서버 마켓’ AWS 서버에 업로드한다. | 실패 |

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 결과보고서 | | |
| | 프로젝트 명 | 작은 서버 | |
| | 팀 명 | 원숭이띠 미혼남 | |
| | Confidential Restricted | Version 1.4 | 2018-MAY-29 |

| | | | | | |
|--|--------------|---|---|--|----|
| | 서버파일 다운로드 | 서버를 구동하는 모바일 기기의 서버 파일 다운로드 | 1) 초기 화면에서 좌측 상단의 메뉴바 버튼을 터치하여 선택한다. 2) '서버 마켓'을 터치하여 선택한다. 3) 마켓에 존재하는 서버 파일을 선택한다. 4) 서버 파일의 설명이 팝업으로 나타나면서 '다운 받기'를 터치한다. | 다운로드를 원하는 서버 파일을 외부 인터넷망의 '서버 마켓' AWS 서버에서 안드로이드 내부 저장소로 다운로드한다. | 성공 |
|--|--------------|---|---|--|----|