**S/W멤버십 과제 보고서**

**과제명 : All Seeing Eye Relay Chat**

* 소 속 : 부산 소프트웨어 멤버십
* 작성자 : 이광오, 정재현, 김승현
* 작성일 : 2014. 02. 09

**S/W멤버십 과제 기획서**

|  |  |  |  |
| --- | --- | --- | --- |
| **과 제 명** | All Seeing Eye Relay Chat | | |
| **과제구분** | 창의과제 | | |
| **과제기간** | 2014. 11. 01 ~ 01. 31 (3개월) | | |
| **지 역** | 부산 멤버십 | **참여인원** | 3명 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **회 원 명** | **학교** | **학과** | **학년** | **연락처** | **E-mail** |
| 이광오 | 동의대 | 소프트웨어공학 | 4 | 010-3252-9046 | pu8046@gmail.com |
| 정재현 | 동의대 | 컴퓨터공학 | 4 | 010-9235-7648 | jaehon2002@naver.com |
| 김승현 | 부경대 | 컴퓨터공학 | 4 | 010-2696-1208 | shyeunee92@naver.com |

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **내용** | | |
| **개발 목적**  **및 동기** | 오픈 소스 활동에서 가장 중요한 것들 중 하나가 바로 개발자 간의 의사소통이다. IRC는 실시간으로 전 세계 사람들과 실시간으로 소통할 수 있기에 오픈 소스 활동에서 굉장히 편리한 수단이 될 수 있지만 부재중일 경우 지나간 대화를 확인하기 힘들다는 단점이 있다. 본 과제는 이러한 단점을 해소해 줄 것이다. | | |
| **개발 환경**  **및 일정** | * 운영체제 : Windows 7 / 8.1, Android 4.4 KitKat / 5.0 Lollipop * 개발 툴 : Android Developer Tools, Visual Studio 2010, dStudio 1.0 * 개발언어 : Java, C#, mSL * 라이브러리 : NDde * 개발기간 : 2014. 11. 01 ~ 2015. 01. 31 (3개월) | | |
| **창의성/**  **우수성** | IRC는 접속을 유지하였을 때만 대화를 주고받을 수 있기 때문에 접속이 종료되면 대화 내용을 얻을 수 없으며, 부재중일 경우에도 다른 사용자가 호출하는 것을 사용자는 알 수가 없다. 하지만 본 과제는 사용자가 부재중 시 스마트 기기로 원격 연결하여 지속적인 접속을 유지할 수 있도록 하며, 로그 수집을 통하여 부재 등의 사유로 원격 연결이 지체되더라도 끊임없는 대화의 연속성을 보장해준다. | | |
| **활용성/**  **사업성** | 사용 범위에 따라 지나간 대화내용을 언제든지 확인할 수 있는 단순한 Log Bot에서부터 필요에 따라 원격 IRC 클라이언트로의 기능 수행까지 지원한다. 기존의 여러 문제점을 보완하고 기능을 개선함으로써 진일보한 시스템이라 할 수 있으며, 프리웨어로 공개할 경우 많은 사용자들에게 각광 받을 수 있으리라 예상한다 | | |
| **지원부서** | S/W멤버십 | **기술지원 연구원** | 배 보 람 |
| **전화번호** | 051-636-1542 | **E-Mail** | bbrto21@gmail.com |

**목 차**

**- 내용 목차 -**

[1. 개발 목적 5](#_Toc411240301)

[1) IRC and Open Source Projects 5](#_Toc411240302)

[2. 개발 목표 6](#_Toc411240303)

[1) 필요할 때만 사용하는 절약형 어플리케이션 6](#_Toc411240304)

[2) 실시간 구독 및 중요 메시지 Push 알림 6](#_Toc411240305)

[3) 원격 컨트롤러를 통한 기본 기능 수행 6](#_Toc411240306)

[4) 직관성과 사용자 경험 6](#_Toc411240307)

[3. 개발내용 7](#_Toc411240308)

[1) 전체 시스템 아키텍처 7](#_Toc411240309)

[2) 클라이언트 아키텍처 7](#_Toc411240310)

[3) 안드로이드 어플리케이션 아키텍처 8](#_Toc411240311)

[4) 중앙 서버 아키텍처 8](#_Toc411240312)

[4. 세부 사항 9](#_Toc411240313)

[1) 클라이언트 간의 통신 방법 9](#_Toc411240314)

[2) 인증 및 연동 10](#_Toc411240315)

[3) Windows Application 11](#_Toc411240316)

[4) Android Application 13](#_Toc411240317)

[5. 개발 일정 20](#_Toc411240318)

[1) 개발 일정 20](#_Toc411240319)

[2) 역할 분담 20](#_Toc411240320)

[6. 용어 정리 21](#_Toc411240321)

[1) IRC 21](#_Toc411240322)

[2) mIRC Scripting Languages 21](#_Toc411240323)

[3) GCM(Google Cloud Messaging) 22](#_Toc411240324)

[4) DDE 22](#_Toc411240325)

[6. 참고 문헌 23](#_Toc411240326)

**- 그림 목차 –**

[그림 1 Internet Relay Chat 5](#_Toc411240272)

[그림 2 open source 6](#_Toc411240273)

[그림 3 System Architecture 7](#_Toc411240274)

[그림 4 Client Architecture 7](#_Toc411240275)

[그림 5 Android Application Architecture 8](#_Toc411240276)

[그림 6 Central Server Architecture 8](#_Toc411240277)

[그림 7 클라이언트 간의 통신 9](#_Toc411240278)

[그림 8 인증 및 연동 10](#_Toc411240279)

[그림 9 Status Window 11](#_Toc411240280)

[그림 10 Channel Window 11](#_Toc411240281)

[그림 11 사용자 설정 다이얼로그 12](#_Toc411240282)

[그림 12 단축 실행 설정 다이얼로그 12](#_Toc411240283)

[그림 13 자동조인 컨트롤러 13](#_Toc411240284)

[그림 14 Splash 13](#_Toc411240285)

[그림 15 Authorize 14](#_Toc411240286)

[그림 16 Channel List 14](#_Toc411240287)

[그림 17 Query List 15](#_Toc411240288)

[그림 18 Notice(1) 15](#_Toc411240289)

[그림 19 Notice(2) 16](#_Toc411240290)

[그림 20 Setting 17](#_Toc411240291)

[그림 21 Channel(1) 17](#_Toc411240292)

[그림 22 Channel(2) 18](#_Toc411240293)

[그림 23 Query 18](#_Toc411240294)

[그림 24 Disconnect/Connect 19](#_Toc411240295)

[그림 25 Notifications 19](#_Toc411240296)

[그림 26 개발 일정 20](#_Toc411240297)

[그림 27 역할 분담 20](#_Toc411240298)

[그림 28 mIRC Scripting Languages 21](#_Toc411240299)

[그림 29 Google Cloud Messaging 22](#_Toc411240300)

# 1. 개발 목적

## 1) IRC and Open Source Projects

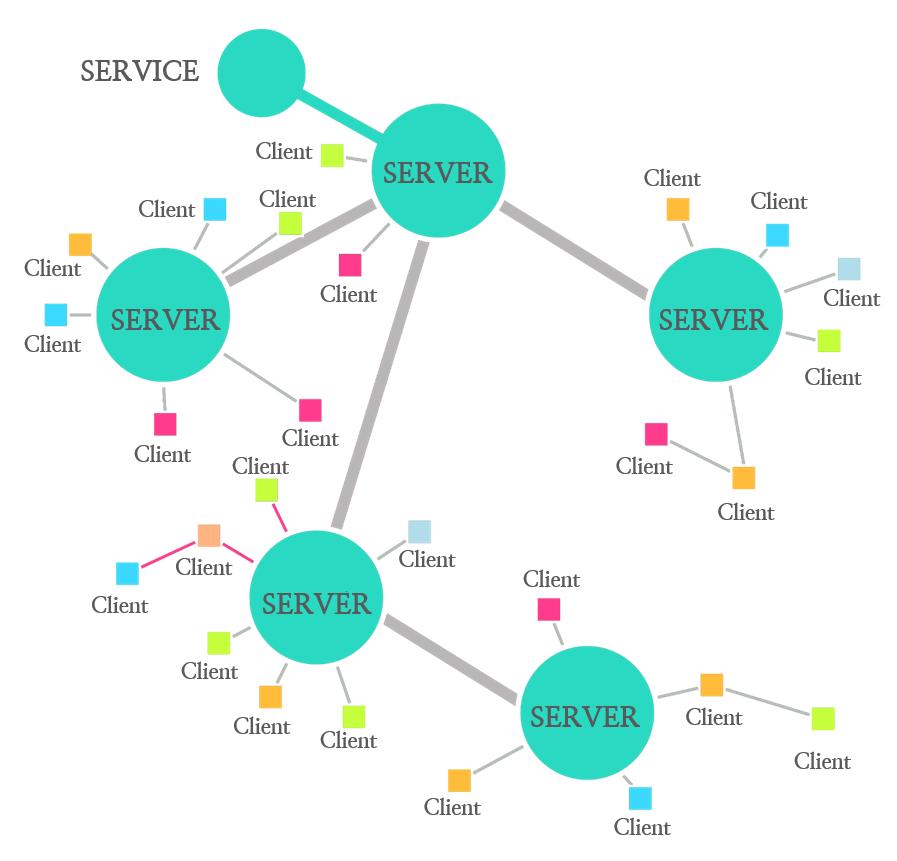


그림 Internet Relay Chat

IRC는 인터넷 릴레이 챗(Internet Relay Chat)의 머리글자를 딴 것으로, 실시간 대화를 뜻한다. 여러 서버를 병렬로 실시간으로 연결하여 대화를 나눌 수 있는 범세계적인 채팅 프로그램으로, 일반 PC통신과 비슷하지만, PC통신은 개인적인 성격이 강한 반면, IRC는 IRC 클라이언트 프로그램이나 IRC 클라이언트를 제공하는 서버에 접속하기만 하면 시간이나 공간에 구애 받지 않고 전세계의 어떤 사람과도 대화가 가능하다. 또, 동시에 다중 대화가 가능한 채팅 프로그램이다.

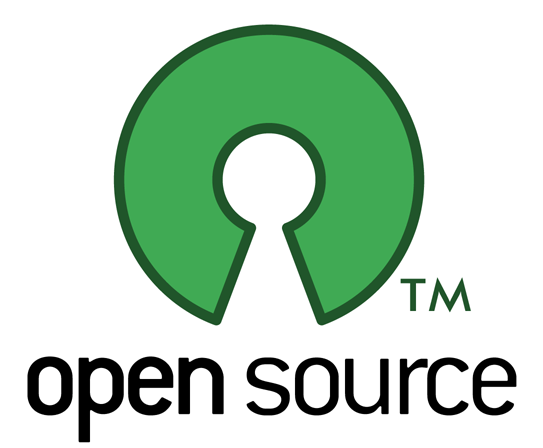


그림 open source

오픈 소스 개발자들은 대부분의 개발 관련 토론을 IRC와 Mailing List를 통해 진행한다. Mailing List의 경우 보관함에 메일이 저장되므로 언제나 열람이 가능하지만, IRC의 경우 실시간 채팅 프로그램이기 때문에 실시간으로 접속하지 않으면 해당 시간에 이루어지는 대화 내용을 알 수 없다. 따라서 항상 상주하는IRC Client를 이용하여 대화 기록을 어플리케이션 DB에 저장하고 어플리케이션을 통해 지나간 대화를 시간과 장소에 구애 받지 않고 열람할 수 있도록 하며 나아가 IRC 기본 기능 또한 충실하게 가져와 종전의 단점을 최대한 개선하고 장점만을 모아 높은 활용성을 지진 프로그램을 개발하는 것이 이번 과제의 목표이다.

# 2. 개발 목표

## 1) 필요할 때만 사용하는 절약형 어플리케이션

IRC를 사용하려면 항상 채널에 접속해 있어야만 대화내용을 알 수 있고, 호출에 대한 알림을 받아볼 수 있다. 그래서 사용자가 컴퓨터를 사용하고 있지 않고 밖에 있는 경우엔 대화내용을 알 수 없고, 채널에서 자신을 호출하여도 알 수 있는 방법이 없다. 대안으로 많은 이들이 Mobile용 IRC를 사용하고 있지만 이 또한 항상 연결이 지속되어야 하기 때문에 배터리의 소모가 빠르다는 단점이 있다. 이러한 단점을 해결하기 위해서는 우선적으로 채널 상주 부담을 줄이는 것이 시급한 상황이다.

## 2) 실시간 구독 및 중요 메시지 Push 알림

종전의 방법으로는 연결이 지속되어야 대화내용을 볼 수 있지만 사용자가 부재중일 경우에도 대화내용을 볼 수 있어야 하며 언제든지 원하는 키워드 메시지에 대한 알림을 받을 수 있어야 한다.

## 3) 원격 컨트롤러를 통한 기본 기능 수행

구독만 가능한 일방적인 단방향 통신만으로는 IRC의 제 기능을 수행할 수가 없다. 사용자가 원한다면 언제든지 어플리케이션을 통해 대화에 참여하고 파일을 전송할 수 있어야 하며, 여러 정보를 조회하는 등 PC와 거의 같은 기능을 수행할 수 있어야 한다.

## 4) 직관성과 사용자 경험

이 프로젝트의 결과물은 사용자가 어플리케이션 자체를 ‘보는’ 행위에서 많은 일들이 일어난다. 따라서 프로그램은 UI와 UX에 중점을 두어 사용자 편의성에 초점을 맞춘다.

# 3. 개발내용

## 1) 전체 시스템 아키텍처

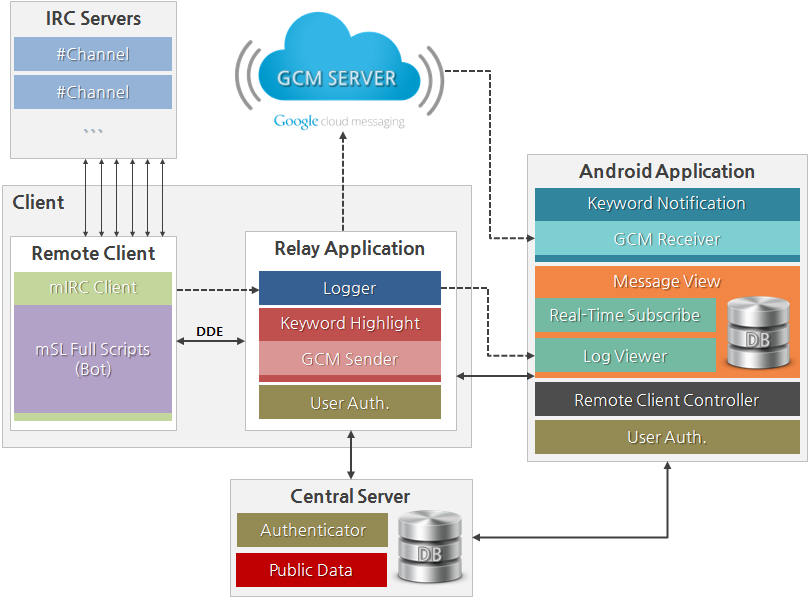


그림 System Architecture

## 2) 클라이언트 아키텍처

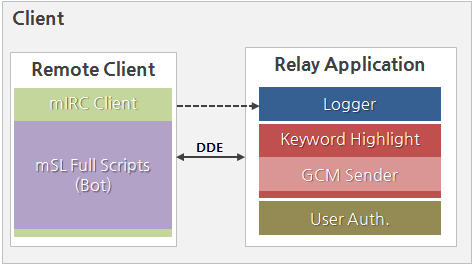


그림 Client Architecture

- Remote Client : mIRC를 기반으로 하며 내장 스크립트 언어인 mSL를 이용하여 시스템을 재구축하였다. 서버 및 채널에 지속적으로 상주함으로써 끊임없이 대화내용을 기록하며, Relay Application과 Dynamic Data Exchange를 이용한 통신으로 Logger를 탑재한 원격 IRC 클라이언트가 된다.

- Relay Application : Log 파일을 파일입출력을 통해 읽고 안드로이드 어플리케이션으로 전송해준다. 또한 Keyword Highlight를 통해 안드로이드 어플리케이션에서 사용자가 직접 설정한 키워드를 저장하고 Remote Client와 통신을 하여 해당 단어를 찾아내는 즉시 GCM을 통해 안드로이드 어플리케이션으로 Push 알림을 전송한다.

- User Auth. : 사용자 인증에 관련된 모듈로써 임의의 난수를 생성하여 중앙 서버로 보내어 안드로이드 어플리케이션과 자신의 PC에 설치된 클라이언트를 정확하고 안전하게 연동 시켜준다

## 3) 안드로이드 어플리케이션 아키텍처

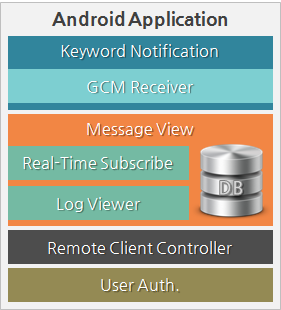


그림 Android Application Architecture

* Keyword Notification : Relay Application에서 해당 Keyword가 발견되는 즉시 GCM Receiver로 통신이 오게 되고 사용자에게 Notification 메시지를 출력하여 사용자가 언제든지 호출에 대응할 수 있도록 한다.
* Message View : Real-Time Subscribe를 통해 해당 채널에서 진행되고 있는 대화를 실시간으로 확인할 수 있다. 또한 Log Viewer를 통해 이미 지나간 대화도 언제든지 다시 확인할 수 있다.
* Remote Client Controller : 내장된 Controller를 통해 IRC Client를 원격으로 제어할 수 있다. 마치 IRC Client를 사용하듯이 똑같이 실시간으로 대화를 할 수 있고 채널 및 사용자에 대한 정보조회가 가능하며, 1:1 대화 및 파일 전송까지 수행할 수 있다.
* User Auth. : 사용자 인증에 관련된 모듈로써 클라이언트가 생성한 임의의 난수를 사용자가 확인하고 직접 입력하여 중앙 서버를 통해 정확하게 매칭시켜준다.

## 4) 중앙 서버 아키텍처

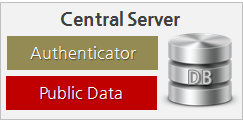


그림 Central Server Architecture

* Authenticator : 서버에 내장된 인증 시스템이다. PC 클라이언트와 안드로이드 어플리케이션을 정확하고 안전하게 연동시켜주는 역할을 한다.
* Public Data : 각종 사용자 정보를 DB에 저장하며, 사용자들의 사용 현황을 비롯한 여러 정보들을 지속적으로 축적한다.

# 4. 세부 사항

## 1) 클라이언트 간의 통신 방법

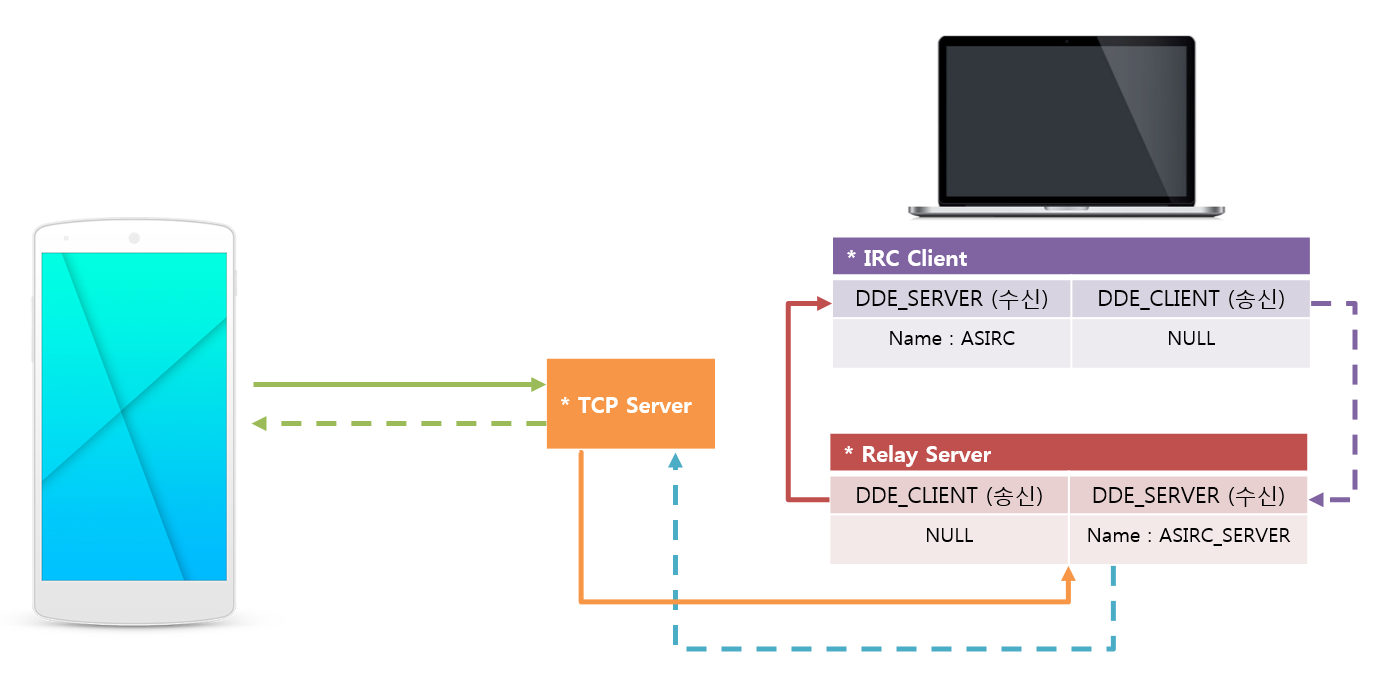


그림 클라이언트 간의 통신

* Windows Application과 Android Application의 상호 연동의 핵심은 DDE를 이용한 통신이다. 기본적으로 IRC Client와 Relay Server는 exe 파일이며, 독립적인 프로그램이므로 통신이 까다롭다. 수많은 방식을 고려하였는데 Socket을 이용한 통신은 중앙 서버의 막대한 부하를 초래할 수 있어 제외되었으며, 파일 입출력을 통한 데이터 통신은 전송 방식과 속도에 있어서 상당히 비합리적인 방식이라 제외되었다. 이에 Dynamic Data Exchange라 불리는 IPC 통신을 적용하였다. 내부 프로세스에서 처리되고, 리소스를 적게 사용하므로 주로 단문을 전송하는 본 프로젝트에서는 가장 적합한 방식이다.

IRC Client와 Relay Server는 각각 DDE 서버와 클라이언트를 1개씩 내장하고 있다.

IRC Client에서 데이터가 전송되면 IRC Client의 DDE Client가 Relay Server의 DDE Server로 데이터를 전송하고 이를 TCP Server로 넘겨 Android로 전송해준다.

Android에서 데이터를 전송할 경우 반대의 순서로 TCP Server가 데이터를 Relay Server의 DDE Client로 넘겨주고 이를 IRC Client의 DDE Server로 전송하여 최종적으로 데이터 교환을 종료하게 된다.

## 인증 및 연동

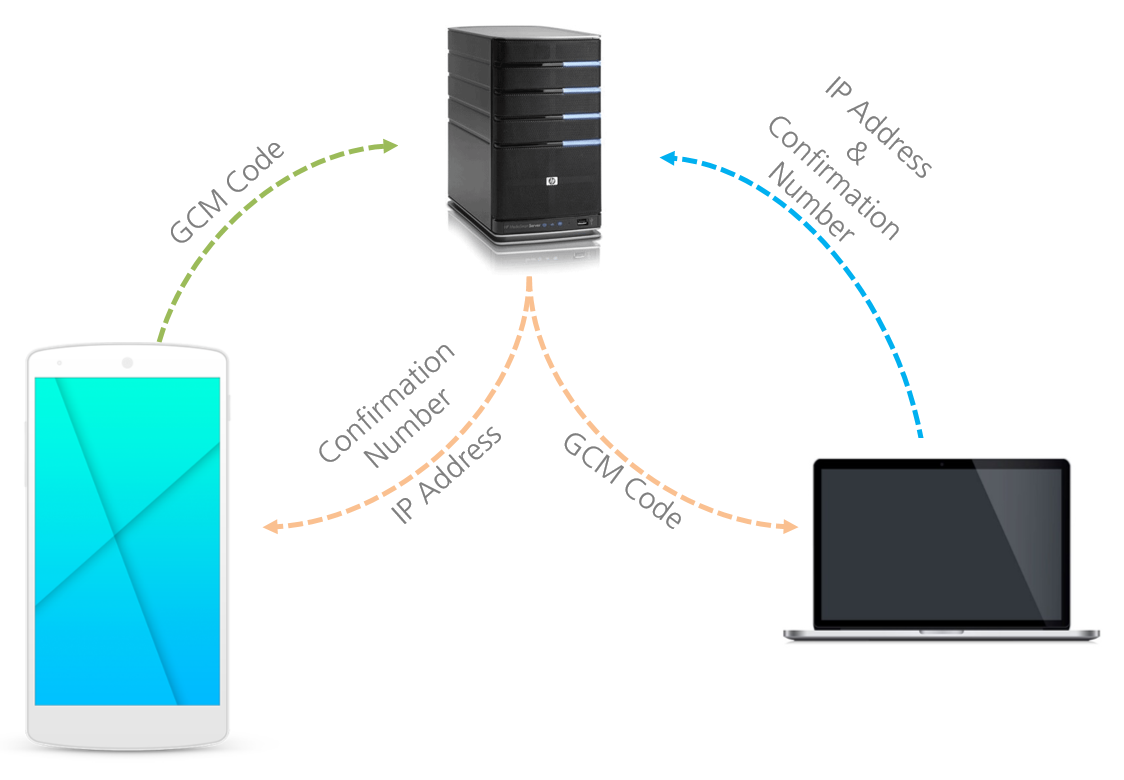


그림 인증 및 연동

- 최초 설치 이후의 시나리오를 가정해본다면 각 어플리케이션은 서로의 정보를 모르는 상태이다. 중앙 서버가 모든 것을 통제하지 않고 각 개인의 어플리케이션이 개별적으로 연동하는 방식을 취하기 때문에 중앙 서버는 최초 연동을 지원해주는 인증 서버 역할을 맡는다.

사용자가 Android Application을 실행하게 되면 Android Application은 중앙 서버로 접속하여 자신의 고유한 GCM 코드를 넘겨준다. 이 코드는 이후 Windows Application에서 중요 메시지에 대한 알림을 Push하기 위한 것이다. 중앙 서버는 이 코드를 DB 테이블에 저장하고 중복되지 않는 난수를 생성하여 DB에 저장함과 동시에 Android로 보내준다. 사용자는 이 출력된 난수를 확인하여 Windows Application의 인증 설정에 입력한다. W Windows Application은 난수 값과 함께 자신의 IP 주소를 중앙 서버로 전송한다. 중앙 서버는 DB 테이블에서 일치하는 난수를 검색하고 해당 난수에 저장된 GCM 코드를 Windows Application으로 보내주는 동시에 받았던 IP를 Android Application으로 보내준다. 이로써 상호간의 데이터 교환이 성립되어 중앙 서버 없이 각 어플리케이션의 통신이 가능해진다.

인증해제의 경우 굳이 중앙 서버를 거치지 않아도 된다. 서로의 정보를 알기 때문에 Windows Application에서는 GCM을 통해서 인증 해제 데이터를 Android Application으로 알리고 Android Application에서는 Windows Application의 IP를 통해 인증 해제를 알린다.

## 3) Windows Application

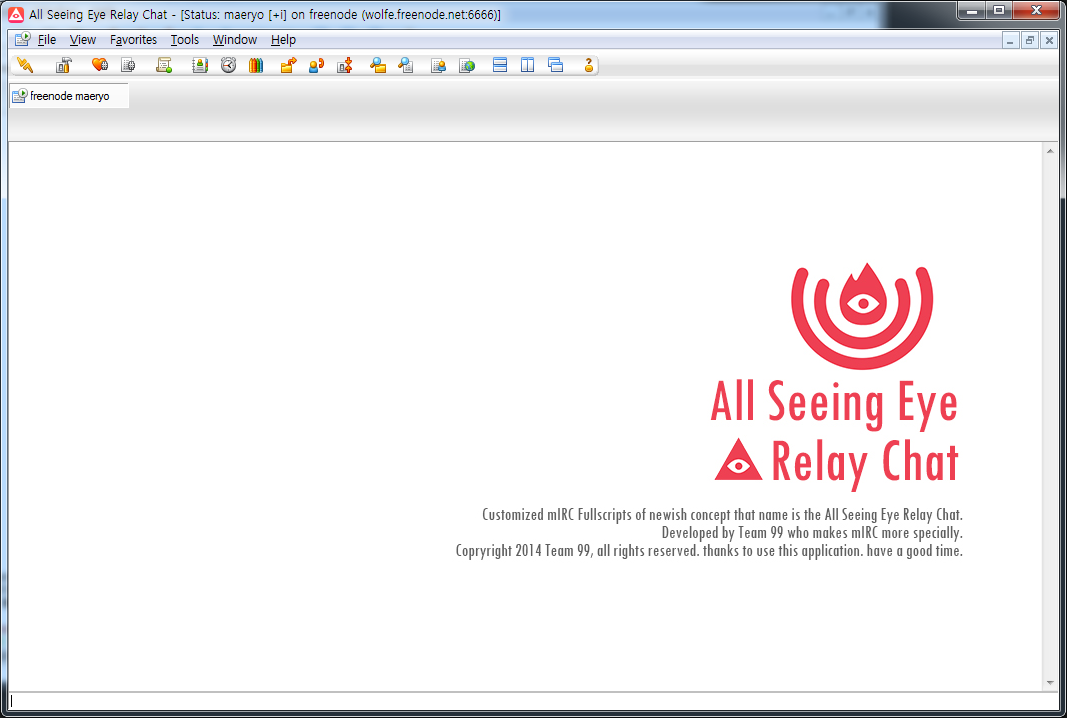


그림 Status Window

- 최초 실행 시 출력되는 윈도우 화면이다. 서버 MOTD 메시지를 포함하여 각종 raw 메시지 numeric에 따른 여러 알림 및 에러 메시지들이 출력되는 창이다.

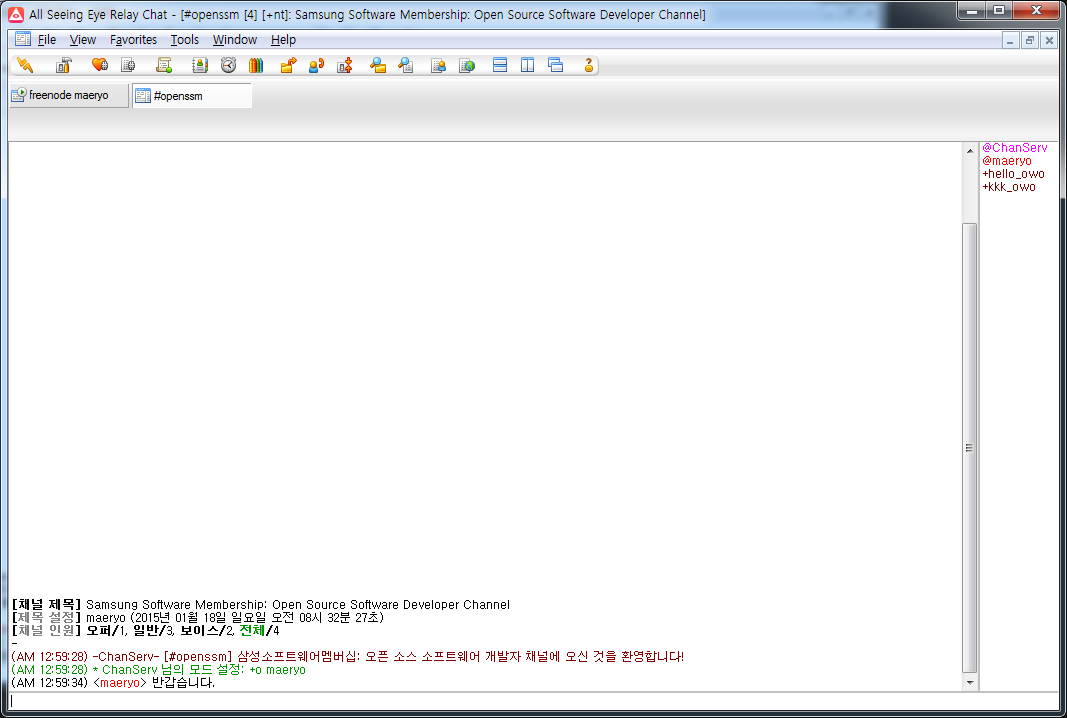


그림 Channel Window

- 채널 접속 화면이다. 기존 mIRC보다 조금 더 직관적이고 간편하게 사용할 수 있도록 초점을맞추었다. 기본적인 대화를 할 수 있는 창이다.

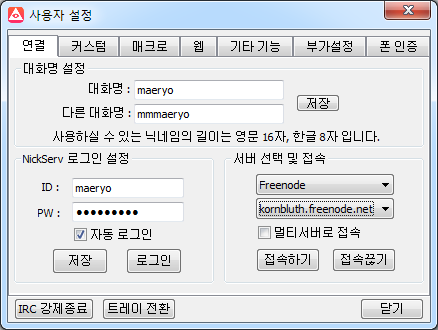


그림 사용자 설정 다이얼로그

- mIRC에 기본적으로 내장된 설정 다이얼로그는 굉장히 다양한 기능을 담고 있으나 일반 사용자에게는 불필요한 기능이 너무나 많으며 사용법이 어렵다는 단점이 있다. 그리하여 대화에 있어서 필수적인 기능들과 편의 기능 위주로 다이얼로그를 새로 제작하였다. 이 다이얼로그에서는 대화명을 설정하고 서버에 접속하는 것부터 스마트 기기와 연동하는 부분까지 꼭 필요한 기능들 위주로 설정이 가능하다.

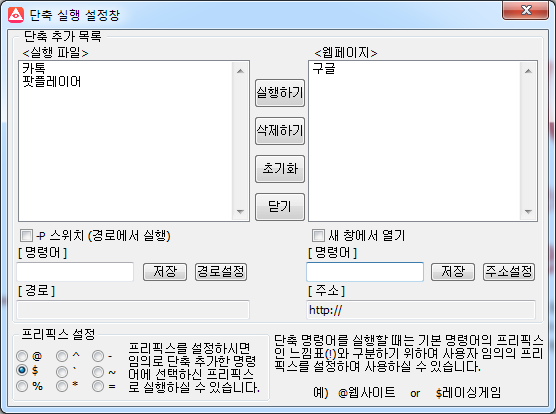


그림 단축 실행 설정 다이얼로그

- 본 프로젝트는 스마트 폰 사용자에 초점을 두었지만 때에 따라 반대의 경우가 생길 수 있다. 기본적으로 IRC는 PC에 최적화되어 있으므로 사용 시간은 오히려 윈도우 클라이언트가 더 많을 수 있다는 점이다. 그렇기에 장시간 이용하는 사용자들의 편의성을 조금 더 고려하였다. 해당 다이얼로그에서는 사용자가 간편하게 프로그램들을 실행하는 명령어를 설정할 수 있다.

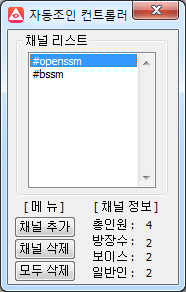


그림 자동조인 컨트롤러

- 프로그램을 실행할 때마다 자주 가는 채널을 다시 가는 것은 상당히 번거로운 일이다.

기본적으로 mIRC 자체에서 자동조인을 지원하지만 Target Network인 Freenode 서버는 동시 접속 가능한 채널이 120개이므로 들어갈 채널이 많을 경우 상당한 시간을 요구한다. 이는 수많은 채널을 빠르게 들어갈 경우 서버의 트래픽을 과도하게 소모하여 Excess Flood로 서버에서 접속을 해제 당할 수 있기 때문이다. 본 프로젝트의 경우는 방식을 변경하여 기존의 속도 문제(빠르게 접속하거나, 느리게 접속하는 방식)에서 벗어나 ircd에서 지원하는 방식대로 한번에 모든 채널을 들어가고 대신 약간의 Excess Flood를 감수하는 방식을 취했다. 또한 접속하고 있는 채널일 경우 간단한 인원 정보를 출력하도록 하였다.

## 4) Android Application

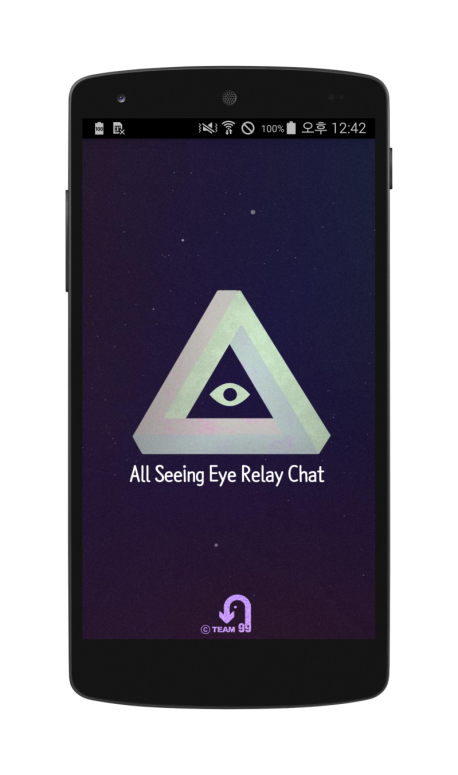


그림 Splash

- Android Application이 실행되면 첫 화면이 나타나면서 Windows Client에서 쌓아놓은 Log를 불러온다. 만약 인증하기 전이라면 Windows Client와 연동을 위한 인증 화면으로 넘어와서 중앙 서버와 통신한다.

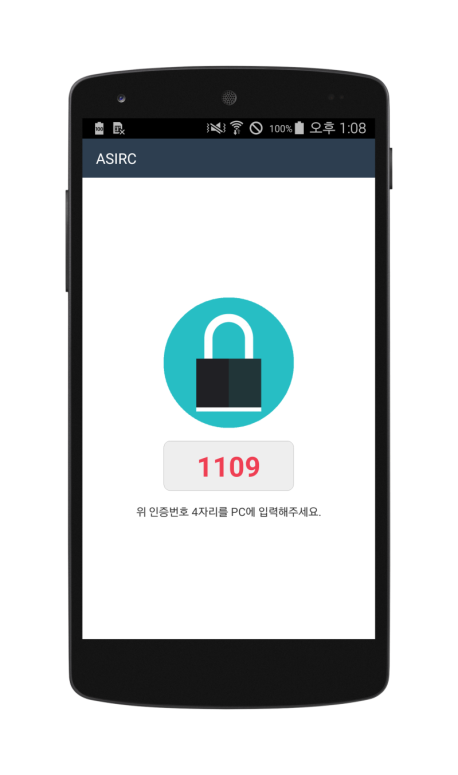


그림 Authorize

- 중앙 서버와 Android Application이 연결되면 중앙 서버로부터 인증 번호를 받고, 화면에 나타난 인증 번호를 이용하여 Windows Application이 인증을 완료하게 되면 각 클라이언트 간의 상호 통신을 시작한다.

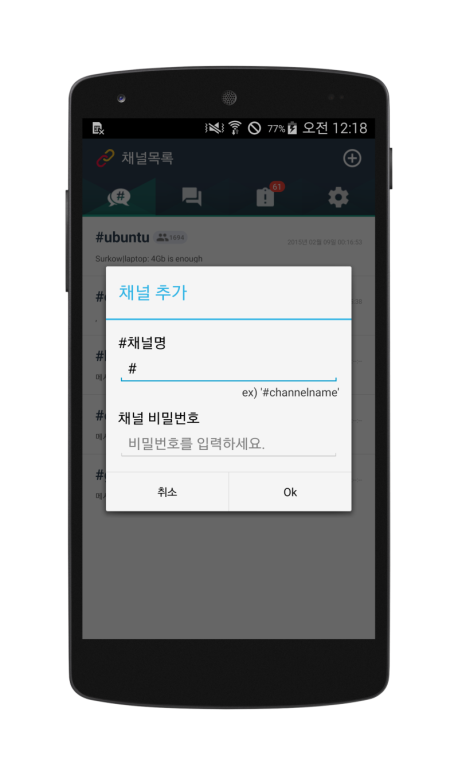
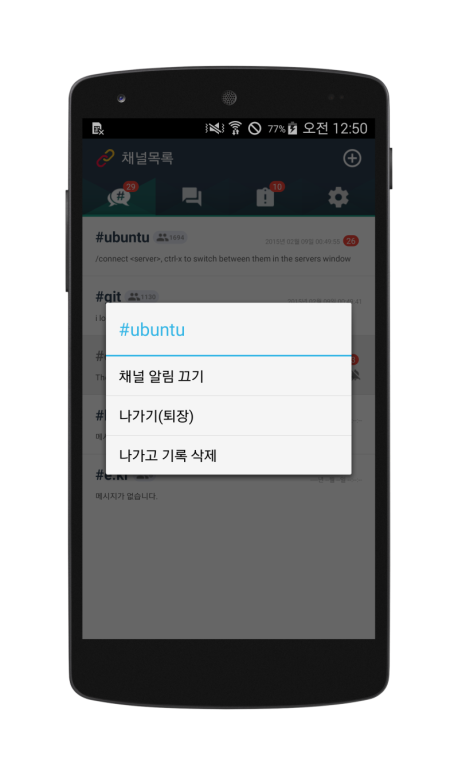
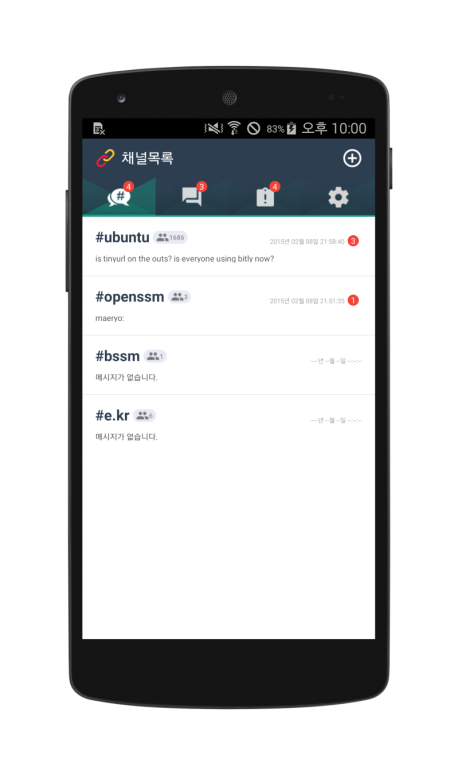


그림 Channel List

- 인증이 된 후에는 Android Application의 메인 화면이 나타난다. 메인 화면은 4개의 탭 화면으로 구성되어있고, 왼쪽부터 채널목록, 대화목록, 알림, 설정 탭이 있다. 먼저 채널목록 탭은 현재 Android Application에서 접속되어있는 목록과 현재 구독 중은 아니지만 기록이 남아있는 채널들을 채널 이름, 현재 접속인원, 가장 최근 대화내용과, 안 읽은 메시지 개수 등을 나타내어 보여준다. 각 리스트 아이템을 클릭 시 해당 채널의 로그를 볼 수 있으며, 길게 클릭할 시 해당 채널에 대한 설정을 할 수 있다. 채널은 ActionBar에 있는 추가 아이콘을 클릭하여 추가할 수 있도록 하였다.

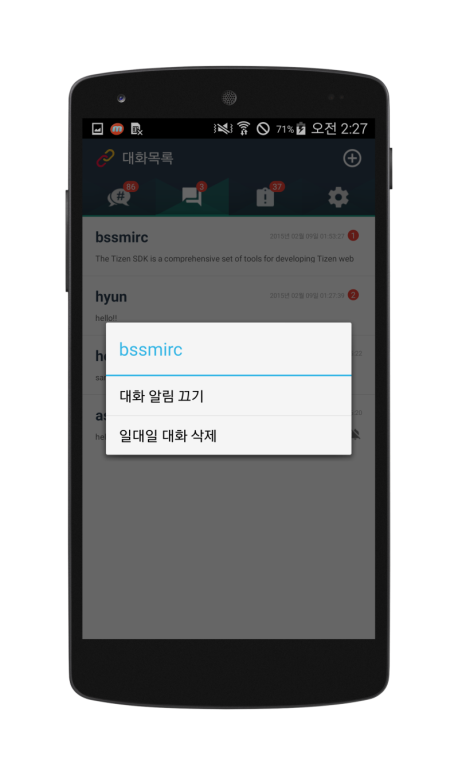
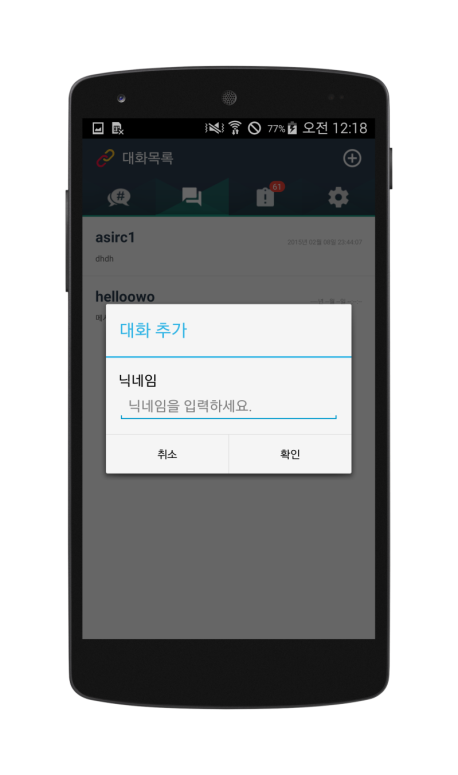
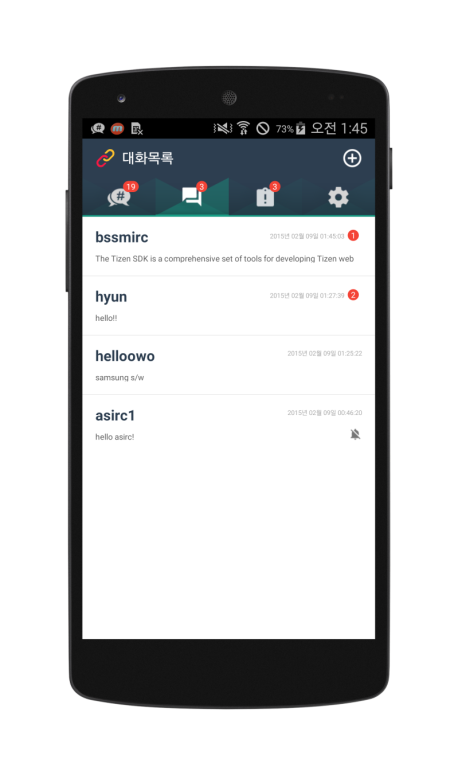


그림 Query List

- 대화목록 탭은 IRC 사용자와 일대일로 주고받은 대화기록을 보여주는 탭이다. 구성은 채널 목록과 비슷하게 했으나, 채널이름 대신 사용자의 이름이 보여진다. 각 리스트 아이템 롱 클릭 시 해당 대화에 대한 설정을 할 수 있으며 일대일 대화 기록을 지울 수 있다. 일대일 대화이기 때문에 채널과는 달리 입장, 퇴장이 없다. 채널목록 탭과 마찬가지로, ActionBar의 추가 아이콘을 클릭하면 사용자의 닉네임을 입력하여 일대일 대화를 추가할 수 있다.

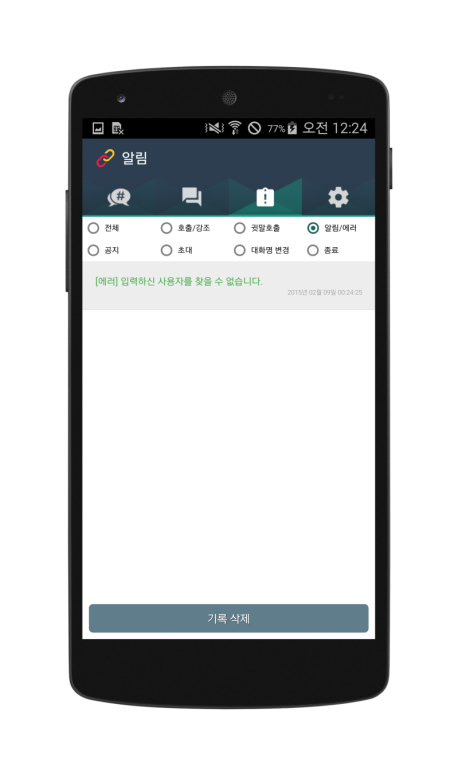
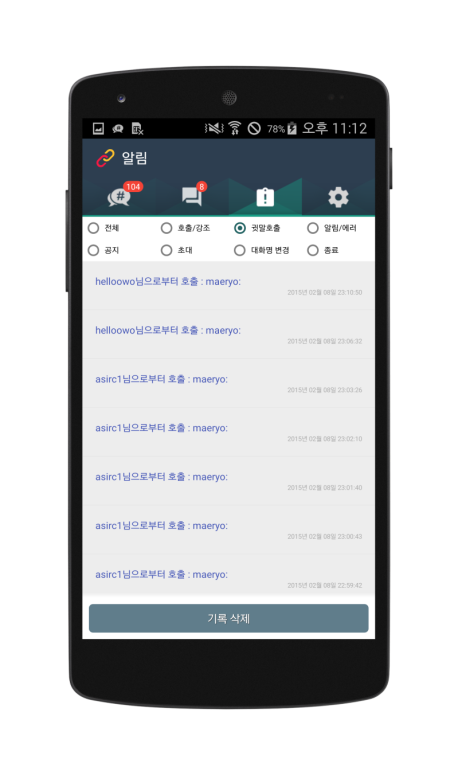
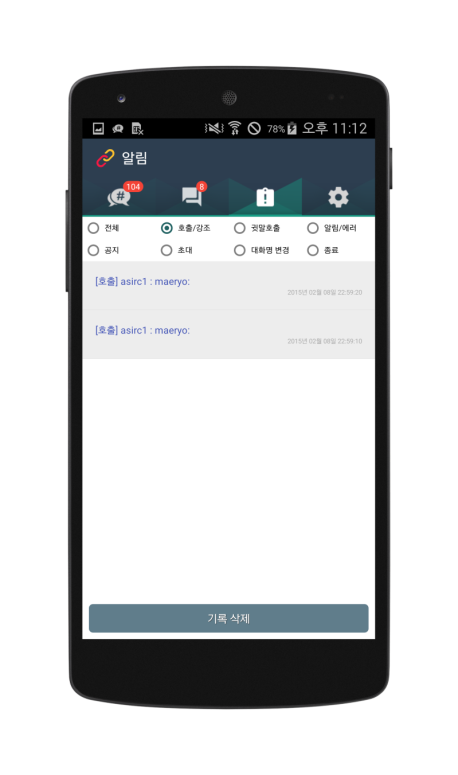


그림 Notice(1)

- IRC를 사용하다 보면, 대화 이외의 다양한 메시지를 받을 수 있다. 이에 호출/강조 메시지가 오더라도 일반 대화와 섞여서 쉽게 알 수 없기 때문에 알림 탭을 두어서 사용자가 새로운 알림을 한눈에 쉽게 볼 수 있도록 하였다. 알림 기록들 중 새로운 메시지는 회색 배경에 텍스트 색상을 타입에 따라 다르게 보여주고, 예전에 확인한 알림 메시지는 흰색배경에 텍스트 색상은 회색으로 사용자가 새로운 메시지를 쉽게 확인 할 수 있도록 하였다. 또한 리스트 위에는 라디오 버튼이 있는데, 라디오 버튼으로 새로운 알림을 유형 별로 쉽게 볼 수 있도록 하였다. 전체는 새로운 알림 메시지를 포함하여 이전에 저장되어있던 기록도 볼 수 있다. 나머지 버튼은 새로운 알림 메시지를 위한 버튼으로, 해당 유형의 새로운 메시지를 보여준다. 호출/강조는 접속해 있는 채널에서 사용자를 호출하거나, 사용자가 지정한 키워드를 포함하고 있는 메시지를 보여주고, 귓속말 호출은 일대일 대화에서의 사용자 호출 메시지를 보여준다. 알림/에러는 사용자가 명령어 수행 시 IRC로부터 날아온 메시지를 보여준다.

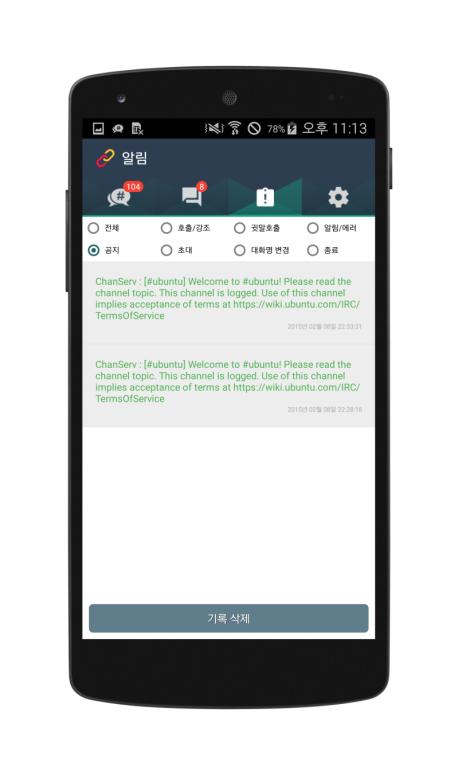
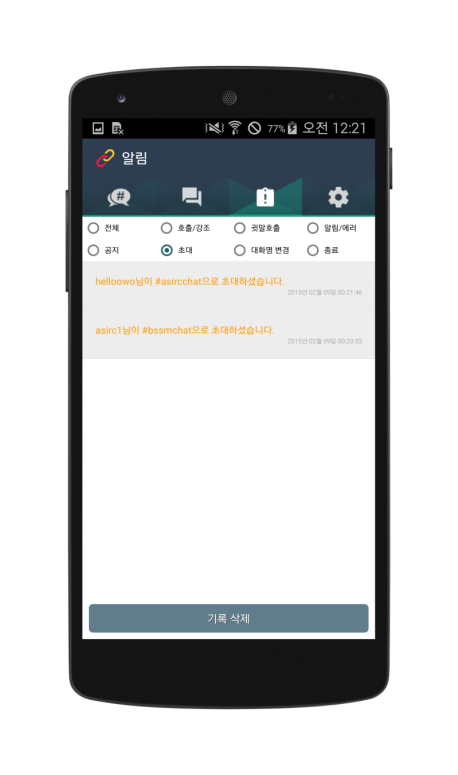
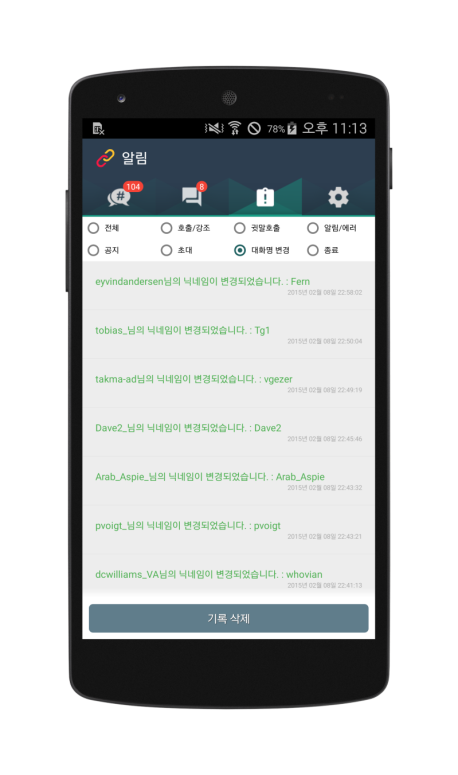


그림 Notice(2)

- 공지는 ChanServ나 NickServ등을 비롯한 봇/사용자들로부터 날아온 Notice 메시지를 보여준다. 초대 메시지는 IRC 사용자가 채널로 초대했을 때 오는 메시지로, 이 알림을 클릭하게 되면 바로 채널에 입장할 수 있도록 하였다. 대화명 변경은 내가 구독중인 채널에 접속해 있는 사람들이 대화명을 변경했을 때의 메시지이고, 종료는 사용자가 IRC사용을 종료하였을 때 메시지이다.

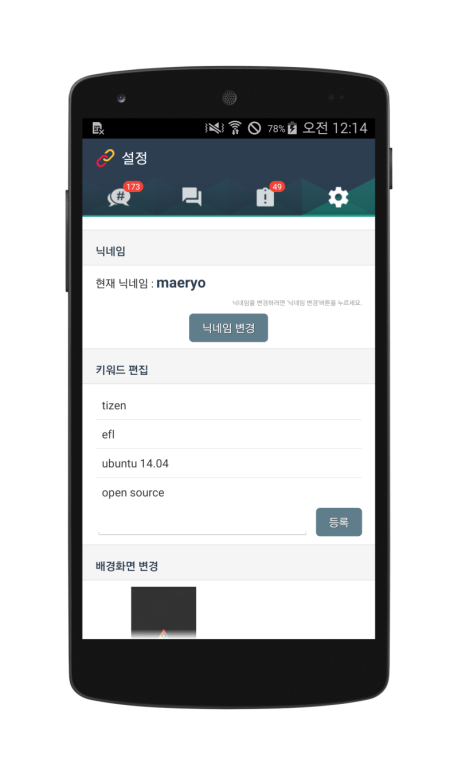


그림 Setting

- 설정 탭은 닉네임 변경, IRC메시지 알림을 받기 위한 키워드 편집, 채팅창 배경화면 변경, PC 연동해제, 모든 데이터 삭제, 버전확인 등을 할 수 있도록 구성하였다.

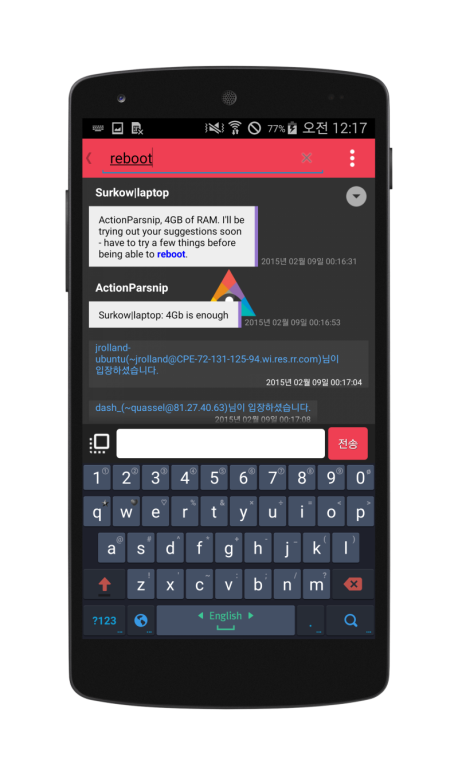
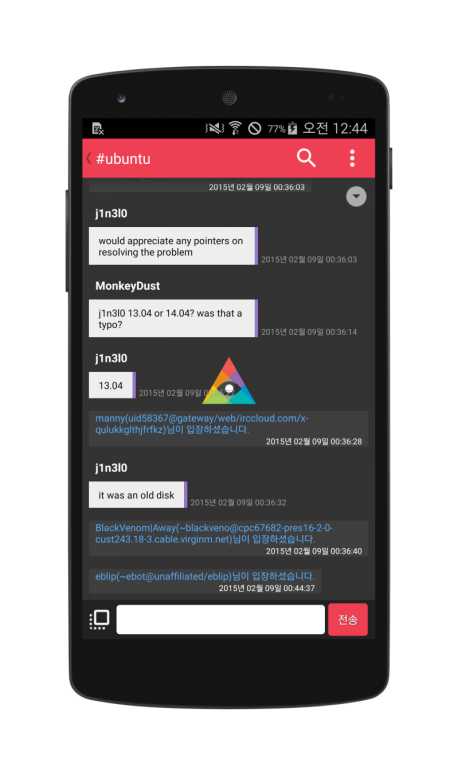
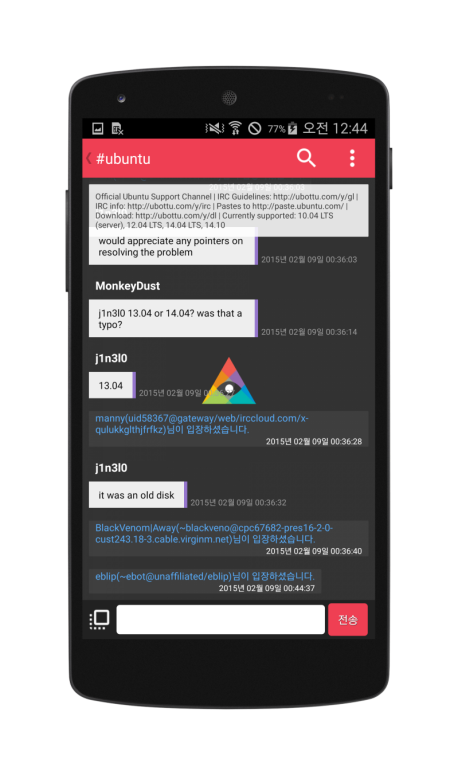


그림 Channel(1)

- 현재 구독중인 채널에 들어가게 되면 그 동안 쌓여있는 로그를 볼 수 있고, 실시간으로 전송되는 메시지 또한 받아볼 수 있다. 또한 해당 채널의 토픽이 있는 경우엔 상단의 공지메시지로 보여진다. 이 공지 메시지는 클릭하여 접어둘 수 있다. 상단의 돋보기 아이콘을 누르면 해당 채널의 대화 검색을 할 수 있고, 검색 결과는 하이라이트 처리되어 보여진다. 계속하여 검색하면 이전 기록까지 검색을 할 수 있고, 만약 검색 결과가 없거나 끝까지 탐색을 다 했을 경우엔 토스트 메시지로 결과 없음을 출력한다.

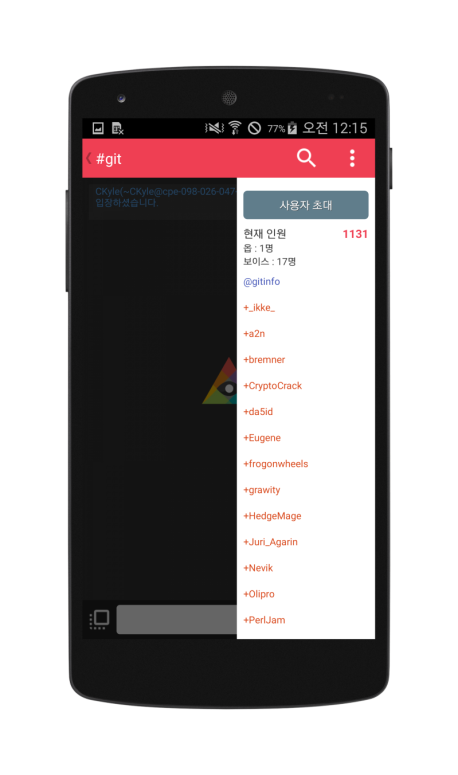
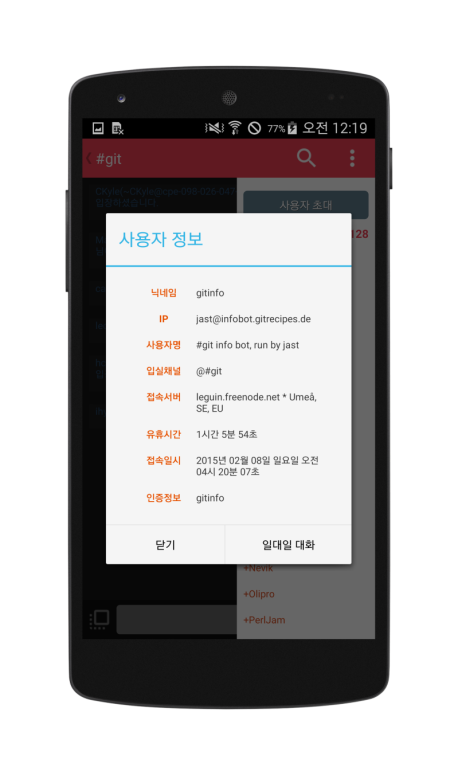


그림 Channel(2)

- 상단 오른쪽에 있는 버튼을 누르면 해당 채널에 접속해 있는 인원을 리스트로 보여주고, 각 닉네임은 사용자의 모드에 따라 다른 색으로 보여준다. 그리고 닉네임을 클릭하면 해당 닉네임의 사용자 정보를 보여 준다. 채널의 경우 일대일 대화와는 달리 재입장이 가능하고, 퇴장을 하게 되면 해당 채널의 기록을 받을 수 없기 때문에 메시지의 부재를 나타내기 위하여 구분자를 출력하도록 하였다.

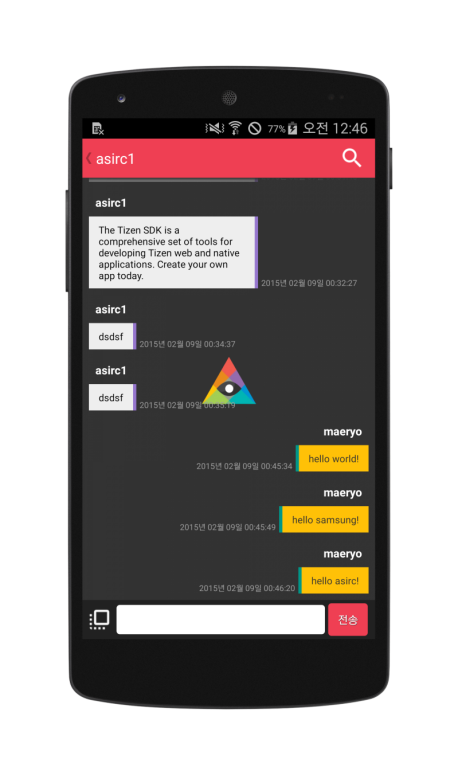
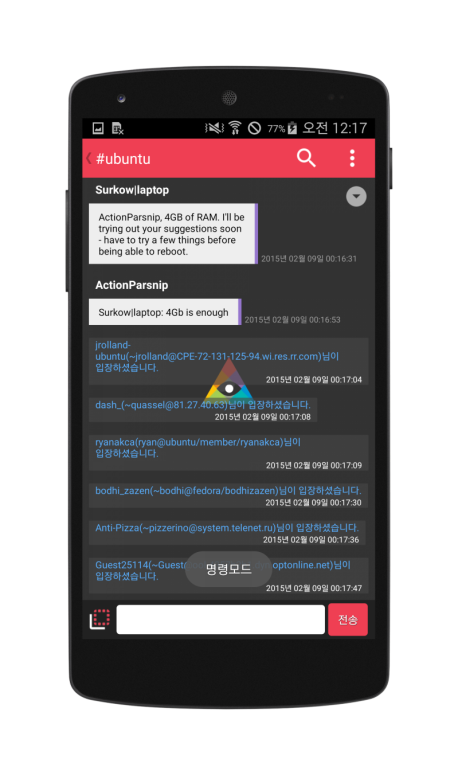
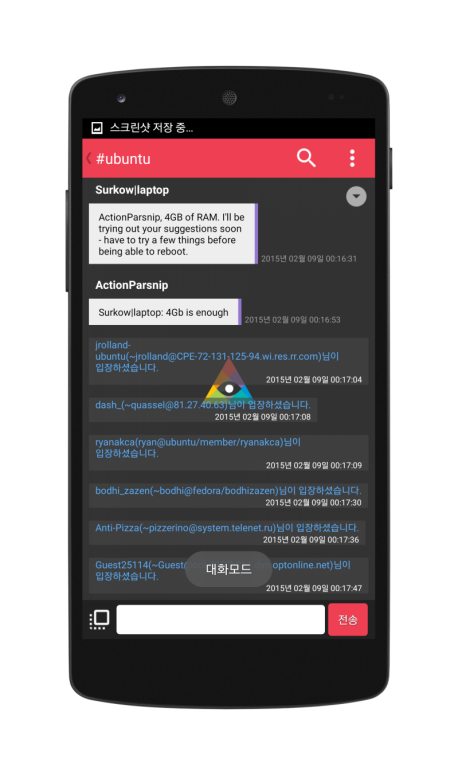


그림 Query

- 일대일 대화창은 채널 대화창에서 토픽, 접속자 목록을 제외한 레이아웃으로 구성되어있다. 그 외 기능은 채널 대화창과 같다. 대화창의 하단에는 메시지를 보낼 수 있는 메시지 박스가 있고, 그 왼쪽에는 메시지의 모드를 바꿀 수 있는 버튼이 있다. 일반적으로 대화모드로 설정되어있고, 버튼을 누르게 되면 IRC명령어를 보낼 수 있는 명령모드로 바뀌게 된다.

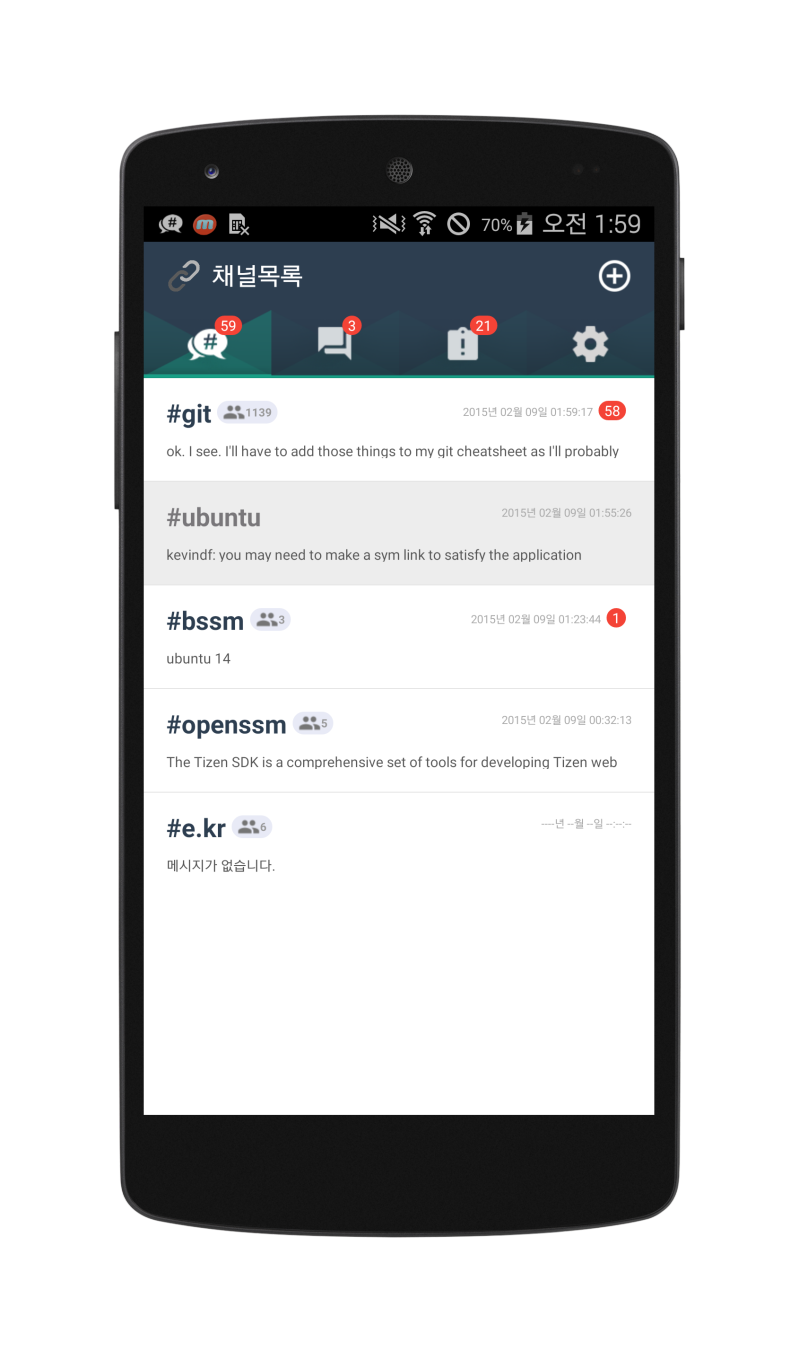


그림 Disconnect/Connect

- PC 어플리케이션과 연결 상태를 나타내 주는 아이콘을 이용하여 사용자 임의대로 연결하고, 끊을 수 있게 하였다.



그림 Notifications

- 해당 채널이나 대화방에 없을 때 호출/강조 메시지가 수신될 경우 Notification으로 사용자에게 보여준다. 이러한 알림은 사용자의 채널 또는 대화 알림 설정으로 제한할 수 있도록 하였다. 그리고 Notification 클릭 시 해당 채널 또는 대화방으로 접속하게 된다.

# 5. 개발 일정

## 1) 개발 일정

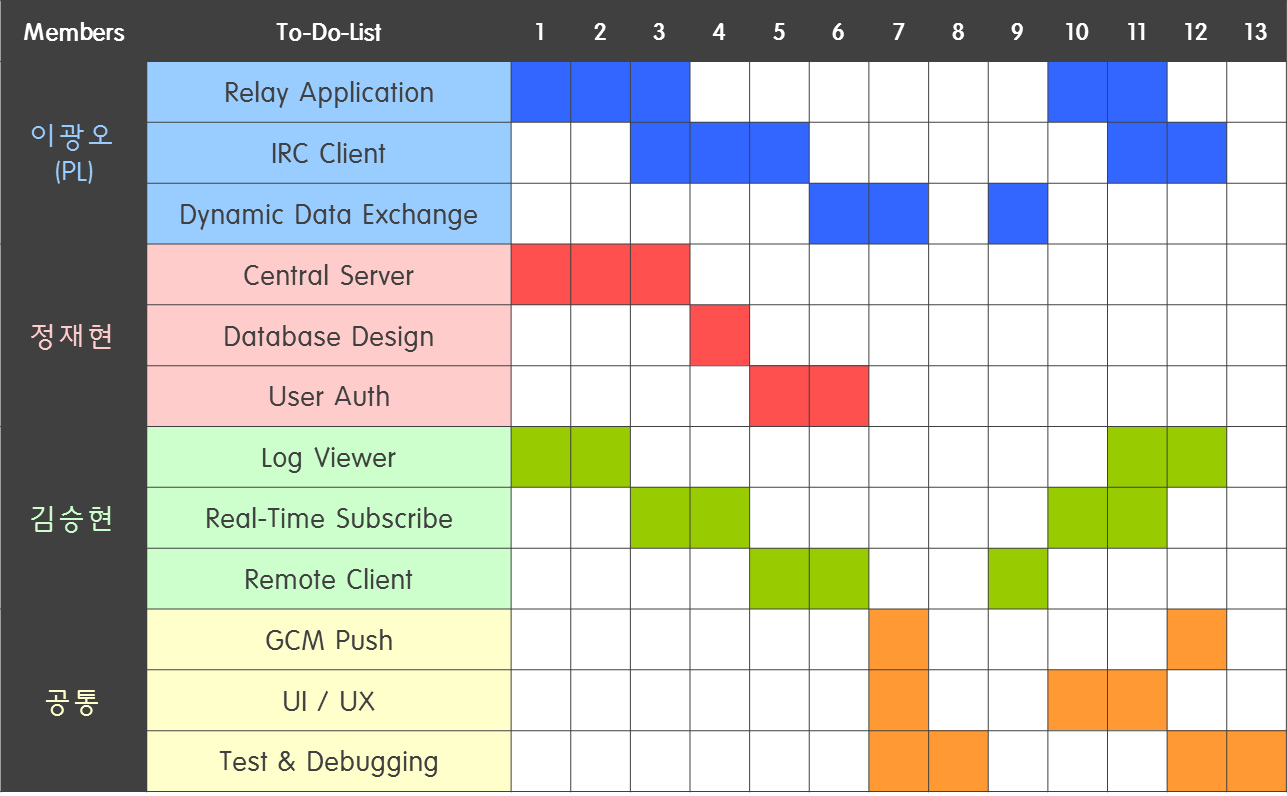


그림 개발 일정

## 2) 역할 분담



그림 역할 분담

# 6. 용어 정리

## 1) IRC

Internet Relay Chat의 줄임말로, 인터넷 실시간 대화를 뜻한다. 핀란드의 Jarkko Oikarinen이 1988년에 개발하였다. 전세계를 실시간으로 연결해 대화(채팅)를 나눌 수 있는 범세계적인 채팅 프로그램으로, 처음에는 스칸디나비아반도를 중심으로 사용되다가 지금은 세계 전역에서 사용되고 있다. 일반 PC통신과 비슷하지만, PC통신은 개인적인 성격인 강한 반면, IRC는 IRC 클라이언트 프로그램이나 IRC 클라이언트를 제공하는 서버에 접속하기만 하면 시간이나 공간에 구애 받지 않고 전세계의 어떤 사람과도 대화가 가능하다. 동시에 다중 대화가 가능한 채팅 프로그램이다. 가장 큰 특징은 다른 채팅 프로그램과 달리, 서버들이 가까운 서버들끼리 직접·간접으로 연결되어 있어 IRC 서버 가운데 어느 한 서버에 연결하기만 하면 자동적으로 전세계 서버와 연결된다는 점이다. 또 수많은 사용자들을 위해 별도로 대화를 할 수 있는 다양한 채널을 가지고 있고, 관심 분야에 따라 이 채널에서 저 채널로 옮겨 다닐 수도 있다. 채널은 공개채널·개인채널·비밀채널로 구분되며, 채널 고유의 이름이 없을 때는 자동적으로 새 채널 이름이 생성된다. 대화는 대부분 영어로 이루어지지만, 다른 언어를 사용할 수도 있다.

## 2) mIRC Scripting Languages

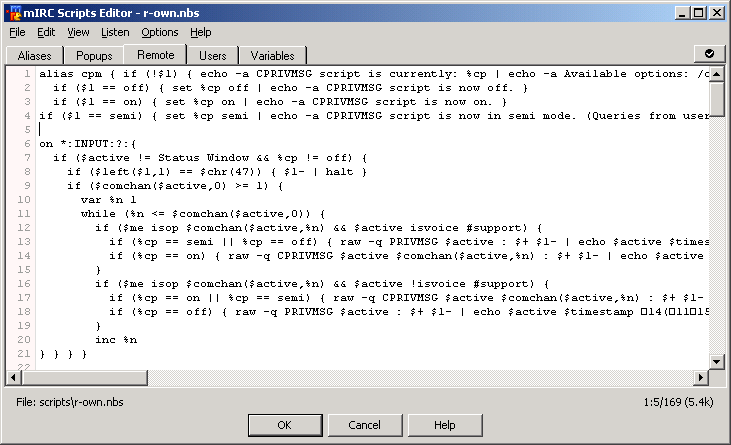


그림 mIRC Scripting Languages

줄여서 mSL이라고도 한다. mSL은 mIRC 기반 프로그램 안에서 어떻게 동작하도록 제어할 것인지 작성하는 프로그래밍 언어이다. mIRC 스크립트는 적게는 사용자의 편의를 위한 줄임글 명령 정의로부터, 자동 채널 관리, 게임, 재미를 위한 기능을 비롯해 크게는 거대한 응용 프로그램을 만드는 데까지 여러 용도로 사용된다. mIRC 사용자들 사이에서 관련된 포럼이 몇몇 형성되어 있다. mIRC 스크립트에서 기본으로 지원하는 명령 및 식별자는 IRC 제어로부터 계산, GUI까지 다양하다. 스크립트만으로 처리하기 힘들거나 느린 경우에는 외부의 DLL 파일이나 COM 객체를 사용할 수 있다.

## 3) GCM(Google Cloud Messaging)



그림 Google Cloud Messaging

GCM은 Google Cloud Messaging의 약자로써 Google에서 제공하는 메시지 발송 서비스이다. GCM은 사용자 개개인의 디바이스 내 어플리케이션 단위로 Targeting을 하여 메시지를 전송할 수 있는 서비스이다. 해당 서비스를 이용하기 위해선 최소 버전이 Android 2.2 Froyo 이상이 되어야 한다.

## 4) DDE

DDE(Dynamic Data Exchange)는 어플리케이션간의 데이터 전송을 위해 사용하는 프로토콜이다. DDE 프로토콜은 메시지와 지침들의 집합이다. 이것은 데이터 교환을 위해 데이터를 공유하고, 공유된 메모리를 사용하는 어플리케이션간에 메시지를 전송한다. 어플리케이션은 데이터 전송과 연속적인 교환을 위해 DDE 프로토콜을 사용할 수 있다.

DDE는 공통 교환 영역으로 공유 메모리를 사용하고, 프로토콜 또는 명령어와 메시지 형식의 집합을 제공하는 IPC(Interprocess Communication)이다. DDE는 어플리케이션이 요청한 데이터는 클라이언트로 간주되고, 어플리케이션이 제공하는 데이터는 서버로 간주되는 클라이언트/서버 모델을 사용한다.

DDE는 서비스, 토픽, 아이템이라는 세 가지의 계층적인 이름을 사용한다. DDE의 클라이언트와 서버간의 대화는 서비스와 토픽을 한 쌍으로 사용해서 확립된다. 이것은 채널 쌍이라고 이름 지을 수 있다. 이것은 대략 전화번호와 동등하다. 이름의 아이템 부분은 특정 데이터를 확인하거나 대화가 연결된 클라이언트에 의해 요청된 명령어를 확인하는데 사용된다.

대화를 확립하기 원하는 DDE 클라이언트는 연결하기를 원하는 서비스/토픽의 이름 쌍(channel)을 명세 한다. 윈도우는 모든 최상위 레벨의 윈도우에 요청을 전송 한다. 첫 번째 서버는 클라이언트에 연결되고, 대화가 확립된다.

클라이언트는 토픽 또는 서비스 이름이 명세 되지 않았거나 둘 다 명세 되지 않았다면 떠날 수도 있다. 예를 들면, 만약 토픽 이름이 명세 되지 않았다면 대화는 하나의 서버에 의해 지원되는 모든 토픽상에 확립될 수 있다. 만약 둘 다 명세 되지 않았다면 대화는 모든 서버들의 모든 토픽상에 확립될 수 있다.

# 6. 참고 문헌

- 국가지정 컴퓨터연구정보센터 : <http://www.cseric.or.kr/new_cseric/main.asp>

- RFC 1459(IRC: Protocol) : <http://tools.ietf.org/html/rfc1459>

- RFC 2810(IRC: Architecture) : <http://tools.ietf.org/html/rfc2810>

- RFC 2811(IRC: Channel Management) : <http://tools.ietf.org/html/rfc2811>

- RFC 2812(IRC: Client Protocol) : <http://tools.ietf.org/html/rfc2812>

- RFC 2813(IRC: Server Protocol) : <http://tools.ietf.org/html/rfc2813>