

# 2019년도 1학기 네트워크 프로그래밍 최종보고서

2019. 06. 21

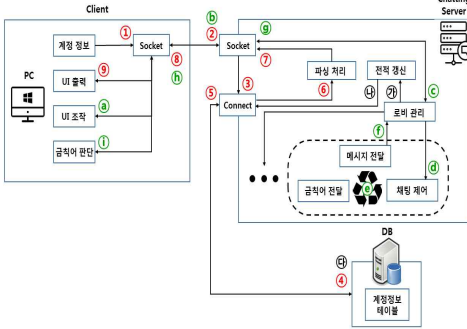
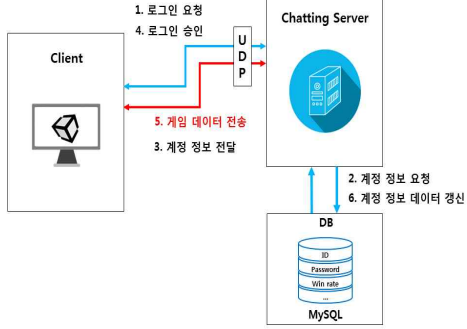
소 속	송실대학교 IT대학 컴퓨터학부
과 목	네트워크 프로그래밍
담 당 교 수	최 종 선
프로젝트 명	Talkative Game (T.G)

Talkative Game				
채팅 서버에서 동작하는 Unity기반 멀티 플레이 금칙어 게임				
조편성	3조			
팀 명	봉천주민들			
팀 원 (총: 3명)	이름	학번	개인별 역할	팀내 기여도
	김한재	20132334	팀장, 서버 개발	40 %
	함승우	20132478	DB 개발, 클라이언트 개발	40 %
	황용훈	20132483	클라이언트 개발	20 %
팀 구성	<div><div>김한재 (Team Leader)</div><div><div>김한재</div><div>서버 개발</div><div>내용<ul style="list-style-type: none"><li>- Socket 통신 개발</li><li>- 로비 관리 모듈 개발</li><li>- 메시지 전달 모듈 개발</li><li>- 채팅 제어 모듈 개발</li><li>- 금칙어 전달 모듈 개발</li></ul></div></div><div><div>함승우</div><div>DB, 클라이언트 개발</div><div>내용<ul style="list-style-type: none"><li>- 전적 갱신 모듈 개발</li><li>- 파싱 처리 모듈 개발</li><li>- Connect 모듈 개발</li><li>- Socket 통신 개발</li><li>- 계정 정보 모듈 개발</li><li>- 금칙어 판단 모듈 개발</li></ul></div></div><div><div>황용훈</div><div>클라이언트 개발</div><div>내용<ul style="list-style-type: none"><li>- UI 출력 모듈 개발</li><li>- UI 조작 모듈 개발</li></ul></div></div></div>			

## 〈 개발계획서 요약 〉

(조) 팀 명		( 3조 ) 봉천주민들		
프로젝트 명		T.G (Talkative Game) 채팅 서버에서 동작하는 Unity기반 멀티 플레이 금칙어 게임		
배경 및 당위성		기존 금칙어 게임은 그 유래가 다른 게임의 에디터 툴(Editor Tool)을 기반으로 제작된 게임인 점에서 해당 플랫폼을 설치해야 하는 문제점이 따른다. 따라서 독립적인 금칙어 게임만을 위한 채팅 서버와 클라이언트 환경을 구현함으로써 해당 문제를 해결하고 접근성을 높인 게임을 만들어 보기로 하였다.		
제안 내용	최종 목표	본 프로젝트에서는 채팅 서버 구현을 통해 다른 플랫폼에 독립적이며 접근성이 높은 Unity 기반 멀티플레이 금칙어 게임을 개발한다.		
	시스템 개요	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>&lt;그림 1&gt; 시스템 개요</p> </div> <div style="text-align: center;"> <p>&lt;그림 2&gt; 통신 개요</p> </div> </div> <p>클라이언트에서 게임 진행과 관련된 요청은 채팅 서버에 소켓 통신으로 요청한다. 계정 정보(ID 및 전적 정보)와 같이 DB에 접근할 필요가 있을 경우 채팅 서버에서 통신한다.</p> <ul style="list-style-type: none"> <li>● 서버 <ul style="list-style-type: none"> <li>- 채팅 서버와 DB 서버 총 2개의 서버 형태 구성</li> <li>- 채팅 서버와 클라이언트와의 통신 방법은 UDP/IP 통신</li> <li>- 채팅 서버는 DB와 직접 연결</li> </ul> </li> <li>● 클라이언트 <ul style="list-style-type: none"> <li>- 채팅 서버를 경유하여 로그인</li> <li>- 메시지를 주고받는 채팅 프로그램을 통해 게임을 진행</li> <li>- 금칙어 판단은 클라이언트에서 수행하여 채팅 서버로 전달</li> </ul> </li> </ul>		
	개발 방법	<ul style="list-style-type: none"> <li>● 서버 <ul style="list-style-type: none"> <li>- C와 Unix Socket 통신을 이용한 채팅 서버 구현 및 MySQL을 이용한 DB 구축</li> <li>- Multi-Thread를 이용한 독립적인 방을 제공하고 UDP/IP 기반의 단말 간 통신 구현</li> <li>- Sever는 DB와 SQL Connection을 사용한 직접 연결 구현</li> </ul> </li> <li>● 클라이언트 <ul style="list-style-type: none"> <li>- Socket 통신을 이용한 로그인 정보 전달 및 서버 접속 구현</li> <li>- Unity GUI(NGUI, EZGUI)를 이용한 Windows OS 기반의 채팅 게임 구현</li> <li>- 금칙어 Logic은 C#을 이용한 구현(규칙 판단 기능, 금칙어 전달 기능)</li> </ul> </li> </ul>		
기대효과		<p>학습적 측면</p> <ul style="list-style-type: none"> <li>- C, Unity를 이용한 코딩 기술을 발전시킬 수 있다.</li> <li>- Linux Socket 통신을 이용한 소켓 프로그래밍을 할 수 있다.</li> </ul> <p>실용적 측면</p> <ul style="list-style-type: none"> <li>- 서로의 말 습관을 파악하며 긴장감 있는 심리게임을 기대할 수 있다.</li> <li>- 해당 게임의 접근성을 늘림에 따라서 여러 가지 메신저(카카오톡, 텔레그램)등의 미니게임으로 탑재되는 것을 기대할 수 있다. (단체방의 아이스 브레이킹 활용)</li> </ul>		
중심어		금칙어 게임	채팅서버	Unity
		Multi-User	Reliable-UDP	MySQL

## 〈 개발계획서 세부 〉

(조) 팀 명	( 3조 ) 봉천주민들
제목	T.G (Talkative Game) 채팅 서버에서 동작하는 Unity기반 멀티 플레이 금칙어 게임
시스템 구성도	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>&lt;그림 3&gt; 시스템 구성도</p> </div> <div style="text-align: center;">  <p>&lt;그림 4&gt; 통신 구성도</p> </div> </div>
주요 SW 모듈	<p>○ 서버</p> <ul style="list-style-type: none"> <li>- Socket 통신 : 채팅 서버와 Client간의 통신</li> <li>- 로비 관리 : Client 요청에 채팅 방(Thread) 생성 및 관리</li> <li>- 메시지 전달 : 같은 방에 있는 Client들에게 메시지를 전달</li> <li>- 금칙어 전달 : 전체 Client로부터 받은 금칙어 목록을 지정한 Client에게 전달</li> <li>- 채팅 제어 : 금칙어 지정 단계에서 채팅 활성화/비활성 제어</li> <li>- 전적 갱신 : 게임 종료시 DB에 최신 정보 전달</li> <li>- 파싱 처리 : DB 테이블 정보 파싱</li> <li>- Connect : SQL Connection을 이용한 DB 테이블 접근</li> </ul> <p>○ 클라이언트</p> <ul style="list-style-type: none"> <li>- Socket 통신 : Client와 채팅 서버간의 통신</li> <li>- 계정 정보 : 회원가입 및 로그인시 필요한 정보 전달</li> <li>- UI 출력 : 로비 상태, 채팅 진행 상황 및 접속자 정보 출력</li> <li>- UI 조작 : 방 생성, 입장, 게임 시작 정보 조작</li> <li>- 금칙어 판단 : Client가 배정 받은 금칙어 입력 여부 판단</li> </ul> <p>○ DB 테이블</p> <ul style="list-style-type: none"> <li>- 계정 정보 테이블 : 계정 정보(ID, Password, 전적 등)을 저장하고 관리</li> </ul>
위험요소	<ul style="list-style-type: none"> <li>- 채팅 과정에서 ASCII Code 한글 인코딩 문제</li> <li>- 지원 불가능 문자의 누락</li> <li>- 멀티 쓰레드 사용 시 동기화 문제</li> <li>- 예외 상황(강제 종료, 연결 끊김 등) 발생 시 처리 문제</li> </ul>
위험요소의 해결 방법	<ul style="list-style-type: none"> <li>- EUC-KR를 사용하여 한글 인코딩 진행</li> <li>- 타 문자로의 대체</li> <li>- Mutex를 이용한 동기화 처리</li> <li>- SIGPIPE, SIGUSR을 이용한 시그널 핸들링</li> </ul>

## 목 차

1. 프로젝트 개요 .....	1
1-1. 개발 기술의 개요 .....	1
1-2. 개발 기술의 필요성 .....	1
1-3. 개발 기술의 예상효과 및 필요방안 .....	1
2. 관련기술 현황 .....	2
2-1. 관련 기술 .....	2
2-2. 요구 분석 .....	2
3. 수행내용 및 결과 .....	4
3-1. 개발 목표 .....	4
3-2. 개발내용 .....	4
4. 장애요소 및 해결방법 .....	12
5. 개발 후기 .....	13

## <그림 목차>

[그림 1] ‘Talkative Game’ 개요 .....	1
[그림 2] 메시지 프로토콜 Header .....	2
[그림 3] 메시지 프로토콜 .....	3
[그림 4] ‘Talkative Game’ 시스템 구성도 .....	5
[그림 5] 로비 관리 실행 과정 .....	6
[그림 6] 로비에 만들어진 채팅방 NewRoom1 .....	7
[그림 7] New Room1에 접속했을 때 서버로 오는 메시지 .....	7
[그림 8] 계정 정보 생성 과정 .....	8
[그림 9] 계정 정보 생성 화면 .....	8
[그림 10] 채팅방 화면 .....	9
[그림 11] network 테이블 계정 정보 .....	10

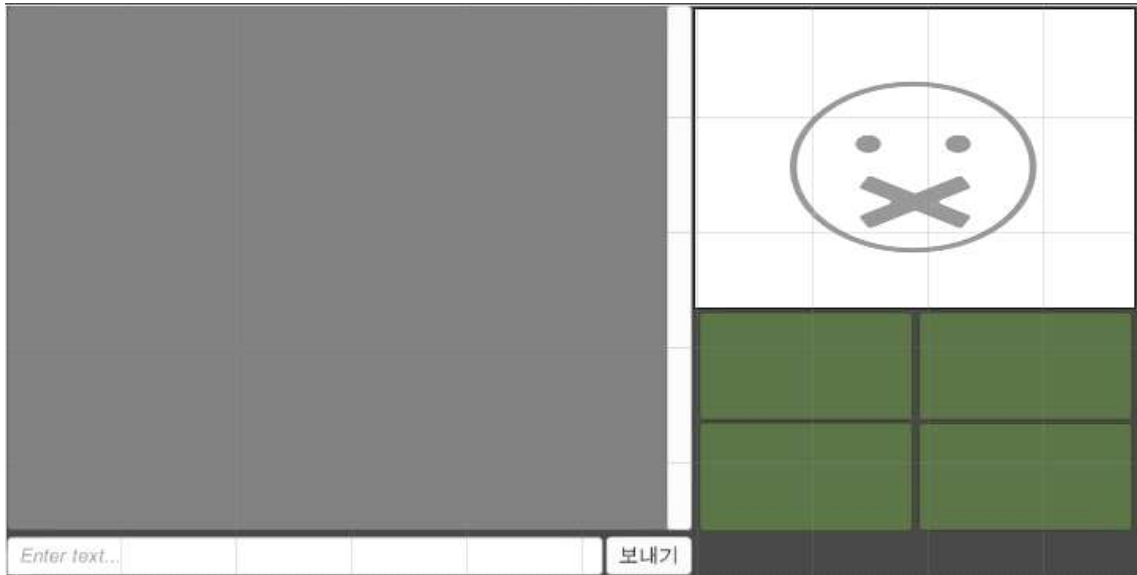
## <표 목차>

[표 1] 요구 분석 .....	1
[표 2] 연구개발 내용 및 성과내용 .....	4
[표 3] 개발 방법에 따른 개발 결과 .....	11
[표 4] 장애요소 및 해결방법 .....	12
[표 5] 에로사항 .....	13
[표 6] 개발후기 .....	14

## 1. 프로젝트의 개요

- 본 프로젝트에서는 Socket 통신을 활용한 멀티 플레이 금칙어 게임 ‘Talkative Game’을 제작

### 1-1 개발 기술의 개요



[그림 1] ‘Talkative Game’ 개요

- ‘Talkative Game’은 스타크래프트의 ‘금칙어 게임’을 모티브로 함
- 사용자 간 지정한 금칙어를 말하면 패배하는 게임
- 하나의 채팅 룸에서 최대 4인 플레이 가능

### 1-2 개발 기술의 필요성

- 해당 게임의 접근성 향상을 위해 플랫폼 독립
- 플랫폼 통합 엔진 Unity를 이용한 게임 구동
- 기존 플랫폼의 문제 발생 시 해결 불가

### 1-3 개발 기술의 예상효과 및 활용방안

- 신규 플레이어 유입 기대 및 새로운 채팅 프로그램으로 이식 가능성 기대 ex) 카카오톡
- 접속 환경 다양화
- 문제 발생 시 직접 유지 보수 가능

## 2. 관련 기술 현황

### 2-1 관련 기술

#### 2-1-1. TCP 기반 채팅 서버

- 신뢰성 있는 데이터 전송
- 연결의 지속성 필요시 사용
- 지연 상황이 발생되지 않을 때 사용

#### 2-1-2. UDP 기반 채팅 서버

- 신속한 데이터 전송
- 패킷 손실 간헐적 발생
- 다중 클라이언트 구현 부적합

#### 2-1-3. 금칙어 게임

- 불특정 다수의 플레이어와 게임
- 다양한 사망 이펙트 존재
- 랭킹 시스템 제공

### 2-2 요구 분석

- [표 1] 참조

[표 1] 요구 분석

	TCP 기반 채팅 서버	UDP 기반 채팅 서버	금칙어 게임	Talkative Game
신뢰성	보장	없음	있음	있음
전송 지연	간혹 발생	없음	없음	없음
인원수 제한	멀티 플레이	싱글 플레이	멀티 플레이	멀티 플레이
프로토콜	TCP	UDP	Reliable UDP	Reliable UDP
플랫폼	클라이언트 종속	클라이언트 종속	Windows OS	다양한 환경에서 플레이 가능



## 2-2-1. Talkative Game

### ○ Server

- 빠른 데이터 전송을 위한 UDP 사용
- 신뢰성 확보를 위하여 Reliable-UDP 구성
- 멀티 플레이 구현을 위한 Multi-Thread 형태의 Server 구현

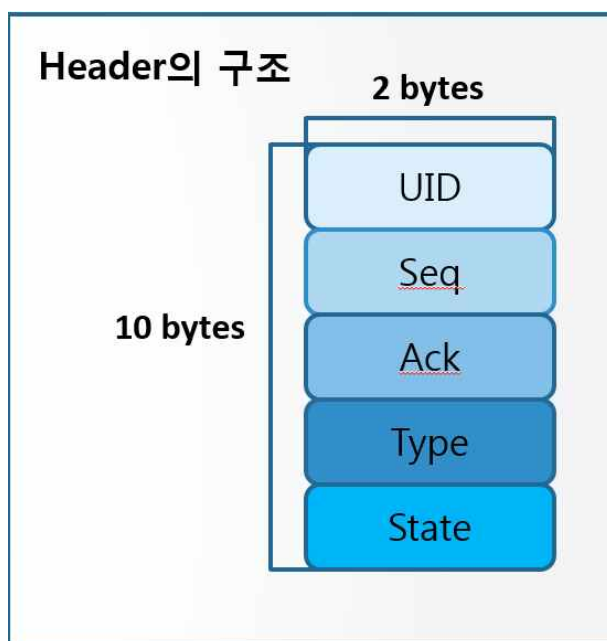
### ○ Client

- Unity를 이용하여 다양한 환경에서 게임 진행
- UI 및 시각 효과 제공 ex) 사망 이펙트
- 서버 부하 방지를 위한 Client 내 금칙어 판단 Logic 구현

### ○ DB

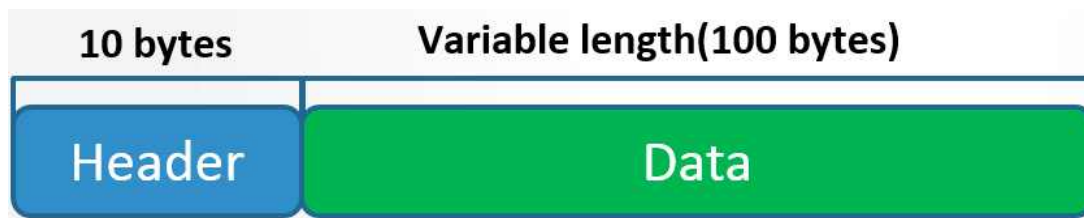
- 보안을 위한 DB와 Server간 직접 통신

### ○ 메시지 프로토콜



[그림 2] 메시지 프로토콜 Header

- UID : User ID
- Seq : Packet의 번호
- Ack : 응답문자
- Type : 값에 따라 Client 또는 Server가 실행해야할 함수 ID
- State : 게임 진행 중 Client 상태
- Win : 계정 정보 갱신 요청(state에 포함하여 변경됨)



[그림 3] 메시지 프로토콜

- 특징상 동일한 패턴의 메시지 송수신으로 하나의 프로토콜 사용
- Header 정보를 이용하여 Reliable-UDP 구현

### 3. 수행내용 및 결과

#### 3-1 개발 목표

본 프로젝트에서는 채팅 서버 구현을 통해 다른 플랫폼에 독립적이며 접근성이 높은 Unity 기반 멀티플레이 금치어 게임을 개발한다.

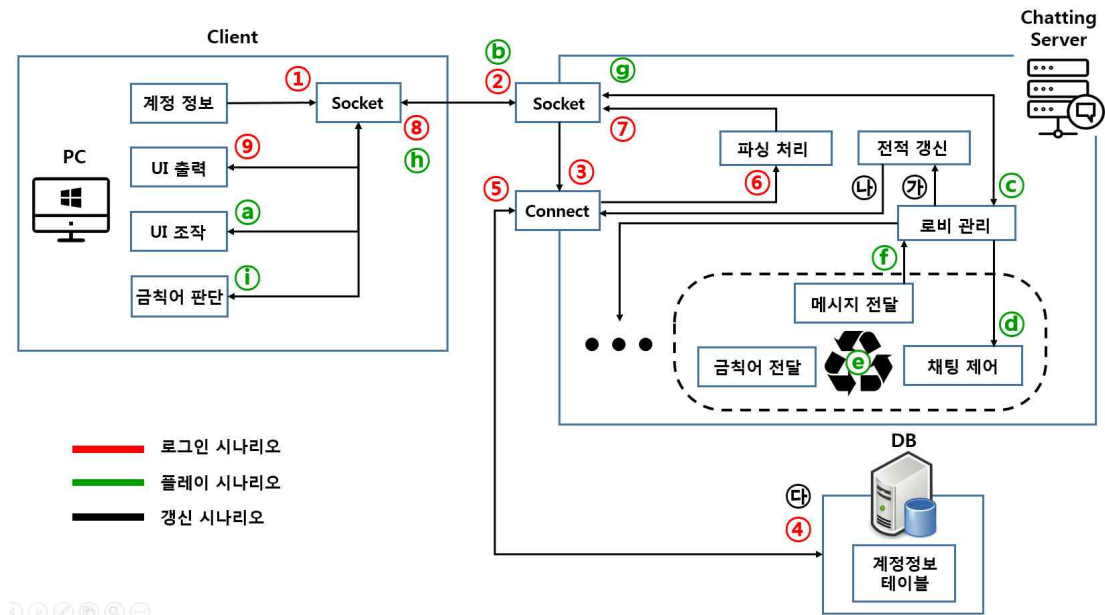
#### 3-2 개발내용

[표 2] 연구개발 내용 및 성과내용

개발목적	연구개발 내용	성과내용
<b>Server</b>	<ul style="list-style-type: none"> <li>Client와 Reliable UDP Socket 통신</li> <li>Multi-Thread를 이용한 다수의 채팅 방 생성 및 관리</li> <li>같은 방에 있는 Client에게 메시지 및 금치어 전달</li> <li>Client의 채팅 창 제어</li> <li>게임 종료 시 DB로 최신 정보 전달</li> <li>DB 정보 저장을 위한 파싱 처리</li> <li>Client 요청에 따라 DB 접근</li> </ul>	<ul style="list-style-type: none"> <li>TCP 통신에 비해 빠르고 가벼움, UDP의 Unreliable 단점 극복</li> <li>다수의 사용자가 서로 독립된 방에서 게임진행 가능</li> <li>기본적인 채팅 기능 제공 및 금치어 게임 진행</li> <li>금치어 지정 시에 채팅 활성화/비활성화 처리</li> <li>전적 정보 갱신</li> <li>알맞은 쿼리문 처리</li> <li>Client 유저 정보 전달</li> </ul>
<b>Client</b>	<ul style="list-style-type: none"> <li>Server에게 로그인 및 계정생성 요청</li> <li>Client에서 UI 출력 및 조작</li> <li>금치어를 판단하는 Logic 구현</li> </ul>	<ul style="list-style-type: none"> <li>직접적인 DB 접근을 막아 보안의 이점</li> <li>UI 전달 정보가 많을 경우 Server의 부하 방지</li> <li>Client가 자신의 금치어 사용여부 판단함에 따라 Server의 부하 감소</li> </ul>
<b>DB</b>	<ul style="list-style-type: none"> <li>계정 정보 테이블 생성</li> </ul>	<ul style="list-style-type: none"> <li>사용자 정보 저장 및 관리</li> </ul>

### 3-2-1 개발 내용 및 범위

- ‘Talkative Game’은 Unity 기반 멀티플레이 게임
- 클라이언트는 C#으로 개발
- 서버는 C로 개발
- ‘Talkative Game’은 로그인 시나리오, 플레이 시나리오, 갱신 시나리오와 같은 총 3개의 시나리오로 구성



[그림 4] ‘Talkative Game’ 시스템 구성도

#### 3-2-1-1. 로그인 시나리오

- 클라이언트로부터 로그인 요청
- 서버에서 DB에 있는 계정 정보 테이블 접근
- 결과 값에서 필요한 자료 파싱 처리 진행
- 파싱 처리 이후 결과 값 Socket으로 전달
- 계정 정보 일치하면 로그인 성공, 실패 시 에러 메시지 전달

#### 3-2-1-2. 플레이 시나리오

- 클라이언트에서 방 만들기 또는 방 진입 요청
- 서버에서 방 만들기 또는 방 진입 처리
- 클라이언트에서 게임 시작 요청
- 클라이언트 요청에 따라 채팅 제어 이후 금칙어 등록
- 정해진 Logic에 따라 각 플레이어 별 금칙어 전달 및 게임 진행

- 클라이언트에서 금칙어 처리 및 사망자 표시
- 최종 1인이 남을 때 까지 반복 진행

### 3-2-1-3. 전적 갱신 시나리오

- 게임 종료 후 승자 승리 횟수 증가
- 최종 1인에 대한 최신 전적 정보 갱신

## 3-2-2 구현 내용

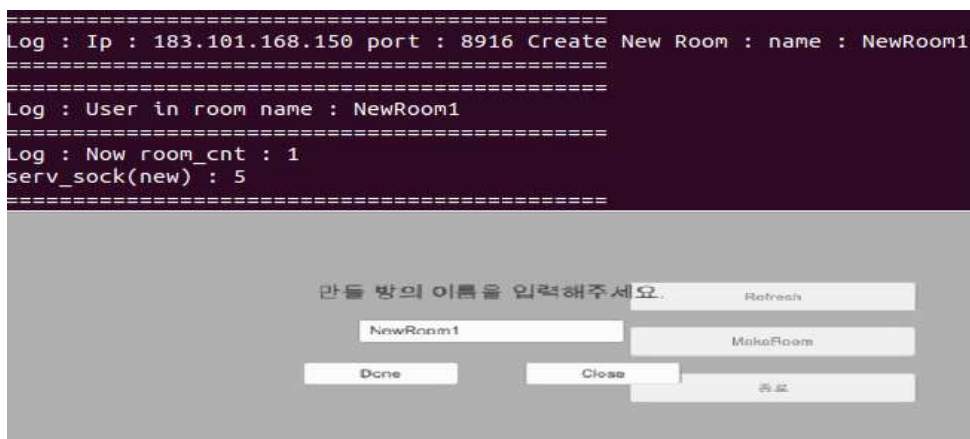
### ○ Server

#### - Socket 통신

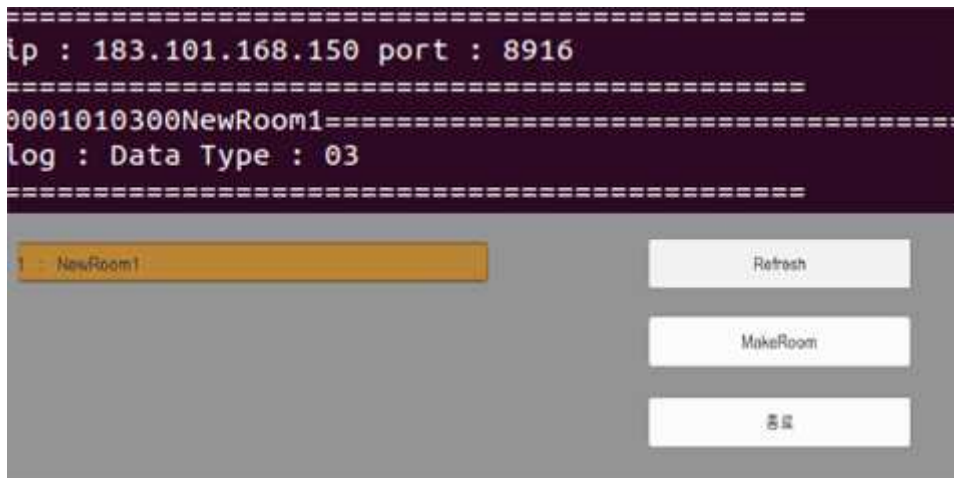
- C에서 제공하는 socket을 이용하여 UDP socket 할당
- Client와 통신을 할 때 Reliable-UDP를 사용하여 통신 하도록 구현
- 메인 서버 수신 데이터를 통해 Type 확인

#### - 로비 관리

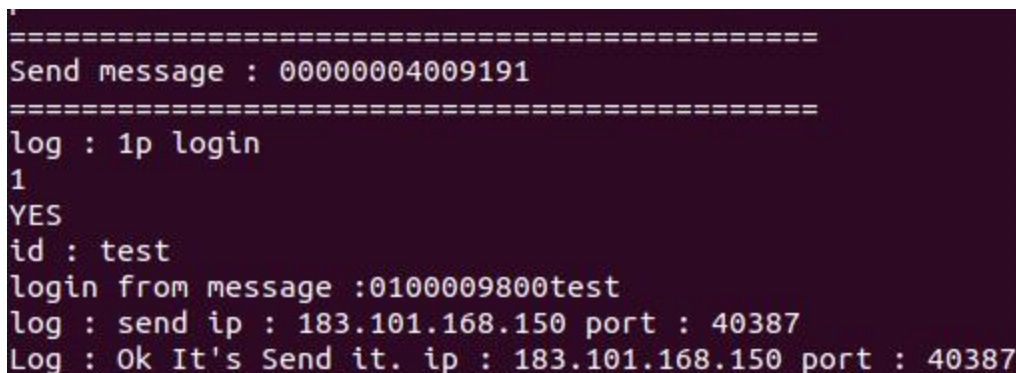
- Multi-Thread로 다수의 Client가 게임을 할 수 있도록 처리
- Thread의 인자로 로비에 할당될 서버 Port 번호를 보내 다수의 사용자가 로비에 접속할 수 있도록 처리
- 유저가 로비 접속 시 접속된 유저 정보를 클라이언트로 보내도록 처리



[그림 5] 로비 관리 실행 과정



[그림 6] 로비에 만들어진 채팅방 NewRoom1



[그림 7] New Room1에 접속했을 때 서버로 오는 메시지

#### - 메시지 전달(금칙어 전달)

- 같은 방에 접속되어 있는 모든 사용자에게 메시지를 보내는 sendto\_all() 구현
- 메시지 전달시 Client에서 수신되는 Type에 따라 금칙어 사용 여부를 판단
- 금칙어 사용 시 모든 플레이어에게 해당 플레이어 사망 메시지를 전달

#### - 채팅 제어

- 프로토콜 header에 있는 type에 따라 sendto\_all() 미사용
- 금칙어 지정 단계에서 채팅창 비활성화

#### - 전적 갱신

- SQL 쿼리문을 이용하여 게임 종료 시 전적 갱신
- DB 테이블 정보 최신화

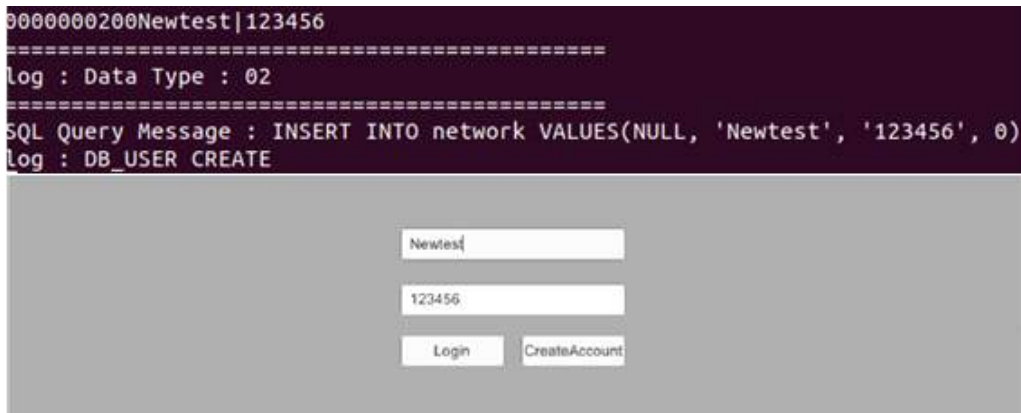
## ○ Client

### - Socket 통신

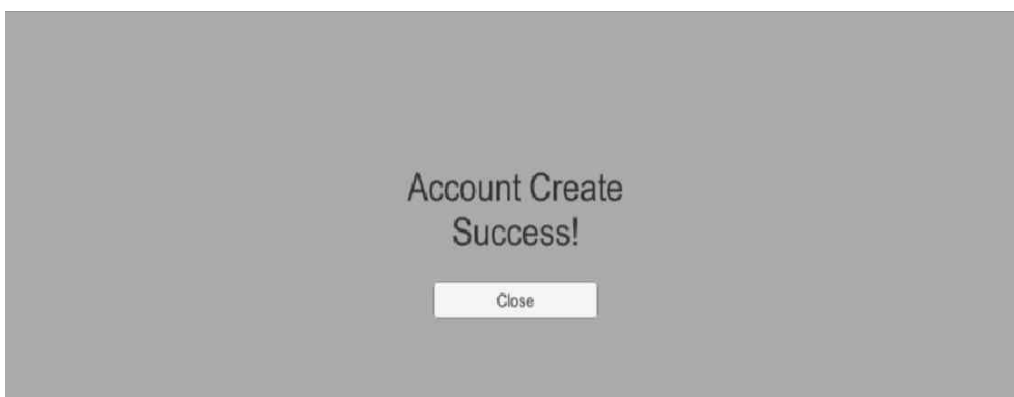
- C#은 데이터 전송 시 byte[]로 송수신
- byte 송수신에 따른 서버 측 수신 문제 발생
- Encoding을 이용하여 byte[]을 string 타입으로 변환
- 메시지 프로토콜 위치 크기에 맞춰 문자열 파싱 처리

### - 계정 정보 전달

- 구분문자와 Encoding을 통한 ID, Password 메시지 프로토콜 구현
- 프로토콜을 통한 로그인 결과 반환 구현



[그림 8] 계정 정보 생성 과정



[그림 9] 계정 정보 생성 화면

### - UI 출력 및 조작

- Unity UI 일종인 UGUI를 사용
- 로그인 화면 (로그인, 계정 생성 버튼)
- 로비 화면 (방 리스트, 방 생성, 새로고침, 나가기 버튼)
- 채팅 화면 (접속자 정보, 채팅UI, 금칙어 설정 패널, 게임시작버튼)

- 서버 정보를 기반으로 로비 방 리스트 UI 동적 생성



[그림 10] 채팅방 화면

#### - 금칙어 판단

- Client에서 사용자 메시지 검사
- 서버로 결과 전달
- 같은 방에 있는 Client에게 Broadcast
- Unity의 쓰레드 내부의 Unity함수 사용 불가 문제로 인한 미완성 모듈

### ○ DB

#### - C Server와의 연동

- webserver를 통한 DB접근 방식 기획
- 보안 문제 발생 방지를 위해 C server-DB 방식으로 재기획
- mysql.h의 함수를 통해 DB 접근

#### - 파싱 처리

- 로그인을 위한 ID 및 패스워드 정보 파싱(구분문자 사용)

#### - 계정 정보 테이블

- 유저 정보 DB 생성(network\_db)
- 구성요소는 ID\_NUM, id, password, wincount로 구성
- [그림 11] 참조



```
mysql> use network_db;
Database changed
mysql> SELECT * FROM network;
```

id_num	id	password	wincount
4	test1	1234	0
6	test	123456	0
9	paper1541	051200	0
10	qwerty	123456	0
11	qwerty123	1234567	0
12	zxcv	123456	0
13	asdfasdf	1234	0
14	zxcvbn	1234	0
15	asdxzcqwe	1234	0
16	1234321	1234321	0
17	qwer1234	123456	0
18	test2	123456	0
19	test3	123456	0
20	zxcvqw1234	123456	0
21	123456	1234	0
22	Account1234	051200	0
23	Account12345	051212	0
24	Account1234566	123456	0
25	CreateAccount	123456	0
26	Newtest	123456	0

```
20 rows in set (0.00 sec)
```

[그림 11] network 테이블 계정 정보

### 3-2-3 개발결과

[표 3] 개발 방법에 따른 개발 결과

구 분		개발 방법	개발 결과
DB	ID	MySQL을 Linux에 설치하고, MySQLh 의 함수를 통해 서버에게 데이터 전송	계정 정보 테이블
	Password		
Client	Socket	C#의 Socket 함수를 이용해 UDP 통신 구현	Socket 통신 모듈
	계정 정보	Encoding 기능을 통해 byte로 변환하여 데이터 전송	계정 정보 전달 모듈
	UI 출력	방 목록을 Server로부터 받아 해당 방 이름의 Button 동적 생성	UI 출력 모듈
	UI 조작	Unity OnClick 이벤트를 통해 여러 기능 수행(로그인, 계정생성, 방 입장)	UI 조작 모듈
	금칙어 판단	문자열을 Server에 전달하고, Thread를 이용해 메시지 출력	Unity 내 Thread에서 Unity 함수 사용 불가로 인해 실패
Server	Socket	C의 Socket 함수를 이용해 UDP 통신 구현	Socket 통신 모듈
	로비 관리	Thread를 이용하여 다수의 사용자가 게임을 진행할 수 있도록 로비 생성	로비 관리 모듈
	메시지 전달	다수의 사용자에게 메시지가 갈 수 있도록 sendto_all() 구현	메시지 전달 모듈
	채팅 제어	프로토콜 type 값에 따라 sendto_all() 을 호출하지 않도록 처리	채팅 제어 모듈(클라이언트 미완성으로 인한 테스트 실패)
	전적 갱신	게임이 종료 이후 최신 전적 정보를 SQL 쿼리문 이용하여 갱신	전적 갱신 모듈(클라이언트 미완성으로 인한 테스트 실패)

#### 4. 장애요소 및 해결방법

[표 4] 장애요소 및 해결방법

장애요소 항목	장애요소 내용	해결방법
UDP 비 연결성	UDP의 비 연결성으로 인한 클라이언트 접속 여부 확인 어려움	클라이언트 접속 시 서버로 특정 메시지를 보냄으로써 서버에 IP 정보 및 포트 정보 할당
DB 접근시 동기화 문제	다수의 클라이언트에서 DB 동시 접근 시 동기화 문제 발생	Mutex를 이용하여 DB 접근 동기화 처리
다중 클라이언트 연결 처리 문제	멀티 룸 형태를 제공할 경우 클라이언트가 서로 다른 로비 진입 설정 처리 필요	쓰레드에 생성될 서버의 port 값을 인자로 보내 방 생성 시 port 정보를 함께 할당
C와 DB 간 직접적인 통신	Mysql 사용 시 웹서버가 아닌 직접적으로 C Server가 DB에 접근 하도록 처리	Linux 환경에 MySQL 설치 및 MySQL.h를 통해 연결
Unity 오브젝트	방 목록 출력 시 해당 방으로 진입하기 위한 오브젝트 개수 유동적	주기적으로 방의 개수의 명단을 Server로부터 받아 해당 개수만큼 Object를 생성
채팅 UI 구현	채팅 내용이 중복되지 않으면서 Scrollview에 순차적으로 올라가는 방향으로 출력 필요	동적으로 Text 컴포넌트를 생성하여 부모 오브젝트에 vertical 레이아웃으로 배치하도록 구성 컴포넌트가 일정 개수를 넘어갈 시 콘텐츠 패널 확장

## 5. 개발 후기

[표 5] 에로사항

에로사항	개발하면서 가장 어려웠던 점에 대한 극복방법
다중 클라이언트 연결 처리 문제	UDP는 비 연결형 지향이므로 클라이언트 연결 여부 확인 어려움이 존재한다. 쓰레드로 다중 채팅방 지원 시 클라이언트는 서버로 접근 실패를 보이고, 서버는 클라이언트 측으로부터 메시지를 보내지 못하는 문제가 발생하였다. 쓰레드를 생성할 때 포트 정보를 인자로 보내 채팅 방 생성 시에 포트 정보를 할당하여 클라이언트에서 접속할 수 있도록 처리하였다. 클라이언트에서는 채팅 방에 접속 시 접속자 정보를 함께 보내어 서버 측에서 클라이언트의 IP 정보와 포트 정보를 가짐으로써 메시지를 보낼 수 있도록 처리 하였다.
C#과 C의 데이터 전달방식 차이	C는 문자열로 데이터를 받지만, C#은 문자열이 아닌 byte단위로서 데이터를 전송하는 문제가 있다. 이를 C#의 Encoding 기능을 활용해 문자열을 byte 단위로 변환하여, C Server에게 정상적으로 데이터를 전달할 수 있도록 하였다.
Unity Thread 지원 문제	Unity에서는 Thread 내 Unity함수를 사용할 수 없는 치명적인 문제가 존재하였다. 이에 따른 문제 해결방법을 찾지 못해, 실시간 채팅의 구현에 실패하였다.

[표 6] 개발 후기

팀원	개발 후기 (소감)
김한재	<p>사례를 찾기 어려웠던 UDP로 멀티 쓰레드 서버를 개발하면서 메시지 프로토콜의 중요성과 UDP 고유의 문제점을 확실하게 학습하게 되었다. 만약 프로젝트가 원활히 완성되었다면 더 큰 결실이었을 것이라고 생각하지만 제대로 완성을 하지 못한 것에 대해서 팀장으로서 미련이 많이 남을 것 같다. 프로젝트가 좋지 않은 방향으로 가는 어려움을 겪고 있음에도 불구하고 마지막까지 최선을 다해서 본 프로젝트를 수행해준 팀원들이 감사하다.</p>
함승우	<p>Unity의 C#과 C서버 간 데이터를 주고받을 때 서로 읽고 보내는 방식이 달라 처리하기 위한 함수를 찾을 필요가 있었고, 또한 Unity에서 쓰레드 내 Unity 함수를 사용할 수 없다는 사실을 알아차렸지만, 남은 개발 시간이 부족하여 클라이언트를 성공적으로 완성할 수 없어서 아쉬움이 남았다.</p>
황용훈	<p>Unity 라는 툴을 다뤄보면서 어려움도 많았고, 생각대로 되지 않아서 답답한 점도 많았지만, 다음번에 Unity로 작업을 할 일이 있다면 그때는 조금 더 나은 결과물을 낼 수 있지 않을까 싶다. 부족한 능력을 메꿔준 팀원들한테 미안한 마음이 크다. 학업 측면에서 부족함을 많이 깨닫게 되는 프로젝트였다.</p>