**네트워크 게임 프로그래밍**

**팀프로젝트**

신동원

김석현

**목차**

[1. 게임 기획 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084126)

[1.1 게임 개요 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084127)

[1.2 조작 방법 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128)

[1.3 구현 목표](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128) 4

[1.4 최종 구현 내용](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128) 5

[2. Level Design](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084131) 6

[2.1 서버(구현 목표)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 6

[2.2 서버(최종 구현)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 6

[2.3 클라이언트(구현 목표)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084133) 6

[2.4 클라이언트(최종 구현)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 6

[2.5 서버-클라이언트 관계도 Low-High level](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128) 7

[2.6 High level design(최종 구현)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 8

[2.7 Low level design(최종 구현)](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 9

[3. 함수 정리 11](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084136)

[3.1 서버](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084139) 11

[3.2 클라이언트](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084139) 13

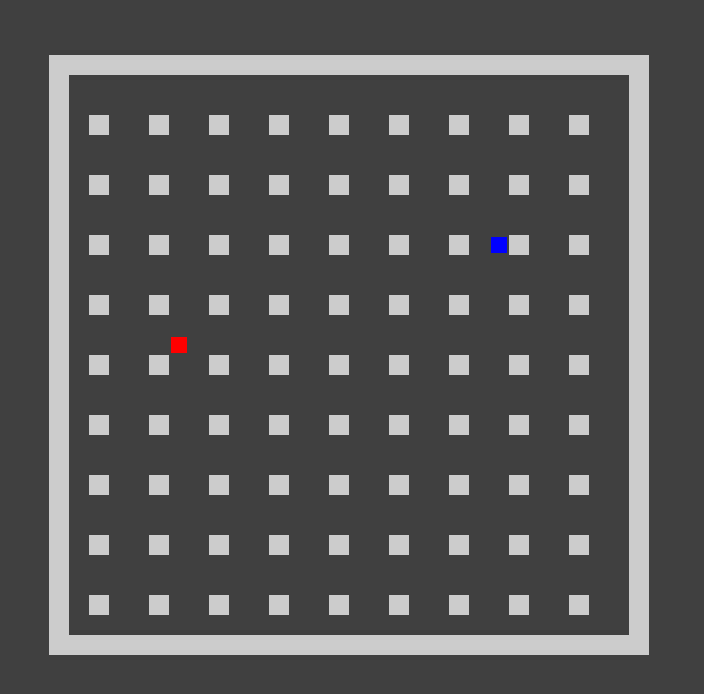
[4. 팀원 역할](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084143) 15

[5. 개발 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084145)5

[5.1 예상 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084147)5

5.2 실제 일정 [1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084147)6

1. **게임 기획**
   1. **게임 개요**



<컨셉 예시> <실제 구현>

게임: 봄버맨 모작게임

플랫폼: PC

규칙: 2명의 플레이어가 일정 시간 후에 폭발하여 4방위로 일정 영역에 데미지를 입히는 폭탄을 설치해 상대를 공격, 생존자 1명이 남으면 해당 플레이어가 최종적으로 승리합니다.

* 1. **조작 방법**

|  |  |
| --- | --- |
| 조작키 | 설명 |
| W, A, S, D | 플레이어의 이동 방향키 W-위/A-왼/S-뒤/D-오 |
| Space bar | 플레이어의 위치에 폭탄을 설치한다. |

* 1. **구현 목표**
* **클라이언트**
  + 타이틀 씬
    - 게임 시작 및 매칭
  + 게임 씬
    - 플레이어 - 이동, 폭탄 설치
    - 게임 정보 수신 및 display
    - 승패 여부 판단
* **서버**
  + 메인 스레드
    - 매칭 스레드 생성
    - 매칭 스레드에 클라이언트 연결
    - 업데이트를 담당할 게임 스레드 생성 및 환경 구성
  + 매칭 스레드
    - 매칭 환경 구성
    - 각 클라이언트를 담당할 메세지 스레드 생성 및 연결
  + 메세지 스레드
    - 클라이언트 입력 데이터 수신
    - 수신한 데이터를 게임 스레드에 업데이트
    - 업데이트된 게임 스레드의 정보를 클라이언트에 송신
  + 게임 스레드
    - 메세지 스레드를 통해 전달받은 데이터를 통해 연산
    - 연상 후 업데이트된 데이터를 메세지 스레드에 전달
  1. **최종 구현 내용**

**클라이언트**

렌더 씬

서버와 연결 확인, 미연결시 타이틀 씬 유지, 연결 되었을 시 GameState 값 True 로 변경 후 플레이어의 이동, 폭탄 설치 등의 데이터 입력 받음

게임 데이터를 서버와 송/수신 및 Update()결과를 화면에 송출

**서버**

타이틀 스레드

프로세스 스레드를 클라이언트 개수에 맞춰 생성

클라이언트와 연결, 클라이언트 정보 표시

클라이언트 연결 확인 후 게임 State 변경

매칭 큐에 Client소켓을 Push back 후 플레이어 생성

프로세스 스레드 1

클라이언트 1의 정보를 획득

MapData[i][j] 형태로 클라이언트 1의 데이터 수신

게임 스레드에서 업데이트 된 데이터를 클라이언트로 송신

프로세스 스레드 2

클라이언트 2의 정보를 획득

MapData[i][j] 형태로 클라이언트 2의 데이터 수신

게임 스레드에서 업데이트 된 데이터를 클라이언트로 송신

게임 스레드

메세지 스레드를 통해 전달받은 데이터를 통해 게임 규칙 연산

연산 후 변경된 데이터를 서버 GameData에 저장 후 프로세스 스레드에 전달

‘

1. **Level design** 
   1. **서버(구현 목표)**

* TCP/IP 프로토콜 방식
* 모듈 별 역할
  + main : 매칭 스레드, 게임 스레드 생성 및 클라이언트 연결
  + matching : 매칭 환경 구성 및 각 클라이언트 담당 메세지 스레드 생성 및 연결
  + message : 클라이언트와 통신
  + game : 수신 데이터를 통해 연산 및 업데이트 데이터 전달
  1. **서버(최종 구현)**
* TCP/IP 프로토콜 방식 – ipv4
* 모듈 별 역할
  + TitleThread : 클라이언트 연결 스레드, 매칭 환경 구성
  + ProcessThread : 클라이언트 담당, 스레드 생성 및 연결, 데이터 송 수신
  + GameThread : 수신 데이터를 통해 연산 및 업데이트 데이터 전달

**2.3 클라이언트(구현 목표)**

모듈 별 역할

title : 게임 시작 및 매칭

game : 게임 플레이 및 서버와의 통신

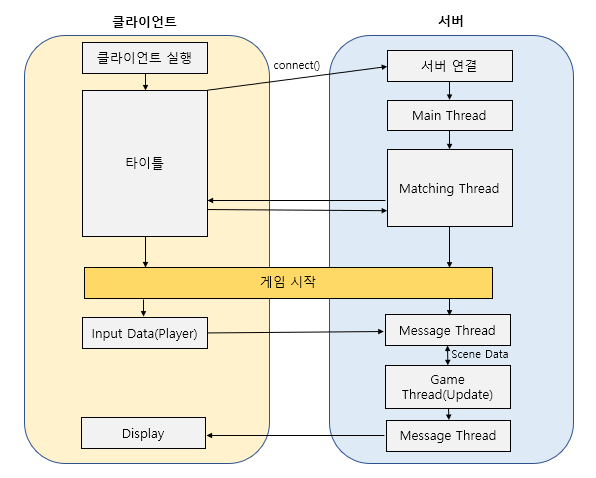
**2.3 클라이언트(최종 구현)**

모듈 별 역할

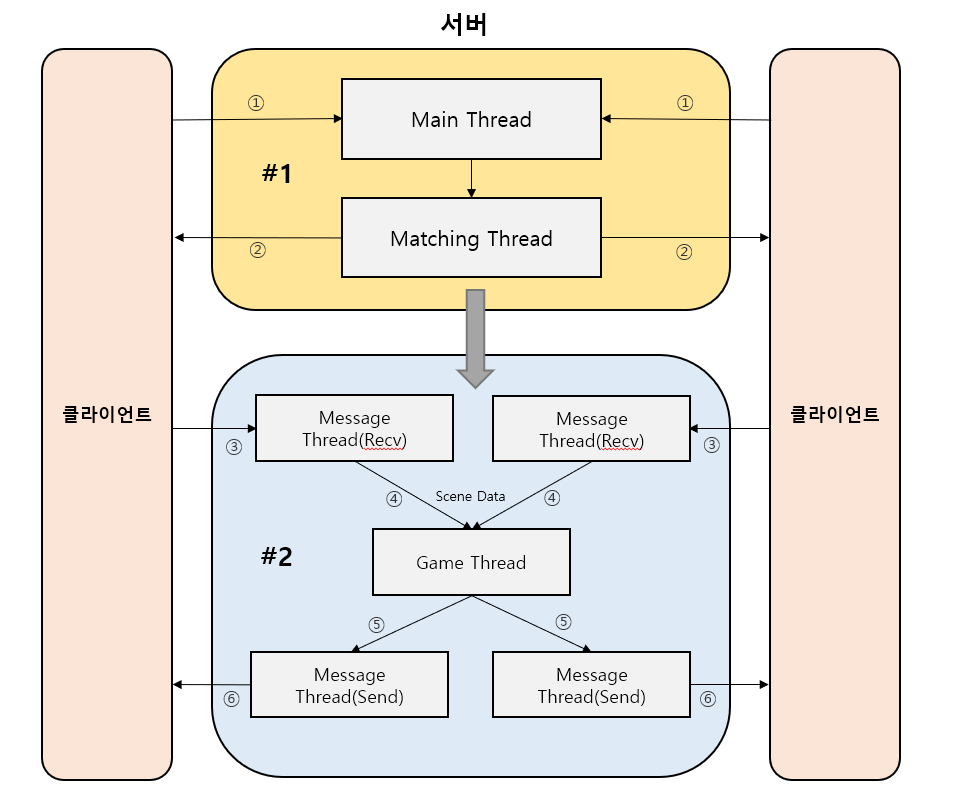
title : 서버와 연결 확인후 Game State 상태 True로 변경

game : 게임 플레이 및 서버와의 mapdata 통신

**2.3 서버-클라이언트 관계도 Low-High (구현 목표)**

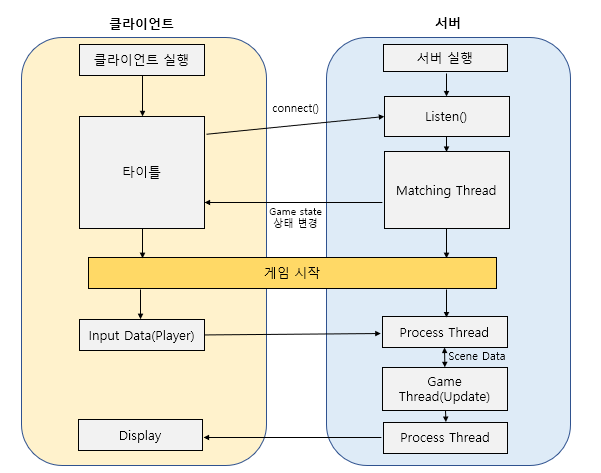


**High level design**



**Low level design**

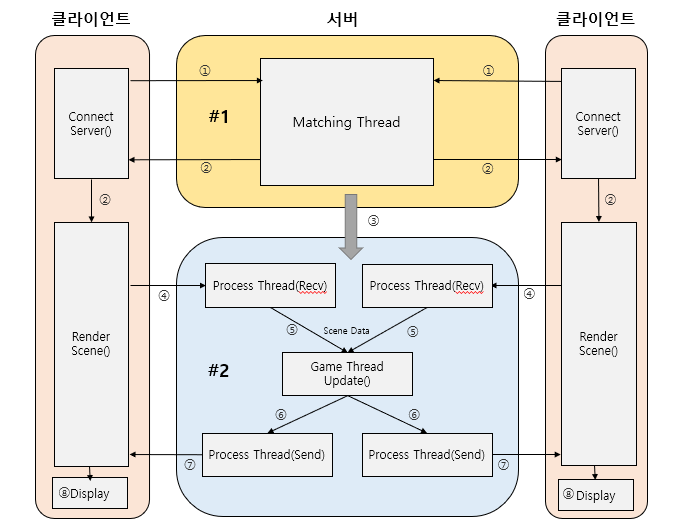
**2.4 High level design(최종 구현)**



[차이점] 기존의 설계형태와 구현 형태는 거의 흡사합니다. 하지만 main 스레드의 역할이 클라이언트로부터 연결을 확인후 matching 스레드를 생성하여, 매칭 큐를 초기화 후 플레이어를 생성하고 연결하였으나 main 스레드의 역할이 불분명하고 빈약하여 matching 스레드와 통합하였습니다.

구현의 세부내용은 아래와 같습니다.

**2.5 Low level design(최종 구현)**



1. 클라이언트 실행 후 명령행 인수 argv[1]에 통신하고자 하는 서버의 ipv4 주소를 입력받아 서버에게 Connect()한다.

서버는 event 생성 후 시그널 상태로 matching 스레드를 생성한다. Matching 스레드는 윈속 초기화 후 클라이언트로부터 listen 상태로 기다리며, 연결이 확인되면 Matching큐 에 연결된 순서대로 클라이언트 소켓을 받아 담는다.

1. 정상적으로 서버가 연결되면 클라이언트는 Game State를 True값으로 변경한다. 서버는 연결을 확인후 서버 상에서 2개의 플레이어를 생성한다. 플레이어는 Matching 큐에 순서대로 0,1번으로 생성되며, 매칭큐에 담긴 정보는 클라이언트로 보내진다.
2. 매칭 쓰레드는 두개의 프로세스 쓰레드를 생성하여, 2개의 클라이언트와 연결 순서대로 배정한다.
3. 클라이언트는 g\_game 과 g\_title 생성 후, 100millis 마다 RebderScene()을 실행한다. 클라이언트는 Keyinput, 플레이어의 위치 정보등의 게임정보를 mapData[MAP\_SIZE][MAP\_SIZE] 형태로 담아 서버로 보낸다. 만약 Game state 값이 false 라면 타이틀 Sence로 변경한다.
4. 서버는 클라이언트로부터 정보를 얻어온 후 각 프로세스 스레드는 클라이언트와 데이터 통신을 시작한다. 이때 프로세스 스레드1이 먼저 오브젝트의 상태가 시그널인지 확인한다. 시그널 상태임을 확인하면 프로세스 1이 실행되고, 완료된 후 이벤트 시그널 상태로 만들며, 프로세스2가 시그널 상태인 오브젝트를 확인해 실행된다.

클라이언트로부터 는 mapData[MAP\_SIZE][MAP\_SIZE]와 key input을 받는다.

1. 클라이언트로부터 받은 mapData와 key input을 서버의 game data에 담는다. 이때 게임 스레드를 생성해 서버의 게임데이터는 Update()로 보내 처리한다.
2. 서버는 md[MAP\_SIZE][MAP\_SIZE]를 만들어 서버의 게임 데이터에 담긴 정보를 담아 클라이언트로 보낸다.
3. 서버로부터 온 md[MAP\_SIZE][MAP\_SIZE] 를 기반으로 그래픽 정보를 화면에 표시한다.
4. **함수 정리**
   1. **서버**
5. 통신 데이터 구조

unsigned char ClientNum

vector<SOCKET> MatchingQueue

typedef struct ClientState

{

SOCKET ClientSocket

bool ReadyState = 0

}

class SceneData

{

Private:

PlayerStatus m\_playerStatus[MAX\_PLAYER];

MapData m\_mapData[MAP\_SIZE][MAP\_SIZE];

Public:

update();

keyInput(SOCKET socket, KeyInput key)

}

struct PlayerStatus

{

SOCKET playerSocket;

char speed;

char power;

bool isAlive;

};

1. 함수

송신

Recv\_Con\_data() 클라이언트 로부터 데이터 소켓을 전달받는 함수

Recv\_input\_data() 클라이언트 로부터 입력 데이터를 받는 함수

Recv\_check\_state() 클라이언트로부터 실시간으로 게임 상태를 확인하도록 요청 받아 확인하는 함수

수신

Send\_Con\_data() 클라이언트 에 데이터 소켓을 전달하는 함수

Send\_draw\_data() 서버에서 Update 처리한 데이터를 클라이언트에 보내는 함수

send\_check() 2명의 player의 상태를 보내는 함수

처리

Change\_status() 게임의 상태를 play, wait, end 로 바꿔주는 함수

Check\_player() 2명의 player가 서버와 연결되어 준비상태인지 확인하는 함수

Update() 입력 받은 데이터를 확인해 최신화 하는 함수

Check\_state() 게임의 진행 상황을 서버가 확인하도록 요청하는 함수

Timeout() 일정시간 안에 입력 데이터의 전송이 되지 않고 지연 시 False로 처리하는 함수

1. 변수

struct Point

{

short x;

short y;

};

struct PlayerStatus

{

KeyInput key;

char speed; // 플레이어 속도

char power; // 폭탄 길이

Point position; // 플레이어 위치

bool isAlive; // 생존 여부

};

struct MapData

{

Bool isRock

Bool isBoomb;

};

* 1. **클라이언트**

1. 통신 데이터 구조

struct KeyInput

{

bool key\_UP;

bool key\_Down;

bool key\_Left;

bool key\_Right;

bool key\_Space;

};

1. 함수

송신

Recv\_Con\_data() 서버 로부터 데이터 소켓을 전달받는 함수

Recv\_draw\_data() 서버에서 받은 Update 데이터를 그리는 함수

Recv\_check() Player가 모두 준비되었는지 데이터를 받는 함수

수신

Send\_Con\_data() 서버에 데이터 소켓을 전달하는 함수

Send\_input\_data() 입력 받은 데이터를 서버에 전달하는 함수

처리

Change\_status() 게임의 상태를 play, wait, end 로 바꿔주는 함수

Update() 입력받은 데이터를 확인해 최신화 하는 함수

Check\_state() 게임의 진행 상황을 서버가 확인하도록 요청하는 함수

Timeout() 일정시간 안에 입력 데이터의 전송이 되지 않고 지연 시 False로 처리하는 함수

1. 변수

enum Item

{

EMPTY, // empty

BALLON, // 폭탄 개수 증가

POTION, // 폭탄 길이 증가

};

enum PlayerColor

{

RED,

BLUE,

PLAYEREMPTY,

};

1. **팀원 역할**

|  |  |
| --- | --- |
| 이름 | 역할 |
| 신동원 | 서버 프레임 워크 및 송신/수신 함수 구현 |
| 김석현 | 클라이언트 송신/수신함수 구현, 서버 구현 보조 및 문서 작성 |

1. **개발 일정**

**5.1 예상 일정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **11월** | | | | | | | |
| **구분** | **일** | **월** | **화** | **수** | **목** | **금** | **토** |
| **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **신동원** | **계획서 작성** | | | | | | |
| **김석현** |
|  |
|  | **13** | **14** | **15** | **16** | **17** | **18** | **19** |
| **신동원** | **정기 회의** | **계획서 작성** | | **Message Thread 구현** | | **서버 수신함수 구현** | |
| **김석현** | **클라이언트 수신함수 구현** | | | **보완** |
|  | **세부 함수 구현(1) – update()** | | | **보완** |
|  | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| **신동원** | **정기 회의** | **서버 송신함수 구현, Main Thread 구현** | | | | | **1차 병합** |
| **김석현** | **서버 – 클라이언트**  **접속 테스트** | | **문제점 보완 및**  **클라이언트 송신함수 구현** | | |
|  | **Matching Thread 구현** | | |
|  | **27** | **28** | **29** | **30** | **1** | **2** | **3** |
| **신동원** | **정기 회의** | **Game Thread 구현**  **문제점 보완 및 세부 함수 구현** | | | | 보완 | 2차 병합 |
| **김석현** |
|  | **세부 함수 구현** | | | |
|  | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **신동원** | **정기 회의** | 최종 테스트 및 수정, 개선, 검수 | | | 발표 |  | |
| **김석현** |
|  |

**5.2 실제 일정**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **11월** | | | | | | | | | |
| **구분** | **일** | **월** | | **화** | **수** | | **목** | **금** | **토** |
| **6** | **7** | | **8** | **9** | | **10** | **11** | **12** |
| **신동원** | **계획서 작성** | | | | | | | | |
| **김석현** |
|  |
|  | **13** | **14** | | **15** | **16** | | **17** | **18** | **19** |
| **신동원** | **정기 회의** | **계획서 작성** | | | | | | **서버 수신함수 구현** | |
| **김석현** | **클라이언트 수신함수**  **구현** | |
|  | **20** | **21** | | **22** | **23** | | **24** | **25** | **26** |
| **신동원** | **정기 회의** | **서버 송신함수 구현, Matching Thread 구현** | | | | | | | **접속 문제**  **해결** |
| **김석현** | **클라이언트 송신함수**  **구현** | | | **서버 – 클라이언트**  **접속 테스트** | | | |
|  | **27** | **28** | | **29** | **30** | | **1** | **2** | **3** |
| **신동원** | **정기 회의** | **Process Thread 구현**  **문제점 보완 및 세부 함수 구현** | | | | | | 병합 | |
| **김석현** |
| **세부 함수 구현 및 recv 오류 수정** | | | | | |
|  | **4** | **5** | | **6** | **7** | | **8** | **9** | **10** |
| **신동원** | **정기 회의** | 최종 테스트 및 수정, 개선, 검수 | | | | | | | |
| **김석현** |
|  | **11** | **12** | **13** | | |  | | | |
| **신동원** | 문서 작성, 빌드용 제작 | | **최종 발표** | | |
| **김석현** |