

FSM 기반 “마피아” 통신 프로토콜 보고서

- 2조
- 마수민, 문유정, 배예진, 신지윤

1. 시스템 개요

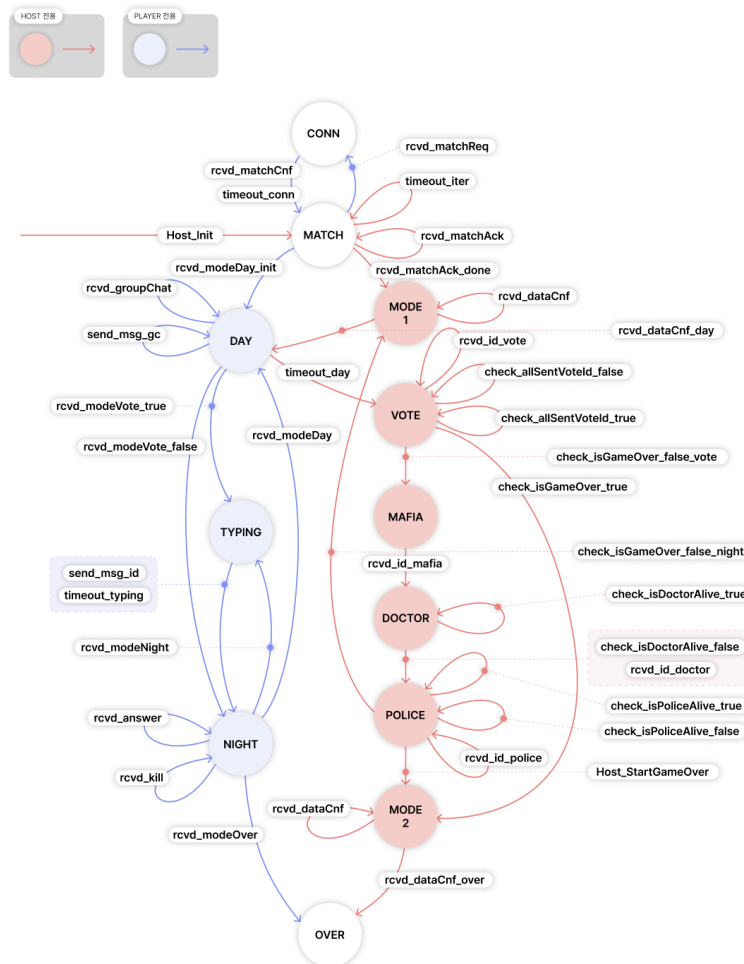
본 시스템은 FSM(Finite State Machine) 구조를 바탕으로

마피아 게임 (역할 분배 및 통신 프로토콜)을 C++ 코드로 구현한 것입니다.

플레이어는 Host 또는 Guest로 나뉘며,

Guest 는 매번 랜덤으로 각자 **Mafia / Doctor / Police / Citizen** 중 하나의 역할을 부여받습니다.

2. FSM 전체 구조



3. 기존 FSM 과 코드 구현의 차이

초기 FSM 설계에 비해 실제 구현 과정에서 다음과 같은 점이 구체적으로 변경 및 단순화되었습니다.

1. TYPING 상태 생략

- 설계도에는 TYPING 상태가 존재했었습니다. 이는 NIGHT 중 HOST 와 소통이 필요한 경우 (ex. 마피아가 죽일 사람 선택 / 경찰이 조사할 사람 선택 / 의사가 살릴 사람 선택) 이동하는 상태였습니다.
- 구현에서는 TYPING 상태는 제외하고, Guest 와 HOST 가 함께 이동하여 송수신하는 흐름으로 단순화하였습니다.

2. MODE_2 상태 제거

- 설계도에서는 POLICE 이후 MODE_2 상태를 거쳐야 OVER 로 전이되었습니다.
- 구현에서는 POLICE 상태에서 바로 OVER 또는 DAY 상태로 전이되도록 하여 중간 허브 상태인 MODE_2 를 제거했습니다.

✓ 단순한 FSM 흐름 유지, 불필요한 중복 상태 제거 목적

3. rcvd_* 이벤트 정리 및 축소

- 설계도에는 다양한 세부 수신 이벤트(rcvd_modeDay_init , rcvd_modeVote_true , rcvd_kill , rcvd_modeOver 등)가 독립적으로 표현되었습니다.
- 구현에서는 이 중 중복되거나 유사한 의미의 이벤트를 통합하여 rcvd_modeDay , rcvd_voteResult , rcvd_nightResult 등으로 단순화하였습니다.

4. 코드별 동작 상세 설명

#1. 상태 : IDLE , MATCH

- IDLE : 초기 대기 상태.
- MATCH : HOST 제외 4명의 GUEST 모집.

→ 별도의 인터페이스 없음.

#2. 상태 : MODE_1

- MODE_1 (역할 분배)
 - Host
 - createPlayers() → 역할 생성. 랜덤 함수 사용.
 - getRoleName() → 역할 문자열 생성. 각 역할을 문자열(ex. Doctor) 로 안내.
 - ACK 수신되면 다음 플레이어에게 본인의 역할 송신
 - 모든 플레이어에게 확인을 받으면 DAY 전이.
 - Guest
 - 역할 수신 후 myRoleName 에 저장.
 - ACK 송신 후 DAY 전이.

핵심 변수

- `change_state` : 외부 트리거 (송수신 여부로 어느 플레이어까지 전달되었는지 확인)
- `waitingAck` , `waitingHostInput` , `currentSendIndex` : FSM 진행 보조

```

COM8 - PuTTY
Player 1 - ID: 3, Role: Citizen

Player 2 - ID: 7, Role: Mafia

Player 3 - ID: 8, Role: Doctor

SEND ROLE to ID 2 : Police

ACK received from player ID 2

다음 플레이어에게 전송할까요? (1 입력)

```

```

SEND ROLE to ID 3 : Citizen

ACK received from player ID 3

다음 플레이어에게 전송할까요? (1 입력)

1 입력 확인. 다음 플레이어로 전송합니다.

SEND ROLE to ID 7 : Mafia

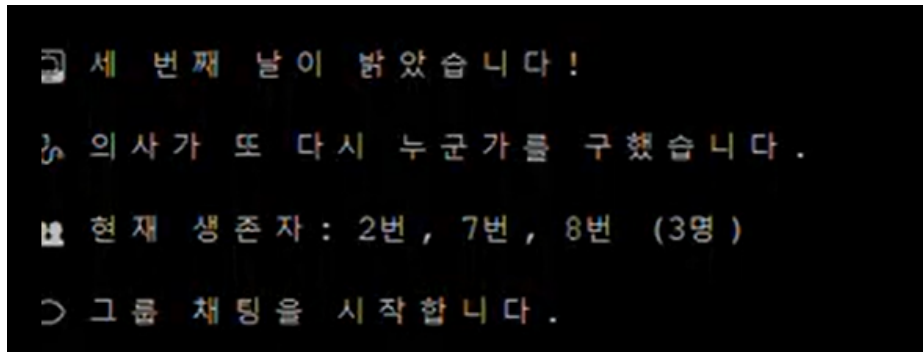
```

- 랜덤으로 역할을 나눈 뒤 첫 번째 플레이어 (사진에서는 2번) 에게 역할 전달
- 플레이어가 본인의 역할을 전달받고 ack 를 송신
- ack 를 수신받으면 다음 플레이어에게 전송할지 문구 출력
- 1을 누르면 다음 플레이어에게 역할 전달
 - 1을 입력받지 않으면 실제 시연 과정에서 꼬임 현상이 발생함.
 - 아마도 타이밍의 문제로 보임. 때문에 보다 안전한 전송을 위해 중간 과정 삽입

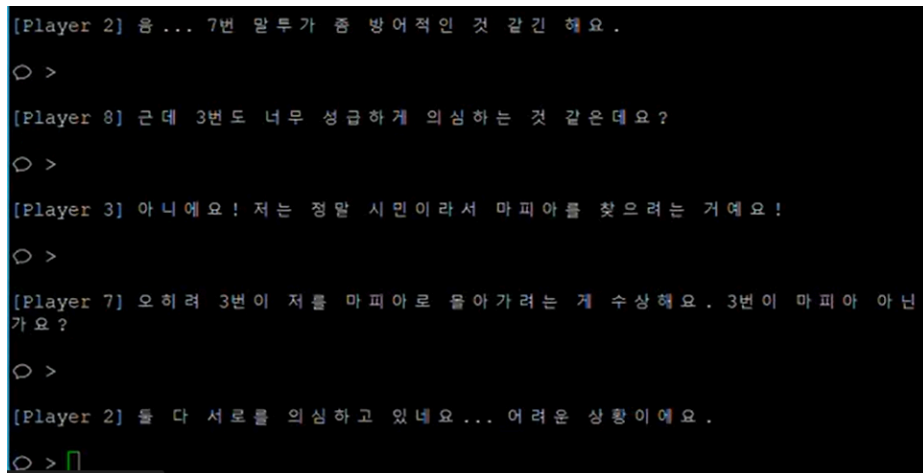
#3. 상태 : DAY

- DAY (단체 채팅)

- 모든 플레이어에게 "낮이 되었습니다" 출력
- HOST 를 제외한 4명의 GUEST 끼리 단체 채팅
- 다음 상태 : VOTE



세 번째 날이 밤았습니다!
의사가 또 다시 누군가를 구했습니다.
현재 생존자 : 2번 , 7번 , 8번 (3명)
그룹 채팅을 시작합니다.



```
[Player 2] 음 ... 7번 알루가 좀 방어적인 것 같긴 해요 .  
O >  
[Player 8] 근데 3번도 너무 성급하게 의심하는 것 같은데요 ?  
O >  
[Player 3] 아니에요 ! 저는 정말 시민이라서 마피아를 찾으려는 거예요 !  
O >  
[Player 7] 오히려 3번이 저를 마피아로 몰아가려는 게 수상해요 . 3번이 마피아 아닌가요 ?  
O >  
[Player 2] 둘 다 서로를 의심하고 있네요 ... 어려운 상황이에요 .  
O > █
```

- HOST 는 메시지 내용을 수신받지 않음. (ack 와 같은 통신 확인 기능 없음)

#4. 상태 : VOTE

- VOTE (생존자 투표)

- Host
 - `change_state == 0` : 초기화
 - 생존자 목록 구성 → 메시지 전송
 - "1" 입력 시 다음 플레이어 전송
 - 응답 수신 후 표 집계
 - `voteResults[]` 분석 → 최다 득표자 선정
 - 동점 시 처형 없음 처리
 - 각 생존자에게 결과 및 게임 상태 전송 → ACK 수신

- Guest

- 생존자 목록 수신 → 본인 제외 ID 중 투표
- 유효 ID만 허용, 자신 투표 불가
- 결과 수신 → 자신이 죽으면 `idead = true`
- 게임 종료 여부 확인 → `ACK` 전송

```

🔊 투표를 시작합니다.

👤 살아있는 플레이어 목록 :
Player ID: 2 Player ID: 3 Player ID: 7 Player ID: 8
-----

[Host] 2번 플레이어에게 투표 요청 : 투표하세요. 본인을 제외한 ID 중 선택 : 3 7 8
▶ 2번 플레이어가 3번에게 투표했습니다.
▶ 다음 플레이어에게 전송하려면 '1'을 입력하세요.
☑ 다음 플레이어로 이동합니다.
[Host] 3번 플레이어에게 투표 요청 : 투표하세요. 본인을 제외한 ID 중 선택 : 2 7 8

```

- `change_state == 3` 에서 다음 FSM 진입 준비
- 마피아 피해 적용, 의사 보호 적용
- 죽었는지 확인 → `idead` 갱신
- 다음 상태
 - `OVER` : 게임 종료
 - `MAFIA` , `DOCTOR` , `POLICE` , `NIGHT` : 역할 기반 전이
- 아래는 게스트 입장에서 유효한 투표를 하는 조건입니다.

```

// 3. 게스트 측: 투표 요청 메시지 수신 → 투표 입력 → 전송
if (myId != 1 && change_state < 2 && L3_event_checkEventFlag(L3_event_msgRcvd)) {
    uint8_t* dataPtr = L3_LLI_getMsgPtr();
    uint8_t size = L3_LLI_getSize();

    pc.printf("\r\n📧 투표 메시지 수신: %.*s", size, dataPtr);

    // 유효한 투표 대상 ID 파싱
    int validIDs[NUM_PLAYERS];
    int validIDCount = 0;
    for (int i = 0; i < size; i++) {

```

```

    if (dataPtr[i] >= '0' && dataPtr[i] <= '9') {
        int id = dataPtr[i] - '0';
        if (id != myId) {
            validIDs[validIDCount++] = id;
        }
    }
}

int valid = 0;
int voteTo = -1;

while (!valid) {
    pc.printf("\r\n 🗳 투표할 플레이어 ID를 입력하세요: ");
    while (!pc.readable());
    char ch = pc.getc();
    pc.printf("%c", ch);

    if (ch < '0' || ch > '9') {
        pc.printf("\r\n ! 숫자가 아닙니다. 다시 입력하세요.");
        continue;
    }

    voteTo = ch - '0';

    if (voteTo == myId) {
        pc.printf("\r\n ! 자신에게는 투표할 수 없습니다.");
        continue;
    }

    bool isValid = false;
    for (int i = 0; i < validIDCount; i++) {
        if (validIDs[i] == voteTo) {
            isValid = true;
            break;
        }
    }

    if (!isValid) {
        pc.printf("\r\n ! 해당 ID는 유효한 투표 대상이 아닙니다. 다시 입력하세요.");
        continue;
    }

    valid = 1;
}

// 투표 결과 전송 (Host = 1)
char ackMsg[4];
sprintf(ackMsg, "%d", voteTo);

```

```

L3_LLI_dataReqFunc((uint8_t*)ackMsg, strlen(ackMsg), 1);
L3_event_clearEventFlag(L3_event_msgRcvd);

change_state = 2;
}

```

```

무표 메시지 수신 : 투표하세요 . 본인을 제외한 ID 중 선택 : 3 7 8
투표할 플레이어 ID를 입력하세요 : 3
게스트 2] 투표 결과 수신 :
표 결과 : [2: 0표] [3: 2표] [7: 2표] [8: 0표]
동점으로 아무도 죽지 않았습니다 .
낮으로 넘어갑니다 .
동점으로 처형된 플레이어 없음
게스트 2] ACK 전송 완료

```

- 동점으로 넘어가는 경우 (DAY 로 이동)

```

7번 플레이어가 처형되었습니다 .
시민 승리 ! 게임 종료 .
7번 플레이어가 처형됨
시민 승리 ! 게임 종료 처리 필요
게스트 8] ACK 전송 완료
[게스트 8] 게임 종료 메시지 수신 :
게스트 8] ACK 전송 완료

```

- 마피아가 처형되어 게임이 종료된 경우 (OVER 로 이동)

#5. 상태 : NIGHT

• NIGHT (대기/죽은 사람)

- "밤이 되었습니다" 출력 & 현재 본인 상태 출력
- 실제 행동(ex. 마피아가 죽일 사람 dm)은 각 역할 FSM에서 처리
- NIGHT 에서는 대기만 진행
- 죽은 사람도 NIGHT 에서 대기

```

내 번호는 3입니다.
내 역할은 Citizen입니다.
내 생존 상태 : 죽음

밤이 되었습니다.

```

#6. 상태 : **MAFIA**

- **MAFIA** (마피아/호스트)
- 흐름도

1. Host : 살아있는 마피아에게 타겟 선택지 전송 (마피아가 죽었으면 애초에 게임 종료)
2. Mafia : ID 입력 후 전송
3. Host : 응답 수신 후 'sentVoteld' 저장
4. 상태 전이 : DOCTOR

```

내 번호는 7입니다.
내 역할은 Mafia입니다.
내 생존 상태 : 살아 있음
[Mafia] 타겟 선택 메시지 수신 : 죽일 ID를 선택하세요 : 2 3 8
[Mafia] 죽일 ID를 입력하세요 : 3[Mafia] 3번을 죽이기로 선택하여 Host에 전송 완료

```

#7. 상태 : **DOCTOR**

- **DOCTOR** (의사/호스트)
- 흐름도

1. Host : 살아있는 의사에게 '살릴 ID' 요청 (의사가 죽었으면 바로 POLICE 로 넘어감)
2. Doctor : ID 입력 후 Host에 전송
3. Host : 응답 수신 → 'doctorTarget' 저장
4. 마피아와 결과 대조
 - 마피아 피해 대상 ≠ 보호 대상 → 피해 발생
5. 내부 변수 초기화 (sentVoteld, doctorTarget)
6. 상태 전이: POLICE


```

# 역할은 Doctor입니다 .
# 생존 상태 : 살아 있음
[의사] 메시지 수신 : 살릴 사람의 ID를 입력하세요 : 2 3 7
[의사] 살릴 ID 입력 : 3[의사] 3번을 살리기로 선택하고 Host에 전송 완료
[Doctor] DOCTOR 단계 완료 → DAY로 전환

이 되었습니다 .

```

#8. 상태 : **POLICE**

- **POLICE** (대기/죽은 사람)
- 흐름도

1. Host : 경찰에게 '정체 확인할 ID' 요청
2. Police : 유효 ID 입력 후 전송
3. Host : 역할 조회 후 결과 회신
4. Police : 결과 수신 후 'ACK' 전송
5. Host : 'POLICE_PHASE_END' 브로드캐스트
6. 모두 : 'POLICE_PHASE_END' 수신 시 'DAY' 전환

```

# 역할은 Police입니다 .
# 생존 상태 : 살아 있음
[Police] 메시지 수신 : 정체를 확인할 ID를 입력하세요 : 3 7 8
[Police] 확인할 ID 입력 : 3[Police] Host에 정체 확인 요청 전송 완료
[Police] 수신된 정체 : Citizen
[경찰] POLICE 단계 종료 → DAY로 전환

이 되었습니다 .

```

#9. 상태 : **OVER**

- **OVER** (대기/죽은 사람)
 - **Host** : 모든 생존자에게 종료 메시지 순차 전송 → **ACK** 확인
 - 게스트는 수신 시 **ACK** 전송
 - 완료 시 **L3STATE_IDLE** 전이

5. 마무리

본 보고서에서는 FSM(Finite State Machine) 기반으로 설계된 마피아 게임의 상태 흐름을 초기 다이어그램과 실제 구현을 비교하는 방식으로 분석하였습니다.

초기 설계는 다양한 가능성과 확장성을 고려하여 구성되었으며, 실제 구현에서는 그 구조를 기반으로 불필요한 상태를 제거하고 로직을 간소화함으로써 안정적인 실행을 우선으로 하였습니다.

이번 FSM 기반 구현을 통해 **게임 로직의 명확한 구조화, 디버깅 용이성, 확장 가능성 확보** 등의 효과를 확인할 수 있었으며, 이는 향후 채팅 기능, 애니메이션 효과, 역할 추가 등 기능 확장에도 유용한 기반이 될 것입니다.

향후에는 사용자 경험 개선을 위한 **UI/UX 개선**, 실시간 통신 최적화 및 FSM 단위 테스트 체계 마련 등을 목표로 추가적인 개선을 이어갈 예정입니다.

교수님, 조교님, 팀원분들 모두 한 학기 동안 정말 감사했습니다!!