목차

Service

Service definition

- 관리자와 사용자 두 가지 역할 필요
- 관리자가 접속한 사용자들을 랜덤으로 일대일 매칭
- 랜덤 매칭된 두 사용자는 키보드로 text message를 입력 받아 대화
- Text message는 ARQ 방식으로 신뢰성 있게 송수신

Service

Service - 1. Initial connection establishment

1. 사용자(단말) 접속 시 관리자와 자동 연결
2. 사용자는 본인의 ID를 직접 지정
3. 관리자는 접속한 사용자의 ID를 Random matching 대기열에 포함

Service - 2. Random matching process

- 4개 미만의 단말이 대기열에 접속한 경우

  다른 단말이 접속할 때까지 대기

- 2개 이상의 단말이 대기열에서 3초 이상 대기, 또는 4개 이상의 단말이 대기열에 접속한 경우

  관리자는 단말을 2개씩 임의로 매칭

  각 단말에 상대방 ID 정보를 전송

  대기열의 단말 개수가 홀수인 경우, 마지막에 접속한 단말은 다시 대기열에서 대기

- 사용자는 관리자가 각 단말에 전송한 ID로 수신자를 지정하고 채팅 시작

Service

Service - 3. Chatting

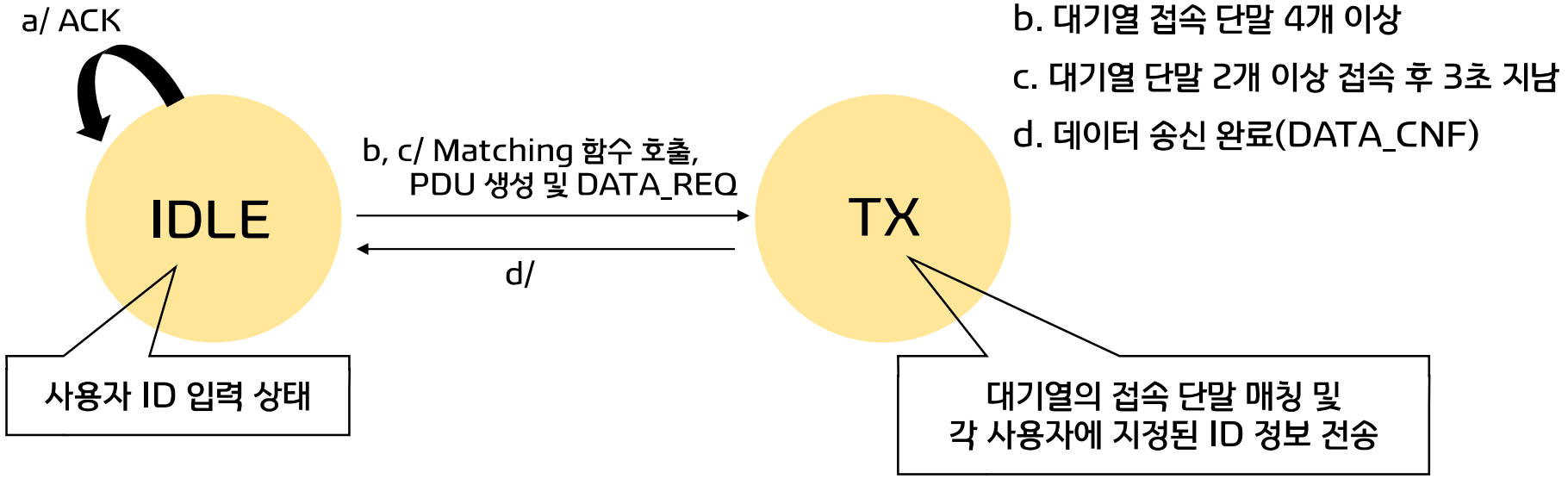
- 두 단말이 연결되면 채팅 시작
- 관리자를 거치지 않고 두 사용자가 1:1로 채팅
- 채팅 도중 text message(PDU) 전송에 실패하면 ARQ 방식으로 재전송
- 지정된 상대 외 message 송수신 무시
- 한 사용자라도 채팅 중단을 원하는 경우 채팅 종료

Service - 4. End Chatting

- 채팅 종료 후 두 사용자의 단말은 자동으로 관리자의 대기열에서 매칭 대기

FSM design

Manager



Event

a. 사용자의 ID 정보 수신

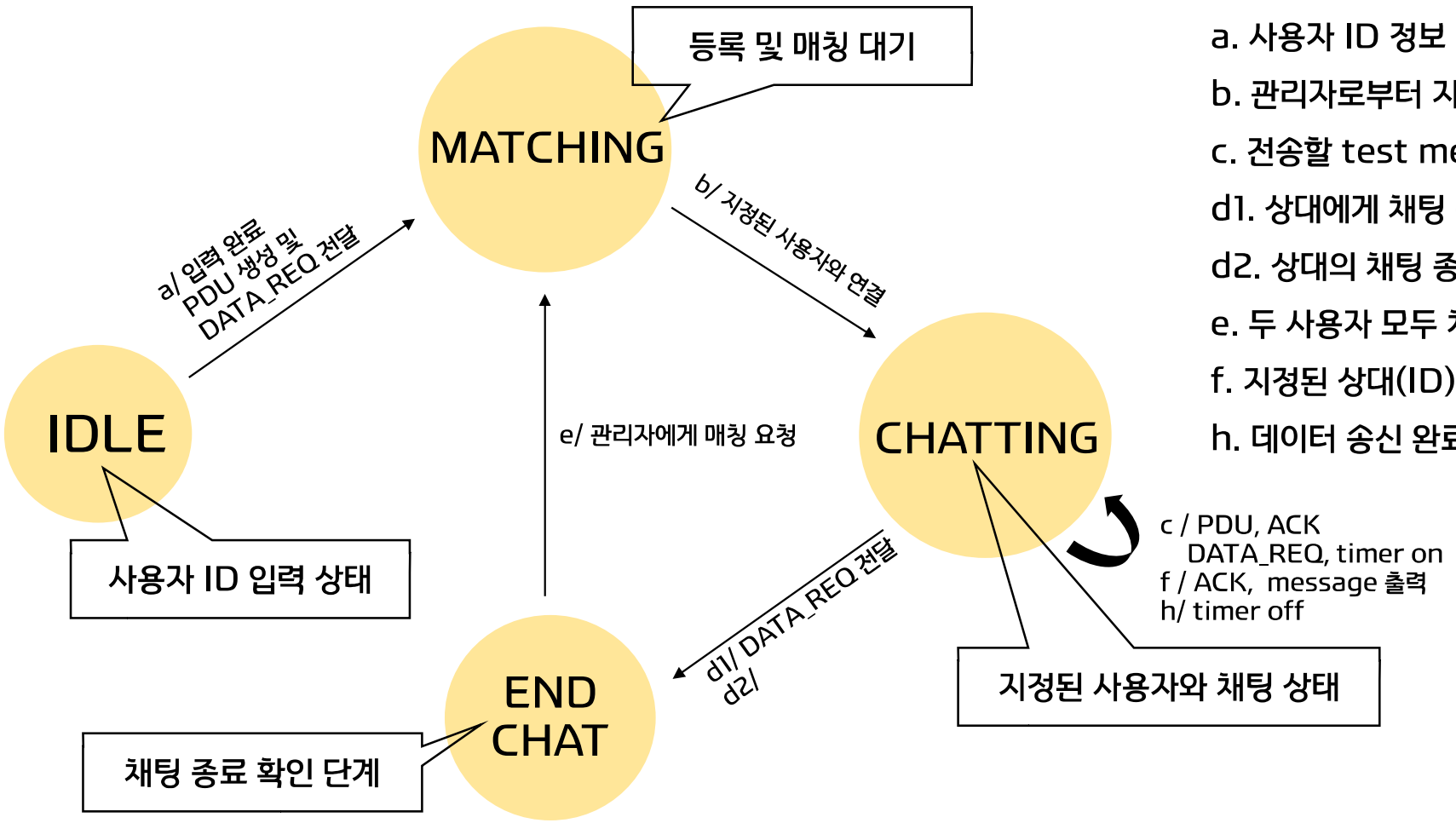
b. 대기열 접속 단말 4개 이상

c. 대기열 단말 2개 이상 접속 후 3초 지남

d. 데이터 송신 완료(DATA_CNF)

FSM design

USER



Event

a. 사용자 ID 정보 입력(SDU) 수신

b. 관리자로부터 지정된 상대의 ID 정보 수신

c. 전송할 test message(SDU) 수신

d1. 상대에게 채팅 종료 명령어(PDU) ‘exit’ 송신

d2. 상대의 채팅 종료 명령어 수신

e. 두 사용자 모두 채팅 종료 확인

f. 지정된 상대(ID)를 확인하여 message 수신

h. 데이터 송신 완료(DATA_CNF)

code - MANAGER

매칭 대기열 관리

```
switch (main_state)
{
    case L3STATE_IDLE: //IDLE state description

        if (L3_event_checkEventFlag(L3_event_msgRcvd)) //msgRcvd is matchRcvd
        {
            //Retrieving data info.
            uint8_t* dataPtr = L3_LLI_getMsgPtr();
            uint8_t size = L3_LLI_getSize();
            uint8_t id = L3_LLI_getSrcId();

            debug("\n ------------------------------------------------\nRCVD Match Req by %i,
                        id, size, dataPtr);

            //매칭 대기열에 추가
            matchArr[matchCnt] = id;
            matchCnt++;
            printArr2(matchCnt);
            checkMatchCdtion();
            pc.printf("the number of matching user: %d\n",matchCnt);

            L3_event_clearEventFlag(L3_event_msgRcvd);
```

접속한 유저가 enter를 치면 관리자에 매칭 요청 전송

↓

관리자가 메시지를 받으면 유저의 ID를 저장해
매칭 대기열에 저장

code - MANAGER

매칭 시작

```
int checkMatchCdtion(){
    if (matchCnt>=2 && matchCnt<4 && L3_timer_getTimerStatus()==0){
        L3_timer_startTimer();
    }
    if (matchCnt>=4 || L3_event_checkEventFlag(L3_event_Timeout)){
        L3_event_setEventFlag(L3_event_matchStart);
        L3_event_clearEventFlag(L3_event_Timeout);
    }
    else {
        return 0;
    }
```

2명 이상 접속 후 일정 시간이 지났을 때 혹은
4명 이상이 접속했을 때 매칭 시작

유저 매칭

```
else if(L3_event_checkEventFlag(L3_event_matchingTx)){
    uint8_t user=0;
    uint8_t mate=0;

    if(txIndex<mtchCpl){
        user = matchShuffledArr[txIndex];
        if(txIndex%2==0){
            mate = matchShuffledArr[txIndex+1];
        }
        else{
            mate = matchShuffledArr[txIndex-1];
        }
```

```
    pc.printf("user:%d,mate:%d\n",user,mate);
    L3_LLI_dataReqFunc(&mate, sizeof(uint8_t), user);

    //user set back-up(for blacklist)
    mtchBackUpArr[user] = mate;

    main_state = TX;

    }
txIndex++;
break;
```

매칭 대기열을 섞고, 2개씩 짝 지어 각 유저에게
상대 ID 전송

code - USER

매칭 시작

```
void L3_initFSM(uint8_t destId)
{
    myDestId = destId;
    //initialize service layer
    pc.attach(&L3service_processInputWord, Serial::RxIrq);
    pc.printf("Wanna start a random chat? press the [Enter]");
}
```

```
switch (main_state)
{
    case L3STATE_IDLE: //IDLE state description
        if (L3_event_checkEventFlag(L3_event_dataToSend)){
            pc.printf("finding user... please wait.\n");
            strcpy((char*)sdu, (char*)originalWord);
            debug("[L3] msg length : %i\n", wordLen);
            L3_LLI_dataReqFunc(sdu, wordLen, myDestId);
            debug_if(DBGMSG_L3, "[L3] sending msg....\n");
            wordLen = 0;
            L3_event_clearEventFlag(L3_event_dataToSend);
            prev_state = main_state;
            main_state = MATCHING;
        }
        break;
```

접속, ID 입력 후
enter 치면 매칭 요청

채팅 종료

```
case CHATTING:
    if (L3_event_checkEventFlag(L3_event_msgRcvd)) //if data reception event ha
    {
        //Retrieving data info.
        uint8_t* dataPtr = L3_LLI_getMsgPtr();
        uint8_t size = L3_LLI_getSize();
        if(L3_msg_isEndChat((char*)dataPtr)){
            pc.printf("\n ----------------------\nChatting Endded\n ------
            main_state = ENDCHAT;
            L3_event_clearEventFlag(L3_event_msgRcvd);
```

```
else if (L3_event_checkEventFlag(L3_event_dataToSend)) //if
{
    //end chat condition check
    if(L3_msg_isEndChat((char*)originalWord)){
        pc.printf("\n ------------------\nChatting Endded\n
        strcpy((char*)sdu, (char*)originalWord);
        debug("[L3] msg length : %i\n", wordLen);
        L3_LLI_dataReqFunc(sdu, wordLen, myDestId);
        debug_if(DBGMSG_L3, "[L3] sending msg....\n");
        wordLen = 0;
        main_state = ENDCHAT;
        L3_event_clearEventFlag(L3_event_dataToSend);
    }
```

상대 혹은 본인이 “EXIT” or “exit” 입력한 경우
채팅 종료

code - USER

재매칭 요청

```
case ENDCHAT:
    if(L3_event_checkEventFlag(L3_event_dataToSend)){
        debug("this is ENDCHAT-dataToSend\n");
        if(endchatCnt==0){
            strcpy((char*)sdu, (char*)originalWord);
            debug("[L3] msg length : %i\n", wordLen);
            L3_LLI_dataReqFunc(sdu, wordLen, MANAGER_ID);
            debug_if(DBGMSG_L3, "[L3] sending msg....\n");
            wordLen = 0;
        }
        endchatCnt++;

        main_state = MATCHING;
        L3_event_clearEventFlag(L3_event_dataToSend);
    }
    break;
```

채팅 종료 후,
첫 매칭 때처럼 enter치면
자동으로 관리자에 매칭 요청 전송