ft_irc - 평가용 문서

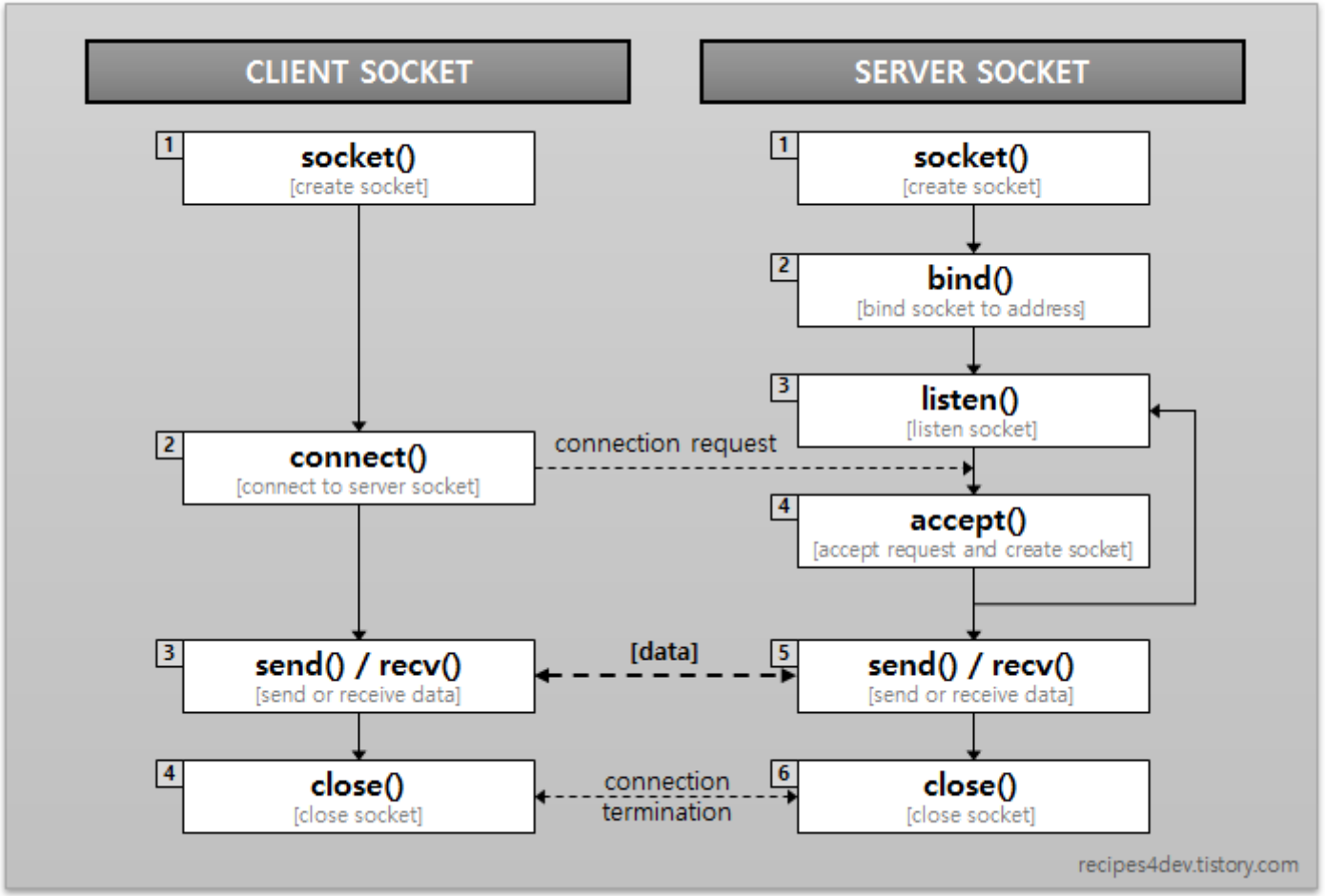
목차

- 메세지
  - PRIVMSG, NOTICE
- 운영자
  - OPER, KILL
- 종료
  - QUIT

소켓 프로그래밍

소켓의 정의

- 소켓은 프로세스 간 통신의 종착점이자 통로이다. 네트워크 너머의 개체와 개체 간 통신을 이어주는 만큼 아래와 같은 특성을 가진다
  - 서로를 규정하기 위한 식별자 : IP, PORT
  - 인터넷 통신을 위한 규약 : TCP, UDP
  - 편리를 위해 구분한 소켓 유형 : 서버 소켓(수문장), 클라이언트 소켓(통신 연결 요청)



- 소켓 통신 흐름도
  - 소켓은 운영체제에서 관리하며 그렇기에 소켓 간의 통신에서 사용하는 입출력 버퍼 또한 커널에서 임시로 할당하고 관리한다.

TCP 통신의 특징

- TCP 규약을 이용한 통신은 "안정성", "데이터 경계가 없음"이 특징이다.
- 안정성
  - 데이터가 정확히 상대에게 보내졌는 지 검증을 함. 그래서 송수신자 전부 데이터를 받으면 답신을 보내며 답신이 보내지지 않았으면 다시 데이터를 보내는 작업을 거친다.
  - 양방향 통신을 지원하여 송수신 측이 동시에 데이터를 송수신해도 문제 없음
- 데이터 경계가 없음

- 보내는 쪽에서 데이터를 분할해서 보낸 들 받는 쪽에서 데이터를 분할해서 받을 지 한 번에 받을 지 모른다는 얘기다. 그래서 TCP 통신을 할 경우에는 1. 고정 길이 지정, 2. 가변 길이 지정, 3. 문자열의 끝을 지정 등. 서로 문자열을 어디까지 읽어들여야 온전한 메세지가 되는지 약속이 필요하다.

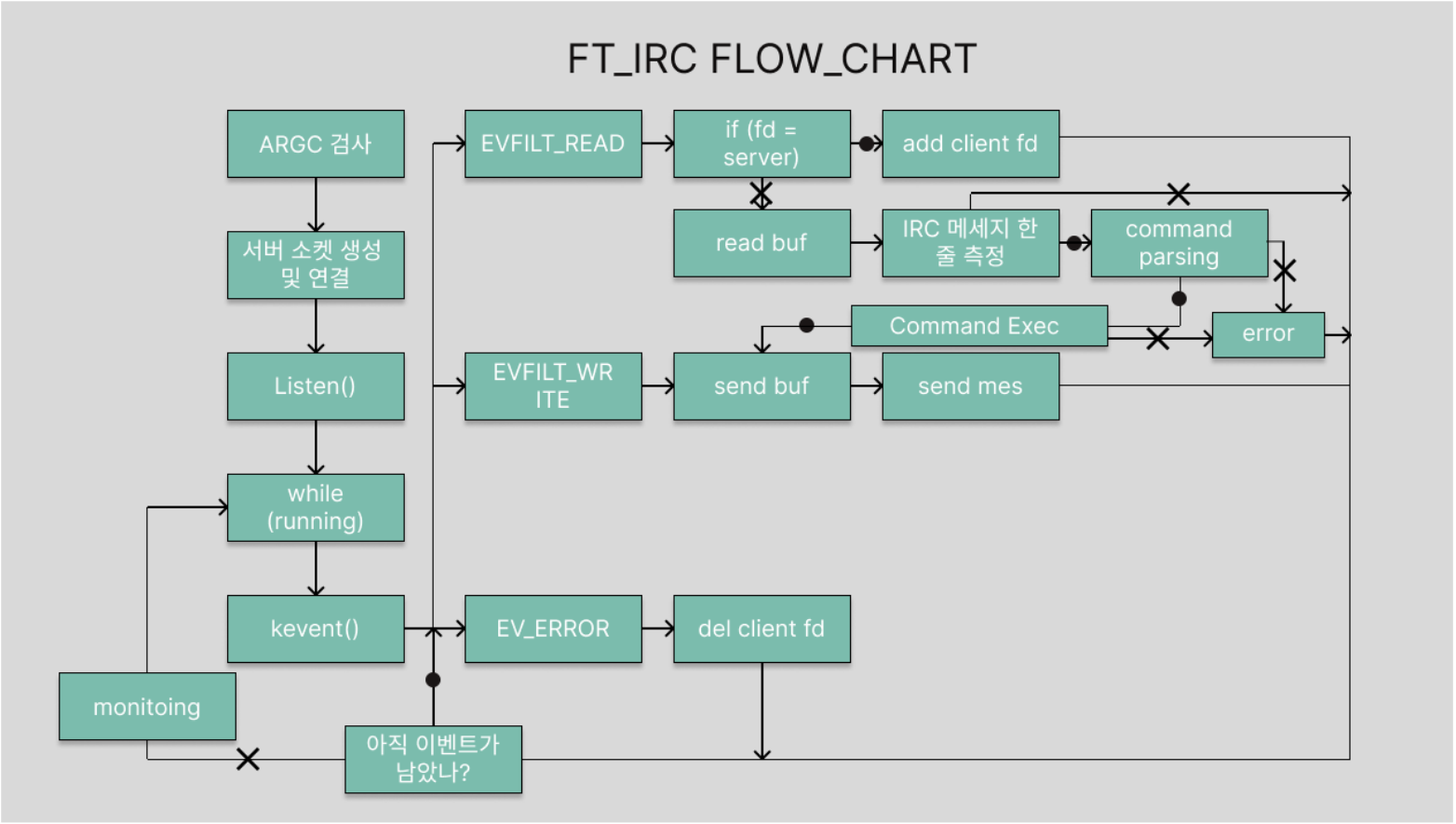
I/O Multiplexing

- 서버가 다수의 클라이언트를 받는 방법으로 고안된 안 중 하나.
- 멀티 프로세스 방식은 메모리를 너무 많이 차지하고 멀티 스레드는 문맥 교환이 너무 잦다. 그래서 단일 프로세스 단일 스레드로 위 두 가지 방식을 극복하고자 튀어 나옴
- I/O multiplexing의 특징
  - 각각의 클라이언트와 연결된 소켓을 구성하여 해당 소켓의 이벤트를 감지하는 방식으로 멀티 클라이언트를 처리함
  - 커널에서 여러 I/O에 대한 이벤트를 감지해서 multiplexing 관련 함수(select, poll, epoll, kqueue 등)를 통해 이벤트 목록을 한 번에 넘겨주고 관리하는 건 비동기적이나 개별 I/O 작업은 서버에서 각각 처리하고 있는 점에서 동기로 볼 수 있다.
  - 또한, multiplexing 관련 함수들은 커널에서 이벤트를 감지하고 각 목록에 대한 이벤트를 띄어줄 때까지 기다리는 점에서 블록이라 볼 수 있지만, fcntl을 통해서 커널이 할당한 임시 버퍼의 내용물을 읽어들이는 과정을 논블럭으로 만들면 직접적인 I/O 작업은 논블럭이라고도 볼 수 있다. 여기에 각 멀티플렉싱 함수에 timeout을 설정하면 결과를 받기도 전에 시간이 지나면 함수를 빠져나오므로 논블럭으로 볼 수 있다.
  - 이처럼 보는 입장에 동기, 비동기, 블럭, 논블럭 적인 면모가 달라질 수 있다.

프로그램의 구조

개괄

- flow chart



서버

- 프로그램의 총체적인 역할 분담과 관리를 맡는다.

클라이언트

- 단일 클라이언트가 지녀야 할 정보와 필요 함수를 지니고 있음

채널

- 단일 채널이 지녀야 할 정보와 필요 함수를 지니고 있음

커맨드

- 정적 클래스 Message가 파싱한 결과물을 확인하고 각 커맨드에 따른 실행 처리를 맡는다

IRC 명령어 설명 및 사용법

기본적인 메세지 구조

- 서버
  - :<source> <command> <parameter> <crlf>
- 사용자
  - <command> <parameter> <crlf>
- 각 문자 집합은 ' '를 구분자로 둔다
- <source> : 서버명 또는 사용자의 이름(nick!user@host)이 올 수 있다
- <command> : 명령어 문자열 또는 숫자 응답이 나올 수 있다
- <parameter> : 명령어를 시행하는 데 필요한 매개변수 모임. <middle>과 <trailing>으로 나뉜다
  - <middle> : ' ', NULL, : (only first character), <crlf>를 제외한 문자열이 올 수 있음.
  - <trailing> : NULL, <crlf>를 포함하지 않는 문자열. 만약, 공백까지 포함해서 한 문자열로 처리하기를 원할 경우, 반드시 맨 첫번째 문자에 ':'을 붙여야 한다
- <crlf> : IRC에서 통신 시, 서로 보내는 메세지의 말미에는 반드시 "\r\n"이 붙어야 한다. 단, 최근 irc에서는 일단 \r, \n만 들어와도 처리는 해준다(irc modern docs).

사용자 인증

- PASS, NICK, USER를 거쳐서 사용자 인증을 하고, 인증을 거치는 데 성공한 경우에는 인증 받은 사용자에게 MOTD를 통해 서버의 정보와 함께 환영 인사를 건넨다.

- PASS
  - 명령어 : PASS <password>
  - 서버의 암호 인증을 위해 사용함
  - 오류 검사
    - 메세지가 명령어 형식이 아님
    - 이미 PASS 등록을 마쳤음
    - 입력한 암호가 서버에 등재된 암호와 다름
- NICK
  - 명령어 : NICK <nickname>
  - 서버 내에서 클라이언트를 부를 고유 별칭을 정한다
  - 별칭 규칙 : 9자 이내, #, :, ' ' 사용 금지
  - 오류 검사
    - 메세지가 명령어 형식이 아님
    - 중복된 별칭임
    - 별칭 규칙에서 벗어남
- USER
  - 명령어 : USER <username> <hostname> <servername> <realname>
  - username : 클라이언트의 ID
  - hostname : 클라이언트의 호스트명
  - servername : 클라이언트가 접속한 IRC 서버명
  - realname : 클라이언트의 실명
  - 클라이언트의 정보를 저장한다.
  - 오류검사
    - 메세지가 명령어 형식이 아님
    - 이미 USER 등록을 하였음
    - USER 등록 전에 PASS와 NICK을 거치지 않았음

연결확인

- PING
  - 명령어 : PING [<token>]
  - 서버 - 클라이언트 간 연결 확인을 위해 사용
  - 오류검사
    - token이 없거나 빈 문자열임
    - 명령어 형식에 안 맞음
- PONG
  - 명령어 : PONG [<token>]
  - PING에 대한 응답

채널

- JOIN
  - 명령어 : JOIN <channel>{,<channel>} [<key>{,<key>}]
  - 특정 채널에 참여하고 싶을 시 사용하는 명령어
  - 채널 참여 시에 채널에 속한 클라이언트에게 모두 응답이 간다
  - 암호가 정해진(+k) 채널에서는 제공한 key가 정확하지 않으면 참여할 수 없다
  - 초대 전용(+i) 채널에서는 채널 멤버에게 초대받지 못했으면 참여할 수 없다
  - 오류검사
    - 명령어 형식에 맞지 않음
    - 채널명이 빈 문자열이거나 접두사가 없거나 200자 이상임
    - 각 클라이언트가 가입할 수 있는 채널 한도를 넘음
    - 채널에서 보유할 수 있는 클라이언트 한도를 넘음
    - 초대 전용 채널인데 별도 초대를 받지 못했음
    - 암호 보유 채널인데 암호가 맞지 않음
- PART
  - 명령어 : PART <channel>{,<channel>}
  - 특정 채널에서 떠나고 싶을 시 사용하는 명령어

    - 채널에 참여한 활성 사용자 목록에서 제거됨
    - 오류검사
        - 명령어 형식에 맞지 않음
        - 채널이 존재하지 않음
        - 해당 채널에 속하고 있지 않음
- INVITE
    - 명령어 : INVITE <nickname> <channel>
    - 채널에 사용자를 초대할 때 사용
    - 초대 전용 채널에서 사용할 때는 채널 운영자만 가능
    - 오류 검사
        - 명령어 형식에 맞지 않음
        - 채널이 존재하지 않음
        - 해당 채널에 속하지 않음
        - 초대 전용 채널에서 채널 운영자 이외의 사람이 초대를 시도함
        - 이미 초대를 받았음
- KICK
    - 명령어 : KICK <channel> <user> [<comment>]
    - 해당 채널에서 사용자를 쫓아낼 때 사용
    - 채널 운영자만 사용 가능
    - 오류검사
        - 명령어 형식에 맞지 않음
        - 채널이 존재하지 않음
        - 채널 운영자 이외의 사람이 명령어를 시도
- TOPIC
    - 명령어 : TOPIC <channel> [<topic>]
    - 해당 채널에서 주제를 정하거나 주제를 확인한다
        - topic 을 입력하면 해당 내용을 주제로 선정
        - topic이 입력되지 않으면 해당 채널의 주제를 출력

    - 주제 보호 채널에서 사용할 때는 채널 운영자만 가능
    - 오류검사
        - 명령 형식에 맞지 않음
        - 채널이 존재하지 않음
        - 주제 보호 모드인데 명령어 사용자가 채널 운영자가 아님

모드

- MODE
    - 명령어 : MODE <channel> {[+ | -] i | t | o | l | k} <value>
    - 서브젝트에서 구현이 필요한 초대 전용(i), 채널 참가자 제한(l), 주제 보호(t), 채널 운영자 변경(o), 채널 암호(k) 5개를 선정하기 위해 필요한 명령어
    - t(선택), o, l, k 전부 value 값이 필요. 물론, - 부호가 붙었을 때는 별도로 필요 없음
    - 오류검사
        - 명령어 형식을 따르지 않음
        - 모르는 모드임
        - 채널이 존재하지 않음
        - 채널 운영자가 존재하지 않거나 지정된 채널 운영자가 모드 명령어를 쓰는 클라이언트와 다름
        - 각 모드에 필요한 부가 내용물이 존재하지 않음
        - k 모드를 제거하는 경우, 설정한 채널 암호가 맞지 않음
        - o 모드를 설정하는 경우, 권한을 넘겨주려는 대상이 존재하지 않음

메세지

- PRIVMSG
    - 명령어 : PRIVMSG <receiver>{,<receiver>} <text>
    - 수신자(receiver)에게 특정 메세지(text)를 보낸다
    - 운영자 권한이 있는 경우, $*을 이용하여 브로드 캐스트 메세지를 보낼 수 있다
    - 오류검사

        - 수신자가 없음
        - 보낼 메세지가 없음
        - 채널로 보내는 경우
            - 존재하지 않는 채널임
            - 지역 채널(@)의 경우, 채널 운영자 여부를 확인했는데 운영자가 아님
            - 보내려는 사람이 채널에 존재하지 않는 사용자임
        - 사용자에게 보내는 경우
            - 존재하지 않는 사용자임
- NOTICE
    - 명령어 : NOTICE <receiver>{,<receiver>} <text>
    - PRIVMSG처럼 기능이 비슷하나 자동 응답을 받지 않는다

운영자

- OPER
    - 명령어 : OPER <OperName> <OperPassword>
    - 전용 운영자명과 암호가 정확하고 운영자가 존재하지 않을 시 운영자가 되는 명령어
    - 오류검사
        - 명령어 형식을 따르지 않음
        - 지정된 운영자명과 다름
        - 지정된 운영자 암호와 다름
        - 이미 운영자가 존재함
- KILL
    - 명령어 : KILL <targetNick> <reason>
    - 운영자만 사용할 수 있으며 지정한 사용자를 서버에서 쫓아낸다
    - 오류검사
        - 명령어 형식을 따르지 않음
        - 쫓아내려는 대상이 존재하지 않음

        - 명령어 사용자가 운영자가 아님

종료

- QUIT
    - 명령어 : QUIT [<reason>]
    - 사용자가 IRC 서버에서 탈출할 때 사용
    - 만약, 사용자가 QUIT 메세지를 사용하지 않고 나간 경우, 서버가 대신 이를 다른 사용자들에게 나갔다고 알려준다