컴퓨터 네트워크 과제 2 2021039057 이규하

Task 1 : 강의자료와 기타 참고자료를 활용하여 socket 통신 기반의 채팅 프로그램을 작성하시오.
작성할 socket 프로그램의 기본 사양을 아래와 같습니다.
1) 언어: Python 2) 방식: 1 대 1 TCP 연결 3) 지원 내용: 문장 전송 (chatting)
4) UI 없음, console 모드에서 문장만 주고받음.

Task 1에서는 Python의 소켓 프로그래밍을 사용하여 순수 1대1 TCP 채팅 프로그램을 구현하는 것입니다. 서버와 클라이언트 간의 텍스트 데이터를 실시간으로 송수신할 수 있는 콘솔 기반 채팅 프로그램을 만들기 위한 환경 및 계획은 다음과 같습니다.

1. **Python 환경 설정 및 라이브러리 조사:** Python의 'socket' 모듈을 사용하여 TCP 통신 구현, (추가적인 외부 라이브러리나 도구는 사용하지 않으며, 표준 라이브러리를 최대한 활용함)
2. **개발 환경**

- OS 환경: Ubuntu 24.04 on Windows 11 WSL2
- Python 버전: 3.9.20 with conda env

```
gyuha_lee@KONGSUNILAPTOP: $ conda activate socket-chat
(socket-chat) gyuha_lee@KONGSUNILAPTOP: $ python
Python 3.9.20 (main, Oct  3 2024, 07:27:41)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
gyuha_lee@KONGSUNILAPTOP: $ cat  /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=24.04
DISTRIB_CODENAME=noble
DISTRIB_DESCRIPTION="Ubuntu 24.04 LTS"
PRETTY_NAME="Ubuntu 24.04 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04 LTS (Noble Numbat)"
```

- 실행 네트워크 환경: 127.0.0.1 (localhost) 상에서 진행

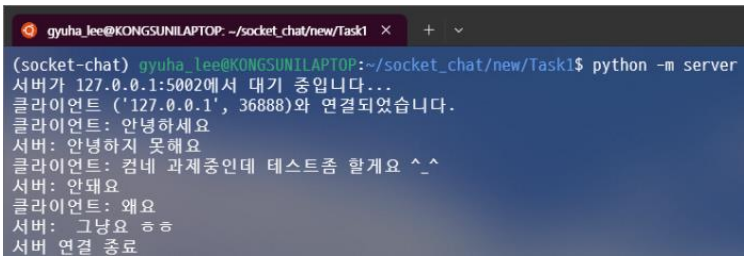
3. **TCP 통신 구조 설계**

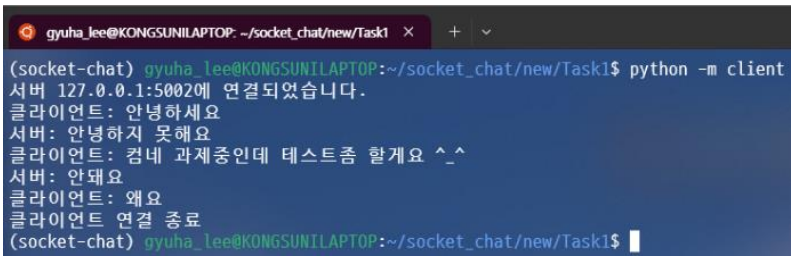
- 서버: 클라이언트의 연결 요청을 대기하고 이를 수락 -> 연결된 클라이언트와 메시지를 송수신하며, 연결 종료 시 리소스를 반환
- 클라이언트: 지정된 서버의 IP와 포트 번호를 통해 연결 요청 전송 -> 연결이 성립되면 사용자 입력 메시지를 서버로 전송하고, 서버의 응답 메시지를 출력
- 통신 흐름: TCP의 3-way handshake로 연결을 설정하고, FIN 패킷 교환으로 연결을 종료

4. **실제 구현 Point**

- 서버 개발: 소켓 생성 및 IP와 포트 바인딩 -> 클라이언트 연결 요청 대기 및 수락 -> 클라이언트와 메시지 송수신 기능 구현.
- 클라이언트 개발: 서버로 연결 요청 기능 구현 -> 사용자 입력을 통해 서버로 메시지 전송 및 응답 출력
- 통합 테스트: 로컬 네트워크 환경에서 서버와 클라이언트를 실행하여 메시지 송수신 테스트 -> 연결 종료 및 비정상적인 연결 해제 상황 처리 확인.
- 오류 처리: 연결 실패, 데이터 전송 실패 등의 상황에서 적절한 오류 메시지 출력.

5. **실제 구동 스크린샷** (좌: TCP 서버를 연 사용자 no.1, 우: 열린 서버에 참가한 사용자 no.2)





Task 2 : Task 1의 내용을 다음과 같이 개선해보았습니다.

1) 1 대 1 TCP 연결이 아닌 **Server-Client 방식으로 여러 Client를 수용하여 여러 명이 한번에 Chat service에 참여**할 수 있도록 구현.

2) 실제 서비스 중인 카카오톡이나 Instagram DM, ZOOM Meeting과 같이 단순 채팅 기능에 **여러 명령 기능 (닉네임 변경, 접속자 목록 표시, 채팅방 내 웹 검색, 연결 종료)**을 구현

2-1) 2에서 언급한 **'채팅방 내 웹 검색'**을 단순 검색 기능이 아닌, **OpenAI API와 엮어 생성형 AI의 검색 결과를 출력**하도록 구현

3) **채팅창의 UI 개선**(메시지가 전송된 날짜 및 시간 표시, 전송자 닉네임 표시)

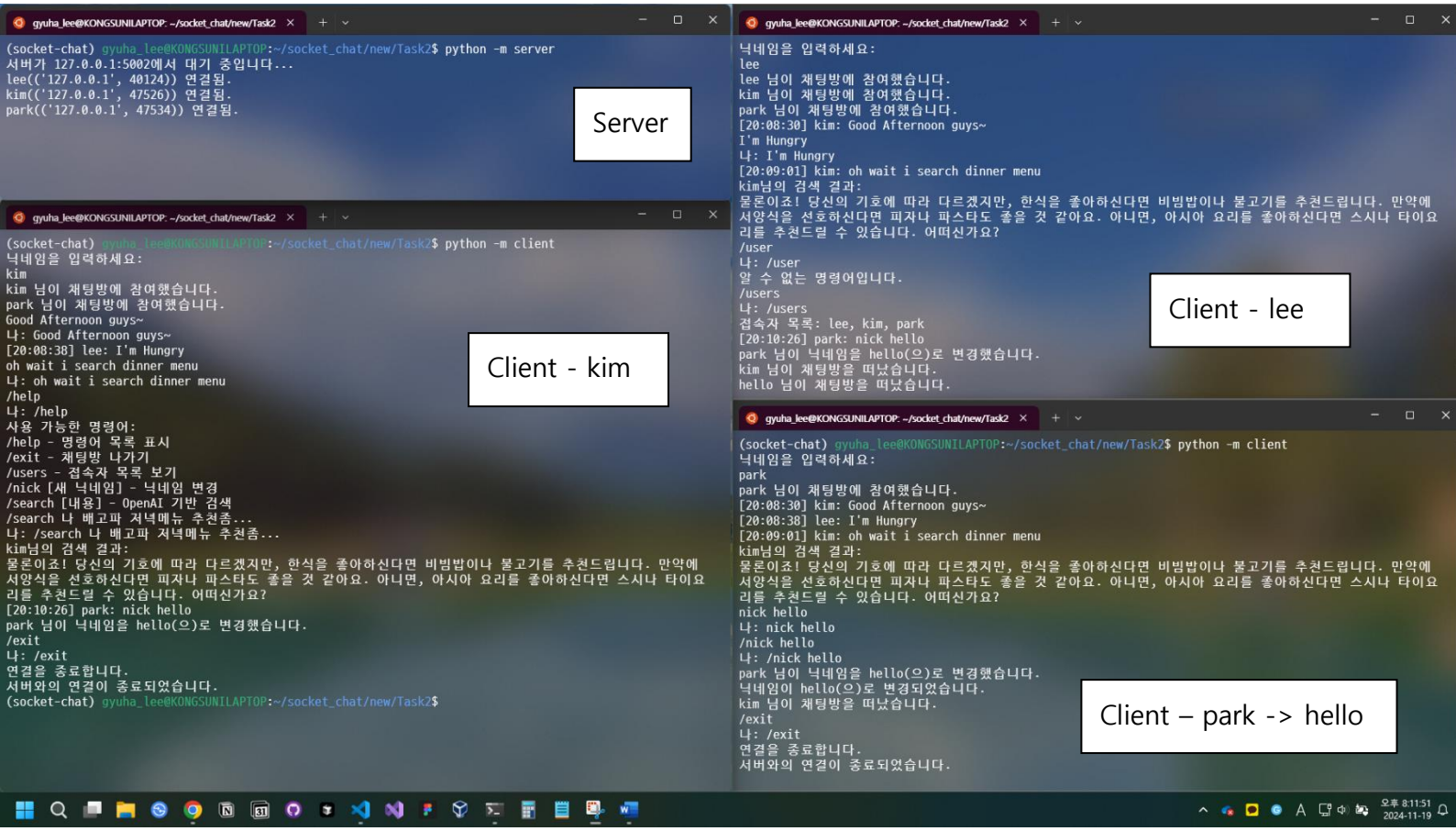
1. **Python 환경 설정 및 라이브러리 조사:** Python의 'socket' 모듈을 사용하여 TCP 통신 구현, 2번 개선점 구현을 위해 'openai', 여러 Client 수용을 위해 'threading', 메시지에 시간 표시를 위해 'datatime' 라이브러리를 사용함
2. **개발 환경:** Task 1과 동일
3. **서버-클라이언트 통신 구조 설계**

- 서버: 다중 클라이언트의 연결 요청을 수락하고, 클라이언트 간 메시지를 브로드캐스트 -> /search 명령어 입력 시 OpenAI API를 호출해 검색 결과를 생성하고 모든 클라이언트에게 전송.
- 클라이언트: 서버에 연결 요청을 보내고, 연결 성공 시 사용자 입력 메시지를 서버로 전송 -> 서버로부터 브로드캐스트된 메시지를 실시간으로 출력.
- 통신 흐름: TCP의 3-way handshake를 통해 연결을 설정하고, FIN 패킷 교환으로 연결을 종료, 클라이언트는 TCP 소켓을 통해 명령어 및 일반 메시지를 전송하며, 서버는 이를 처리 후 클라이언트들에게 브로드캐스트.

4. **실제 구현 Point**

- 서버 개발: TCP 소켓을 생성하고 IP와 포트를 바인딩 -> 클라이언트의 연결 요청을 수락하며, 멀티스레딩으로 각 클라이언트를 독립적으로 처리 -> 메시지 브로드캐스트와 명령어(/search, /help, /users, /nick) 처리를 구현, /search 명령어는 OpenAI API를 호출해 검색 결과를 생성하고 모든 클라이언트에 전송.
- 클라이언트 개발: TCP 소켓을 생성하고 서버의 IP와 포트로 연결 요청 전송 -> 사용자 입력 메시지를 서버로 전송하고, 서버로부터 받은 메시지를 실시간으로 출력 -> 종료 명령어(/exit)를 통해 연결을 종료.
- 통합 테스트: 서버와 클라이언트를 로컬 환경에서 실행해 메시지 송수신과 명령어 처리를 테스트 -> 다중 클라이언트 환경에서 메시지 브로드캐스트, 닉네임 변경, OpenAI API 호출 결과 전송 등을 확인.
- 오류 처리: 서버와 클라이언트 연결 실패, 데이터 전송 실패, OpenAI API 호출 실패 등의 상황을 처리하며 적절한 오류 메시지를 출력.

5. 실제 구동 스크린샷



Task 1와 2 수행 최종 결과

- Task 1으로 진행한 TCP 통신 프로그램은 1:1 연결 구조를 기반으로 Python의 socket 모듈을 사용하여 TCP의 연결 지향적인 특성을 활용하였으며, 클라이언트와 서버 간의 안정적인 데이터 전송을 보장하고자 하였습니다. 서버는 클라이언트의 연결 요청을 수락하고, 메시지를 수신 및 반송하며 종료 명령어를 통해 연결을 안전하게 해제하였습니다. 클라이언트는 사용자의 입력 메시지를 서버로 전송하고, 서버로부터 응답을 출력하는 구조로 동작하였습니다. 테스트 결과, TCP의 신뢰성과 데이터 전달의 정확성이 확인되었으며, 메시지 손실이나 왜곡 없이 안정적으로 동작하였습니다.
- Task 2는 Task 1을 바탕으로 다중 클라이언트 지원과 OpenAI API 기반 검색 기능 및 사용자의 편의 기능을 추가하여 기존의 TCP 1:1 통신 구조를 확장해보았습니다. 서버는 다수의 클라이언트가 연결할 수 있도록 멀티스레딩 구조를 적용하였으며, 각 클라이언트의 연결을 독립적으로 처리하도록 구현하였습니다. 클라이언트는 접속 시 닉네임을 설정하며, 서버로부터 실시간 메시지 브로드캐스트를 통해 다른 사용자들과 대화할 수 있습니다. 또한, /search 명령어를 통해 OpenAI GPT-4 모델을 호출하여 질문에 대한 답변을 생성하고, 이를 모든 클라이언트에게 전송하는 기능을 추가하였습니다. 이외에도 /users 명령어로 현재 접속 중인 사용자 목록을 확인하거나, /nick 명령어로 닉네임을 변경하는 등 다양한 기능을 구현하였습니다. 테스트 결과, 다중 클라이언트 연결과 메시지 브로드캐스트가 안정적으로 작동하였으며, OpenAI API를 통한 검색 결과가 모든 클라이언트에게 실시간으로 전송되는 것을 확인하였습니다. 클라이언트 종료 시 서버와의 연결이 안전하게 해제되는 점도 확인되었습니다.

- 향후 더 나은 개선 방향으로는 OpenAI API 호출의 응답 속도를 최적화하기 위한 캐싱 기능 추가, 채팅방 관리자 기능(예: 특정 사용자 강제 퇴장) 도입, 파일 전송 및 이모티콘 지원, GUI 기반의 사용자 인터페이스로의 확장을 통해 프로그램의 실용성과 사용자 경험을 더 높이면 좋지 않을까 싶습니다.

참고 문서

Beej's Guide to Network Programming. (n.d.). Beej's Guide to Network Programming: Using Internet Sockets. https://beej.us/guide/bgnet/

Stevens, W. R. (1994). TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley.

Python Software Foundation. (n.d.). Python 3.9.0 Documentation: socket — Low-level networking interface. https://docs.python.org/3/library/socket.html

Inria. (n.d.). Computer Networking: Principles, Protocols, and Practice. https://inl.info.ucl.ac.be/CNP3

OpenAI. (n.d.). OpenAI API Documentation. https://platform.openai.com/docs/

OpenAI. (n.d.). OpenAI Python SDK GitHub Repository. https://github.com/openai/openai-python

Python Networking Programming Tutorial. (n.d.). Real Python: Python Socket Programming for Beginners. https://realpython.com/python-sockets/