Projekt PSI

Raport wstępny

Temat projektu: chat tekstowy w architekturze p2p

Mateusz Krakowski

Marcin Łojek

Piotr Zając

Michał Kindeusz

10 stycznia 2023

1 Treść zadania

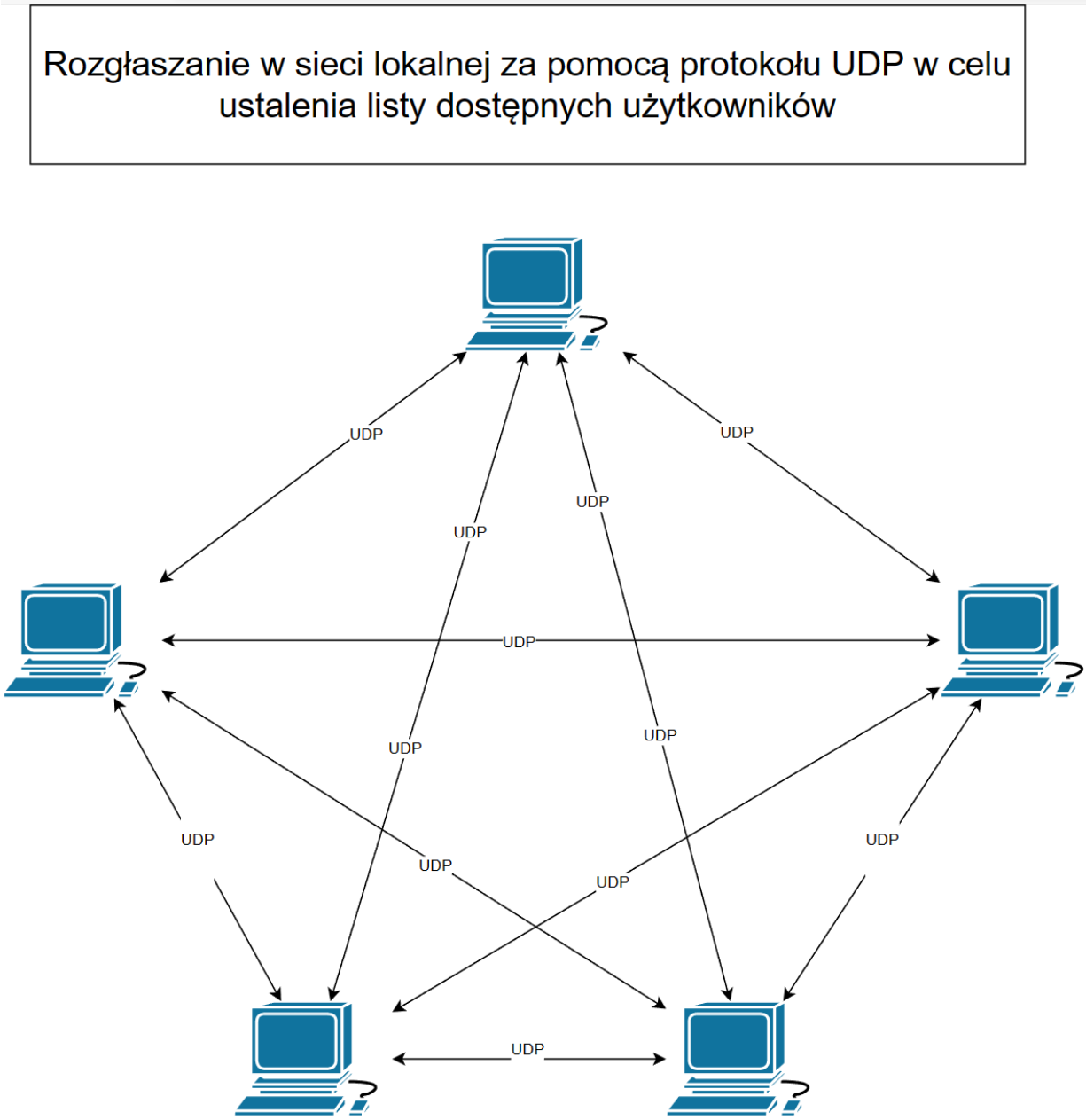
Chat tekstowy używający architekturę peer to peer i wykorzystujący rozgłaszanie w sieci lokalnej. Czyli każdy peer rozgłasza cyklicznie, że istnieje i żyje, a pozostali budują sobie listę aktywnych klientów, którą cyklicznie przeglądają i usuwają nieaktywnych klientów. Sama komunikacja odbywa się bezpośrednio pomiędzy rozmówcami. Można wprowadzić ograniczenie liczby klientów np. do 5, aby nie komplikować zbytnio kodu dynamicznymi strukturami.

2 Opis funkcjonalny black-box

Aplikacja terminalowa realizująca chat grupowy w architekturze peer to peer. Wymagania funkcjonalne:

1. Logowanie się użytkownika za pomocą nicku(domyślnie użytkownicy rozróżniani są przez adres IP)
2. Lista aktualnie dostępnych użytkowników
3. Wysyłanie wiadomości do wszystkich peerów
4. Historia wiadomości tylko dla aktualnej sesji aplikacji, bez historii
5. Brak sztywno narzuconej maksymalnej liczby peerów (Python domyślnie korzysta z dynamicznych struktur, więc nie będzie z tym problemu)

3 Opis i analiza poprawności stosowanych protokołów komunikacyjnych



W projekcie postanowiliśmy wykorzystać protokół UDP zarówno do rozgłaszania w sieci lokalnej w celu ustalenia listy dostępnych peerów, jak i samego rozsyłania wiadomości. Dzieje się to z tą rożnicą, że PINGI wysyłane są na adres rozgłoszeniowy (czyli do wszystkich peerów w sieci lokalnej), natomiast samo wysyłanie wiadomości jest wykonywane do wszystkich peerów z osobna(znamy ich adresy dzięki wcześniejszemu ustaleniu listy dostępnych peerów).

Flowchart działania peera znajduje się w oddzielnym pliku flowchartaplikacji.svg, opis w sekcji "zarys koncepcji implementacji".

4 planowany podział na moduły i strukturę komunikacji między nimi (być może z rysunkiem)

Podział na moduły:

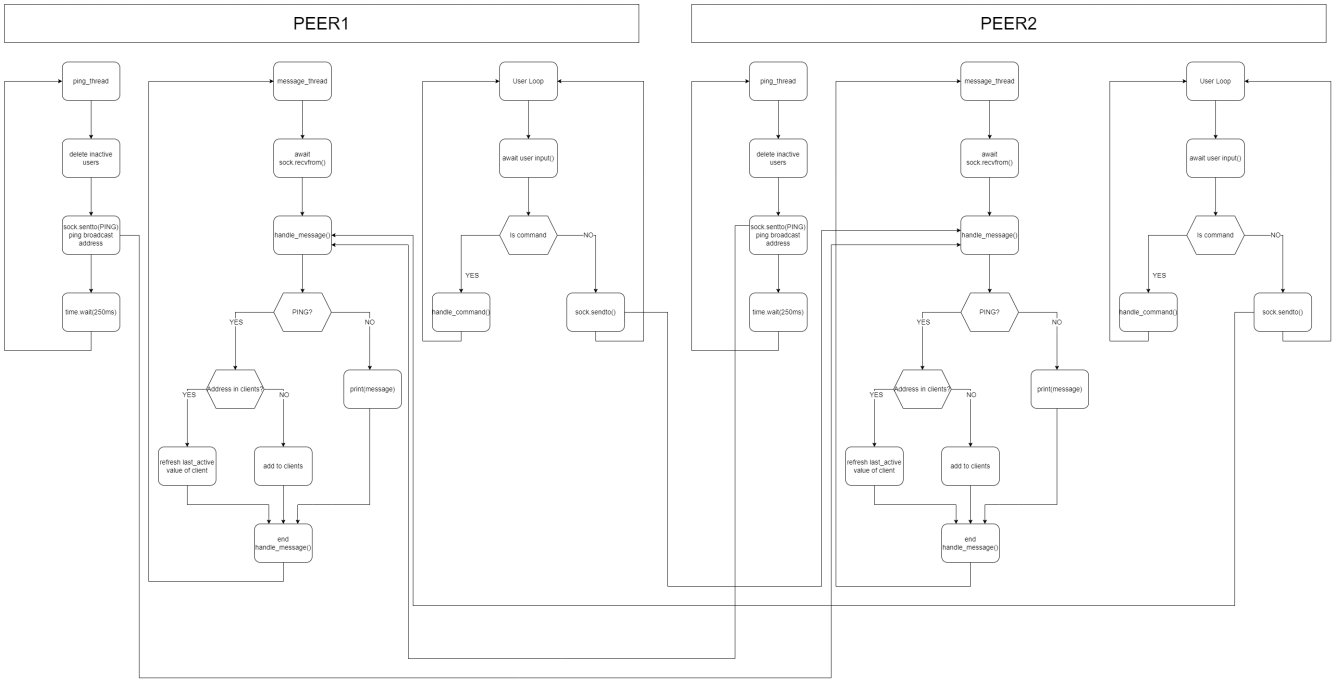
1. peer.py
    - funkcja informująca innych podłączonych klientów o aktywności użytkownika
    - funkcja do odbierania wiadomości od innych użytkowników
    - funkcja do rozsyłania wiadomości do reszty użytkowników
2. ui.py
    - funkcja do pobierania danych wejściowych od użytkownika
    - funkcja do wyświetlania wiadomości wymienionych z danym użytkownikiem
3. client.py
    - definicja klasy Client

    Moduł ui.py odpowiada za warstwę prezentacji, w niej pobierane są dane wejściowe dalej przekazywane do funkcji działających na wątkach zdefiniowanych w peer.py. Wstępnie warstwa prezentacji będzie oparta o terminal i Pythonowe funkcje input() i print(), ale w ramach rozwoju aplikacji jest możliwa zmiana na UI napisane w przeznaczonej do tego bibliotece. Moduł client.py pozwala nam na odseparowanie definicji klasy klienta od reszty kodu. Możliwe jest, że będzie konieczne poszerzenie klasy klienta o atrybuty opcjonalne takie jak nick itd.

5 Zarys koncepcji implementacji

Klasa Client posiada dwa najważniejsze atrybuty:

- address − adres IP i PORT na którym peer słucha
- lastactive − czas w którym ostatnio został odebrany ping od peera. Przechowywane w celu cyklicznego odświeżania listy klientów.



Flowchart w lepszej jakości przesyłamy jako załącznik w formacie svg.

Dla każdego z peerów, w celu budowania listy dostępnych hostów oraz przyjmowania wiadomości działają dwa wątki:

1. wątek pingthread odpowiedzialny za:
   - cykliczne pingowanie adresu rozgłoszeniowego z wykorzystaniem protokołu UDP, ping powinien dotrzeć do wszystkich aktywnych peerów w sieci
   - usuwanie nieaktywnych peerów z listy klientów z wykorzystaniem atrybutu lastactive

   Funkcja tego wątku jest wykonywana co interwał czasowy, wstępnie 250ms. Peer jest uznany za nieaktywny jeśli przez 5s nie wysłał on żadnego pinga.
2. wątek przyjmujący wiadomości(w tym pingi) i aktualizujący listę klientów. Dla klienta niewystępującego na liście, dopisuje go do listy, dla klienta występującego już na liście, aktualizuje wartość atrybutu lastactive. Wątek ten działa w nieskończonej pętli.

3. Dodatkowo potrzebujemy jeszcze jednej nieskończonej pętli, w której użytkownik będzie mógł wpisywać treść wiadomości lub korzystać z innych opcji programu(takich jak c!clients, c!nickname, c!exit). Po otrzymaniu inputu od użytkownika który nie zaczyna się od "c!", wiadomość zostaje rozesłana do każdego z aktywnych peerów z osobna.

   Do realizacji tego wykorzystamy jeden socket UDP wprowadzony w tryb broadcast dla każdego z peerów.

5.1 Użyte technologie i narzędzia

Chat zostanie zrealizowany w Pythonie. System operacyjny Linux Kubuntu. Użyte biblioteki:

- socket
- threading
- time