Katarzyna Chojniak 155978
Wiktoria Białasik 151436

Sprawozdanie Sieci Komputerowe 2
Komunikator Internetowy

1. Opis projektu

Celem projektu było stworzenie funkcjonalnej aplikacji komunikatora w architekturze klient-serwer, która umożliwia użytkownikom rejestrację, logowanie, tworzenie pokoi czatu, wymianę wiadomości w czasie rzeczywistym, zarządzanie listą znajomych oraz uczestnictwo w różnych pokojach czatu. Kod serwera napisany jest w języku C, natomiast klienta w Pythonie. Aplikacja wykorzystuje połączenie z bazą danych SQLite do przechowywania informacji o użytkownikach, pokojach czatu oraz relacjach między użytkownikami (możliwość dodawania znajomych). Do stworzenia interfejsu graficznego użyto biblioteki Tkinter. Do obsługi wielu użytkowników posłużono się wielowątkowością, korzystając z biblioteki pthreads.h.

2. Opis komunikacji pomiędzy serwerem i klientem

Schemat działania komunikacji klient-serwer opiera się na wymianie różnego typu wiadomości, wg ściśle określonych formatów, w zależności od funkcji. Komunikaty wysyłane są ze strony klienta automatycznie (np. zaraz po zalogowaniu) lub poprzez kliknięcie danego przycisku. Dla zaimplementowanych funkcjonalności przedstawia się to następująco:

Funkcja	Komunikat klienta	Działanie serwera	Odpowiedź serwera
Rejestracja	REGISTER username password	Sprawdza, czy nazwa użytkownika nie zaczyna się od słów kluczowych. Sprawdza, czy użytkownik już istnieje.	- Jeśli użytkownik istnieje: informacja o zajętym loginie - Jeśli nie: rejestracja udana
Logowanie	LOGIN username password	Sprawdza dane logowania w bazie danych. Jeśli dane poprawne, wysyła powiadomienie o dołączeniu nowego użytkownika do pokoju.	- Jeśli błędne dane: błędne dane logowania - Jeśli poprawne: zalogowano
Dołączanie do pokoju czatu	JOIN chatroom_name	Sprawdza, czy pokój czatu istnieje. Jeśli nie, dodaje go do bazy danych. Wysyła powiadomienie o dołączeniu nowej osoby do wszystkich aktywnych użytkowników w pokoju.	- Jeśli pokój istnieje: Dołączono do pokoju - Jeśli pokój nowy: Pokój utworzony
Aktywni użytkownicy w pokoju	ACTIVE_USERS chatroom_name	Sprawdza tabelę user_chatrooms i zwraca listę aktywnych użytkowników w pokoju.	Lista aktywnych użytkowników w pokoju
Dodawanie znajomych	ADD_FRIEND receiver	Sprawdza, czy istnieje już wpis w tabeli friends dla tej pary użytkowników (receiver+zalogowany). Użytkownicy stają się znajomymi jeśli oboje wysłali do siebie zaproszenie.	- Jeśli nie istnieje: dodaje status 1 (wysłane zaproszenie) - Jeśli istnieje: status 2 (użytkownicy stają się znajomymi)
Wyświetlanie listy znajomych	OK username	Wyszukuje użytkowników, z którymi nadawca ma status znajomości równy 2 w tabeli friends (znajomych).	Lista znajomych

3. Podsumowanie

Zaimplementowana aplikacja działa w opisany sposób, spełniając pierwotne wymagania projektowe, zarówno po stronie klienta jak i serwera. Serwer w prawidłowy sposób zarządza połączeniami, korzystając z gniazd TCP do wyżej opisanej komunikacji z klientem. Po uruchomieniu serwera za pomocą komendy:
gcc -Wall serverTCPmultithread.c -Isqlite3 sqlite3/sqlite3.c -pthread -o server.out
nasłuchuje on na porcie 1100 na przychodzące połączenia od klientów, którzy uruchamiają aplikację poprzez:
python3 client.py
Każde nowe połączenie realizowane jest jako osobny wątek, co pozwala na przetwarzanie wielu połączeń.

Podczas implementacji, problematycznym aspektem okazało się zarządzanie wielowątkowością aplikacji; zarówno na poziomie łączenia się poszczególnych użytkowników, jak i odświeżania oraz zarządzania poszczególnymi elementami interfejsu graficznego. Dotyczy to również automatycznego pobierania danych infromacji, np. o liście aktywnych użytkowników w danym pokoju. Kluczowym problemem była również implementacja logiki funkcjonalności dodawania znajomych; prawdopodobnie zaproponowany sposób przechowywania informacji o statusie znajomości jako tabeli w bazie danych (oraz konieczność wysłania zaproszenia przez obu użytkowników) nie jest optymalny, ale wystarczający na potrzeby zbudowanej aplikacji.