System komunikacji grupowej typu IRC

Iwo Naglik 136774 Bartosz Przybył 136785

1 Opis protokołu komunikacyjnego

Komunikacja pomiędzy klientem a serwerem w projekcie odbywa się w następującej konwencji:

- Wiadomości wysyłane do serwera od klienta
  - 0aaaaxxxx - operacja ta ma za zadanie sprawdzenie loginu w wektorze danych klientów. aaaa oznacza długość zaproponowanego nicku przez klienta, długość ta maksymalnie może wynieść 9999 bajtów. W przypadku, gdy długość ma mniej niż 1000 bajtów liczba zostaje dopełniona z lewej strony zerami. xxxx w tym przypadku określa nick użytkownika, którym chciałby się posługiwać podczas korzystania z aplikacji. Serwer po otrzymaniu takiej wiadomości sprawdza czy podany nick już czasem nie jest wykorzystany przez innego użytkownika. Jeśli jest wykorzystywany zwraca on klientowi wartość 0, w przeciwnym wypadku zwraca wartość 1
  - 1aaaaxxxx - operacja ta ma za zadanie odebrać wiadomość od danego klienta. aaaa w tym przypadku oznacza długość wysyłanej wiadomości. Podobnie jak w powyższym przypadku, gdy długość ta jest mniejsza niż 1000 bajtów wartość zostaje dopełniona z lewej strony zerami. xxxx jest w tej konwencji konkretną wiadomością. Nie ma potrzeby przesyłania nicku użytkownika, ponieważ serwer sam potrafi zidentyfikować na podstawie numeru deskryptora, który użytkownik wysłał wiadomość
  - 2x - operacja ta oznacza odebranie żądania klienta odnośnie zmiany pokoju, gdzie x oznacza numer pokoju, do którego klient chce się przenieść
  - 3 - wysłanie takiego znaku do serwera oznacza zamknięcie aplikacji przez klienta. Po otrzymaniu takiej wiadomości serwer kończy wątek dla danego klienta
- Wiadomości odbierane przez klienta od serwera
  - 0aaaaxxxxbbbbyyyy - serwer wysyła wiadomość do konkretnego klienta w założonej konwencji. Znaki aaaa i bbbb oznaczają odpowiednio długość nicku użytkownika wysyłającego wiadomosc i długość wiadomości wysyłanej przez użytkownika. Również w tym przypadku może nastąpić dopełnienie tej wartości nieznaczącymi zerami z lewej strony. Łańcuchy xxxx i yyyy są to odpowiednio nick klienta wysyłającego wiadomość i treść konkretnej wiadomości
  - 1aaaaxxxx - przesłanie takiej wiadomości do klienta oznacza dołączenie nowego użytkownika do pokoju, w którym znajduje się klient. Konwencja łańcuchów aaaa i xxxx jest taka sama jak w przypadku wiadomości wysyłanej przez klienta do serwera o początku '0'
  - 2aaaaxxxx - przesłanie takiej wiadomości do klienta oznacza opuszczenie przez użytkownika pokoju, w którym znajduje się dany klient. Konwencja łańcuchów jak w myślniku wyżej

2 Opis implementacji (struktury projektu)

2.1 Serwer

Główną strukturą wykorzystaną w projekcie jest struktura User, która zawiera takie pola jak:

- deskryptor
- nick użytkownika
- pokój, do którego użytkownik jest przypisany

Dane wszystkich użytkowników przechowywane są w wektorze. Ponadto program wykorzystuje również mutexy:

- `room_mutex[6]` - mutexy odpowiadające za synchronizację podczas współbieżnego wysyłania wiadomości do określonego pokoju
- `clients_mutex` - mutex odpowiadający za synchronizację podczas wykonywania współbieżnych operacji na wektorze clients

Serwer na samym początku wykonuje odpowiednie operacje sieciowe mające na celu możliwość połączenia się klientów do serwera. Wraz z każdym kolejnym zaakceptowanym klientem tworzony jest nowy wątek, w którym odbywa się cała komuniacja między klientem a serwerem. Po zamknięciu aplikacji przez klienta stworzony wątek zostaje zakończony oraz dane klienta zostają usunięte z wektora użytkowników.

2.2 Klient

Program klienta obsługuje w swoim ciele 3 wątki. Pierwszy wątek - główny obsługuje akcje użytkownika związane z graficznym interfejsem. Drugi wątek uruchamiany jest tylko w momencie wysłania komunikatu do serwera i kończy się po wykonaniu operacji. Trzeci wątek działa w pętli i odbiera odpowiednie wiadomości od serwera i wykonuje akcje związane z daną wiadomością. Wątek ten zostaje zakończony w momencie wyłączenia aplikacji.

3 Opis sposobu kompilacji i uruchomienia projektu

3.1 Kompilacja i uruchomienie serwera

Po pobraniu całego projektu należy przejść do katalogu Server za pomocą polecenia: `cd Server`

Gdy już jesteśmy w odpowiednim katalogu aby skompilować plik źródłowy z kodem serwera wykonujemy polecenie: `g++ -pthread -Wall server.cpp -o server`

Po przeprowadzonej kompilacji możemy uruchomić nasz serwer za pomocą polecenia: `./server 1234`, gdzie 1234 to numer portu, na którym ma działać aplikacja

3.2 Kompilacja i uruchomienie klienta

Po pobraniu projektu należy otworzyć folder Client w wybranym przez siebie środowisku obsługującym JavaFX. Następnie dodać konfigurację, przekompilować i uruchomić cały program.