Kamil, Kaczmarek 155971
Jakub, Butkiewicz 155833

Sprawozdanie sk2

Radio internetowe

1. Opis projektu

Założeniem naszego projektu, było stworzenie radia internetowego przy użyciu bsd_sockets. Serwer napisany w C++ przyjmuje połączenia od wielu klientów i strumieniuje do nich muzykę. Dla każdego klienta tworzone są nowe wątki, co sprawia że mogą być oni obsłużeni wszyscy na raz. Klient napisany w języku Python łączy się z serwerem, dzięki czemu może otrzymywać i puszczać pliki muzyczne. Co więcej, może on kontrolować pewne działania serwera - głównie chodzi o operacje na kolejce muzycznej, takie jak SKIP, SWAP oraz DELETE. Ostatnia funkcjonalność klienta to wgrywanie swoich plików mp3 na serwer, które automatycznie zostają dodane do kolejki. Do puszczania i kontrolowania muzyki, klient korzysta z biblioteki Pygame oraz Pyaudio. Jeśli chodzi o GUI, zostało one stworzone przy pomocy standardowej biblioteki Pythona - Tkinter.

2. Opis komunikacji pomiędzy serwerem i klientem

Kiedy klient łączy się z serwerem, tworzone są dla niego 3 gniazda sieciowe: control socket, streaming socket oraz update socket

a) **Control socket** - Serwer tworzy wątek, w którym otrzymuje różne żądania od klienta na tym gnieździe:

- FILE - uprzedzenie serwera o zamiarze załadowaniu nowego pliku do kolejki i wysłanie go
- QUEUECHANGE - prośba do serwera o wykonanie jednej z trzech operacji na kolejce
    - SKIP - Pominięcie aktualnie strumieniowanego pliku mp3
    - DELETE - Usunięcie dowolnego pliku z kolejki
    - SWAP - Zamiana dwóch dowolnych plików w kolejce
- END - powiadomienie serwera o zakończeniu połączenia

b) **Streaming socket** - To gniazdo odpowiada za przesył aktualnie granego pliku mp3 oraz moment, od którego ma być puszczony. Kiedy serwer otrzyma na nim wiadomość STREAM wysyła wszystkie potrzebne dane.

c) **Update socket** - to gniazdo odpowiada za ciągłe wysyłanie kolejki utworów do klientów. Dzięki temu są oni na bieżąco ze wszystkimi zmianami. Druga funkcjonalność tego gniazda to wysłanie rozkazu zaprzestania grania aktualnej piosenki, który był spowodowany jej pominięciem lub usunięciem.

3. Podsumowanie (0.5-1 strona) Najważniejsze informacje o implementacji Co sprawiło trudność

Pierwszą z rzeczy, która była dla nas lekko kontrowersyjna to użycie aż trzech gniazd. Jednak pozwoliło to na łatwiejszą synchronizację wątków oraz pomogło to z problemem blokujących się gniazd, dlatego zdecydowaliśmy się wybrać tę opcję.

Kolejnym problemem okazało się, że gdy klient wysyła do serwera wiadomość, np. FILE, a następnie od razu zaczyna wysyłanie pliku(czyli kolejne wykonania metody send). Zdarzało się, że serwer odbierał kilka wiadomości na raz, co powodowało błędy programu. Wpadliśmy na dwa pomysły jak rozwiązać ten problem. Albo wysyłać wiadomości z etykietami, aby serwer wiedział co ma zrobić z daną informacją, albo wykorzystać handshake, czyli po każdej odebranej informacji, serwer odsyła potwierdzenie. Wybraliśmy drugą opcję ze względu na prostotę implementacji.

Największym problemem było strumieniowanie muzyki. Początkowo serwer otwierał plik i wysyłał na bieżąco surowe dane, w częściach. Klient odbierał te segmenty i odtwarzał je za pomocą playera pygame. Niestety dźwięk pozostawiał wiele do życzenia. Zacinał się co prawdopodobnie było spowodowane, czasem który player potrzebował do załadowania części pliku. Pojawiły się także szumy w tle, które pojawiły się wskutek przesyłania surowego pliku mp3. Pygame prawdopodobnie nie radził sobie z odtwarzaniem takich segmentów, zwłaszcza że mogły one posiadać inne dane, np. metadane, tagi itd. Następnie spróbowaliśmy użyć biblioteki ffmpeg w c++, która miała odpowiadać za przesyłanie stricte muzycznych segmentów pliku mp3. Jednak po zaimplementowaniu tej opcji, dalej mieliśmy problem z zacinaniem się utworu. Wtedy uznaliśmy, że prostszą drogą będzie przesył plików mp3 w całości, kiedy klient poprosi o granie muzyki. Aby zasymulować strumieniowanie serwer iteruje przez czas trwania danego utworu i wysyła aktualny moment utworu oraz sam utwór, kiedy klient będzie tego potrzebował.

Podsumowując, ten projekt miał wiele problemów, z którymi jakoś musieliśmy sobie poradzić. Nie jesteśmy pewni, czy zabiegi takie jak użycie aż 3 gniazd jest czymś poprawnym ze standardem, ale uznaliśmy że jest to najprostsze rozwiązanie naszego problemu z synchronizacją i blokującymi gniazdami.