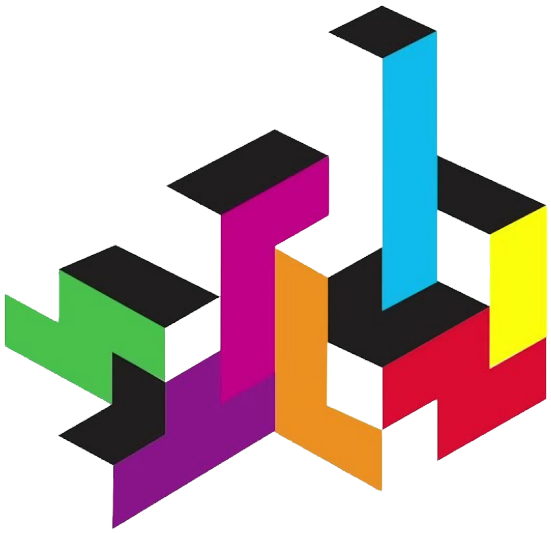
Tetris Multiplayer

Dokumentacja protokołu komunikacji serwera z klientem

Kornel Domeradzki - May 26, 2019



TETRIS
MUTLIPLAYER

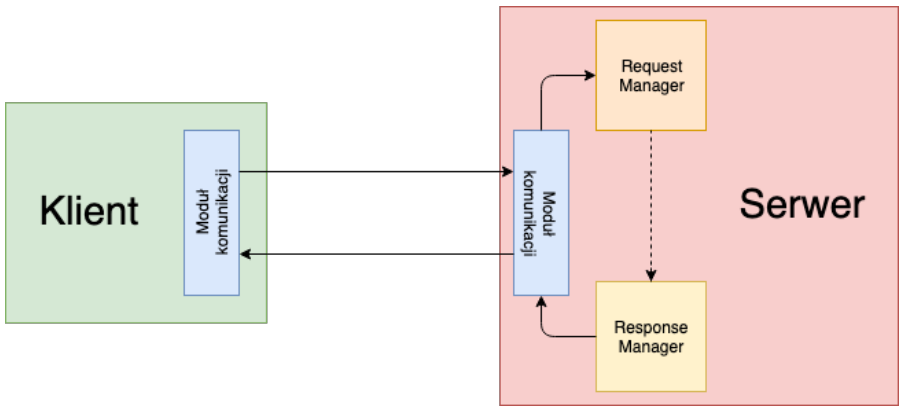
Wstęp

Do przesyłu danych w tym projekcie został utworzony **nowy protokół warstwy aplikacji** mający na celu przesyłanie danych pomiędzy klientem oraz serwerem przy użyciu **TCP** lub **UDP** - typ gniazda może zostać ustawiony w pliku ustawień aplikacji klienckiej oraz wybrany podczas uruchomienia serwera. Program kliencki został napisany przy użyciu **C++**, biblioteki **SFML** oraz graficznego **API OpenGL** a program serwerowy został stworzony przy użyciu **Javy (11),** sterownika do bazy danych MySQL oraz jej standardowej biblioteki. Do serwera zostało także wykorzystane narzędzie automatyzujące Maven.

Opis działania protokołu komunikacji

Protokół komunikacji klienta gry z serwerem działa **w ten sam sposób** przy użyciu gniazda UDP oraz TCP, ponieważ wszystkie funkcje niskiego poziomu oraz mechanizmy przesyłu zostały **wydzielone** do innego modułu. Takie podejście umożliwiło na znacznie łatwiejsze rozbudowanie projektu na wiele typów gniazd (*szczegóły na temat modułu odpowiedzialnego za komunikację gniazd TCP oraz UDP są podane w dalszej części*).

Podstawowy przykład działania protokołu komunikacji wygląda tak: klient wysyła zapytanie, które składa się z nazwy kodu zapytania - *np.: **LOGIN**, **GET_USER_DATA*** oraz treść z informacją na temat zapytania - *np.: **'login hasło'**, **'nazwa_użytkownika'***. Zapytanie jest odczytywane przez serwer i przesyłane do modułu **RequestManager**, który je sprawdza, przetwarza oraz jeżeli jest ono prawidłowe oraz użytkownik ma uprawnienia do danej akcji, wykonuje je. Niektóre zapytania wymagają odpowiedzi oraz dokonania jakichś akcji tuż przed wysłaniem. W takim przypadku używany jest moduł **ResponseManager**, który wysyła wiadomość zwrotną do klienta, który wcześniej dokonał zapytania. ResponseManager teoretycznie może być używany do wysłania wiadomości bez otrzymania wcześniejszego zapytania od podanego klienta aczkolwiek w takich wypadkach używany jest po prostu bezpośredni dostęp do sesji (*więcej o tym w szczegółach na temat modułu odpowiedzialnego za komunikację gniazd TCP oraz UDP*). W skrócie powyższy przepływ danych można opisać poniższym diagramem:



Spis kodów zapytań i odpowiedzi

Poniżej można znaleźć spis wszystkich użytych kodów zapytań oraz odpowiedzi.

Kody zapytań (kody wysyłane od klienta)

Numer Kodu Zapytania	Kod Zapytania	Zawartość Zapytania	Krótki opis
0	LOGIN	login hasło	Przesyła dane logowania użytkownika serwerowi (po czym oczekuje na odpowiedź od serwera)
1	LOGOUT	login	Przesyła informacje o zakończeniu gry oraz prośbę o zamknięcie sesji
2	GET_USER_DATA	Login	Przesyła zapytanie o informacje danego użytkownika
3	GAME_SEARCH	-	Przesyła zapytanie o wrzucenie użytkownika do kolejki gry (rozpoczyna wyszukiwane przeciwnika do gry)
4	CANCEL_SEARCH	-	Przesyła zapytanie o usunięcie zalogowanego gracza z kolejki gry (nieużyte)
5	ABORT_GAME	-	Przesyła zapytanie o przerwanie trwającej rozgrywki rozgrywanej przez zalogowanego gracza
6	GET_UPDATE_LOGS	-	Przesyła zapytanie o informacje na temat wszystkich najnowszych aktualizacji

Kody odpowiedzi (kody wysyłane do klienta)

Numer Kodu Odpowiedzi	Kod Odpowiedzi	Zawartość Odpowiedzi	Krótki Opis
0	INCORRECT_CREDENTIALS	login	Przesyła odpowiedź powiadamiającą o niepoprawnych danych logowania
1	CORRECT_CREDENTIALS	login	Przesyła odpowiedź powiadamiającą o poprawnych danych logowania
2	GAME_SETUP	elo grupa_przywilejów wygrane_nierankingowe przegrane_nierankingowe wygrane_rankingowe przegrane_rankingowe punkty_tetromino czas_gry nazwa_uzytkownika	Przesyła odpowiedź powiadamiającą o dobranym przeciwniku i rozpoczętej grze
3	SEND_USER_DATA	elo grupa_przywilejów wygrane_nierankingowe przegrane_nierankingowe wygrane_rankingowe przegrane_rankingowe punkty_tetromino czas_gry nazwa_uzytkownika	Przesyła odpowiedź z danymi pewnego użytkownika
4	SEND_UPDATE_LOGS	(nagłówek_aktualizacji, zawartość_aktualizacji, autor_aktualizacji, data_aktualizacji) * 5	Przesyła odpowiedź z informacjami na temat wszystkich najnowszych aktualizacji

Komunikacja gniazd TCP oraz UDP oraz Sesje

Projekt do komunikacji może wykorzystywać gniazda TCP lub UDP. By wybrać jeden typ gniazda dla aplikacji klienckiej należy wejść w plik z ustawieniami (ścieżka: $(ProjectDir)/Resources/settings.conf) oraz podmienić ostatnią wartość w pliku tekstowym - 1 oznacza gniazdo TCP a 0 oznacza gniazdo UDP. W przypadku serwera jedyne co trzeba zrobić to wybrać gniazdo gdy zapyta się o to serwer. Gniazd nie można mieszać - to znaczy,

że nie może zajść sytuacja w której klient z gniazdem TCP próbuje się połączyć do serwera działającego na gniazdach UDP.

Poprzez pojęcie ***sesja*** w tym projekcie rozumie się stałe połączenie pomiędzy klientem oraz serwerem dzięki, któremu obydwa programy mogą przesyłać między sobą informacje. Utworzenie klasy sesji umożliwia nam przesył danych niezależnie od używanego gniazda. Dla sesji używających UDP zaimplementowano oddzielny system przesyłania oraz otrzymywania danych co było zmuszone naturą działania gniazd (UDP jest protokołem bezpołączeniowym, nie ma mechanizmów kontroli przepływów i retransmisji oraz w celu ignorowania informacji przesyłanych poprzez niepożądane aplikacji).