Projekt SK

Konrad Baumgart, Jan Borowski

8 stycznia 2012

1 Treść zadania

Projekt 4: Komunikator internetowy typu GG

2 Protokół sieciowy

Każdy klient identyfikowany jest przez swój 2-bajtowy dodatni numer ID, używa swojego hasła do logowania.
Użytkownik ma tylko 2 statusy: zalogowany i niezalogowany.

Logi Serwer wypisuje dane diagnostyczne na standardowe wyjście błędów.

Zapewnienie odczytu pełnych paczek z danymi Za każdym razem zarówno serwer jak i klient poprzedza przesyłane dane informacją o ich długości (w bajtach) zawartą w 2 bajtach. Nie wlicza się w tę długość 2 bajtów przechowujących ilość danych. Ogranicza to na przykład maksymalną długość wiadomości w systemie.
Pierwszą częścią danych jest zawsze kod operacji o długości 1 bajta.
Wszelskie ciągi znaków wysyłamy bez znaku 0 na końcu.
Przesyłając liczby wysyłamy najpierw mniej znaczące bajty, potem bardziej znaczące.

Postępowanie w przypadku otrzymania nieprawidłowych danych Gdy serwer otrzyma od klienta dane niezgodne z niniejszą specyfikacją, rozłącza danego klienta.

Port serwera Serwer nasłuchuje pakietów TCP na porcie 4790.

Łączenie Nawiązując połączenie z serwerem klient może albo się zarejestrować, albo zalogować.

Rejestracja KOD: 1
Można się rejestrować jedynie tuż po nawiązaniu połączenia. Klient przesyła swoje hasło, musi ono mieć conajmniej 3 bajty(znaki). Serwer odpowiada **KOD: 1** wysyłając 2 bajty: numer ID w przypadku następnie rozłącza się.

Logowanie KOD: 2
Klient wysyła swoje ID (2 bajty) do serwera, po nim zaś swoje hasło. Serwer odpowiada **KOD: 2** wysyłając 1 bajt: 1 w przypadku sukcesu lub kończy połączenie w przypadku błędu.

Zakończenie połączenia Zarówno klient jak i serwer mogą zakończyć połączenie po prostu kończąc połączenie TCP.

Poniższe komendy można wykonać tylko będąc zalogowanym.

Sprawdzenie dostępności znajomych KOD: 3
Klient wysyła kilka ID (2-bajtowe, jedno po drugim) do serwera. Serwer odpowiada **KOD: 3** wysyłając wielokrotność 3 bajtów: dla każdego ID o które wystosowano zapytanie 2 bajty zajmuje to ID, zaś w trzecim bajcie jest kod dostępności danego użytkownika.
Można sprawdzić dostępność samego siebie.
Można sprawdzić dostępność maksymalnie 1000 osób.

Wysłanie wiadomości KOD: 5
Klient wysyła 2 bajty - ID odbiorcy, a potem wysyłaną wiadomość
Serwer nie informauje nadawce o powodzeniu operacji wysłania.
Serwer wysyła **KOD: 5** do odbiorcy 2 bajty - ID nadawcy, a potem wysyłaną wiadomość.
Jeżeli odbiorca jest niezalogowany, wiadomości do niego są przechowywane na serwerze i są mu przesyłane w momencie gdy się zaloguje.

3 Realizacja

3.1 Serwer

Stworzyliśmy serwer używając poznanych na zajęciach socketów BSD używając języka C++. By przechowywać dane o użytkownikach, używamy dynamicznych kontenerów z STL. Kod serwera jest krótki, więc pozwoliliśmy sobie go zamknąć w pojedynczym pliku *main.cpp*.

> *Code is the ultimate documentation.*

Wielowątkowość Mieliśmy problem z wieloątkowością - gdybyśmy dla każdego połączenia tworzyli nowy wątek, to z tegoż nowego wątku nie moglibyśmy bezpośrednio wysyłać wiadomości do użytkowników, którzy się zalogowali po nas - nie mielibyśmy dostępu do gniazd dla nich stworzonych. Dlatego nie tworzyliśmy nowych wątków, a użyliśmy funckji *select* by oczekiwać na możliwość odczytu lub zapisu do wielu gniazd.

Buforowanie wiadomości Aby można było wysyłać wiadomości do niezalogowanych użytkowników, na serwerze trzymamy je na mapie *bufferedMessages* w postaci kolejki całych pakietów do wysłania identyfikowanej przez ID użytkownika, do którego mają zostać wysłane. Jeżeli odbiorca zerwie połączenie gdy przesyłamy mu wiadomość, to zostaje ona utracona.

Odczytując dane z gniazda lub zapisując do niego, możemy nie odczytać/zapisać całego pakietu. Dlatego też zawsze gdy odczytujemy dane z gniazdka, wpisujemy je do *readBuffor*a, później zaś, gdy odbierzemy cały pakiet, kopiujemy go i usówamy z readBuffora. Podobnie, wysyłając dane do klienta, zapisujemy je do *writeBuffor*a i staramy się sukcesywnie przesyłać.

3.2 Klient

Klienta stworzyliśmy w języku Java, gdyż byliśmy w stanie potem uruchamić go w środowisku Linux oraz Windows, a także stworzyć wersję dla urządzenia mobilnego z systemem Android.

4 Obsługa programu

4.1 Serwer

Skompilowany przy (użyciu programu make) serwer uruchamia się w konsoli. Po uruchomieniu można obserwować wyjście diagnostyczne na standardowym wyjściu błędów. By łagodnie wyłączyc serwer wystarczy wysłać doń SIGUSR1.

4.2 Klient

Klient uruchamiany jest na maszynie wirtualnej Java. Intuicyjny graficzny interfejs użytkownika pozwala zarejestrować się, zalogować i kożystać z komunikatora.