Sprawozdanie z projektu zaliczeniowego laboratorium sieci komputerowych II

System wymiany komunikatów typu publish/subscribe

Bartłomiej Szymkowiak 141324
Mateusz Ollek 141291

1. Opis protokołu komunikacyjnego

Każda wysyłana pomiędzy klientem a serwerem wiadomość zawiera znak końca danych “~“.
W wiadomościach zawierających różne od siebie informacje są one rozdzielone znakiem “|”.
Serwer nigdy nie wysyła sam wiadomości, zawsze czeka na odpowiednie zapytanie od klienta.

Połączenie z klientem zostaje rozwiązane (a tym samym deskryptor gniazda klienta w kodzie serwera zostaje usunięty) dopiero kiedy program klienta zostanie wyłączony. W szczególności samo wylogowanie użytkownika nie zamyka połączenia.

Logowanie użytkownika:
Klient wysyła wiadomość: LOGIN|HASŁO~
Po odebraniu wiadomości serwer odsyła jedna z trzech informacji zwrotnych:
Logged in~ lub Incorrect login or password!~ lub User is already logged in!~

Uzyskanie listy wszystkich tematów:
Klient wysyła wiadomość: t|~
Po odebraniu wiadomości serwer odsyła:
NAZWA_TEMATU1|ID_TEMATU1|NAZWA_TEMATU2|ID_TEMATU2... |~

Uzyskanie listy subskrybowanych przez użytkownika tematów:
Klient wysyła wiadomość: m|~
Po odebraniu wiadomości serwer odsyła:
NAZWA_TEMATU1|ID_TEMATU1|NAZWA_TEMATU2|ID_TEMATU2... |~

Wylogowanie użytkownika:
Klient wysyła wiadomość: o|~
Serwer po odebraniu wiadomości niczego nie odsyła

Zamknięcie programu klienta / rozłączenie z serwerem:
Klient wysyła wiadomość: ?|~
Serwer po odebraniu wiadomości niczego nie odsyła

Zasubskrybowanie / odsubskrybowanie tematu:
Klient wysyła wiadomość: s|ID_TEMATU~
Serwer po odebraniu wiadomości odsyła: You subscribed to the topic~ lub You unsubscribed to the topic~

Dodanie nowego tematu przez użytkownika:
Klient wysyła wiadomość: n|NAZWA_TEMATU~
Serwer po odebraniu wiadomości odsyła: Topic inserted~ lub This topic alredy exists~

Wysłanie wiadomości do tematu:
Klient wysyła wiadomość: e|ID_TEMATU|TYTUŁ_WIADOMOŚCI|TREŚĆ_WIADOMOŚCI~
Serwer po odebraniu wiadomości odsyła: Message sent~ lub Message not sent~

Odebranie wiadomości od serwera:
Klient wysyła wiadomość: r|~
Serwer odsyła: LICZBA_POZOSTAŁYCH_WIAD|NAZWA_TEMATU|ID_TEMATU|TYTUŁ|WIADOMOŚĆ~

2. Opis implementacji

Serwer jest napisany w języku C i jego współbieżne działanie opiera się na funkcji select. Po uruchomieniu serwera wczytywane są tematy wraz z ich subskrybentami, nie wysłane jeszcze wiadomości oraz dane kont użytkowników. Serwer wywołuje funkcję read do odczytu wiadomości aż nie napotka na znak "~". Na końcu każdej wiadomości jest dodawana "~". Wiadomości tematyczne są wysyłane tylko do użytkowników, którzy subskrybowali dany temat podczas wysyłki wiadomości. Następnie główna pętla programu w skrócie wygląda następująco:

1) Deskryptory z masek pomocniczych są kopiowane do głównych masek rmask i wmask
2) Wykonywana jest funkcja select
3) Obsługiwane są deskryptory pozostałe w masce rmask, następuje ich przetwarzanie (odbiór zapytań od klienta) i dodanie ich do masek pomocniczych pogrupowanych według typu wiadomości jaką trzeba odesłać
4) Obsługiwane są deskryptory pozostałe w masce wmask tzn. następuje przygotowanie i wysyłanie wiadomości

Serwer jest kompilowany z poszczególnych plików:

- server.c - główny plik sewera z pętlą główną programu
- fun_serv.c / fun_serv.h- plik z funkcjami wykorzystywanymi przez serwer oraz jego plik nagłówkowy
- init.c / init.h - plik z funkcjami do wczytywania, aktualizowania i usuwania danych z plików oraz plik nagłówkowy
- structs.h - plik z wykorzystywanymi strukturami oraz dyrektywami include
- folder "messages" zawiera nie rozesłane jeszcze wiadomości, folder "topics" wszystkie tematy
- users.txt - plik z danymi kont użytkowników w formacie: LOGIN HASŁO NICK

Klient jest napisany w języku C# z użyciem .NET. Połączenie jest nawiązywane wywołując BeginConnect, wysyłki i odbiory poszczególnych wiadomości są wykonywane asynchronicznie używając BeginSend i BeginReceive. Odbiór danych jest realizowany aż do napotkania znaku "~" lub zwrócenia niedodatniej wartości przez EndReceive. Wszystkie metody zmieniające stan kontrolek korzystają z Invoke.

Klient składa się z poszczególnych plików:

- Program.cs - główny plik programu uruchamiający formularz FormConnect
- FormConncet.cs - plik z implementacją formularza do nawiązania połączenia z serwerem
- Form1.cs - główny formularz klienta z menu, w którym są otwierane i wyświetlane inne formularze
- FormLogin.cs - formularz implementujący etap logowania użytkownika
- FormReceive.cs - formularz implementujący odbiór wiadomości
- FormSendMsg.cs - formularz implementując wysyłanie wiadomości
- FormTopics.cs - formularz z implementacją przeglądania, subskrybowania oraz tworzenia tematów

3. Sposób kompilacji

Serwer kompiluje się uruchamiając skrypt.sh lub za pomocą polecenia:
`gcc -Wall server.c structs.h init.c init.h fun_serv.c fun_serv.h -o server`
Następnie wystarczy go uruchomić: `./server`

Projekt klienta jest kompilowany w VisualStudio. Można go uruchomić włączając plik wykonywalny projektSK2.exe w folderze projektSK2/bin/Debug.

4. Obsługa programów

Serwer po samym uruchomieniu jest już gotowy do działania. Ewentualnie można wcześniej dodać nowe konta użytkowników w nowej linii pliku users.txt według szablonu: LOGIN HASŁO NICK

Po uruchomieniu klienta należy w oknie Connect podać adres serwera jeżeli jest inny niż standardowy oraz nawiązać połączenie. Następnie widoczny jest ekran logowania. Po zalogowaniu dostępny jest odbiór wiadomości w zakładce "Receive messages". Odbierana i wyświetlana jest jedna wiadomość. Następne wiadomości odbiera się za pomocą "Next message". Po naciśnięciu w menu "Send message" można wysłać wiadomość wybierając subskrybowany temat z rozwijanej listy, uzupełniając tytuł oraz treść wiadomości i naciskając przycisk "Send". Po wejściu w zakładkę "Topics" widoczna jest lista wszystkich tematów, subskrypcje są dodatkowo oznaczone w checkboxie. Po naciśnięciu na nazwę tematu na liście następuje jego subskrybowanie lub rezygnacja z subskrypcji. W tej zakładce można również dodać nowy temat podając jego nazwę w polu "Topic name:" i klikając przycisk "Add". W menu dostępne są jeszcze przyciski do wylogowania "Log out" oraz zamknięcia klienta "Exit".