Projekt zaliczeniowy
Sieci Komputerowe II

System równoważenia obciążenia dla grupy serwerów obliczeniowych

Autorzy:	**Agata Nowicka**	inf127308	agata.r.nowicka@student.put.poznnan.pl
	Tomasz Paluszkiewicz	inf127289	tomasz.paluszkiewicz@student.put.poznnan.pl

1 Protokół komunikacjyny

Komunikacja między serwerem a slave'ami oraz serwerem a klientem opiera się na nagłówku o stałej wielkości, w którym jest zakodowany rozmiar drugiej części wiadomości. W nagłówku znajdują się:

- req_type – zmienna oznaczająca typ wiadomości
- size – rozmiar danych przsyłanych po nagłówku
- key – klucz, który można wykorzystać do np. autoryzacji

Serwer reaguje na 4 typy wiadomości (zmienna req_type w nagłówku):

- req_cnt – prośby o połączenie,
- req_snd – przysłania pliku,
- req_rcv – prośby o przesłanie pliku
- req_res – odpowiedzi na zapytanie

W odpowiedzi serwer przesyla odpowiednie wiadomości.
Dane przesyłane po nagłówku mogą być jedną z trzech struktur: RData_Connect, RData_File, RData_Response. Programy wiedzą w jaki sposób interpretować dane dzięki zmiennej req_type zawartej w nagłówku.
Do zakodowania i odkodowania przesyłanych danych służą funkcje:

- req_send z req_encode i req_receive z req_decode na linuksie
- req_encode i req_decode – w protocol.cs w aplikacji na Windows

Dokładne struktury używane w protokole komunikacyjnym są zawarte w plikach protocol.h (serwery na Linux) oraz protocol.cs (klient na Windows)

2 Implementacja

2.1 Krótki opis kodu źródłowego serwera – access_node

- access_node.c – główny plik programu serwera
- an_connection_handling.h – zawiera deklaracje funkcji odpowiedzialnych za obsługę klientów i slave'ów (definicje w an_connection_handling.c)

- an_connection_lists.h – zawiera deklaracje funkcji odpowiedzialnych za zarządzanie listami połączonych klientów i slave'ów (definicje w an_connection_lists.c)
- queue.h – zawiera deklaracje struktury i funkcji do obsługi kolejek zadań na serwerze (definicje w queue.h)
- protocol.h – zawiera deklaracje oraz definicje struktur służących do przechowywania danych do komunikacji
- tools.h – zawiera deklaracje funkcji kodujących oraz dekodujących dane oraz funkcje pomocnicze (definicje w tools.c)

2.2 Krótki opis kodu źródłowego slave'a – slave_node

- slave_node.c – główny plik programu slave'a
- slave_tools.h – zawiera deklaracje funkcji do wykonywania skryptów przez slave'a (definicje w slave_tools.c)
- protocol.h – zawiera deklaracje oraz definicje struktur służących do przechowywania danych do komunikacji
- tools.h – zawiera deklaracje funkcji kodujących oraz dekodujących dane oraz funkcje pomocnicze (definicje w tools.c)

2.3 Krótki opis kodu źródłowego klienta – CN_project

Funkcje do obsługi interfejsu graficznego rozpoczynają się od "setThreaded" i znajdują się na początku pliku Form1.cs

Funkcje związane z komunikacją i funkcjnalnością:

- void feedback(ref Protocol.Request response) – obsługa odpowiedzi od serwera
- void ReceiveRestCallback(IAsyncResult ar) – kończy odbiór danych i przekazuje informacje do feedback
- void ReceiveCallback(IAsyncResult ar) – rozpoczyna pobieranie treści lub pobiera pozostałość nagłówka
- void SendCallback(IAsyncResult ar) – rozpoczyna odbiór nagłówka lub wysyła resztę wiadomości do serwera
- void sendRequest(ref Protocol.Request req) – rozpoczyna wysyłanie danej wiadomości
- void ConnectCallback(IAsyncResult ar) – kończy operację połączenia do serwera
- void GetHostEntryCallback(IAsyncResult ar) – zbiera informacje o wybranym serwerze i próbuje nawiązać połączenie, następnie wysyła dane o sobie i czeka na odpowiedź od serwera
- void taskDone(ref Protocol.RData_File datafile) – zapisuje odebrany plik i uaktualnia dane o zadaniach

Następne funkcje służą do obsługi kliknięcia na przycisk (buttonConnect_Click(), buttonBrowse_Click() , buttonSend_Click() , buttonReceive_Click()).

3 Kompilacja i obsługa

Aby skompilować access-node serwer oraz slave serwer należy wpisać make w katalogu projektu. W celu kompilacji tylko jednego z nich należy wpisać make access_node dla serwera access_node lub make slave_node dla serwera slave_node. Aby usunąć folder ze skompilowanym programem należy wpisać make clean.

Aplikacja dla klienta powinna być skompilowana i otwierana w Microsoft Visual Studio. Opis kompilacji jest również zawarty jest w pliku README.md w głównym katalogu projektu.