Sieci komputerowe 2

Zdalne wyłączanie systemów operacyjnych

Sprawozdanie z projektu

Zuzanna Ławniczak 151835 gr. lab. 2

Zamysł

Projekt jest w architekturze klient-serwer. Konsolowe programy napisane w języku C++ pozwalają na komunikację między komputerami w sieci.
Główny serwer obsługuje całą komunikację.
Agenci to proste programy nasłuchujące na rozkazy od serwera (np na wyłączenie). Ich dane są zapisywane na serwerze.
Klienci to interakcyjne programy konsolowe, które pozwalają wykonać wiele komend opisanych poniżej.

Pliki projektu

mainserver.cpp

Uruchamianie: ./ms.out

Główny program to serwer współbieżny, który zarządza całą komunikacją między klientami a agentami. Uruchamiany jest na porcie 1444. Nasłuchuje na połączenia od klientów.
Na konsoli serwera wyświetlane są logi.
Na komendę cn dodaje nowego agenta do wektora, który jest atrybutem klienta.
Na komendę sd wyszukuje wskazanego agenta w wektorze i wysyła mu rozkaz wyłączenia się.
Na komendę st używając select łączy się z wszystkimi agentami, których utworzył klient. Ich stan odsyła klientowi.

client.cpp

Uruchamianie: ./c.out <ip serwera> <port>

Kod klienta, który wyświetla w konsoli menu i pozwala użytkownikowi na 4 rozkazy:
cn <nazwa agenta alfanumeryczna> <adres IP> <port> - pozwala na dynamiczne dodanie nowego agenta.
st - pozwana na wyświetlenie statusu wszystkich agentów (włączony, wyłączony), do których uprawnienia ma klient.
sd <nazwa agenta> - pozwala na wyłączenie agenta.
exit - wyłącza program.

agent.cpp

Uruchamianie: ./a.out <port>

Agent to prosty program nasłuchujący na wiadomości od serwera, których odebranie potwierdza. Na komendę **sd** wyłącza się.

Komunikacja klient-serwer

Klienci ustanawiają połączenie z serwerem głównym. Wysyłają mu wiadomości w takim samym formacie jak komendy. Dla komendy sd i cn, serwer wysyła potwierdzenie w postaci krótkiego łańcucha (“a\n”’). Dla komendy sd serwer wyszukuje agenta w wektorze i ustanawia z nim połączenie TCP. Przesyła mu wiadomość “sd” (od shutdown). Agent potwierdza otrzymanie wiadomości i wyłącza się.
Na komendę st serwer dodaje wszystkich agentów do zbioru fds w mechaniźmie select i ustanawia timeout na 5 sekund. Następuje próba połączenia z każdym. Jeśli się uda, serwer zapisuje sobie, ze agent jest włączony. Jeśli nastąpi timeout, serwer zapisuje sobie, że z agentem nie udało się połączyć i jest wyłączony. Następnie ten ciąg informacji przzesyła do klienta, który przekazuje te informacje na wyjście.
Wiadomości są kończone znakiem nowej linii i stąd klient, serwer, agent wiedzą, gdzie się kończy wiadomość dla nich.

Użyte ciekawsze funkcje

W projekcie zwróciłam szczególną uwagę na wielowątkowość i buforowanie, zapoznawszy się z zagadnieniem i kodem z wykładu. Również użyłam do sprawdzania czy agenci są włączeni funkcji select.
Użyte też zostały gniazda sieciowe nieblokujące. Wszystkie połączenia są strumieniowe (TCP), nie użyto nigdzie komunikacji datagramowej. Wprowadzona została też obsługa wielu podstawowych błędów, które udało się wyłapać.

Napotkane problemy

Dużym problemem było napisanie takiej komunikacji, by klienci i serwery wysyłali wiadomości bez zacinania się, tzn. wysyłanie wiadomości tak, by nie było problemów z buforowaniem.
Innym problemem było pisanie w C++ i tak naprawdę używanie funkcji stworzonych pod C. Dzięki sporemu doświadczeniu w tym języku, dość szybko udało się pokonać te wyzwanie.

Wnioski

Pisanie serwerów i klientów w C++ jest bardzo interesujące i na pewno było przydatnym doświadczeniem. Pozwala to też na rozmyślanie nad tym, jak bardzo profesjonalni musieli być autorzy BSD sockets, bo przecież na tym opiera się współczesny świat.