Sprawozdanie z zadania na SK2

Zdalne wyłączanie komputerów

Sebastian Michoń 136770

1 Ogólna zasada działania

1. Node - komputer do wyłączenia - podaje serwerowi(ip serwera i port jako pierwsze 2 argumenty) identyfikator(3. argument)
2. Serwer uzyskuje identyfikator od node'a i przechowuje go w tablicy
3. Klient zadaje zapytanie o zbiór identyfikatorów: te z wymienionych, które są zapisane w serwerze zostają wysłane do klienta.
4. Klient może wysłać polecenia shutdown i reset z podanym czasem i identyfikatorami do wyłączenia do serwera, ten wykonuje dane polecenie na podanym zbiorze identyfikatorów.

2 Protokół komunikacji

1. Klient zarządzający (client) i klient do wyłączenia (node) porozumiewają się z serwerem z użyciem protokołu TCP - zależy mi na niezawodności przesłania identyfikatora node'a i polecenia od clienta, a także przesłania na sprzęt node'a odpowiedniej komendy wyłączającej sprzęt
2. Client po zainicjowaniu połączenia przesyła nasłuchującemu serwerowi - a ściślej wątkowi, który z tego serwera wychodzi bufor tekstu w następującym formacie:
   komenda
   (czas) <lista identyfikatorów>
   Gdzie:
   komenda - jedna cyfra - 0, jeśli shutdown, 1, jeśli reset, 2, jeśli poszukiwanie identyfikatora.
   czas - tylko jeśli komenda to 0 albo 1 - czas, za jaki ma zostać wykonana komenda.
   <lista identyfikatorów> - jeśli komenda=2, lista identyfikatorów, które chcę zdobyć, żeby uzyskać je (możliwość wykonania na nich operacji shutdown i reset). Jeśli 2 - lista identyfikatorów, na których ma zostać wykonane podane polecenie.
3. Serwer po parsingu tych danych wysyła wszystkim node'om powiązanym z danymi identyfikatorami bufor tekstu, w którym znajduje się komenda, którą klient ma wykonać na systemie.
4. Wszystkie komunikaty mają długość 1024 znaków

3 Struktura serwera

1. Serwer tworzy socketa, manipuluje jego opcjami(setsockopt), wiąże tego socketa z portem 1234 i adresem, następnie zaczyna nasłuchiwać na tym sockecie do 20 połączeń i je akceptować - nawiązane połączenia obsługuje funkcja handleConnection.
2. Funkcja handleConnection przekazuje nowemu wątku wykonującemu funkcję ThreadBehavior deskryptor, z którym może się komunikować i wraca do maina.

3. Funkcja ThreadBehavior odbiera od klienta bufor tekstu: jeśli zaczyna się on od litery n - dane pochodzą od node'a, identyfikator jest dodawany na wolną pozycję w tablicy, następnie blokuję wątek funkcją pthread_cond_wait - po wysłaniu sygnału na zmienną warunkową związaną z tym waitem node otrzymuje polecenie od serwera, które ma wykonać.

4. Jeśli bufor zaczyna się od cyfry 2, jest to zapytanie od klienta o identyfikatory - jeśli identyfikator znajduje się w tablicy identyfikatorów serwera, to jest on wysyłany z powrotem; jeśli nie, to nie.

5. Jeśli bufor zaczyna się od cyfry 0 albo 1, jest to zapytanie o wyłączenie określonego zbioru node'ów powiązanych z odpowiednimi identyfikatorami - serwer zapisuje odpowiednie polecenia do tablicy poleceń, a następnie wysyła sygnał na zmienną warunkową.

4 Struktura klienta

1. W aplikacji okienkowej można podać serwer i port, z którym chcę się skomunikować, ponadto czas, za jaki chcę zrestartować/wyłączyć komputer i identyfikatory, które chcę uzyskać.

2. Wpisanie nazw identyfikatorów oddzielonych spacjami przy podaniu adresu i portu serwera wysyła zapyanie do serwera o podane identyfikatory używając funkcji outer_processing - jeśli istnieją, trafiają na listę po lewej stronie; jeśli nie, są pomijane.

3. Po kliknięciu na przyciski reset/shutdown uruchamia się trigger odpowiadający za przedsięwzięcie odpowiednich operacji po stronie node'a dla identyfikatorów zaznaczonych na liście identyfikatorów - zamraża on wszystkie przyciski poza Cancelem, przechodząc do funkcji outer_processing. Przycisk Cancel zamyka okno i związane z nim operacje.

4. funkcja outer_processing wsadza dane od gui do struktury i wrzuca je do nowego wątku wykonującego funckję parse_connection, wychodząc z funkcji bez pthread_joina - dzięki temu aplikacja się nie wiesza.

5. funkcja parse_connection tworzy socketa, a następnie łączy się z podanym przez użytkownika adresem i portem. Dalej wywołuję funkcję zarządzającą połączeniem handleConnection i zamykam socketa.

6. Funkcja handleConnection zdobywa informacje w określonym formacie od funkcji attain_wisdom o użytkowniku i jego prawach, wysyła je do serwera i otrzymuje odpowiesź - polecenie do wykonania na komputerze, na którym uruchomiony jest klient.

7. Timer odczytuje co 1 sekundę informację o tym, czy thread się zakończył - jeśli tak, to wykonywane jest polecenie wysłane przez serwer, odblokowywane są przyciski.

8. Przycisk "remove identificators" pozwala usunąć wybrane na liście identyfikatory z listy.

5 Struktura node'a

1. Po podaniu serwera, portu i identyfikatora sprawdzana jest nazwa identyfikatora; jeśli nie składa się wyłącznie ze znaków alfanumerycznych, proces się kończy.

2. Następnie tworzony jest wątek i nawiązywana jest komunikacja z serwerem w taki sam sposób, jak w przypadku zwykłego klienta.

3. Node wysyła serwerowi identyfikator, następnie oczekuje na komunikaty zwrotne - jeśli zaczyna się takowy od ! - error, podany identyfikator jest już używany, jeśli nie od ! - polecenie, które node ma wykonać na swoim komputerze.

6 Uruchomienie

1. Uruchomienie serwera:
   cd ...(miejsce, gdzie zaszyty jest projekt)
   gcc ser.c -o ser -Wall -l pthread
   ./ser

2. Uruchomienie klienta:
   cd .../QtServerClientApp (w miejscu, gdzie jest projekt)
   qmake
   make
   ./QtServerClientApp

3. Uruchomienie node'a:
   cd ...(miejsce, gdzie zaszyty jest projekt)
   gcc node.c -o node -Wall -l pthread
   ./node localhost 1234 Random
   #Wiąże node'a z identyfikatorem o nazwie Random - wywołanie po stronie klienta w polu "Identificators to add" nazwy Random i kliknięcie przycisku "Add identificators" skutkuje dodaniem togo identyfikatora do listy po lewej stronie.