1 Wstęp

Celem projektu jest zaprojektowanie i implementacja systemu umożliwiającego rozproszone przetwarzanie na komputerach w sieci. Za pomocą zaimplementowanego przez nas rozwiązania zaistnieje możliwość wykonywania skryptów na zdalnym komputerze oraz zbieranie danych wytworzonych poprzez wywołane polecenia. Dodatkowe informacje będą zapewniały logi zapisywane i przechowywane w bazie danych serwera. System zostanie napisany w języku C++ z wykorzystaniem technik obiektowych, a komunikacja sieciowa zostanie zaimplementowana z wykorzystaniem systemu gniazd.

2 Nadzorca

Jest wielowątkowym programem, który na przemian wysyła wiadomości do serwera, oraz nasłuchuje odpowiedzi od niego (każde z połączeń z serwerem w oddzielnym wątku). Umożliwia również wydanie polecenia zakończenia wykonywania wszystkich zadań. Do komunikacji z każdym z serwerów tworzona jest inna instancja nadzorcy. Instancja nadzorcy ginie, gdy serwer zakończy wykonywanie wszystkich zleconych zadań.

3 Serwer

Serwer jest dwuwątkowym programem. Jeden wątek służy do wykonywania kolejnych zadań. Drugi ma za zadanie komunikować się z nadzorcą; wysyłać mu raporty o wykonanych zadaniach, oraz przyjmować kolejne polecenia. Jeżeli wątek wykonujący jest już zajęty, nowo otrzymane polecenia są umieszczane w kolejce i realizowane po kolei. Po wykonaniu danego zadania i utworzeniu binarnego pliku wynikowego, serwer wysyła nadzorcy ten plik.

4 Komunikacja

Używamy protokołu TCP. Serwery cały czas nasłuchują, nowo stworzona instancja nadzorcy odzywa się do serwera, któremu chce coś zlecić. Polecenia skierowane do serwerów są wysyłane binarnie, podczas gdy wyniki są przesyłane w plikach binarnych. Po zleceniu zadań, nadzorca zamyka połączenie, przestawia się w tryb nasłuchu i czeka na binarny plik wynikowy od serwera.

4.1 Wiadomości

Podstawą komunikacji jest wiadomość binarna będąca strukturą składającą się z:

```
1 struct Message {
2   int type;
3   void *content;
4 };
```

Listing 1: struktura wiadomości

4.2 Możliwe typy wiadomości wysyłanej przez nadzorcę

- init - rozpoczęcie komunikacji przez nadzorcę, w treści podany jest numer portu na którym będzie nasłuchiwał nadzorca
- execute - w treści znajdują się polecenia dla serwera
- kill:all - wymusza na serwerze natychmiastowe zaprzestanie wykonywania wszystkich
- kill - wymusza na serwerze natychmiastowe zaprzestanie wykonywania zadania podanego w treści wiadomości
- end - informuje serwer o zakończeniu pracy nadzorcy

4.3 Możliwe typy wiadomości wysyłanej przez nadzorcę

- init:received - potwierdzenie portu, na którym będzie nasłuchiwał nadzorca
- execute:received - potwierdzenie otrzymania zadań
- execute:started - informacja o rozpoczęciu wykonywania zadań
- execute:stopped - informacja o zakończeniu wykonywania wszystkich zadań (treścią jest wynik wykonania operacji)
- killed - informacja o zaprzestaniu wykonywania zadań, w treści wiadomości info o zabitym zadaniu, lub dopisek 'all'

4.4 Kolejność wiadomości

1. N init
2. S init:received
3. N execute
4. S execute:received
5. S execute:started
6. N execute
7. S execute:received
8. N execute
9. S execute:received
10. S execute:started
11. S execute:started
12. S execute:stopped
13. N kill (opcjonalnie)

14. S killed (jako odpowiedź na kill)
15. N end

5 Logi

Cała historia wymiany wiadomości pomiędzy nadzorcą i serwerem będzie zapisywana w logach. Logi z kolei, będą przechowywane w bazie danych. Pozwoli to na wydajniejsze przeglądanie logów w porównaniu do tradycyjnej metody zapisywania logów w plikach.

Zapisywane są następujące informacje:

- typ wiadomości
- treść wiadomości (w przypadku, gdy treścią komunikatu jest plik z poleceniami lub plik z danymi, zapisywana będzie jedynie ścieżka do pliku lub treść wiadomości nie będzie przechowywana w ogóle)
- data wysłania/otrzymania wiadomości,
- informacja o serwerze