**Dokumentacja Projektu**

**Igor Bukowski, s31489, Gr. 36c**

**1. Wstęp**

Aplikacja implementuje system rozproszony oparty na protokołach TCP i UDP, który obsługuje komunikację klient-serwer, wykonywanie podstawowych operacji arytmetycznych oraz analizę metryk działania. Głównym celem projektu jest stworzenie systemu do obsługi zapytań od wielu klientów jednocześnie, umożliwiającego wykonywanie operacji na liczbach z użyciem wielowątkowości oraz dynamicznego raportowania statystyk.

**2. Opis funkcjonalności**

Aplikacja uruchamiana jest komendą:

java -jar CCS.jar <port>

* : Numer portu, na którym serwer będzie nasłuchiwał połączeń (TCP i UDP).

Po uruchomieniu serwer realizuje następujące funkcje:

1. **Usługa UDP**:
   * Nasłuchuje na zadanym porcie i odbiera komunikaty od klientów.
   * Obsługuje protokół wykrywania usług („CCS DISCOVER” -> „CCS FOUND”).
2. **Usługa TCP**:
   * Akceptuje wielu klientów jednocześnie, wykorzystując pulę wątków.
   * Przetwarza komendy arytmetyczne („ADD”, „SUB”, „MUL”, „DIV”).
   * Obsługuje błędy (np. niepoprawne komendy, dzielenie przez zero).
3. **Monitorowanie metryk**:
   * Liczba aktywnych klientów.
   * Całkowita liczba operacji.
   * Statystyki operacji (ADD, SUB, MUL, DIV).
   * Liczba błędów.
   * Sumaryczny wynik wszystkich operacji.
   * Raportowanie co 10 sekund.

**3. Szczegółowy opis protokołów**

Ogólne założenia

**Protokół UDP (User Datagram Protocol) jest** wykorzystywany w aplikacji do zapewnienia prostego mechanizmu wykrywania serwera przez klientów oraz przesyłania komunikatów między nimi. UDP jest bezpołączeniowym protokołem, co oznacza, że nie wymaga nawiązywania trwałego połączenia między nadawcą a odbiorcą, co czyni go szybszym, ale mniej niezawodnym niż TCP.

Funkcjonalności w aplikacji

1. Rozgłaszanie serwera:
   * Serwer w trybie UDP nasłuchuje na określonym porcie.
   * Klient wysyła komunikaty inicjujące w celu wykrycia serwera w sieci lokalnej.
   * Jeśli serwer odbierze komunikat z treścią "CCS DISCOVER", odpowiada komunikatem "CCS FOUND", wskazując swoją dostępność.
2. Obsługa komunikacji:
   * Komunikaty są przesyłane w postaci ciągów znaków.
   * Każda wiadomość jest opakowana w datagram i wysyłana do serwera, który następnie przetwarza dane i wysyła odpowiedź.
3. Najważniejsze cechy:
   * Brak gwarancji dostarczenia wiadomości.
   * Minimalne opóźnienia dzięki niskim narzutom protokołu.
   * Przeznaczenie do wykrywania serwera w lokalnej sieci i przekazywania krótkich wiadomości.
4. Schemat działania:
   * Klient: Wysyła wiadomość "CCS DISCOVER" na określony port serwera.
   * Serwer: Odbiera wiadomość, a następnie odpowiada wiadomością "CCS FOUND" na adres i port klienta.

**Protokół TCP**

**Ogólne założenia**

Protokół TCP (Transmission Control Protocol) jest używany w aplikacji do obsługi bardziej zaawansowanej komunikacji między klientami a serwerem. TCP zapewnia niezawodne przesyłanie danych poprzez mechanizm potwierdzeń (ACK), retransmisję utraconych pakietów i kontrolę kolejności przesyłanych danych.

**Funkcjonalności w aplikacji**

1. **Obsługa połączeń z wieloma klientami:**
   * Serwer otwiera gniazdo nasłuchujące na określonym porcie TCP.
   * Klienci nawiązują połączenia z serwerem, przesyłając polecenia do przetwarzania.
   * Każde połączenie jest obsługiwane przez osobny wątek, co umożliwia równoczesną komunikację z wieloma klientami.
2. **Przesyłanie poleceń:**
   * Klienci wysyłają komendy w formacie tekstowym, np. ADD 10 5.
   * Serwer analizuje komendę, wykonuje obliczenia i zwraca wynik klientowi.
   * Obsługiwane operacje: dodawanie, odejmowanie, mnożenie, dzielenie.
3. **Zarządzanie metrykami:**
   * Serwer rejestruje liczbę klientów, operacji, błędów i sumę wyników.
   * Metryki są regularnie raportowane w konsoli, co ułatwia monitorowanie stanu aplikacji.
4. **Schemat działania:**
   * Klient: Nawiązuje połączenie z serwerem i wysyła polecenie w formacie tekstowym.
   * Serwer: Przetwarza polecenie, zwraca wynik lub błąd, a następnie rejestruje metryki operacji.
   * Klient: Odbiera odpowiedź i kończy działanie po przesłaniu wszystkich komend.
5. **Obsługa wyjątków:**
   * Serwer i klient obsługują błędy sieciowe oraz niepoprawne polecenia, zwracając odpowiednie komunikaty błędów.

**4. Szczegóły implementacyjne**

**Główne klasy:**

1. **CCS**:
   * Uruchamia usługi UDP i TCP.
   * Zarządza pulą wątków dla klientów TCP.
   * Obsługuje komunikację UDP w osobnym wątku.
2. **Client**:
   * Reprezentuje klienta TCP.
   * Przetwarza komendy i zwraca wyniki do klienta.
   * Obsługuje błędy, np. niepoprawne komendy lub dzielenie przez zero.
3. **MetricsAnalyzer**:
   * Zarządza metrykami serwera (liczba klientów, operacji, błędów itp.).
   * Odpowiada za periodyczne raportowanie statystyk.

**Obsługa wielowątkowości:**

* Pulę wątków tworzą instancje ExecutorService.
* Klienci TCP są obsługiwani jednocześnie w osobnych wątkach.
* Usługa UDP i raportowanie statystyk działają w osobnych wątkach.

**Raportowanie statystyk:**

Raport generowany jest co 10 sekund i zawiera informacje takie jak:

* Liczba aktywnych klientów.
* Liczba operacji (ADD, SUB, MUL, DIV).
* Liczba błędów.
* Suma wyników operacji.

**5. Instalacja i uruchomienie aplikacji**

**Wymagania:**

* Java JDK 8 lub nowsza.

**Kompilacja:**

javac CCS.java

**Uruchomienie:**

1. Uruchom serwer:

java -jar CCS.jar <port>

1. Wysyłaj zapytania klientów z użyciem klienta TCP lub narzędzi sieciowych (np. nc, telnet).

**Testowanie:**

* Testy można przeprowadzić, uruchamiając wiele klientów TCP, którzy przesyłają komendy.
* Użyj narzędzia do generowania zapytań UDP, by sprawdzić działanie usługi wykrywania.

**6. Problemy i możliwości rozwoju**

**Problemy:**

1. Brak uwierzytelniania klientów.
2. Brak obsługi retransmisji pakietów w protokole UDP.
3. Pojedynczy punkt awarii – serwer stanowi centralny punkt komunikacji.

**Możliwości rozwoju:**

1. Dodanie uwierzytelniania klientów.
2. Implementacja redundancji dla serwera.
3. Rozszerzenie protokołu o kolejkowanie i buforowanie zapytań.
4. Wsparcie dla szyfrowania komunikacji.