Zadanie UDP: Rozproszony System Uśredniający
SKJ (2024)

Wstęp

Zadanie polega na napisaniu aplikacji realizującej *Rozproszony System Uśredniający*.

Sposób działania aplikacji

Na aplikację składa się jeden program implementujący klasę DAS (*Distributed Averaging System*) uruchamiany poleceniem:

`java DAS <port> <number>`

gdzie `<port>` jest liczbą określającą numer portu UDP, a `<number>` to liczba całkowita. Aplikacja działa w dwóch trybach: *master* i *slave*. Wybór trybu odbywa się automatycznie w momencie uruchomienia aplikacj na podstawie stanu bieżącego systemu oraz podanego parametru. Po uruchomieniu aplikacja próbuje utworzyć gniazdo UDP otwierając port UDP o numerze zadanym parametrem `<port>`. Operacja ta może mieć dwojaki wynik:

- **Aplikacji udaje się otworzyć żądany port.** W tym przypadku aplikacja wchodzi w tryb *master*.
- **Aplikacji nie udaje się otworzyć portu ponieważ jest on już zajęty.** Zakłada się, że taka sytuacja może zaistnieć tylko wtedy, gdy na danej maszynie już pracuje jedna instancja tej aplikacji pracująca w trybie *master*. W takim przypadku aplikacja ta podejmuje pracę w trybie *slave*.

Tryb master

Aplikacja pracujaca w trybie master zapamiętuje wartość przekazaną jako parametr `<number>` a następnie **cyklicznie** w pętli odbiera komunikaty przez gniazdo na porcie `<port>`. Dalsze działanie zależ od wartości liczby otrzymanej w komunikacie:

- Jeśli wartość ta jest różna od 0 oraz -1, proces wypisuje tę wartość na konsolę oraz zapamiętuje ją.

- Jeśli wartość ta jest równa zero, proces kolejno:
  1. Wylicza wartość średnią z wszystkich niezerowych liczb, które odebrał od początku swojej pracy, uwzględniając w tym wartość `<number>`.
  2. Wypisuje te wartość na konsolę.
  3. Korzystając z gniazda, na którym pracuje, wysyła komunikat rozgłoszeniowy do wszystkich komputerów w swojej sieci lokalnej na port o numerze `<port>` (czyli taki sam na jakim pracuje) zawierający wyliczoną wartość średnią.
- Jeśli wartość ta jest równa -1, proces kolejno:
  1. Wypisuje tę wartość na konsolę.
  2. Korzystając z gniazda, na którym pracuje, wysyła komunikat rozgłoszeniowy do wszystkich komputerów w swojej sieci lokalnej na port o numerze `<port>` (czyli taki sam na jakim pracuje) zawierający otrzymaną wartość (-1).
  3. Zamyka używane gniazdo oraz kończy pracę.

Tryb slave

Aplikacja pracująca w trybie slave tworzy gniazdo UDP otwierając port UDP o losowym numerze (polegając w tym zakresie na systemie operacyjnym) a następnie korzystając z tego gniazda, wysyła do procesu pracującego na tej samej maszynie na porcie o numerze `<port>` komunikat zawierajacy wartość parametru `<number>`. Po wykonaniu tej czynności proces kończy pracę.

Wymagania i sposób oceny

1. W celu realizacji zadania należy zaprojektować i napisać proces implementujący własny protokół komunikacyjny, umożliwiający realizację opisanej powyżej funkcjonalności. Kwestię tego, jakiej treści pakiety są przesyłane między poszczególnymi procesami pozostawia się autorowi.
2. Proces jest uruchamiany z dwoma parametrami zgodnymi z powyższą specyfikacją. Jeśli liczba parametrów jest niepoprawna lub nie są one poprawnymi liczbami, proces ma zgłosić błąd i zakończyć pracę.
3. Poprawny i pełny projekt wart jest **400 punktów**. Za zrealizowanie poniższych funkcjonalności można otrzymać punkty do podanej wartości:
   - **maksymalnie 100 punktów** za uzyskanie funkcjonalności aplikacji pracującej jedynie w trybie slave.
   - **maksymalnie 200 punktów** za uzyskanie funkcjonalności aplikacji pracującej w trybie slave oraz w trybie master potrafiącej jedynie odbierać komunikaty.

- **maksymalnie 400 punktów** za uzyskanie funkcjonalności aplikacji pracującej w trybie slave oraz w trybie master potrafiącej zarówno odbierać komunikaty jak i wysyłać je za pomocą rozgłaszania w sieci lokalnej (pełna funkcjonalność).

4. Aplikację piszemy w języku Java zgodnie ze standardem Java 8 (JDK 1.8). Do komunikacji przez sieć można wykorzystać jedynie podstawowe klasy do komunikacji z wykorzystaniem protokołu UDP.

5. Projekty powinny zostać zapisane do odpowiednich katalogów w systemie EDUX w nieprzekraczalnym terminie 08.12.2024 (termin może zostać zmieniony przez prowadzącego grupę).

6. Spakowany plik projektu powinien obejmować:

   - Plik *Dokumentacja(nr.indeksu)Zad1.pdf*, opisujący, co zostało zrealizowane, co się nie udało, jak zainstalować, gdzie ewentualnie są błędy, których nie udało się poprawić. W szczególności, plik musi zawierać szczegółowy opis zaprojektowanego i zaimplementowanego protokołu (brak opisu protokołu lub jego fragmentaryczność może spowodować znaczące obniżenie oceny rozwiązania zadania).
   - Pliki źródłowe (dla JDK 1.8) (włącznie z wszelkimi bibliotekami nie należącymi do standardowej instalacji Javy, których autor użył) - aplikacja musi dać się bez problemu skompilować na komputerach w laboratorium w PJA.

   UWAGA: PLIK Z DOKUMENTACJĄ JEST WARUNKIEM KONIECZNYM PRZYJĘCIA PROJEKTU DO OCENY.

7. Prowadzący oceniać będą w pierwszym rzędzie poprawność działania programu i zgodność ze specyfikacją, ale na ocenę wpływać będzie także zgodność wytworzonego oprogramowania z zasadami inżynierii oprogramowania i jakość implementacji.

8. JEŚLI NIE WYSZCZEGÓLNIONO INACZEJ, WSZYSTKIE NIEJASNOŚCI NALEŻY PRZEDYSKUTOWAĆ Z PROWADZĄCYM ZAJĘCIA POD GROŹBĄ NIEZALICZENIA PROGRAMU W PRZYPADKU ICH NIEWŁAŚCIWEJ INTERPRETACJI.

```
// sprawdzic czy port i number to integer
    // sprawdzi czy port jest wiekszy od 0 a mniejszy do ...
    // sprawdzic czy numer jest wiekszy 0 a mniejszy od ...
    // co z klasami wewnetrznymi ...
    // to ma dzialac na localhoscie ...
    // czy to ma wyrzucac blad czy sie wylaczac jesli nie dziala
    // co z dokumentacja i zasadami inzynierii oprogramowania i jakosci implementacji

    // co jak serwer zaczyna z -1 lub 0
    // jakie liczby przesyla klient

    // rodzielenie klas i nie powtarzac sie
    // zamienic int na string

    // to ma miec .java i .class
    // komunikat rozgloszeniowy - podsiec to tyle aredsow ip ile w niej komputerow + 2 (wejscie do podsieci + adres rozgloszeniowy - mozna go znalezc na podstawie maski dodajac ten numer)

    // mozna zrobic try z deklaracja i inicjalizacja serwera a potem serwer start i jesli serwer juz dziala to throw exception
    // a potem w catch zrobic klienta ktory wysle wiadomosc
```