

Dokumentacja Projektu z przedmiotu Sieci Komputerowe i Programowanie Sieciowe(SKJ)

Igor Bukowski

s31489

Gr. 36c

1. Wstęp

Aplikacja realizuje system rozproszony oparty na protokole UDP, który umożliwia obliczenie średniej z liczb przesyłanych między procesami działającymi w trybie *master* oraz *slave*. Celem projektu było zaimplementowanie rozproszonego systemu uśredniającego, który działa na podstawie komunikacji między procesami w dwóch trybach: *master* i *slave*.

2. Opis funkcjonalności

Aplikacja jest uruchamiana za pomocą komendy:

```
java DAS <port> <number>
```

- **<port>**: Numer portu UDP, na którym odbywa się komunikacja.
- **<number>**: Liczba całkowita, którą proces w trybie *slave* wysyła do procesu *master*.

Po uruchomieniu aplikacji, wybierany jest jeden z dwóch trybów:

- **Tryb master**: Aplikacja nasłuchuje na zadanym porcie UDP i odbiera liczby. Na podstawie odebranych liczb, oblicza średnią (po uwzględnieniu liczby początkowej) i rozgłasza ją do innych komputerów w sieci lokalnej. Proces kończy się po otrzymaniu komunikatu -1, który także zostaje rozgłoszony.
- **Tryb slave**: Aplikacja wysyła liczbę do procesu działającego na porcie port na tej samej maszynie. Po wysłaniu liczby, proces oczekuje na potwierdzenie (ACK) i kończy działanie.

3. Szczegółowy opis protokołu z zaimplementowanego kodu

Ogólne założenia

Protokół implementowany w kodzie jest oparty na komunikacji za pomocą protokołu UDP (User Datagram Protocol) z wykorzystaniem **DatagramSocket**. Protokół działa w trybie **Master-Slave**, gdzie jeden proces pełni rolę główną (Master), a pozostałe procesy są w trybie podrzędnym (Slave). Celem protokołu jest przesyłanie liczb, ich zbieranie w procesie Master, obliczanie średniej oraz rozsyłanie wyników do procesów Slave.

Struktura protokołu

1. Parametry wejściowe:

- Port – port, na którym odbywa się komunikacja.
- Liczba – wartość numeryczna wysyłana przez proces Slave lub obsługiwana przez Master.

2. Tryby działania:

- **Master:**
 - Nasłuchuje na podanym porcie.
 - Zbiera liczby od procesów Slave.
 - Oblicza średnią przestanych liczb na żądanie.
 - Rozsyła obliczony wynik do wszystkich podłączonych procesów Slave.
 - **Slave:**
 - Wysyła liczbę do procesu Master.
 - Oczekuje na potwierdzenie (ACK) lub odpowiedź w postaci średniej.
 - Obsługuje czas oczekiwania na odpowiedź (timeout).
-

Proces działania protokołu

1. Uruchomienie

- **Master:**
 - Jeśli procesowi uda się otworzyć gniazdo na podanym porcie, wchodzi w tryb **Master**.
 - Rozpoczyna nasłuchiwanie na porcie i przechowuje odbierane liczby.
 - **Slave:**
 - Jeśli otwarcie portu się nie powiedzie (zajęty port), proces uruchamia się w trybie **Slave**.
 - Wysyła liczbę do procesu Master i oczekuje na odpowiedź.
-

2. Komunikacja

a) Komunikaty wysyłane przez Slave:

- **Liczba:** Proces Slave wysyła liczbę w formacie 4-bajtowego integera do procesu Master.

b) Komunikaty odbierane przez Master:

- **0:** Żądanie obliczenia średniej.
 - Master oblicza średnią wszystkich zebranych liczb.
 - Wynik jest przesyłany do wszystkich Slave poprzez broadcast.
 - **-1:** Żądanie zakończenia pracy.
 - Master rozsyła komunikat końca pracy do wszystkich Slave.
 - Master zamyka swoje gniazdo i kończy działanie.
 - **Liczba:** Master dodaje liczbę do swojej listy i wysyła potwierdzenie ACK do nadawcy.
-

3. Wysłanie odpowiedzi (ACK i broadcast)

- **ACK:** Po odebraniu liczby Master przesyła do danego Slave potwierdzenie w postaci 4-bajtowego integera o wartości 1.
 - **Broadcast:** Wynik obliczeń (średnia lub komunikat zakończenia pracy) jest rozsyłany do wszystkich Slave za pomocą adresu broadcast.
-

4. Obsługa wyjątków

- **Timeout:** Jeśli proces Slave nie otrzyma odpowiedzi od Master w określonym czasie (2 sekundy), kończy działanie.
 - **Błędy sieciowe:** Każdy błąd jest logowany w konsoli.
-

Struktura komunikatów

1. **Liczba przesyłana:** 4-bajtowa reprezentacja liczby całkowitej w formacie **Big Endian**.
 2. **Żądania od Slave do Master:**
 - 0: Oblicz średnią.
 - -1: Zakończ pracę.
 - liczba > 0: Nowa liczba do dodania.
 3. **Odpowiedzi od Master:**
 - 1: Potwierdzenie odbioru (ACK).
 - średnia: Wynik obliczeń.
 - -1: Komunikat zakończenia pracy.
-

Diagram przepływu

1. Slave wysyła liczbę do Master.
2. Master odbiera liczbę, wysyła ACK do Slave.
3. Slave oczekuje na odpowiedź, kończy działanie po timeout.
4. Master oblicza średnią po żądaniu od Slave (0) i rozsyła wynik.

5. Proces może być zakończony komunikatem -1.

Zalety protokołu

1. **Prostota:** Łatwa implementacja przy użyciu UDP.
2. **Broadcast:** Efektywne przesyłanie wyników do wszystkich podłączonych Slave.
3. **Timeout:** Zabezpieczenie przed nieskończonym oczekiwaniem na odpowiedź.

Ograniczenia

1. Brak zabezpieczeń przed utratą pakietów.
2. Protokół działa tylko w sieci lokalnej (ze względu na broadcast).
3. Master jest pojedynczym punktem awarii (single point of failure).

Możliwości rozwoju

1. Dodanie wsparcia dla retransmisji pakietów.
2. Obsługa wielu Masterów dla redundancji.
3. Szyfrowanie komunikacji dla zwiększenia bezpieczeństwa.

4. Zainstalowanie i Uruchomienie Aplikacji

1. Środowisko:

- Aplikacja została napisana w języku Java (JDK 1.8).
- Do kompilacji i uruchomienia programu wymagana jest instalacja JDK 1.8 lub nowszej wersji.

2. Instalacja:

- Należy skompilować aplikację, używając następującego polecenia:

```
javac DAS.java
```

3. Uruchomienie:

- Program uruchamiamy w trybie *master* lub *slave*, przekazując odpowiednie argumenty:

```
java DAS <port> <number>
```

- Jeśli port jest już zajęty (np. przez inny proces w trybie *master*), aplikacja automatycznie przechodzi w tryb *slave*.

4. Testowanie:

Aby przetestować program za pomocą skryptu w bashu należy uruchomić skrypt o nazwie **test_application.sh <port> <liczba>**, skrypt wysyła w trybie *slave* 10 liczb losowych z przedziału 1-100 do trybu *master* oraz otrzymuje potwierdzenie dojścia liczb.

Lub:

- Uruchomić dwa procesy na tym samym hoście (jeden w trybie *master*, drugi w trybie *slave*).
- Sprawdzić, czy komunikacja odbywa się poprawnie, a średnia jest poprawnie rozgłaszana w sieci.

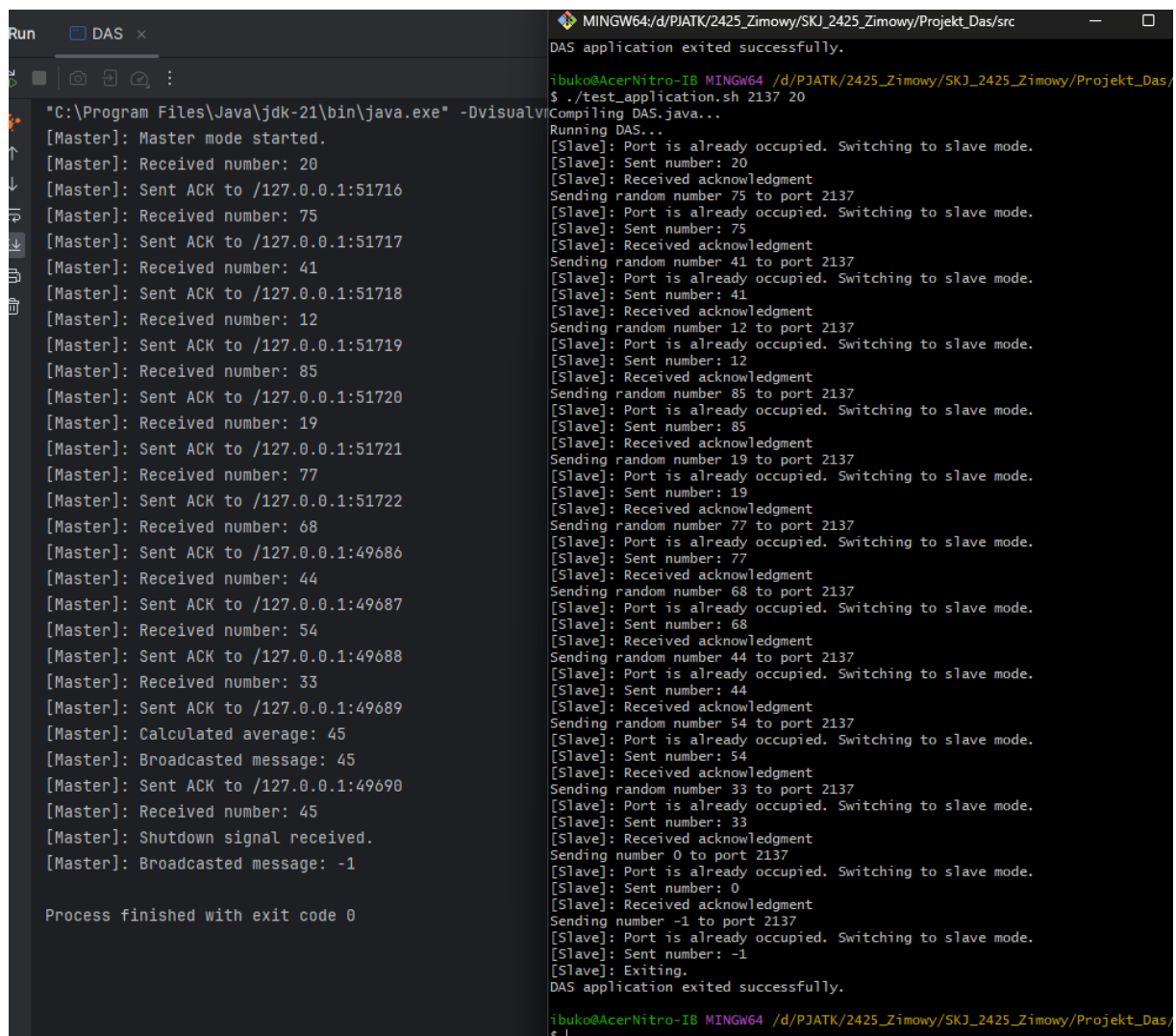
5. Problemy i Błędy

- **Problem z czasem oczekiwania w trybie slave:** Czas oczekiwania na odpowiedź z *master* wynosi 2 sekundy. Może to być zbyt krótki czas w przypadku dużego opóźnienia w sieci. Możliwe rozwiązanie: wydłużenie czasu oczekiwania.
- **Brak odpowiedzi w przypadku braku połączenia:** Jeśli aplikacja nie może nawiązać połączenia (np. z powodu błędu sieciowego lub braku innego procesu działającego w trybie *master*), aplikacja kończy działanie.

6. Podsumowanie

Aplikacja spełnia wymagania specyfikacji i działa w trybach *master* oraz *slave*. Procesy komunikują się za pomocą UDP, przysyłając liczby oraz rozgłaszając obliczoną średnią. Implementacja obejmuje również potwierdzenie dostarczenia wiadomości w trybie *slave*.

ZDJĘCIE POGLĄDOWE JAK DZIAŁA PROGRAM PONIŻEJ



The screenshot shows a Java application running in a terminal window. The application is a Master-Slave communication program. The Master process starts and sends a series of random numbers to the Slave process. The Slave process receives these numbers, sends back acknowledgments, and then sends the numbers back to the Master. The Master process calculates the average of the received numbers and broadcasts the result. The Slave process receives the broadcast and sends back a shutdown signal. The Master process then broadcasts a message and exits.

```
Run  DAS x
C:\Program Files\Java\jdk-21\bin\java.exe -DvisualVM
[Master]: Master mode started.
[Master]: Received number: 20
[Master]: Sent ACK to /127.0.0.1:51716
[Master]: Received number: 75
[Master]: Sent ACK to /127.0.0.1:51717
[Master]: Received number: 41
[Master]: Sent ACK to /127.0.0.1:51718
[Master]: Received number: 12
[Master]: Sent ACK to /127.0.0.1:51719
[Master]: Received number: 85
[Master]: Sent ACK to /127.0.0.1:51720
[Master]: Received number: 19
[Master]: Sent ACK to /127.0.0.1:51721
[Master]: Received number: 77
[Master]: Sent ACK to /127.0.0.1:51722
[Master]: Received number: 68
[Master]: Sent ACK to /127.0.0.1:49686
[Master]: Received number: 44
[Master]: Sent ACK to /127.0.0.1:49687
[Master]: Received number: 54
[Master]: Sent ACK to /127.0.0.1:49688
[Master]: Received number: 33
[Master]: Sent ACK to /127.0.0.1:49689
[Master]: Calculated average: 45
[Master]: Broadcasted message: 45
[Master]: Sent ACK to /127.0.0.1:49690
[Master]: Received number: 45
[Master]: Shutdown signal received.
[Master]: Broadcasted message: -1
Process finished with exit code 0
```

MINGW64:/d/PJATK/2425_Zimowy/SKJ_2425_Zimowy/Projekt_Das/src
DAS application exited successfully.
ibuko@AcerNitro-IB MINGW64 /d/PJATK/2425_Zimowy/SKJ_2425_Zimowy/Projekt_Das/
\$./test_application.sh 2137 20
Compiling DAS.java...
Running DAS...
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 20
[Slave]: Received acknowledgment
Sending random number 75 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 75
[Slave]: Received acknowledgment
Sending random number 41 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 41
[Slave]: Received acknowledgment
Sending random number 12 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 12
[Slave]: Received acknowledgment
Sending random number 85 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 85
[Slave]: Received acknowledgment
Sending random number 19 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 19
[Slave]: Received acknowledgment
Sending random number 77 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 77
[Slave]: Received acknowledgment
Sending random number 68 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 68
[Slave]: Received acknowledgment
Sending random number 44 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 44
[Slave]: Received acknowledgment
Sending random number 54 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 54
[Slave]: Received acknowledgment
Sending random number 33 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 33
[Slave]: Received acknowledgment
Sending number 0 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: 0
[Slave]: Received acknowledgment
Sending number -1 to port 2137
[Slave]: Port is already occupied. Switching to slave mode.
[Slave]: Sent number: -1
[Slave]: Exiting.
DAS application exited successfully.
ibuko@AcerNitro-IB MINGW64 /d/PJATK/2425_Zimowy/SKJ_2425_Zimowy/Projekt_Das/
\$