

Dokumentacja projektu – Zadanie UDP: Rozproszony System Uśredniający

Autor: Aleksandra Fus

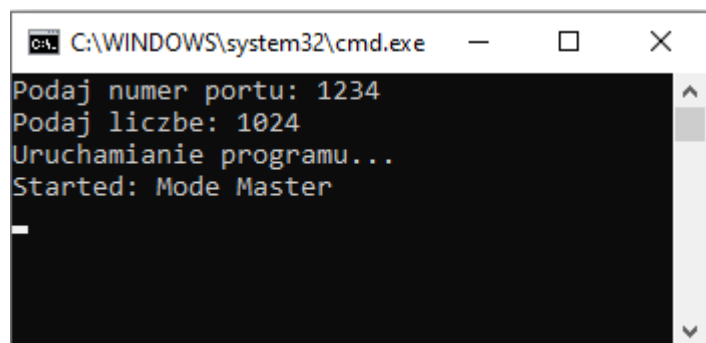
Numer indeksu: s30395

Uruchomienie programu

Za kompilację oraz uruchomienie programu, odpowiedzialny jest plik **runDAS.bat**.

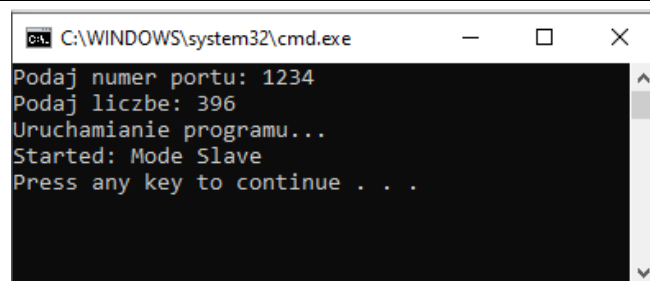
Po jego uruchomieniu, należy podać numer portu, na którym będzie przeprowadzana komunikacja UDP oraz liczbę całkowitą. Przykład działania pliku runDas.txt :

Pierwsze uruchomienie, program przechodzi w tryb Master.

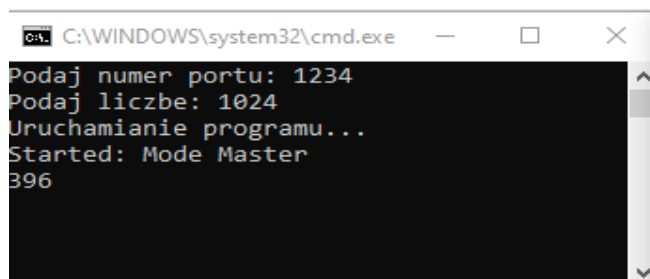


```
C:\WINDOWS\system32\cmd.exe
Podaj numer portu: 1234
Podaj liczbe: 1024
Uruchamianie programu...
Started: Mode Master
```

Nowo uruchomiony program przechodzi w tryb Slave, pobiera liczbę całkowitą oraz przekazuje ją głównemu programowi, za pomocą komunikacji UDP.



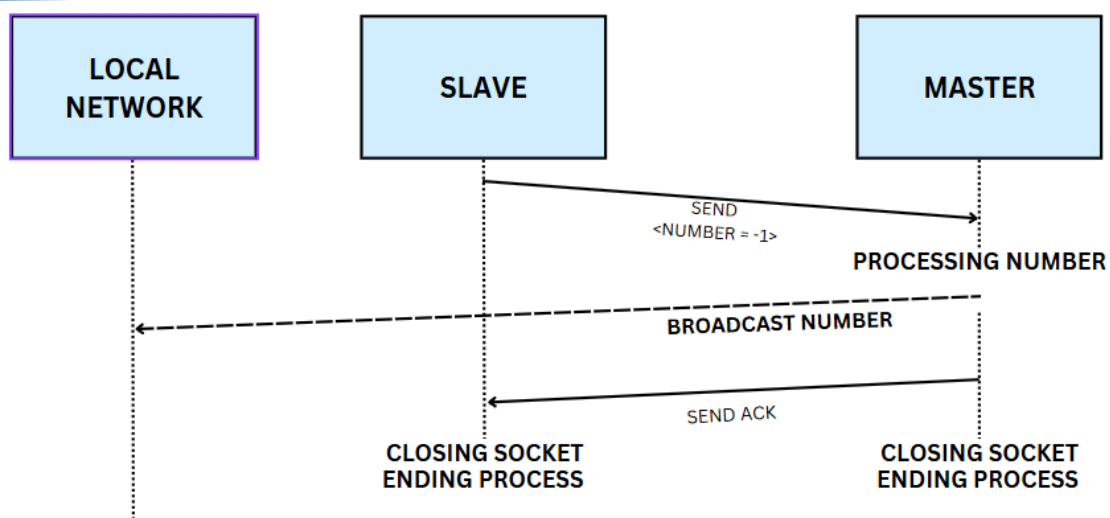
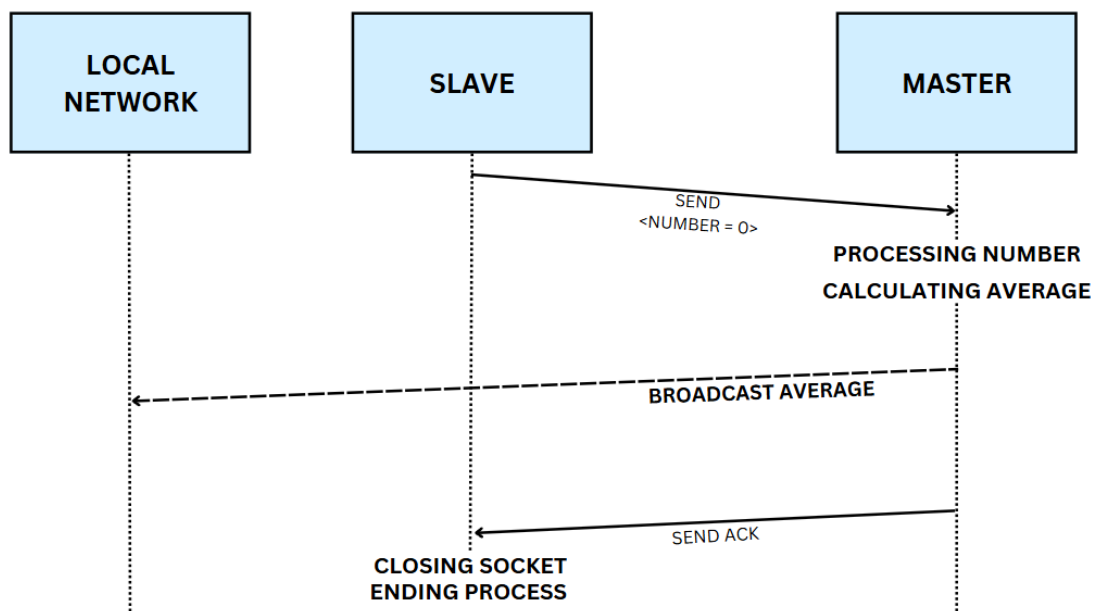
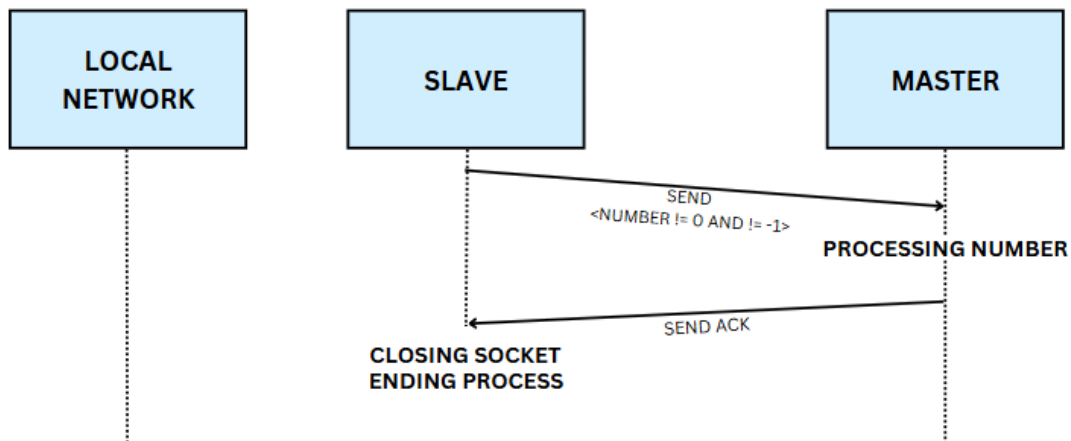
```
C:\WINDOWS\system32\cmd.exe
Podaj numer portu: 1234
Podaj liczbe: 396
Uruchamianie programu...
Started: Mode Slave
Press any key to continue . . .
```



```
C:\WINDOWS\system32\cmd.exe
Podaj numer portu: 1234
Podaj liczbe: 1024
Uruchamianie programu...
Started: Mode Master
396
```

Opis projektu

KONCEPCJA KOMUNIKACJI APLIKACJI



WERSJA JAVA:

Java 8

POLA KLASY:

Aplikacja składa się z klasy DAS, która implementuje wymaganą logikę projektu. Klasa ta, posiada pięć pól:

- *inputPort* (numer portu przekazanego programowi ; jest typu Integer),
- *inputNumber* (liczba całkowita przekazana programowi ; jest typu Integer),
- *currentMode* (tryb, w jakim obecnie pracuje program ; jest typu Mode, czyli stworzonego wewnątrz klasy DAS typu wyliczeniowego)
- *socket* (gniazdo, poprzez które będzie odbywała się komunikacja UDP ; jest typu DatagramSocket)
- *numberList* (lista, służąca do zapamiętywania liczb otrzymanych od użytkownika ; jest typu ArrayList<Integer>)

Parametrami konstruktora klasy DAS jest *[Integer inputPort]* i *[Integer inputNumber]*.

SZCZEGÓŁOWY OPIS METOD:

- **main(String[] args)** – inicjuje wykonywanie programu. Sprawdzane jest, czy podczas jego uruchomienia, zostały przekazane dwa argumenty oraz zależnie od tego kontynuuje wykonywane operacje, bądź kończy cały program. Tworzona jest instancja DAS z argumentami port i numer, które zostały przekazane przez użytkownika.

- **runMaster()** – jest uruchamiana w momencie, gdy w konstruktorze klasy DAS udaje się otworzyć gniazdo DatagramSocket socket oraz program przechodzi w tryb Master. Zapamiętywana jest liczba całkowita, podana przez użytkownika, poprzez dodanie jej do listy numberList. Aplikacja tworzy pakiet typu DatagramPacket, podając jako parametry tablicę bajtów wielkości 129 (zgodnie z zasadami optymalizacji i przyjętymi praktykami) oraz jej długość. Następnie, poprzez gniazdo przyjmowany jest pakiet, a do wcześniej wspomnianej tablicy bajtów wstawiane są dane. Program sprawdza, czy otrzymany pakiet zawiera tekst „ACK”, co oznaczałoby, że jest potwierdzeniem otrzymania wiadomości i nie należy go przetwarzać. Zależnie od tego, jaka wartość została przyjęta, program wykonuje odpowiednie działania:

- dla wartości 0 : liczy średnią arytmetyczną ze wszystkich niezerowych liczb, otrzymanych od początku działania, uwzględniając wartość inputNumber, wypisuję nowo przyjętą wartość, a następnie wysyła komunikat rozgłoszeniowy za pomocą metody sendMessage(...) z ustawioną flagą isBroadcast na true oraz kontynuuje działanie programu.
- dla wartości -1 : wypisuje nowo przyjętą wartość na konsolę, wysyła

komunikat rozgłoszeniowy za pomocą metody `sendMessage(...)` z ustawioną flagą `isBroadcast` na `true`, zawierający otrzymaną wartość, wysyła wiadomość potwierdzającą otrzymanie pakietu, zamyka gniazdo socket oraz kończy pracę programu.

- domyślnie : wypisuje przyjętą wartość oraz dodaje ją do listy dotychczas przyjętych liczb całkowitych.

Ostatecznie, wysyłana jest wiadomość potwierdzająca otrzymanie pakietu, aby zapobiec możliwej utracie datagramów.

- `sendMessage(String message, InetAddress address, int port, boolean isBroadcast)`

– służy do wysłania komunikatu rozgłoszeniowego do wszystkich komputerów w sieci lokalnej na port ustanowiony podczas uruchamiania programu. Tworzona jest tablica bajtów, zawierająca zakodowany argument `mes` w sekwencji bajtów. Następnie, wywoływana jest metoda `setBroadcast(isBroadcast)` na instancji `DatagramSocket` socket, która została wcześniej zadeklarowana. Metoda ta, aktywuje, bądź dezaktywuje flagę `SO_BROADCAST`, która umożliwi rozsyłanie datagramów rozgłoszeniowych. Tworzony jest pakiet `DatagramPacket` `sendPacket`, przyjmujący jako argumenty wcześniej wspomnianą tablicę bajtów, zawierającą zakodowaną wiadomość, długość tablicy, adres IP `255.255.255.255` opakowany klasą `InetAddress.getByName(...)`, który jest adresem rozgłoszeniowym, numer portu `inputPort`. Ostatecznie, stworzony pakiet jest wysyłany poprzez gniazdo socket do wszystkich urządzeń w sieci lokalnej.

- **`runSlave()`** – jest uruchamiana w momencie, gdy w konstruktorze klasy `DAS` nie udaje się otworzyć gniazda `DatagramSocket` socket oraz program przechodzi w tryb `Slave`. Aplikacja otwiera wcześniej wspomniany socket na losowym porcie, nie podając żadnych argumentów w konstruktorze oraz ustawia `Timeout` za pomocą metody `setSoTimeout(...)` na dwie sekundy. Dzięki temu, jeśli program w trybie `Slave` nie otrzyma potwierdzenia `ACK` od `Mastera` w odpowiednim czasie, spróbuje wysłać ostatni pakiet ponownie. Tworzona zostaje tablica bajtów `data`, przechowująca podaną przez użytkownika w chwili uruchamiania programu liczbę całkowitą, zapisaną jako zakodowana sekwencja bajtów. Następnie, tworzony jest pakiet typu `DatagramPacket` z argumentami: tablicą bajtów `data`, jej długością, adresem IP opakowanym klasą `InetAddress.getByName(„localhost”)`, numerem portu `inputPort`. W pętli `while`, program próbuje wysłać pakiet z danymi trzy razy, jeśli jego odbiorca nie potwierdził otrzymania go. Aby to wykonać, tworzony jest pomocniczy `DatagramPacket` `ackPacket`, który jako argument metody `socket.receive(ackPacket)` przyjmuje przychodzące datagramy. Jeśli czas oczekiwania przekroczy dwie sekundy, powstaje wyjątek `SocketTimeoutException`, którego przechwycenie polega na wypisaniu komunikatu na konsolę oraz zwiększenie licznika prób. W przeciwnym wypadku, sprawdzane jest, czy otrzymany pakiet jest potwierdzeniem od odbiorcy i zapamiętanie tej informacji. Końcowo, gniazdo socket jest zamykane, a jeśli nie otrzymano potwierdzenia, wyświetlana jest informacja o tym na konsoli.

MOŻLIWE PROBLEMY:

Możliwe jest zapętlenie programu, gdy średnia obliczona przez Mastera, po otrzymaniu liczby 0, będzie równa 0. Wtedy, wysyłany jest komunikat rozgłoszeniowy do wszystkich urządzeń w sieci lokalnej z wartością „0”. Master odbiera ten komunikat i traktuje go jako polecenie do obliczenia średniej arytmetycznej ze wszystkich niezerowych liczb, które otrzymał. Powstaje tutaj nieskończona pętla, ponieważ nieustannie będzie się wykonywał ten sam kod. Ewentualnym rozwiązaniem tego problemu, jest dodanie do przesyłanych pakietów, dodatkowych flag, które pomogłyby rozróżniać typy komunikatów.