

Opis

projektu nr 2

# technologia RMI

z przedmiotu:

Programowanie równoległe i rozproszone

Temat:

Monte Carlo - Ruch na rondzie. (Quinn, 10.13)

Autorzy:

Tomasz Giereś

Robert Gaca

## 1) Algorytm.

Algorytm, który jest podstawą działania programu opiera się na kilku założeniach:

- rondo (okrąg) podzielony jest na 16 równych części, zwanych sektorami,
- ruch na rondzie jest podzielony jest na cykle, zwane też iteracjami, podczas których samochody wjeżdżają na rondo, zjeżdżają z niego lub przejeżdżają do kolejnych sektorów,
- w każdym z sektorów może znajdować się najwyżej jeden samochód,
- sektory numerujemy od 0 do 15 zgodnie z ruchem wskazówek zegara, w tym kierunku odbywa się też ruch na rondzie,
- 4 drogi dojazdowe mieszczą się przy czterech równo rozłożonych na rondzie sektorach, o numerach 0, 4, 8 i 12, zwanych też *offsetami*,
- na każdym z wjazdów, na drodze dojazdowej, czekają samochody chcące wjechać na rondo, tworzą one kolejkę,
- pierwszy samochód z kolejki może w następnym cyklu wjechać na rondo na sektor  $n$  przy którym znajduje się wjazd, jeśli sektor  $n-1$  jest pusty lub samochód, który się na nim znajduje zjedzie z ronda w drogę dojazdową, znajdującą się przy sektorze  $n$ .

Parametrami tej symulacji są:

- częstotliwość pojawiania się nowych pojazdów na każdej z dróg dojazdowych,
- tablica prawdopodobieństw, która opisuje dla każdego wjazdu informacje o prawdopodobieństwie zjazdu w każdą z dróg dojazdowych lub zawróceniu,
- pozycje dróg dojazdowych, tj. numer sektora, przy którym się znajdują

## 2) Budowa, opis

Kod źródłowy znajduje się w plikach o rozszerzeniach *.java*.

Plikami konfiguracyjnymi są:

- *machines.txt* – zawierający adresy ip rejestrów i zarejestrowane nazwy obiektów reprezentujących symulację; każdy wpis wprowadzamy w osobnej linii w formacie:  
*adres\_ip\_rejestru nazwa\_obiektu*
- *params.txt* – zawierający tablicę prawdopodobieństw wykorzystywanych w symulacji i rozsyłanych przez klienta do serwerów.

–

### 3) Uruchomienie programu, obsługa i wygenerowane wyniki dla przykładowych danych

Serwer przy uruchamianiu „ręcznym” przyjmuje parametr, który będzie nazwą stworzonego obiektu symulacji, np. polecenie `java Server tcs1` uruchomi serwer z parametrem `tcs1`.

Aby program mógł uzyskać wszystkie potrzebne do pracy uprawnienia należy uruchomić go poleceniem:

```
java -Djava.security.manager -Djava.security.policy=policy-file Server tcs1
```

Dla łatwiejszego uruchamiania symulacji i kompilacji programu stworzony został plik Makefile zawierający wpisy:

- do kompilacji, uruchamiany polecenie: `make`,
- do uruchomienia klienta, polecenie: `make client`,
- do uruchomienia serwerów, domyślnie stworzone są 4 wpisy uruchamiane poleceniem: `make server_tcsx`, gdzie x to liczba od 1 do 4;
- do przywrócenia katalogu do stanu początkowego, polecenie: `make reset`

Uruchamianie programu z domyślnymi parametrami, 4 serwery na localhost:

1. `make` – kompilacja
2. na każdym komputerze obsługującym jeden lub więcej serwerów należy uruchomić rejestr obiektów poleceniem: `rmiregister`
3. uruchomienie 4 serwerów poleceniami:
  - `make server_tcs1`
  - `make server_tcs2`
  - `make server_tcs3`
  - `make server_tcs4`Aktywowaliśmy serwery rejestrujące symulacje `tcs1`, `tcs2`, `tcs3` i `tcs4`.
4. Uruchomienie klienta:  
`make client`

## 5. Diagram klas

