# Projekt - Symulacja Cyfrowa

Algorytm CSMA p-persistent metodą interakcji procesów

Iwona Pawełek

24/05/2017

## 1. Pełny tekst rozwiązywanego zadania

W sieci bezprzewodowej stacje nadawcze konkurują o dostęp do łącza. W losowych odstępach czasu CGPk k-ta stacja nadawcza generuje pakiety gotowe do wysłania. Po uzyskaniu dostępu do łącza zgodnie z algorytmem A, k-ty terminal podejmuje próbę transmisji najstarszego pakietu ze swojego bufora. Czas transmisji wiadomości z k-tej stacji nadawczej do k-tej stacji odbiorczej wynosi CTPk. Jeśli transmisja pakietu zakończyła się sukcesem, stacja odbiorcza przesyła potwierdzenie ACK (ang. Acknowledgment) poprawnego odebrania wiadomości. Czas transmisji ACK wynosi CTIZ. Jeśli transmisja pakietu nie powiodła się, stacja odbiorcza nie przesyła ACK. Odbiór pakietu uznajemy za niepoprawny, jeśli w kanale transmisyjnym wystąpiła kolizja. Przez kolizję rozumiemy nałożenie się jakiejkolwiek części jednego pakietu na inny pakiet (pochodzący z innego nadajnika). Brak wiadomości ACK po czasie (CTPk+ CTIZ) od wysłania pakietu jest dla stacji nadawczej sygnałem o konieczności retransmisji pakietu. Każdy pakiet może być retransmitowany maksymalnie LR razy. Dostęp do łącza w przypadku retransmisji opiera się na tych samych zasadach co transmisja pierwotna. Jeśli mimo LR-krotnej próby retransmisji pakietu nie udało się poprawnie odebrać, wówczas stacja nadawcza odrzuca pakiet i – jeśli jej bufor nie jest pusty – przystępuje do próby transmisji kolejnego pakietu.
Opracowano symulator sieci bezprzewodowej zgodnie z metodą interakcji procesów (M4).
Za pomocą symulacji wyznacz wartość parametru L, która zapewni średnią pakietową stopę błędów (uśrednioną po K odbiornikach) nie większą niż 0.1, a następnie:

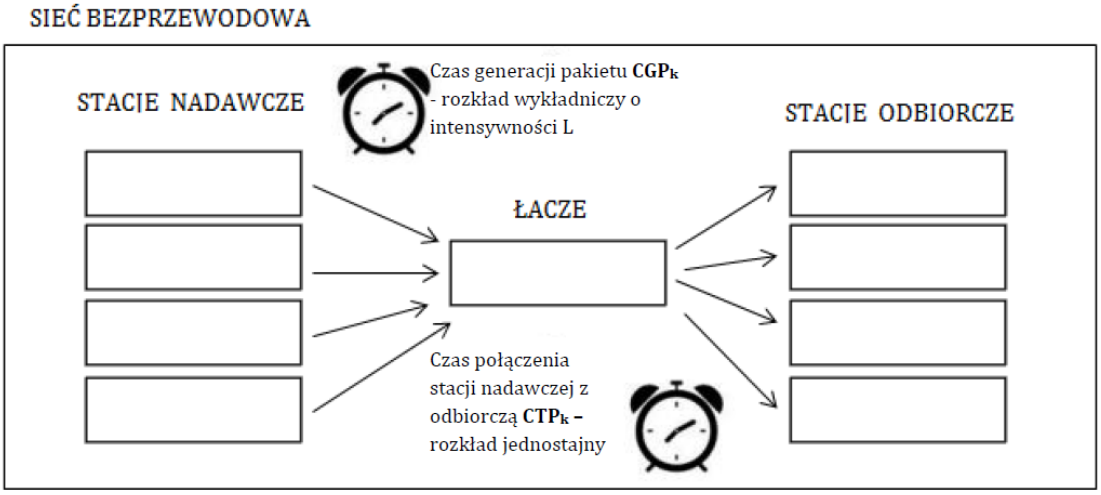
- średnią liczbę retransmisji pakietów,
- przepływność systemu mierzoną liczbą poprawnie odebranych pakietów w jednostce czasu,
- średnie opóźnienie pakietu, tzn. czas jaki upływa między pojawieniem się pakietu w buforze, a jego poprawnym odebraniem,
- średni czas oczekiwania, tzn. czas między pojawieniem się pakietu w buforze, a jego opuszczeniem

Sporządź wykres zależności przepływności systemu oraz średniej i maksymalnej pakietowej stopy błędów w zależności od wartości L.

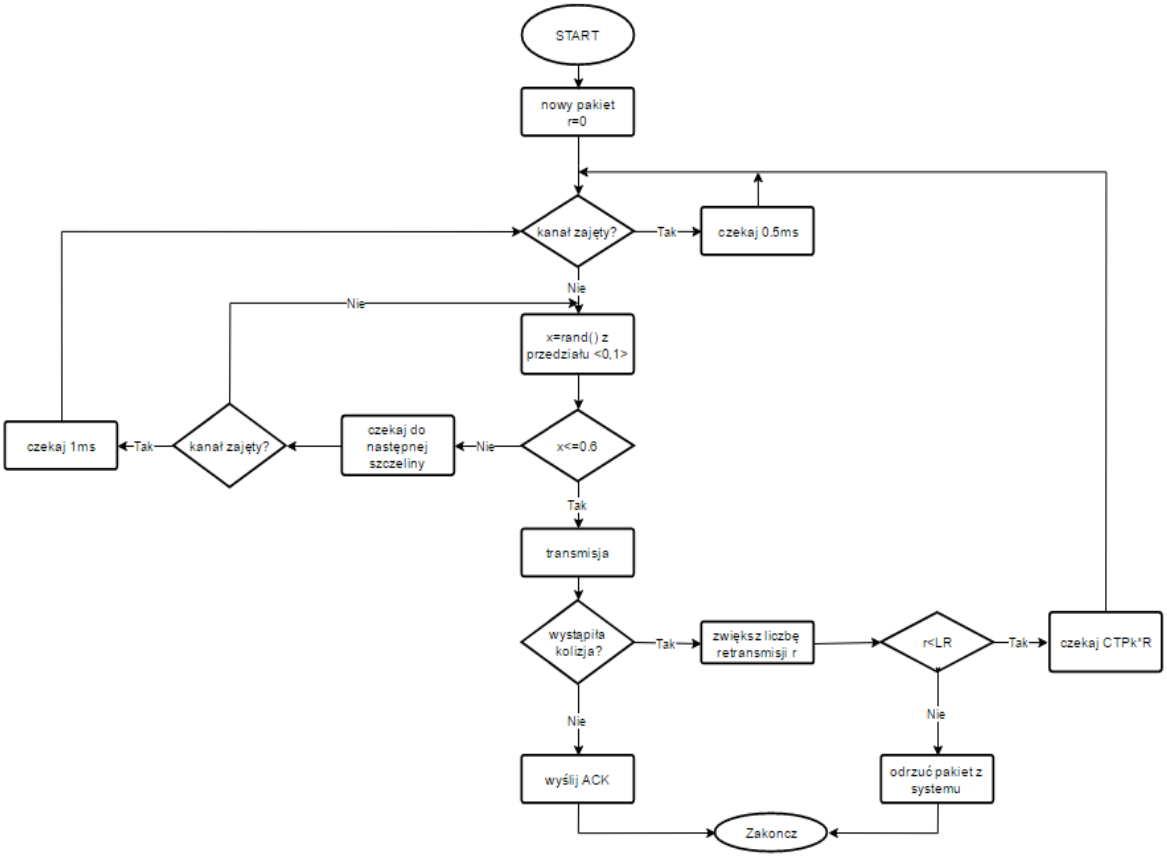
Protokół CSMA (ang. Carrier Sense Multiple Access) z wymuszaniem transmisji prawdopodobieństwem p (ang. p-persistent) – w protokole tym czas jest podzielony na szczeliny o długości CSC. Po wygenerowaniu nowego pakietu, stacja nadawcza sprawdza zajętość kanału transmisyjnego. Jeśli kanał jest zajęty, to dalsze odpytywanie kanału odbywa się w odstępach co 0.5 ms. Gdy stacja wykryje, że kanał jest wolny, rozpoczyna transmisję w najbliższej szczelinie z prawdopodobieństwem PT. Z prawdopodobieństwem (1-PT) stacja wstrzymuje się z transmisją do następnej szczeliny, w której ponownie sprawdza status kanału. Jeśli następna szczelina okaże się również wolna, terminal rozpoczyna transmisję z prawdopodobieństwem PT lub wstrzymuje się z prawdopodobieństwem (1-PT). Ta procedura jest powtarzana tak długo, aż pakiet zostanie wysłany lub kanał stanie się zajęty. W tym ostatnim przypadku terminal nasłuchuje kanał w odstępach co 1 ms i gdy wykryje, że jest wolny, rozpoczyna opisaną wyżej procedurę od nowa. W przypadku retransmisji, stacja nadawcza sprawdza stan kanału po losowym czasie CRP równym R*CTPk, gdzie R jest losową liczbą z przedziału od <0, ($2^r$-1)>, a r jest numerem aktualnej retransmisji (przy każdej retransmisji czas ten jest losowany ponownie). Wówczas uruchamiana jest taka sama procedura jak w przypadku transmisji pierwotnej.  c) PT = 0.6

## 2. Krótki opis modelu symulacyjnego

### 2.1 Schemat modelu symulacyjnego




Rys.1. Schemat modelu symulacyjnego




Rys.2. Schemat blokowy algorytmu z uwzględnieniem retransmisji

## 2.2 Opis klas wchodzących w skład systemu i ich atrybutów

| **Obiekt** | **Nazwa klasy** | **Opis** | **Atrybuty** |
|---|---|---|---|
| Sieć bezprzewodowa | Siec | Klasa zawierająca wszystkie obiekty istniejące w systemie oraz dodatkowe metody służące do wspomagania klasy Statystyka | - lista stacji nadawczych *List<StacjaNadawcza()*<br>- lista stacji odbiorczych *List<Stacja>()*<br>- obiekt *Lacze* reprezentujący medium transmisyjne<br>- generatory odpowiedzialne za losowanie czasu CTPk, zmiennej R oraz prawdopodobieństwa transmisji *UniwersalnyGeneratorLosowy(kernel*) |
| Stacja nadawcza | StacjaNadawcza | Klasa generująca i przechowywująca pakiety do czasu ich transmisji. | - pole umożliwiające identyfikację stacji nadawczej *int*<br>- kolejkę pakietów do nadania przechowuje pole *Queue<Pakiet>()*<br>- generator wykładniczy *WykladniczyGeneratorLosowy(lambda, kernel)* do losowania CGPk |
| Pakiet | Pakiet | Klasa reprezentująca pakiet pojawiający się w losowych odstępach czasu. Dziedziczy po klasie Proces | - pole umożliwiające identyfikację pakietu typu *int*<br>- pole zliczające liczbę retransmisji pakietu typu *int*<br>- pole zawierające czas transmisji pakietu typu *double*<br>- pole identyfikujące kolizję oraz poprawny odbior typu *bool*<br>- pola określające czas pojawienia się w buforze, czas nadania i czas odbioru pakietu typu *double* |
| Łącze | Lacze | Klasa reprezentująca medium transmisyjne systemu. Zawiera pakiety które są właśnie transmitowane oraz metody wykrywające kolizję pakietów. | - lista pakietów oczekujących typ *List<Pakiet>*<br>- zmienna informująca o stanie kanału typu *bool* |
| Stacja odbiorcza | Stacja | Klasa reprezentuje stację odbiorczą przechowującą odebrane pakiety. Dodatkowo posiada metody służące do wspomagania klasy Statystyka | - pole umożliwiające identyfikację stacji nadawczej typu *int*<br>- kolejkę zawierającą odebrane pakiety *Queue<Pakiet>()*<br>- pole przechowywujące liczbę odebranych i straconych pakietów typu *int* |

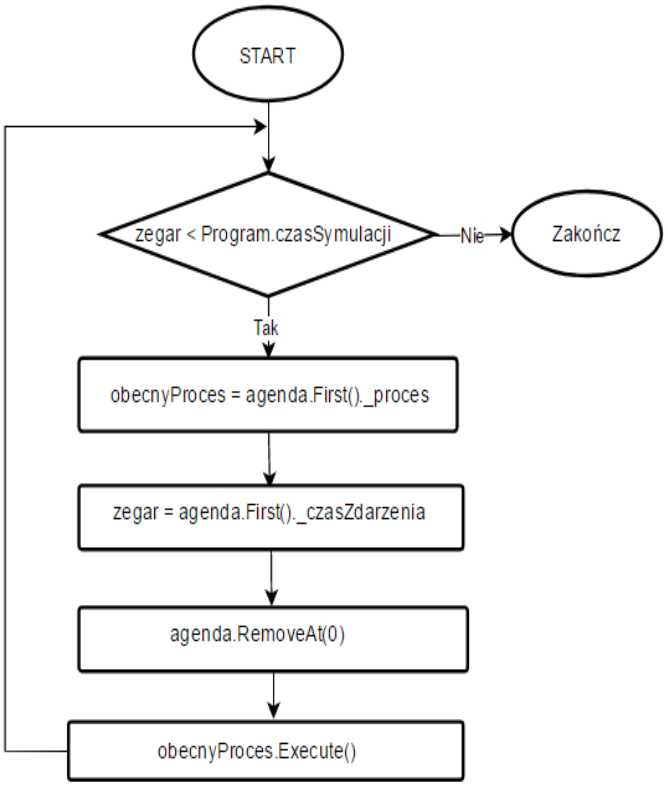
| **Obiekt** | **Nazwa klasy** | **Opis** | **Atrybuty** |
|---|---|---|---|
| - | Proces | Klasa zawiera implementację wirtualnej metody *Execute()* oraz metody Activate . Jest klasą bazową dla klasy pakiet. | - pole faza typu *int*<br>- pole wykonano typu *bool*<br>- pole przechowujące zdarzenie typu *Zdarzenie* |
| agenda | Zdarzenie | Klasa reprezentująca zdarzenie w systemie. | - pole przechowujące czas wystąpienia zdarzenia typu *double*<br>- pole przechowujące proces typu *Proces*<br>- pole przechowujące priorytet zdarzenia, które właśnie wystąpiło typu *int* |
| - | Ziarno | Statyczna klasa odpowiedzialna za wygenerowanie ziaren i zapisanie ich do pliku. | Metody generujące ziarno i zapisujące do pliku |
| - | Statystyka | Statyczna klasa wspomagająca wykonywanie obliczeń w systemie. | Pola przechowujące wymagane statystyki oraz metody służące do ich obliczenia |
| Generatory w nadajnikach oraz generatory w sieci | UniwersalnyGeneratorLosowy WykladniczyGeneratorLosowy | Klasy odpowiadające właściwym generatorom wykorzystywanym w systemie | Tworzą nowe obiekty generatorów typu *GeneratorRownomierny i GeneratorWykladniczy i* wywołują ich metody |
| Obiekty tworzone są w klasie UniwersalnyGeneratorLosowy | GeneratorRownomierny | Klasa zawiera definicje metod odpowiedzialnych za losowe wyznaczanie wartości dla generatora równomiernego. | - pola przechowujące wartości kernel, M, A, Q, R typu *int*<br>- metodę zwracająca kernel |
| Obiekty tworzone są w klasie WykladniczyGeneratorLosowy | GeneratorWykladniczy | Klasa zawiera definicje metod odpowiedzialnych za losowe wyznaczanie wartości dla generatora wykładniczego. | - pole przechowujące wartość lambda typu *double*<br>- pole przechowujące generator równomierny typu *GeneratorRownomierny* |
| - | Program | Klasa przechowująca pola statyczne dla symulacji, odpowiedzialna za interakcję z użytkownikiem | - pola przechowujące liczbę nadajników, liczbę retransmisji LR, czas symulacji, wartość lambda, zegar symulacji, wartość PT oraz główną pętlę symulacji |

## 3. Opis przydzielonej metody symulacyjnej

W symulowanym systemie wyróżniono jeden proces, jakim jest Pakiet. Proces realizuje poszczególne fazy:

1. Pojawienie się pakietu w systemie
   a) Planowanie pojawienia się nowego pakietu
   b) Dodanie aktywnego pakietu do stacji nadawczej
   c) Przejście do kolejnej fazy
2. Sprawdzenie pierwszeństwa
   a) Sprawdzenie czy pakiet jest pierwszy w buforze
   b) Jeśli pierwszy w buforze przejście do kolejnej fazy
   c) Jeśli nie , czeka na swoje wywołanie
3. Nasłuchiwanie I
   a) Sprawdzenie stanu kanału
   b) Jeśli kanał wolny to przejście do następnej fazy
   c) Jeśli kanał zajęty to sprawdza ponownie za 0.5ms
4. Losowanie
   a) Losuje liczbę
   b) Jeśli wylosowana liczba jest mniejsza niż 0.6 to przechodzi do fazy 6
   c) Jeśli wylosowana liczba jest większa od 0.6 to czeka do następnej szczeliny i przechodzi do fazy 5
5. Nasłuchiwanie II
   a) Sprawdzenie stanu kanału
   b) Jeśli kanał wolny to przejście do fazy 4
   c) Jeśli kanał zajęty to przejście do fazy 3 po 1ms
6. Analiza kanału
   a) Sprawdzenie, czy pakiet trafił na początek szczeliny
   b) Sprawdzenie, czy w kanale są oczekujące pakiety
   c) Jeśli tak ustawienie flag kolizji pakietów
   d) Dodanie pakietu do kanału i przejście do fazy 7
7. Transmisja
   a) Ustawienie kanału jako zajęty
   b) Zaplanuj czas transmisji pakietu
   c) Przejście do fazy 9
8. Retransmisja
   a) Zwiększenie liczby retransmisji
   b) Jeśli liczba retransmisji mniejsza od LR to zaplanuj ponowną transmisję po czasie R*CTPk
   c) Jeśli liczba retransmisji większa od LR usuń pakiet z systemu i zaplanuj aktywację pierwszego pakietu w swoim nadajniku

9. Sprawdzenie kolizji
    a) Jeśli wystąpiła kolizja usuń pakiet z kanału, zwolnij kanał i przejdź do fazy 8
    b) Jeśli nie, przejdź do fazy 10 i zaplanuj ustawienie ACK
10. Odbiór pakietu
    a) Dodanie pakietu do stacji odbiorczej
    b) Usunięcie pakietu z kanału
    c) Usunięcie pakietu z stacji nadawczej
    d) Zwolnienie kanału
    e) Jeśli jest, aktywacja pierwszego pakietu w buforze nadajnika, który pakiet opuścił




Rys.3. Schemat blokowy pętli głównej

## 4. Parametry wywołania programu

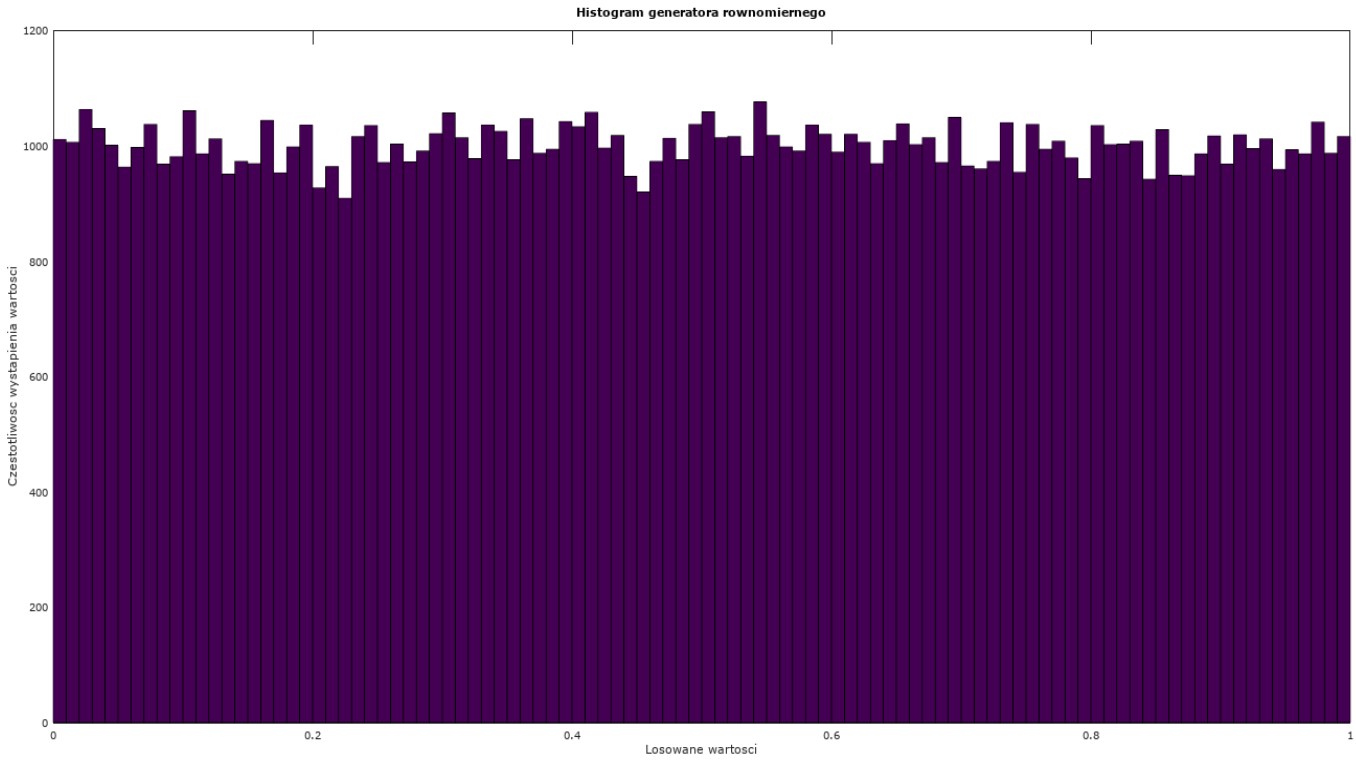
Niektóre parametry uległy zmianie w stosunku do początkowych założeń projektowych. Zmiany zostały wprowadzone w celu uzyskania średniej pakietowej stopy błędu na poziomie 0.1. Wartość parametru lambda 0.0067. Czas symulacji równy 200 sekund. Liczba symulacji równa 10. Liczba nadajników wynosi 10. Liczba retransmisji LR równa 5. Koniec fazy początkowej został wyznaczony na 12 pakietów. Prawdopodobieństwo PT na poziomie 0.6. CSC oraz CTIZ równe 1ms. Zmianie uległy parametry: liczba nadajników (wstępnie miała ona wynosić 4) oraz wartość LR (wstępnie miała ona wynosić 15).

# 5. Generatory

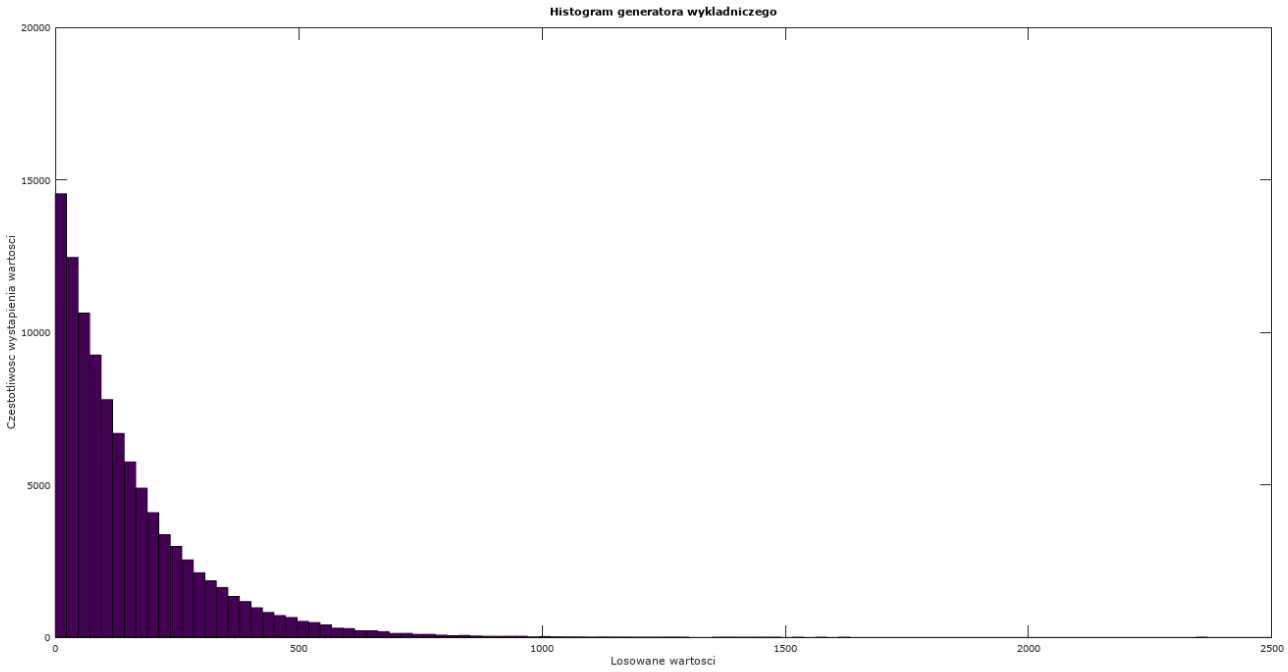
## 5.1 Opis zastosowanych generatorów liczb losowych z histogramami

Generatory pseudolosowe zastosowano w celu zapewnienia jak najlepszej losowości zdarzeń w trakcie wykonywania symulacji. Generatory zostały wykorzystane do wygenerowania:
- czasu utworzenia nowego pakietu CGPk (generator wykładniczy)
- losowego czasu trwania pakietu CTPk (generator równomierny)
- prawdopodobieństwa PT (generator równomierny)
- wartości R wykorzystywanej do sprawdzenia kanału po losowym czasie w przypadku retransmisji (generator równomierny).

Do początkowego zainicjowania generatora równomiernego wykorzystano ziarno o wartości 123. Liczba losowań wynosi 100000. Wylosowane wartości przedstawiono na histogramie za pomocą 100 przedziałów.




Do początkowego zainicjowania generatora wykładniczego wykorzystano ziarno o wartości 123 oraz lambdę na poziomie 0.0067. Liczba losowań wynosi 100000. Wylosowane wartości przedstawiono na histogramie za pomocą 100 przedziałów.




### 5.2 Wyjaśnienie, w jaki sposób została zapewniona niezależność sekwencji losowych w różnych symulacjach

Aby zapewnić niezależność sekwencji losowych należy mieć pewność, że każdy generator inicjowany będzie ziarnem, które nie będzie się powtarzać. Ziarna zostały wygenerowane generatorem równomiernym inicjowanym wartością 1. Odstęp między poszczególnymi ziarnami wynosi 100000. Wszystkie wartości ziaren zostały zapisane w pliku ziarna.txt, który jest zlokalizowany w folderze …\Symulacja\Zadanie1\bin\Debug.
W programie przewidziano 10 symulacji. System posiada 10 nadajników a każdy z nich posiada własny generator wykładniczy inicjowany innym ziarnem. Ponadto utworzona sieć posiada 3 generatory równomierne również inicjowane innymi ziarnami. Ostatecznie liczba generowanych ziaren wynosi 130 ziaren. Ziarna zostały przydzielone w momencie utworzenia generatora.

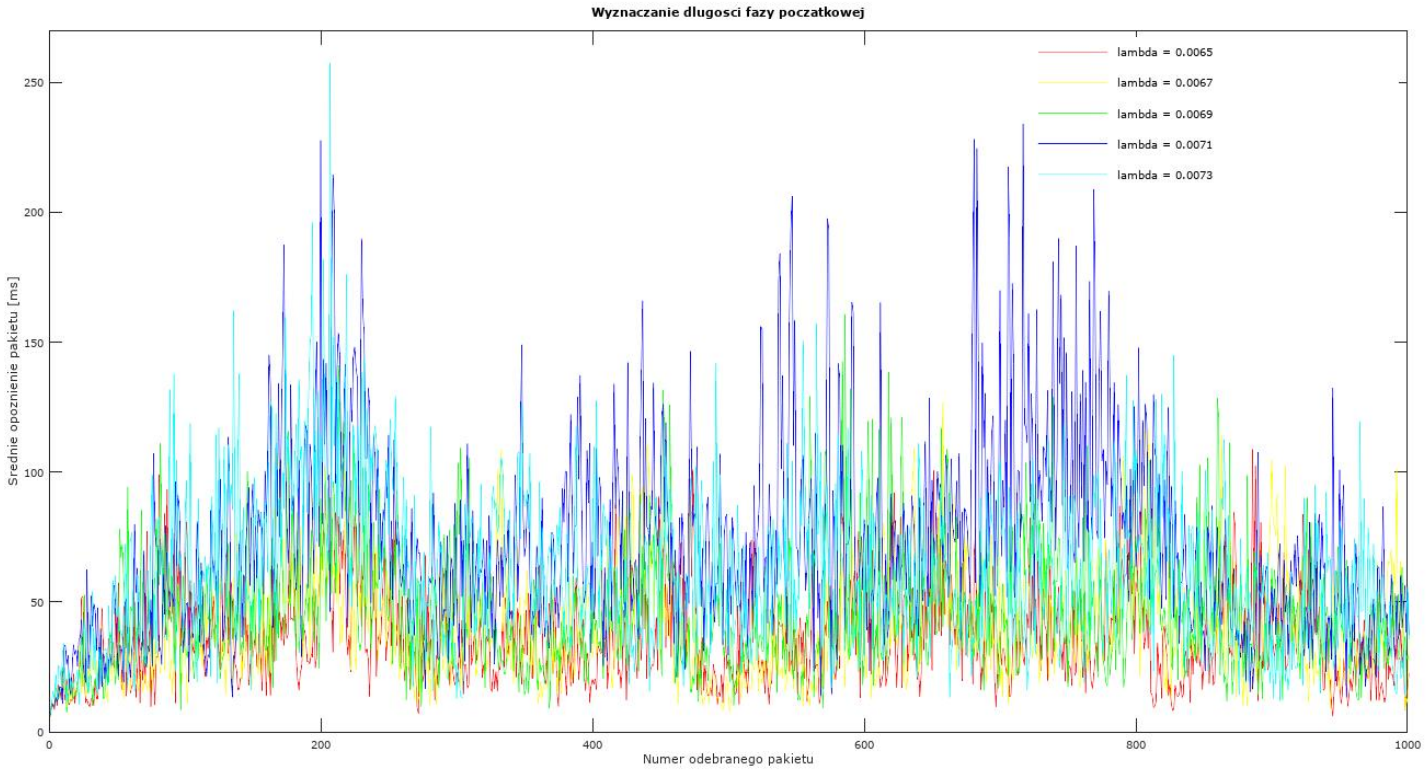
## 6. Metody testowania i weryfikacji poprawności działania programu

W celu weryfikacji poprawności działania programu w początkowych fazach kod na bieżąco zostawał poddawany analizie wykonywanej przy pomocy dostępnych narzędzi środowiska Visual Studio. Wykorzystano Debuger oraz krokowe wykonywanie programu. Również na bieżąco wyświetlane były informacje w konsoli programu w jakiej fazie symulacji znajduje się pakiet. Następnie przeprowadzono analizę programu dla rożnych parametrów wejściowych jednocześnie kontrolując, czy wprowadzane zmiany powodują zmianę wyników w sposób racjonalny. Ostatnim etapem była analizy statystyk oraz porównanie ich z wynikami symulacji kolegów.

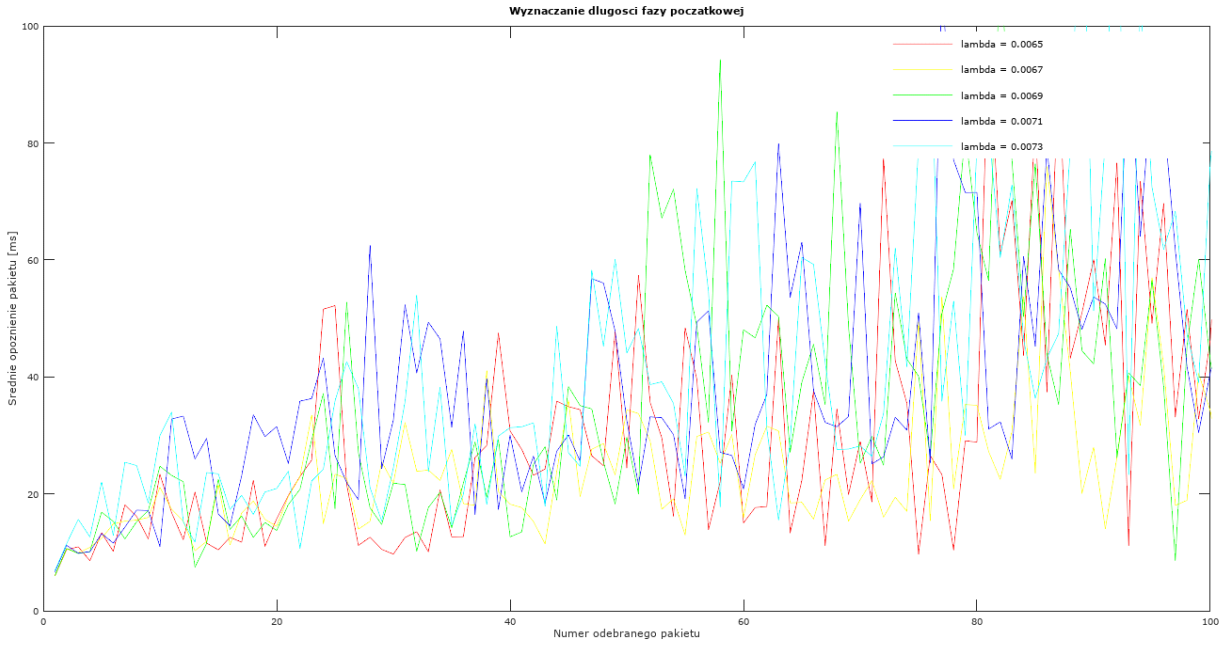
# 7. Wyniki symulacji

## 7.1 Wyznaczenie długości fazy początkowej

Długość fazy początkowej została określona na podstawie wykresu średniego opóźnienia pakietu podanego w ms w zależności od numeru odebranego pakietu dla którego policzono średnią. Za koniec fazy początkowej uznaje się moment w którym wykres średniego opóźnienia przestaje narastać.

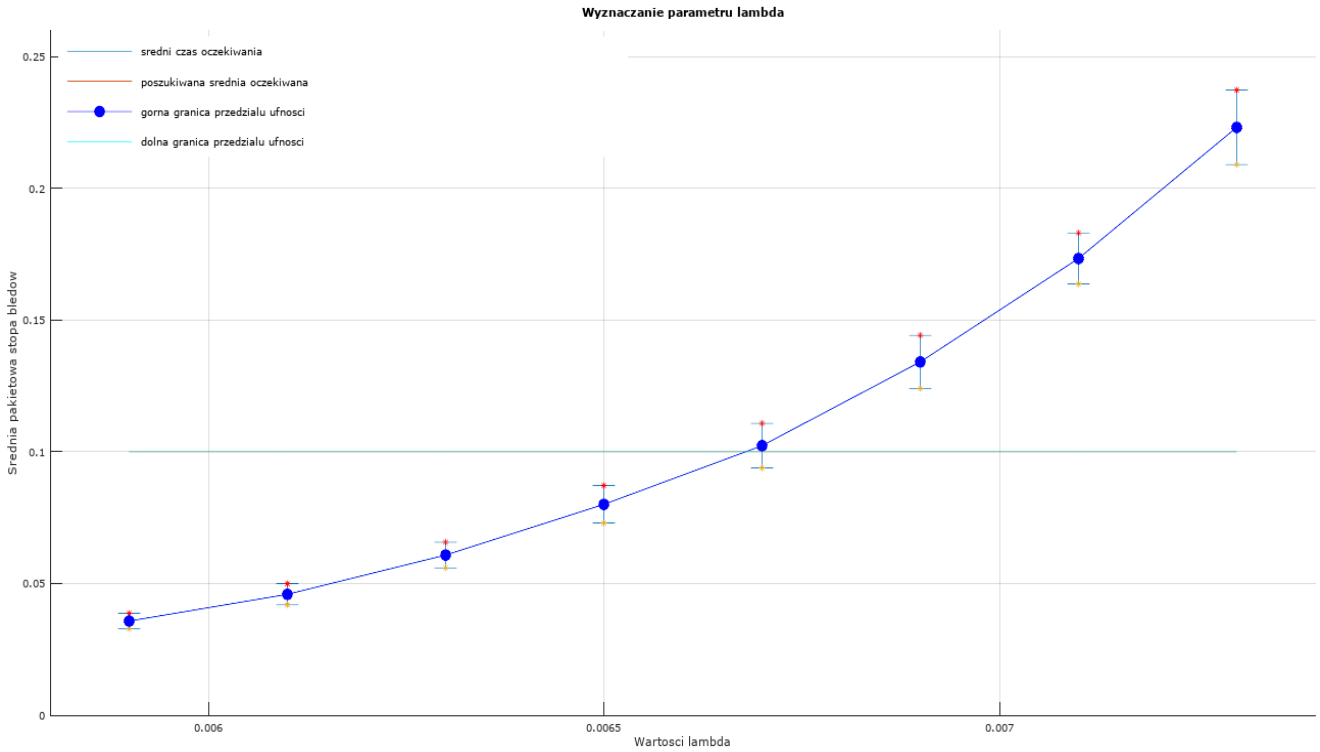



Wyznaczanie długości fazy początkowej uwzględnia różne wartości parametrów lambda. Obserwując powyższy wykres w dużym przybliżeniu można zauważyć, że faza początkowa jest niewielka i trwa około 12 pakietów.



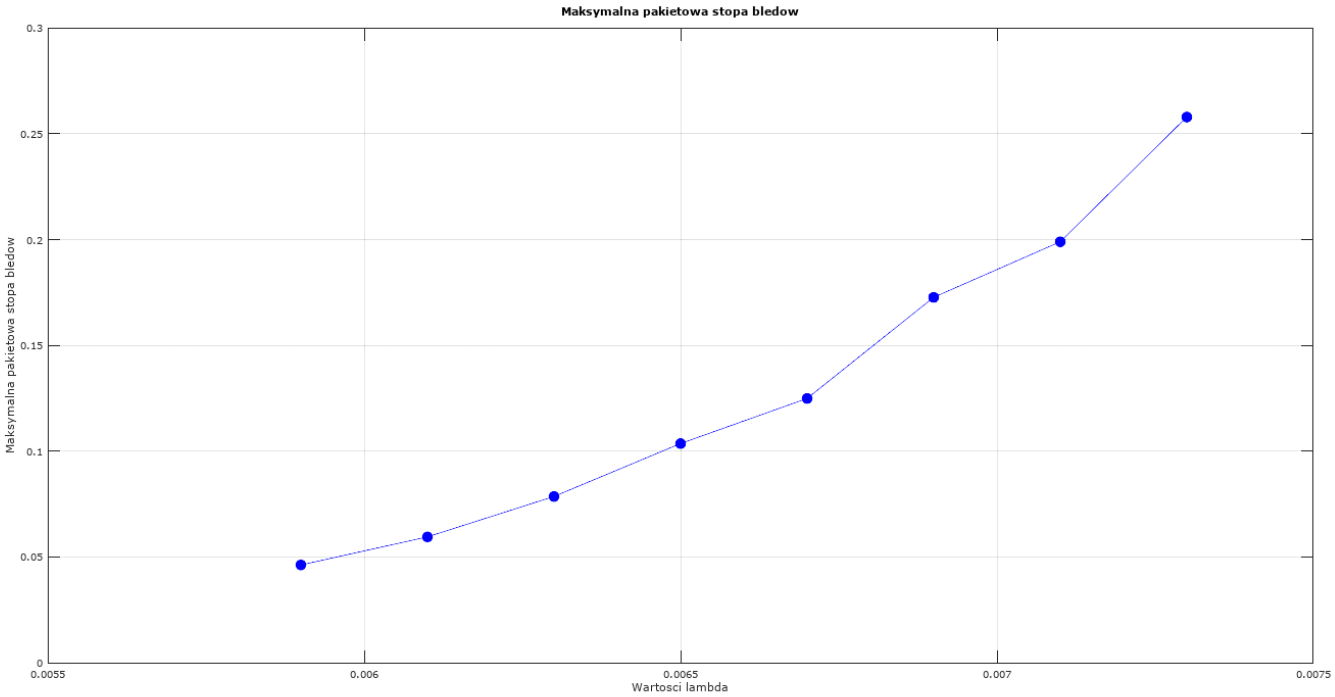

## 7.2 Wyznaczenie wartości parametru lambda

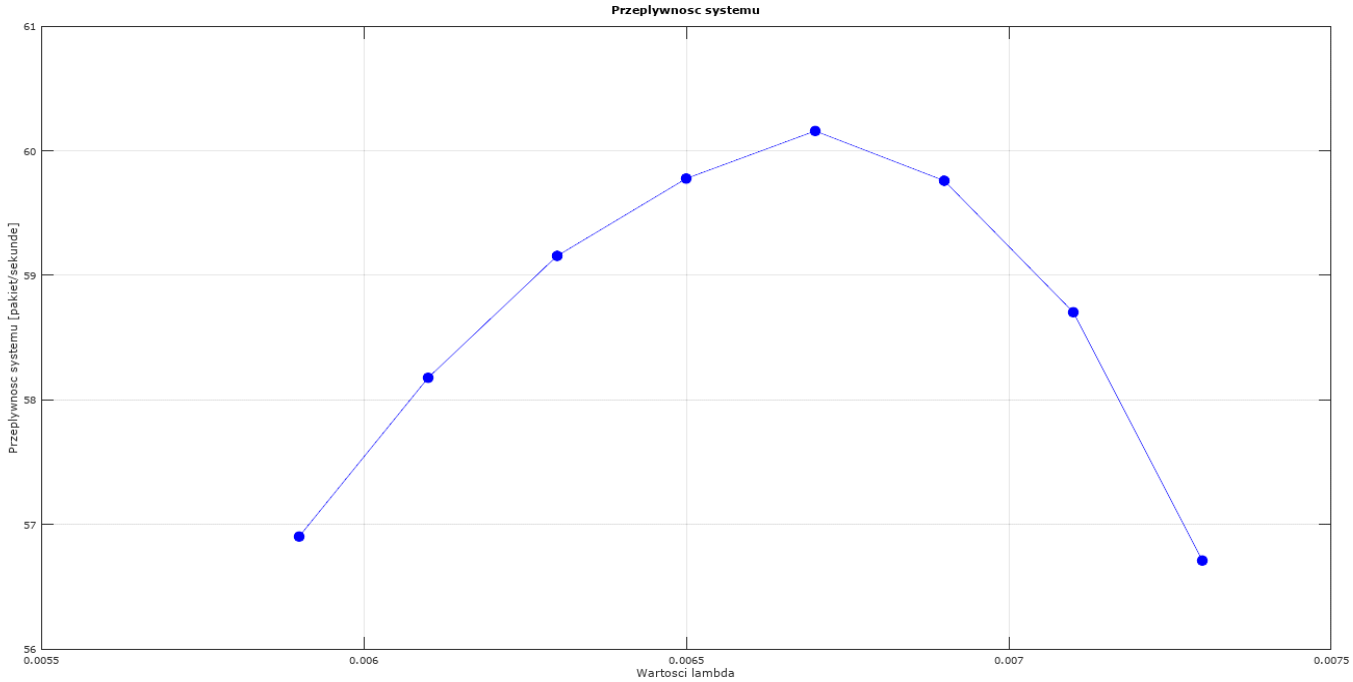
Wyznaczenie parametru lambda polegało na sporządzeniu wykresu średniej pakietowej stopy błędu w zależności od parametru lambda z przedziałami ufności. Dla każdej lambdy wykonano 10 symulacji z których każda trwała 500 sekund. Przeprowadzone symulacje pomijały fazę początkową wyznaczoną na 12 pakietów.



Na podstawie wykresu można odczytać, że parametrem spełniającym wymagania jest lambda na poziomie 0.0067. Dokładna wartość odczytana z wykorzystaniem programu Octave wynosi 0,0066748.

## 7.3 Wykres maksymalnej pakietowej stopy błędów i przepływności w zależności od wartości parametru lambda



### 7.4 Tabelka z wynikami symulacji dla każdego przebiegu symulacyjnego

Dla wyznaczonej fazy początkowej oraz wartości lambda wyniki przebiegów symulacyjnych wraz z wynikiem końcowym w postaci uśrednionych wyników po przebiegach i uwzględnionymi przedziałami ufności. Czas symulacji został ustawiony na 200 sekund. Podana średnia liczba retransmisji została obliczona jako średnia retransmisji na pakiet.

| Nr symulacji | Średnia pakietowa stopa błędów | Maksymalna pakietowa stopa błędów | Średnia liczba retransmisji | Średni czas oczekiwania [ms] | Średnie opóźnienie [ms] | Przepływność [pakiet/s] |
|---|---|---|---|---|---|---|
| **1** | 0.115818 | 0.126553 | 0.6963 | 22.4476 | 43.7794 | 59.1328 |
| **2** | 0.119228 | 0.133582 | 0.7061 | 25.9197 | 46.1778 | 59.7106 |
| **3** | 0.085283 | 0.091945 | 0.6461 | 14.9733 | 35.8598 | 61.2413 |
| **4** | 0.101654 | 0.113982 | 0.6669 | 20.3403 | 41.3873 | 60.3080 |
| **5** | 0.107831 | 0.117778 | 0.6835 | 21.576 | 41.0774 | 59.6719 |
| **6** | 0.129339 | 0.136499 | 0.7366 | 26.1762 | 46.6120 | 58.5884 |
| **7** | 0.096205 | 0.109118 | 0.6773 | 18.8924 | 40.0852 | 60.9353 |
| **8** | 0.074321 | 0.084219 | 0.6599 | 12.8336 | 35.4224 | 61.3581 |
| **9** | 0.105894 | 0.121044 | 0.6516 | 20.2650 | 40.3055 | 59.9057 |
| **10** | 0.116064 | 0.127803 | 0.7088 | 22.1291 | 44.3831 | 59.5411 |
| **Średnia** | 0.10516 | 0.11625 | 0.68330 | 20.555 | 41.509 | 60.039 |
| **Przedział ufności** | +/- 0.014454 | +/-0.014992 | +/-0.025094 | +/-3.6983 | +/-3.3711 | +/-0.79758 |

## 8. Wnioski

Wyniki programu pokazują, że wymagana wartość średniej pakietowej stopy błędów została uzyskana dla parametru lambda o wartości 0.0067, 10 nadajników i odbiorników i liczby retransmisji na poziomie 5. Dla zadanej liczby LR=15 oraz 4 nadajników nie można było uzyskać wymaganych wartości.
Analizując wykresy można zaobserwować, że wraz z wzrostem parametru lambda zwiększa się zarówno średnia jak i maksymalna pakietowa stopa błędu. Duże wartości lambdy skutkują zwiększoną aktywnością generacji pakietów i przepełnianiem się buforów. Zwiększa się liczba kolizji i spada przepustowość systemu. Zbyt mała liczba nadajników i duża wartość LR uniemożliwiają wywołanie kolizji.

## 9. Wydruk kodu programu jako załącznik do raportu