KATEDRA RADIOKOMUNIKACJI
WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI
POLITECHNIKA POZNAŃSKA

Symulacja sieci bezprzewodowej metodą interakcji procesów. Zadanie nr 4.

Martino Sebastiani

Poznań, 2020

SPIS TREŚCI

1. Pełny tekst rozwiązywanego zadania

W sieci bezprzewodowej stacje nadawcze konkurują o dostęp do łącza. W losowych odstępach czasu **CGP**$_k$ *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 **CTP**$_k$. 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 lub błąd. Przez kolizję rozumiemy nałożenie się jakiejkolwiek części jednego pakietu na inny pakiet (pochodzący z innego nadajnika). Dodatkowo każda transmisja pakietu może zakończyć się błędem TER. Brak wiadomości ACK po czasie (**CTP**$_k$+ **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.

Opracuj symulator sieci bezprzewodowej zgodnie z metodą **M**.

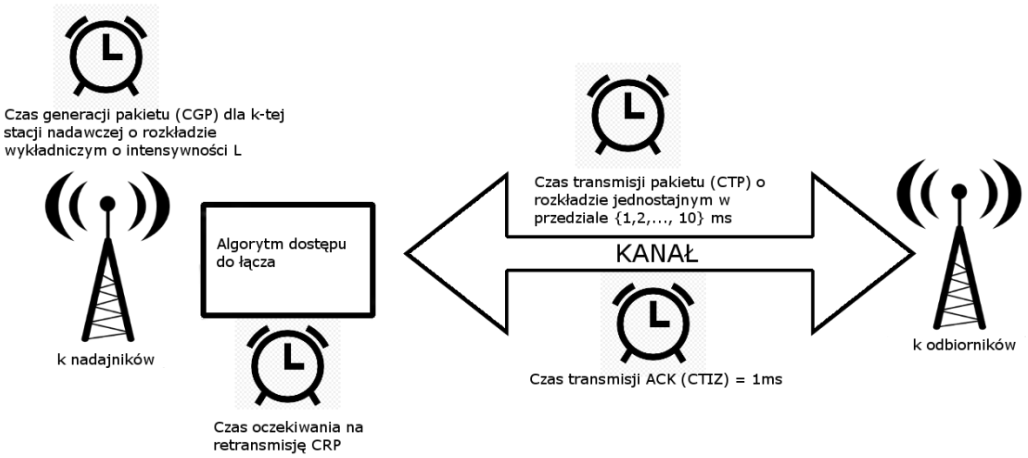
Za pomocą symulacji wyznacz:

● Wartość parametru **L**, która zapewni średnią pakietową stopę błędów (uśrednioną po ***K*** odbiornikach) nie większą niż 0.1, a następnie:

- o pakietową stopę błędów w każdym z odbiorników mierzoną jako iloraz liczby pakietów straconych do liczby przesłanych pakietów,
- o średnią liczbę retransmisji pakietów,
- o przepływność systemu mierzoną liczbą poprawnie odebranych pakietów w jednostce czasu,
- o średnie opóźnienie pakietu, tzn. czas jaki upływa między pojawieniem się pakietu w buforze, a jego poprawnym odebraniem,
- o średni czas oczekiwania, tzn. czas między pojawieniem się pakietu w buforze, a jego opuszczeniem
- o Sporządź wykres zależności średniej liczby retransmisji pakietów od parametru P

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

a. Schemat modelu symulacyjnego



b. Opis klas wchodzący w skład systemu i ich atrybutów

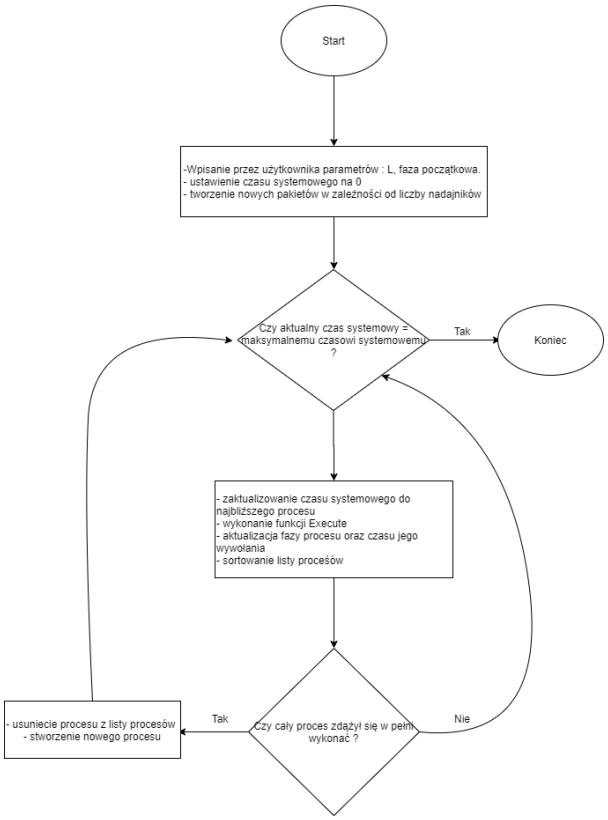
Obiekt	**Nazwa klasy implementującej obiekt**	**Opis**	**Atrybuty**
System telekomunikacyjny	TelecomunicationSystem	Klasa gromadząca wszystkie pozostałe elementy systemu	*-MaximumRetransmisionAttempt* – stała typu int mówiąca o maksymalnej liczbie prób retransmisji. - *NumberOfStations* – stała typu int mówiąca o liczbie stacji nadawczych oraz odbiorczych - *Clock* – aktualny czas systemowy typu int - *MaxSystemTime* – czas, w jakim będzie trwała symulacja systemu - CurrentClock – aktualny czas systemowy - *SeedsTable* – tablica seedów - *Transmitters* – vector przechowujący listę nadajników
Stancja nadawcza	Transmitter	Klasa reprezentująca stacje nadawczą. Generuje w swoim buforze pakiet w losowych odstępach czasu CGP_k. Po	- lista pakietów typu *queue<Package*>* - *CTIZ* – stała typu int = 1ms

		uzyskaniu dostępu do łącza stacja próbuje przesłać pakiet, której czas trwania transmisji wynosi CTP_k. Jeśli **nie** otrzyma od stacji nadawczej ACK wówczas próbuje go retransmitować po upływie czasu **CRP.** Jeśli po **LR** próbach nie udało się przesłać pakietu, usuwa go z buforu i próbuje przesłać następny pakiet.	- *ReceiveACK* – zmienna boolowska czy otrzymał ACK.
Kanał	Channel	Klasa reprezentująca kanał w systemie telekomunikacyjnym. W kanale będą sprawdzane czy nie występuje kolizja, bądź nie wystąpił błąd **TER**.	- lista *QueuePackage<Package*>* mówiąca o pakietach znajdujących się aktualnie w kanale - flaga *free* typu bool czy kanał jest wolny
Pakiet	Package	Klasa reprezentująca pakiet. W niej będą przechowywane zmiennie, z której stacji pochodzi pakiet oraz o liczbie prób retransmisji pakietu	- *SourceStation* – zmienna typu int mówiąca, z której stacji pochodzi pakiet - ID- stała typu int żeby wiedzieć który pakiet jest nadawany/ nie zgubić w systemie - RHelpVariable – zmienna pomocnicza do wyznaczenia CRP - R – aktualna liczba retransmisji - TimeOfAppear – czas pojawienia się pakietu w systemie - SendingPackage – flaga mówiąca czy pakiet jest wysyłany - PackageSentCorrectly – flaga czy pakiet został poprawnie nadany TER- flaga wystąpienia błędu TER - CTPK – czas transmisji pakietu

			- LeavingTime – czas opuszczenia pakietu po poprawnym odebraniu - SendingPackageTime – czas wysłania pakietu z bufora

2. Opis przydzielonej metody symulacyjnej

a. Schemat blokowy pętli głównej



b. Zdefiniowane procesy

W symulowanym systemie wyróżniam jeden proces: Pakiet. W ramach procesu Pakiet, wyróżniam następujące fazy:

- Pakiet pojawia się w buforze
  - Zaplanowanie pojawienie się następnego pakietu
  - Ustaw pakiet w kolejce do listy pakietów (bufor)
  - Przejście do następnej fazy
- Wysłanie pakietu w kanał
  - Sprawdzenie czy pakiet jest 1 w kolejce. Jeśli tak to kontynuuj, jeśli nie zakończ fazę.
  - Wyślij pakiet

    - Wpisz na listę zawiadomień o zdarzeniach komunikat o próbie przesłania pakietu po czasie CTP_k
- Wysłanie ACK
    - Sprawdzenie czy pakiet został poprawnie nadany poprzez flagę *CorrectPackage=true.* Jeśli tak, kontynuuj, jeśli nie zakończ fazę.
    - Wyślij ACK
    - Wpisz na listę zawiadomień o zdarzeniach komunikat o próbie przesłania pakietu po czasie *CTIZ.*
- Otrzymanie przez nadajnik ACK
    - *Sprawdzenie czy ReceiveACK=true lub RetransmisionPermission = false.* Jeśli warunki się zgadzają - kontynuuj, jeśli nie - przejdź do fazy ostatniej.
    - Usuń pakiet z bufora
    - Jeśli bufor nie jest pusty, przejdź do fazy 2.
- Retransmisja pakietu
    - Zaktualizuj r
    - Wpisz na listę zawiadomień o zdarzeniach komunikat o próbie retransmisji pakietu po czasie *CRP.*
    - *Przejdź do fazy 2.*

3. Parametry wywoływania programu

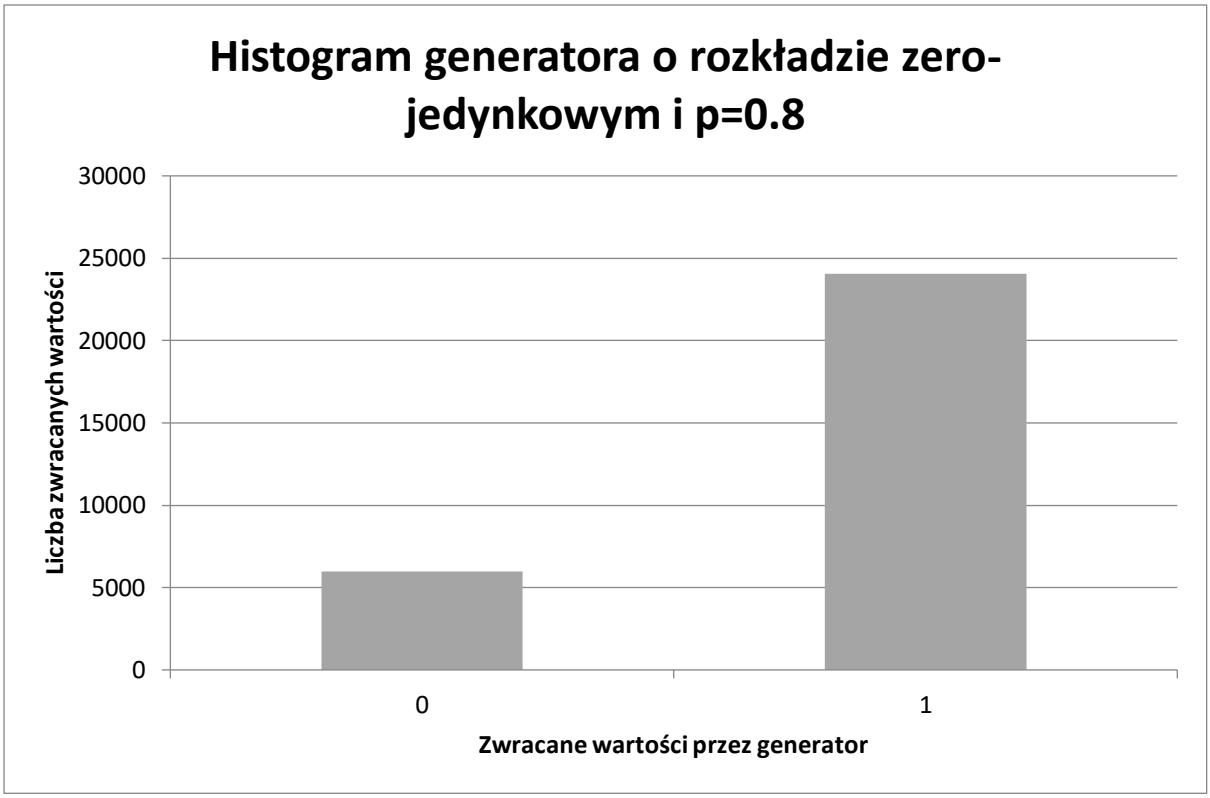
Użytkownik w pierwszej kolejności wpisuje lambdę, która odpowiada za częstotliwość generacji pakietów w stacjach nadawczych. Następnie program pyta o długość czasu trwania symulacji, użytkownik podaje liczbę w sekundach. Potem program zapyta o czas, od którego ma zacząć zbierać statystyki (preferowana jest wyznaczona faza początkowa). Na końcu użytkownik wpisuje, z którego zestawu ziaren chciałby skorzystać (ma do wyboru od 0 do 9). Po wykonaniu pojedynczej symulacji użytkownik proszony jest o napisanie nazwy pliku (najlepiej z rozszerzeniem .txt) gdzie wygeneruje mu plik z liczbą retransmisji poprawnie nadanego pakietu dla każdej stacji nadawczej.

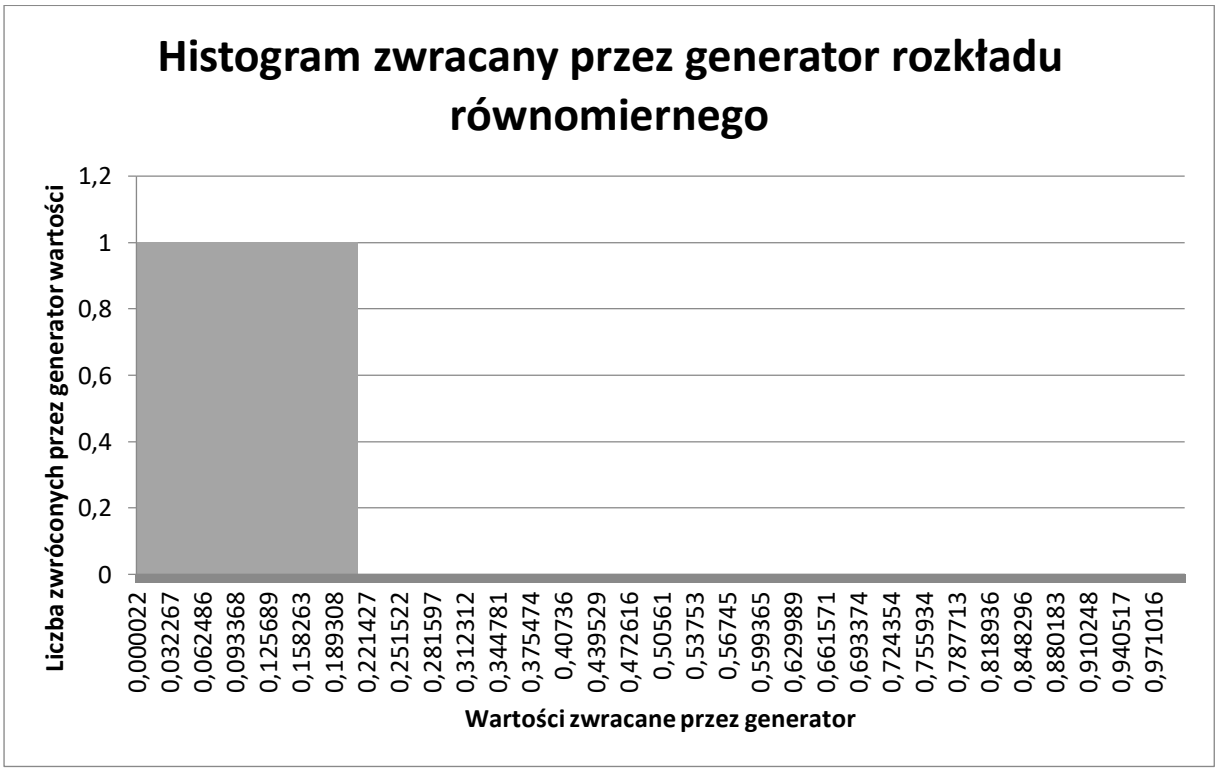
4. Generatory

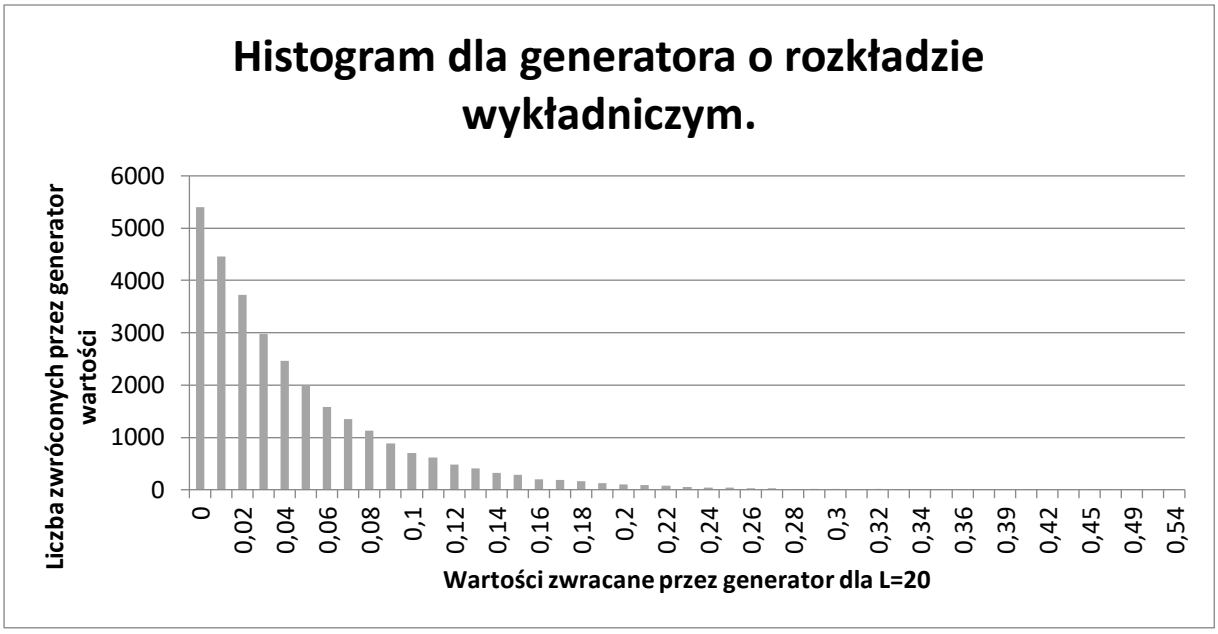
a. Opis zastosowanych generatorów liczb losowych z histogramami

```
#ifndef GENERATOR_H
#define GENERATOR_H
#include <math.h>
class Generator
{
public:

    Generator();
    ~Generator();
    double Floor(double number_)
    {
            double fraction_ = number_;
            double integer_ = 0;
            fraction_ = modf(number_, &integer_);
            if (number_ >= 0.0)
```

```
		{
			return integer_;
		}
		else
		{
			return integer_ - 1.0;
		}

	}
	double Uniform( int& x) // from 0 to 1
	{
		double h = Floor(x / Q);
		x = A * (x - Q * h) - R * h;
		if (x < 0.0)
		{
			x += M;
		}
		return x / M;
	}
	double Uniform(int& x, double min, double max)
	{
		return Uniform(x)* (max-min)+min;
	}
	 double Exponential(double lambda,int& x)
	{
		lambda = 1 / lambda;
		double ret = -log(Uniform(x)) / lambda;
		return ret;
	}
	 double ZeroOne(double p, int& x)
	 {
		 double k_ = 0.0;
		 k_=Uniform(x);
	 if (k_ < p)
			return 1.0;
		 else
			return 0.0;
	 }
private:
	double M = 2147483647.0;
	double A = 16807.0;
	const int Q = 127773;
	const int R = 2836;
	const int Range = 2147483647; //2^31-1
	const double k_Pi_ = (double)
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881;// Dotychczas poznane rozwinięcie liczby PI wg wikipedia.org.pl
};

#endif // ! GENERATOR_H
```


Histogram generatora o rozkładzie zero-jedynkowym i p=0.8
30000
25000
20000
15000
10000
5000
0
Liczba zwracanych wartości
0
1
Zwracane wartości przez generator


Histogram zwracany przez generator rozkładu równomiernego
1,2
1
0,8
0,6
0,4
0,2
0
Liczba zwróconych przez generator wartości
0,000022
0,032267
0,062486
0,093368
0,125689
0,158263
0,189308
0,221427
0,251522
0,281597
0,312312
0,344781
0,375474
0,40736
0,439529
0,472616
0,50561
0,53753
0,56745
0,599365
0,629989
0,661571
0,693374
0,724354
0,755934
0,787713
0,818936
0,848296
0,880183
0,910248
0,940517
0,971016
Wartości zwracane przez generator



b. Wyjaśnienie, w jaki sposób została zapewniona niezależność sekwencji losowych w różnych symulacjach

Do wygenerowania ziaren został napisany oddzielny program, który wygenerował losowe ziarna. Każde źródło potrzebuje 1 ziarno: do generacji pakietu. Dla 4 nadajników daje 4 ziarna na jeden przebieg symulacyjny dodatkowo 3 ziarna są potrzebne do wyznaczenia TER, CTPk, CRP. Łączna liczba ziaren zaimplementowanych w tablicy seedów wynosi 70 ziaren, aby można było wykonać 10 przebiegów symulacyjnych.

5. Krótki opis zastosowanej metody testowania i weryfikacji poprawności działania programu

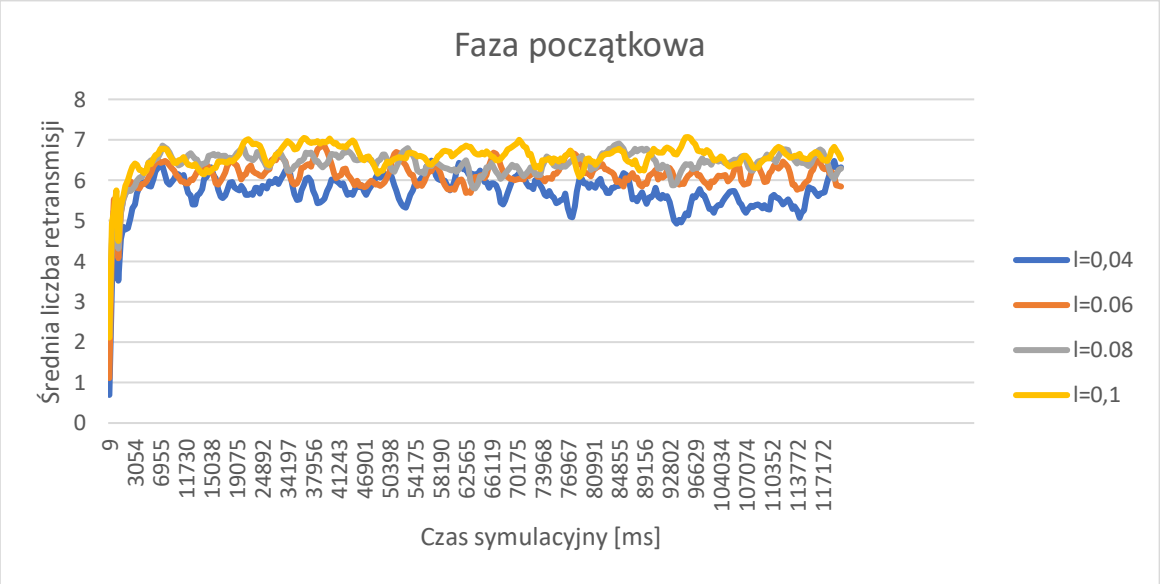
Do weryfikacji poprawności działania programu użyłem pracy krokowej aby dokładnie sprawdzać co dzieje się po każdej pętli symulacyjnej w każdym buforze który jest w nadajniku. Miałem podgląd na kalendarz zdarzeń, liczbę wygenerowanych, retransmitowanych i utraconych pakietów po każdej pętli. Dokładniejszą metodą testowanie był tryb debugera. Ta metoda jest dużo trudniejsza i potrzeba dużo więcej czasu, ale można kontrolować zachowywanie się systemu linijka po linijce kodu. Ostatnim sposobem sprawdzania poprawności działania programu było zmienianie parametrów takich jak lambda, długość fazy początkowej, zmiana długości czasu symulacji i obserwacja wyników.

W celu wyznaczenia wartości parametru lambda, gdzie pakietowa stopa błędów wynosi ok. 10%, maksymalna liczba retransmisji zostało zmniejszone do **6**.

6. Wyniki symulacji

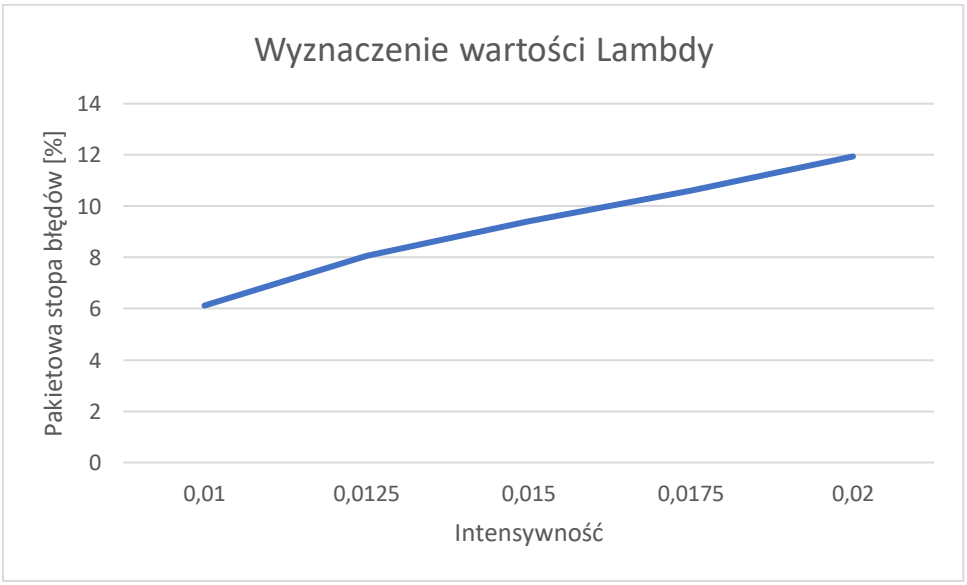
a. Wyznaczenie długości fazy początkowej

W celu wyznaczenia fazy początkowej wykonano 40 symulacji (10x symulacji na każdą lambdę) o intensywnościach L = 0.04, 0.06, 0.08, 0.1 oraz maksymalnym czasem systemowym równym 2 minuty. Sporządzony został wykres zależności średniej liczby retransmisji od czasu systemowego.



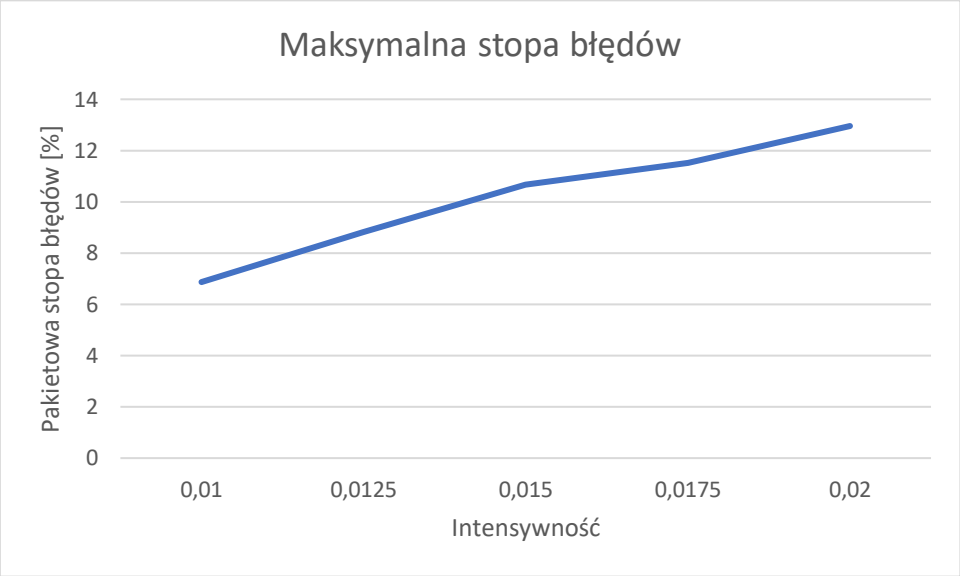
Z przebiegu wykresów wynika, że program stabilizuje się po ok. 10s.

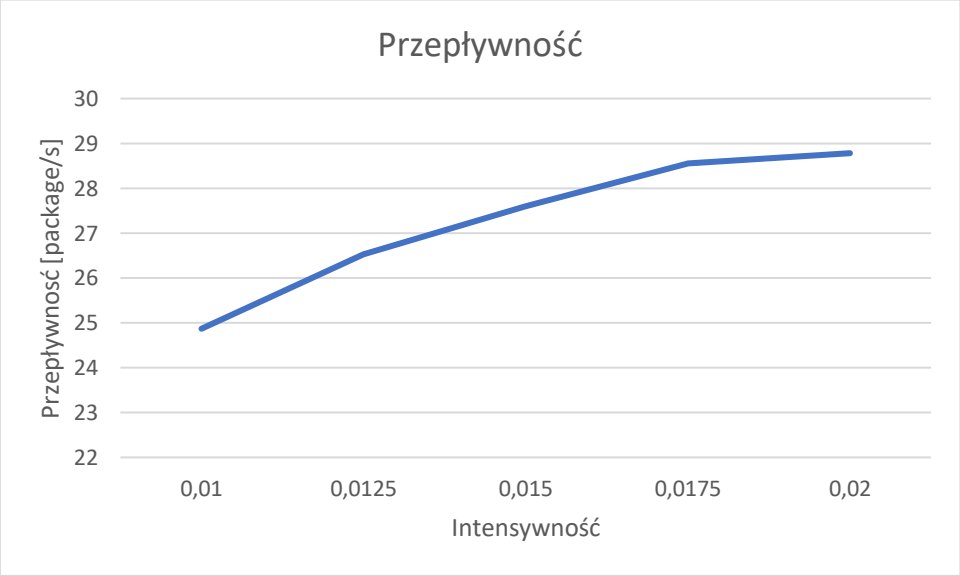
b. Wyznaczenie wartości parametru lambda



Wybrana wartość lambdy = **0.015**

c. Wykres maksymalnej pakietowej stopy błędów i przepływności w zależności od wartości parametru lambda





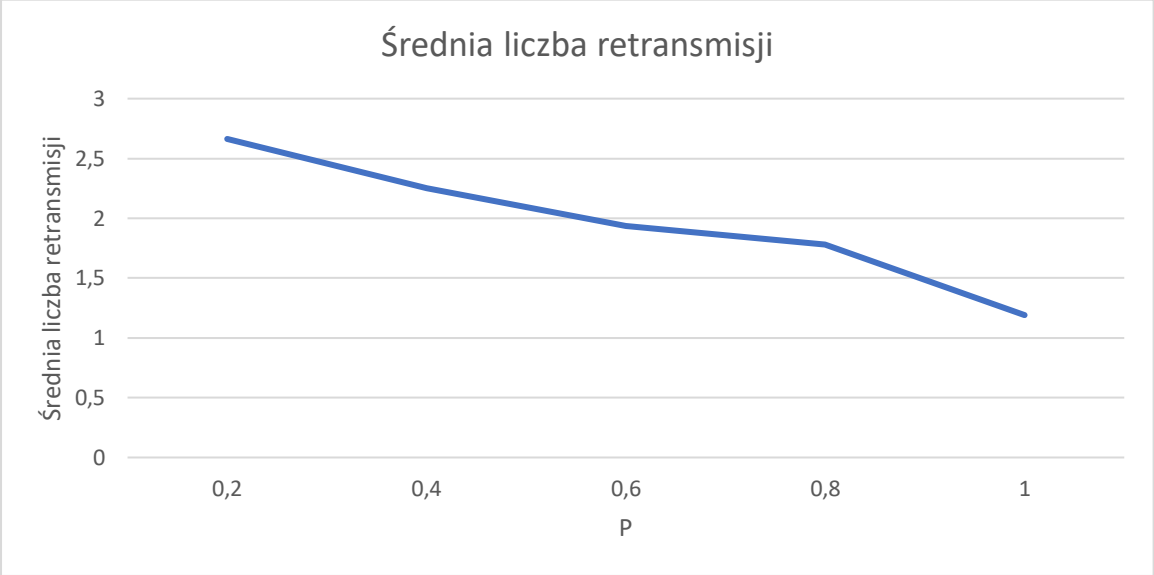
d. Tabelka z wynikami symulacji dla każdego przebiegu symulacyjnego

Pakietowa stopa błędów [%]	Maksymalna stopa błędów [%]	Średnia liczba retransmisji	Przepływność [package/s]	Średnie opóźnienie pakietu [ms]	Średni czas oczekiwania [ms]
18,119	20,0204	5293,75	28,654545	30,4355	23,3005
4,29755	5,182	5794,25	25,581818	53,9645	60,3612
14,0905	15,8784	5425,5	28,990909	32,6611	27,0328
10,513	11,7717	5830,5	28,154545	39,4225	34,5052
4,50297	5,98958	5992,25	27,490909	46,5392	52,3681
9,76218	10,842	5585,75	28,118182	40,6041	33,4164
5,43636	6,1745	5527,5	26,209091	47,0343	47,0343
12,4164	14,218	5444,5	28.600000	35,2739	28,8445
6,8501	7,82828	5964	27,254545	46,052	43,4016
7,99019	8,78713	6330,5	27,918182	44,1618	41,0157

e. Wyniki końcowe w postaci uśrednionych wyników po wszystkich przebiegach + przedziały ufności dla każdego z sześciu parametrów

Pakietowa stopa błędów [%]	Maksymalna stopa błędów [%]	Średnia liczba retransmisji	Przepływność [package/s]	Średnie opóźnienie pakietu [ms]	Średni czas oczekiwania [ms]
9,397825	10,669199	5718,85	27,59696956	41,61489	39,12803
+13,43 / -5,37	+15,02 / -6,32	+6005,54 / - 5432,16	+35,46 / -19,73	+48,17 / - 35,06	+49,75 / - 28,51

f. Wykres zależności średniej liczby retransmisji pakietów od parametru P dla wyznaczonej wartości parametru lambda



7. Wnioski

Podczas pisania projektu dużo czasu zajęło mi opanowanie metody interakcji procesów, w jaki sposób napisać symulator, który nie posiada stricte zdarzeń czasowych czy warunkowych jako osobnych obiektów, tylko opiera się na procesie jako sekwencji zdarzeń związanych z danym obiektem; w tym przypadku był to obiekt pakiet. Dużą pomocą były materiały udostępnione przez wykładowcę oraz prowadzącego, które pomogły w zrozumieniu metody.

Z początku myślałem, że parametry z góry narzucone oddadzą pożądane wyniki. Na szczęście algorytm ALOHA jest bardzo czuły na intensywność, stąd zwykła zmiana maksymalnej liczby retransmisji (10->6) pozwoliły na uzyskanie zadawalających dla mnie wyników.

Po wielu trudach szukania błędu oraz konsultacjach udało się naprawić symulator, gdzie z początku wykonywałem operacje tylko na jednym procesie i po naprawieniu z łatwością udało się odszukać fazę początkową.

Z początku musiałem sobie przypomnieć sporo rzeczy z programowania w C++, ponieważ ostatnio używałem go 2 lata temu. Bardzo pomogła mi opcja debbugowania, gdzie mogłem na bieżąco śledzić co dzieje się w programie oraz wyszukiwać odpowiednich błędów.