Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

по лабораторной работе №4

по курсу «Системный анализ и машинное моделирование»

Вариант 19 г

|  |  |
| --- | --- |
| Выполнил:  студент гр. 351002  Прокопович А.Ю. | Проверил:  ст. преподаватель кафедры ПОИТ  Мельник Н.И. |

Минск 2016

# **ЗАДАНИЕ**

Одноканальная СМО с неограниченной очередью. Входной поток простейший с интенсивностью 2 заяв/час. Найти среднюю длину очереди и среднее время ожидания в очереди, если поток обслуживания имеет фиксированное время обслуживания: tобсл=0,4 час; Данную СМО можно записать в нотации Кендалла: M/D/1/∞

Построить аналитическую и имитационную модели и сравнить результаты исследования.

Диаграмма интенсивностей переходов представлена на рисунке ниже:

…

λ

λ

λ

λ

μ

μ

μ

μ

…

…

…

1

2

N

0

*λ -* интенсивность потока.

*µ -* интенсивность канала.

Состояние системы определяется как количество заявок, находящихся в ней в данный момент времени.

1. **ХАРАКТЕРИСТИКИ СМО**

Для системы вида M/D/1/∞ формулу для расчета средней длины очереди можно поучить из формулы Полячека-Хинчина для СМО вида M/G/1/∞:

, и так как в данном случае время обработки заявки детерминировано, vµ = 0, следует .

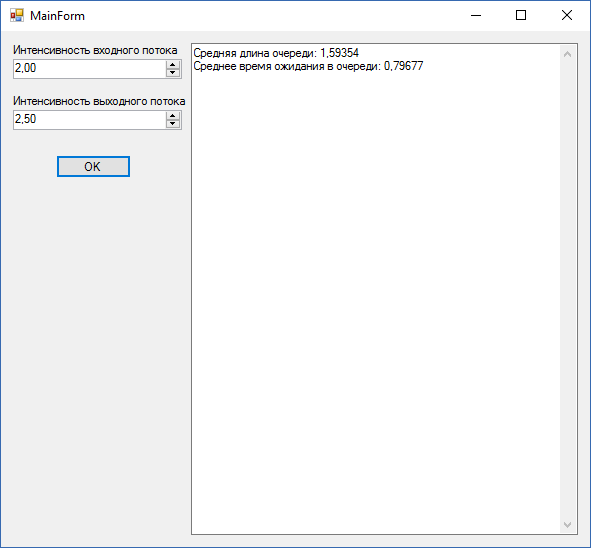
Отсюда по формуле Литтла получаем, что среднее время пребывания заявки в очереди равно: 

Можно рассчитать значения для заданных величин λ = 2, µ = 1 / tобсл = 1/0.4 = 2.5, ω = λ/µ = 2 / 2.5 = 0.8





1. **РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

****

# **ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ**

public class ExponentialGenerator

{

private static Random generator = new Random();

private readonly double L;

public ExponentialGenerator(double L)

{

this.L = L;

}

public double Next()

{

return -1 / L \* Math.Log(generator.NextDouble());

}

}

public class Simulator

{

private const int Accuracy = 100;

public const int TaskCount = 1000000;

private ExponentialGenerator random;

private double inputIntensity;

private double outputIntensity;

public Simulator(double inputIntensity, double outputIntensity)

{

this.inputIntensity = inputIntensity;

this.outputIntensity = outputIntensity;

random = new ExponentialGenerator(inputIntensity);

}

public List<State> Simulate()

{

List<State> result = new List<State>();

long finishProcessingTime = 0;

long nextTaskTime = 0;

long currentTime = 0;

long prevTime = 0;

State systemState = new State();

int taskCount = TaskCount;

int completedTasks = 0;

while (completedTasks != TaskCount)

{

if (nextTaskTime == currentTime)

{

if (taskCount != 0)

{

systemState.Buffer++;

nextTaskTime += Convert.ToInt64(Accuracy\*random.Next());

}

taskCount--;

}

if (finishProcessingTime == currentTime || !systemState.IsBusy)

{

if (systemState.IsBusy)

completedTasks++;

if (systemState.Buffer > 0)

{

systemState.Buffer--;

systemState.IsBusy = true;

finishProcessingTime += Convert.ToInt64(Accuracy/outputIntensity);

}

else

{

finishProcessingTime = nextTaskTime;

systemState.IsBusy = false;

}

}

result.Add(new State()

{

Buffer = systemState.Buffer,

IsBusy = systemState.IsBusy,

Period = currentTime - prevTime

});

prevTime = currentTime;

currentTime = Math.Min(nextTaskTime, finishProcessingTime);

}

return result;

}

}

private void okButton\_Click(object sender, EventArgs e)

{

Simulator simulator = new Simulator((double)inputNumericUpDown.Value, (double)outputNumericUpDown.Value);

var result = simulator.Simulate();

var maxState = result.Max(t => t.Buffer + (t.IsBusy ? 1 : 0));

List<long> propCount = new List<long>();

resultTextBox.Text = "";

long allTime = result.Sum(t => t.Period);

for (int i = 0; i <= maxState; i++)

{

propCount.Add(result.Where(t => t.Buffer + (t.IsBusy ? 1 : 0) == i).Sum(t => t.Period));

}

double averageBufferLength = 0.0;

var maxBuffer = result.Max(t => t.Buffer);

for (int i = 0; i <= maxBuffer; i++)

{

averageBufferLength += 1.0 \* result.Where(t => t.Buffer == i).Sum(t => t.Period) \* i / allTime;

}

resultTextBox.Text += $"Средняя длина очереди: {averageBufferLength:0.00000}" + Environment.NewLine;

resultTextBox.Text += $"Среднее время ожидания в очереди: {averageBufferLength / 2:0.00000}" + Environment.NewLine;

resultTextBox.Text += $"Все время обработки: {allTime}" + Environment.NewLine;

for (int i = 0; i <= maxState; i++)

{

resultTextBox.Text += $"Вероятность состояния {i:00}: {1.0 \* propCount[i] / allTime:0.00000}" + Environment.NewLine;

}

}