Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Отчет по лабораторным работам № 3, 4

по дисциплине «Системный анализ и машинное моделирование»

Вариант 9

Выполнил:

студент группы 951006

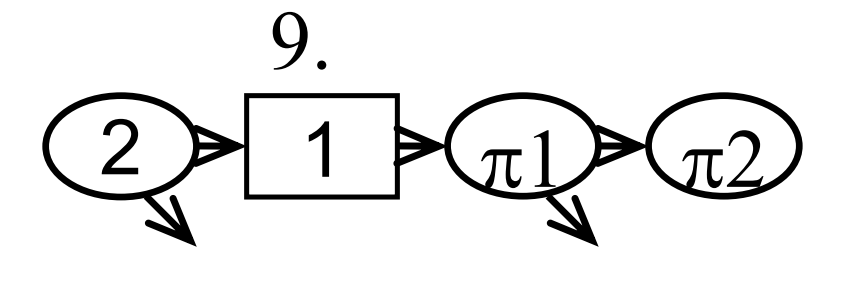
Бакыт Мади

Проверил:

Мельник Н. И.

**1. Построение и исследование аналитической модели дискретно-стохастической системы массового обслуживания**

построить граф состояний P-схемы:



|  |  |
| --- | --- |
| e | = {1, 2} – количество тактов до выдачи заявки; |
| q | = {0, 1} – количество заявок в очереди; |
| pi1 | = {0, 1} – количество заявок в канале 1; |
| pi2 | = {0, 1} – количество заявок в канале 2 |

Общий вид кодировки состояния системы: **{e, q, pi1, pi2}**

* по графу построить аналитическую модель и, решив ее, определить вероятности состояний:

Решив систему уравнений (при π1=0.4, π2=0.5), получим:

* рассчитать теоретическое значение показателей:

*A (абсолютная пропускная способность):*

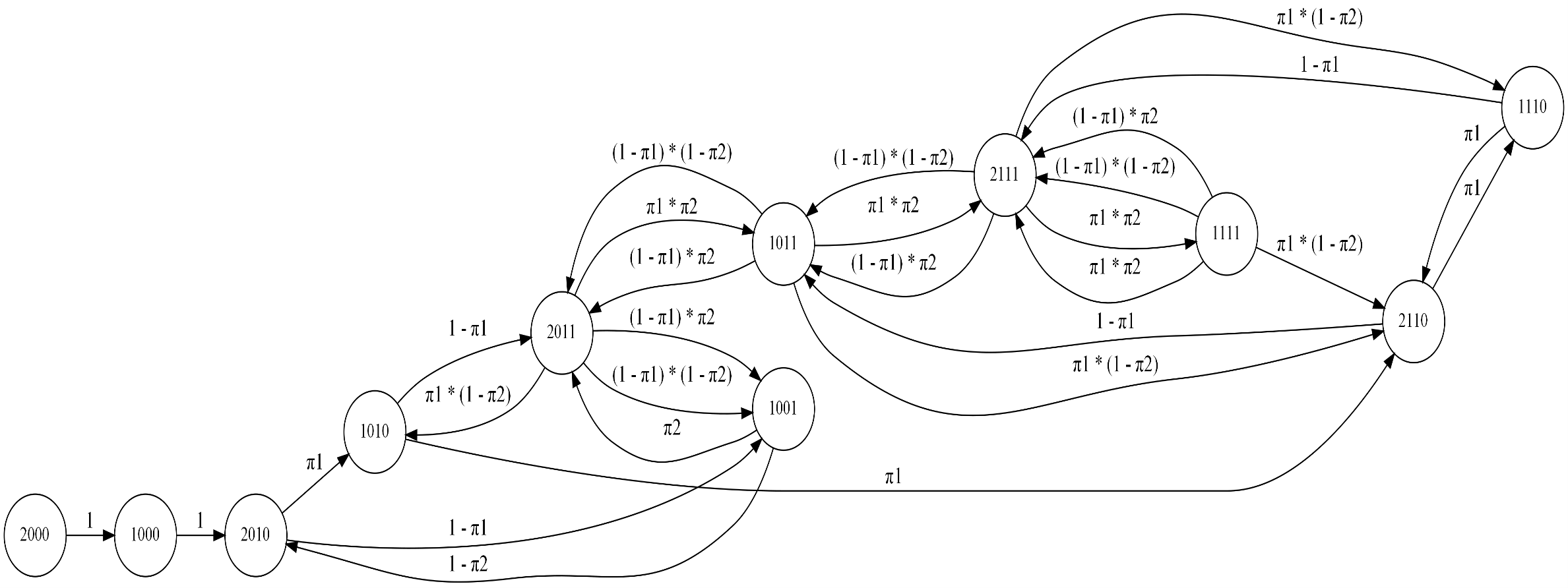
*Q (относительная пропускная способность) и Pотк (вероятность отказа):*

*Lоч (средняя длина очереди) и Lc (среднее число заявок в системе):*

*Wоч (среднее время пребывания заявки в очереди) и Wс (среднее время пребывания заявки в системе):*

00

*Kкан (коэффициент загрузки канала):*



**ЛИСТИНГ ПРОГРАММНОГО КОДА**

**package** by.bsuir.lab

**import** kotlin.random.Random

**class** Task {

**var** refusedOnEmitter = **false**

**var** refusedOnChannel = **false**

**var** ticksInQueue = 0

**var** ticksInChannelOne = 0

**var** ticksInChannelTwo = 0

**val** isRefused: **Boolean**

**get**() = refusedOnEmitter || refusedOnChannel

**val** ticksInSystem: **Int**

**get**() = ticksInQueue + ticksInChannelOne + ticksInChannelTwo

}

**fun** Task?.toState() = **if** (**this** == **null**) 0 **else** 1

**fun** main() {

**val** ticksCount = 1\_000\_000

**var** ticksToEmit = 2

**val** stateMap = mutableMapOf<String, Int>()

**val** emittedTasks = mutableListOf<Task>()

**var** queueTask: Task? = **null**

**var** channelOneTask: Task? = **null**

**var** channelTwoTask: Task? = **null**

**for** (i **in** 1..ticksCount) {

**val** stateName = "$ticksToEmit${queueTask.toState()}${channelOneTask.toState()}${channelTwoTask.toState()}"

stateMap[stateName] = stateMap.getOrDefault(stateName, 0) + 1

ticksToEmit--

**var** emitted = **false**

**if** (ticksToEmit == 0) {

ticksToEmit = 2

emitted = **true**

}

**val** channelOneFinished = Random.nextDouble() > 0.4

**val** channelTwoFinished = Random.nextDouble() > 0.5

**if** (channelTwoTask != **null**) {

channelTwoTask.ticksInChannelTwo++

**if** (channelTwoFinished) {

channelTwoTask = **null**

}

}

**if** (channelOneTask != **null**) {

channelOneTask.ticksInChannelOne++

**if** (channelOneFinished) {

**if** (channelTwoTask == **null**) {

channelTwoTask = channelOneTask

} **else** {

channelOneTask.refusedOnChannel = **true**

}

channelOneTask = **null**

}

}

**if** (queueTask != **null**) {

queueTask.ticksInQueue++

**if** (channelOneTask == **null**) {

channelOneTask = queueTask

queueTask = **null**

}

}

**if** (emitted) {

**val** task = Task()

emittedTasks.add(task)

**if** (channelOneTask == **null**) {

channelOneTask = task

} **else** **if** (queueTask == **null**) {

queueTask = task

} **else** {

task.refusedOnEmitter = **true**

}

}

}

println("==================== Состояния ====================")

**for** (state **in** stateMap) {

println("P\_${state.key} = ${state.value.toDouble() / ticksCount}")

}

**val** outputIntensity = emittedTasks.count { !it.isRefused }.toDouble() / ticksCount

**val** successProbability = emittedTasks.count { !it.isRefused }.toDouble() / emittedTasks.size

**val** refuseProbability = emittedTasks.count { it.isRefused }.toDouble() / emittedTasks.size

**val** blockProbability = 0

**val** averageQueueLength = emittedTasks.sumOf { it.ticksInQueue }.toDouble() / ticksCount

**val** averageSystemLength = emittedTasks.sumOf { it.ticksInSystem }.toDouble() / ticksCount

**val** passedEmitter = emittedTasks.filter { !it.refusedOnEmitter }

**val** passedChannel = passedEmitter.filter { !it.refusedOnChannel }

**val** averageTimeInQueue = emittedTasks.sumOf { it.ticksInQueue }.toDouble() / passedEmitter.size

**val** averageTimeInSystem =

passedEmitter.sumOf { it.ticksInQueue + it.ticksInChannelOne }.toDouble() / passedEmitter.size +

passedChannel.sumOf { it.ticksInChannelTwo }.toDouble() / passedChannel.size

**val** channelOneBusyProbability = emittedTasks.sumOf { it.ticksInChannelOne }.toDouble() / ticksCount

**val** channelTwoBusyProbability = emittedTasks.sumOf { it.ticksInChannelTwo }.toDouble() / ticksCount

println("")

println("==================== Характеристики ====================")

println("A = $outputIntensity")

println("Q = $successProbability")

println("Pотк = $refuseProbability")

println("Pбл = $blockProbability")

println("Lоч = $averageQueueLength")

println("Lс = $averageSystemLength")

println("Wоч = $averageTimeInQueue")

println("Wс = $averageTimeInSystem")

println("K1 = $channelOneBusyProbability")

println("K2 = $channelTwoBusyProbability")

}

**==================== Состояния ====================**

**P\_2000 = 1.0E-6**

**P\_1000 = 1.0E-6**

**P\_2010 = 0.10393**

**P\_1010 = 0.089797**

**P\_2110 = 0.085812**

**P\_1110 = 0.047835**

**P\_2111 = 0.067889**

**P\_1011 = 0.1411**

**P\_2011 = 0.242368**

**P\_1001 = 0.207796**

**P\_1111 = 0.013471**

**==================== Характеристики ====================**

**A = 0.335972**

**Q = 0.671944**

**Pотк = 0.328056**

**Pбл = 0**

**Lоч = 0.215007**

**Lс = 1.679833**

**Wоч = 0.45235775915573856**

**Wс = 4.121112222950421**

**K1 = 0.792202**

**K2 = 0.672624**