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 25

Выполнил:

студент группы 451006

Снитовец М. В.

Проверил:

Мельник Н. И.

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

* построить граф состояний P-схемы:

1

π

π

2

|  |  |
| --- | --- |
| p | = {1, 2} – количество тактов до выдачи новой заявки; |
| c1 | = {0, 1} – количество заявок в канале 1; |
| q | = {0, 1} – количество заявок в очереди; |
| c2 | = {0, 1} – количество заявок в канале 2 |

Общий вид кодировки состояния системы: **{p, c1, q, c2}**

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

Заменим последнее уравнение нормировочным уравнением:

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

|  |  |
| --- | --- |
| P2000 = 0  P1000 = 0  P2100 = 0.164  P1100 = 0.143  P2101 = 0.244 | P1001 = 0.143  P1011 = 0.107  P2111 = 0.092  P1101 = 0.084  P1111 = 0.023 |

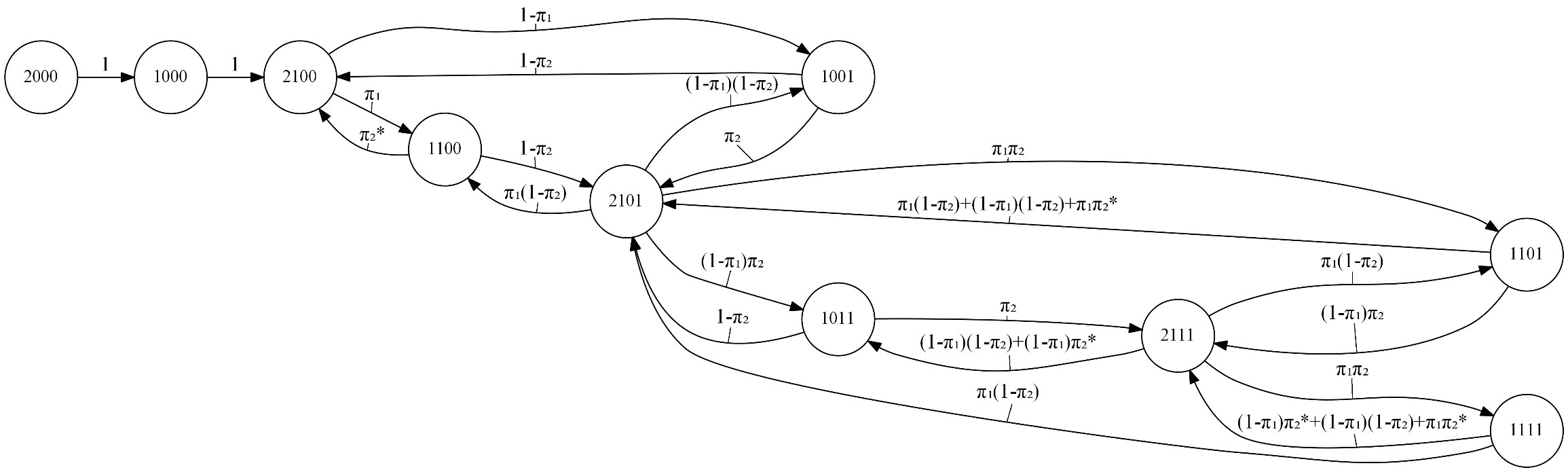
* рассчитать теоретическое значение показателей:

*A (абсолютная пропускная способность):*

*Pотк (вероятность отказа):*

*Lоч (средняя длина очереди):*

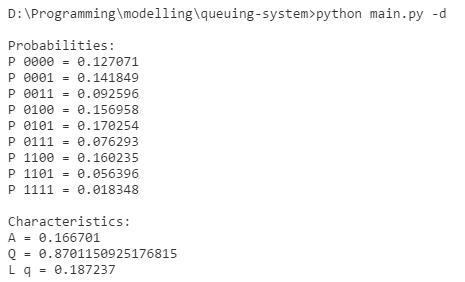
*Wоч (среднее время пребывания заявки в очереди):*



**2. Построение и исследование имитационной модели дискретно-стохастической системы массового обслуживания**

* для заданной СМО простроить имитационную модель и исследовать ее:

Результаты работы программы (имитация одного миллиона тактов):



**Вывод**:

В ходе лабораторной работы была аналитически смоделирована дискретно- стохастическая СМО и разработана программа, имитирующая ее поведение. Построенная модель позволяет статистически подсчитать необходимые характеристики СМО. Полученные статистические значения искомых характеристик близки к теоретически рассчитанным: отклонения при моделировании одного миллиона тактов составляют не более нескольких тысячных.

Исходный код программы:

**// main.py**

**import** sys  
  
**from** transitions **import** TRANSITIONS  
**from** simulator **import** simulate, UnitsSettings  
**from** characteristics **import** \*  
  
START\_STATE = **'2000'**TACTS\_COUNT = 1000000  
  
DEFAULT\_SETTINGS = UnitsSettings(pi1=0.5, pi2=0.5)  
  
  
**def** read\_probability(value\_name):  
 result = **None  
 while** result **is None**:  
 raw\_input = input(**'{}: '**.format(value\_name))  
 **try**:  
 result = float(raw\_input)  
 **except** ValueError:  
 print(**'Invalid input.'**)  
 **return** result  
  
  
**def** read\_units\_settings():  
 values = {}  
 **for** setting **in** UnitsSettings.\_fields:  
 values[setting] = read\_probability(setting)  
 **return** UnitsSettings(\*\*values)  
  
  
**def** main():  
 **if** len(sys.argv) > 1 **and** sys.argv[1] == **'-d'**:  
 units\_settings = DEFAULT\_SETTINGS  
 **else**:  
 units\_settings = read\_units\_settings()  
  
 characteristics = [AbsoluteThroughput(TACTS\_COUNT), RejectProbability(TACTS\_COUNT), AverageQueueLength(TACTS\_COUNT),  
 AverageQueueTime(TACTS\_COUNT)]  
  
 states\_statistics, characteristics\_results = simulate(TRANSITIONS, units\_settings, TACTS\_COUNT, START\_STATE,  
 characteristics)  
  
 print(**'\nProbabilities:'**)  
 **for** state, count **in** sorted(states\_statistics.items(), key=**lambda** x: int(x[0])):  
 print(**'P {} = {}'**.format(state, count / TACTS\_COUNT))  
  
 print(**'\nCharacteristics:'**)  
  
 **for** name, result **in** characteristics\_results.items():  
 print(**'{} = {}'**.format(name, result))  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

**// simulate.py**

**from** random **import** random  
**from** collections **import** defaultdict, namedtuple  
  
UnitsSettings = namedtuple(**'UnitsSettings'**, [**'pi1'**, **'pi2'**])  
  
  
**def** simulate(transitions, units\_settings, tacts\_count, start\_state, characteristics):  
 result = defaultdict(int)  
 pi1, pi2 = units\_settings.pi1, units\_settings.pi2  
  
 current\_state = start\_state  
 **for** \_ **in** range(tacts\_count):  
 pi1\_state = random() <= pi1  
 pi2\_state = random() <= pi2  
  
 result[current\_state] += 1  
  
 **for** characteristic **in** characteristics:  
 characteristic.update(current\_state, pi1\_state, pi2\_state)  
  
 current\_state = next((x **for** x **in** transitions **if** x[0] == current\_state **and** x[2](pi1\_state, pi2\_state)))[1]  
  
 **return** result, {x.name: x.result **for** x **in** characteristics}

// transitions.py

TRANSITIONS = [  
 [**'2000'**, **'1000'**, **lambda** pi1, pi2: 1],  
 [**'1000'**, **'2100'**, **lambda** pi1, pi2: 1],  
  
 [**'2100'**, **'1100'**, **lambda** pi1, pi2: pi1],  
 [**'2100'**, **'1001'**, **lambda** pi1, pi2: **not** pi1],  
  
 [**'1100'**, **'2100'**, **lambda** pi1, pi2: pi1],  
 [**'1100'**, **'2101'**, **lambda** pi1, pi2: **not** pi1],  
  
 [**'1001'**, **'2100'**, **lambda** pi1, pi2: **not** pi2],  
 [**'1001'**, **'2101'**, **lambda** pi1, pi2: pi2],  
  
 [**'2101'**, **'1100'**, **lambda** pi1, pi2: pi1 **and** (**not** pi2)],  
 [**'2101'**, **'1001'**, **lambda** pi1, pi2: (**not** pi1) **and** (**not** pi2)],  
 [**'2101'**, **'1101'**, **lambda** pi1, pi2: pi1 **and** pi2],  
 [**'2101'**, **'1011'**, **lambda** pi1, pi2: (**not** pi1) **and** pi2],  
  
 [**'1101'**, **'2101'**, **lambda** pi1, pi2: (pi1 **and** (**not** pi2)) **or** ((**not** pi1) **and** (**not** pi2)) **or** (pi1 **and** pi2)],  
 [**'1101'**, **'2111'**, **lambda** pi1, pi2: (**not** pi1) **and** pi2],  
  
 [**'1011'**, **'2101'**, **lambda** pi1, pi2: **not** pi2],  
 [**'1011'**, **'2111'**, **lambda** pi1, pi2: pi2],  
  
 [**'2111'**, **'1101'**, **lambda** pi1, pi2: pi1 **and** (**not** pi2)],  
 [**'2111'**, **'1011'**, **lambda** pi1, pi2: ((**not** pi1) **and** (**not** pi2)) **or** ((**not** pi1) **and** pi2)],  
 [**'2111'**, **'1111'**, **lambda** pi1, pi2: pi1 **and** pi2],  
  
 [**'1111'**, **'2111'**, **lambda** pi1, pi2: ((**not** pi1) **and** pi2) **or** ((**not** pi1) **and** (**not** pi2)) **or** (pi1 **and** pi2)],  
 [**'1111'**, **'2101'**, **lambda** pi1, pi2: pi1 **and** (**not** pi2)],  
]