**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Теория надежности**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: доц. каф. ПО и ВТАС

Кабалянц Петр Степанович

**Белгород 2020**

**Задание для выполнения к работе**

На сервере есть n каналов передачи сообщений. Среднее время обработки сообщений τ. На сервер поступают сообщения в среднем количестве λ сообщений в минуту. Для хранения сообщений в очереди на сервере выделено место для (n+1) сообщения. Определить основные характеристики сервера:

1. вероятность очереди;

2.среднее число занятых каналов;

3. среднюю длину очереди;

4. среднее число сообщений на сервере.

Предполагается, что сообщение не получает отказ при занятости всех каналов и очереди длины не больше (n+1). Параметры варианта определяются по формулам:

n = 3 + [(i + j) / 8], λ = 1 + i / 4, τ = 5 / (5 + j). Здесь квадратные скобки означают взятие целой части, а i, j - последние цифры зачетки.

**Ход выполнения работы**

m = 5 - допустимый размер очереди

n = 3 + [(10 + 1) / 8] = 4 канала

τ = 5 / (5 + 1) = 0,83 c - среднее время обработки сообщения

λ = 1 + 10 / 4 = 3,5 - среднее кол-во сообщений в минуту

= 1 / τ = 1 / 0,83 = 1,2 - интенсивность потока обслуживания

= λ / = 3,5 / 1,2 = 2,92 - приведенная интенсивность потока

Вычислим предельные вероятности:

Посчитаем требуемые характеристики:

- вероятность отказа в обработке

*-* пропускная способность

*-* абсолютная пропускная способность

*-* среднее число занятых каналов

*-* вероятность образования очереди

*-* средняя длина очереди

- среднее время ожидания в очереди

Изображение выглядит как снимок экрана, ноутбук, компьютер, монитор

Автоматически созданное описание

*Приложение*

import random

import math

import numpy

from scipy.special import factorial as fact

i = 10

j = 1

n = 3 + ((i + j) // 8) # Количество каналов передачи сообщений

m = n + 1 # Размер очереди

lambd = 1 + i / 4 # Интенсивность потока заявок

tau = 5 / (5 + j) # Среднее время обработки сообщений

print("Количество каналов передачи n = {}\nРазмер очереди m = {}\nСообщений в минуту λ = {}\nСреднее время обработки сообщений τ = {}".format(n, m, lambd, tau))

mu = 1 / tau # Интенсивность потока обслуживания

ro = lambd / mu # Приведенная интенсивность потока заявок (интенсивность нагрузки)

print("Интенсивность потока обслуживания μ = {}\nПриведенная интенсивность потока ρ = {}\n".format(mu, ro))

P = [0] # Предельные вероятности (среднее относительное время, которое канал занят (p0 - все свободны))

for i in range (0, n + 1):

P[0] += (ro \*\* i) / fact(i)

P[0] += (ro \*\* (n + 1)) \* (1 - (ro / n) \*\* m) / (fact(n) \* (n - ro))

P[0] = P[0] \*\* -1

for i in range (1, n + 1):

P.append((ro \*\* i) / fact(i) \* P[0])

for r in range(1, m + 1):

P.append((ro \*\* (n + r)) / (n \*\* r \* fact(n)) \* P[0])

print("Предельные вероятности P = {}\n".format(P))

P\_o = P[n + m] # Вероятность отказа (все каналы заняты)

Q = 1 - P\_o # Относительная пропускная способность

A = lambd \* Q # Абсолютная пропускная способность

k = ro \* (1 - ((ro \*\* (n + m))/(n \*\* m \* fact(n))) \* P[0] ) # Среднее число занятых каналов

# Средняя длина очереди

L\_och = (ro \*\* (n + 1) \* P[0] \* (1 - (m + 1 - m \* ro/n) \* ((ro/n) \*\* m))) / (n \* fact(n) \* ((1 - ro/n) \*\* 2))

T\_och = L\_och/lambd

P\_och = 0

for i in range(n, n+m):

P\_och += P[i]

print("Теоретические значения:")

print("Относительная пропускная способность Q = ", Q)

print("Абсолютная пропускная способность A = ", A)

print("Вероятность отказа в обработке P\_отк = ", P\_o)

print("Среднее число занятых каналов ~k = ", k)

print("Вероятность образования очереди: ", P\_och)

print("Среднея длина очереди L\_оч = ", L\_och)

print("Среднее время ожидания в очереди: {}\n".format(T\_och))

# Добавить сообщение в очередь

def pushMessage(message, queue):

if (len(queue) < m):

queue.append([message, 0])

return True

return False

# Наращивает время ожидания в очереди

def incWaitTime(queue, dt):

for i in range(0, len(queue)):

#if (queue[i][0] > 0):

queue[i][1] += dt

# Проверяет каналы на наличие свободных и помещает в них сообщения из очереди

def queueToChannel(channels, queue, averageQueueTime, queueCount):

pos = freeChannel(channels)

while (pos != -1 and len(queue) > 0):

channels[pos] = queue[0][0]

averageQueueTime[0] += queue[0][1]

queueCount[0] += 1

queue.pop(0)

pos = freeChannel(channels)

# Возвращает номер свободного канала, иначе - -1

def freeChannel(channels):

for i in range(0, n):

if channels[i] == 0:

return i

return -1

# Обработка уже имеющихся сообщений

def messagesProcessing(channels, dt, queue, averageQueueTime, queueCount):

for i in range (0, len(channels)):

if (channels[i] > dt):

channels[i] -= dt

else:

channels[i] = 0

incWaitTime(queue, dt)

queueToChannel(channels, queue, averageQueueTime, queueCount)

# Получение нового сообщения

def newMessage(message, channels, queue):

pos = freeChannel(channels)

if (pos != -1):

channels[pos] = message

return True

if (pushMessage(message, queue)):

return True

return False

channels = [0 for i in range (0, n)] # Каналы связи (0, если не обрабатывается, иначе - оставшееся время)

queue = []

maxTime = 1000 # Время работы

busyChannels = 0 # Занятые каналы

unProcessedMessages = 0 # Необработанные сообщения

totalMessages = 0 # Всего сообщений

dt = 0.01 # Δt

averageTime = 0 # Среднее время обработки сообщения

averageQueueTime = [0] # Среднее время ожидания в очереди

queueCount = [0] # Общее количество сообщений в очереди

averageQueueCount = 0 # Среднее количество сообщений в очереди

for currentTime in range (0, int(maxTime / dt)):

if (random.random() < 1 - math.exp(-1 \* lambd \* dt)): # Если сообщение пришло

message = tau # Назначаем ему время обработки

averageTime += message

totalMessages += 1

if(newMessage(message, channels, queue) != True): # Отправляем сообщение в свободный канал или очередь

unProcessedMessages += 1 # Если все каналы заняты, сообщение не обработано

averageQueueCount += len(queue)

busyChannels += n - channels.count(0)

messagesProcessing(channels, dt, queue, averageQueueTime, queueCount) # Обрабатываем сообщение

busyChannels = busyChannels / (maxTime / dt)

averageTime = averageTime / totalMessages

if (queueCount[0] > 0):

averageQueueTime[0] = averageQueueTime[0] / queueCount[0]

averageQueueCount = averageQueueCount / int(maxTime / dt)

print("Результаты эксперимента ({} минут, Δt = {}):".format(maxTime, dt))

print("Всего сообщений: {}, отказов: {}, попало в очередь: {}".format(totalMessages, unProcessedMessages, queueCount[0]))

print("Среднее время обработки сообщения: ", averageTime)

print("\nОтносительная пропускная способность Q = ", (totalMessages - unProcessedMessages) / totalMessages)

print("Абсолютная пропускная способность А = ", (totalMessages - unProcessedMessages) / maxTime)

print("Вероятность отказа при обработке P\_отк = ", unProcessedMessages / totalMessages)

print("Среднее число занятых каналов при этом составило ~k = ", busyChannels)

print("Вероятность образования очереди: ", queueCount[0]/totalMessages)

print("Среднее количество сообщений в очереди: ", averageQueueCount)

print("Среднее время ожидания в очереди: ", averageQueueTime[0])