Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

О Т Ч Е Т

по лабораторной работе № 4

Название: «Обслуживающий аппарат»

Дисциплина: Моделирование

Студент ИУ7-76Б (Группа) А. А. Петрова (И.О. Фамилия)

Преподаватель И. В. Рудаков (И.О. Фамилия)

2022 г.

Задание

Промоделировать систему принципом «Δt» и событийным, состоящую из источника информации, буферной памяти и обслуживающего аппарата. Источник информации подает сообщения по равномерному закону распределения, а на обработку сообщения выбираются по нормальному закону распределения (10 вариант). С заданной вероятностью часть обработанных сообщений снова поступает в очередь (буферную память). Определить максимальный размер буферной памяти, при котором не будет потерь сообщений.

Математическая формализация

На рисунке ниже представлена схема моделируемой системы.

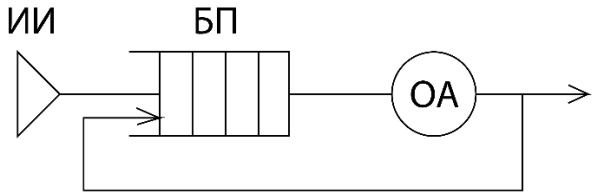

Рисунок 1: моделируемая система

Где ИИ – это источник информации, БП – буферная память, а ОА – обслуживающий аппарат.

Принцип «Δt»

Этот принцип заключается в последовательном анализе состояний всех блоков в момент времени $t + \Delta t$ по заданному состоянию блоков в момент времени t. При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов. В результате этого анализа принимается решение о том, какие общесистемные события должны имитироваться программой на данный момент времени.

Основной недостаток этого принципа – это значительные затраты вычислительных ресурсов, а при недостаточно малом Δt появляется опасность пропуска отдельных событий в системе, исключающее возможность получения правильных результатов при моделировании.

Событийный принцип

Характерное свойство моделируемых систем обработки информации – это то, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщений в систему, окончания реализации задания (процесса), возникновения прерываний и аварийных сигналов. Следовательно, моделирование и продвижение текущего времени в системе удобно проводить, используя событийный принцип.

При использовании данного принципа состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент поступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояний каждого из блоков системы.

Реализация

В листингах ниже представлена реализация принципа «Δt» и событийного принципа.

Листинг 1: принцип «Δt»

```
def time_based_modelling(self, request_count, dt):
    generator = self.generator
    processor = self.processor

    gen_period = generator.get_time()
    proc_period = gen_period
    current_time = 0
    while processor.processed_requests < request_count:
        if gen_period <= current_time:
            generator.request()
            gen_period += generator.get_time()
        if current_time >= proc_period:
            processor.process()
            if processor.queue_size > 0:
                proc_period += processor.get_time()
            else:
                proc_period = gen_period + processor.get_time()

        current_time += dt

    return (processor.processed_requests, processor.reentered_requests,
            processor.max_queue_size, round(current_time, 3))
```

Листинг 2: событийный принцип

```
def event_based_system(self, request_count):
    generator = self.generator
    processor = self.processor

    gen_period = generator.get_time()
```

```
    proc_period = gen_period + processor.get_time()

    while processor.processed_requests < request_count:
        if gen_period <= proc_period:
            generator.request()
            gen_period += generator.get_time()
        if gen_period >= proc_period:
            processor.process()
            if processor.queue_size > 0:
                proc_period += processor.get_time()
            else:
                proc_period = gen_period + processor.get_time()

    return (processor.processed_requests, processor.reentered_requests,
            processor.max_queue_size, round(proc_period, 3))
```

Результаты работы

Для всех тестов использовались следующие параметры распределений:

- равномерное: a = 0, b = 10;
- нормальное: m = 0, σ = 1.

Количество запросов: 10000.

Δt = 0.01.

Ниже представлены результаты работы программы в зависимости от вероятности (P) повторного попадания заявки в очередь.

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	0	4	49304.47
Дельта t	10000	0	4	50256.46

Рисунок 2: работа программы при P = 0%

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	2464	4	37630.498
Дельта t	10000	2500	4	38104.25

Рисунок 3: работа программы при P = 25%

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	4991	5	25011.606
Дельта t	10000	5000	4	24822.38

Рисунок 4: работа программы при P = 50%

Метод	Кол-во обработанных запросов	Кол-во возвращенных запросов	Максимальный размер очереди	Время работы
Событийный	10000	7500	4	12475.961
Дельта t	10000	7500	4	12604.04

Рисунок 5: работа программы при P = 75%