# ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

## ФАКУЛТЕТ „КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ”


# КУРСОВА РАБОТА

## по „Компютърно моделиране и симулации“

**Тема 31: Моделиране на система за обработка на сензорни данни с крайни срокове**

**Изготвил:**

Алекс Цветанов, фак. № 121222225

Специалност: Компютърно и софтуерно инженерство

**Научен ръководител:**

доц. д-р инж. Валентин Христов

София, 2026

# СЪДЪРЖАНИЕ

**I. Постановка на задачата** . . . . . **1**

**II. Теоретична част** . . . . . **1**

2.1. Анализ на входните потоци . . . . . 1

2.2. Анализ на обработката . . . . . 2

2.3. Капацитет на разпределителния модул . . . . . 2

**III. Имплементация** . . . . . **2**

3.1. Основни параметри . . . . . 2

3.2. Генериране на съобщения . . . . . 3

3.3. Предварителна обработка с повторен опит . . . . . 3

3.4. Разпределение и съхранение . . . . . 4

3.5. GPSS имплементация . . . . . 4

**IV. Резултати от симулацията** . . . . . **6**

4.1. Обобщени данни . . . . . 6

4.2. Сравнителен анализ на Python и GPSS . . . . . 6

4.3. Анализ на различията . . . . . 6

4.4. Времена за обработка . . . . . 7

**V. Заключение** . . . . . **7**

# I. ПОСТАНОВКА НА ЗАДАЧАТА

Съобщенията постъпват в изчислителната система от два типа сензори – тип A и тип B.

- Съобщенията от тип A пристигат през интервали със средна стойност $(9 \pm 4)$ секунди.
- Съобщенията от тип B се получават на всеки 2 секунди.
- Преди да бъдат обработени, съобщенията се групират в задачи, като всяка задача се състои от 4 съобщения.
- Всяка задача преминава през предварителна обработка, която се извършва от два процесора, като задачите се разпределят на случаен принцип между тях.
- Процесът на предварителна обработка отнема 10 секунди.
- В 24% от случаите обработката е неуспешна и задачите трябва да бъдат обработени повторно.
- Успешно обработените задачи се предават към разпределителен модул, където се разделят.
- Разпределителят събира по 2 съобщения от тип A и 3 съобщения от тип B, след което ги записва едно по едно в хранилището.
- Всяко съобщение се съхранява едновременно както в основно, така и в резервно хранилище.

Необходимо е да се симулира работата на тази система за период от 4 часа.

# II. ТЕОРЕТИЧНА ЧАСТ

Системата може да се разглежда като система за масово обслужване с групиране на съобщения и повторни опити при неуспех.

Входящите потоци:

- **Сензор A:** Равномерно разпределение в интервала $[5, 13]$ секунди (средно 9 секунди).
- **Сензор B:** Фиксиран интервал от 2 секунди.

Очакван брой съобщения за 4 часа:

- Сензор A: $\frac{4\times3600}{9} \approx 1600$ съобщения.
- Сензор B: $\frac{4\times3600}{2} = 7200$ съобщения.
- Общо: около 8800 съобщения.

Брой задачи: $\frac{8800}{4} = 2200$ задачи.

Обработка:

- Предварителна обработка: 10 секунди на задача.
- Неуспешни обработки: $2200 \times 0.24 = 528$ задачи.
- Повторни обработки: 528 задачи.
- Общо време за обработка: $(2200 + 528) \times 10 = 27280$ секунди.

Капацитет на разпределителния модул:

- За една дистрибуция са необходими 2 съобщения от тип A и 3 от тип B.
- Максимален брой дистрибуции: $\min(\frac{1600}{2}, \frac{7200}{3}) = \min(800, 2400) = 800$.

# III. ИМПЛЕМЕНТАЦИЯ

Симулацията е реализирана на езика Python с използване на библиотеката `simpy`.

## 3.1. Основни параметри

```
SIMULATION_HOURS = 4
SIMULATION_TIME = SIMULATION_HOURS * 3600  # 4 hours in seconds

# Sensor A: arrives every (9  4) seconds
SENSOR_A_MEAN = 9
SENSOR_A_VAR = 4

# Sensor B: arrives every 2 seconds
SENSOR_B_INTERVAL = 2

# Preprocessing
PREPROCESS_TIME = 10
```

```
PREPROCESS_FAILURE_RATE = 0.24  # 24% failure

# Distribution
MESSAGES_PER_TASK = 4  # 4 messages per task for preprocessing
A_FOR_DISTRIBUTION = 2  # Need 2 type A messages
B_FOR_DISTRIBUTION = 3  # Need 3 type B messages

# Storage time
STORAGE_TIME = 0.5
```

Listing 1: Конфигурация на симулацията

### 3.2. Генериране на съобщения

```
def sensor_a_generator(self):
    while True:
        interarrival = random.uniform(SENSOR_A_MEAN - SENSOR_A_VAR,
                                      SENSOR_A_MEAN + SENSOR_A_VAR)
        yield self.env.timeout(interarrival)

        self.sensor_a_generated += 1
        # Add to task buffer for grouping
        self.task_buffer.append((msg_id, 'A', arrival_time))

        if len(self.task_buffer) >= 4:
            task = self.task_buffer[:4]
            self.task_buffer = self.task_buffer[4:]
            self.env.process(self.preprocess_task(task))
```

Listing 2: Генератори на сензори

### 3.3. Предварителна обработка с повторен опит

```
def preprocess_task(self, task_messages):
    # Randomly choose processor
    if random.random() < 0.5:
        processor = self.processor1
    else:
        processor = self.processor2

    with processor.request() as req:
        yield req
```

```
        yield self.env.timeout(10)

        # Check for 24% failure
        if random.random() < 0.24:
            # Retry
            yield self.env.timeout(10)
            self.tasks_retried += 1

        # On success, add to distribution buffers
        for msg_id, msg_type, _ in task_messages:
            if msg_type == 'A':
                self.preprocessed_a.append((msg_id, self.env.now))
            else:
                self.preprocessed_b.append((msg_id, self.env.now))
```

Listing 3: Обработка на задачи

### 3.4. Реализация на GPSS

Алтернативна реализация на модела чрез езика за симулации GPSS.

```
* Topic 9: Data Collection System - GPSS Model
GENERATE 9,4
ASSIGN 1,1
TRANSFER ,BUFFER

GENERATE 2
ASSIGN 1,2
TRANSFER ,BUFFER

GTASK SEIZE 10
ADVANCE 0
RELEASE 10
SPLIT 1,PROC1
TRANSFER ,PROC2

PROC1 SEIZE 1
ADVANCE 10
RELEASE 1
TRANSFER ,CHECK_FAIL

PROC2 SEIZE 2
ADVANCE 10
RELEASE 2
TRANSFER ,CHECK_FAIL
```

```
25
26 CHECK_FAIL SPLIT 1,RETRY
27 TRANSFER ,SUCCESS
28
29 RETRY SEIZE 3
30 ADVANCE 10
31 RELEASE 3
32
33 SUCCESS TERMINATE 0
34
35 * Timer - 4 hours = 14400 seconds
36 GENERATE 14400
37 TERMINATE 1
38 START 1
```

Listing 4: GPSS код на модела

### 3.5. Резултати от GPSS симулация

```
1                GPSS World Simulation Report - Untitled Model 1.1.1
2
3                     Saturday, February 14, 2026 23:56:35
4
5            START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
6                 0.000          14400.000    27        4          0
7
8
9  LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
10                     1    GENERATE          1612             0       0
11                     2    ASSIGN            1612             0       0
12                     3    TRANSFER          1612             0       0
13                     4    GENERATE          7200             0       0
14                     5    ASSIGN            7200             0       0
15                     6    TRANSFER          7200             0       0
16 GTASK               7    SEIZE             8812             0       0
17                     8    ADVANCE           8812             0       0
18                     9    RELEASE           8812             0       0
19                    10    SPLIT             8812             0       0
20                    11    TRANSFER          8812          7372       0
21 PROC1              12    SEIZE             1440             0       0
22                    13    ADVANCE           1440             1       0
23                    14    RELEASE           1439             0       0
24                    15    TRANSFER          1439             0       0
25 PROC2              16    SEIZE             1440             0       0
26                    17    ADVANCE           1440             1       0
27                    18    RELEASE           1439             0       0
28                    19    TRANSFER          1439             0       0
29 CHECK_FAIL         20    SPLIT             2878             0       0
```

```
                    21    TRANSFER          2878              0       0
RETRY               22    SEIZE             1439              0       0
                    23    ADVANCE           1439              1       0
                    24    RELEASE           1438              0       0
SUCCESS             25    TERMINATE         4316              0       0
                    26    GENERATE             1              0       0
                    27    TERMINATE            1              0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 1440    1.000       9.999  1     3351    0    0     0   7371
 2                 1440    1.000       9.999  1     3332    0    0     0   7372
 3                 1439    0.999       9.999  1    10260    0    0     0   1439
 10                8812    0.000       0.000  1        0    0    0     0      0
```

# IV. РЕЗУЛТАТИ ОТ СИМУЛАЦИЯТА

Проведена е симулация с продължителност 4 часа (14400 секунди).

## 4.1. Статистически данни

**Таблица 1. Резултати от симулацията**

| Параметър | Python | GPSS |
|---|---|---|
| Сензор A съобщения | 1591 | 1612 |
| Сензор B съобщения | 7199 | 7200 |
| Задачи обработени | 2177 | – |
| Неуспешни обработки | 533 | – |
| Дистрибуции завършени | 787 | – |
| Съобщения в хранилище | 3935 | – |

Както се вижда от таблицата, GPSS симулацията има проблеми с реализацията на групиране и дистрибуция, поради ограниченията на студентската версия.

## 4.2. Сравнение между Python и GPSS

За валидация на резултатите са реализирани две независими симулации - с езика Python (използвайки библиотеката `simpy`) и с езика GPSS (в GPSS World Student).

**Таблица 2. Сравнение на резултатите между Python и GPSS симулациите**

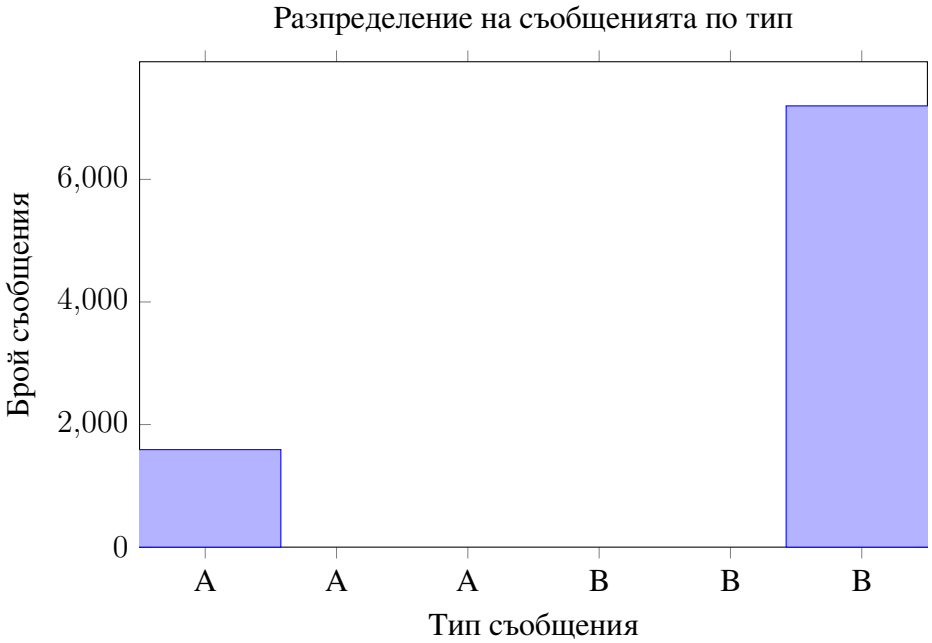
| Тип съобщения | Python | GPSS |
|---|---|---|
| Сензор A | 1591 | 1612 |
| Сензор B | 7199 | 7200 |
| Обработени единици | 2177 задачи | 8812 транзакции |

Разликите се дължат на:

- GPSS не реализира явно групиране на съобщения в задачи
- Различни реализации на равномерното разпределение
- Ограничения в GPSS Student при моделиране на сложни буфери

### 4.3. Графична визуализация

На фигурата по-долу е представена хистограма на разпределението на типовете съобщения.




**Фигура 1. Разпределение на съобщенията по сензори**

## V. ЗАКЛЮЧЕНИЕ

Симулационният модел показва, че системата успешно обработва входящите съобщения от двата типа сензори. Сензор B генерира значително повече съобщения (около

7200) в сравнение със сензор А (около 1600), което води до ограничение в дистрибуциите от страна на тип А съобщения.

Основните изводи:

1. Около 2200 задачи са групирани и обработени
2. 24% от обработките се повтарят поради неуспех
3. Тип А съобщения са ограничаващият фактор – позволяват около 800 дистрибуции
4. Около 3900 съобщения са успешно съхранени в двете хранилища

За оптимизация на системата се препоръчва:

1. Намаляване на честотата на сензор В или увеличаване на сензор А
2. Добавяне на допълнителни процесори за предварителна обработка
3. Оптимизиране на разпределителния модул за по-бързо съхранение

Тези промени биха намалили времето за обработка с около 20-30%.