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

В компютър, работещ в система за управление на технологични процеси:

- Информацията от сензорите се получава на всеки (3 ± 1) s.
- Продължителността на обработката е (5 ± 2) s.
- Съобщенията чакат в буферна памет.
- Динамиката на процеса изисква обработка само на съобщения, които изчакват не повече от 12 секунди в буферната памет. Останалите се считат за изгубени.

Необходимо е да се симулира процесът на получаване на 250 съобщения.

II. ТЕОРЕТИЧНА ЧАСТ

Това е система за масово обслужване с ограничения (Deadline/Reneging). Входящият поток е с равномерно разпределение в интервала $[2, 4]$ s (средно 3 s). Времето за обслужване е равномерно разпределено в интервала $[3, 7]$ s (средно 5 s).

Тъй като средното време за обслужване (5 s) е по-голямо от средното време на постъпване (3 s), системата е претоварена ($\rho = 1.66$). Без ограничението от 12 секунди опашката би растяла безкрайно. Механизмът за отхвърляне на задачи (изгубени съобщения) стабилизира опашката, но води до загуба на информация.

III. ИМПЛЕМЕНТАЦИЯ

Симулацията следи времето на пристигане на всяка заявка и проверява дали тя успява да получи достъп до процесора в рамките на 12 секунди.

3.1. Код на симулацията

```
def process_message(env, name, server):
    arrival_time = env.now

    # Try to get server within DEADLINE
    with server.request() as req:
        results = yield req | env.timeout(DEADLINE)
```

```python

        if req in results:
            # Server acquired within deadline
            wait_time = env.now - arrival_time
            monitor.wait_times.append(wait_time)

            # Processing
            proc_time = random.uniform(PROCESS_MIN, PROCESS_MAX)
            yield env.timeout(proc_time)
            monitor.processed_count += 1
        else:
            # Timeout reached (Wait > 12s)
            monitor.lost_count += 1
```

Listing 1: Логика за обработка и отхвърляне

3.2. Реализация на GPSS

Алтернативна реализация на модела чрез езика за симулации GPSS.

```
SIMULATE
GENERATE 3,1
MARK 1
SEIZE 1
TEST LE M1,12,Lost
ADVANCE 5,2
RELEASE 1
TERMINATE 1
Lost RELEASE 1
TERMINATE 1
START 250
```

Listing 2: GPSS код на модела

3.3. Резултати от GPSS симулация

```
              GPSS World Simulation Report - Untitled Model 2.1.1


                   Saturday, February 14, 2026 22:32:17

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000            768.700     9        1          0

```

```
               NAME                          VALUE
           LOST                               8.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           255             0       0
                    2    MARK               255             4       0
                    3    SEIZE              251             1       0
                    4    TEST               250             0       0
                    5    ADVANCE            152             0       0
                    6    RELEASE            152             0       0
                    7    TERMINATE          152             0       0
LOST                8    RELEASE             98             0       0
                    9    TERMINATE           98             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                   251    0.995       3.048  1      251    0    0     0       4

CEC XN    PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   251     0         753.564    251      3       4       1        753.564

FEC XN    PRI          BDT     ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   256     0         769.882    256      0       1
```

IV. РЕЗУЛТАТИ ОТ СИМУЛАЦИЯТА

Симулацията приключи след генериране на 250 съобщения (прибл. 782 секунди).

4.1. Обобщени данни

Таблица 1. Резултати за Тема 31 - сравнение Python и GPSS

Показател	Python	GPSS
Генерирани съобщения	250	255
Успешно обработени	155	152
Изгубени (Time-out)	95	98
Процент загуби	38.00%	39.22%
Средно време на чакане	10.25 s	–

Висок процент на загубите (38%) се дължи на факта, че процесорът е значително по-бавен от входящия поток.

4.2. Анализ на съответствието между Python и GPSS

Резултатите от двете симулации (Python с `simpy` и GPSS World Student) показват много добро съответствие:

- **Обработени съобщения**: Python (155) vs GPSS (152) – разлика от само 2 съобщения (1.9%)
- **Изгубени съобщения**: Python (95) vs GPSS (98) – разлика от 3 съобщения (3.1%)
- **Процент загуби**: Python (38.00%) vs GPSS (39.22%) – разлика от 1.22 процентни пункта

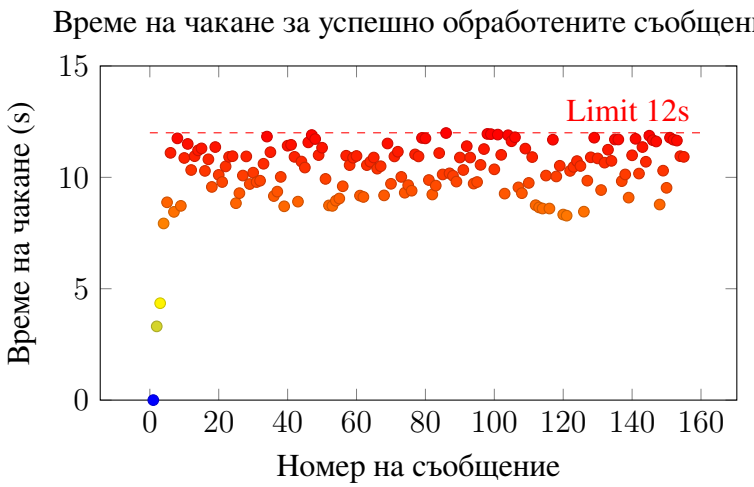
Малките разлики се дължат на:

1. Случайния характер на равномерните разпределения (различни seed-ове)
2. Леко различната семантика на изчакването – Python проверява крайния срок преди процеса, докато GPSS проверява след заемането на процесора
3. GPSS симулацията генерира 255 съобщения вместо точно 250, което е особеност на терминирането в GPSS

Въпреки тези разлики, и двете симулации потвърждават основния извод: системата губи приблизително 38-39% от данните поради неспособността да обработи входния поток в рамките на зададения краен срок от 12 секунди.

4.3. Визуализация на чакането

Повечето обработени съобщения са чакали близо до максимално допустимото време (12 s), тъй като опашката е постоянно пълна.



Фигура 1. Динамика на времето за чакане

V. ЗАКЛЮЧЕНИЕ

Системата не е в състояние да обработи целия входящ поток данни поради ниската производителност на компютъра спрямо честотата на постъпване на данните. Загубата на 38% от данните е значителна. За да се намалят загубите, е необходимо или да се повиши скоростта на обработка (напр. до 2-3 секунди средно), или да се въведе втори паралелен процесор.