MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Artem Volokyta

Software Security

Lab Work 6

Queue model streams and drone swarms

kulubecioglu Mehmet

Class Number 12

IM-14 FIOT

Kyiv

IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE

2024

1. Introduction

This project aims to simulate a queue management system using multiple drones. Drones are modeled as devices processing incoming data packets, each with a specific processing capacity. The main goal of the system is to distribute incoming packets evenly among drones, process them within the queues, and analyze how queue sizes change over time.

2. Simulation Overview

The simulation is implemented in Python and consists of the following components:

Code Workflow:

The code models a total of 5 drones using a for loop and various data structures.

Incoming data packets are distributed among drones at a rate controlled by the R variable.

At the end of each iteration, drone queues are updated, and packets exceeding their maximum waiting time are removed.

Core Parameters:

n: Number of drones (5).

queue_max_size: Maximum queue capacity per drone (20).

R: Total number of packets entering the system per second (100).

max_lifetime: Maximum time a packet can wait in the queue (5 iterations).

iterations: Total number of simulation steps (100).

Main Sections of the Code:

Initialization: Drone queues and other variables are defined.

Packet Distribution: Incoming packets are assigned to drones with the shortest queues.

Processing: Drones process the packets in their queues to clear space. Packets exceeding their maximum waiting time are removed.

Visualization: The queue sizes are plotted using the Matplotlib library.

3. Code Details

1. **Defining the Initial State:**

```
n = 5
max_efficiency = 20
queue_max_size = 20
R = 100
iterations = 100
max_lifetime = 5
```

This section defines the general parameters of the system.

2. **Drone Processing Speeds:**

```
s = np.array([10, 15, 20, 12, 8])
```

3. Updating Queues:

```python
packet_lifetimes = [[] for _ in range(n)]  # Paketlerin yaşlarını takip eden listeler

# Paket dağıtımı ve kuyruk yönetimi fonksiyonu
def distribute_tasks_with_queues(s, R, queues, packet_lifetimes, max_lifetime):
    n = len(s)
    p = np.zeros(n)
    remaining_R = R

    # Kuyruklarda paket varsa, önce bunlar işlenir
    for i in range(n):
        if queues[i] > 0:
            to_process = min(queues[i], s[i])
            p[i] += to_process
            queues[i] -= to_process
            remaining_R -= to_process

    # Kalan paketler, uygun drone'lara dağıtılır
    while remaining_R > 0:
        for i in range(n):
            if remaining_R == 0:
                break
            if queues[i] < queue_max_size:
                queues[i] += 1
                remaining_R -= 1

    # Paket yaşlarını güncelle ve eski paketleri çıkar
    for i in range(n):
        packet_lifetimes[i] = [age + 1 for age in packet_lifetimes[i]] + [0] * int(queues[i] - len(packet_lifetimes[i]))
        packet_lifetimes[i] = [age for age in packet_lifetimes[i] if age <= max_lifetime]
        queues[i] = len(packet_lifetimes[i])

    return p, queues, packet_lifetimes
```

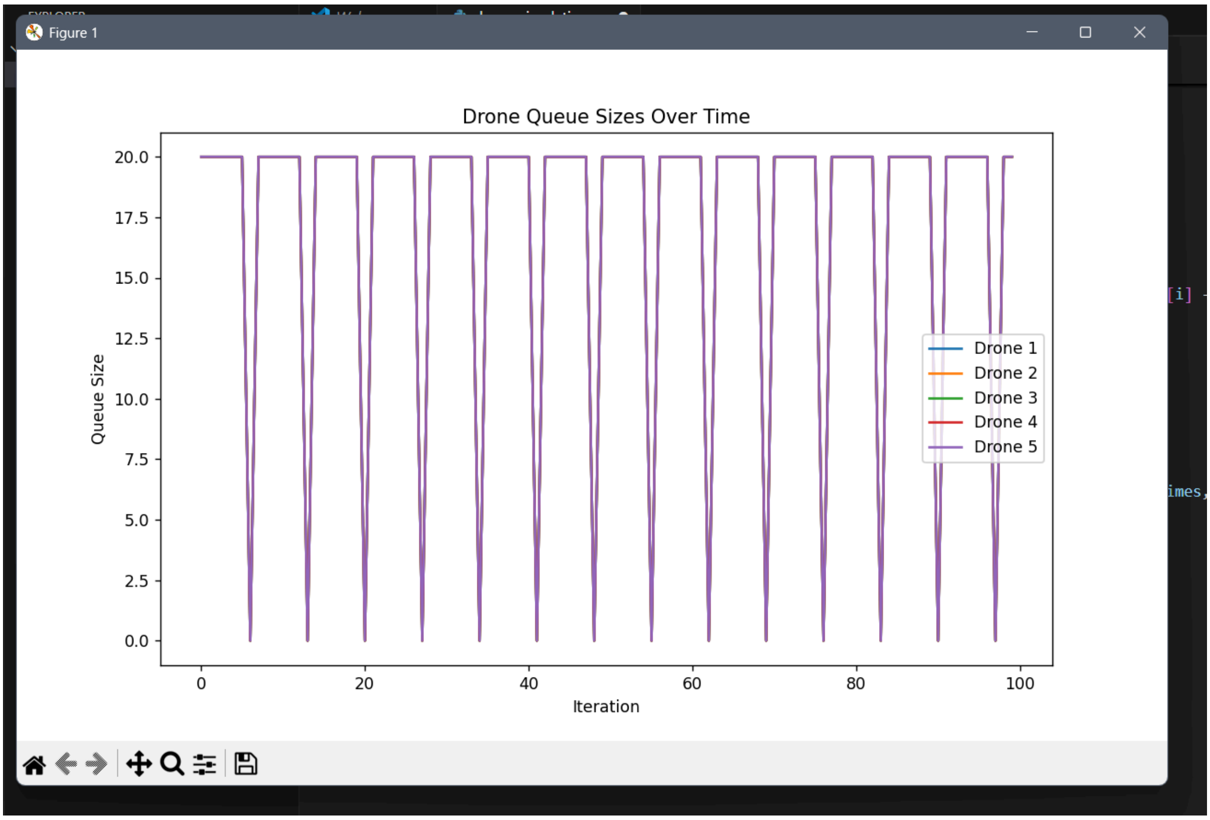
4. Simulation:

```python
queue_sizes_over_time = np.zeros((iterations, n))
for t in range(iterations):
    _, queues, packet_lifetimes = distribute_tasks_with_queues(s, R, queues, packet_lifetimes, max_lifetime)
    queue_sizes_over_time[t] = queues
```

5. Plotting the Graph:

```python
plt.figure(figsize=(10, 6))
for i in range(n):
    plt.plot(range(iterations), queue_sizes_over_time[:, i], label=f'Drone {i + 1}')
plt.xlabel('Iteration')
plt.ylabel('Queue Size')
plt.title('Drone Queue Sizes Over Time')
plt.legend()
plt.show()
```

6. Results and Observations



The following observations were made based on the simulation:

- Drone queues frequently reach their maximum capacity before clearing and refilling.
- The stability of the system is directly affected by drone processing speeds and the rate of incoming packets.