Exercise

Authors:

Abate Kevin: **10812892**

Pigato Lorenzo: **10766953** [Team Leader]

EQ2

Consider the following pseudocode for a ESP32-based IoT monitoring system

```
// Global Timer Handle
declare timer_handle as esp_timer_handle_t

// Initialization
function setup_camera():
    initialize_camera(QVGA)

function app_main():
    call setup_camera()
    call setup_timer()
    loop forever:
        delay(100 ms)

function setup_timer():
    declare timer_config as esp_timer_create_args_t
    set timer_config.callback to process_frame
    set timer_config.name to "10_sec_timer"
    call esp_timer_create(&timer_config,
    &timer_handle)
    call esp_timer_start_periodic(timer_handle,10_000_000) // 10s

// Called every 10 seconds
function process_frame(arg):
    image = capture_camera_frame()
    person_count = estimate_number_of_people(image)
    if person_count == 0:
        payload = create_message(size=1KB)
    else if person_count == 1:
        payload = create_message(size=3KB)
    else:
        payload = create_message(size=6KB)
```

Assuptions

- The system is operated with IEEE 802.15.4 in beacon-enabled mode (CFP only)
- The number of people present in the camera frame at any instant follows a Poisson distribution with an average rate of lambda = 0.15 persons/frame

Libraries

In [ ]:
```
import math
```

First question

> Compute the Probability Mass Function of the output rate of the ESP32 P(r = r_0), P(r = r_1), P(r = r_2), where r_0, r_1 and r_2 are the output rates when there are 0, 1 or more than 1 persons in the captured frame, respectively.

In [41]:
```
Lambda = 0.15 # person/frame
time_period = 10 #second

message_size_0_person = 1 #KB

message_size_1_person = 3 #KB

message_size_over_2_person = 6 #KB
```

In [42]:
```
r0 = 1 / time_period  #KB/s
r1 = 3 / time_period  #KB/s
r2 = 6 / time_period  #KB/s
```

Calculate the **Poisson Probabilities**

In [43]:
```
results=[]
for x in range(2):
  p_x = (math.exp(-Lambda) * (Lambda**x)) / math.factorial(x)
  results.append(p_x)

for x in range(2):
  print(f"P(r = {x}) = {results[x]}")

p_over_2_person = 1 - (results[0] + results[1])
print(f"P(r >= 2) = {p_over_2_person}")
```

```
P(r = 0) = 0.8607079764250578
P(r = 1) = 0.12910619646375868
P(r >= 2) = 0.010185827111183543
```

Second Question

> Based on the output rate PMF, compute a consistent slot assignment for the CFP in a monitoring system composed of 1 PAN coordinator and 3 camera nodes. Assume nominal bit rate R=250kbps, packets of L=128bytes, 1 packet fits exactly in one slot. Compute Ts (slot time), Number of slots in the CFP, Tactive, Tinactive and the duty cycle of the system.

In [44]:
```python
num_camera_nodes = 3

num_nodes = num_camera_nodes + 1 # 3 camera nodes + 1 PAN coordinator

nominal_bit_rate_R = 250 * 1000 # bps

packet_size_L = 128 # bytes

bytes_to_bits = 8

superframe_time = 10 * 1e3 #ms
```

In [45]:
```python
Ts = (packet_size_L * bytes_to_bits)*1e3 / nominal_bit_rate_R
print(f"\nTs (slot time): {Ts:.6f} ms")
```

```
Ts (slot time): 4.096000 ms
```

In [46]:
```python
E_r = (r0 * results[0]) + (r1 * results[1]) + (r2 * p_over_2_person)

print(f"\nExpected output rate E[r]: {E_r:.4f} KB/s per node")
```

```
Expected output rate E[r]: 0.1309 KB/s per node
```

In [47]:
```python
expected_total_rate = num_camera_nodes * E_r
print(f"Total expected rate for {num_camera_nodes} nodes: {expected_total_rate
```

```
Total expected rate for 3 nodes: 0.39274 KB/s
```

In [ ]:
```python
worst_case_rate_per_node = 6 # KB every 10 seconds
total_worst_case_rate = num_camera_nodes * worst_case_rate_per_node
print(f"\nTotal worst-case rate for {num_camera_nodes} nodes: {total_worst_cas
```

```
Total worst-case rate for 3 nodes: 18 KB every 10 seconds
```

In [48]:
```python
worst_case_pacekts_per_superframe = total_worst_case_rate / (packet_size_L / 1

Ncfp = math.floor(worst_case_pacekts_per_superframe) + 1
```

```python
print(f"\nTotal worst-case packets per superframe: {Ncfp} packets")
```

```
Total worst-case packets per superframe: 141 packets
```

In [49]:
```python
#The Ncfp solts plus the beacon slot * time slot
active_time = (Ncfp+1) * Ts
print(f"\nTotal active time: {active_time:.4f} ms")

inactive_time = superframe_time - active_time
print(f"Total inactive time: {inactive_time:.4f} ms")

duty_cycle = (active_time / superframe_time) * 100
print(f"Duty cycle: {duty_cycle:.4f} %")
```

```
Total active time: 581.6320 ms
Total inactive time: 9418.3680 ms
Duty cycle: 5.8163 %
```

Third Exercise

> How many additional cameras can be added to keep the duty cycle below 10%?

In [60]:
```python
packets = worst_case_rate_per_node * 1000 / packet_size_L

packets = math.floor(packets) + 1

print(f"Total packets: {packets}")

max_active_time = superframe_time * 0.1

max_slots = max_active_time / Ts

max_slots = math.floor(max_slots)

print(f"Maximum slots: {max_slots}")
```

```
Total packets: 47
Maximum slots: 244
```

In [61]:
```python
maximum_camera = max_slots / packets

maximum_camera = math.floor(maximum_camera)

print(f"Maximum camera: {maximum_camera}")
```

```
Maximum camera: 5
```

In [62]:
```python
tot_slot_updated = maximum_camera * packets

new_active_time = (tot_slot_updated+1) * Ts

new_inactive_time = superframe_time - new_active_time
```

```python
new_duty_cycle = (new_active_time / superframe_time) * 100

print(f"New duty cycle: {new_duty_cycle:.4f} %")
```

```
New duty cycle: 9.6666 %
```

We can conlude that the max number of cameras that we add is equal to 2. Below we can notice the proof that with 3 cameras the dury cycle over the 10%.

In [65]:
```python
cameras_over_the_boundaries = maximum_camera + 1

tot_slot_updated_updated = cameras_over_the_boundaries * packets

new_active_time = (tot_slot_updated_updated+1) * Ts

new_inactive_time = superframe_time - new_active_time

new_duty_cycle = (new_active_time / superframe_time) * 100

print(f"New duty cycle: {new_duty_cycle:.4f} %")
```

```
New duty cycle: 11.5917 %
```

> We can conclude that 2 is the correct response.