IoT Homework 2024/2025
PART 3 – Exercise3: RFID

Leader: Giovanni Ni 10831328

Xinyue Gu 10840236

Exercise 3 - RFID

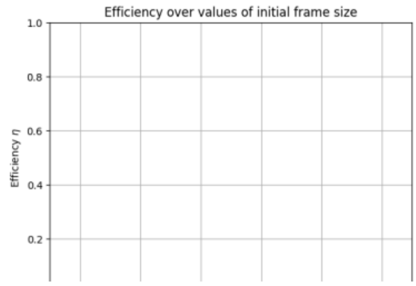
A RFID system based on Dynamic Frame ALOHA is composed of N=4 tags

1. Find the overall collision resolution efficiency **η** in the different cases in which the initial frame size is set to **r1=**1,2,3,4,5,6

- Assume that after the first frame, the frame size is correctly set to the current backlog size
- Assume as given the duration of the arbitration period with N=2,3 tags when r=N

(L_2=4, L_3=51/8)

2. After computing the values of the efficiency with the different frame sizes, **produce a plot** with values of **η over r1** (as figure) -> add in the report



3. For what values of **r1** we have the maximum value for **η? Comment.**

1. $\eta = \frac{N}{L_N}$ in our case N = 4 tags

   We should calculate L4 for r1 = 1,2,3,4,5,6

   The general formula is:

$$L_4^* = r1 + \sum_{i=0}^{3} P(S=i) * L_{4-i}$$

$$= r1 + L_4 P(S=0) + L_3 P(S=1) + L_2 P(S=2) + L_1 P(S=3)$$

P(S=k) stands for having exactly k slots with only a tag.
So we should calculate this value for different cases (r1 = 1,2,3,4,5,6).
I wrote a Python script that enumerates all possible assignments and counts how many of them satisfy the constraint: '*exactly k slots contain exactly one tag*.
Anyway I'm going to show my mathematical logic in first cases.

<u>r1 = 1</u> at beginning we have only one slot
The probability of having 1,2,3 slots with only 1 tag is 0, so P(S=0) = 1

$$L_4^* = r1 + L_4 * 1$$

At the second round we have 4 slots and 4 tags, so L4 on the right of the equation is equal to the left one:

$$L4 = 4 + L_4 P(S=0) + L_3 P(S=1) + L_2 P(S=2) + L_1 P(S=3)$$

P(S=0) having exactly 0 slot with only a tag

(1/4)^4 * N (number of combination with exactly 0 slot with only a tag.

- each slot contains all 4 tags: 4 combinations.
- 2 slots contains 2 tags, and the other two slots contain 0 tags: 36 combinations.

P(S=0) = (4+36) / 4^4

The general logic is finding number of combinations of each case and multiply by (1/4)^r1, obtaining the probability.
So I use my code to find the number of combinations in each case.

```
0 slots with 1 tag: 40 combinations
1 slots with 1 tag: 48 combinations
2 slots with 1 tag: 144 combinations
4 slots with 1 tag: 24 combinations
```

P(S=0) = 40/256
P(S=1) = 48/256
P(S=2) = 144/256
P(S=3) = 0

$$L4 = 4 + L_4 P(S=0) + L_3 P(S=1) + L_2 P(S=2) + L_1 P(S=3) \rightarrow L4 = 8.824$$

L4* = 1+8.824

$$\eta_1 = \frac{4}{9.824} = 0.407$$

r1=2

L4* = r1 + L4 P(S=0) + L3P(S=1)

```
0 slots with 1 tag: 8 combinations
1 slots with 1 tag: 8 combinations
```

P(S=0) = 8/16
P(S=1) = 8/16
L4* = 2+8.824*P(S=0) + 51/8 * P(S=1) = 9.5995

$$\eta_2 = \frac{4}{9.5995} = 0.4167$$

r1 = 3

L4* = r1 + L4 P(S=0) + L3P(S=1) +L2P(S=2)

```
0 slots with 1 tag: 21 combinations
1 slots with 1 tag: 24 combinations
2 slots with 1 tag: 36 combinations
```

P(S=0) = 21/81
P(S=1) = 24/81
P(S=2) = 36/81

L4* = 3 + 8.824* P(S=0) + 51/8 * P(S=1) +4*P(S=2) = 8.954

$$\eta_3 = \frac{4}{8.954} = 0.4467$$

r1=4:

$$\eta_4 = \frac{4}{8.824} = 0.453$$

r1=5

L4* = r1 + L4 P(S=0) + L3P(S=1) +L2P(S=2) + ~~L1P(S=3)~~

```
0 slots with 1 tag: 65 combinations
1 slots with 1 tag: 80 combinations
2 slots with 1 tag: 360 combinations
4 slots with 1 tag: 120 combinations
```

P(S=0) = 65/625

P(S=1) = 80/625

P(S=2) = 360/625

L4* = 5 + 8.824*P(S=0) + 51/8 * P(S=1) +4* P(S=2) = 9.038

$$\eta_5 = \frac{4}{9.038} = 0.4426$$

r1 = 6

L4* = r1 + L4 P(S=0) + L3P(S=1) +L2P(S=2) + ~~L1P(S=3)~~

```
0 slots with 1 tag: 96 combinations
1 slots with 1 tag: 120 combinations
2 slots with 1 tag: 720 combinations
4 slots with 1 tag: 360 combinations
```

P(S=0) = 96/1296

P(S=1) = 120/1296

P(S=2) = 720/1296

L4* = 6+ 8.824 *P(S=0) + 51/8 * P(S=1) +4 * P(S=2) = 9.466

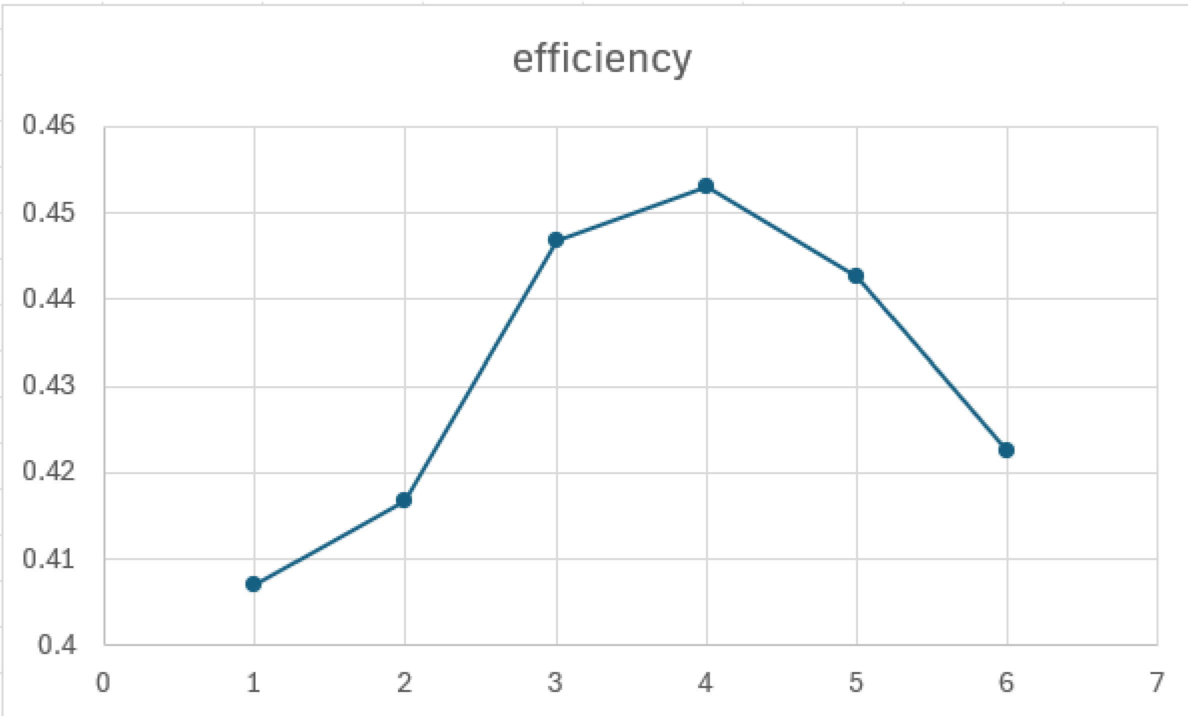
$$\eta_6 = \frac{4}{9.466} = 0.4225$$

Code I used to find number of combinations:

```
from itertools import product
from collections import Counter


positions_6slots = list(product(range(6), repeat=4)) # in the range I insert the number  of available slots, and repeat = 4 (tags number)
                                                      # change the range manually every round 1,2,3,4,5,6
resolved_count_distribution_6slots = Counter()

for p in positions_6slots:
    slot_counts = Counter(p)
    num_slots_with_one = sum(1 for count in slot_counts.values() if count == 1)
    resolved_count_distribution_6slots[num_slots_with_one] += 1

resolved_count_distribution_6slots = dict(sorted(resolved_count_distribution_6slots.items()))

for num_slots, count in resolved_count_distribution_6slots.items():
    print(f"{num_slots} slots with 1 tag: {count} combinations")
```

---

r1	efficiency
1	0.407
2	0.4167
3	0.4467
4	0.453
5	0.4426
6	0.4225



We have the maximum value for the efficiency at r1 =4.

This is obvious because we have 4 tags and the initial frame size is set to 4. These two number are equals. In this case, the chance of exactly one tag per slot is maximized. it's a balance between too many collisions and too many empty slots.