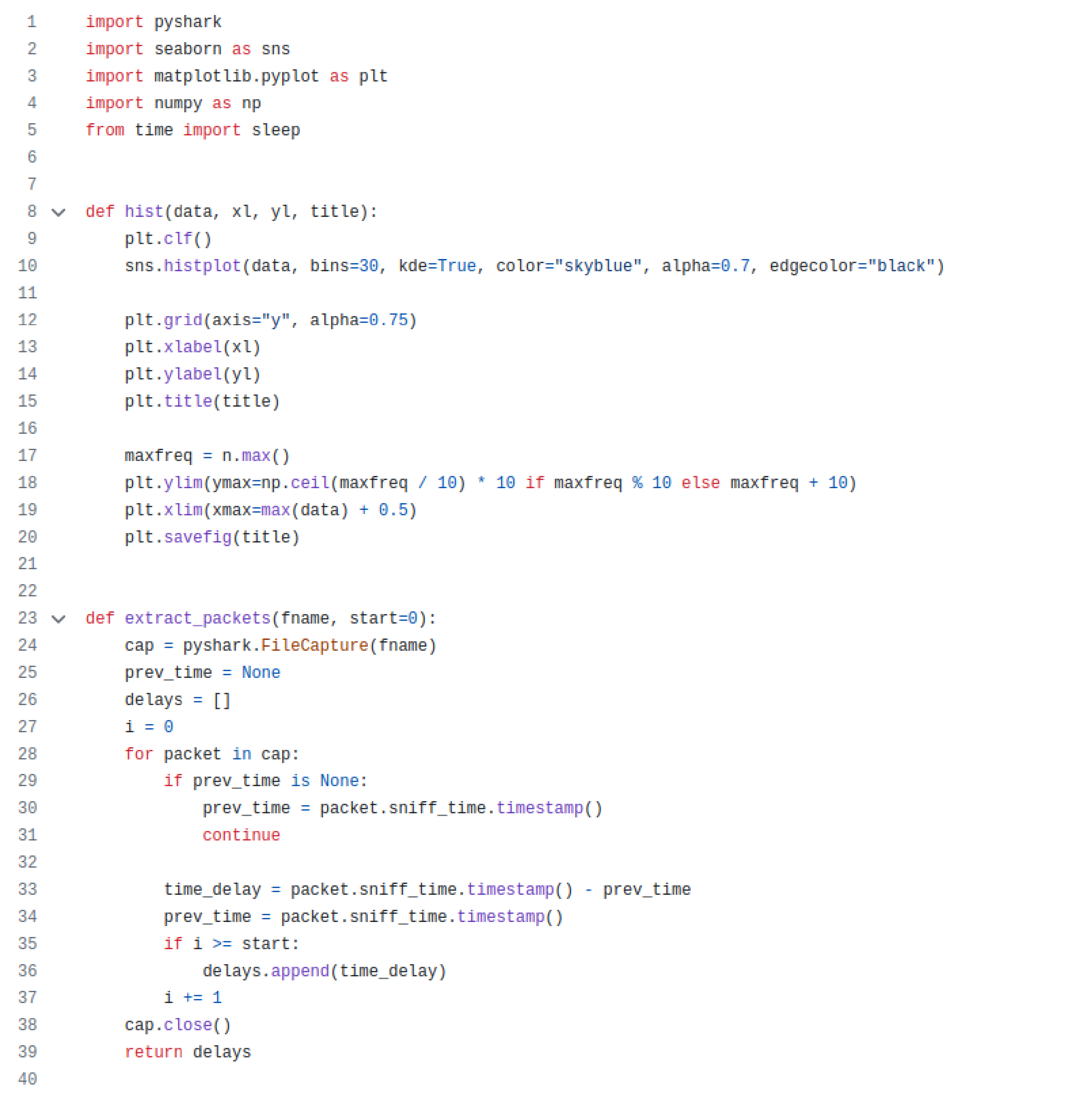
Скрытые Каналы. Лабораторная работа #2.

Соколов А.Д. Б20-505

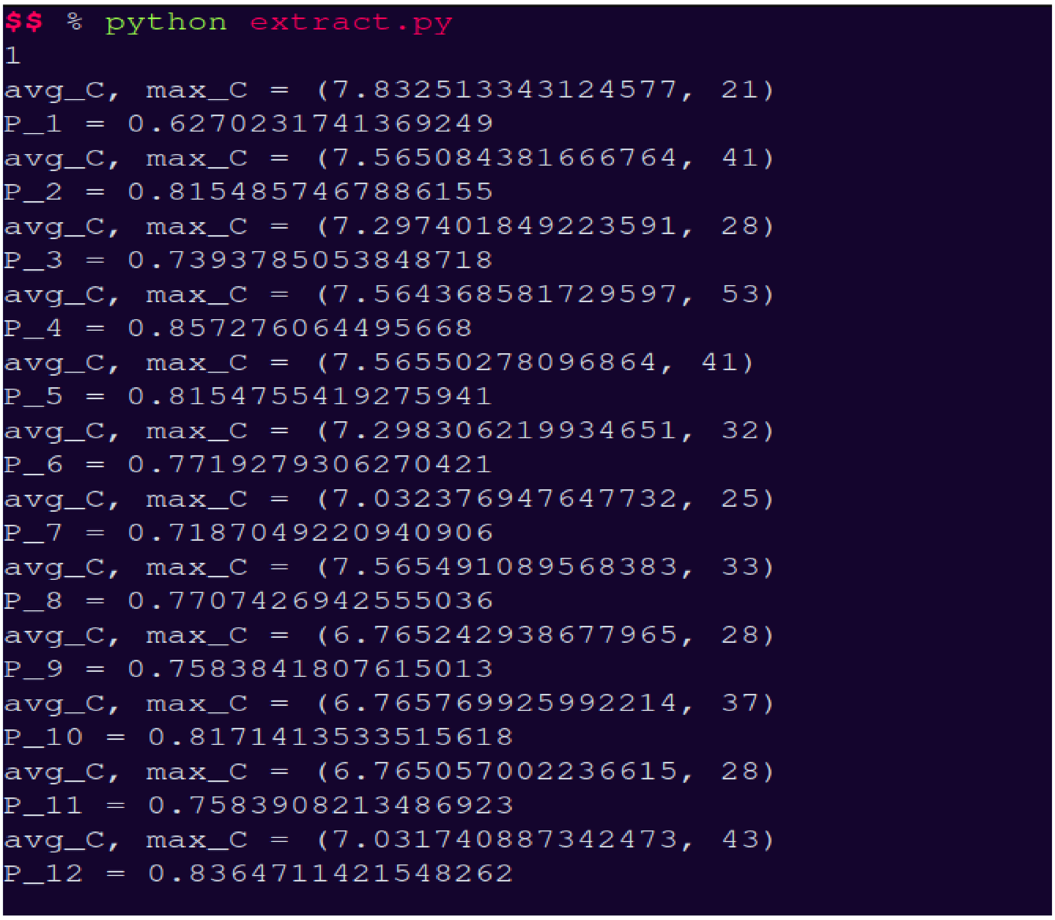
Задание 1/2



```python
import pyshark
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from time import sleep


def hist(data, xl, yl, title):
    plt.clf()
    sns.histplot(data, bins=30, kde=True, color="skyblue", alpha=0.7, edgecolor="black")

    plt.grid(axis="y", alpha=0.75)
    plt.xlabel(xl)
    plt.ylabel(yl)
    plt.title(title)

    maxfreq = n.max()
    plt.ylim(ymax=np.ceil(maxfreq / 10) * 10 if maxfreq % 10 else maxfreq + 10)
    plt.xlim(xmax=max(data) + 0.5)
    plt.savefig(title)


def extract_packets(fname, start=0):
    cap = pyshark.FileCapture(fname)
    prev_time = None
    delays = []
    i = 0
    for packet in cap:
        if prev_time is None:
            prev_time = packet.sniff_time.timestamp()
            continue

        time_delay = packet.sniff_time.timestamp() - prev_time
        prev_time = packet.sniff_time.timestamp()
        if i >= start:
            delays.append(time_delay)
        i += 1
    cap.close()
    return delays

```

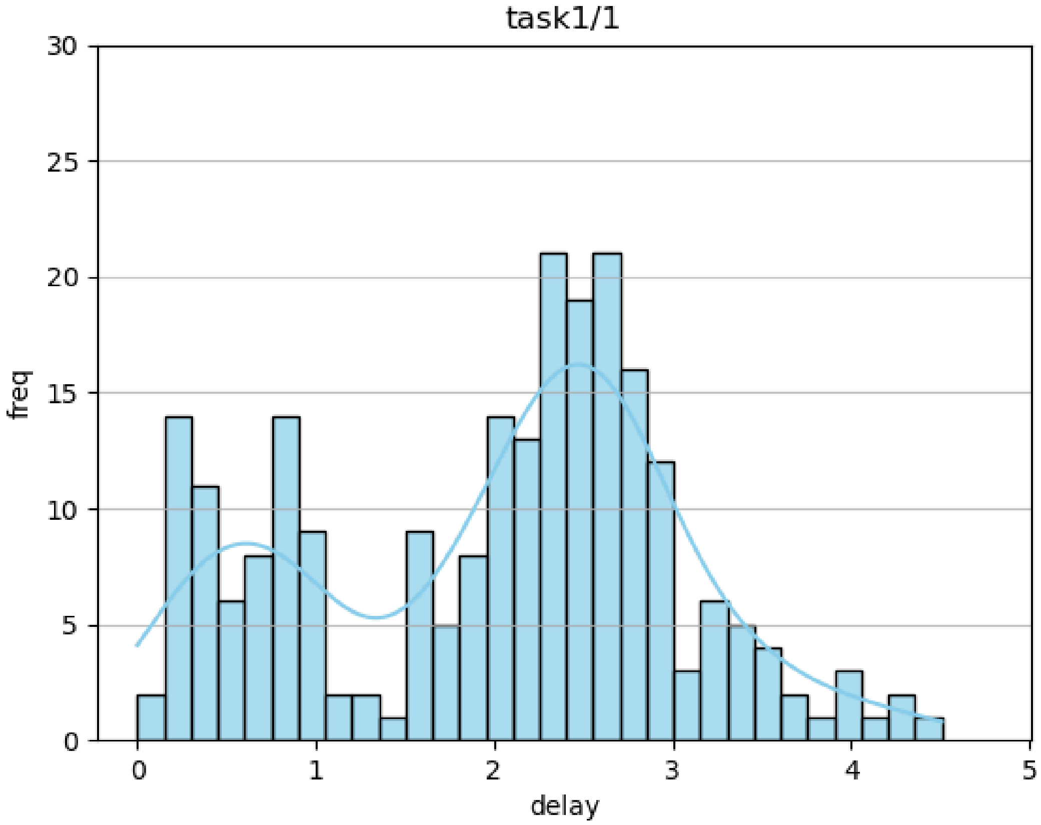
```python

print("1")
for i in range(1, 13):
    delays = extract_packets(f"{i}.pcapng")
    n, ints = np.histogram(delays, bins=30)
    avg = sum(delays) / len(delays)

    max_delay = max(delays)
    avg_C = (
        sum(n[i] * (ints[i + 1] - ints[i]) for i in range(len(ints) - 1)) / max_delay
    )

    hist(delays, "delay", "freq", f"task1/{i}")
    max_C = max(n)
    print(f"{avg_C, max_C = }")
    print(f"P_{i} = {1 - avg_C / max_C}")
print()
print("2")
for i in range(1, 13):
    delays = extract_packets(f"{i}.pcapng", 99)
    n, ints = np.histogram(delays[99:], bins=30)
    avg = sum(delays) / len(delays)

    max_delay = max(delays)
    avg_C = (
        sum(n[i] * (ints[i + 1] - ints[i]) for i in range(len(ints) - 1)) / max_delay
    )

    hist(delays, "delay", "freq", f"task2/{i}")
    max_C = max(n)
    print(f"{avg_C, max_C = }")
    print(f"P_{i} = {1 - avg_C / max_C}")
print()
```

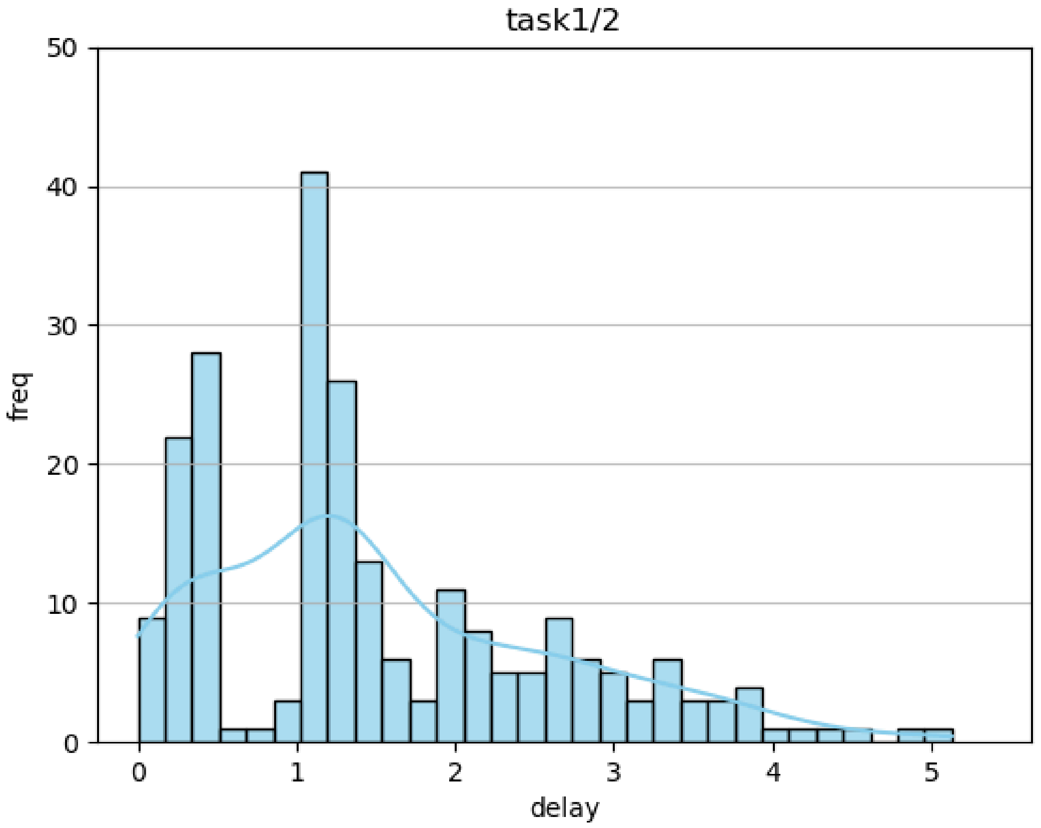
Вероятности

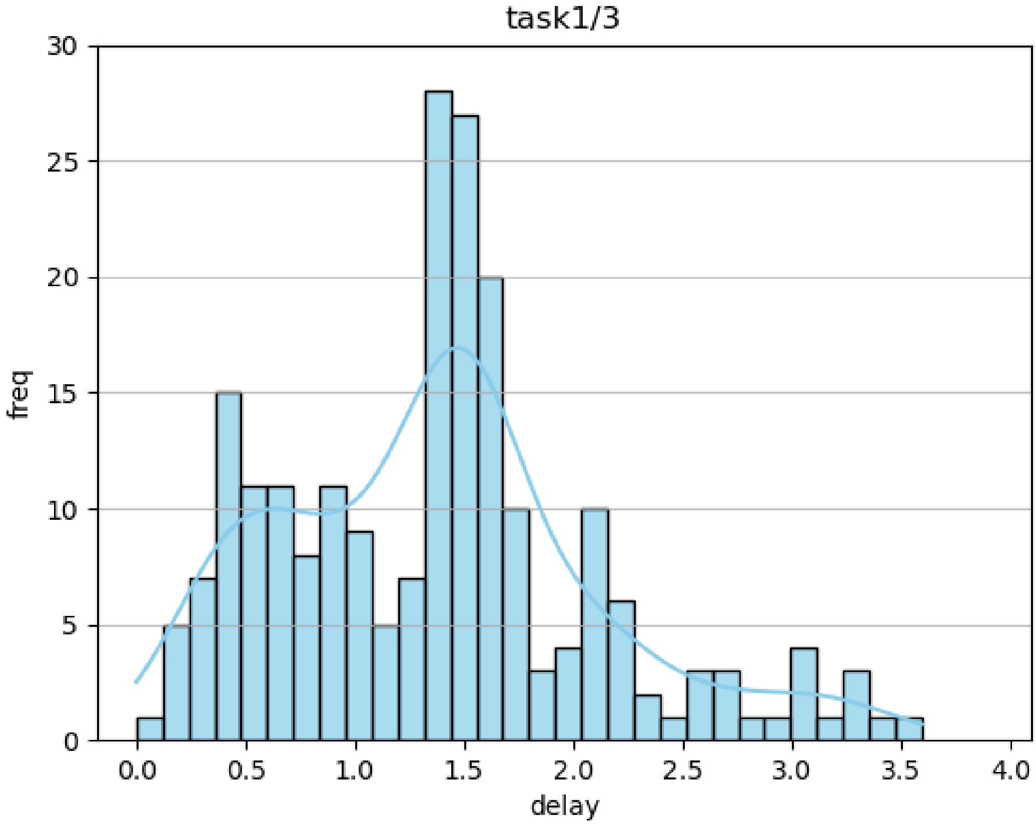


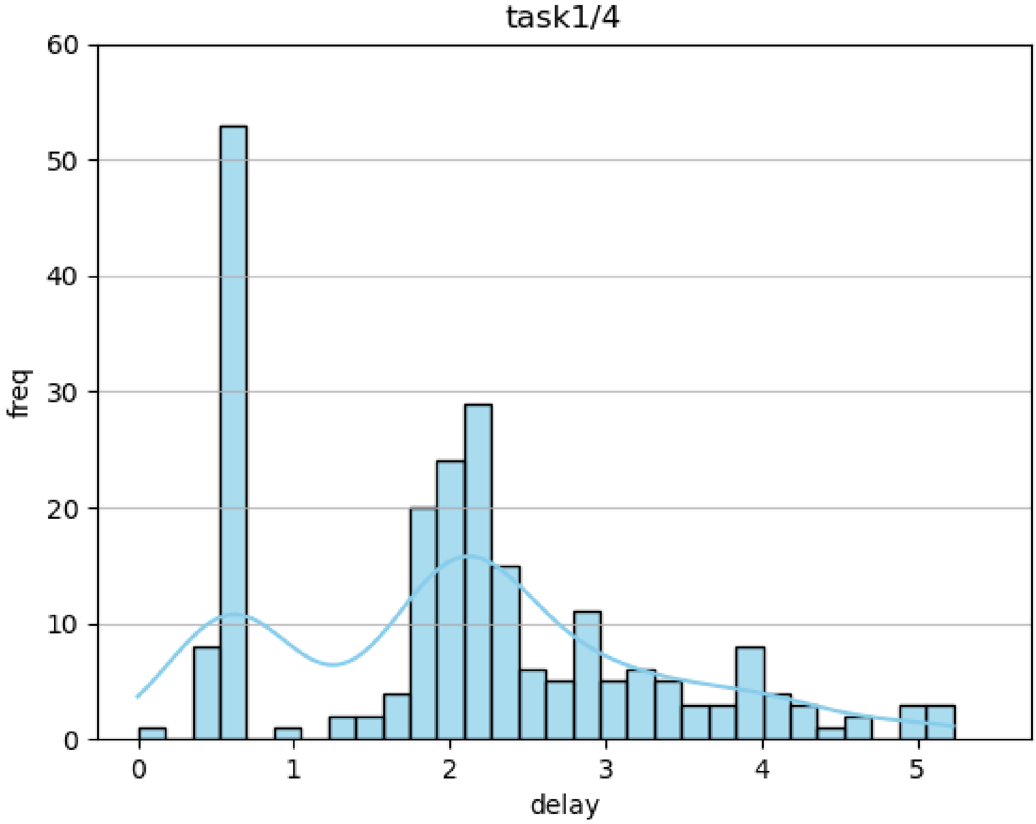
```
$$ % python extract.py
1
avg_C, max_C = (7.832513343124577, 21)
P_1 = 0.6270231741369249
avg_C, max_C = (7.565084381666764, 41)
P_2 = 0.8154857467886155
avg_C, max_C = (7.297401849223591, 28)
P_3 = 0.7393785053848718
avg_C, max_C = (7.564368581729597, 53)
P_4 = 0.857276064495668
avg_C, max_C = (7.56550278096864, 41)
P_5 = 0.8154755419275941
avg_C, max_C = (7.298306219934651, 32)
P_6 = 0.7719279306270421
avg_C, max_C = (7.032376947647732, 25)
P_7 = 0.7187049220940906
avg_C, max_C = (7.565491089568383, 33)
P_8 = 0.7707426942555036
avg_C, max_C = (6.765242938677965, 28)
P_9 = 0.7583841807615013
avg_C, max_C = (6.765769925992214, 37)
P_10 = 0.8171413533515618
avg_C, max_C = (6.765057002236615, 28)
P_11 = 0.7583908213486923
avg_C, max_C = (7.031740887342473, 43)
P_12 = 0.8364711421548262
```

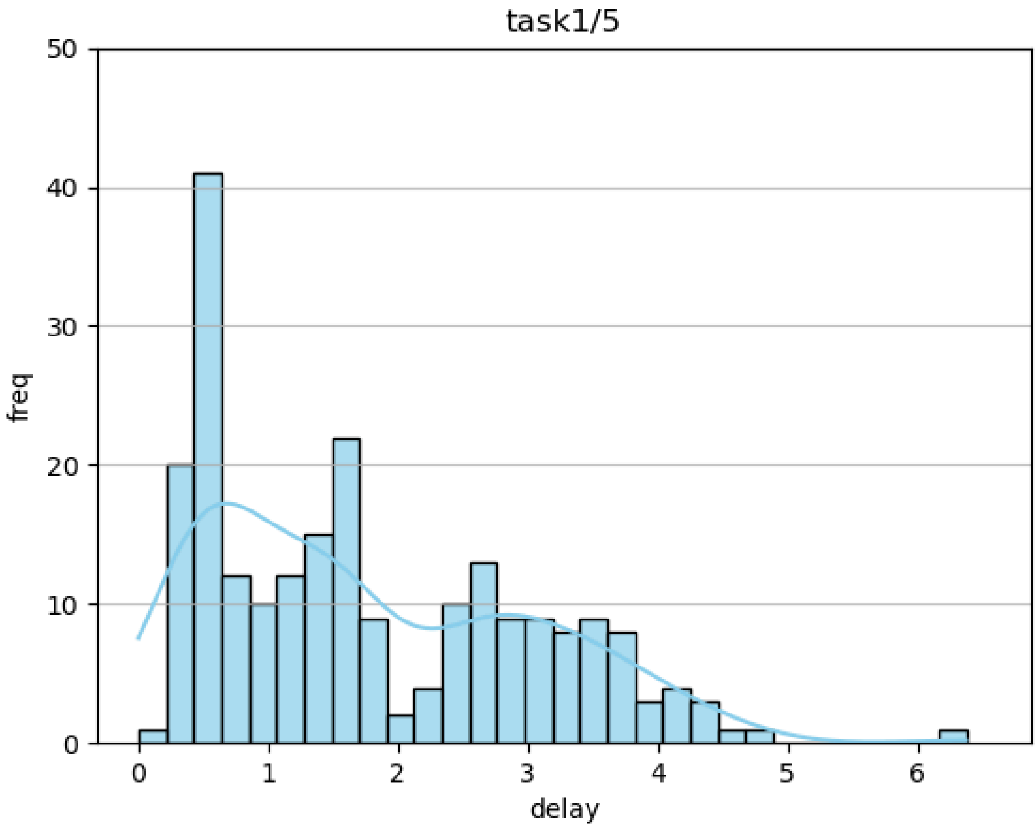
Гистограммы

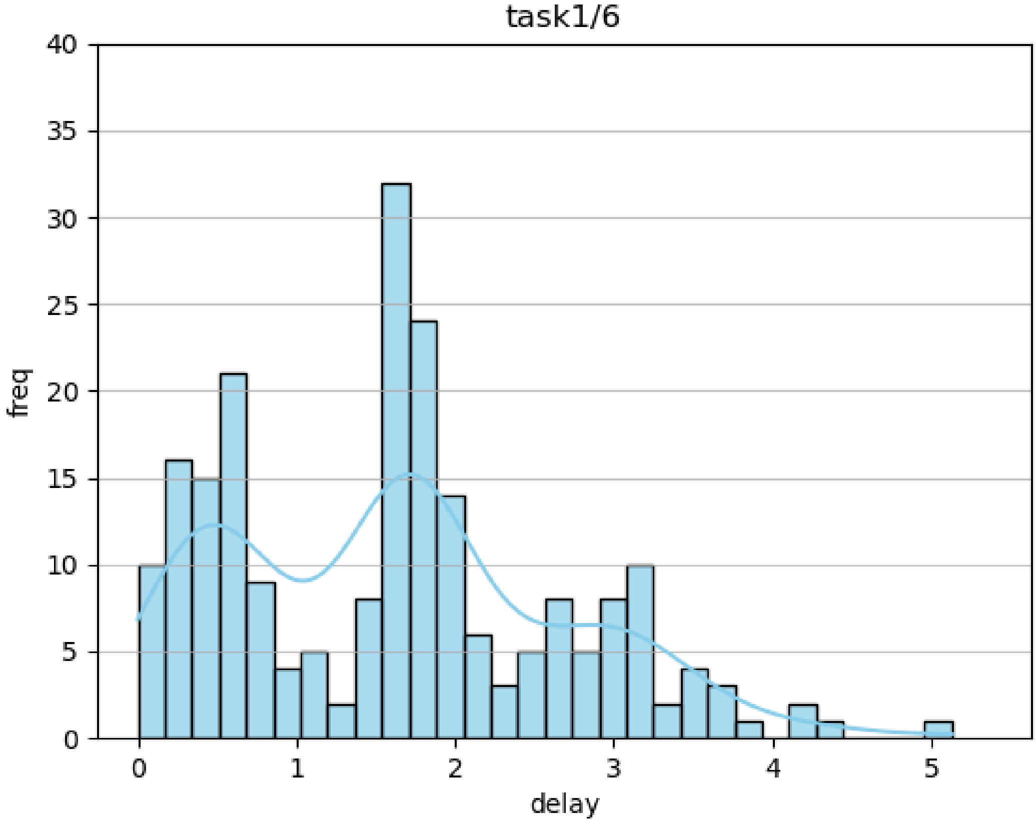

task1/1
freq
delay
0
5
10
15
20
25
30
0
1
2
3
4
5

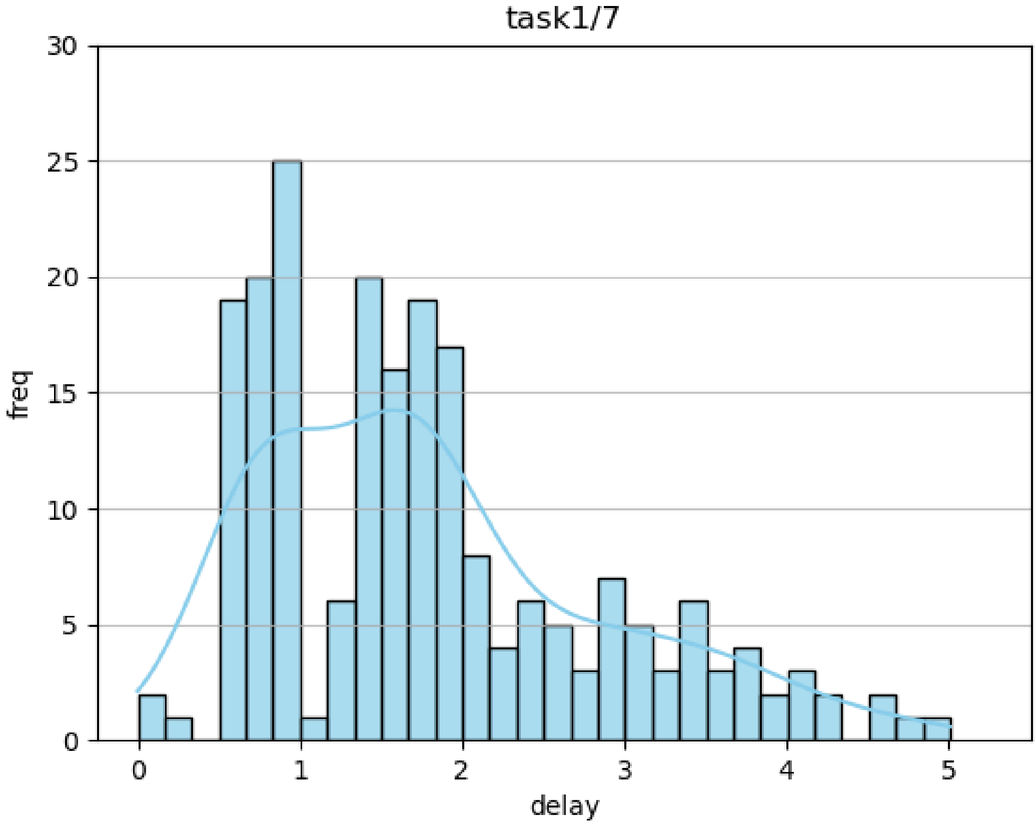

task1/2
freq
delay
0
10
20
30
40
50
0
1
2
3
4
5

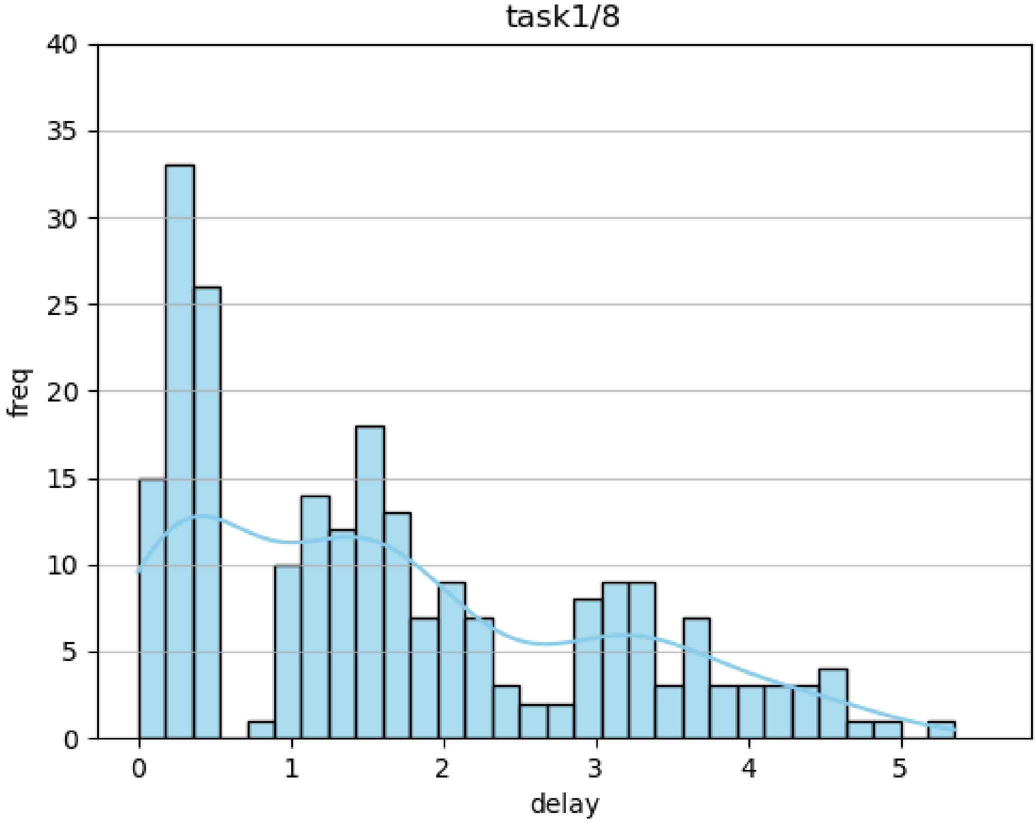

task1/3
freq
0
5
10
15
20
25
30
delay
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

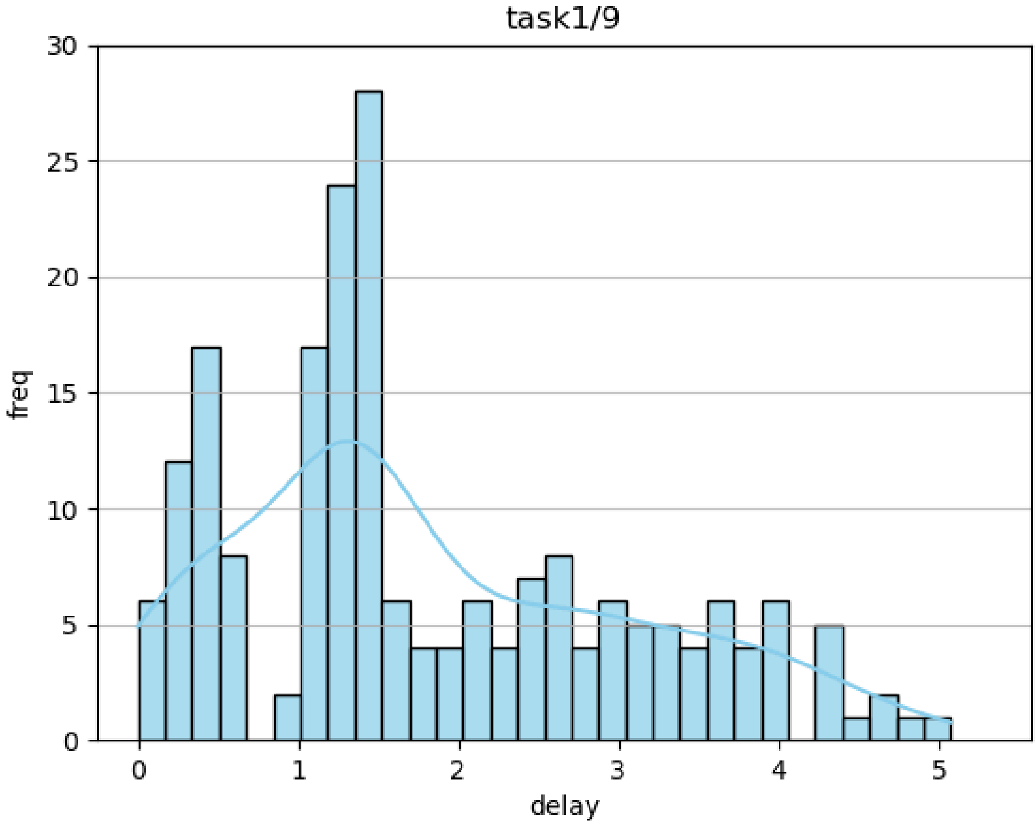

task1/4
freq
0
10
20
30
40
50
60
delay
0
1
2
3
4
5

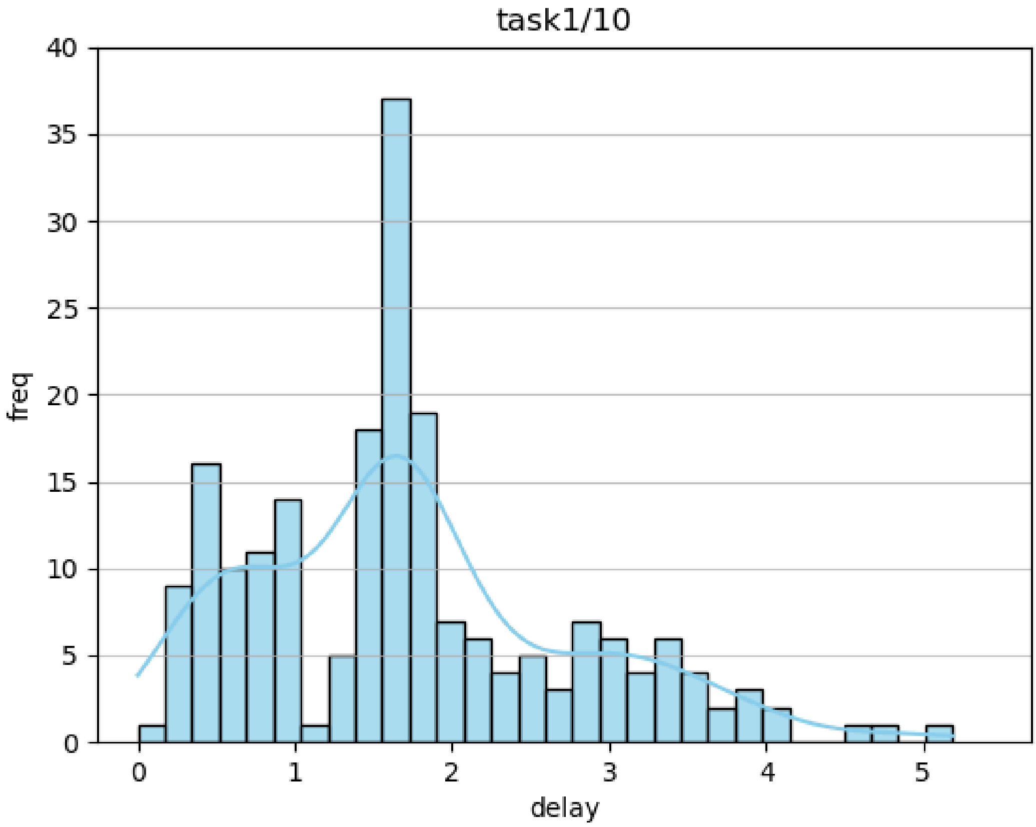

task1/5
freq
50
40
30
20
10
0
0
1
2
3
4
5
6
delay

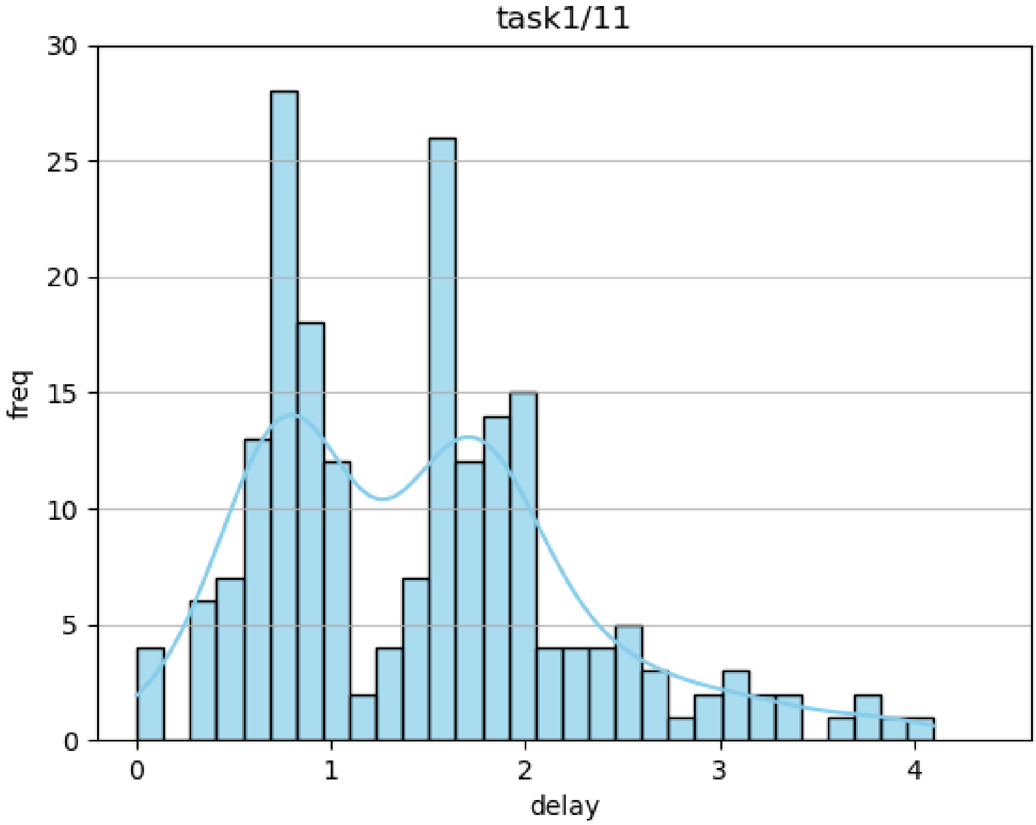

task1/6
freq
40
35
30
25
20
15
10
5
0
0
1
2
3
4
5
delay

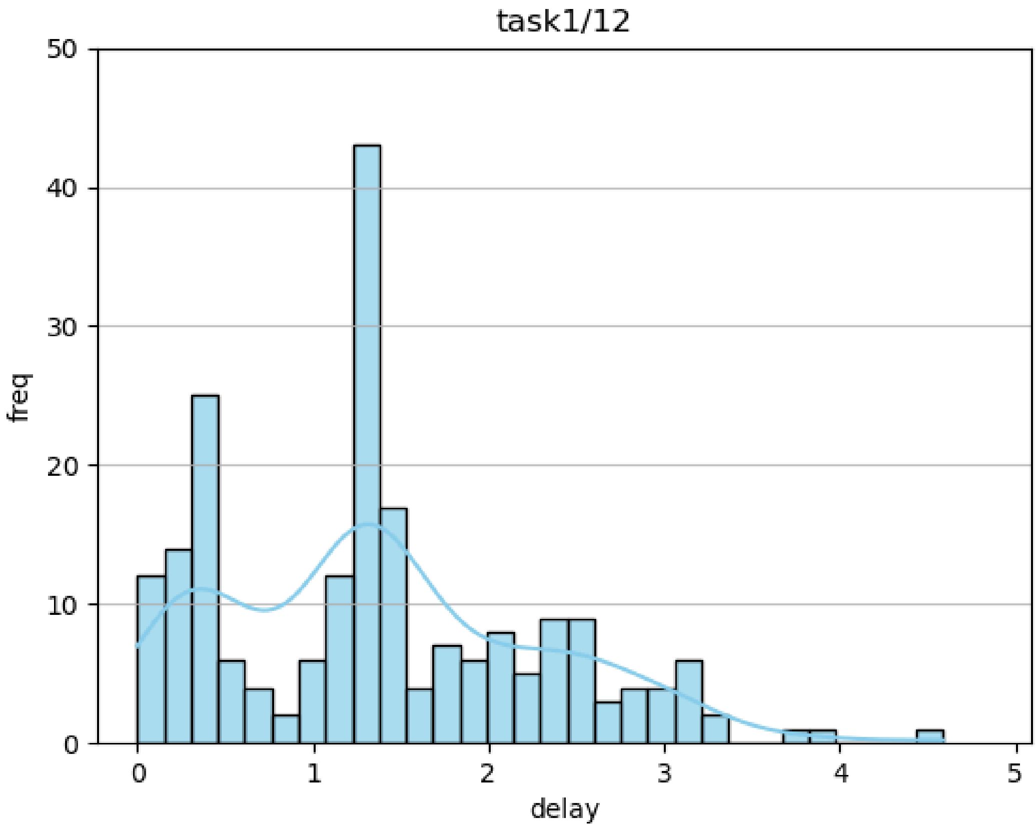

task1/7
freq
delay
0
5
10
15
20
25
30
0
1
2
3
4
5

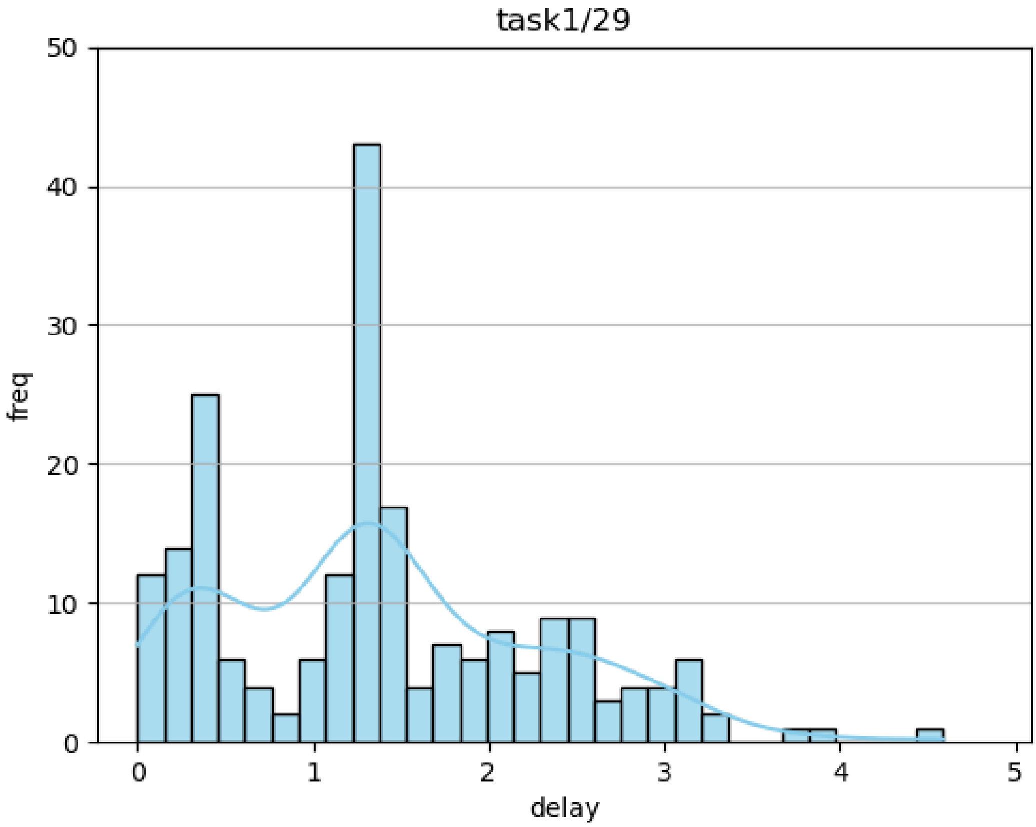

task1/8
freq
delay
0
5
10
15
20
25
30
35
40
0
1
2
3
4
5


task1/9
freq
0
5
10
15
20
25
30
delay
0
1
2
3
4
5

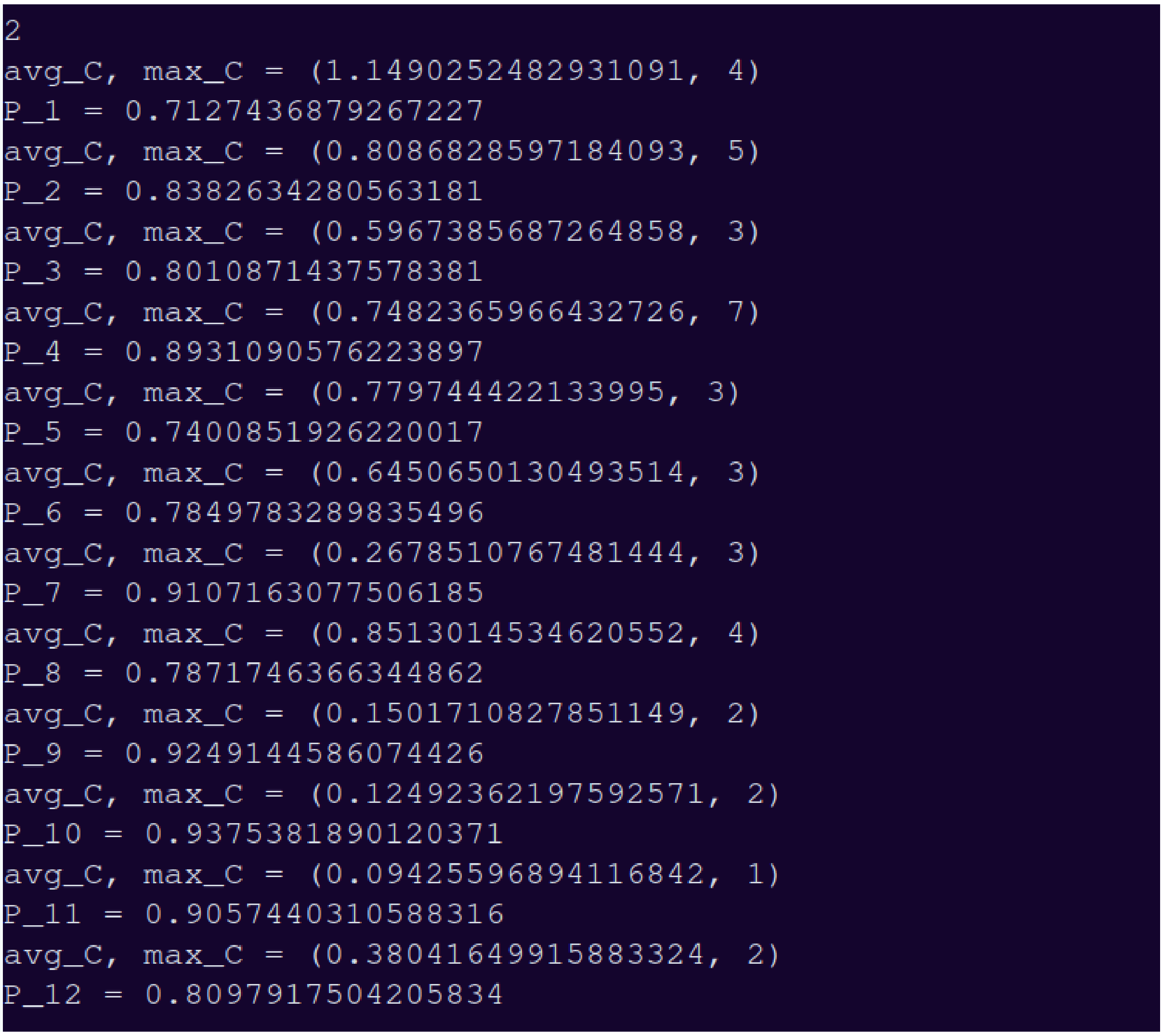

task1/10
freq
0
5
10
15
20
25
30
35
40
delay
0
1
2
3
4
5


task1/11
freq
delay
30
25
20
15
10
5
0
0
1
2
3
4


task1/12
freq
50
40
30
20
10
0
0
1
2
3
4
5
delay

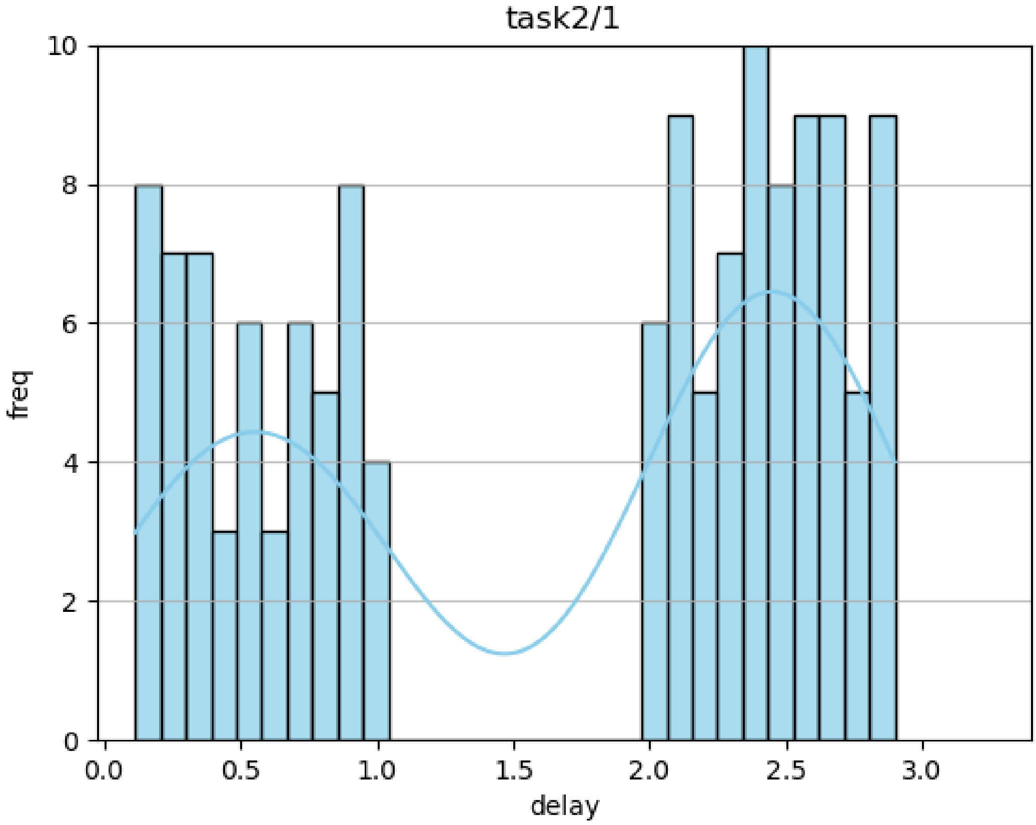

task1/29
freq
50
40
30
20
10
0
0
1
2
3
4
5
delay

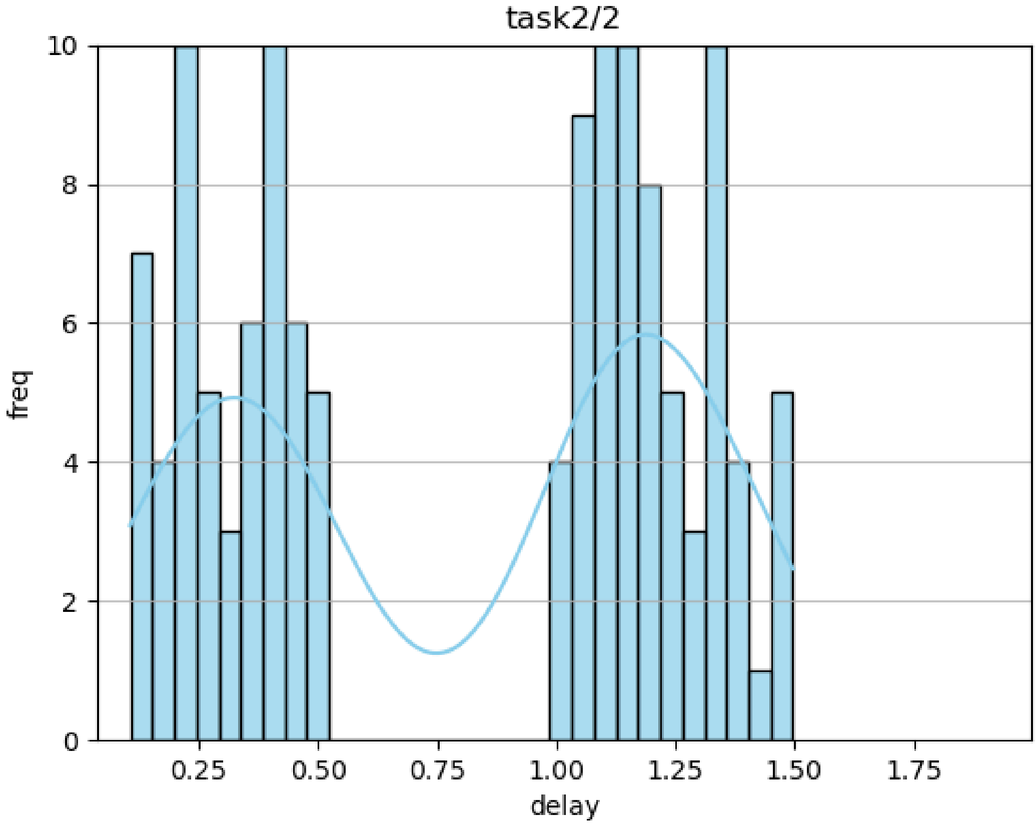
После того как проверяем с сотого пакета:

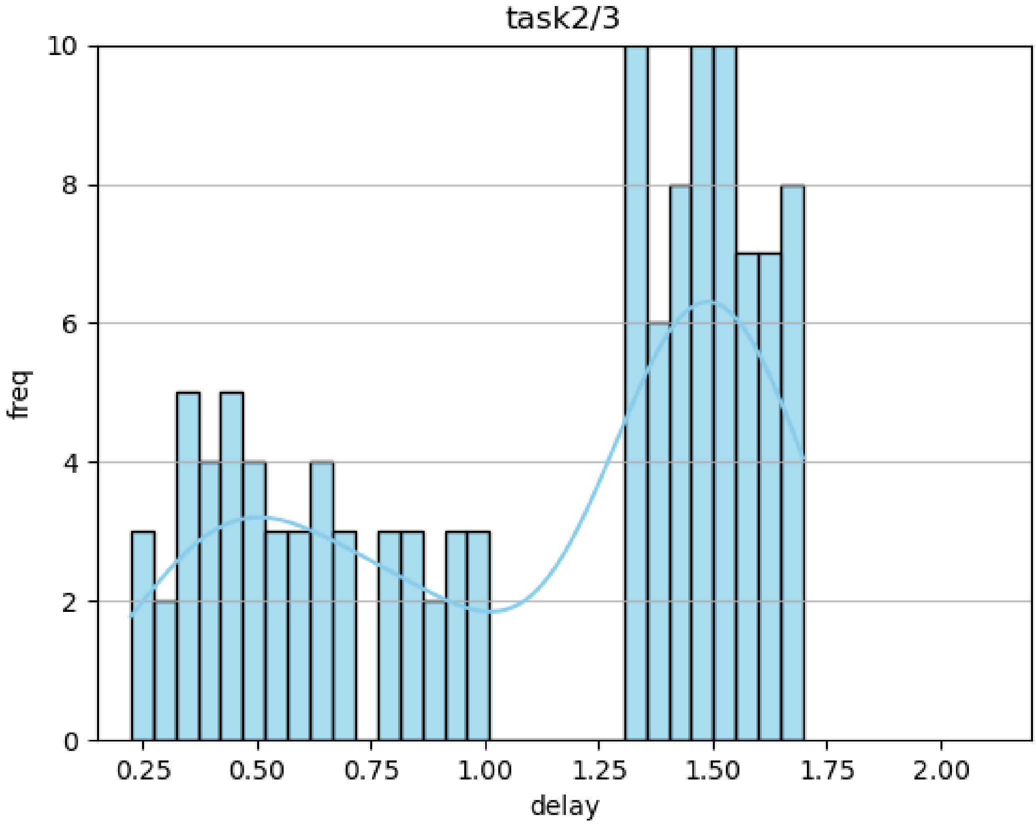


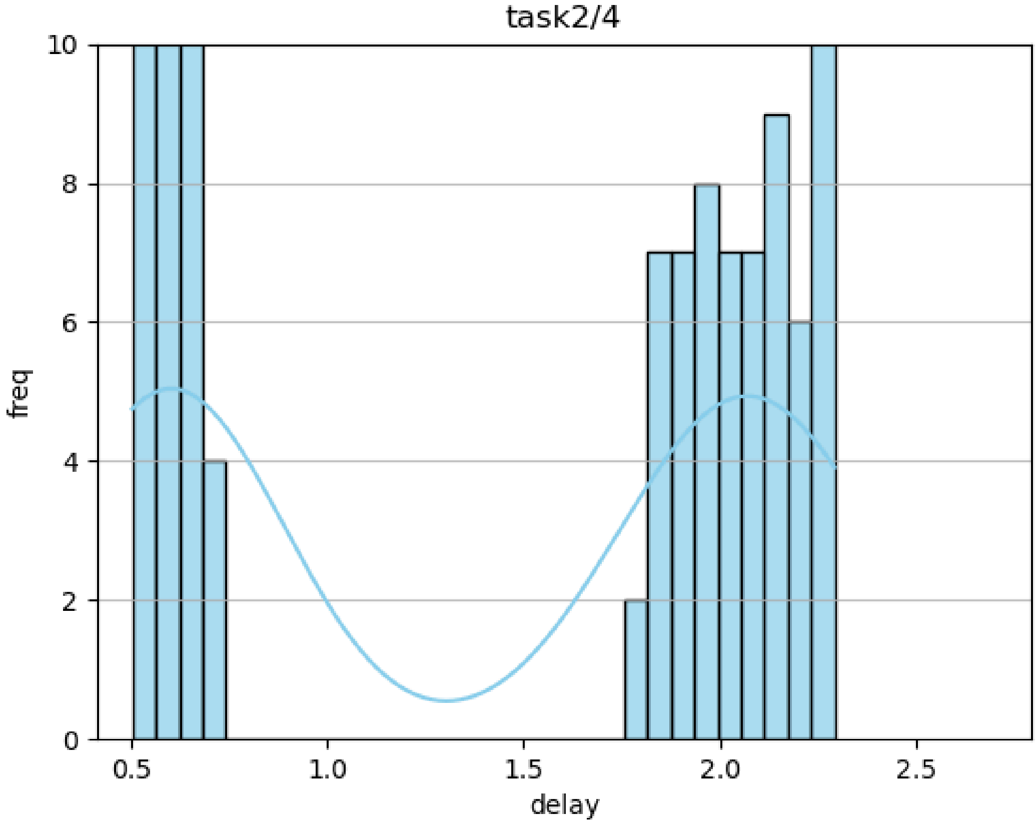
```
2
avg_C, max_C = (1.1490252482931091, 4)
P_1 = 0.7127436879267227
avg_C, max_C = (0.8086828597184093, 5)
P_2 = 0.8382634280563181
avg_C, max_C = (0.5967385687264858, 3)
P_3 = 0.8010871437578381
avg_C, max_C = (0.7482365966432726, 7)
P_4 = 0.8931090576223897
avg_C, max_C = (0.779744422133995, 3)
P_5 = 0.7400851926220017
avg_C, max_C = (0.6450650130493514, 3)
P_6 = 0.7849783289835496
avg_C, max_C = (0.2678510767481444, 3)
P_7 = 0.9107163077506185
avg_C, max_C = (0.8513014534620552, 4)
P_8 = 0.7871746366344862
avg_C, max_C = (0.1501710827851149, 2)
P_9 = 0.9249144586074426
avg_C, max_C = (0.12492362197592571, 2)
P_10 = 0.9375381890120371
avg_C, max_C = (0.09425596894116842, 1)
P_11 = 0.9057440310588316
avg_C, max_C = (0.38041649915883324, 2)
P_12 = 0.8097917504205834
```

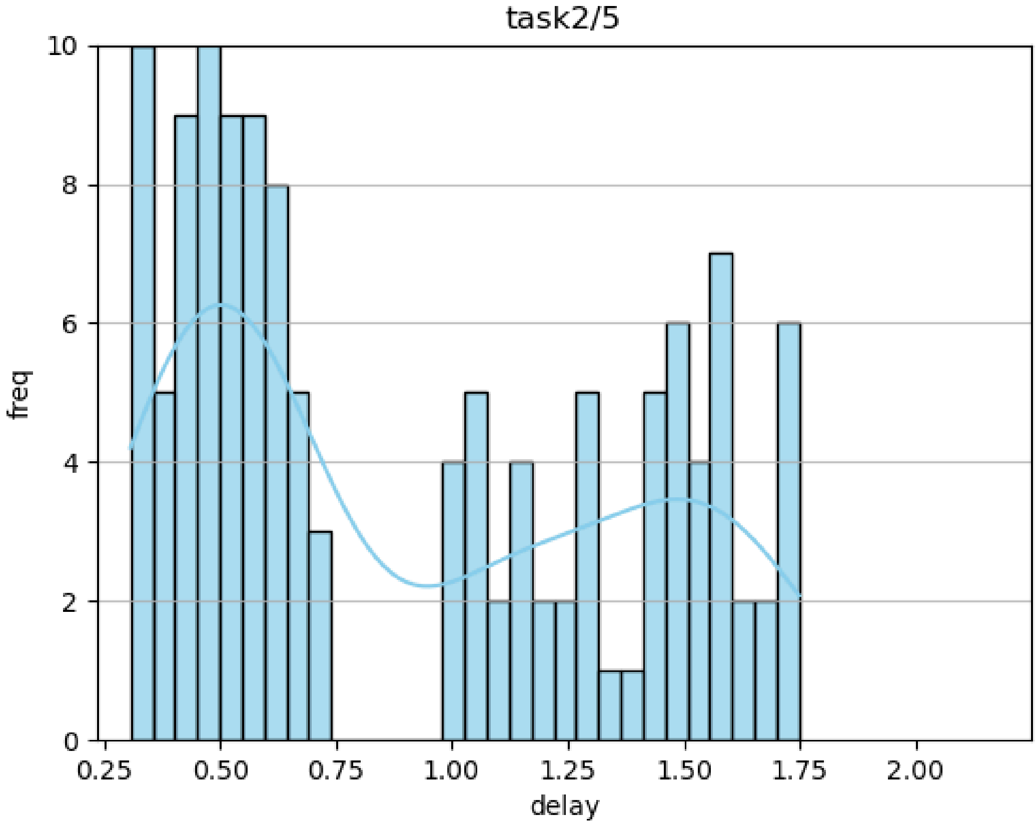
Гистограммы:

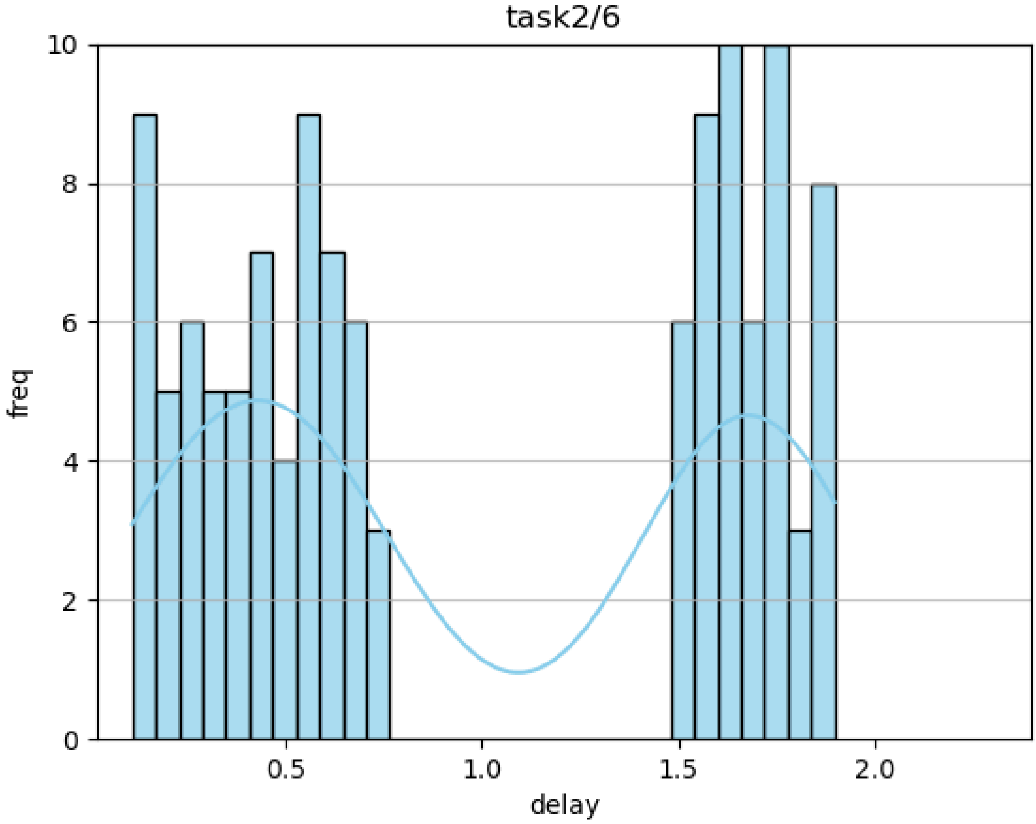

task2/1
freq
10
8
6
4
2
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
delay

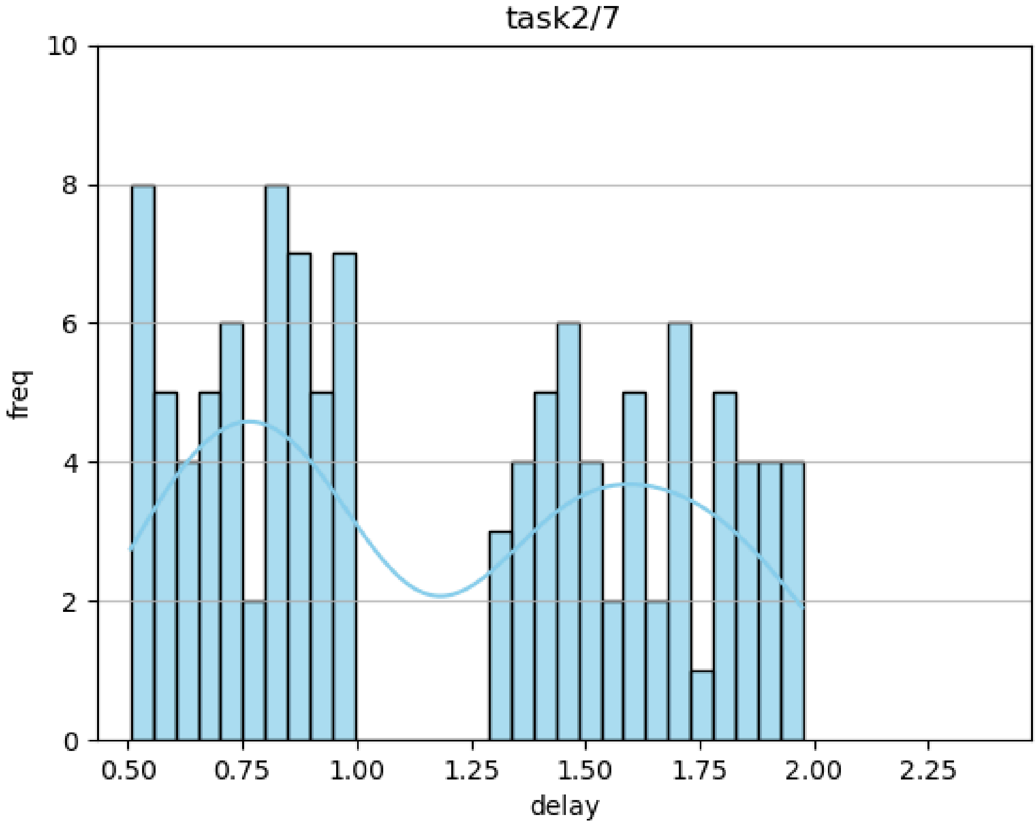

task2/2
freq
10
8
6
4
2
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
delay

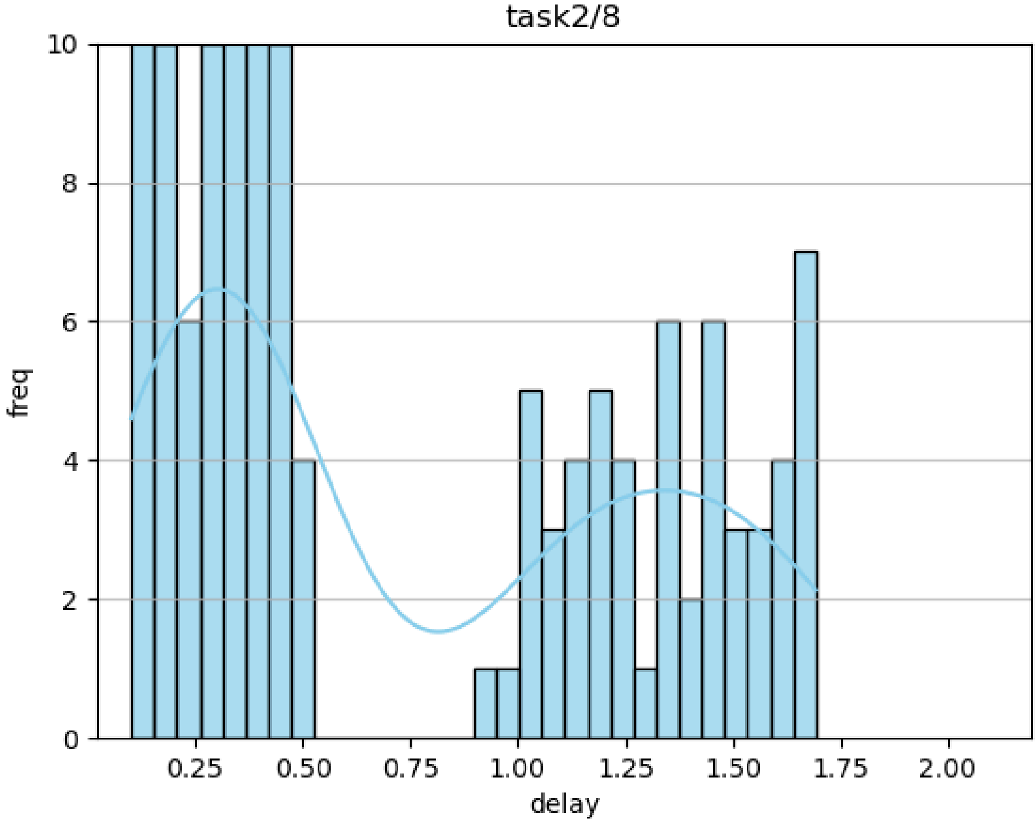

task2/3
freq
0
2
4
6
8
10
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
delay

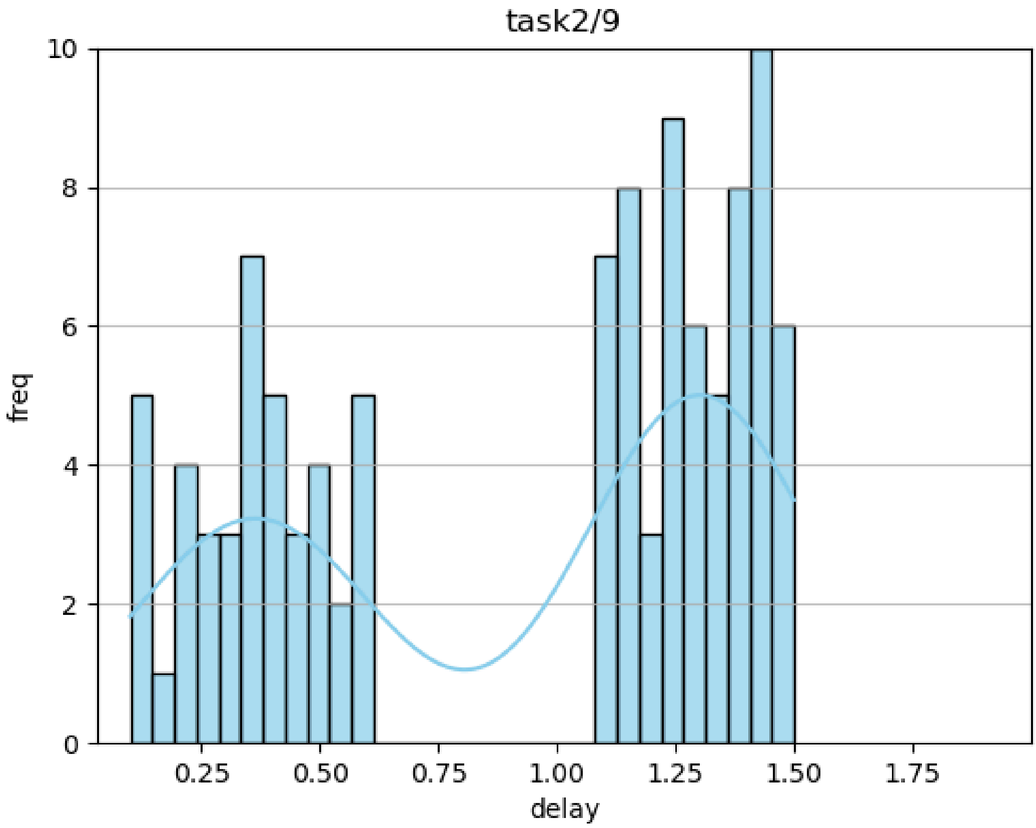

task2/4
freq
0
2
4
6
8
10
0.5
1.0
1.5
2.0
2.5
delay

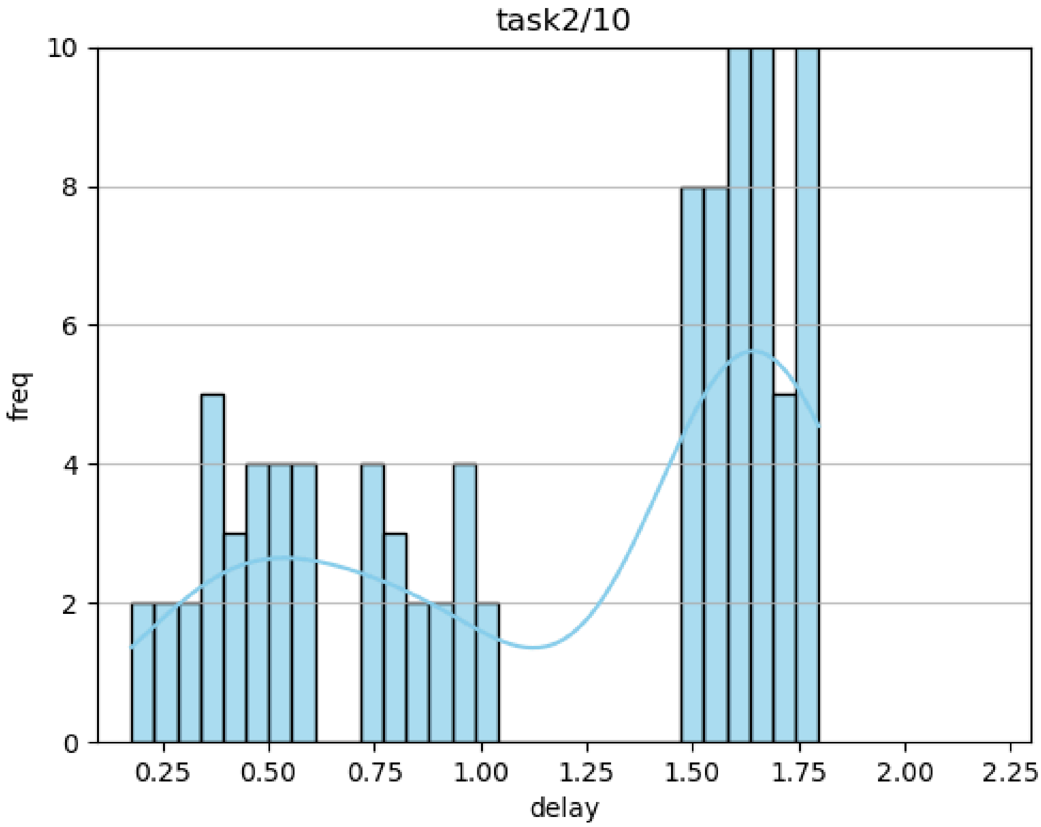

task2/5
freq
10
8
6
4
2
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
delay

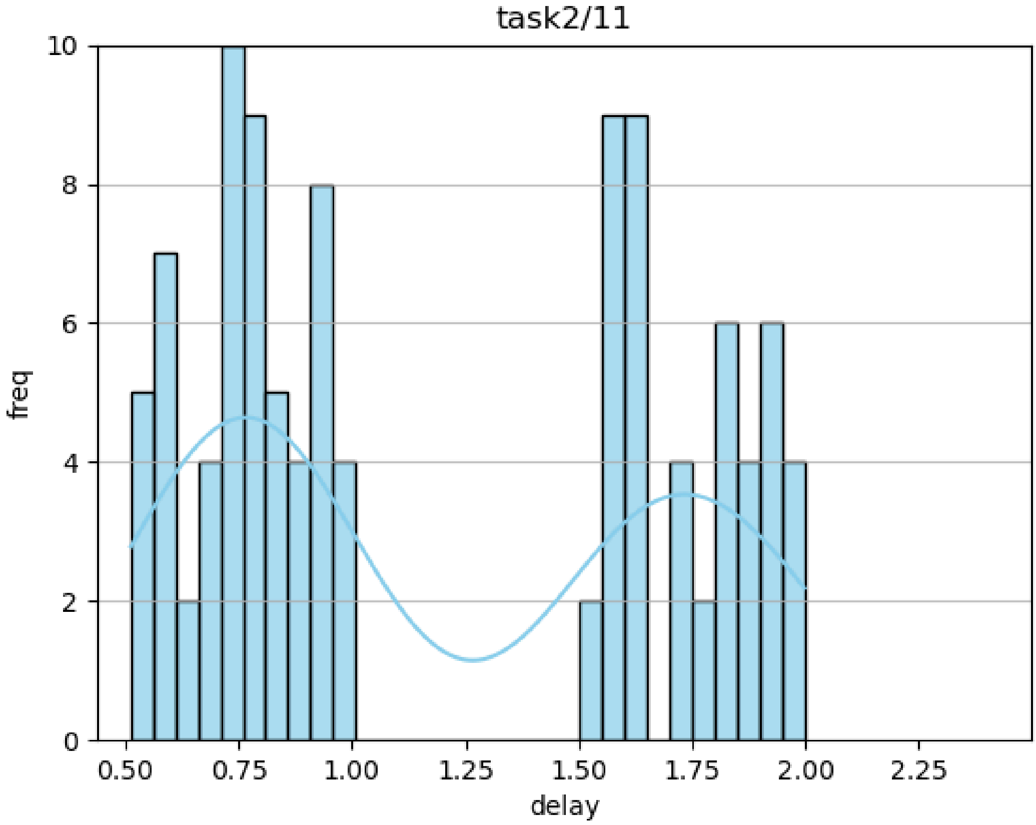

task2/6
freq
10
8
6
4
2
0
0.5
1.0
1.5
2.0
delay

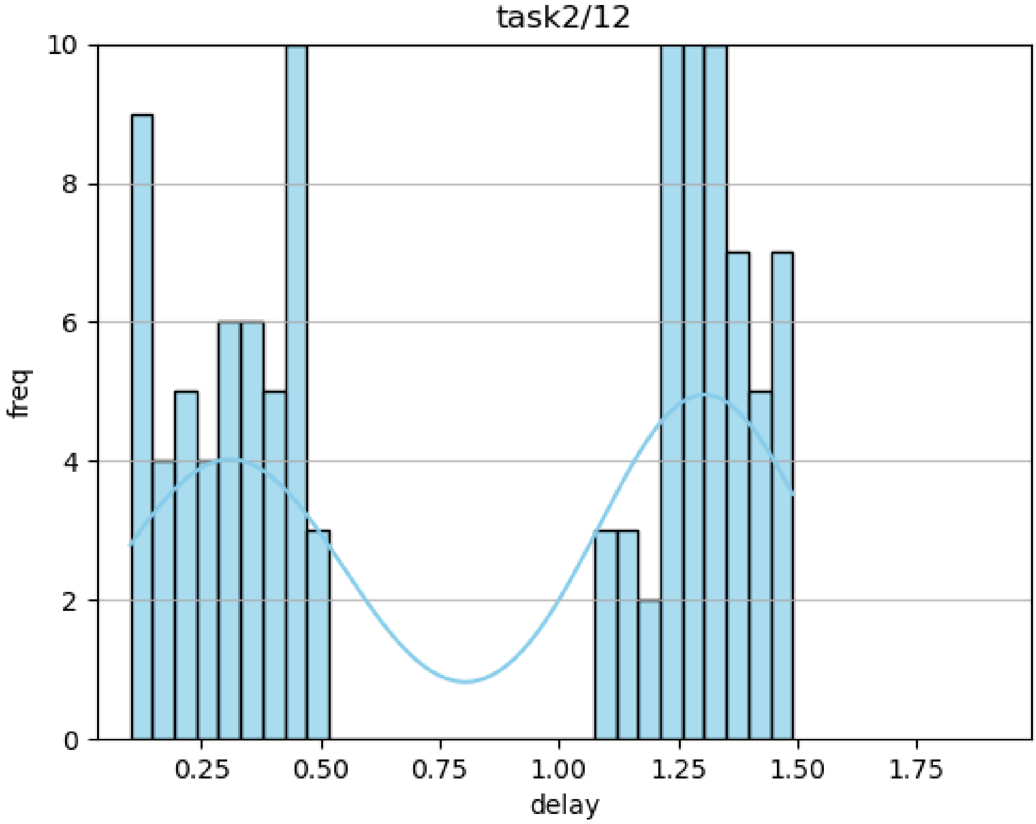

task2/7
freq
0
2
4
6
8
10
delay
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25


task2/8
freq
0
2
4
6
8
10
delay
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

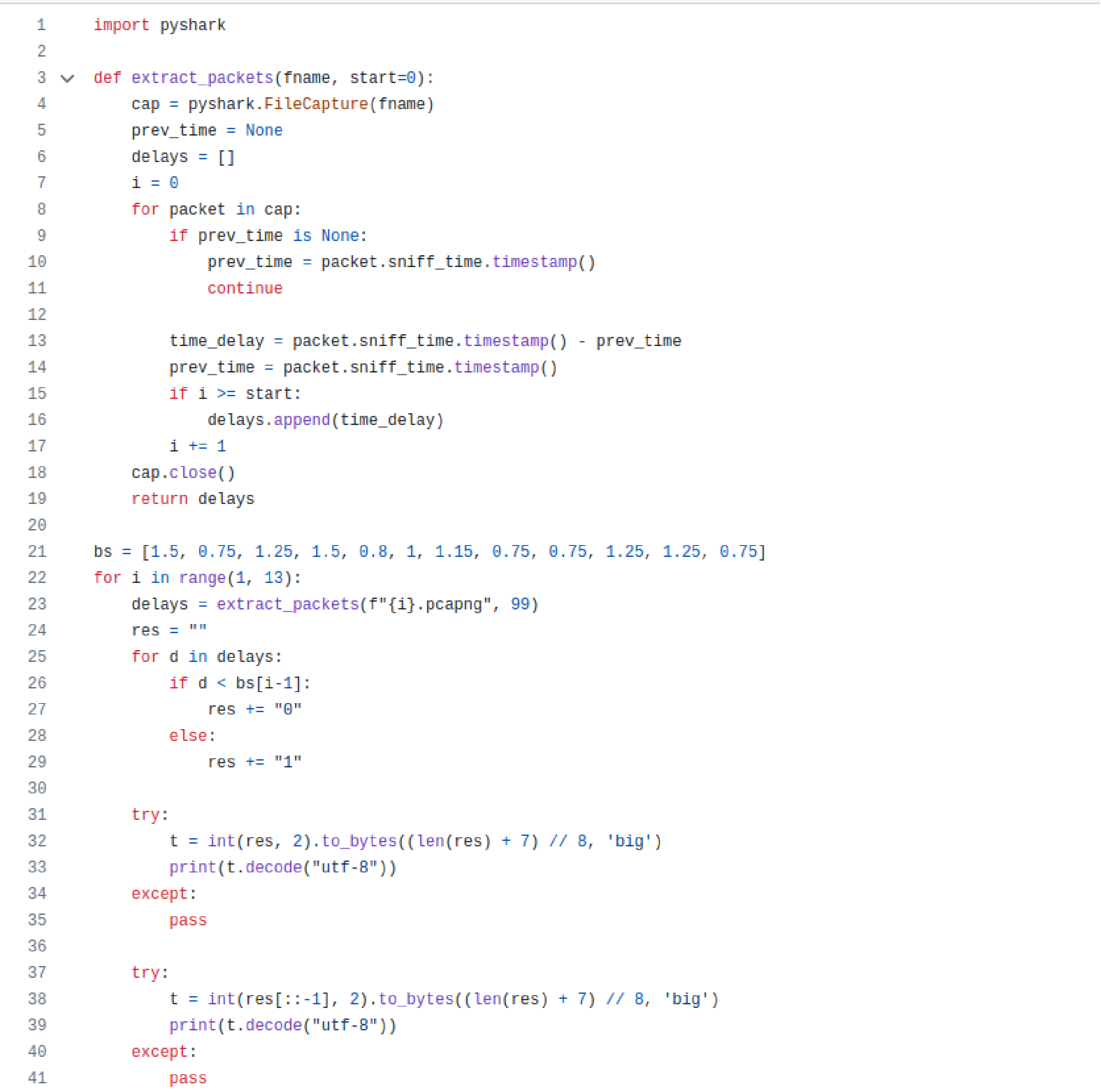

task2/9
freq
10
8
6
4
2
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
delay


task2/10
freq
10
8
6
4
2
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
delay

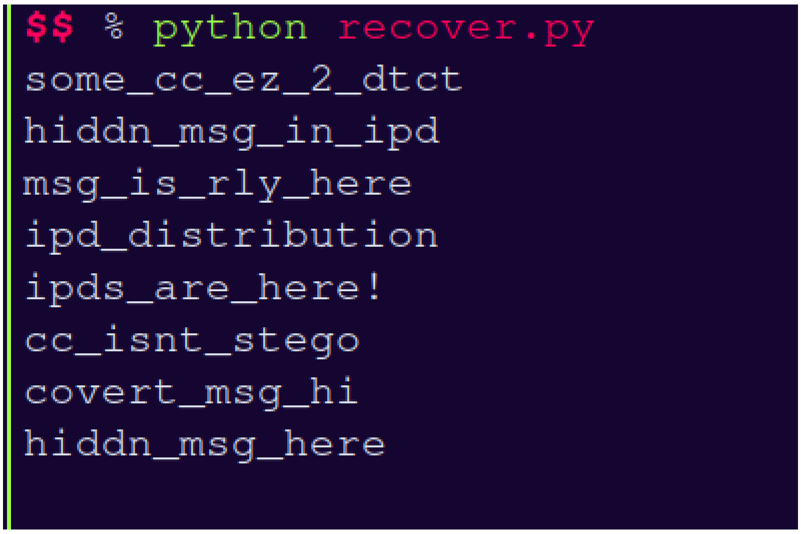

task2/11
freq
0
2
4
6
8
10
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
delay


task2/12
freq
0
2
4
6
8
10
0.25
0.50
0.75
1.00
1.25
1.50
1.75
delay

Тут уже явно видно разделение на два множества
Исходя из гистограмм я определил границу у восстановил сообщения следующим скриптом:



```python
import pyshark

def extract_packets(fname, start=0):
    cap = pyshark.FileCapture(fname)
    prev_time = None
    delays = []
    i = 0
    for packet in cap:
        if prev_time is None:
            prev_time = packet.sniff_time.timestamp()
            continue

        time_delay = packet.sniff_time.timestamp() - prev_time
        prev_time = packet.sniff_time.timestamp()
        if i >= start:
            delays.append(time_delay)
        i += 1
    cap.close()
    return delays

bs = [1.5, 0.75, 1.25, 1.5, 0.8, 1, 1.15, 0.75, 0.75, 1.25, 1.25, 0.75]
for i in range(1, 13):
    delays = extract_packets(f"{i}.pcapng", 99)
    res = ""
    for d in delays:
        if d < bs[i-1]:
            res += "0"
        else:
            res += "1"

    try:
        t = int(res, 2).to_bytes((len(res) + 7) // 8, 'big')
        print(t.decode("utf-8"))
    except:
        pass

    try:
        t = int(res[::-1], 2).to_bytes((len(res) + 7) // 8, 'big')
        print(t.decode("utf-8"))
    except:
        pass
```



```
$$ % python recover.py
some_cc_ez_2_dtct
hiddn_msg_in_ipd
msg_is_rly_here
ipd_distribution
ipds_are_here!
cc_isnt_stego
covert_msg_hi
hiddn_msg_here
```

Выводы

При малом объеме информации и значительной разнице между интервалами будет очень просто обнаружить скрытый канал, так как перебор будет небольшим, а разница заметной

Мне кажется, что можно применить тот же подход. Просто в распределении уже будет n явных частей, но вероятность надо будет считать уже по другой формуле. Например P = prod(1 - mu_i / max_i)