Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №3
Генерація мережевого трафіку за допомогою iperf

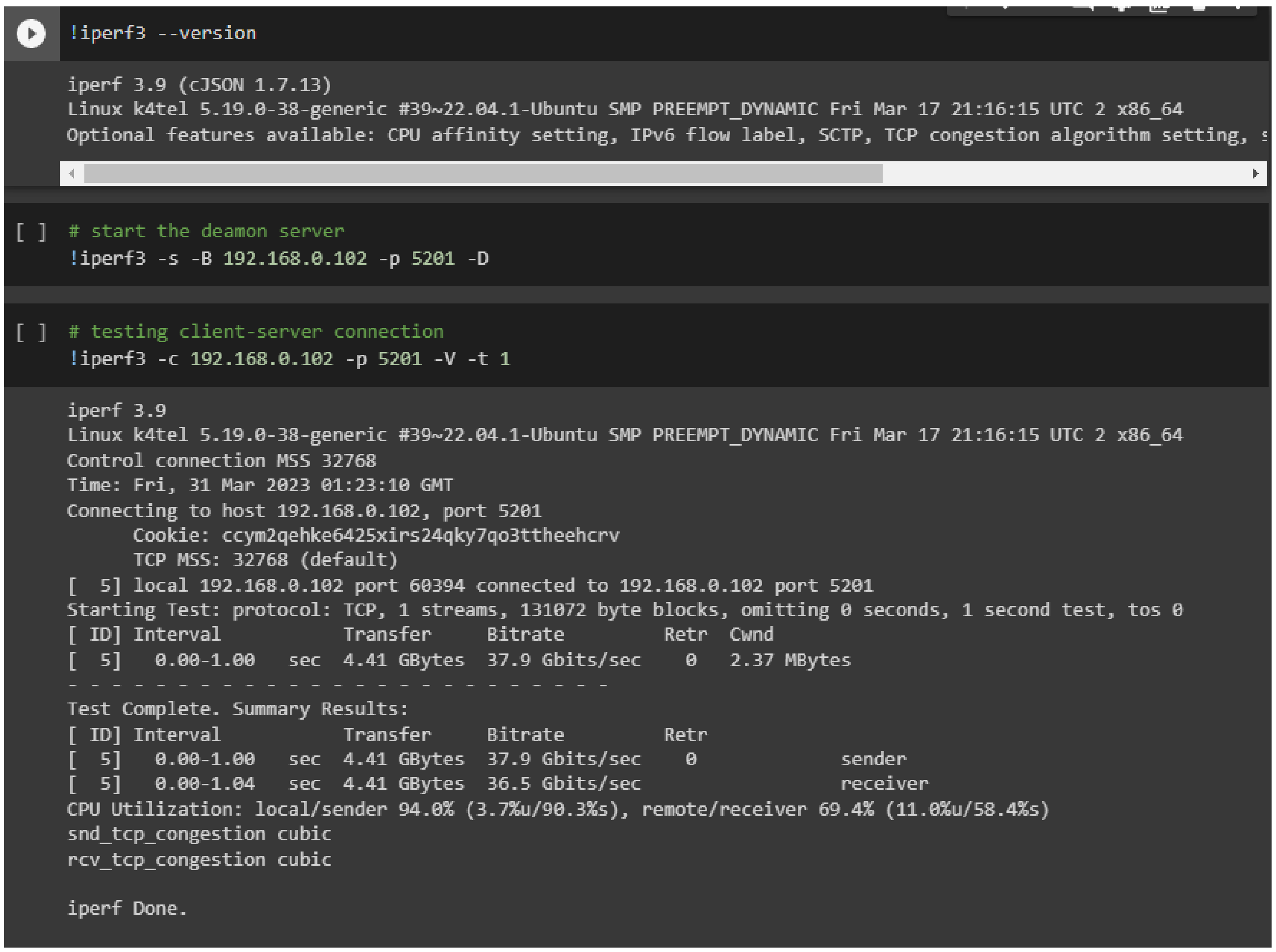
Виконала

студентка групи ІТ-91:	Перевірив:
Луцай Катерина	Каплунов А. В.

Київ 2023

Мета: написати функцію для підключення клієнта до сервера, використовуючи модуль subprocess..

Хід виконання роботи:



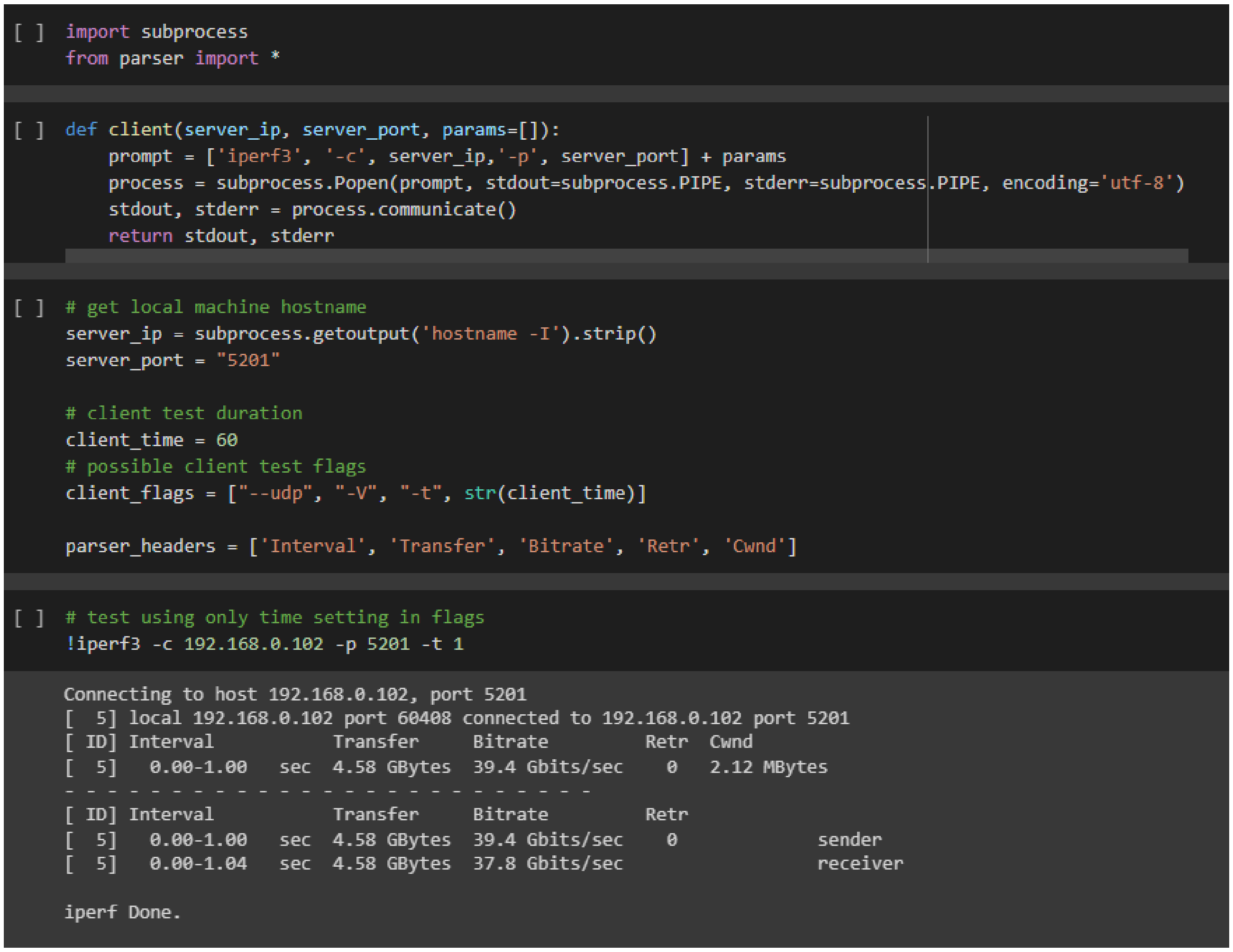
```
!iperf3 --version

iperf 3.9 (cJSON 1.7.13)
Linux k4tel 5.19.0-38-generic #39~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 21:16:15 UTC 2 x86_64
Optional features available: CPU affinity setting, IPv6 flow label, SCTP, TCP congestion algorithm setting, s
```

```
[ ] # start the deamon server
    !iperf3 -s -B 192.168.0.102 -p 5201 -D
```

```
[ ] # testing client-server connection
    !iperf3 -c 192.168.0.102 -p 5201 -V -t 1

iperf 3.9
Linux k4tel 5.19.0-38-generic #39~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 21:16:15 UTC 2 x86_64
Control connection MSS 32768
Time: Fri, 31 Mar 2023 01:23:10 GMT
Connecting to host 192.168.0.102, port 5201
      Cookie: ccym2qehke6425xirs24qky7qo3ttheehcrv
      TCP MSS: 32768 (default)
[  5] local 192.168.0.102 port 60394 connected to 192.168.0.102 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 1 second test, tos 0
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  4.41 GBytes  37.9 Gbits/sec    0   2.37 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-1.00   sec  4.41 GBytes  37.9 Gbits/sec    0             sender
[  5]   0.00-1.04   sec  4.41 GBytes  36.5 Gbits/sec                  receiver
CPU Utilization: local/sender 94.0% (3.7%u/90.3%s), remote/receiver 69.4% (11.0%u/58.4%s)
snd_tcp_congestion cubic
rcv_tcp_congestion cubic

iperf Done.
```

Parser.py

```
import re
import pandas as pd


def parser(output, samples=10, headers=None):
    # Define a regex pattern to match numerical values
    pattern = r'\d*\.\d+|\d+'
    result = []
    start_ind = 5

    for line in output.split('\n'):
        if not line:
            continue
        elif line.startswith('[ ID]'):
            if len(result) < samples:
                if headers is None:
                    start_ind = max(start_ind, line.rfind("]")) + 1
                    headers = [h.strip() for h in
line[start_ind:].split("  ") if h.strip() != ""]

                result = []
                print(f"Headers of the output data: {',
'.join(headers)}")
            else:
                break
        elif headers and line.startswith('[  '):
            # Extract all numerical values from the string using
the regex pattern
            num_matches = re.findall(pattern, line[start_ind:])
            if len(num_matches) < len(headers):
                continue
            else:
                # Convert the rest of the values to floats
                nums = [float(num) for num in num_matches]

                # Convert the first value to a float by subtracting
two integers
                nums[1] = abs(nums[1] - nums[0])
                # Slice only significant values
                nums = nums[1:]
                row_values = nums[:len(headers)] if len(nums) >
len(headers) else nums
                row_values[-1] = nums[-1]

                if len(result) < samples:
                    result.append(row_values)

        else:
            continue

    res_table = pd.DataFrame(result, columns=headers)
    return res_table
```



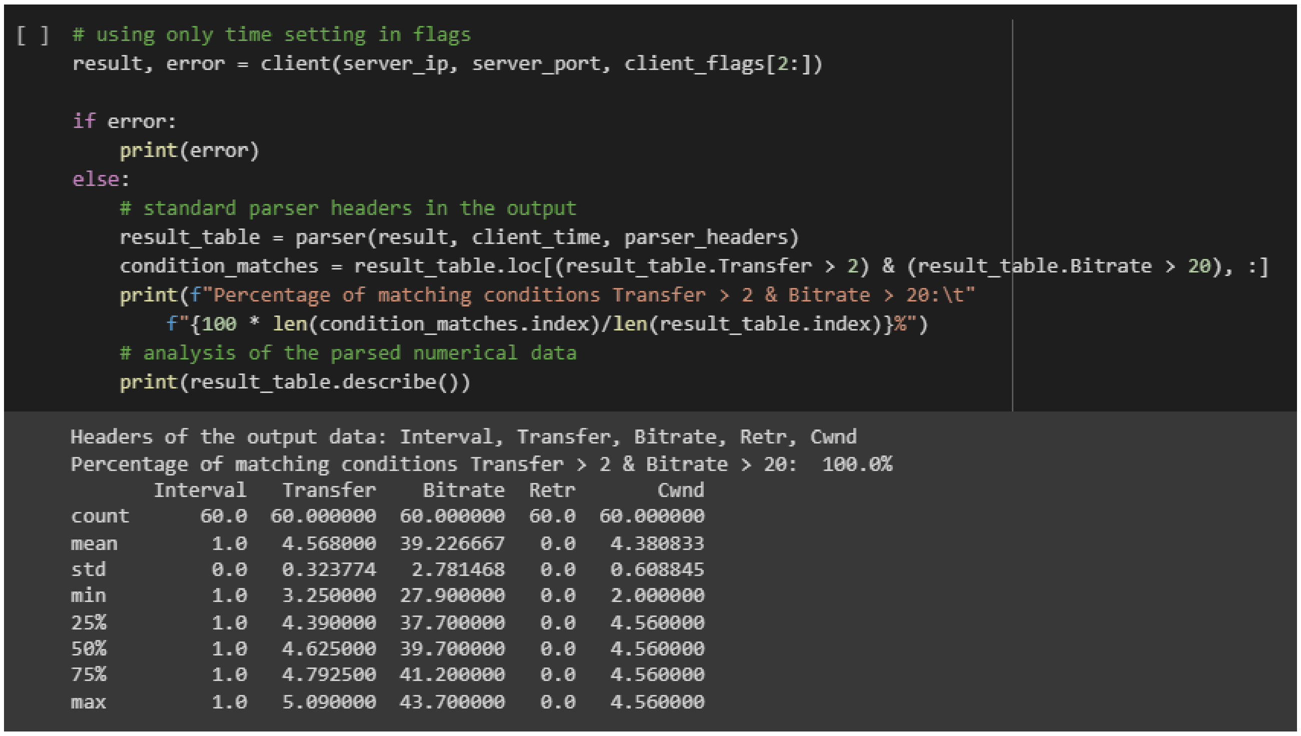
```
[ ] import subprocess
    from parser import *
```

```
[ ] def client(server_ip, server_port, params=[]):
        prompt = ['iperf3', '-c', server_ip,'-p', server_port] + params
        process = subprocess.Popen(prompt, stdout=subprocess.PIPE, stderr=subprocess.PIPE, encoding='utf-8')
        stdout, stderr = process.communicate()
        return stdout, stderr
```

```
[ ] # get local machine hostname
    server_ip = subprocess.getoutput('hostname -I').strip()
    server_port = "5201"

    # client test duration
    client_time = 60
    # possible client test flags
    client_flags = ["--udp", "-V", "-t", str(client_time)]

    parser_headers = ['Interval', 'Transfer', 'Bitrate', 'Retr', 'Cwnd']
```

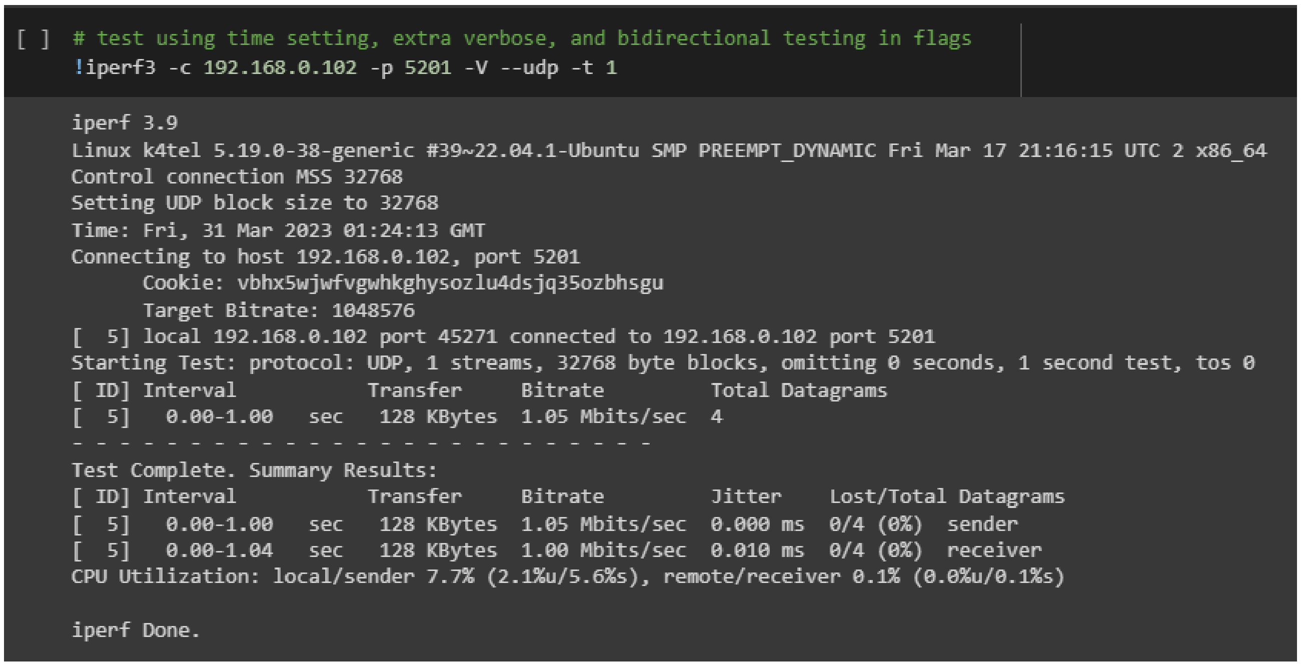
```
[ ] # test using only time setting in flags
    !iperf3 -c 192.168.0.102 -p 5201 -t 1
```

```
Connecting to host 192.168.0.102, port 5201
[  5] local 192.168.0.102 port 60408 connected to 192.168.0.102 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  4.58 GBytes  39.4 Gbits/sec    0   2.12 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-1.00   sec  4.58 GBytes  39.4 Gbits/sec    0             sender
[  5]   0.00-1.04   sec  4.58 GBytes  37.8 Gbits/sec                  receiver

iperf Done.
```

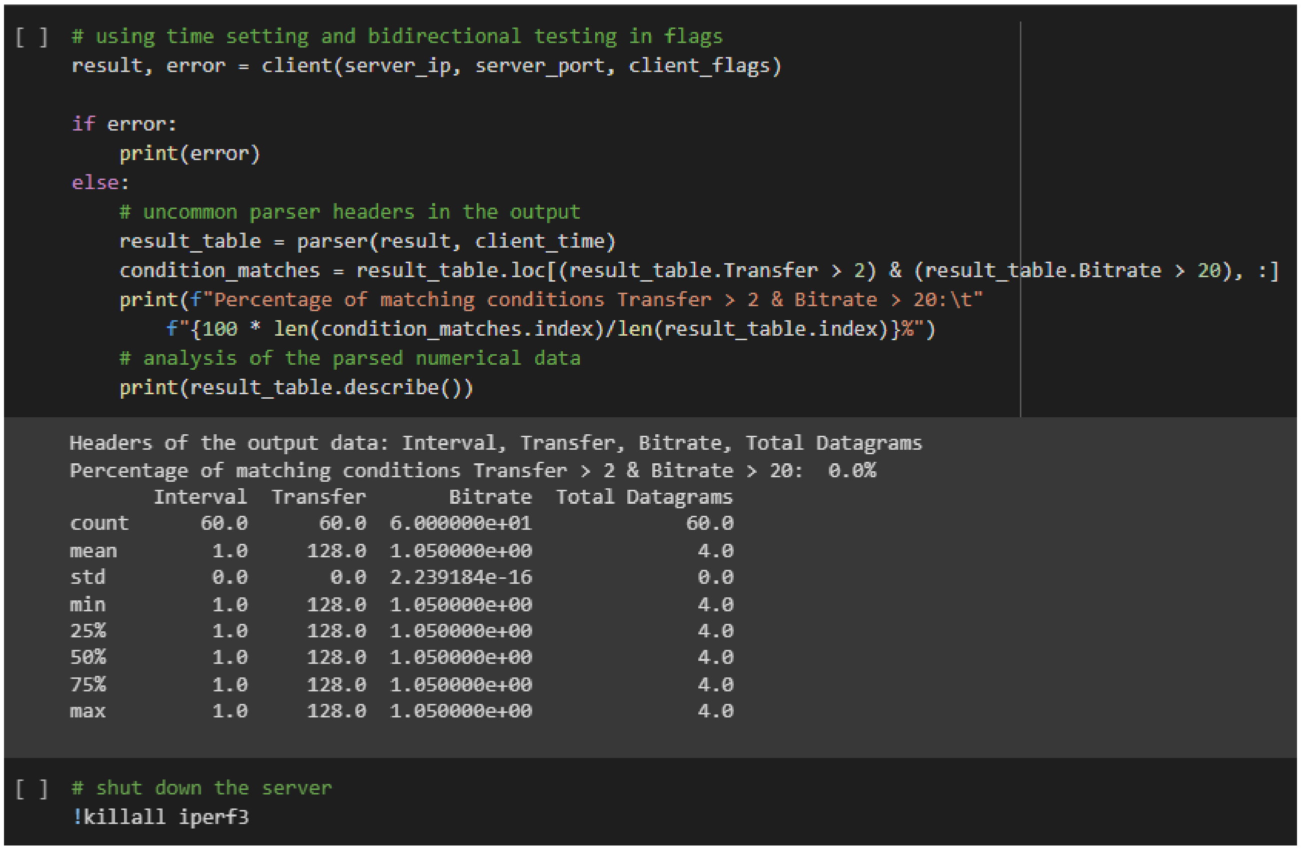


```
[ ] # using only time setting in flags
    result, error = client(server_ip, server_port, client_flags[2:])

    if error:
        print(error)
    else:
        # standard parser headers in the output
        result_table = parser(result, client_time, parser_headers)
        condition_matches = result_table.loc[(result_table.Transfer > 2) & (result_table.Bitrate > 20), :]
        print(f"Percentage of matching conditions Transfer > 2 & Bitrate > 20:\t"
            f"{100 * len(condition_matches.index)/len(result_table.index)}%")
        # analysis of the parsed numerical data
        print(result_table.describe())
```

```
Headers of the output data: Interval, Transfer, Bitrate, Retr, Cwnd
Percentage of matching conditions Transfer > 2 & Bitrate > 20:  100.0%
       Interval   Transfer    Bitrate  Retr       Cwnd
count      60.0  60.000000  60.000000  60.0  60.000000
mean        1.0   4.568000  39.226667   0.0   4.380833
std         0.0   0.323774   2.781468   0.0   0.608845
min         1.0   3.250000  27.900000   0.0   2.000000
25%         1.0   4.390000  37.700000   0.0   4.560000
50%         1.0   4.625000  39.700000   0.0   4.560000
75%         1.0   4.792500  41.200000   0.0   4.560000
max         1.0   5.090000  43.700000   0.0   4.560000
```



```
[ ] # test using time setting, extra verbose, and bidirectional testing in flags
    !iperf3 -c 192.168.0.102 -p 5201 -V --udp -t 1

    iperf 3.9
    Linux k4tel 5.19.0-38-generic #39~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 21:16:15 UTC 2 x86_64
    Control connection MSS 32768
    Setting UDP block size to 32768
    Time: Fri, 31 Mar 2023 01:24:13 GMT
    Connecting to host 192.168.0.102, port 5201
          Cookie: vbhx5wjwfvgwhkghysozlu4dsjq35ozbhsgu
          Target Bitrate: 1048576
    [  5] local 192.168.0.102 port 45271 connected to 192.168.0.102 port 5201
    Starting Test: protocol: UDP, 1 streams, 32768 byte blocks, omitting 0 seconds, 1 second test, tos 0
    [ ID] Interval           Transfer     Bitrate         Total Datagrams
    [  5]   0.00-1.00   sec   128 KBytes  1.05 Mbits/sec  4
    - - - - - - - - - - - - - - - - - - - - - - - - -
    Test Complete. Summary Results:
    [ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
    [  5]   0.00-1.00   sec   128 KBytes  1.05 Mbits/sec  0.000 ms  0/4 (0%)  sender
    [  5]   0.00-1.04   sec   128 KBytes  1.00 Mbits/sec  0.010 ms  0/4 (0%)  receiver
    CPU Utilization: local/sender 7.7% (2.1%u/5.6%s), remote/receiver 0.1% (0.0%u/0.1%s)

    iperf Done.
```



```
[ ] # using time setting and bidirectional testing in flags
    result, error = client(server_ip, server_port, client_flags)

    if error:
        print(error)
    else:
        # uncommon parser headers in the output
        result_table = parser(result, client_time)
        condition_matches = result_table.loc[(result_table.Transfer > 2) & (result_table.Bitrate > 20), :]
        print(f"Percentage of matching conditions Transfer > 2 & Bitrate > 20:\t"
            f"{100 * len(condition_matches.index)/len(result_table.index)}%")
        # analysis of the parsed numerical data
        print(result_table.describe())

    Headers of the output data: Interval, Transfer, Bitrate, Total Datagrams
    Percentage of matching conditions Transfer > 2 & Bitrate > 20:  0.0%
           Interval  Transfer       Bitrate  Total Datagrams
    count      60.0      60.0  6.000000e+01             60.0
    mean        1.0     128.0  1.050000e+00              4.0
    std         0.0       0.0  2.239184e-16              0.0
    min         1.0     128.0  1.050000e+00              4.0
    25%         1.0     128.0  1.050000e+00              4.0
    50%         1.0     128.0  1.050000e+00              4.0
    75%         1.0     128.0  1.050000e+00              4.0
    max         1.0     128.0  1.050000e+00              4.0

[ ] # shut down the server
    !killall iperf3
```

Код у git-hub: https://github.com/K4TEL/rt-sys.git

Висновки: було написано функціонал для підключення клієнта до сервера, використовуючи Python модуль subprocess, згенеровано мережевий трафік за допомогою iperf..