

# **Лабораторная работа №3**

**Измерение и тестирование пропускной способности сети.  
Воспроизводимый эксперимент**

Ланцова Яна Игоревна

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>5</b>  |
| <b>2</b> | <b>Задачи</b>                         | <b>6</b>  |
| <b>3</b> | <b>Выполнение лабораторной работы</b> | <b>7</b>  |
| <b>4</b> | <b>Выводы</b>                         | <b>15</b> |

# Список иллюстраций

|      |   |    |
|------|---|----|
| 3.1  | Создание подкаталога . . . . .                                | 7  |
| 3.2  | Копирование файла emptynet.py . . . . .                       | 7  |
| 3.3  | Содержание файла lab_iperf3_topo.py . . . . .                 | 8  |
| 3.4  | Создание топологии и ее основные параметры . . . . .          | 9  |
| 3.5  | Вывод внесенных изменений . . . . .                           | 10 |
| 3.6  | Изменение скрипта lab_iperf3_topo.py . . . . .                | 10 |
| 3.7  | Проверка работы внесенных изменений . . . . .                 | 10 |
| 3.8  | Настройка параметров производительности . . . . .             | 11 |
| 3.9  | Запуск скрипта с настройкой параметров производительности . . | 12 |
| 3.10 | Создание копии скрипта lab_iperf3_topo2.py . . . . .          | 12 |
| 3.11 | Изменения кода в скрипте lab_iperf3.py . . . . .              | 13 |
| 3.12 | Запуск скрипта lab_iperf3.py . . . . .                        | 13 |
| 3.13 | Создание Makefile . . . . .                                   | 14 |
| 3.14 | Проверка работы Makefile . . . . .                            | 14 |

## **Список таблиц**

# 1 Цель работы

Основной целью работы является знакомство с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получение навыков проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.

## 2 Задачи

1. Воспроизвести посредством API Mininet эксперименты по измерению пропускной способности с помощью iPerf3.
2. Построить графики по проведённому эксперименту.

### 3 Выполнение лабораторной работы

С помощью API Mininet создадим простейшую топологию сети, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8. В каталоге /work/lab\_iperf3 для работы над проектом создадим подкаталог lab\_iperf3\_topo и скопируем в него файл с примером скрипта mininet/examples/emphynet.py, описывающего стандартную простую топологию сети mininet (рис. 3.1; 3.2).

```
mininet@mininet-vm:~$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ ls
iperf.csv iperf_results.json results
mininet@mininet-vm:~/work/lab_iperf3$ mkdir lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3$ cd ~/work/lab_iperf3/lab_iperf3_topo
```

Рис. 3.1: Создание подкаталога

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ ls
emphynet.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emphynet.py lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ ls
lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$
```

Рис. 3.2: Копирование файла emphynet.py

Изучим содержание скрипта lab\_iperf3\_topo.py. В нем написан скрипт по созданию простейшей топологии из двух хостов h1 и h2, а также коммутатора s3 и контроллера c0. В начале файла видим импорт необходимых библиотек (рис. 3.3).

```

from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def emptyNet():

    "Create an empty network and add nodes to it."

    net = Mininet( controller=Controller, waitConnected=True )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1' )
    h2 = net.addHost( 'h2', ip='10.0.0.2' )

    info( '*** Adding switch\n' )
    s3 = net.addSwitch( 's3' )

    info( '*** Creating links\n' )
    net.addLink( h1, s3 )
    net.addLink( h2, s3 )

    info( '*** Starting network\n' )
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()

```

Рис. 3.3: Содержание файла lab\_iperf3\_topo.py

Основные элементы:

- `addSwitch()`: добавляет коммутатор в топологию и возвращает имя коммутатора;
- `addHost()`: добавляет хост в топологию и возвращает имя хоста;
- `addLink()`: добавляет двунаправленную ссылку в топологию (и возвращает ключ ссылки; ссылки в Mininet являются двунаправленными, если не указано иное);
- `Mininet`: основной класс для создания и управления сетью;
- `start()`: запускает сеть;
- `pingAll()`: проверяет подключение, пытаясь заставить все узлы пинговать друг друга;
- `stop()`: останавливает сеть;
- `net.hosts`: все хосты в сети;
- `dumpNodeConnections()`: сбрасывает подключения к/от набора узлов;



- `setLogLevel('info' | 'debug' | 'output')`: устанавливает уровень вывода Mininet по умолчанию; рекомендуется `info`.

Запустим скрипт создания топологии `lab_iperf3_topo.py` и посмотрим ее основные параметры (рис. 3.4).

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s3 ...
*** Waiting for switches to connect
s3
*** Running CLI
*** Starting CLI:
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
s3 lo: s3-eth1:h1-eth0 s3-eth2:h2-eth0
c0
mininet> links
h1-eth0<->s3-eth1 (OK OK)
h2-eth0<->s3-eth2 (OK OK)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=8981>
<Host h2: h2-eth0:10.0.0.2 pid=8985>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=8990>
<Controller c0: 127.0.0.1:6653 pid=8974>
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
```

Рис. 3.4: Создание топологии и ее основные параметры

Внесем в скрипт `lab_iperf3_topo.py` изменение, позволяющее вывести на экран информацию о хосте `h1`, а именно имя хоста, его IP-адрес, MAC-адрес. Для этого после строки, задающей старт работы сети, добавим строку `print("Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC())`. Посмотрим вывод (рис. 3.5).

```

mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ ls
lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s3 ...
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address ea:10:73:56:8a:e3
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 3.5: Вывод внесенных изменений

Внесем в скрипт lab\_iperf3\_topo.py изменение, позволяющее вывести на экран информацию обоих хостов сети, а именно имя хоста, его IP-адрес, MAC-адрес(рис. 3.6).

```

info( '*** Starting network\n')
net.start()

print("Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC())
print("Host", h2.name, "has IP address", h2.IP(), "and MAC address", h2.MAC())

info( '*** Running CLI\n' )
CLI( net )

```

Рис. 3.6: Изменение скрипта lab\_iperf3\_topo.py

Здесь:

- IP() возвращает IP-адрес хоста или определенного интерфейса;
- MAC() возвращает MAC-адрес хоста или определенного интерфейса.

Проверим корректность отработки изменённого скрипта (рис. 3.7).

```

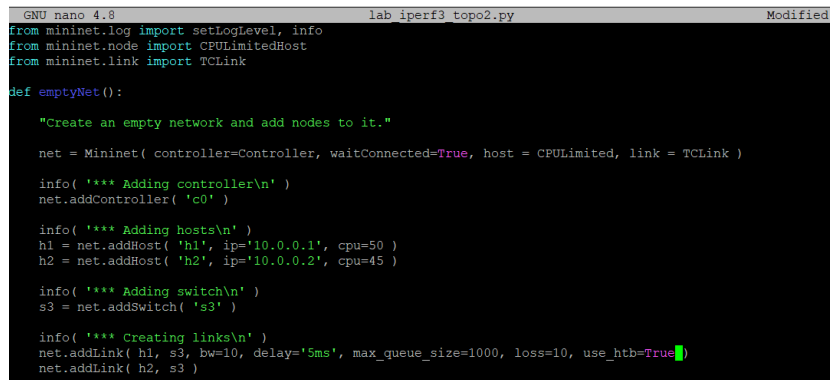
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s3 ...
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address 2a:5d:65:1b:a7:62
Host h2 has IP address 10.0.0.2 and MAC address 52:63:6f:6a:17:3c
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 3.7: Проверка работы внесенных изменений

Действительно, нам вывелась информация об IP и mac адресах хостов.

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов `CPUimitedHost` и `TCLink`. Добавим в скрипт настройки параметров производительности (рис. 3.8).



```
GNU nano 4.8 lab_iperf3_topo2.py Modified
from mininet.log import setLogLevel, info
from mininet.node import CPUimitedHost
from mininet.link import TCLink

def emptyNet():
    "Create an empty network and add nodes to it."

    net = Mininet( controller=Controller, waitConnected=True, host = CPUimited, link = TCLink )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1', cpu=50 )
    h2 = net.addHost( 'h2', ip='10.0.0.2', cpu=45 )

    info( '*** Adding switch\n' )
    s3 = net.addSwitch( 's3' )

    info( '*** Creating links\n' )
    net.addLink( h1, s3, bw=10, delay='5ms', max_queue_size=1000, loss=10, use_htb=True )
    net.addLink( h2, s3 )
```

Рис. 3.8: Настройка параметров производительности

В скрипте `lab_iperf3_topo2.py` изменим строку описания сети, указав на использование ограничения производительности и изоляции. Также изменим функцию задания параметров виртуального хоста `h1`, указав, что ему будет выделено 50% от общих ресурсов процессора системы. Аналогичным образом для хоста `h2` зададим долю выделения ресурсов процессора в 45%. В скрипте изменим функцию параметров соединения между хостом `h1` и коммутатором `s3`. А именно добавим двунаправленный канал с характеристиками пропускной способности, задержки и потерь:

- параметр пропускной способности (`bw`) выражается числом в Мбит;
- задержка (`delay`) выражается в виде строки с заданными единицами измерения (например, `5ms`, `100us`, `1s`);
- потери (`loss`) выражаются в процентах (от 0 до 100);
- параметр максимального значения очереди (`max_queue_size`) выражается в пакетах;
- параметр `use_htb` указывает на использование ограничителя интенсивности входящего потока Hierarchical Token Bucket (HTB)

Запустим на отработку сначала скрипт lab\_iperf3\_topo2.py, затем lab\_iperf3\_topo.py и сравним результат(рис. 3.9). Увидим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo2.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(10.00Mbit 5ms delay 10.00000% loss) (10.00Mbit 5ms delay 10.00000% loss) *** Starting network
*** Configuring hosts
h1 (cfs 5000000/1000000us) h2 (cfs 4500000/1000000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (10.00Mbit 5ms delay 10.00000% loss) ... (10.00Mbit 5ms delay 10.00000% loss)
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address c2:cc:76:95:d0:6a
Host h2 has IP address 10.0.0.2 and MAC address c2:bc:8e:2f:a6:c6
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 3.9: Запуск скрипта с настройкой параметров производительности

Построим графики по проводимому эксперименту. Сделаем копию скрипта lab\_iperf3\_topo2.py и поместим его в подкаталог iperf(рис. 3.10).

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cp lab_iperf3_topo.py lab_iperf3_topo2.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo2.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cp lab_iperf3_topo2.py lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mkdir -p ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv ~/work/lab_iperf3/lab_iperf3_topo/lab_iperf3.py ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cd ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -l
total 4
-rwxrwxr-x 1 mininet mininet 1342 Sep 20 09:01 lab_iperf3.py
```

Рис. 3.10: Создание копии скрипта lab\_iperf3\_topo2.py

Изменим код в скрипте lab\_iperf3.py так, чтобы(рис. 3.11):

- на хостах не было ограничения по использованию ресурсов процессора;
- каналы между хостами и коммутатором были по 100 Мбит/с с задержкой 75 мс, без потерь, без использования ограничителей пропускной способности и максимального размера очереди.
- После функции старта сети опишем запуск на хосте h2 сервера iPerf3, а на хосте h1 запуск с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл, прокомментируем строки, отвечающие за запуск CLI-интерфейса:

```

GNU nano 4.8                                lab_iperf3.py                                Modified
import time
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.node import CPULimitedHost
from mininet.link import TCLink

def emptyNet():
    "Create an empty network and add nodes to it."

    net = Mininet( controller=Controller, waitConnected=True, host = CPULimitedHost, link = TCLink )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1' )
    h2 = net.addHost( 'h2', ip='10.0.0.2' )

    info( '*** Adding switch\n' )
    s3 = net.addSwitch( 's3' )

    info( '*** Creating links\n' )
    net.addLink( h1, s3, bw=100, delay='75ms' )
    net.addLink( h2, s3, bw=100, delay='75ms' )

    info( '*** Starting network\n' )
    net.start()

    info( '*** Traffic generation\n' )
    h2.cmdPrint( 'iperf3 -s -D -l1' )
    time.sleep(10)
    h1.cmdPrint( 'iperf3 -c', h2.IP(), '-J > iperf_result.json' )

    print("Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC())
    print("Host", h2.name, "has IP address", h2.IP(), "and MAC address", h2.MAC())

    # info( '*** Running CLI\n' )
    # CLI( net )

```

Рис. 3.11: Изменения кода в скрипте lab\_iperf3.py

Запустим на отработку скрипт lab\_iperf3.py (рис. 3.12).

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ sudo python lab_iperf3.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) *** Starting
network
*** Configuring hosts
h1 (cfs -l/1000000us) h2 (cfs -l/1000000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) ... (100.00Mbit 75ms delay) (100.00Mbit 75ms delay)
*** Waiting for switches to connect
s3
*** Traffic generation
*** h2 : ('iperf3 -s -D -l1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address b6:c9:e2:88:c4:e8
Host h2 has IP address 10.0.0.2 and MAC address 96:5c:00:76:03:95
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch Makefile

```

Рис. 3.12: Запуск скрипта lab\_iperf3.py

Построим графики из получившегося JSON-файла. Создадим Makefile для проведения всего эксперимента. В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 3.13).

```
GNU nano 4.8                                Makefile                                Modified
all: iperf_result.json plot

iperf_result.json:
    sudo python lab_iperf3.py

plot: iperf_result.json
    plot_iperf.sh iperf_result.json

clean:
    -rm -f *.json *.csv
    -rm -rf results
```

Рис. 3.13: Создание Makefile

Проверьте корректность отработки Makefile (рис. 3.14):

```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make clean
rm -f *.json *.csv
rm -rf results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make
sudo python lab_iperf3.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) *** Starting
network
*** Configuring hosts
h1 (cfs -l/1000000us) h2 (cfs -l/1000000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) ... (100.00Mbit 75ms delay) (100.00Mbit 75ms delay)
*** Waiting for switches to connect
s3
*** Traffic generation
*** h2 : ('iperf3 -s -D -l',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address 76:85:17:df:20:5b
Host h2 has IP address 10.0.0.2 and MAC address 7a:60:96:06:09:3a
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
...
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Рис. 3.14: Проверка работы Makefile

## 4 Выводы

В результате выполнения данной лабораторной работы я познакомилась с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получила навыки проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.