

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Моделирование сетей передачи данных

Студент: Абрамян А. А.

Группа: НПИбд-01-20

МОСКВА

2023 г.

Цель работы

Целью данной 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.

Описание процесса выполнения работы

Постановка задачи

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

Порядок выполнения работы

Эксперименты по измерению пропускной способности с помощью iPerf3.

С помощью API Mininet создайте 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 создайте подкаталог lab_iperf3_topo и скопируйте в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet:

```
cd ~/work/lab_iperf3
```

```
mkdir lab_iperf3_topo
```

```
cd ~/work/lab_iperf3/lab_iperf3_topo
```

```
cp ~/mininet/examples/emptynet.py ~/work/lab_iperf3/lab_iperf3_topo
```

```
mv emptynet.py lab_iperf3_topo.py
```

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
login as: mininet
mininet@192.168.56.5's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Sat Dec  2 05:13:09 2023
mininet@mininet-vm:~$ ls
mininet  oflops  oftest  openflow  pox  work
mininet@mininet-vm:~$ cd ~/work/lab_iperf3/
mininet@mininet-vm:~/work/lab_iperf3$ ls
iperf.csv  iperf_result.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/
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cp ~/mininet/examples/emphynet.py ~/work/lab_iperf3/lab_iperf3_topo
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$
```

Изучите содержание скрипта lab_iperf3_topo.py:

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
#!/usr/bin/env python

"""
This example shows how to create an empty Mininet object
(without a topology object) and add nodes to it manually.
"""

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\n' )
    net.stop()

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

Основные элементы:

– addSwitch(): добавляет коммутатор в топологию и возвращает имя коммутатора;

- `ddHost()`: 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`.

– Запустите скрипт создания топологии `lab_iperf3_topo.py`:

`sudo python 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
*** Running CLI
*** Starting CLI:
mininet>
```

После отработки скрипта посмотрите элементы топологии и завершите работу mininet:

- 1 mininet> net
- 2 mininet> links
- 3 mininet> dump
- 4 mininet> exit

```

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=831>
<Host h2: h2-eth0:10.0.0.2 pid=835>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=840>
<Controller c0: 127.0.0.1:6653 pid=824>
mininet> exit
*** 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/lab_iperf3_topo$

```

Внесите 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, добавьте строку:

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

```

"""
This example shows how to create an empty Mininet object
(without a topology object) and add nodes to it manually.
"""

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()
    print( "Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC() )
    info( '*** Running CLI\n' )
    CLI( net )

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

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$

```

Здесь:

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

Проверьте корректность обработки изменённого скрипта.

```
Host h1 has IP address 10.0.0.1 and MAC address 1e:36:78:8d:62:5e
*** Running CLI
*** Starting CLI:
mininet> █
```

Измените скрипт lab_iperf3_topo.py так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети. Проверьте корректность обработки изменённого скрипта.

```
Host h1 has IP address 10.0.0.1 and MAC address 52:30:b1:43:04:a4
Host h2 has IP address 10.0.0.2 and MAC address 2a:d9:2b:f4:59:29
*** Running CLI
*** Starting CLI:
mininet> █
```

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов CPULimitedHost и TCLink. Добавьте в скрипт настройки параметров производительности:

- Сделайте копию скрипта lab_iperf3_topo.py:

cp lab_iperf3_topo.py lab_iperf3_topo2.py

```
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$ █
```

В начале скрипта lab_iperf3_topo2.py добавьте записи об импорте классов CPULimitedHost и TCLink:

...

from mininet.node import CPULimitedHost

from mininet.link import TCLink

```
mininet@mininet: /work/lab_iperf3/lab_iperf3_topo
GNU nano 4.8 lab_iperf3_topo2.py
#!/usr/bin/env python

"""
This example shows how to create an empty Mininet object
(without a topology object) and add nodes to it manually.
"""

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 )

    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()
    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 )

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify     ^C Cur Pos     M-U Undo
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell    ^_ Go To Line   M-E Redo
```

В скрипте lab_iperf3_topo2.py измените строку описания сети, указав на использование ограничения производительности и изоляции:

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

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

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

В скрипте lab_iperf3_topo2.py измените функцию задания параметров виртуального хоста h1, указав, что ему будет выделено 50% от общих ресурсов процессора системы:

...

```
h1 = net.addHost( 'h1', ip='10.0.0.1', cpu=50 )
```

```
INFO: ... Adding hosts() []  
h1 = net.addHost( 'h1', ip='10.0.0.1', cpu=50 )
```

Аналогичным образом для хоста h2 задайте долю выделения ресурсов процессора в 45%.

```
INFO: ... Adding hosts() []  
h1 = net.addHost( 'h1', ip='10.0.0.1', cpu=50 )  
h2 = net.addHost( 'h2', ip='10.0.0.2', cpu=45 )
```

В скрипте lab_iperf3_topo2.py измените функцию параметров соединения между хостом h1 и коммутатором s3:

...

```
net.addLink( h1, s3, bw=10, delay='5ms', max_queue_size=1000, loss=10,  
use_htb=True )
```

```
INFO: ... Creating links() []  
net.addLink( h1, s3, bw=10, delay='5ms', max_queue_size=1000, loss=10, use_htb=True ) []  
net.addLink( h2, s3 )
```

Здесь добавляется 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).
- Запустите на отработку сначала скрипт lab_iperf3_topo2.py, затем lab_iperf3_topo.py и сравните результат.


```

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 5e:dc:9a:47:04:5e
Host h2 has IP address 10.0.0.2 and MAC address 26:dd:aa:8d:dd:38
*** Running CLI
*** Starting CLI:
mininet> 

```

```

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 c6:43:3c:2f:af:45
Host h2 has IP address 10.0.0.2 and MAC address 4e:d0:d9:61:d9:19
*** Running CLI
*** Starting CLI:
mininet> 

```

В первом случае хосту h1 выделено 50% от общих ресурсов процессора системы, хосту h2 – 45%;

Также в первом случае двунаправленный канал с характеристиками пропускной способности, задержки и потерь: 10.00 Mbit, 5ms, 10% соответственно.

Графики по проведённому эксперименту

Постройте графики по проводимому эксперименту:

– Сделайте копию скрипта lab_iperf3_topo2.py и поместите его в подкаталог iperf3:

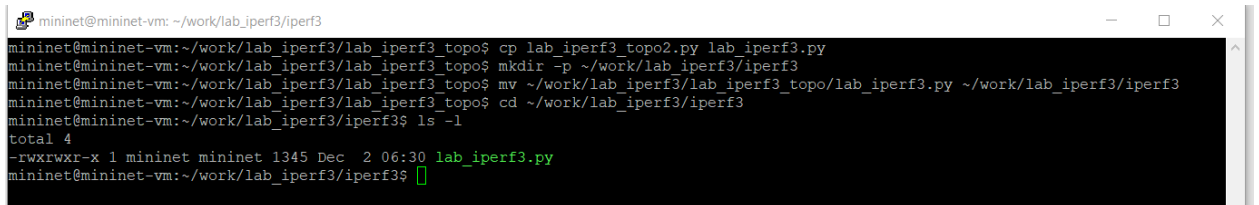
```
cp lab_iperf3_topo2.py lab_iperf3.py
```

```
mkdir -p ~/work/lab_iperf3/iperf3
```

```
mv ~/work/lab_iperf3/lab_iperf3_topo/lab_iperf3.py ~/work/lab_iperf3/iperf3
```

```
cd ~/work/lab_iperf3/iperf3
```

```
ls -l
```



```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
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 1345 Dec  2 06:30 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

В начале скрипта lab_iperf3.py добавьте запись

```
import time
```



```
from mininet.link import TCLink
import time
def emptyNet():
```

– Измените код в скрипте lab_iperf3.py так, чтобы:

– 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.



```
INFO: *** Adding hosts ***
h1 = net.addHost( 'h1', ip='10.0.0.1' )
h2 = net.addHost( 'h2', ip='10.0.0.2' )

net.addLink( h1, s3, bw=100, delay='75ms' )
net.addLink( h2, s3, bw=100, delay='75ms' )
```

После функции старта сети опишите запуск на хосте h2 сервера iPerf3, а на хосте h1 запуск с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл, прокомментируйте строки, отвечающие за запуск CLI-интерфейса:

...

```
net.start()
```

```
info( '*** Starting network\n')
```

```
info( '*** Traffic generation\n')
```

```
h2.cmdPrint( 'iperf3 -s -D -1' )
```

```
time.sleep(10) # Wait 10 seconds for servers to start
```

```
h1.cmdPrint( 'iperf3 -c', h2.IP(), '-J > iperf_result.json' )
```

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

```
# CLI( net )
```

...

```
"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 -1' )
time.sleep(10) # Wait 10 seconds for servers to start
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 )

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line    M-E Redo
```

– В отчёте поясните синтаксис вызова iPerf3, заданный в скрипте.

Запустили на втором хосте генератор трафика в серверном режиме (отсоединяется от терминала и не держит его, после одного соединения отсоединяется).

Задержка 10 секунд, чтобы сервер успел подняться.

Когда клиент отработает сеть останавливается и мы покидаем рабочий интерфейс.

Запустите на отработку скрипт lab_iperf3.py:

```
sudo python lab_iperf3.py
```

```
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 -1/1000000us) h2 (cfs -1/1000000us)
*** 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 -1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address 9a:0c:45:fd:6c:42
Host h2 has IP address 10.0.0.2 and MAC address 6a:d9:cd:ca:57:4a
*** 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$
```

Постройте графики из получившегося JSON-файла:

```
plot_iperf.sh iperf_result.json
```

```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls
iperf.csv iperf_result.json lab_iperf3.py results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Создайте Makefile для проведения всего эксперимента:

```
touch Makefile
```

В Makefile пропишите запуск скрипта эксперимента, построение графиков и

очистку каталога от результатов:

```
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
```

```
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
```

Проверьте корректность отработки Makefile:

```
make clean
```

```
make
```

```

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
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us)
*** 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 -1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address 9e:13:e9:32:58:f7
Host h2 has IP address 10.0.0.2 and MAC address 76:b4:8f:ba:81:c2
*** 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$ █

```

Завершите соединение с виртуальной машиной mininet и выключите её.

Вывод

Итогом лабораторной работы стало знакомство с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получение навыков проведения интерактивного эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.