

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

дисциплина: Моделирование сетей передачи данных

Студент: Быстров Глеб

Группа: НПИбд-01-20

МОСКВА

2023 г.

Цель работы

В данной лабораторной работе мне будет необходимо познакомиться с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получить навыки проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.

Описание процесса выполнения работы

3.4. Последовательность выполнения работы

1. 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/emphynet.py, описывающего стандартную простую топологию сети mininet (рис. 3.1).

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

Рис. 3.1. Работа с каталогами и файлами

2. Изучите содержание скрипта lab_iperf3_topo.py (рис. 3.2).

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.

```
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. 3.2. Скрипт lab_iperf3_topo.py

3. Запустите скрипт создания топологии lab_iperf3_topo.py (рис. 3.3).

```
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
```

Рис. 3.3. Запуск скрипта lab_iperf3_topo.py

4. После отработки скрипта посмотрите элементы топологии и завершите работу mininet (рис. 3.4).

```

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=963>
<Host h2: h2-eth0:10.0.0.2 pid=966>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=971>
<Controller c0: 127.0.0.1:6653 pid=956>
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links

```

Рис. 3.4. Просмотр элементов топологии

- 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 (рис. 3.5):

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

Здесь:

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

```

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 )

```

Рис. 3.5. Изменение в скрипте `lab_iperf3_topo.py`

- Проверьте корректность отработки изменённого скрипта.

Измените скрипт `lab_iperf3_topo.py` так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети. Проверьте корректность отработки изменённого скрипта (рис. 3.6).

```

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 d6:f6:b8:39:be:a9
Host h2 has IP address 10.0.0.2 and MAC address 42:81:9e:93:f8:a3
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 3.6. Отработка изменённого скрипта

7. 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` (рис. 3.7).

```

*** Done
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

```

Рис. 3.7. Копирование скрипта

8. 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` (рис. 3.8).

```

from mininet.node import CPULimitedHost
from mininet.link import TCLink

```

Рис. 3.8. Записи об импорте

9. В скрипте `lab_iperf3_topo2.py` измените строку описания сети, указав на использование ограничения производительности и изоляции: (рис. 3.9).

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

Рис. 3.9. Использование ограничения производительности и изоляции

10. В скрипте lab_iperf3_topo2.py измените функцию задания параметров виртуального хоста h1, указав, что ему будет выделено 50% от общих ресурсов процессора системы: (рис. 3.10).

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

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

```
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', cpu=50 )
    h2 = net.addHost( 'h2', ip='10.0.0.2', cpu=45 )
```

Рис. 3.10. Функции задания параметров

11. 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 )
```

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;

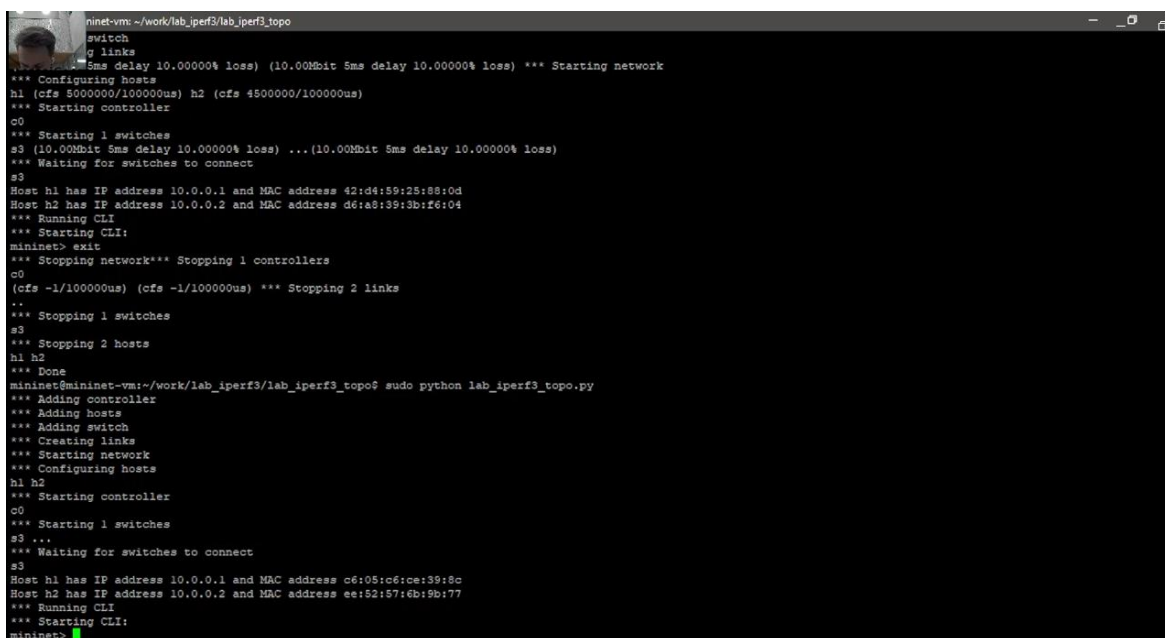
- параметр `use_htb` указывает на использование ограничителя интенсивности входящего потока Hierarchical Token Bucket (HTB) (рис. 3.11).

```
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. 3.11. Измененная функция

12. Запустите на отработку сначала скрипт `lab_iperf3_topo2.py`, затем `lab_iperf3_topo.py` и сравните результат (рис. 3.12).



```
mininet> ./lab_iperf3_topo2.py
switch
g links
... 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 42:d4:59:25:88:0d
Host h2 has IP address 10.0.0.2 and MAC address d6:a8:39:3b:f6:04
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
(cfs -1/1000000us) (cfs -1/1000000us) *** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
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 c6:05:c6:ce:39:8c
Host h2 has IP address 10.0.0.2 and MAC address ee:52:57:6b:9b:77
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 3.12. Работа двух скриптов

13. Постройте графики по проводимому эксперименту:

- Сделайте копию скрипта `lab_iperf3_topo2.py` и поместите его в подкаталог `iperf`:

```
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 (рис. 3.13).

```
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 1346 Dec  2 08:39 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nano lab
```

Рис. 3.13. Работа с файлами и каталогами

14. В начале скрипта lab_iperf3.py добавьте запись (рис. 3.14).

```
import time
```

```
from mininet.node import CPULimitedHost
from mininet.link import TCLink
import time

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

Рис. 3.14. Добавление записи

15. 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 (рис. 3.15).

```
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()
```

Рис. 3.15. Изменение кода в скрипте

16. 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 (рис. 3.16):

```
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 )
```

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



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

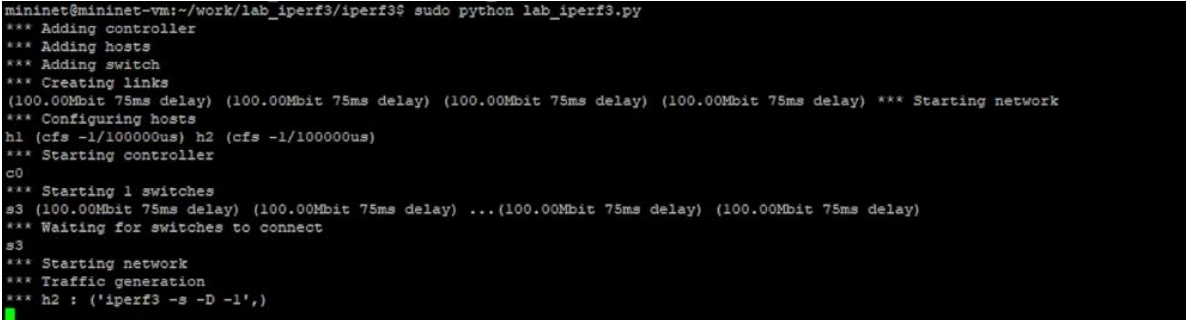
info( '*** Traffic generation\n')
h2.cmdPrint( 'iperf3 -s -D -1' )
time.sleep(10) # Wait 10 seconds for servers to start
h1.cmdPrint( 'iperf3 -c', h2.IP(), '-J > iperf_result.json' )

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. 3.16. Синтаксис вызова iPerf3

17. Запустите на отработку скрипт lab_iperf3.py (рис. 3.17):

```
sudo python lab_iperf3.py
```

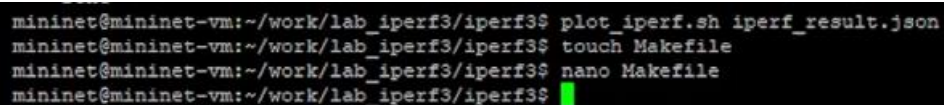


```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ sudo python lab_iperf3.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) *** Starting network
*** Configuring hosts
h1 (cfs -l/100000us) h2 (cfs -l/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
*** Starting network
*** Traffic generation
*** h2 : ('iperf3 -s -D -1',)
```

Рис. 3.17. Отработка скрипта

18. Постройте графики из получившегося JSON-файла (рис. 3.18):

plot_iperf.sh iperf_result.json

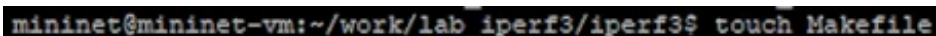


```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch Makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nano Makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Рис. 3.18. Построение графиков

19. Создайте Makefile для проведения всего эксперимента (рис. 3.19):

touch Makefile



```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch Makefile
```

Рис. 3.19. 123

20. В Makefile пропишите запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 3.20):

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

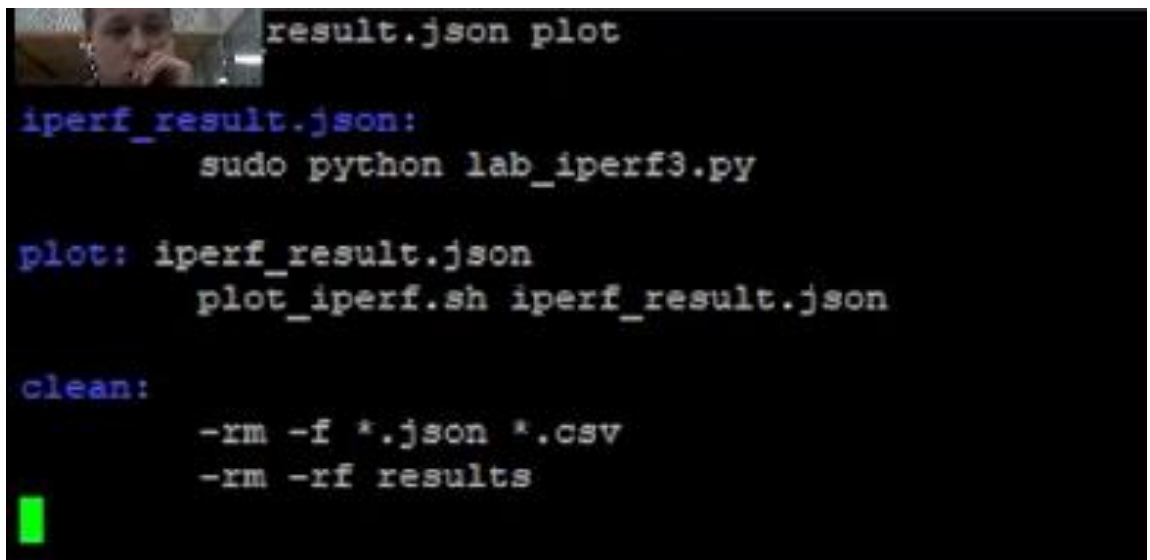


Рис. 3.20. Makefile

21. Проверьте корректность отработки Makefile (рис. 3.21):

make clean

make

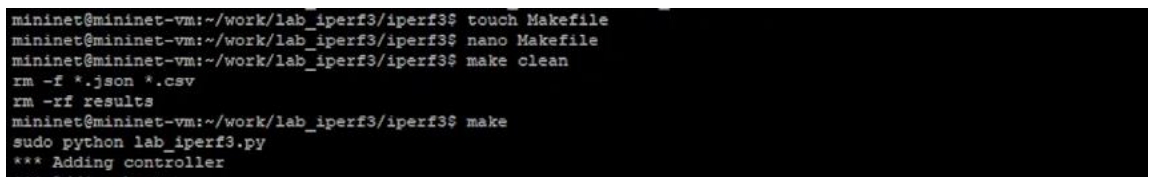


Рис. 3.21. Проверка отработки

22. Завершите соединение с виртуальной машиной mininet и выключите её (рис. 3.22).

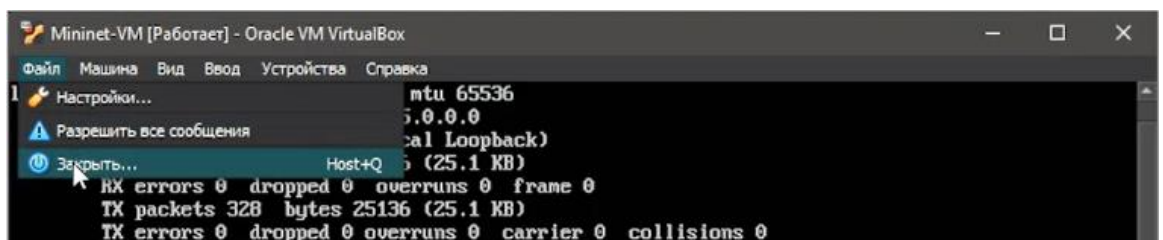


Рис. 3.22. Выключение виртуальной машины

Вывод

В данной лабораторной работе мне успешно удалось познакомиться с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получить навыки проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.