Лабораторная работа №3

Измерение и тестирование пропускной способности сети. Воспроизводимый эксперимент

Тазаева Анастасия Анатольевна

Содержание

Список иллюстраций

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 Задание

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

3 Выполнение лабораторной работы

1. Запустила виртуальную среду с mininet. Из основной ОС подключилась к виртуальной машине.

2. С помощью API Mininet создала 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 создала подкаталог `lab_iperf3_topo` и скопировала в него файл с примером скрипта `mininet/examples/emptynet.py`, описывающего стандартную простую топологию сети mininet (рис. 3.1)

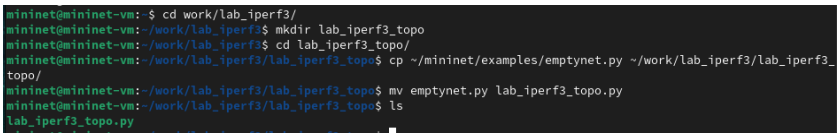

Рис. 3.1: Копирование файла с примером скрипта mininet/examples/emptynet.py

Изучила содержание скрипта. 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.

Запустила скрипт создания топологии `lab_iperf3_topo.py` (рис. 3.2):

```
sudo python lab_iperf3_topo.py
```

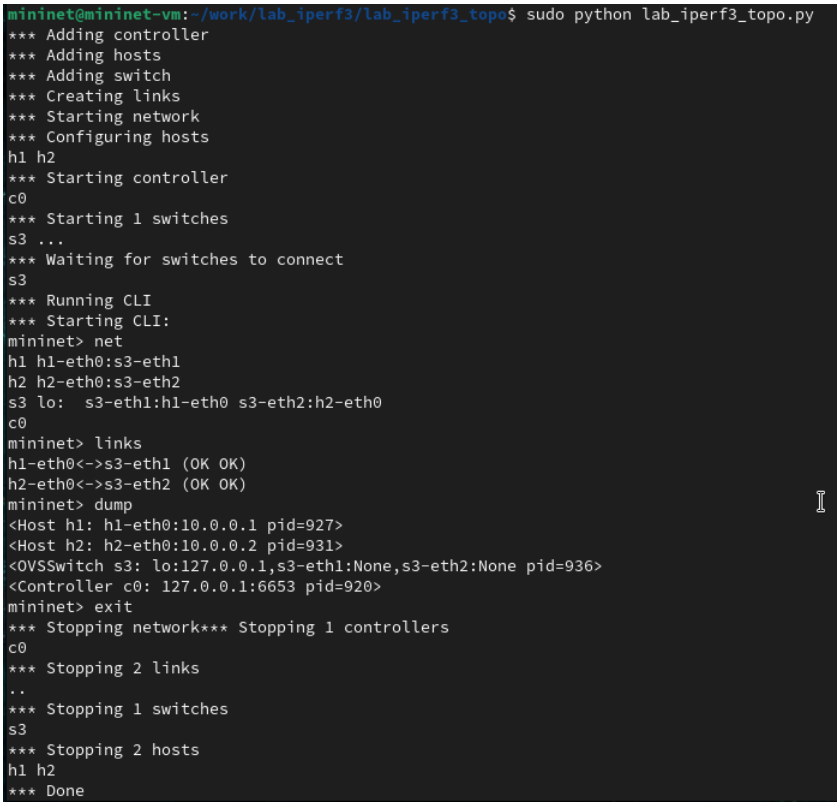

Рис. 3.2: Запуск lab_iperf3_topo.py, выполнение команд

После отработки скрипта посмотрела элементы топологии и завершила работу mininet (рис. 3.2):

```
mininet> net
mininet> links
mininet> dump
mininet> exit
```

3. Внесла 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, добавила строку (рис. 3.3):

```
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.

```
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()
```

Рис. 3.3: Изменение файла lab_iperf3_topo.py

4. Проверила корректность отработки изменённого скрипта (рис. 3.4). Теперь перед запуском CLI появляется надпись с информацией о хосте h1.

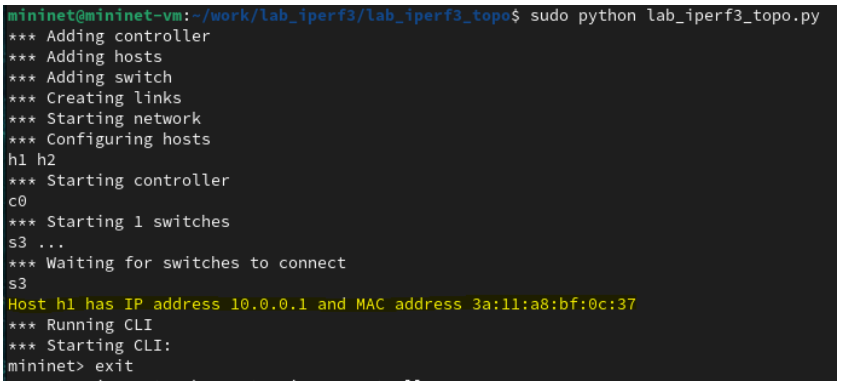

Рис. 3.4: Проверка запуска скрипта lab_iperf3_topo.py

5. Изменила скрипт `lab_iperf3_topo.py` так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети (рис. 3.5). Проверила корректность отработки изменённого скрипта (рис. 3.6).

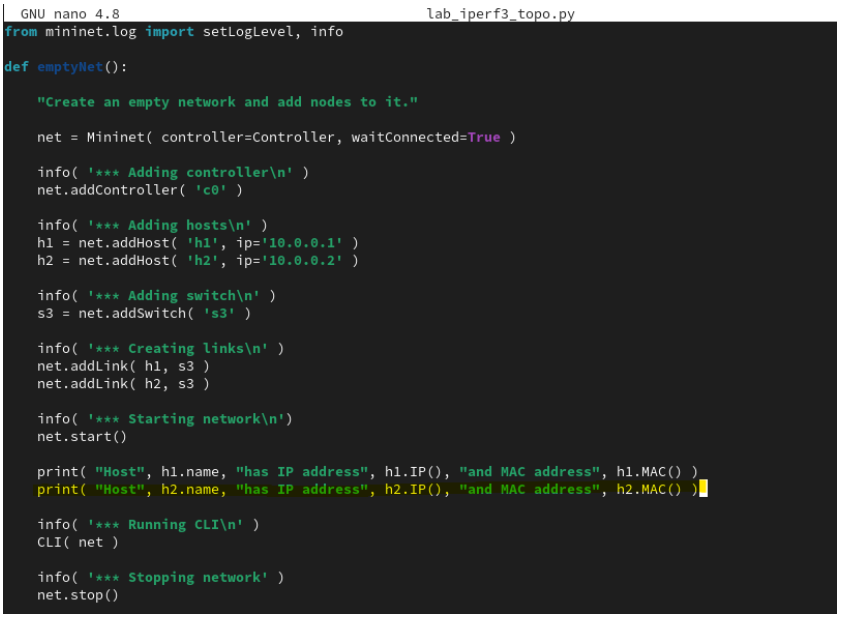

Рис. 3.5: Изменение файла lab_iperf3_topo.py. Часть 2

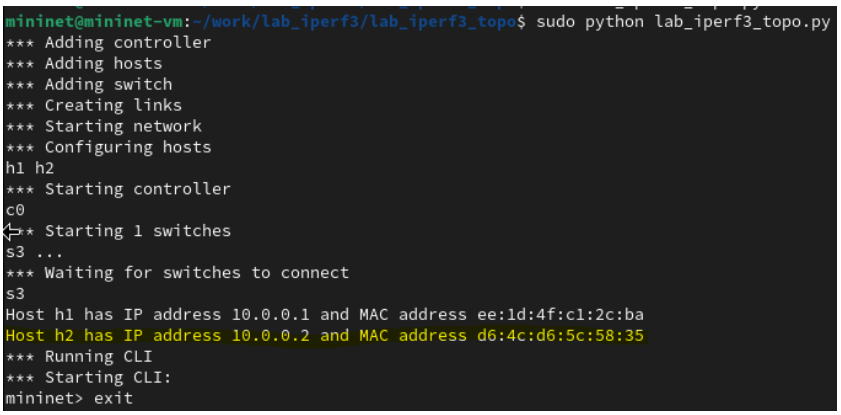

Рис. 3.6: Проверка запуска скрипта lab_iperf3_topo.py. Часть 2

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

- Сделала копию скрипта `lab_iperf3_topo.py`:

```
cp lab_iperf3_topo.py lab_iperf3_topo2.py
```

- В начале скрипта `lab_iperf3_topo2.py` добавила записи об импорте классов CPULimitedHost и TCLink (рис. 3.7):

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

- В скрипте `lab_iperf3_topo2.py` изменила строку описания сети, указав на использование ограничения производительности и изоляции (рис. 3.7):

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

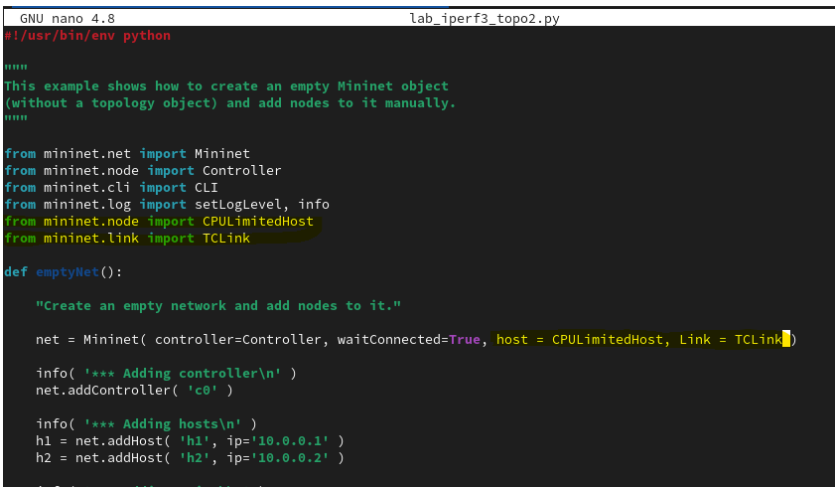

Рис. 3.7: lab_iperf3_topo2.py. Добавление записи об импорте и изменение строки описания сети

- В скрипте `lab_iperf3_topo2.py` изменила функцию задания параметров виртуального хоста h1, указав, что ему будет выделено 50% от общих ресурсов процессора системы (рис. 3.8):

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

- Аналогичным образом для хоста h2 задала долю выделения ресурсов процессора в 45% (рис. 3.8):

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

– В скрипте `lab_iperf3_topo2.py` изменила функцию параметров соединения между хостом h1 и коммутатором s3 (рис. 3.8):

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

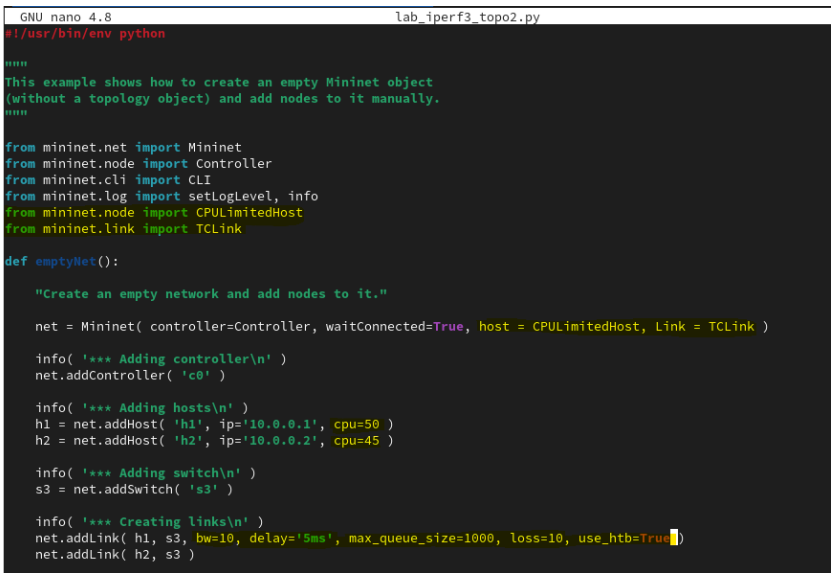

Рис. 3.8: lab_iperf3_topo2.py. Задание cpu и изменение параметров соединения между хостом h1 и коммутатором s3

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) выража-ется в пакетах; - параметр use_htb указывает на использование ограничителя интенсив-ности входящего потока Hierarchical Token Bucket (HTB).

Запустила на отработку сначала скрипт `lab_iperf3_topo2.py` (рис. 3.9), затем `lab_iperf3_topo.py` (рис. 3.10). Теперь появляется дополнительная информация, которую и меняли пунктом выше.

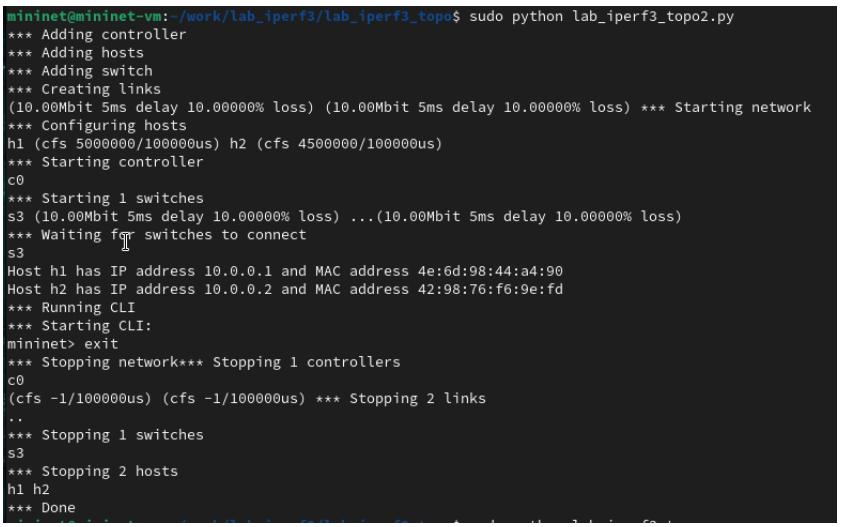

Рис. 3.9: Запуск скрипта lab_iperf3_topo2.py.

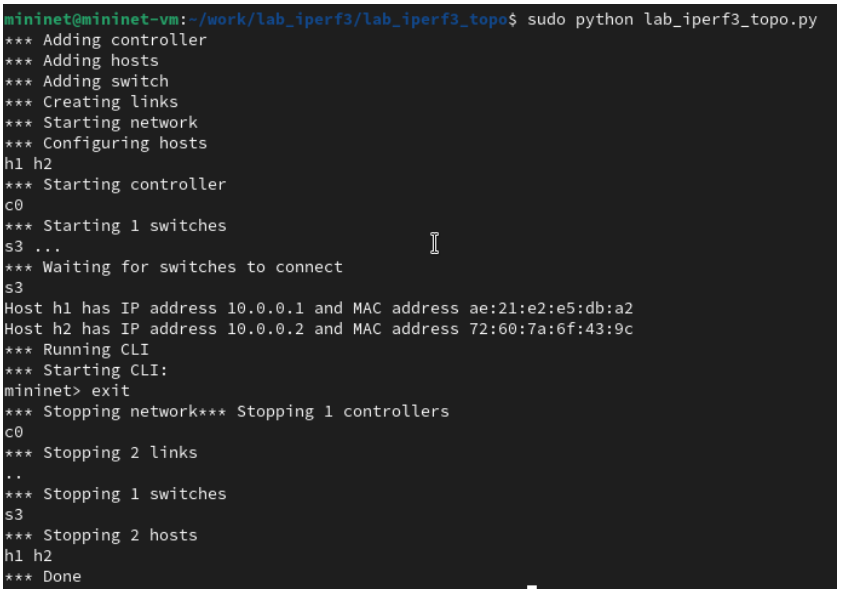

Рис. 3.10: Запуск скрипта lab_iperf3_topo.py.

7. Теперь необходимо построить графики по проводимому эксперименту :

- Сделала копию скрипта `lab_iperf3_topo2.py` и поместила его в подкаталог iperf (рис. 3.11):

```
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
```

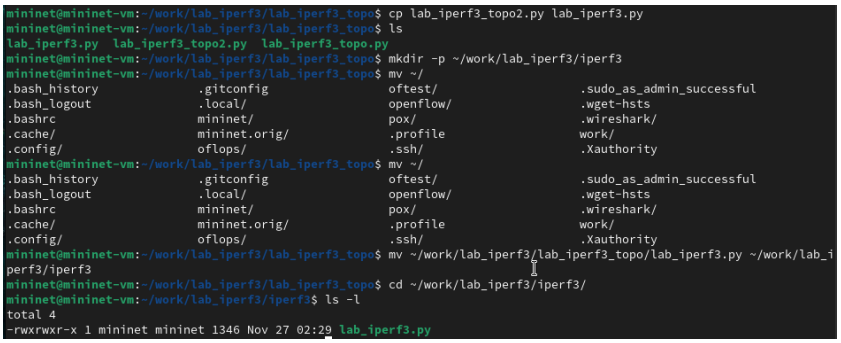

Рис. 3.11: Создание файла lab_iperf3.py

- В начале скрипта lab_iperf3.py добавила запись (рис. 3.12):

```
...
import time
...
```

- Изменила код в скрипте lab_iperf3.py так, чтобы на хостах не было ограничения по использованию ресурсов процессора и 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. 3.12).

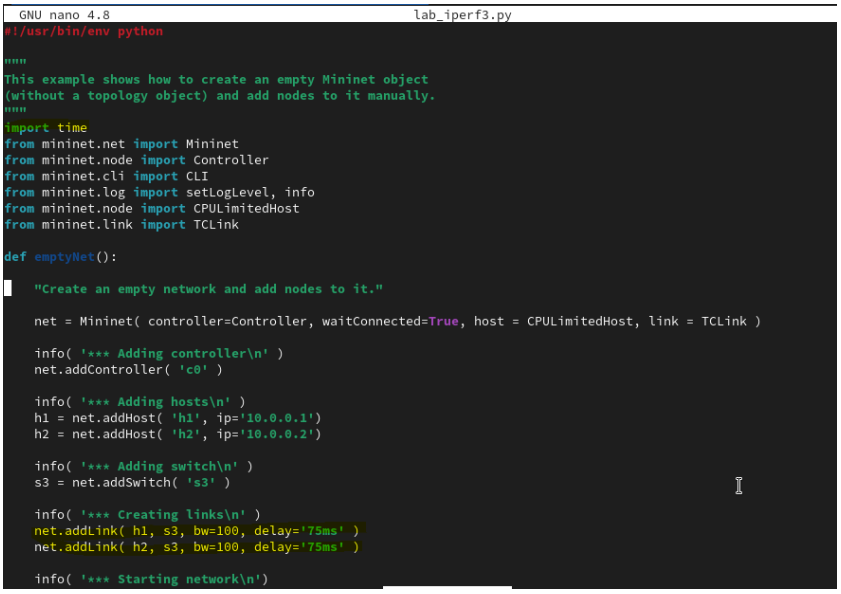

Рис. 3.12: lab_iperf3.py. Изменение параметров

- После функции старта сети описала запуск на хосте h2 сервера iPerf3, а на хосте h1 запуск с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл, закомментировала строки, отвечающие за запуск CLI-интерфейса (рис. 3.13):

```
...
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 )
...
```

```
    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' )

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

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

Рис. 3.13: lab_iperf3.py. Запуск сервера и клиента

Здесь мы в фоновом режиме запускаем сервер - хост h2 с опцией -1 (только 1 клиент), и щапускаем клиент хост h1 с опцией -J, перенаправляя результаты в файл JSON.

- Запустила на отработку скрипт lab_iperf3.py (рис. 3.14):

```
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 -1/100000us) 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')
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
```

Рис. 3.14: lab_iperf3.py. Запуск скрипта

- Построила графики из получившегося JSON-файла (рис. 3.15):

```
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 -l
total 20
-rw-rw-r-- 1 mininet mininet  967 Nov 27 02:55 iperf.csv
-rw-r--r-- 1 mininet mininet 7766 Nov 27 02:53 iperf_result.json
-rwxrwxr-x 1 mininet mininet 1349 Nov 27 02:52 lab_iperf3.py
drwxrwxr-x 2 mininet mininet 4096 Nov 27 02:55 results
```

Рис. 3.15: Построение графиков

- Создала Makefile для проведения всего эксперимента:

```
touch Makefile
```

- В Makefile прописала запуск скрипта эксперимента, построение графи- ков и очистку каталога от результатов (рис. 3.16)

```
 GNU nano 4.8                                  Makefile
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. 3.16: Код Makefile

- Проверила корректность отработки Makefile (рис. 3.17 и 3.18):

```
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$ ls -l
total 8
-rwxrwxr-x 1 mininet mininet 1349 Nov 27 02:52 lab_iperf3.py
-rw-rw-r-- 1 mininet mininet  180 Nov 27 03:01 Makefile
```

Рис. 3.17: Проверка отработки Makefile. Часть 1

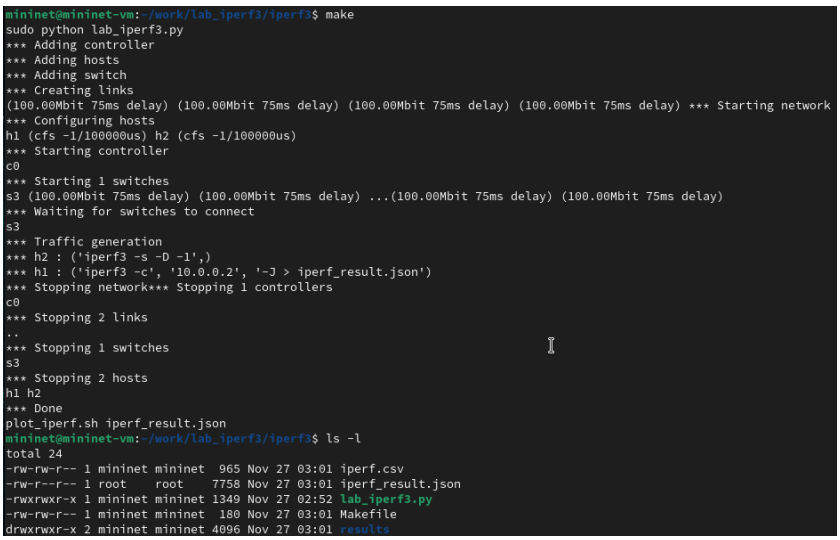

Рис. 3.18: Проверка отработки Makefile. Часть 2

8. Завершила соединение с виртуальной машиной mininet и выключила её.

4 Выводы

В ходе 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.