Лабораторная работа №3

Измерение и тестирование пропускной способности сети. Воспроизводимый эксперимент

Тазаева Анастасия Анатольевна

Содержание

# 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. 1)

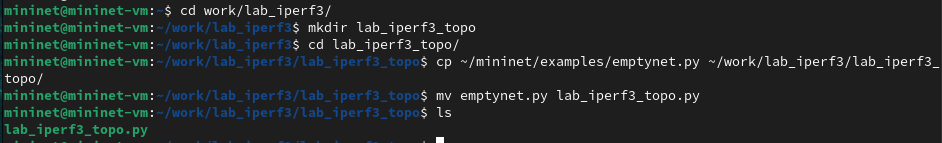


Рис. 1: Копирование файла с примером скрипта mininet/examples/emptynet.py

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. 2):

sudo python lab\_iperf3\_topo.py

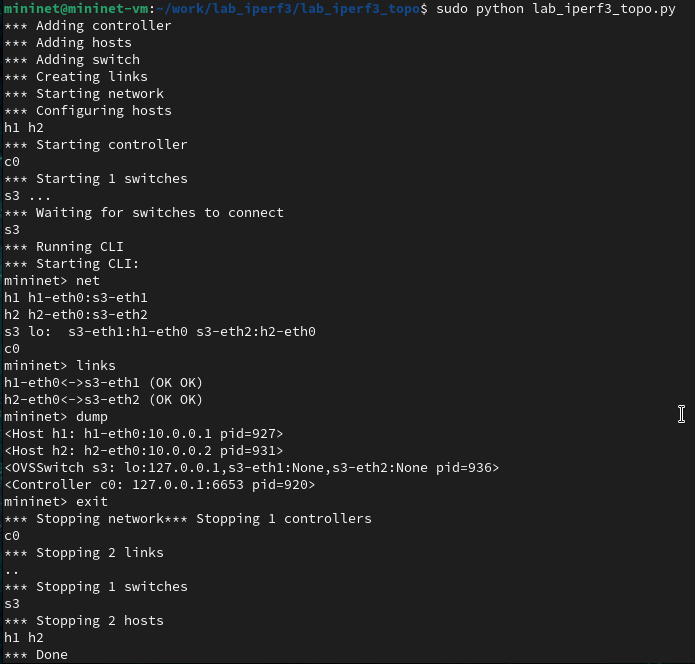


Рис. 2: Запуск lab\_iperf3\_topo.py, выполнение команд

После отработки скрипта посмотрела элементы топологии и завершила работу mininet (рис. 2):

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

1. 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):

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.

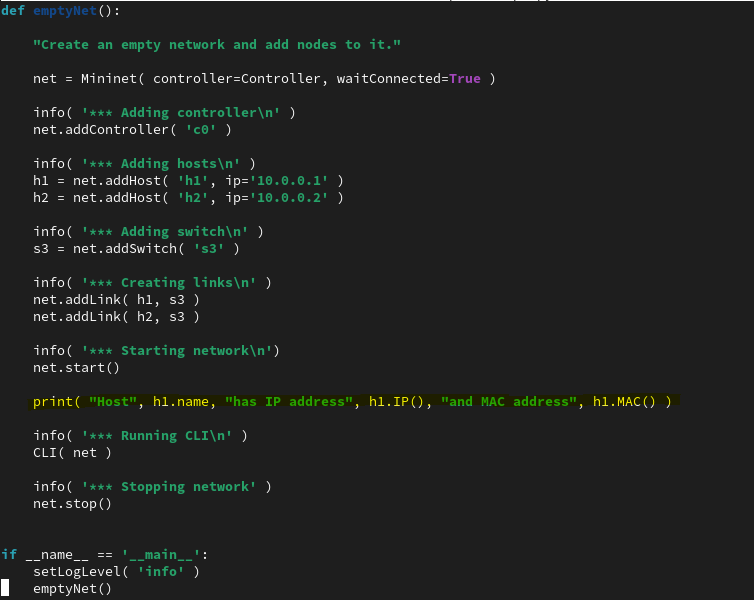


Рис. 3: Изменение файла lab\_iperf3\_topo.py

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

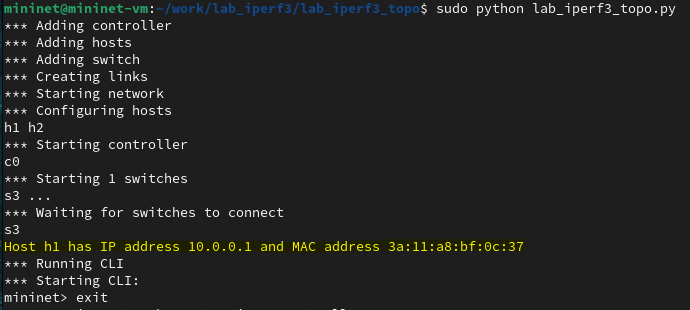


Рис. 4: Проверка запуска скрипта lab\_iperf3\_topo.py

1. Изменила скрипт lab\_iperf3\_topo.py так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети (рис. 5). Проверила корректность отработки изменённого скрипта (рис. 6).

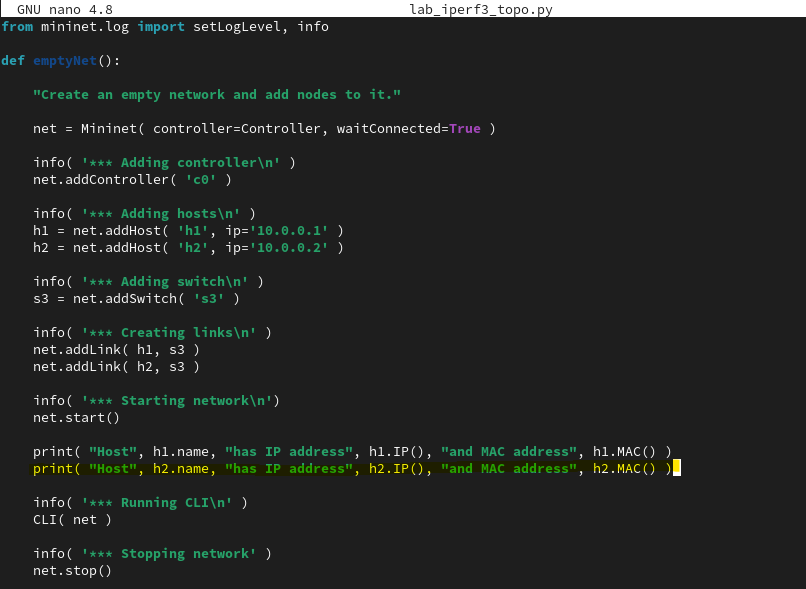


Рис. 5: Изменение файла lab\_iperf3\_topo.py. Часть 2

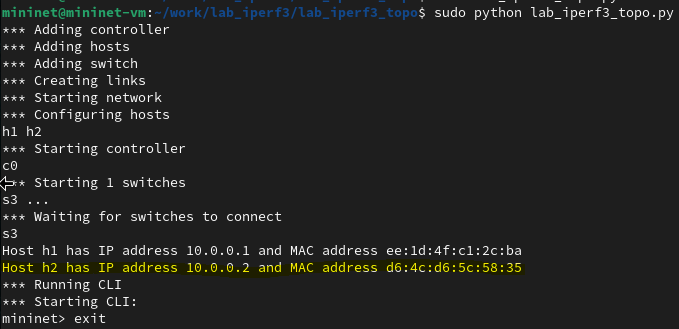


Рис. 6: Проверка запуска скрипта lab\_iperf3\_topo.py. Часть 2

1. 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. 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. 7):

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

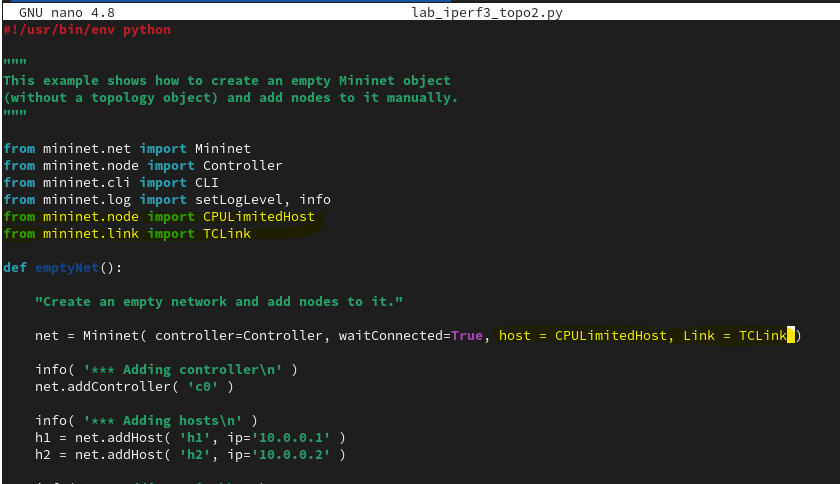


Рис. 7: lab\_iperf3\_topo2.py. Добавление записи об импорте и изменение строки описания сеnи

* 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. 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. 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. 8):

...  
net.addLink( h1, s3, bw=10, delay='5ms', max\_queue\_size=1000, loss=10, use\_htb=True )  
...

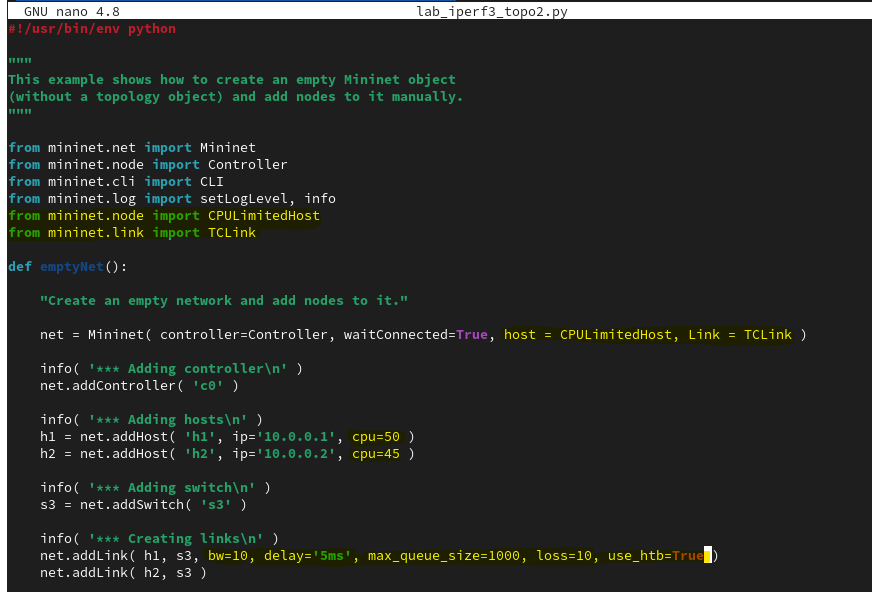


Рис. 8: lab\_iperf3\_topo2.py. Задание cpu и изменение параметров соединения между хостом h1 и коммутатором s3

Здесь добавляется двунаправленный канал с характеристиками пропуск- ной 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. 9), затем lab\_iperf3\_topo.py (рис. 10). Теперь появляется дополнительная информация, которую и меняли пунктом выше.

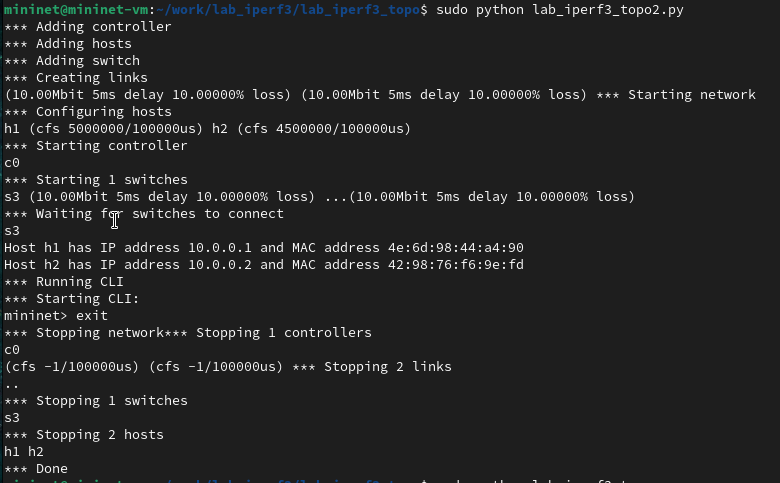


Рис. 9: Запуск скрипта lab\_iperf3\_topo2.py.

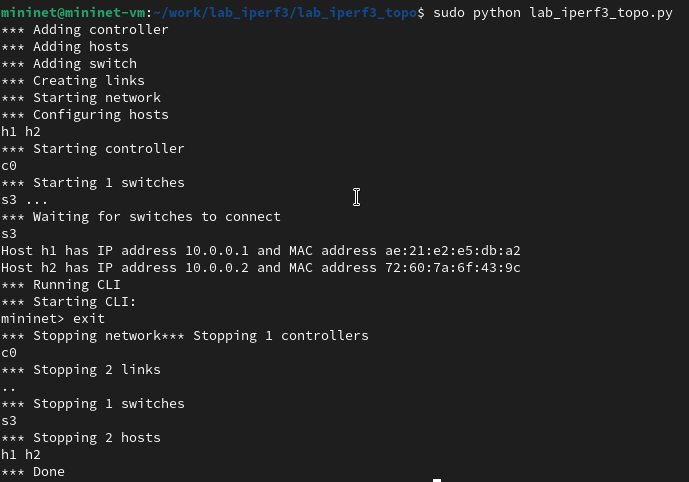


Рис. 10: Запуск скрипта lab\_iperf3\_topo.py.

1. Теперь необходимо построить графики по проводимому эксперименту :

* Сделала копию скрипта lab\_iperf3\_topo2.py и поместила его в подкаталог iperf (рис. 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

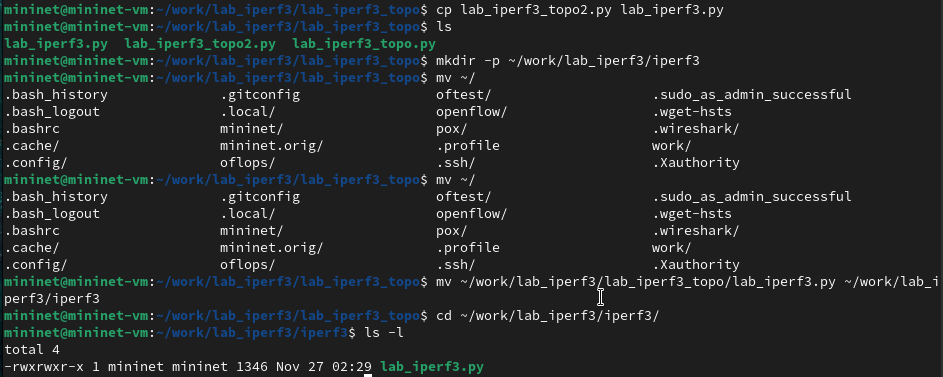


Рис. 11: Создание файла lab\_iperf3.py

* В начале скрипта lab\_iperf3.py добавила запись (рис. 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. 12).

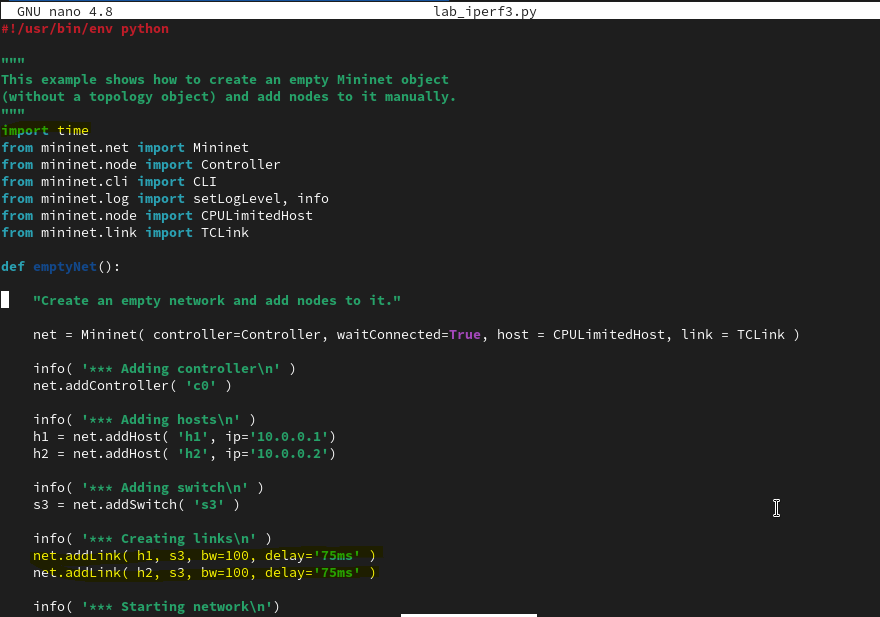


Рис. 12: lab\_iperf3.py. Изменение параметров

* 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. 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 )  
...

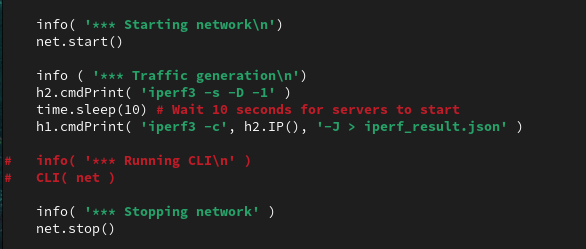


Рис. 13: lab\_iperf3.py. Запуск сервера и клиента

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. 14):

sudo python lab\_iperf3.py

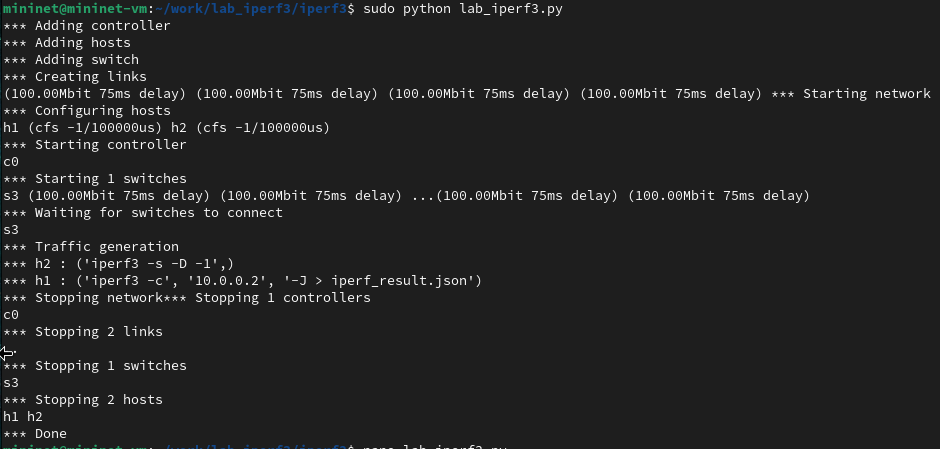


Рис. 14: lab\_iperf3.py. Запуск скрипта

* Построила графики из получившегося JSON-файла (рис. 15):

plot\_iperf.sh iperf\_result.json

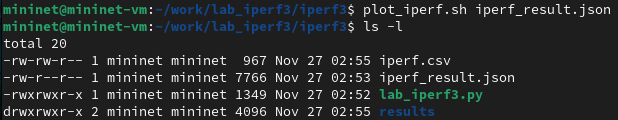


Рис. 15: Построение графиков

* Создала Makefile для проведения всего эксперимента:

touch Makefile

* В Makefile прописала запуск скрипта эксперимента, построение графи- ков и очистку каталога от результатов (рис. 16)

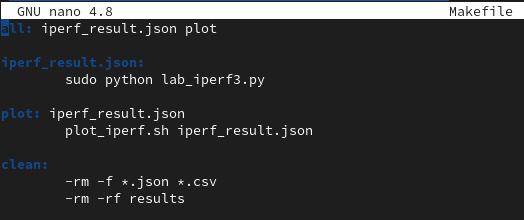


Рис. 16: Код Makefile

* Проверила корректность отработки Makefile (рис. 17 и 18):

make clean  
make

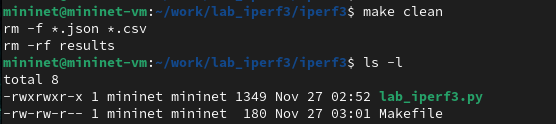


Рис. 17: Проверка отработки Makefile. Часть 1

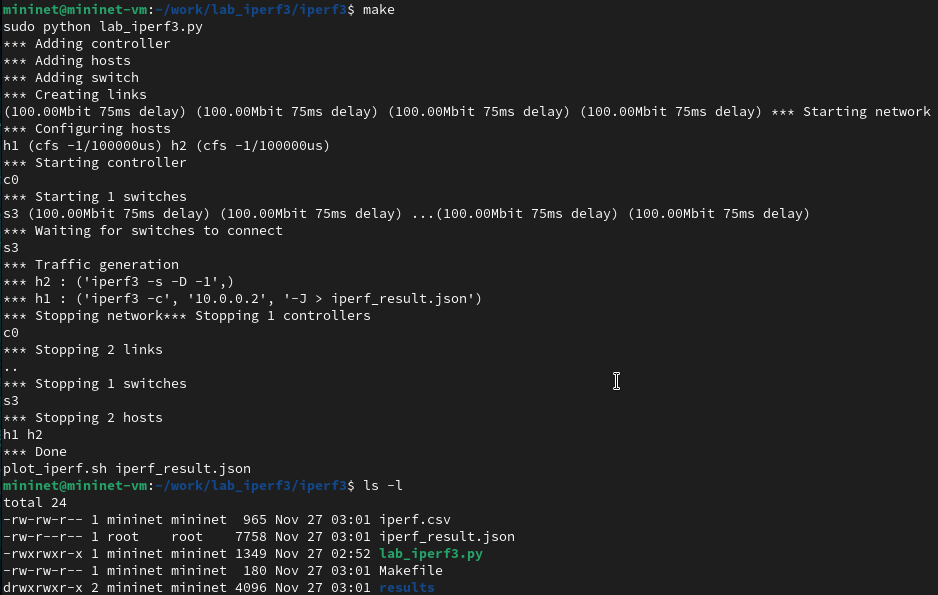


Рис. 18: Проверка отработки Makefile. Часть 2

1. 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.