

# **Лабораторная работа № 3**

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

**Хрусталев Влад Николаевич**

# **Содержание**

<b>1 Цель работы</b>	<b>4</b>
<b>2 Теоретическое введение</b>	<b>5</b>
<b>3 Задание</b>	<b>6</b>
<b>4 Выполнение лабораторной работы</b>	<b>7</b>
<b>5 Выводы</b>	<b>31</b>
<b>Список литературы</b>	<b>32</b>

# Список иллюстраций

4.1	Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet) . . . . .	8
4.2	Содержание файла lab_iperf3_topo.py . . . . .	10
4.3	Запуск скрипта создания топологии и дальнейший просмотр элементов . . . . .	12
4.4	Внесение изменения в скрипт, позволяющего вывести на экран информацию о хостах h1 и h2 (имя, IP-адрес, MAC-адрес) . . . . .	14
4.5	Проверка корректности отработки скрипта . . . . .	16
4.6	Настройка параметров производительности . . . . .	18
4.7	Запуск скрипта с настройкой параметров производительности и без нее . . . . .	20
4.8	Создание копии скрипта lab_iperf3_topo2.py . . . . .	22
4.9	Изменен ия кода в скрипте lab_iperf3.py . . . . .	24
4.10	Запуск скрипта lab_iperf3.py . . . . .	26
4.11	Создание Makefile . . . . .	28
4.12	Проверка работы Makefile . . . . .	30

# **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 Теоретическое введение

Mininet[1] – это эмулятор компьютерной сети. Под компьютерной сетью подразумеваются простые компьютеры – хосты, коммутаторы, а так же OpenFlow-контроллеры. С помощью простейшего синтаксиса в примитивном интерпретаторе команд можно разворачивать сети из произвольного количества хостов, коммутаторов в различных топологиях и все это в рамках одной виртуальной машины(ВМ). На всех хостах можно изменять сетевую конфигурацию, пользоваться стандартными утилитами(ifconfig, ping) и даже получать доступ к терминалу. На коммутаторы можно добавлять различные правила и маршрутизировать трафик.

iPerf3[2] представляет собой кроссплатформенное клиент-серверное приложение с открытым исходным кодом, которое можно использовать для измерения пропускной способности между двумя конечными устройствами. iPerf3 может работать с транспортными протоколами TCP, UDP и SCTP:

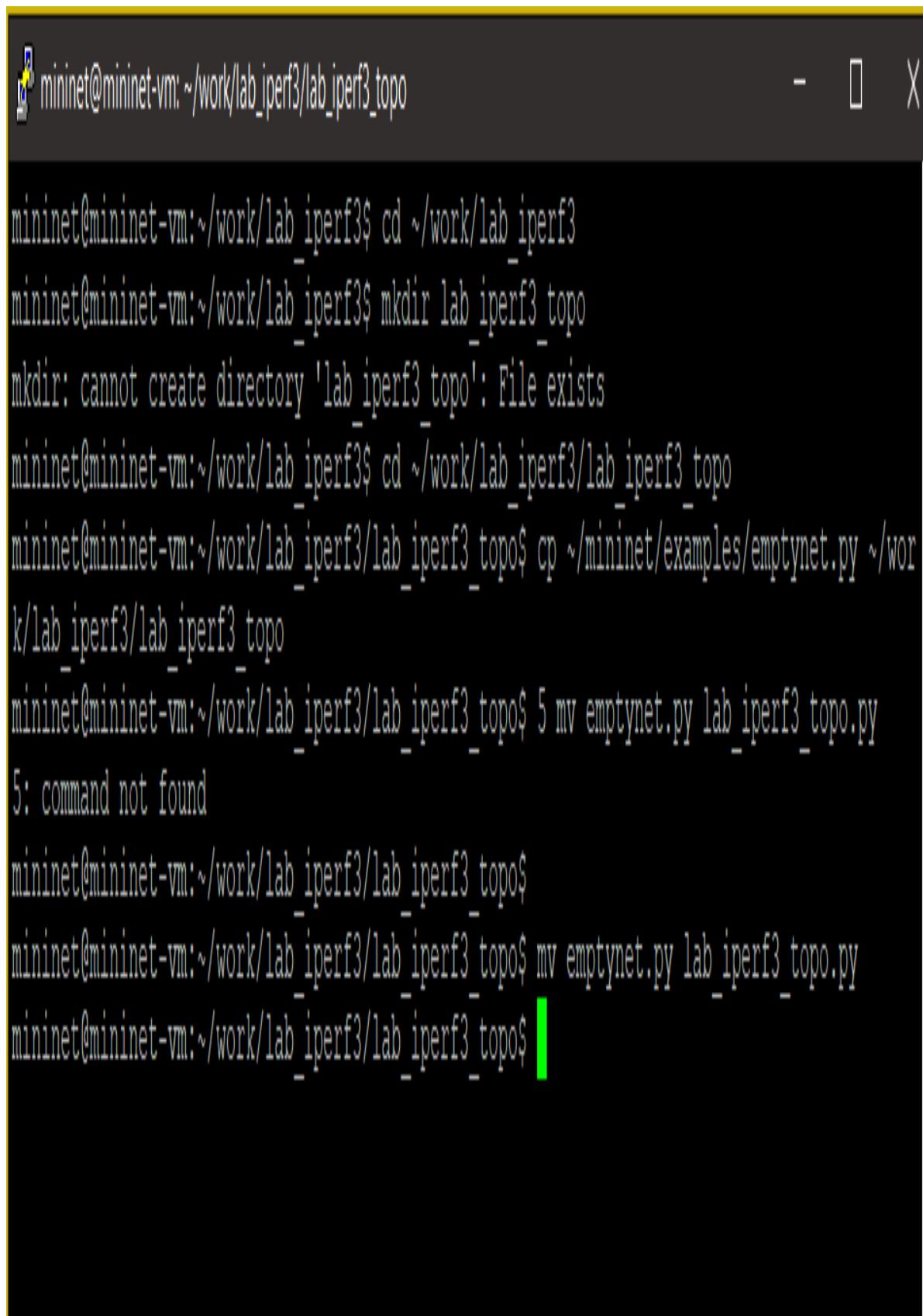
- TCP и SCTP:
  - измеряет пропускную способность;
  - позволяет задать размер MSS/MTU;
  - отслеживает размер окна перегрузки TCP (CWnd).
- UDP:
  - измеряет пропускную способность;
  - измеряет потери пакетов;
  - измеряет колебания задержки (jitter);
  - поддерживает групповую рассылку пакетов (multicast).

## **3 Задание**

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

## **4 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. Для этого в каталоге /work/lab\_iperf3 для работы над проектом создадим подкаталог lab\_iperf3\_topo и скопируем в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet (рис. 4.1):



```
mininet@mininet-vm:~/work/lab_iperf3$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ mkdir lab_iperf3_topo
mkdir: cannot create directory 'lab_iperf3_topo': File exists
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/emptynet.py ~/work/lab_iperf3/lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_topo.py
5: command not found
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$
```

Рис. 4.1: Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet)

Изучим содержание скрипта lab\_iperf3\_topo.py (рис. 4.2).

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.

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.

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
GNU nano 4.8                               lab_iperf3_topo.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

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. 4.2: Содержание файла lab\_iperf3\_topo.py

Запустим скрипт создания топологии `lab_iperf3_topo.py`. После отработки скрипта просмотрим элементы топологии и завершим работу mininet (рис. 4.3):

```
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 ...
*** 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=752>
<Host h2: h2-eth0:10.0.0.2 pid=756>
<OVSSwitch s3: lo:127.0.0.1, s3-eth1:None, s3-eth2:None pid=761>
<Controller c0: 127.0.0.1:6653 pid=745>
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$
```

Рис. 4.3: Запуск скрипта создания топологии и дальнейший просмотр элементов

Объединим два пункта лабораторной работы и внесём изменения сразу для обоих хостов. А именно 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. 4.4):

Здесь:

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

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo
```

```
GNU nano 4.8          lab_iperf3_topo.py      Modified
```

```
#!/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()  
    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() )
```

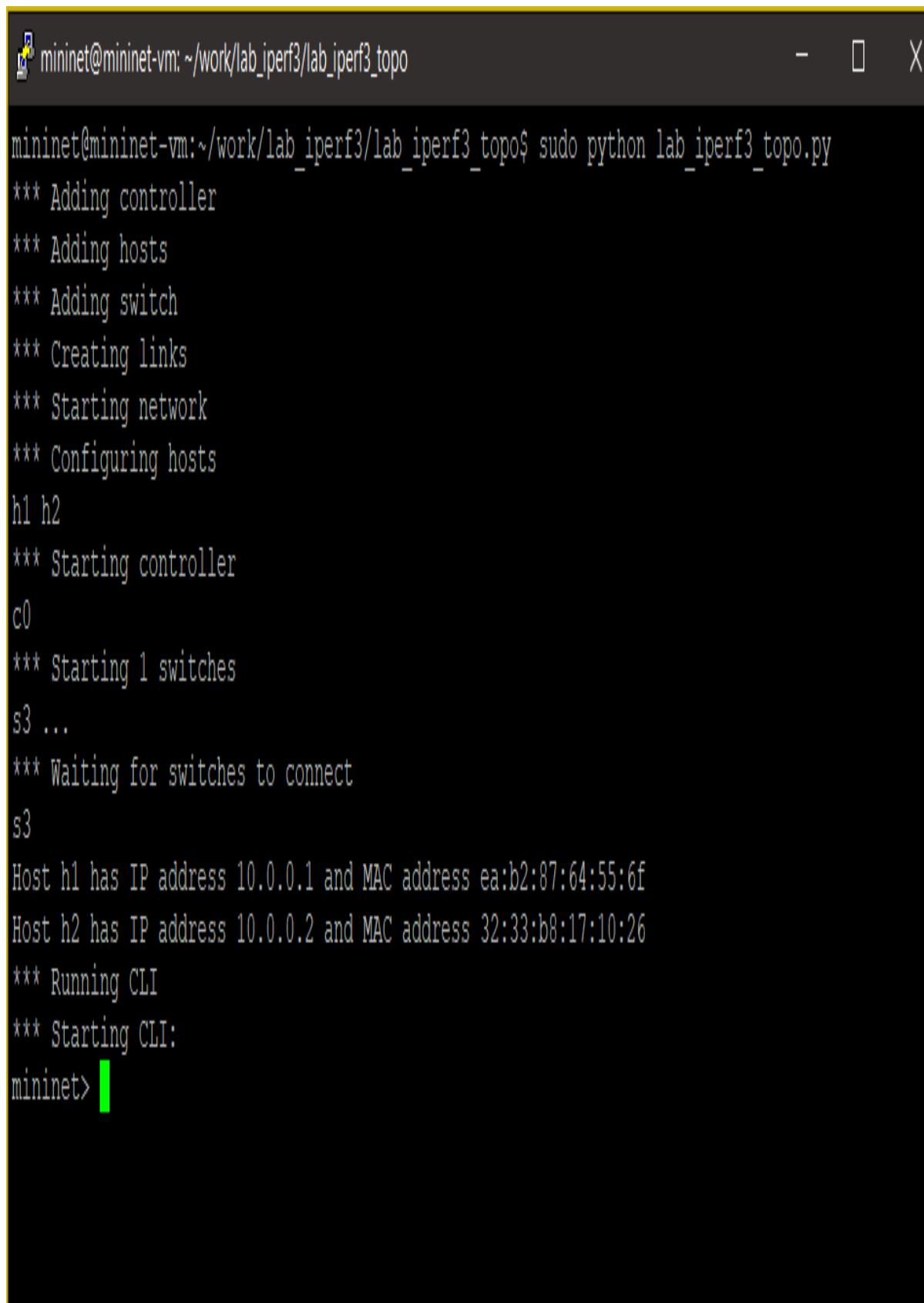
```
    info( "*** Running CLI\n" )  
    CLI( net )
```

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

```
if __name__ == '__main__':
```

Рис. 4.4: Внесение изменения в скрипт, позволяющего вывести на экран информацию о хостах h1 и h2 (имя, IP-адрес, MAC-адрес)

Проверим корректность отработки изменённого скрипта (рис. 4.5):

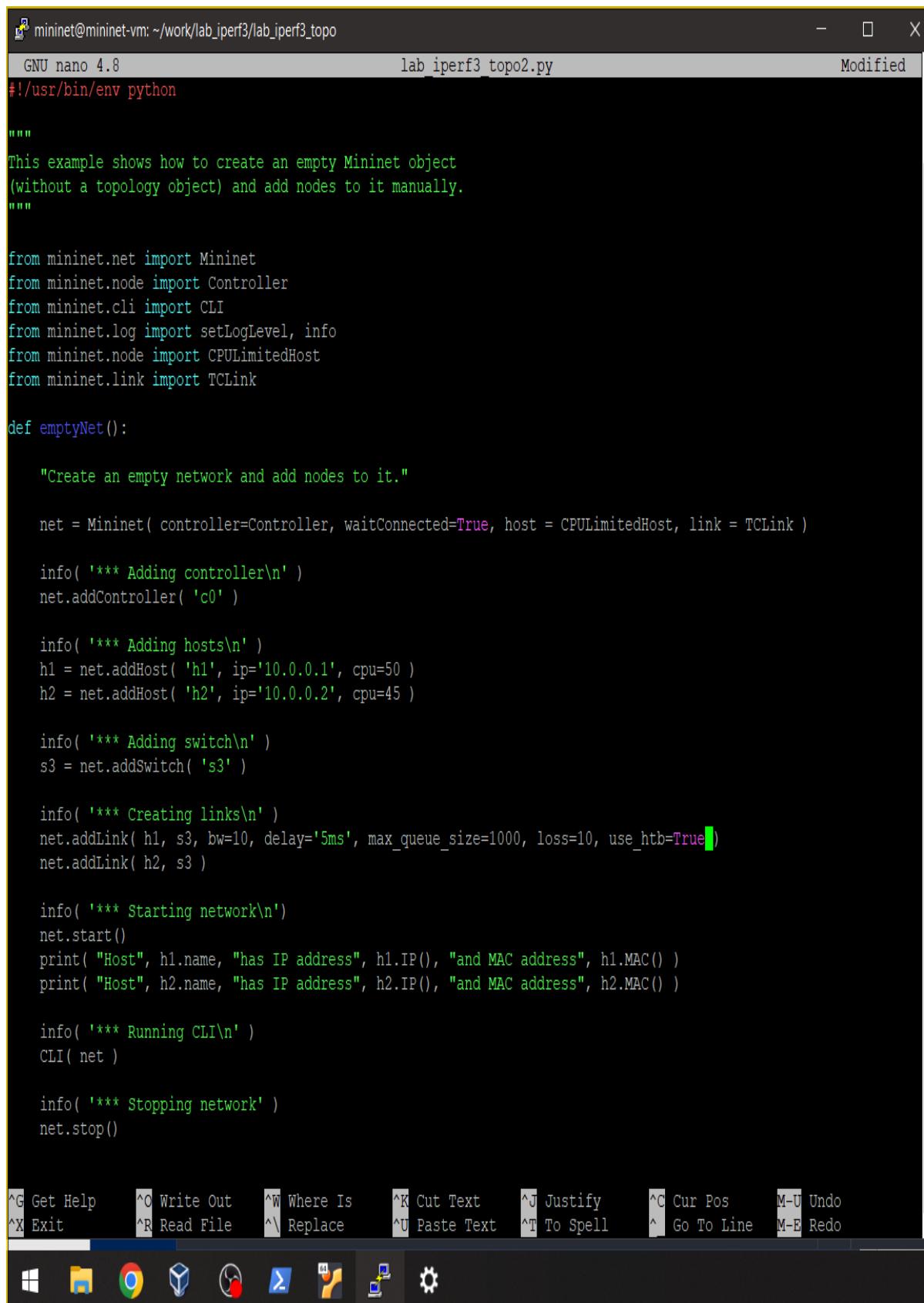


```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo - X
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s3 ...
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address ea:b2:87:64:55:6f
Host h2 has IP address 10.0.0.2 and MAC address 32:33:b8:17:10:26
*** Running CLI
*** Starting CLI:
mininet> [REDACTED]
```

Рис. 4.5: Проверка корректности отработки скрипта

Нам вывелась информация об IP и mac адресах хостов.

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов CPULimitedHost и TCLink. Добавим в скрипт настройки параметров производительности. Для начала сделаем копию скрипта lab\_iperf3\_topo.py. В начале скрипта lab\_iperf3\_topo2.py добавим записи об импорте классов CPULimitedHost и TCLink. Далее изменим строку описания сети, указав на использование ограничения производительности и изоляции. Следующим шагом 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 конце изменим функцию параметров соединения между хостом h1 и коммутатором s3 (рис. 4.6).



Screenshot of a terminal window titled "mininet@mininet-vm: ~/work/lab\_iperf3/lab\_iperf3\_topo". The window shows a Python script named "lab\_iperf3\_topo2.py" which is being edited. The script demonstrates how to create an empty Mininet object and add nodes to it manually. It includes imports for Mininet, Controller, CLI, log, node, and link modules. The script defines a function "emptyNet()" that creates a controller, adds two hosts (h1 and h2) with specific IP addresses and CPU limits, adds a switch (s3), and creates links between them. It also prints host information and runs a CLI. The terminal window has a dark theme with white text and a black background. A menu bar at the top right shows "Modified". A status bar at the bottom provides keyboard shortcuts for various functions like Get Help (^G), Write Out (^O), and Exit (^X).

```
GNU nano 4.8          lab_iperf3_topo2.py          Modified
#!/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, 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 )

    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 )

    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 )

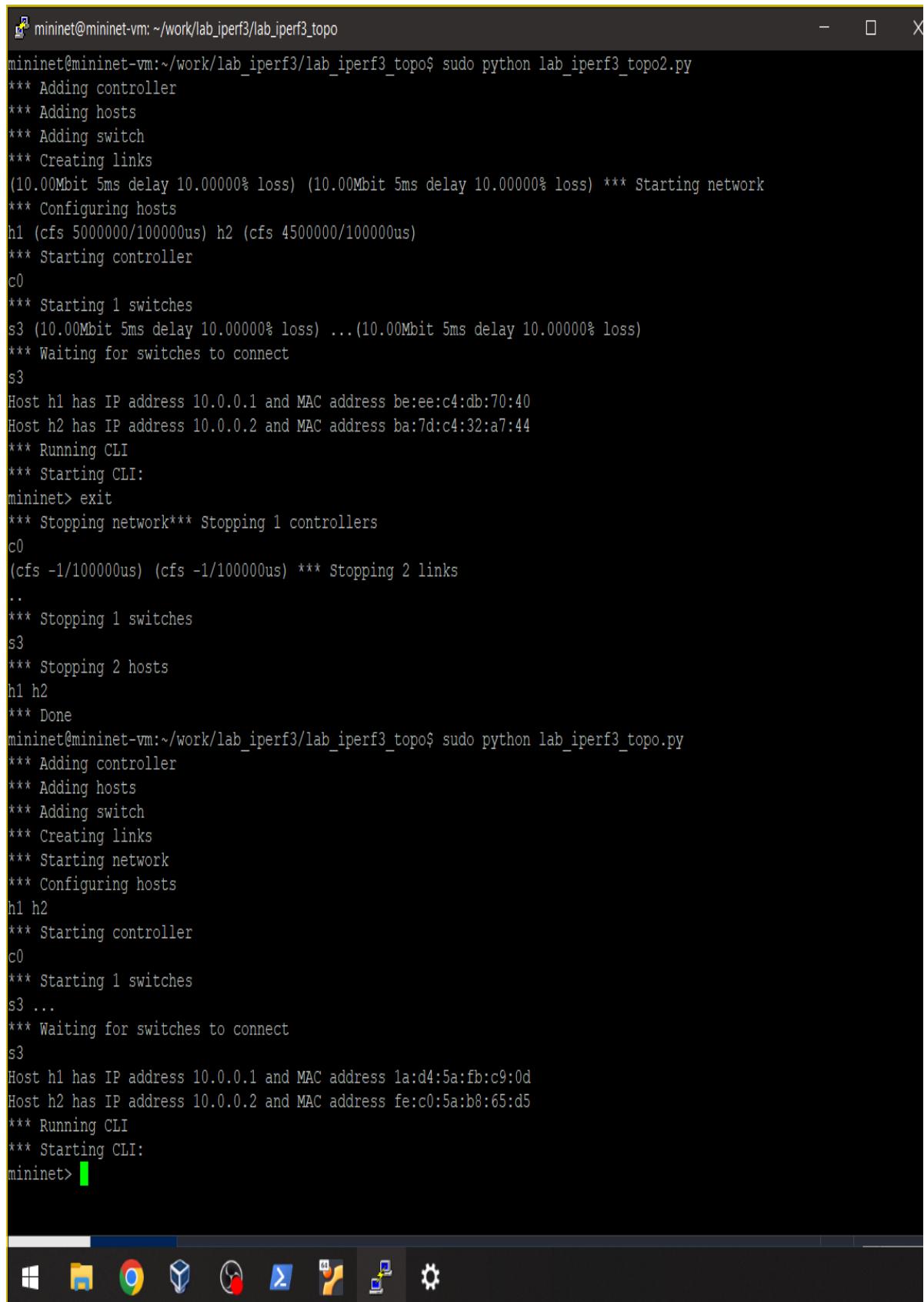
    info( "*** Stopping network" )
    net.stop()

^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

```

Рис. 4.6: Настройка параметров производительности

Запустим на отработку сначала скрипты `lab_iperf3_topo2.py`, затем `lab_iperf3_topo.py` и сравним результат (рис. 4.7). Видим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.



Screenshot of a Windows desktop environment showing a terminal window with network configuration logs. The terminal window title is "mininet@mininet-vm: ~/work/lab\_iperf3/lab\_iperf3\_topo". The log output shows two runs of the script: one with parameters and one without. Both runs show the creation of a controller, hosts, switch, and links, followed by starting the network and waiting for switches to connect. The second run shows different host IP and MAC addresses.

```
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 5000000/100000us) h2 (cfs 4500000/100000us)
*** 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 be:ee:c4:db:70:40
Host h2 has IP address 10.0.0.2 and MAC address ba:7d:c4:32:a7:44
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
(cfs -1/100000us) (cfs -1/100000us) *** 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 1a:d4:5a:fb:c9:0d
Host h2 has IP address 10.0.0.2 and MAC address fe:c0:5a:b8:65:d5
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 4.7: Запуск скрипта с настройкой параметров производительности и без нее

Сделаем копию скрипта lab\_iperf3\_topo2.py и поместим его в подкаталог iperf (рис. 4.8).

```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ cp lab_iperf3_topo2.py lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ mkdir -p ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ mv ~/work/lab_iperf3/lab_iperf3_topo/lab_iperf3.py ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ cd ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -l
total 24
-rw-rw-r-- 1 mininet mininet 976 Oct  9 08:52 iperf.csv
-rw-r--r-- 1 root    root    7779 Oct  9 08:52 iperf_result.json
-rwxrwxr-x 1 mininet mininet 1345 Oct 11 08:25 lab_iperf3.py
-rw-rw-r-- 1 mininet mininet 179 Oct  9 08:51 Makefile
drwxrwxr-x 2 mininet mininet 4096 Oct  9 08:52 results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ rm -rf iperf.csv iperf_result.json Makefile results/
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -l
total 4
-rwxrwxr-x 1 mininet mininet 1345 Oct 11 08:25 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Рис. 4.8: Создание копии скрипта lab\_iperf3\_topo2.py

В начале скрипта lab\_iperf3.py добавим запись об импорте time и изменим код в скрипте так, чтобы (рис. 4.9): - 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

Screenshot of a terminal window titled "mininet@mininet-vm: ~/work/lab\_iperf3/iperf3". The window displays Python code for creating a network using the Mininet library and generating traffic with iperf3.

```
GNU nano 4.8          lab_iperf3.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.
"""

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( "*** 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() )

[Goto Line] [Wrote 56 lines]
^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
```

Рис. 4.9: Изменен ия кода в скрипте lab\_iperf3.py

Запустим на отработку скрипт lab\_iperf3.py (рис. 4.10).

```
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
*** Starting network
*** 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. 4.10: Запуск скрипта lab\_iperf3.py

Построим графики из получившегося JSON-файла (рис. 4.12). Создадим Makefile для проведения всего эксперимента. В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 4.11).

The screenshot shows a terminal window titled "mininet@mininet-vm: ~/work/lab\_iperf3/iperf3". The window contains a terminal session where a Makefile is being edited in the nano text editor. The terminal shows the following content:

```
GNU nano 4.8                                     Makefile                                     Modified ^

all: iperf_result.json plot

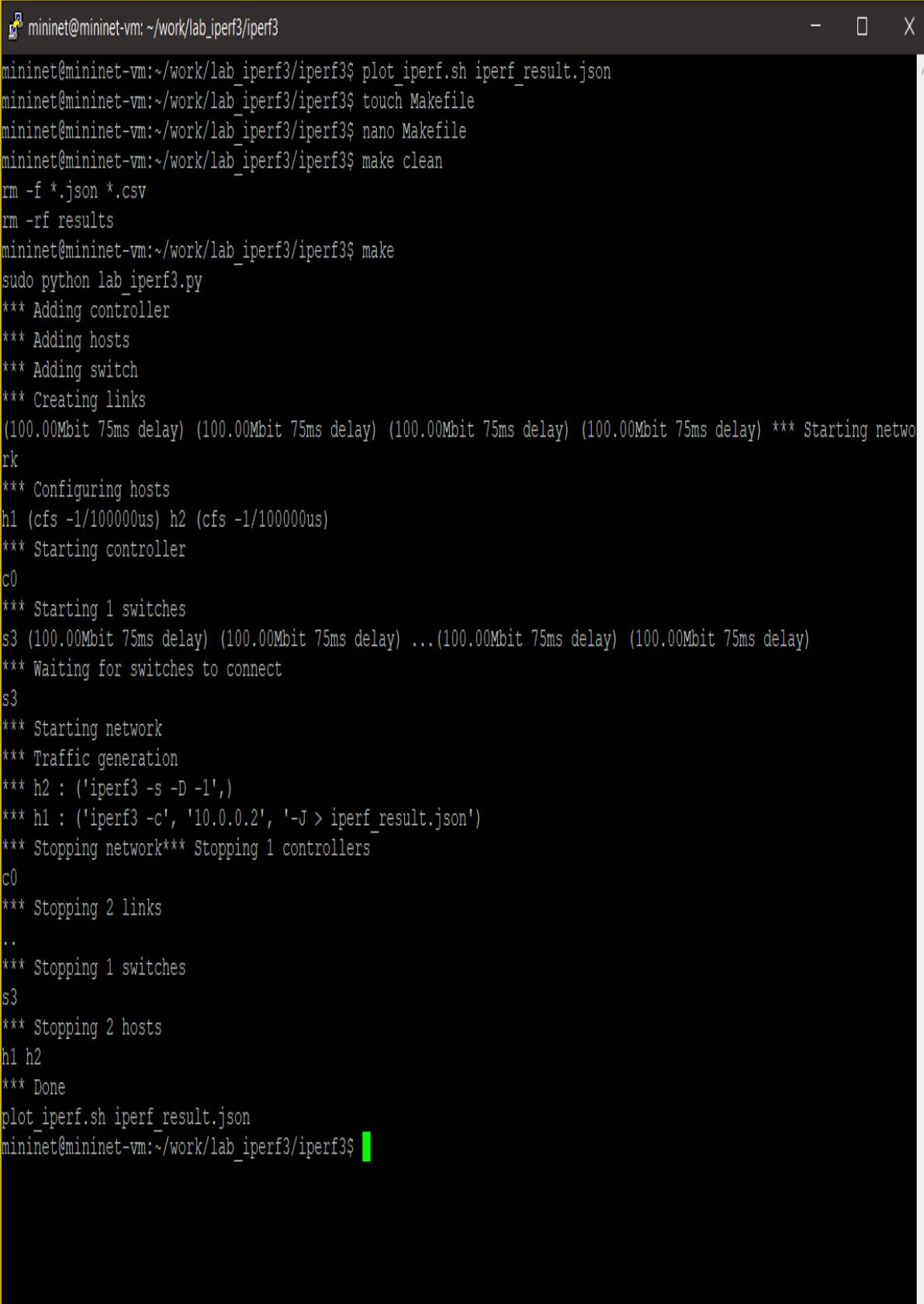
iperf_result.json:
    sudo python lab_iperf3.py

plot: iperf_result.json
    plot_iperf.sh iperf_result.json

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

Рис. 4.11: Создание Makefile

Проверьте корректность отработки Makefile (рис. 4.12).



```
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$ 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 -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
*** Starting network
*** 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
plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Рис. 4.12: Проверка работы Makefile

## **5 Выводы**

В ходе выполнения лабораторной работы я познакомился 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.

## **Список литературы**

1. Mininet [Электронный ресурс]. Mininet Project Contributors. URL: <https://mininet.org/> (дата обращения: 07.10.2025).
2. Iperf [Электронный ресурс]. iPerf - The ultimate speed test tool for TCP, UDP; SCTP. URL: <https://iperf.fr/> (дата обращения: 07.10.2025).