

# **Лабораторная работа №3**

**Моделирование сетей передачи данных**

Еюбоглу Тимур

# Содержание

0.1	Цель лабораторной работы . . . . .	4
<b>1</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>2</b>	<b>Вывод</b>	<b>22</b>
2.1	Вывод . . . . .	22
<b>3</b>	<b>Список литературы. Библиография</b>	<b>23</b>

# Список иллюстраций

1.1	Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet) . .	5
1.2	Открытие файла lab_iperf3_topo.py . . . . .	6
1.3	Запуск скрипта создания топологии и дальнейший просмотр элементов . . . . .	7
1.4	Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес) . . . . .	8
1.5	Проверка корректности отработки скрипта . . . . .	9
1.6	Внесение изменения в скрипт, позволяющего вывести на экран информацию о двух хостах (имя, IP-адрес, MAC-адрес) . . . . .	10
1.7	Проверка корректности отработки скрипта . . . . .	11
1.8	Создание копии скрипта lab_iperf3_topo.py . . . . .	11
1.9	Изменение скрипта lab_iperf3_topo2.py: добавление импорта классов, изменение строки описания сети, изменение функции задания параметров виртуального хоста h1 и h2, изменение функции параметров соединения между хостом h1 и коммутатором s3 . . . . .	13
1.10	Запустим на отработку скрипт lab_iperf3_topo2.py . . . . .	14
1.11	Создание копии скрипта lab_iperf3_topo2.py и его дальнейшее помещение в подкаталог iperf . . . . .	15
1.12	Добавление в скрипт lab_iperf3.py записи об импорте time; снятие ограничений по использованию ресурсов процессора; добавление кода, чтобы каналы между хостами и коммутатором были по 100 Мбит/с с задержкой 75 мс, без потерь . . . . .	16
1.13	Описание запуска на хосте h2 сервера iPerf3, на хосте h1 запуска с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл. Комментирование строк, отвечающих за запуск CLI-интерфейса . . . . .	18
1.14	Запуск скрипта lab_iperf3.py на отработку . . . . .	19
1.15	Построение графиков и создание Makefile для проведения всего эксперимента . . . . .	19
1.16	Добавление скрипта в Makefile . . . . .	20
1.17	Проверка корректности отработки Makefile . . . . .	21

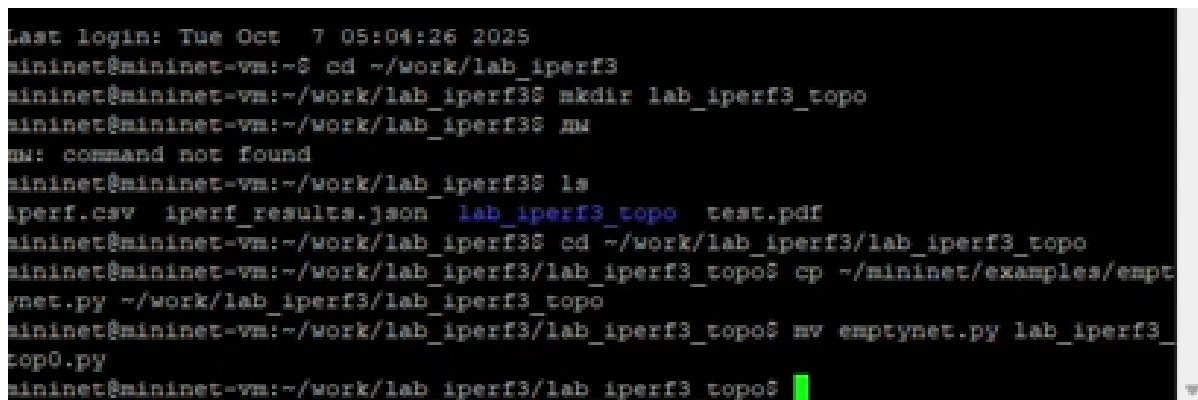
# Список таблиц

## 0.1 Цель лабораторной 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.

# 1 Выполнение лабораторной работы

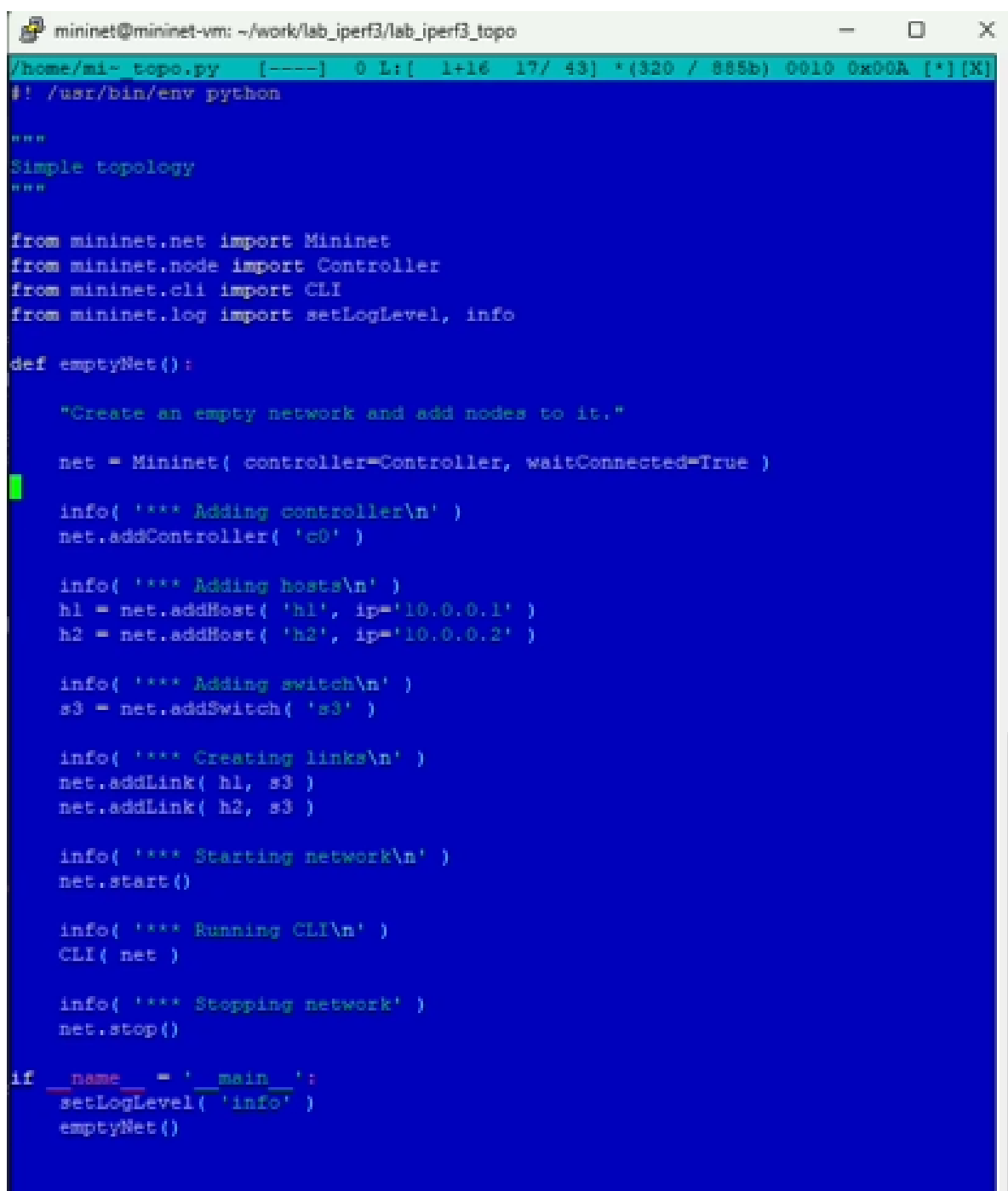
С помощью API Mininet создадим простейшую топологию сети, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8. Для этого в каталоге /work/lab\_iperf3 для работы над проектом создадим подкаталог lab\_iperf3\_topo и скопируем в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet (рис. 1.1):



```
Last login: Tue Oct  7 05:04:26 2025
mininet@mininet-vm:~$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ mkdir lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3$ mv
mv: command not found
mininet@mininet-vm:~/work/lab_iperf3$ ls
iperf.csv  iperf_results.json  lab_iperf3_topo  test.pdf
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/empt
ynet.py ~/work/lab_iperf3/lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_
top0.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$
```

Рис. 1.1: Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet)

Изучим содержание скрипта lab\_iperf3\_topo.py (рис. 1.2):



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/min- topo.py [-----] 0 L:[ 1+16 17/ 43] *(320 / 885b) 0010 0x00A [*][X]
#!/usr/bin/env python

'''
Simple topology
'''

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. 1.2: Открытие файла lab\_iperf3\_topo.py

Запустим скрипт создания топологии lab\_iperf3\_topo.py. После отработки скрипта посмотрим элементы топологии и завершим работу mininet (рис. 1.3):

```

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=877>
<Host h2: h2-eth0:10.0.0.2 pid=881>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=886>
<Controller c0: 127.0.0.1:6653 pid=870>
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. 1.3: Запуск скрипта создания топологии и дальнейший просмотр элементов

Следующим шагом внесём 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, добавим нужную строку (рис. 1.4):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mininet/topo.py [-----] 0 L: [ 1+35 36/ 45] *(808 / 972b) 0010 0x00A [*][X]
#!/usr/bin/env python

"""
Simple topology
"""

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 )

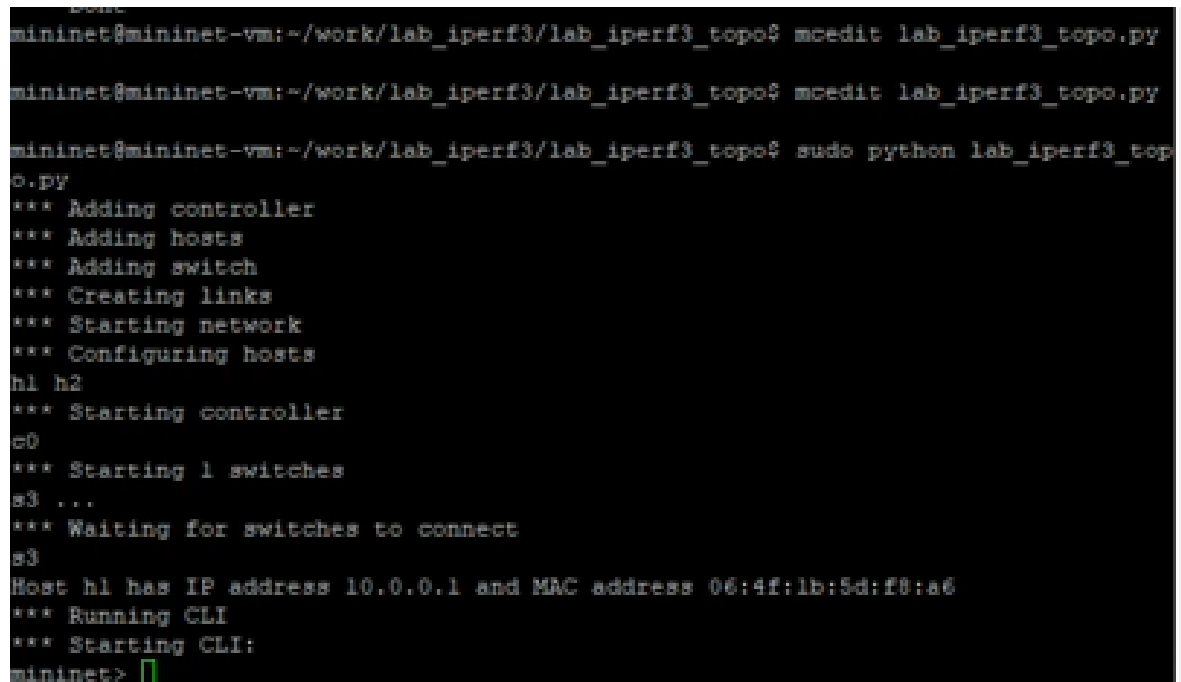
    info( '*** Stopping network\n' )
    net.stop()

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

Рис. 1.4: Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес)



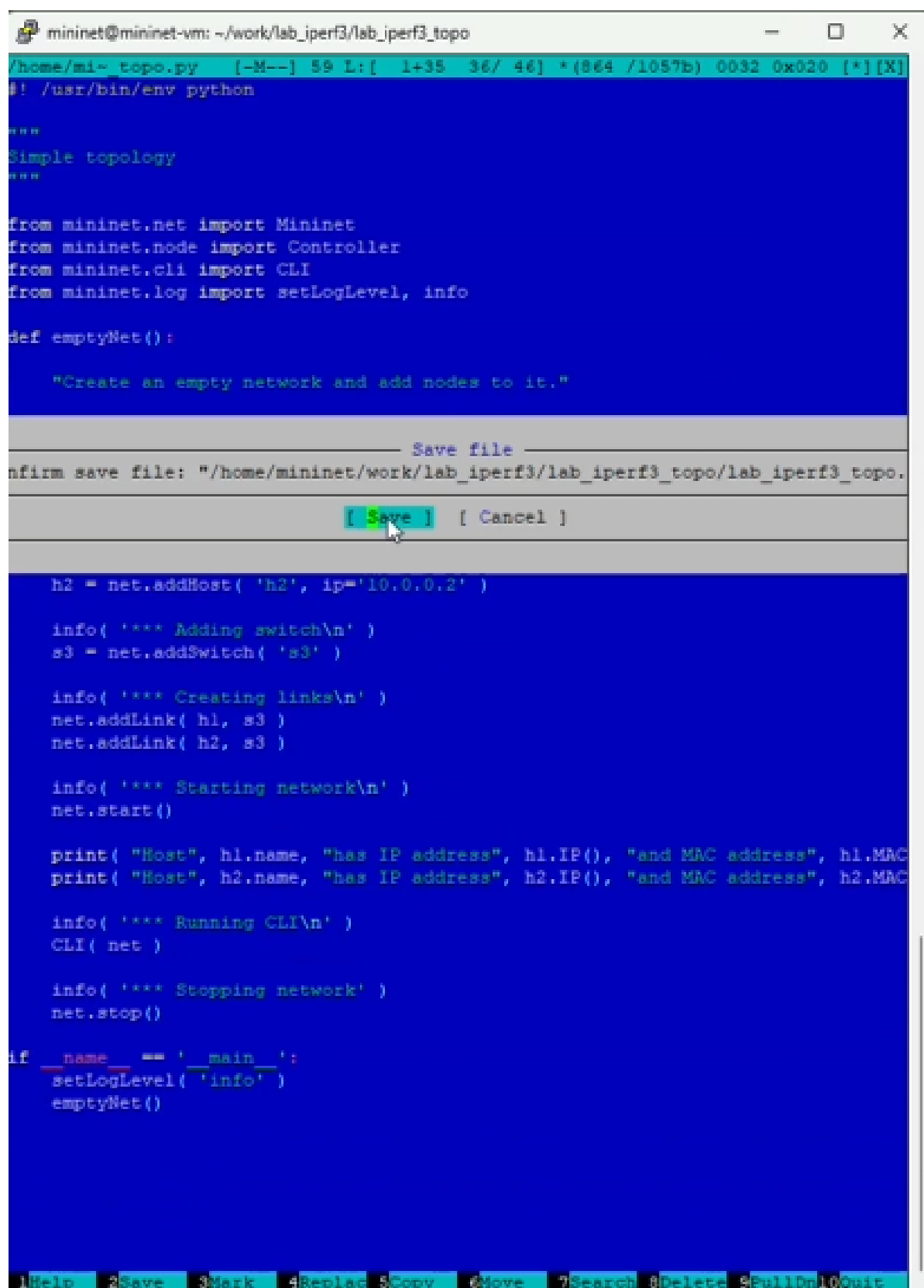
Проверим корректность отработки изменённого скрипта (рис. 1.5):



```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mcedit lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mcedit 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
Host h1 has IP address 10.0.0.1 and MAC address 06:4f:1b:5d:f8:a6
*** Running CLI
*** Starting CLI:
mininet> █
```

Рис. 1.5: Проверка корректности отработки скрипта

Затем изменим скрипт `lab_iperf3_topo.py` так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети и проверим корректность отработки изменённого скрипта (рис. 1.6 - рис. 1.7)



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mininet-vm/work/lab_iperf3/lab_iperf3_topo/lab_iperf3_topo.py [-M--] 59 L: [ 1+35 36/ 46] *(864 /1057b) 0032 0x020 [*][X]
#!/usr/bin/env python

"""
Simple topology
"""

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."

    h1 = net.addHost( 'h1', ip='10.0.0.1' )

    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\n' )
    net.stop()

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

Save file

Confirm save file: "/home/mininet/work/lab\_iperf3/lab\_iperf3\_topo/lab\_iperf3\_topo.py"

[ Save ] [ Cancel ]

1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9FullEdit 10Quit

Рис. 1.6: Внесение изменения в скрипт, позволяющего вывести на экран информацию о двух хостах (имя, IP-адрес, MAC-адрес)

```

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 9a:f0:c2:c3:08:90
Host h2 has IP address 10.0.0.2 and MAC address aa:30:35:9b:85:a9
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 1.7: Проверка корректности отработки скрипта

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов `CPUimitedHost` и `TCLink`. Добавим в скрипт настройки параметров производительности. Для начала сделаем копию скрипта `lab_iperf3_topo.py` (рис. 1.8):

```

*** 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$ cp lab_iperf3_topo.py lab_iperf3_topo2.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$

```

Рис. 1.8: Создание копии скрипта `lab_iperf3_topo.py`

В начале скрипта `lab_iperf3_topo2.py` добавим записи об импорте классов `CPUimitedHost` 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. 1.9):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mi~topo2.py [-M--] 80 L:[ 1+29 30/ 47] *(795 /1225b) 0116 0x074 [*][X]
in/env python

pology

net.net import Mininet
net.node import Controller, CPULimitedHost
net.cli import CLI
net.log import setLogLevel, info
net.link import TCLink

Net():

te an empty network and add nodes to it."

Mininet( controller=Controller, waitConnected=True, host = CPULimitedHost, link

    '*** Adding controller\n' )
addController( 'c0' )

    '*** Adding hosts\n' )
net.addHost( 'h1', ip='10.0.0.1', cpu=50 )
net.addHost( 'h2', ip='10.0.0.2', cpu=45 )

    '*** Adding switch\n' )
net.addSwitch( 's3' )

    '*** Creating links\n' )
addLink( h1, s3, bw=10, delay='5ms', max_queue_size=1000, loss=10, use_hb=True )
addLink( h2, s3 )

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

( "Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC() )
( "Host", h2.name, "has IP address", h2.IP(), "and MAC address", h2.MAC() )

    '*** Running CLI\n' )
net )

    '*** Stopping network' )
stop()

__ == '__main__':
    gLevel( 'info' )
    Net()
```

Рис. 1.9: Изменение скрипта lab\_iperf3\_topo2.py: добавление ипорта классов, изменение строки описания сети, изменение функции задания параметров виртуального хоста h1 и h2, изменение функции параметров соединения между хостом h1 и коммутатором s3

Запустим на отработку скрипт lab\_iperf3\_topo2.py (рис. 1.10):

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mcedit lab_iperf3_topo2.py
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 10.000000% loss) (10.00Mbit 5ms delay 10.000000% loss) *** Starting network
*** Configuring hosts
h1 (cfs 5000000/1000000us) h2 (cfs 4500000/1000000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (10.00Mbit 5ms delay 10.000000% loss) ... (10.00Mbit 5ms delay 10.000000% loss)
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address 26:00:4e:2c:93:7e
Host h2 has IP address 10.0.0.2 and MAC address 76:0f:16:13:fe:94
*** 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$
```

Рис. 1.10: Запустим на отработку скрипт lab\_iperf3\_topo2.py

Перед завершением лабораторной работы, построим графики по проводимому эксперименту. Для этого сделаем копию скрипта lab\_iperf3\_topo2.py и поместим его в подкаталог iperf (рис. 1.11):

```

*** Stopping 2 hosts
h1 h2
*** Done
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
-rw-rw-r-- 1 mininet mininet 1224 Oct  7 08:46 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ █

```

Рис. 1.11: Создание копии скрипта lab\_iperf3\_topo2.py и его дальнейшее помещение в подкаталог iperf

В начале скрипта lab\_iperf3.py добавим запись об импорте time и изменим код в скрипте так, чтобы (рис. 1.12): - 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

```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
/home/mi-perf3.py [-M--] 36 L:[ 1+33 34/ 48] *(868 /1225b) 0010 0x00A [*][X]
#!/usr/bin/env python

"""
Simple topology
"""

from mininet.net import Mininet
from mininet.node import Controller, CPULimitedHost
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink
import time

def emptyNet():

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

    net = Mininet( controller=Controller, waitConnected=True, host = CPULimitedHost

    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, cls=TCLink, bw=100, delay='75ms' )
    net.addLink( h2, s3, cls=TCLink, bw=100, delay='75ms' )

    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__':
    setLogLevel( 'info' )
    emptyNet()
```

Рис. 1.12: Добавление в скрипт lab\_iperf3.py записи об импорте time; снятие ограничений по использованию ресурсов процессора; добавление кода, чтобы каналы между хостами и коммутатором были по 100 Мбит/с с задержкой 75 мс, без потерь 16



После функции старта сети опишем запуск на хосте h2 сервера iPerf3, а на хосте h1 запуск с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл, прокомментируем строки, отвечающие за запуск CLI-интерфейса (рис. 1.13):

```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
/home/mi-perf3.py [-----] 65 L: [ 1+39 40/ 50] *(1085/1255b) 0010 0x00A [*][X]
#!/usr/bin/env python

"""
Simple topology
"""

from mininet.net import Mininet
from mininet.node import Controller, CPULimitedHost
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink
import time

def emptyNet():

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

    net = Mininet( controller=Controller, waitConnected=True, host = CPULimitedH

    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, cls=TCLink, bw=100, delay='75ms' )
    net.addLink( h2, s3, cls=TCLink, 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' )

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

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

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

Рис. 1.13: Описание запуска на хосте h2 сервера iPerf3, на хосте h1 запуска с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл. Комментирование строк, отвечающих за запуск CLI-интерфейса

Запустим на отработку скрипт lab\_iperf3.py (рис. 1.14):

```
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) (100.00M
bit 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) (1
00.00Mbit 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
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Рис. 1.14: Запуск скрипта lab\_iperf3.py на отработку

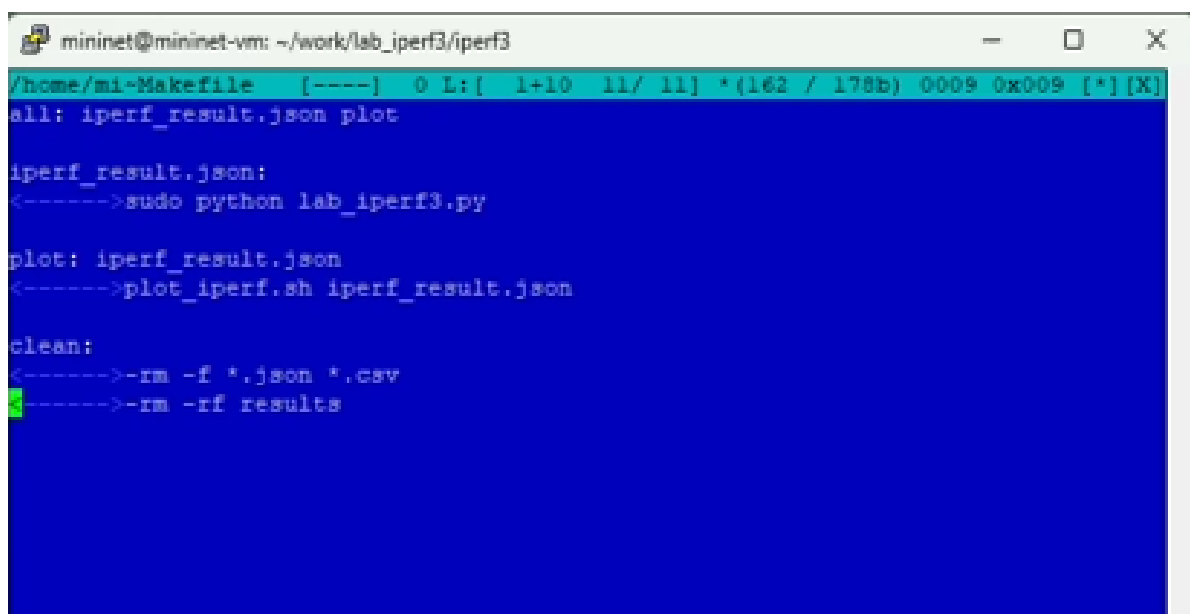
Построим графики из получившегося JSON-файла и создадим Makefile для проведения всего эксперимента (рис. 1.15):

```
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
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$
```

Рис. 1.15: Построение графиков и создание Makefile для проведения всего эксперимента

В Makefile пропишем запуск скрипта эксперимента, построение графиков и

очистку каталога от результатов (рис. 1.16):



```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
/home/mi-Makefile [----] 0 L:[ 1+10 11/ 11] *(162 / 178b) 0009 0x009 [*]{X}
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. 1.16: Добавление скрипта в Makefile

Проверим корректность отработки Makefile (рис. 1.17):

```

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) (100.00M
bit 75ms delay) *** Starting network
*** Configuring hosts
h1 (cfs -l/1000000us) h2 (cfs -l/1000000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) ... (100.00Mbit 75ms delay) (1
00.00Mbit 75ms delay)
*** Waiting for switches to connect
s3
*** Traffic generation
*** h2 : ('iperf3 -s -D -l',)
*** 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. 1.17: Проверка корректности отработки Makefile

## **2 Вывод**

### **2.1 Вывод**

В ходе выполнения лабораторной работы познакомились с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получили навыки проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet

### **3 Список литературы. Библиография**

[1] Mininet: <https://mininet.org/>