

Отчёт по лабораторной работе №3

Моделирование сетей передачи данных

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

Выполнила: Коннова Татьяна Алексеевна,
НПИбд-01-22, 1132221814

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	20
4	Список литературы. Библиография	21

Список иллюстраций

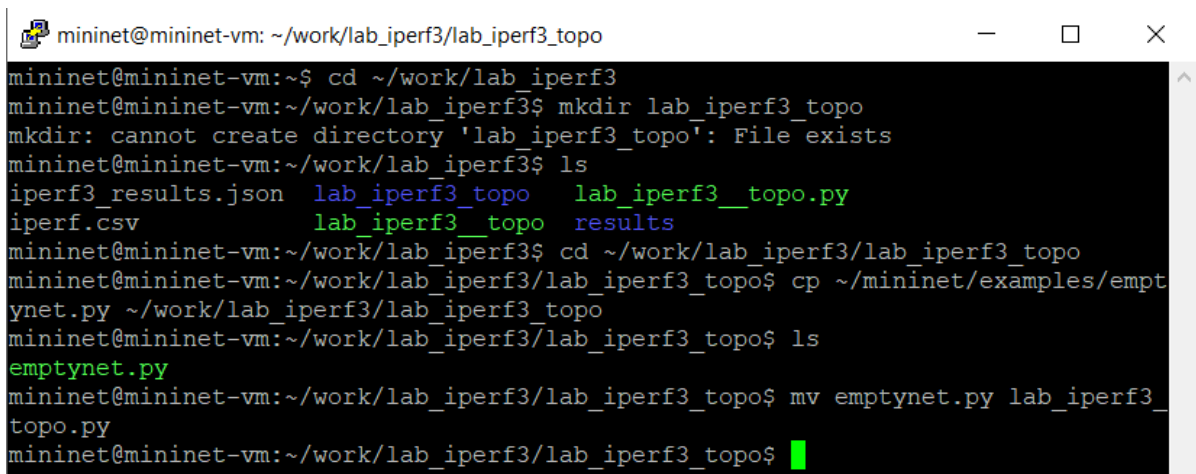
2.1	Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet) .	5
2.2	Открытие файла lab_iperf3_topo.py	6
2.3	Запуск скрипта создания топологии и дальнейший просмотр элементов	7
2.4	Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес)	8
2.5	Проверка корректности отработки скрипта	9
2.6	Внесение изменения в скрипт, позволяющего вывести на экран информацию о двух хостах (имя, IP-адрес, MAC-адрес)	10
2.7	Проверка корректности отработки скрипта	11
2.8	Создание копии скрипта lab_iperf3_topo.py	11
2.9	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
2.10	Запуск скрипта lab_iperf3_topo2.py на отработку	14
2.11	Создание копии скрипта lab_iperf3_topo2.py и его дальнейшее помещение в подкаталог iperf	14
2.12	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	15
2.13	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	16
2.14	Запуск скрипта lab_iperf3.py на отработку	17
2.15	Построение графиков и создание Makefile для проведения всего эксперимента	17
2.16	Добавление скрипта в Makefile	18
2.17	Проверка корректности отработки Makefile	19

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 Выполнение лабораторной работы

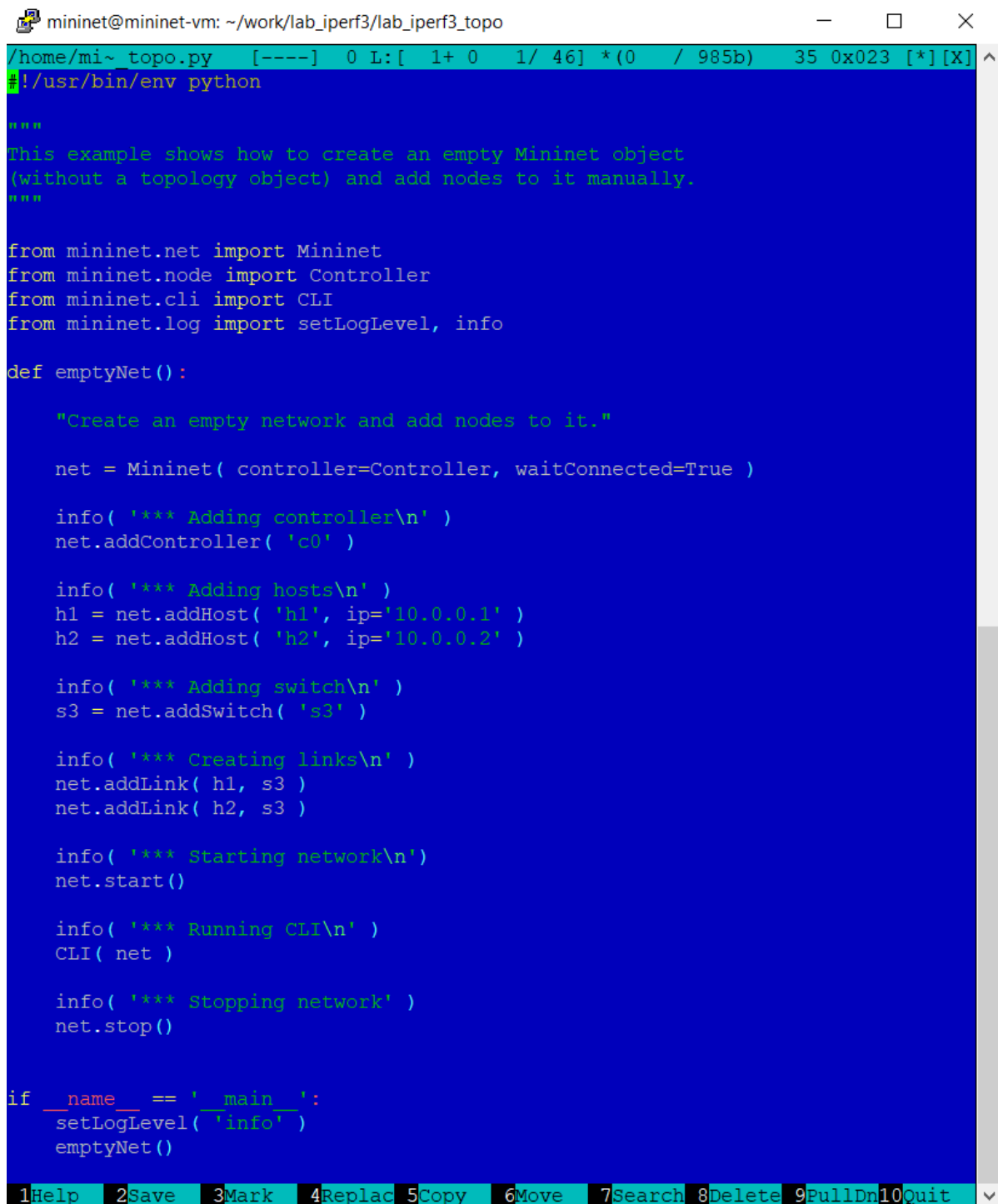
С помощью API Mininet создадим простейшую топологию сети, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8. Для этого в каталоге /work/lab_iperf3 для работы над проектом создадим подкаталог lab_iperf3_topo и скопируем в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet (рис. 2.1):



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
mininet@mininet-vm:~$ 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$ ls
iperf3_results.json  lab_iperf3_topo  lab_iperf3_topo.py
iperf.csv             lab_iperf3_topo  results
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$ ls
emptynet.py
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. 2.1: Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet)

Изучим содержание скрипта lab_iperf3_topo.py (рис. 2.2):



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mi~ topo.py [----] 0 L:[ 1+ 0 1/ 46] *(0 / 985b) 35 0x023 [*][X]
#!/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\n' )
    net.stop()

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

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

Рис. 2.2: 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. 2.3):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
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=823>
<Host h2: h2-eth0:10.0.0.2 pid=827>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=832>
<Controller c0: 127.0.0.1:6653 pid=816>
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. 2.3: 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. 2.4):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mi~ topo.py [-M--] 35 L:[ 1+35 36/ 48] *(852 /1071b) 32 0x020 [*][X]
#!/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

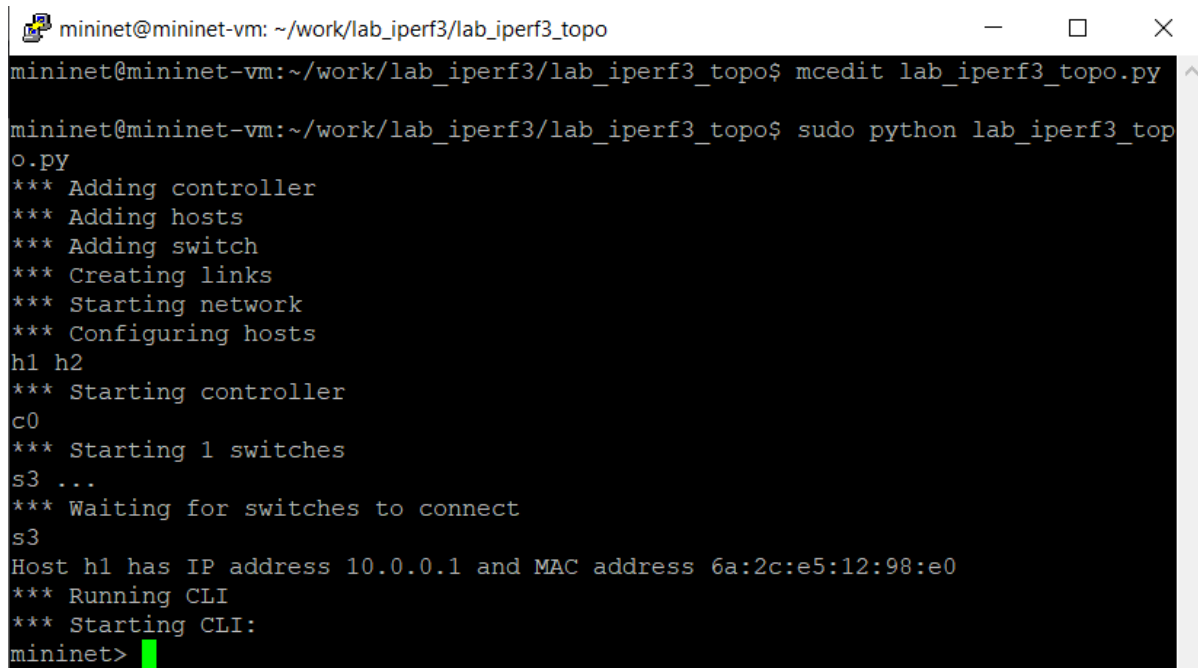
    info( '*** Running CLI\n' )
    CLI( net )

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

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

Рис. 2.4: Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес)

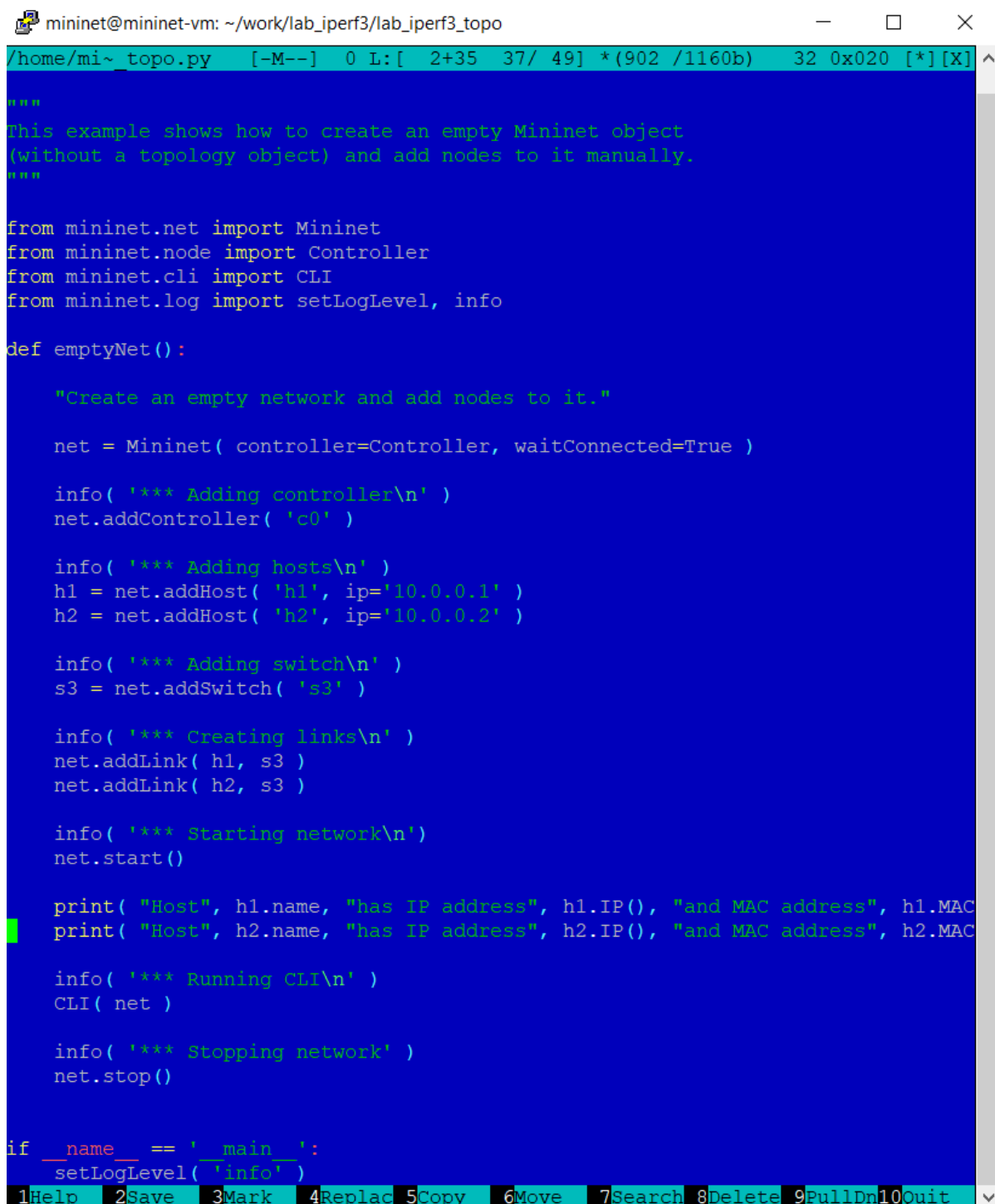
Проверим корректность отработки изменённого скрипта (рис. 2.5):

A terminal window titled 'mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo'. The user enters 'mcedit lab_iperf3_topo.py' and then 'sudo python lab_iperf3_topo.py'. The script outputs several status messages: '*** 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', and 'Host h1 has IP address 10.0.0.1 and MAC address 6a:2c:e5:12:98:e0'. It then shows '*** Running CLI' and '*** Starting CLI:'. The prompt 'mininet>' is visible at the bottom with a green cursor.

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
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 6a:2c:e5:12:98:e0
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 2.5: 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. 2.6 - рис. 2.7):



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mininet_topo.py [-M--] 0 L:[ 2+35 37/ 49] *(902 /1160b) 32 0x020 [*][X]
"""
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 )

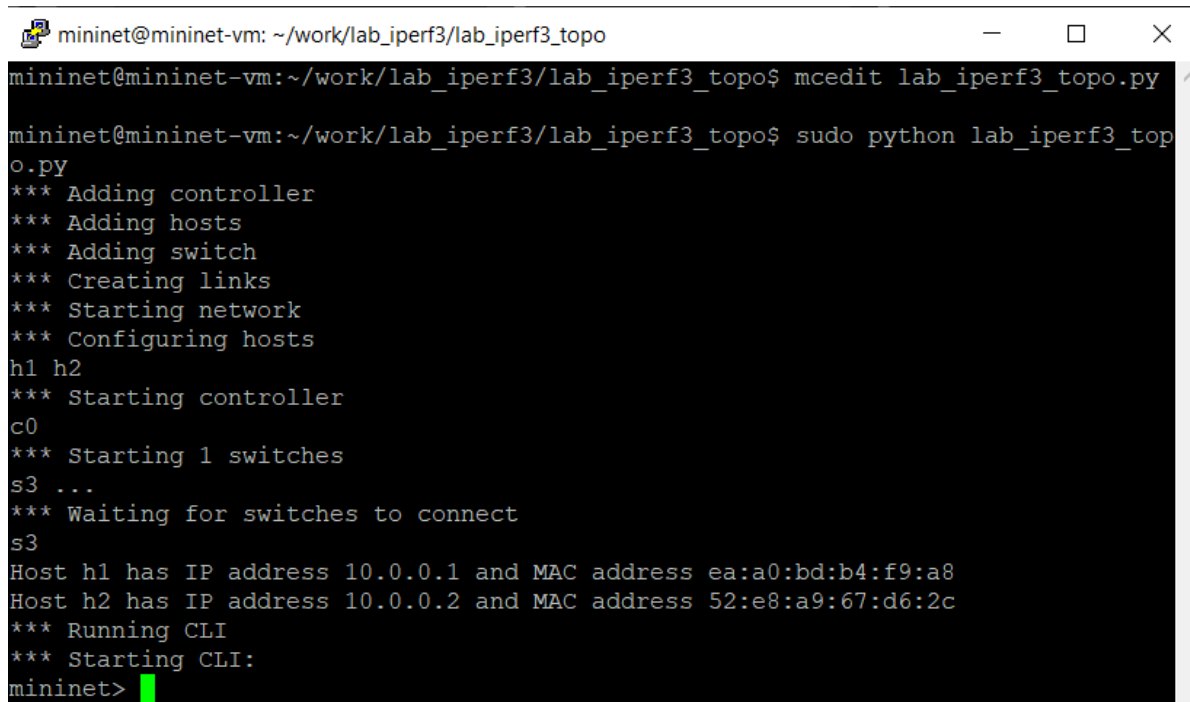
    info( '*** Running CLI\n' )
    CLI( net )

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

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

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

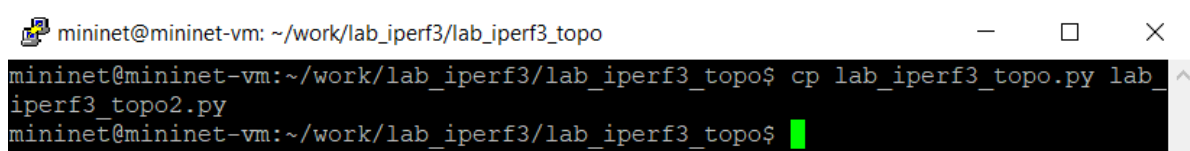
Рис. 2.6: Внесение изменения в скрипт, позволяющего вывести на экран информацию о двух хостах (имя, IP-адрес, MAC-адрес)

A terminal window titled 'mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo'. The user enters 'mcedit lab_iperf3_topo.py' and then 'sudo python lab_iperf3_topo.py'. The script output shows: '*** 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:a0:bd:b4:f9:a8', 'Host h2 has IP address 10.0.0.2 and MAC address 52:e8:a9:67:d6:2c', '*** Running CLI', '*** Starting CLI:', and finally 'mininet>' with a green cursor.

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
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 ea:a0:bd:b4:f9:a8
Host h2 has IP address 10.0.0.2 and MAC address 52:e8:a9:67:d6:2c
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 2.7: Проверка корректности отработки скрипта

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` (рис. 2.8):

A terminal window titled 'mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo'. The user enters 'cp lab_iperf3_topo.py lab_iperf3_topo2.py'. The prompt returns to 'mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo\$' with a green cursor.

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
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. 2.8: Создание копии 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. 2.9):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
/home/mi~topo2.py [-M--] 0 L:[ 1+31 32/ 51] *(839 /1354b) 32 0x020 [*][X] ^
#!/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 = CPULimitedH

    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_h
    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

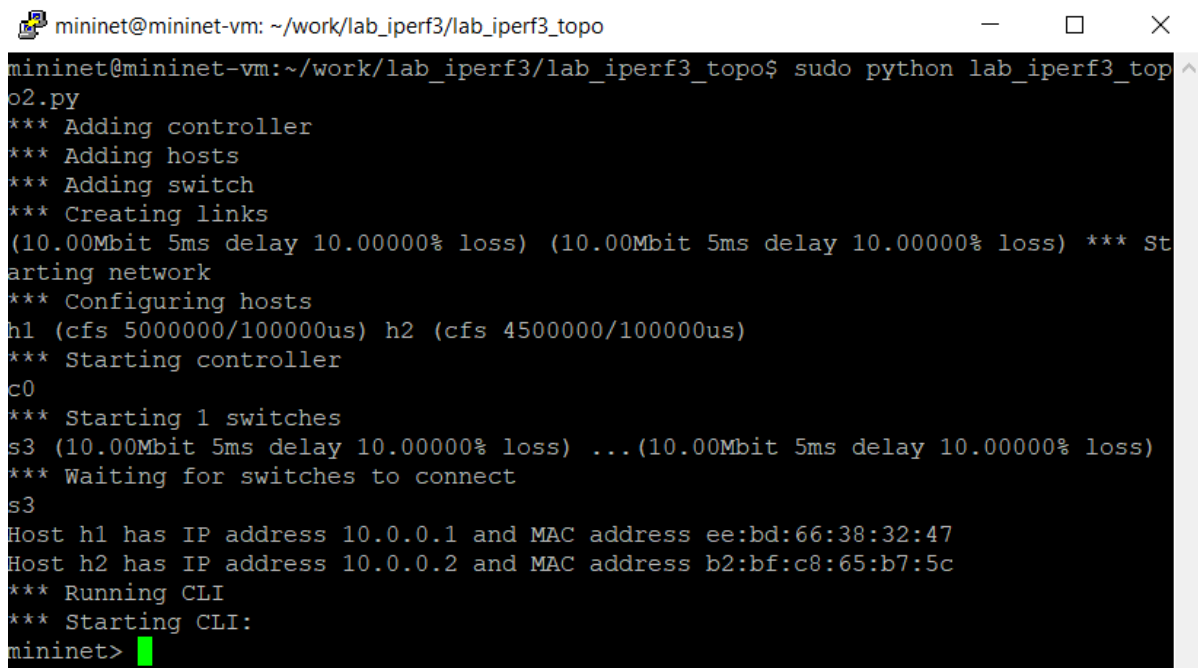
    info( '*** Running CLI\n' )
    CLI( net )

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

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit v
```

Рис. 2.9: 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

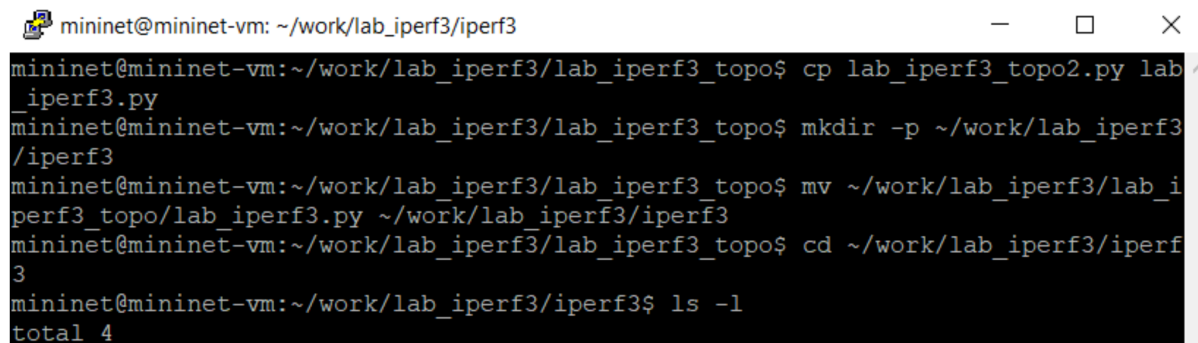
Запустим на отработку скрипт lab_iperf3_topo2.py (рис. 2.10):



```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
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 ee:bd:66:38:32:47
Host h2 has IP address 10.0.0.2 and MAC address b2:bf:c8:65:b7:5c
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 2.10: Запуск скрипта lab_iperf3_topo2.py на отработку

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. 2.11):

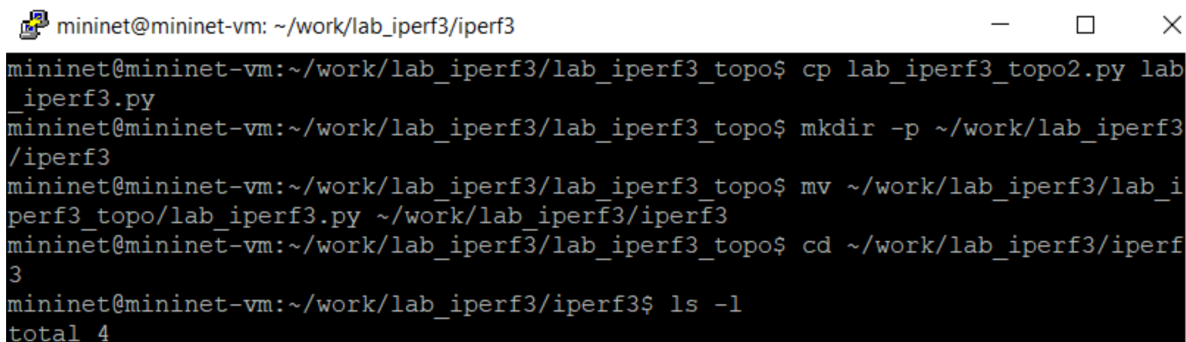


```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cp lab_iperf3_topo2.py lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mkdir -p ~/work/lab_iperf3/iperf3
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv ~/work/lab_iperf3/lab_iperf3_topo/lab_iperf3.py ~/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
```

Рис. 2.11: 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. 2.12): - на хостах не было ограничения по использованию ресурсов процессора; - каналы между хостами и коммутатором были

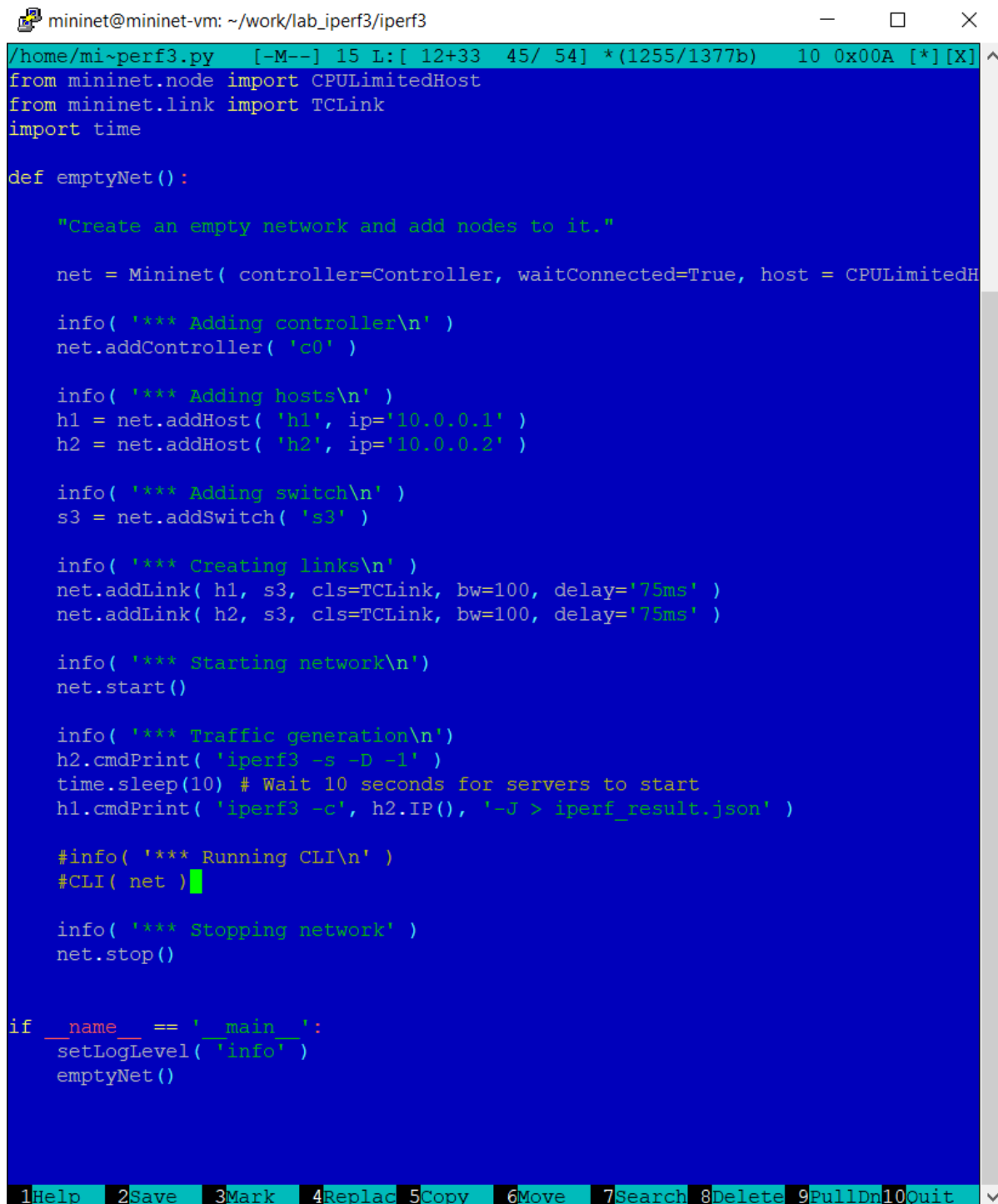
по 100 Мбит/с с задержкой 75 мс, без потерь, без использования ограничителей пропускной способности и максимального размера очереди

A terminal window titled 'mininet@mininet-vm: ~/work/lab_iperf3/iperf3' with standard window controls. The terminal shows a series of commands and their outputs: copying a file, creating a directory, moving a file, changing the directory, and listing files.

```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
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
```

Рис. 2.12: 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

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. 2.13):



```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
/home/mi~perf3.py [-M--] 15 L:[ 12+33 45/ 54] *(1255/1377b) 10 0x00A [*][X]
from mininet.node import CPULimitedHost
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 -l' )
    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. 2.13: 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. 2.14):


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

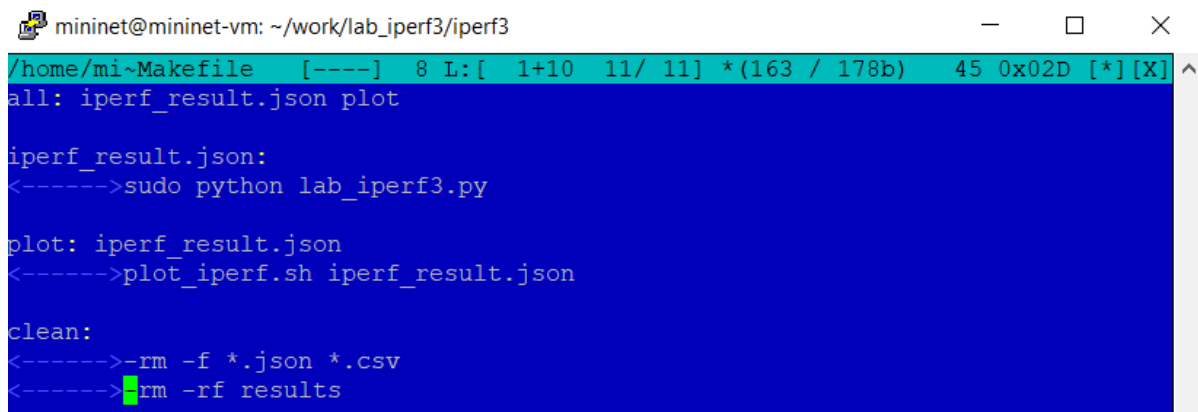
Рис. 2.14: Запуск скрипта lab_iperf3.py на отработку

Построим графики из получившегося JSON-файла и создадим Makefile для проведения всего эксперимента (рис. 2.15):

```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
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. 2.15: Построение графиков и создание Makefile для проведения всего эксперимента

В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 2.16):



The screenshot shows a terminal window with a title bar indicating the user is 'mininet' on a 'mininet-vm' machine, in the directory '~/work/lab_iperf3/iperf3'. The terminal content displays a Makefile with the following rules:

```
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
```

The terminal shows the execution of the 'all' target, which runs 'sudo python lab_iperf3.py' to generate 'iperf_result.json' and then 'plot_iperf.sh iperf_result.json' to generate the 'plot' target. The 'clean' target is also shown, which removes '*.json', '*.csv', and the 'results' directory.

Рис. 2.16: Добавление скрипта в Makefile

Проверим корректность отработки Makefile (рис. 2.17):

```

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

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ █

```

Рис. 2.17: Проверка корректности отработки Makefile

3 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

4 Список литературы. Библиография

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