

# **Отчёт по лабораторной работе №3**

**Дисциплина: Моделирование сетей передачи данных**

**Боровиков Даниил Александрович НПИбд-01-22**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

3.1	Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet) . . .	7
3.2	Открытие файла lab_iperf3_topo.py . . . . .	8
3.3	Запуск скрипта создания топологии и дальнейший просмотр элементов . . . . .	9
3.4	Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес) . . . . .	10
3.5	Проверка корректности отработки скрипта . . . . .	11
3.6	Внесение изменения в скрипт, позволяющего вывести на экран информацию о двух хостах (имя, IP-адрес, MAC-адрес) . . . . .	12
3.7	Проверка корректности отработки скрипта . . . . .	13
3.8	Создание копии скрипта lab_iperf3_topo.py . . . . .	13
3.9	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 . . . . .	14
3.10	Запуск скрипта lab_iperf3_topo2.py на отработку . . . . .	15
3.11	Создание копии скрипта lab_iperf3_topo2.py и его дальнейшее помещение в подкаталог iperf . . . . .	15
3.12	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
3.13	Описание запуска на хосте h2 сервера iPerf3 [1], на хосте h1 запуска с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл. Комментирование строк, отвечающих за запуск CLI-интерфейса . . . . .	17
3.14	Запуск скрипта lab_iperf3.py на отработку . . . . .	18
3.15	Построение графиков и создание Makefile для проведения всего эксперимента . . . . .	18
3.16	Добавление скрипта в Makefile . . . . .	19
3.17	Проверка корректности отработки Makefile . . . . .	20
3.18	Содержимое папки results . . . . .	20
3.19	Сохранение папки results . . . . .	21

## **Список таблиц**

# 1 Цель работы

[1].

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

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/emphynet.py, описывающего стандартную простую топологию сети mininet (рис. 3.1):

```
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 69fa6826576af937fcef92dc91d92ad
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 69fa6826576af937fcef92dc91d92ad
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 69fa6826576af937fcef92dc91d92ad
root@mininet-vm:~# logout
mininet@mininet-vm:~$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ mkdir lab_iperf3_topo
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/emphynet.py ~/work/lab_iperf3/lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emphynet.py lab_iperf3_topo.py
```

Рис. 3.1: Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet)

Изучим содержание скрипта lab\_iperf3\_topo.py (рис. 3.2):

```
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\n' )
    net.stop()

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

Рис. 3.2: 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. 3.3):



```

mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano 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
*** 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=757>
<Host h2: h2-eth0:10.0.0.2 pid=761>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=766>
<Controller c0: 127.0.0.1:6653 pid=750>
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. 3.3: 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. 3.4):

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

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

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

Рис. 3.4: Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес)

Проверим корректность отработки изменённого скрипта (рис. 3.5):

```
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 f6:e7:31:79:be:3b
*** Running CLI
*** Starting CLI:
mininet> |
```

9 октября 2025 г.  
Чт 18:25 (Местное в

Рис. 3.5: 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. 3.6 - рис. 3.7):

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

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To       Свернуть все окна
```

Рис. 3.6: 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 ce:5e:26:92:b6:17
Host h2 has IP address 10.0.0.2 and MAC address 0e:4d:b8:1d:0a:55
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 3.7: 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. 3.8):

```

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$ nano lab_iperf3_topo2.py

```

Рис. 3.8: 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. 3.9):

```
mininet@mininet-vm: ~/work, x + v
GNU nano 4.8 lab_iperf3_topo2.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
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\n' )
    net.stop()

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

Рис. 3.9: 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

Запустим на отработку скрипт lab\_iperf3\_topo2.py (рис. 3.10):

```
mininet@mininet-vm: ~/work, X + v
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 5000000/1000000us) h2 (cfs 4500000/1000000us)
*** 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 06:99:7a:69:36:e3
Host h2 has IP address 10.0.0.2 and MAC address 42:da:4c:e3:d1:c5
*** 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$ 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 b6:39:f6:a3:db:93
Host h2 has IP address 10.0.0.2 and MAC address 1e:13:03:13:6f:cc
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
```

Рис. 3.10: Запуск скрипта lab\_iperf3\_topo2.py на отработку

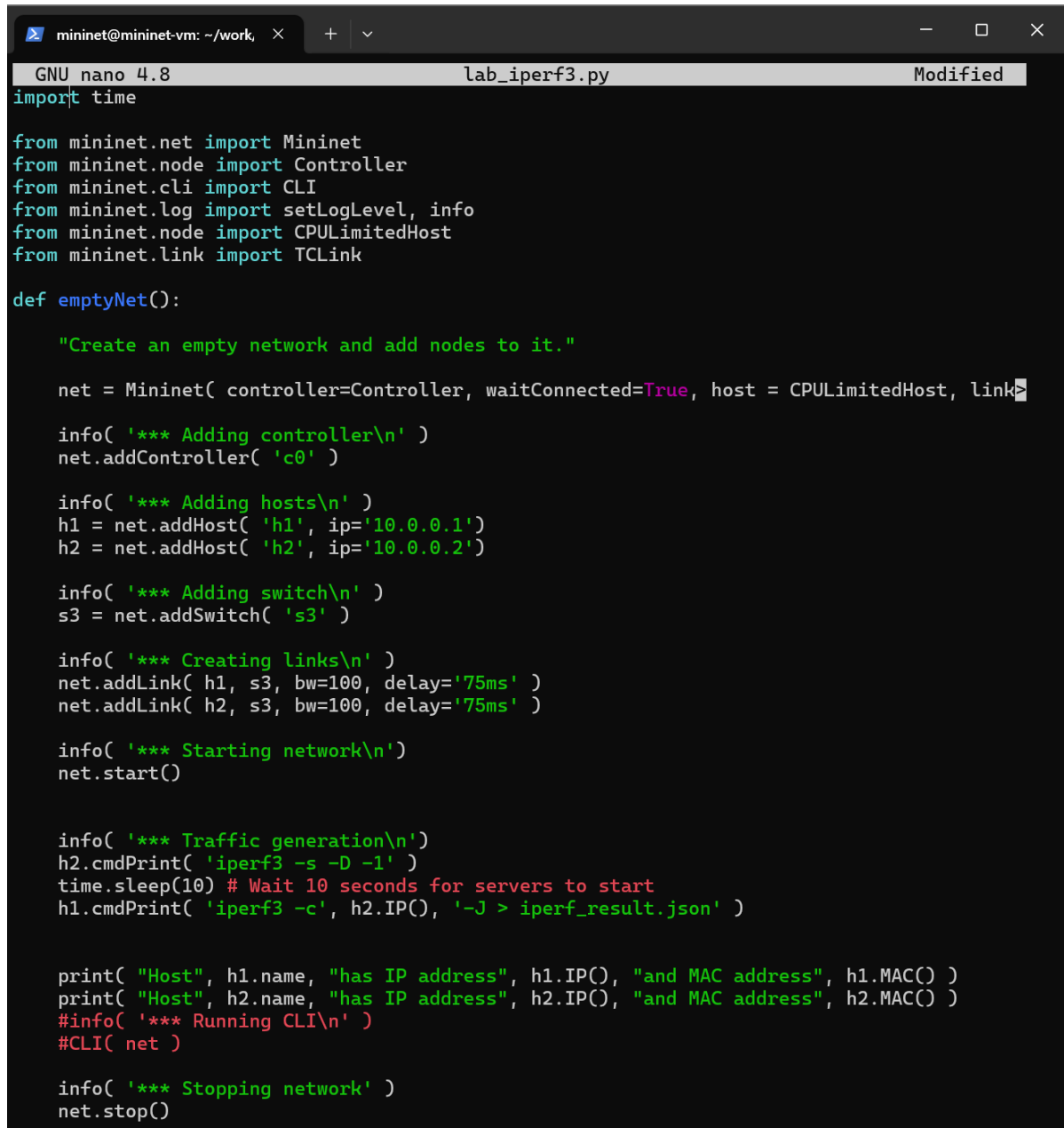
Перед завершением лабораторной работы, построим графики по проводимому эксперименту. Для этого сделаем копию скрипта lab\_iperf3\_topo2.py и поместим его в подкаталог iperf (рис. 3.11):

```
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
-rwxrwxr-x 1 mininet mininet 1345 Oct  9 08:37 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ |
```

Рис. 3.11: Создание копии скрипта lab\_iperf3\_topo2.py и его дальнейшее помеще-  
ние в подкаталог iperf

В начале скрипта lab\_iperf3.py добавим запись об импорте time и изменим код в скрипте так, чтобы (рис. 3.12): - на хостах не было ограничения по использо-  
ванию 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, x + v
GNU nano 4.8 lab_iperf3.py Modified
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=

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

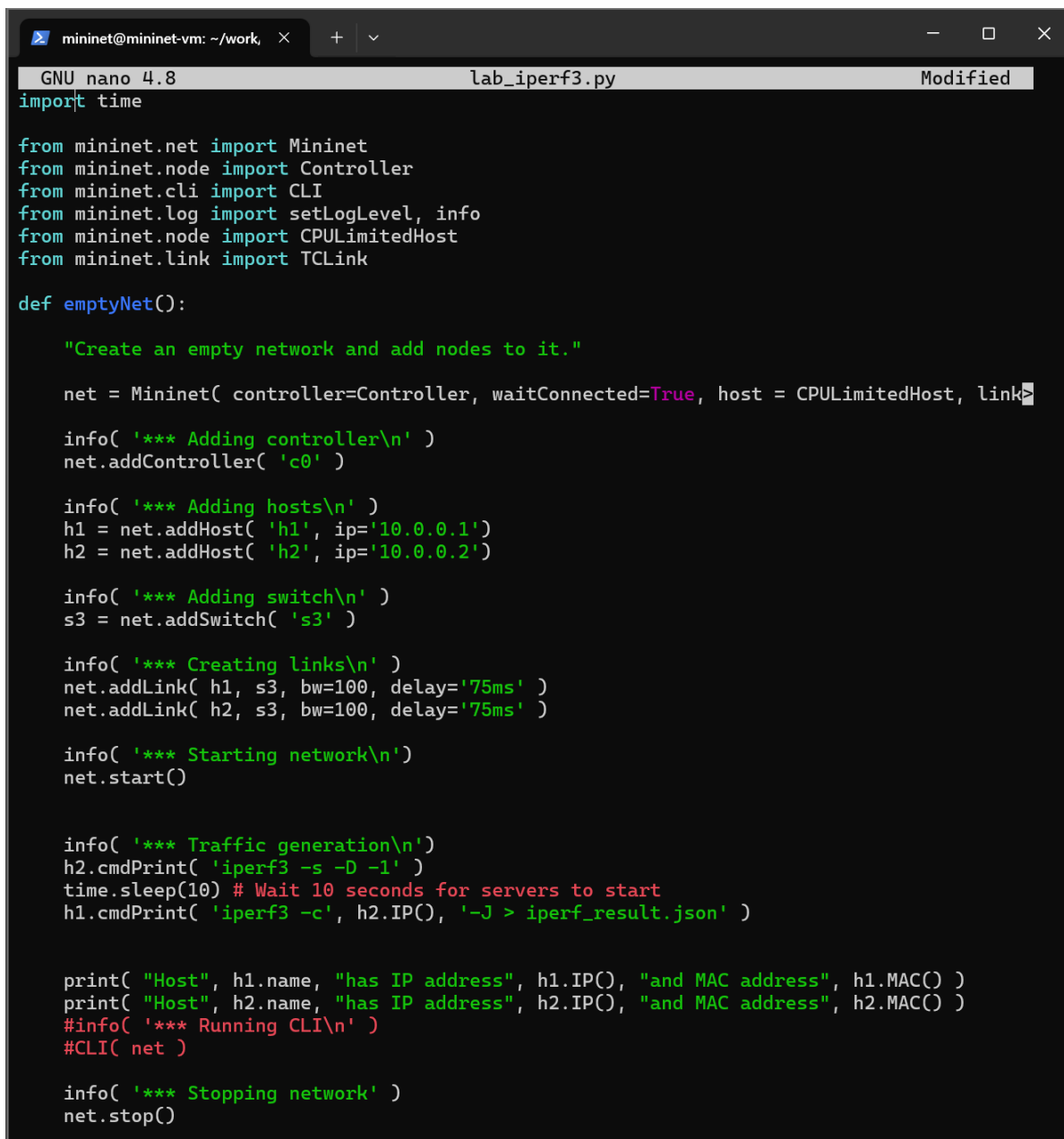
    info( '*** Stopping network' )
    net.stop()
```

Рис. 3.12: 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, закомментируем строки, отвечающие за запуск CLI-интерфейса (рис. 3.13):



```
mininet@mininet-vm: ~/work, × + ▾
GNU nano 4.8 lab_iperf3.py Modified
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=

    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( '*** 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' )

    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()
```

Рис. 3.13: Описание запуска на хосте h2 сервера iPerf3 [1], 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. 3.14):

```

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) (100.00Mbit 75ms d
elay) *** 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')
Host h1 has IP address 10.0.0.1 and MAC address 02:93:c0:88:a9:89
Host h2 has IP address 10.0.0.2 and MAC address ba:be:12:9c:39:20
*** 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. 3.14: Запуск скрипта lab\_iperf3.py на отработку

Построим графики из получившегося JSON-файла и создадим Makefile для проведения всего эксперимента (рис. 3.15):

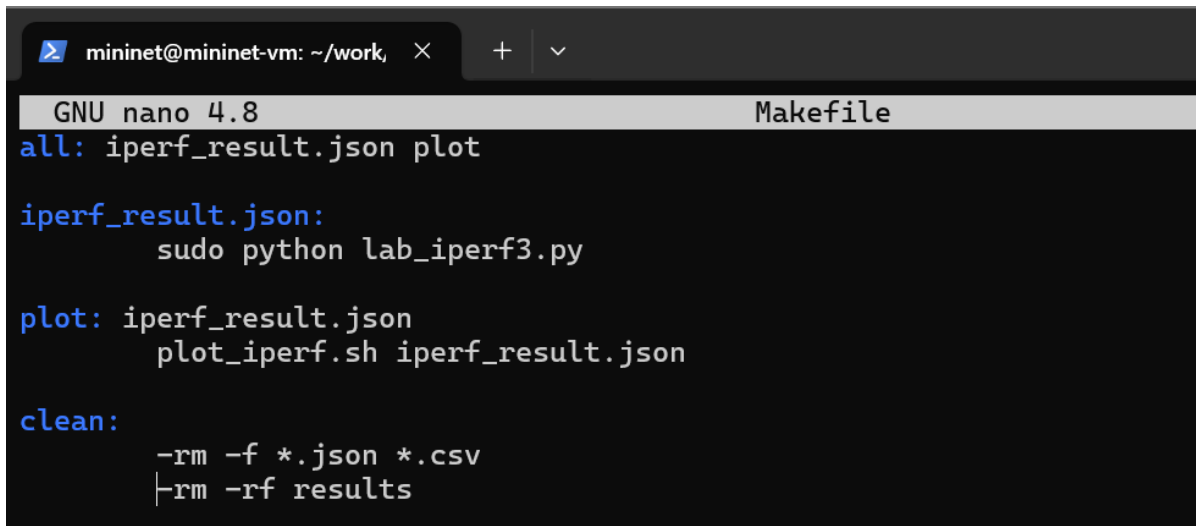
```

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. 3.15: Построение графиков и создание Makefile для проведения всего эксперимента

В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 3.16):

A screenshot of a terminal window with a dark background. The top bar shows the user 'mininet@mininet-vm' in the directory '~/work'. The terminal is running GNU nano 4.8, editing a file named 'Makefile'. The content of the Makefile is as follows:

```
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: Добавление скрипта в Makefile

Проверим корректность отработки Makefile (рис. 3.17):

```

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) (100.00Mbit 75ms d
elay) *** 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')
Host h1 has IP address 10.0.0.1 and MAC address e2:35:bf:6c:b6:5b
Host h2 has IP address 10.0.0.2 and MAC address a6:16:70:81:1b:d2
*** 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$ |

```

Свернуть все окна

Рис. 3.17: Проверка корректности отработки Makefile

Проверим содержимое папки results. (рис. 3.18):

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -s
total 24
4 iperf.csv 8 iperf_result.json 4 lab_iperf3.py 4 Makefile 4 results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ cd results
mininet@mininet-vm:~/work/lab_iperf3/iperf3/results$ ls -l
total 88
-rw-rw-r-- 1 mininet mininet 524 Oct 9 08:52 1.dat
-rw-rw-r-- 1 mininet mininet 9748 Oct 9 08:52 bytes.pdf
-rw-rw-r-- 1 mininet mininet 9609 Oct 9 08:52 cwnd.pdf
-rw-rw-r-- 1 mininet mininet 9036 Oct 9 08:52 MTU.pdf
-rw-rw-r-- 1 mininet mininet 8978 Oct 9 08:52 retransmits.pdf
-rw-rw-r-- 1 mininet mininet 8987 Oct 9 08:52 RTT.pdf
-rw-rw-r-- 1 mininet mininet 9183 Oct 9 08:52 RTT_Var.pdf
-rw-rw-r-- 1 mininet mininet 9621 Oct 9 08:52 throughput.pdf
mininet@mininet-vm:~/work/lab_iperf3/iperf3/results$ |

```

Рис. 3.18: Содержимое папки results

Сохраним содержимое папки results на операционную систему (рис. 3.19):

```
PS C:\Users\mrbor> scp -r mininet@192.168.56.105:/home/mininet/work/lab_iperf3/iperf3/resu
lts/ D:\Mininet\
mininet@192.168.56.105's password:
MTU.pdf 100% 9036 735.4KB/s 00:00
cwnd.pdf 100% 9609 3.1MB/s 00:00
1.dat 100% 524 170.6KB/s 00:00
RTT.pdf 100% 8987 4.3MB/s 00:00
retransmits.pdf 100% 8978 4.3MB/s 00:00
RTT_Var.pdf 100% 9183 4.4MB/s 00:00
bytes.pdf 100% 9748 2.3MB/s 00:00
throughput.pdf 100% 9621 3.1MB/s 00:00
PS C:\Users\mrbor> |
```

Рис. 3.19: Сохранение папки results

## 4 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. iPerf3 [Электронный ресурс]. URL: <https://d2cpnw0u24fjm4.cloudfront.net/wp-content/uploads/iPerf3-User-Documentation.pdf>.