

Отчёт по лабораторной работе №3

Моделирование сетей передачи данных

Измерение и тестирование пропускной способности сети. Воспроизводимый эксперимент

Выполнил: Исаев Булат Абубакарович,
НПИбд-01-22, 1132227131

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
3	Вывод.....	15
4	Список литературы. Библиография	15

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. 1):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
* Management:      https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Oct 31 11:08:14 2025
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 iperf.csv iperf_results.json 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
cp: missing destination file operand after '/home/mininet/mininet/examples/emptynet.py'
Try 'cp --help' for more information.
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
mv: command not found
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_topo

Command 'mv' not found, did you mean:

  command 'imv' from deb renameutils (0.12.0-7)
  command 'qmv' from deb renameutils (0.12.0-7)
  command 'mmv' from deb mmv (1.01b-19build1)
  command 'mv' from deb coreutils (8.30-3ubuntu2)

Try: sudo apt install <deb name>
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emptynet.py lab_iperf3_topo.py
mv: cannot stat 'emptynet.py': No such file or directory
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv lab_iperf3_topo lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ touch lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
```

Рис. 1: Создание подкаталога, копирование файла с примером скрипта (описывающего стандартную простую топологию сети mininet)

Изучим содержание скрипта lab_iperf3_topo.py (рис. 2):

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

"""
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. 2: Открытие файла lab_iperf3_topo.py

Запустим скрипт создания топологии lab_iperf3_topo.py. После отработки скрипта просмотрим элементы топологии и завершим работу mininet (рис. 3):

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

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=1308>
<Host h2: h2-eth0:10.0.0.2 pid=1310>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=1315>
<Controller c0: 127.0.0.1:6653 pid=1301>
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: 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. 4):

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

GNU nano 4.8 lab_iperf3_topo.py Modified

```
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' )

^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 Line
```

Рис. 4: Внесение изменения в скрипт, позволяющего вывести на экран информацию о хосте h1 (имя, IP-адрес, MAC-адрес)

Проверим корректность отработки изменённого скрипта (рис. 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 fe:1f:f1:a4:f7:1e
*** Running CLI
*** Starting CLI:
mininet>
mininet>
Interrupt
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
***
```

Рис. 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. 6 - рис. 7):

```

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

```

^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 ^A Replace ^U Paste Text ^T To Spell ^I Go To Line M-E Redo

Рис. 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$ nano lab_iperf3_topo.py
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
Host h1 has IP address 10.0.0.1 and MAC address 96:37:e3:09:f6:ee
Host h2 has IP address 10.0.0.2 and MAC address 12:ce:b1:51:e2:cb
*** Running CLI
*** Starting CLI:
mininet>

```

Рис. 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. 8):

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

Рис. 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. 9):

```
mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo          lab_iperf3_topo2.py      Modified
GNU nano 4.8
*** Adding controller\n' )

"""

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()

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

Рис. 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. 10):

```

^[[Amininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo2.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ 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 10.0000% loss) (10.00Mbit 5ms delay 10.0000% loss) *** Starting network
*** Configuring hosts
h1 (cfs 5000000/100000us) h2 (cfs 4500000/100000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (10.00Mbit 5ms delay 10.0000% loss) ... (10.00Mbit 5ms delay 10.0000% loss)
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address 5a:21:3d:31:68:19
Host h2 has IP address 10.0.0.2 and MAC address ca:e1:40:91:6f:bd
*** Running CLI
*** Starting CLI:
mininet> -

```

Рис. 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. 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 24
-rw-rw-r-- 1 mininet mininet 976 Oct  9 08:52 iperf.csv
-rw-r--r-- 1 mininet mininet 7779 Oct  9 08:52 iperf_result.json
-rwxrwxr-x 1 mininet mininet 1346 Oct 31 09:14 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$ -

```

Рис. 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. 12): - на хостах не было ограничения по использованию ресурсов процессора; - каналы между хостами и коммутатором были по 100 Мбит/с задержкой 75 мс, без потерь, без использования ограничителей пропускной способности и максимального размера очереди

```

mininet@mininet-vm: ~/work/lab_iperf3/iperf3
GNU nano 4.8                               lab_iperf3.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.
"""

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()

    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' )

^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. 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. 13):

```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
GNU nano 4.8                               lab_iperf3.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.
"""

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() )

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

^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. 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. 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 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')
Host h1 has IP address 10.0.0.1 and MAC address ce:17:60:7d:c2:5e
Host h2 has IP address 10.0.0.2 and MAC address 06:5b:9d:9b:1c:37
*** 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. 14: Запуск скрипта lab_iperf3.py на отработку

Построим графики из получившегося JSON-файла и создадим Makefile для проведения всего эксперимента (рис. 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. 15: Построение графиков и создание Makefile для проведения всего эксперимента

В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 16):

```

GNU nano 4.8
Makefile

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. 16: Добавление скрипта в Makefile

Проверим корректность отработки Makefile (рис. 17):

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nano Makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make clean
rm -f *.json *.csv
rm -rf results
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
:0
*** Starting 1 switches
:3 (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) ... (100.00Mbit 75ms delay) (100.00Mbit 75ms delay)
)
*** Waiting for switches to connect
:3
*** Starting network
*** 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 5a:56:d8:93:ec:01
Host h2 has IP address 10.0.0.2 and MAC address fa:cb:92:66:1a:f0
*** Stopping network*** Stopping 1 controllers
:0
*** Stopping 2 links
..
*** Stopping 1 switches
:3
*** Stopping 2 hosts
h1 h2
*** Done
plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ -

```

Рис. 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/>