

Отчёт по лабораторной работе №3

Моделирование сетей передачи данных

Ищенко Ирина Олеговна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	16

Список иллюстраций

3.1	Содержание файла lab_iperf3_topo.py	7
3.2	Создание топологии и ее основные параметры	8
3.3	Изменение скрипта lab_iperf3_topo.py	9
3.4	Проверка работы внесенных изменений	10
3.5	Настройка параметров производительности	11
3.6	Запуск скрипта с настройкой параметров производительности и без нее	12
3.7	Создание копии скрипта lab_iperf3_topo2.py	12
3.8	Изменения кода в скрипте lab_iperf3.py	13
3.9	Запуск скрипта lab_iperf3.py	14
3.10	Создание Makefile	14
3.11	Создание Makefile	15
3.12	Проверка работы Makefile	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 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.

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.

Изучим содержание скрипта `lab_iperf3_topo.py` (Рисунок 3.1).

В нем написан скрипт по созданию простейшей топологии из двух хостов `h1` и `h2`, а также коммутатора `s3` и контроллера `c0`. В начале файла видим импорт необходимых библиотек.

```

mininet@mininet-vm: ~/work/lab_iperf3/lab_iperf3_topo
* Support: https://ubuntu.com/advantage
New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Oct  7 06:12:26 2025
mininet@mininet-vm:~$ ls
abpf_demo  ebpf_final  lab  mininet  mininet.orig  oflops  oftest  openflow  pox  work
mininet@mininet-vm:~$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ mkdir lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3$ cp ~/mininet/examples/emphynet.py ~/work/lab_iperf3/lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3$ ls
iperf.csv  iperf_results.json  lab_iperf3_topo  results
mininet@mininet-vm:~/work/lab_iperf3$ cd lab_iperf3_topo
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ mv emphynet.py lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cat 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()
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$

```

Рисунок 3.1: Содержание файла lab_iperf3_topo.py

Основные элементы:

- `addSwitch()`: добавляет коммутатор в топологию и возвращает имя коммутатора;
- `addHost()`: добавляет хост в топологию и возвращает имя хоста;
- `addLink()`: добавляет двунаправленную ссылку в топологию (и возвращает ключ ссылки; ссылки в Mininet являются двунаправленными, если не указано иное);

- Mininet: основной класс для создания и управления сетью;
- start(): запускает сеть;
- pingAll(): проверяет подключение, пытаясь заставить все узлы пинговать друг друга;
- stop(): останавливает сеть;
- net.hosts: все хосты в сети;
- dumpNodeConnections(): сбрасывает подключения к/от набора узлов;
- setLogLevel(„info“ | „debug“ | „output“): устанавливает уровень вывода Mininet по умолчанию; рекомендуется info.

Запустим скрипт создания топологии lab_iperf3_topo.py и посмотрим ее основные параметры (Рисунок 3.2).

```
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=866>
<Host h2: h2-eth0:10.0.0.2 pid=870>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=875>
<Controller c0: 127.0.0.1:6653 pid=859>
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$
```

Рисунок 3.2: Создание топологии и ее основные параметры

Внесем в скрипт lab_iperf3_topo.py изменение, позволяющее вывести на экран информацию обоих хостов сети, а именно имя хоста, его IP-адрес, MAC-адрес (Рисунок 3.3).


```

lab_iperf3_topo.py > emptyNet

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()

```

Рисунок 3.3: Изменение скрипта lab_iperf3_topo.py

Здесь:

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

Проверим корректность отработки изменённого скрипта (Рисунок 3.4).

```

*** 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 be:02:5e:b2:3b:0e
Host h2 has IP address 10.0.0.2 and MAC address 12:f4:b2:92:e7:25
*** Running CLI
*** Starting CLI:
mininet>
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done

```

Рисунок 3.4: Проверка работы внесенных изменений

Действительно, нам вывелась информация об IP и mac адресах хостов.

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов `CPULimitedHost` и `TCLink`. Добавим в скрипт настройки параметров производительности (Рисунок 3.5).

В скрипте `lab_iperf3_topo2.py` изменим строку описания сети, указав на использование ограничения производительности и изоляции. Также 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 скрипте изменим функцию параметров соединения между хостом `h1` и коммутатором `s3`. А именно добавим двунаправленный канал с характеристиками пропускной способности, задержки и потерь:

- параметр пропускной способности (`bw`) выражается числом в Мбит;
- задержка (`delay`) выражается в виде строки с заданными единицами измерения (например, `5ms`, `100us`, `1s`);
- потери (`loss`) выражаются в процентах (от 0 до 100);
- параметр максимального значения очереди (`max_queue_size`) выражается в пакетах;

- параметр `use_htb` указывает на использование ограничителя интенсивности входящего потока Hierarchical Token Bucket (HTB)

```
lab_iperf3_topo2.py > emptyNet
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' )
    emptyNet()
```

Рисунок 3.5: Настройка параметров производительности

Запустим на отработку сначала скрипт `lab_iperf3_topo2.py`, затем `lab_iperf3_topo.py` и сравним результат (Рисунок 3.6). Увидим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.

```

*** Done
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo2.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(10.00Mbit 5ms delay 10.00000% loss) (10.00Mbit 5ms delay 10.00000% loss) *** Starting
network
*** Configuring hosts
h1 (cfs 5000000/100000us) h2 (cfs 4500000/100000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (10.00Mbit 5ms delay 10.00000% loss) ... (10.00Mbit 5ms delay 10.00000% loss)
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address 86:ab:ca:07:f2:01
Host h2 has IP address 10.0.0.2 and MAC address 7a:a6:35:a1:29:b6
*** 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> link
invalid number of args: link end1 end2 [up down]
mininet> links
h1-eth0<->s3-eth1 (OK OK)
h2-eth0<->s3-eth2 (OK OK)
mininet> dump
<CPULimitedHost h1: h1-eth0:10.0.0.1 pid=1631>
<CPULimitedHost h2: h2-eth0:10.0.0.2 pid=1633>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=1638>
<Controller c0: 127.0.0.1:6653 pid=1624>
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
(cfs -1/100000us) (cfs -1/100000us) *** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$

```

Рисунок 3.6: Запуск скрипта с настройкой параметров производительности и без нее

Построим графики по проводимому эксперименту.

Сделаем копию скрипта lab_iperf3_topo2.py и поместим его в подкаталог iperf (Рисунок 3.7).

```

*** 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_t
opo/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 1296 Oct  7 06:31 lab_iperf3.py

```

Рисунок 3.7: Создание копии скрипта lab_iperf3_topo2.py

Изменим код в скрипте lab_iperf3.py так, чтобы (Рисунок 3.8):

- 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.

- 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:



```
mininet@mininet-vm: ~/work/lab_iperf3/iperf3
#!/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
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 = 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( '*** Traffic generation\n' )
    h2.cmdPrint( 'iperf3 -s -D -1' )
    time.sleep(10)
    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()

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

Рисунок 3.8: Изменен ия кода в скрипте lab_iperf3.py

Запустим на отработку скрипт lab_iperf3.py (Рисунок 3.9).

```

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 75m
s 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) (100.00Mb
it 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 7e:f8:f4:e9:40:98
Host h2 has IP address 10.0.0.2 and MAC address 22:50:87:c5:44:62
*** 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$

```

Рисунок 3.9: Запуск скрипта lab_iperf3.py

Построим графики из получившегося JSON-файла. Создадим Makefile для проведения всего эксперимента (Рисунок 3.10).

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls
iperf_result.json lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls
iperf.csv iperf_result.json lab_iperf3.py results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ cd results
mininet@mininet-vm:~/work/lab_iperf3/iperf3/results$ ls
l.dat cwnd.pdf retransmits.pdf RTT Var.pdf
bytes.pdf MTU.pdf RTT.pdf throughput.pdf
mininet@mininet-vm:~/work/lab_iperf3/iperf3/results$ cd ..
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch Makefile

```

Рисунок 3.10: Создание Makefile

В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (Рисунок 3.11).

```

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

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

```

Рисунок 3.11: Создание Makefile

Проверьте корректность отработки Makefile (Рисунок 3.12).

```

mininet@mininet-vm: ~/work/lab_iperf3/iperf3$ make clean
rm -f *.json *.csv
rm -rf results
mininet@mininet-vm: ~/work/lab_iperf3/iperf3$ ls
lab_iperf3.py Makefile
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 -1/1000000us) h2 (cfs -1/1000000us)
*** 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 -l',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address d6:91:23:04:d3:53
Host h2 has IP address 10.0.0.2 and MAC address e6:0a:c8:3c:32:af
*** 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$ ls
iperf.csv iperf_result.json lab_iperf3.py Makefile results
mininet@mininet-vm: ~/work/lab_iperf3/iperf3$

```

Рисунок 3.12: Проверка работы Makefile

4 Выводы

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