

Лабораторная работа №3

Измерение и тестирование пропускной способности сети.
Воспроизводимый эксперимент

Кузнецова София Вадимовна

Содержание

Цель работы	4
Теоретическое введение	5
Выполнение лабораторной работы	6
Выводы	15
Список литературы	16

Список иллюстраций

0.1	Копирование файла empytnet.py	6
0.2	Содержание файла lab_iperf3_topo.py	7
0.3	Создание топологии и ее основные параметры	8
0.4	Изменение скрипта lab_iperf3_topo.py	8
0.5	Проверка работы внесенных изменений	9
0.6	Проверка работы внесенных изменений	9
0.7	Настройка параметров производительности	10
0.8	Запуск скрипта с настройкой параметров производительности и без нее	11
0.9	Запуск скрипта с настройкой параметров производительности и без нее	11
0.10	Создание копии скрипта lab_iperf3_topo2.py	11
0.11	Изменения кода в скрипте lab_iperf3.py	12
0.12	Запуск скрипта lab_iperf3.py	12
0.13	Запуск скрипта lab_iperf3.py	13
0.14	Запуск скрипта lab_iperf3.py	13
0.15	Создание Makefile	13
0.16	Проверка работы Makefile	14
0.17	Проверка работы Makefile	14

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.

Теоретическое введение

Mininet[@mininet] – это эмулятор компьютерной сети. Под компьютерной сетью подразумеваются простые компьютеры — хосты, коммутаторы, а так же OpenFlow-контроллеры. С помощью простейшего синтаксиса в примитивном интерпретаторе команд можно разворачивать сети из произвольного количества хостов, коммутаторов в различных топологиях и все это в рамках одной виртуальной машины(ВМ). На всех хостах можно изменять сетевую конфигурацию, пользоваться стандартными утилитами(ifconfig, ping) и даже получать доступ к терминалу. На коммутаторы можно добавлять различные правила и маршрутизировать трафик.

iPerf3[@iperf] представляет собой кроссплатформенное клиент-серверное приложение с открытым исходным кодом, которое можно использовать для измерения пропускной способности между двумя конечными устройствами. iPerf3 может работать с транспортными протоколами TCP, UDP и SCTP:

- TCP и SCTP:
 - измеряет пропускную способность;
 - позволяет задать размер MSS/MTU;
 - отслеживает размер окна перегрузки TCP (CWnd).
- UDP:
 - измеряет пропускную способность;
 - измеряет потери пакетов;
 - измеряет колебания задержки (jitter);
 - поддерживает групповую рассылку пакетов (multicast).

Выполнение лабораторной работы

С помощью API Mininet создадим простейшую топологию сети, состоящую из двух хостов и коммутатора с назначеннной 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.

```
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/emptynet
.py ~/work/lab_iperf3/lab_iperf3_topo
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. 0.1: Копирование файла emptynet.py

Изучим содержание скрипта lab_iperf3_topo.py. 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.

```

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()

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

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

```

Рис. 0.2: Содержание файла lab_iperf3_topo.py

Основные элементы:

- addSwitch(): добавляет коммутатор в топологию и возвращает имя коммутатора;
- ddHost(): 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.

```

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=905>
<Host h2: h2-eth0:10.0.0.2 pid=909>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=914>
<Controller c0: 127.0.0.1:6653 pid=898>
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2

```

Рис. 0.3: 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.

```

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 )

```

Рис. 0.4: 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.

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

Рис. 0.5: Проверка работы внесенных изменений

```
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 36:d6:3d:4a:54:b0
Host h2 has IP address 10.0.0.2 and MAC address d2:6e:8a:43:3c:be
*** Running CLI
*** Starting CLI:
mininet>
```

Рис. 0.6: Проверка работы внесенных изменений

Действительно, нам вывелаась 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. 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) выра-

жается в пакетах; - параметр use_htb указывает на использование ограничителя интенсивности входящего потока Hierarchical Token Bucket (HTB)

```
GNU nano 4.8          lab_iperf3_topo2.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.node import CPULimitedHost
from mininet.link import TCLink
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, host = CPULimitedHost>
    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=>
    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() )
```

Рис. 0.7: Настройка параметров производительности

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

```

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 86:2a:c3:5b:71:7f
Host h2 has IP address 10.0.0.2 and MAC address 92:17:91:8b:a9:30
*** Running CLI
*** Starting CLI:
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

```

Рис. 0.8: Запуск скрипта с настройкой параметров производительности и без нее

```

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 66:da:f0:37:9e:e1
Host h2 has IP address 10.0.0.2 and MAC address be:92:9d:db:4c:90

```

Рис. 0.9: 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.

```

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 1346 Oct  6 03:32 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ 

```

Рис. 0.10: Создание копии скрипта lab_iperf3_topo2.py

Изменим код в скрипте lab_iperf3.py так, чтобы:

- 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, закомментируем строки, отвечающие за запуск CLI-интерфейса:

```
GNU nano 4.8          lab_iperf3.py          Modified
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel, info
import time

def emptyNet():
    "Create an empty network and add nodes to it."
    net = Mininet( controller=Controller, waitConnected=True, host = CPULimitedHost>
    info( '*** Adding controller\n' )
    net.addController( 'c0' )

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

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

    info( '*** Creating links\n' )
    net.addLink( h1, s3, 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)
    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. 0.11: Изменения кода в скрипте lab_iperf3.py

Запустим на отработку скрипт lab_iperf3.py.

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

Рис. 0.12: Запуск скрипта lab_iperf3.py

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -l
total 4
-rwxrwxr-x 1 mininet mininet 1346 Oct  6 03:32 lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nano lab_iperf3.py
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nano lab_iperf3.py
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) (100.0
0Mbit 75ms delay)
*** Waiting for switches to connect
s3
*** Starting network
*** 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 e6:05:71:26:6e:0d
Host h2 has IP address 10.0.0.2 and MAC address fe:64:40:b8:4e:e9
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done

```

Рис. 0.13: Запуск скрипта lab_iperf3.py

```

mininet@mininet-vm:~/work/lab_iperf3/iperf3$ ls -l
total 12
-rw-r--r-- 1 root      root      7769 Oct  6 03:44 iperf_result.json
-rw-rw-r-x 1 mininet   mininet  1519 Oct  6 03:43 lab_iperf3.py

```

Рис. 0.14: Запуск скрипта lab_iperf3.py

Построим графики из получившегося JSON-файла. Создадим Makefile для проведения всего эксперимента. В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов.

	GNU nano 4.8	Makefile	Modified
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. 0.15: Создание Makefile

Проверьте корректность отработки Makefile.

```
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$ nano Makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make clean
rm -f *.json *.csv
rm -rf results
```

Рис. 0.16: Проверка работы 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 -1/100000us) h2 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay) (100.00Mbit 75ms delay) ... (100.00Mbit 75ms delay) (100.0
0Mbit 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 ba:72:b8:75:5d:9d
Host h2 has IP address 10.0.0.2 and MAC address 4a:42:db:7f:9a:72
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
...
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ █
```

Рис. 0.17: Проверка работы Makefile

Выводы

В результате выполнения данной 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.

Список литературы