

Отчет по лабораторной работе №3 Дисциплина: Моделирование сетей передачи данных

Леснухин Даниил Дмитриевич
Российский университет дружбы народов
Москва

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.

Задание

1. Воспроизвести посредством API Mininet эксперименты по измерению пропускной способности с помощью iPerf3.
2. Построить графики по проведённому эксперименту.

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

Установка необходимого программного обеспечения

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/emphynet.py, описывающего стандартную простую топологию сети mininet. 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.

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ echo emphynet.py
emphynet.py
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$ ls
lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ |
```

Рис. 1: Копирование файла emphynet.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

Внесем в скрипт lab_iperf3_topo.py изменение, позволяющее вывести на экран информацию обоих

```

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> links
h1-eth0<->s3-eth1 (OK OK)
h2-eth0<->s3-eth2 (OK OK)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=7221>
<Host h2: h2-eth0:10.0.0.2 pid=7223>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=7228>
<Controller c0: 127.0.0.1:6653 pid=7214>
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
s3 lo: s3-eth1:h1-eth0 s3-eth2:h2-eth0
c0
mininet> |

```

Рис. 2: Создание топологии и ее основные параметры

```

mininet@mininet-vm: ~/work, X + v
/home/mininet/work/lab_iperf3/lab_iperf3_topo/lab_iperf3
#!/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=

    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(), "
    print( "Host", h2.name, "has IP address", h2.IP(), "

```

Здесь:

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

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)

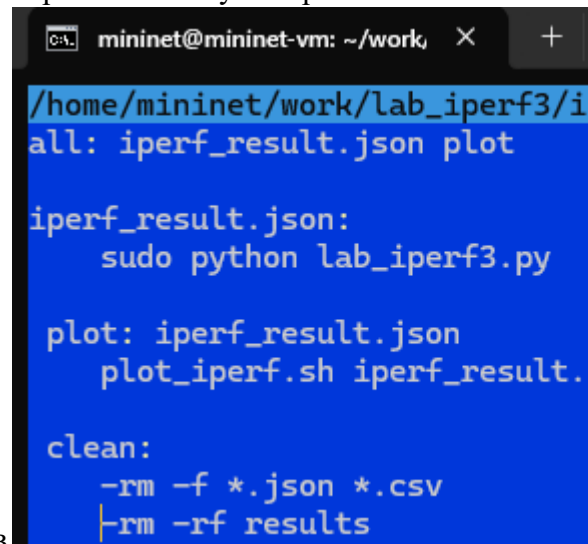
Mininet предоставляет функции ограничения производительности и изоляции с помощью классов `CPULimitedHost` и `TCLink`. Добавим в скрипт настройки параметров производительности:

В начале скрипта `lab_iperf3_topo2.py` добавим записи об импорте классов `CPULimitedHost` и `TCLink`:

Запустим на отработку скрипт `lab_iperf3.py`

Создадим Makefile для проведения всего эксперимента. В Makefile пропишем запуск скрипта экс-

перимента, построение графиков и очистку каталога от результатов



```
mininet@mininet-vm: ~/work
/home/mininet/work/lab_iperf3/i
all: iperf_result.json plot

iperf_result.json:
    sudo python lab_iperf3.py

plot: iperf_result.json
    plot_iperf.sh iperf_result.

clean:
    -rm -f *.json *.csv
    -rm -rf results
```

```

/home/mininet/work/lab_iperf3/lab_iperf3_topo/lab_iperf3_topo2.py  [-M--] 89 L:[ 1+31 32/ 50] *(926
#!/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.node import CPULimitedHost
from mininet.link import TCLink

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, 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() )

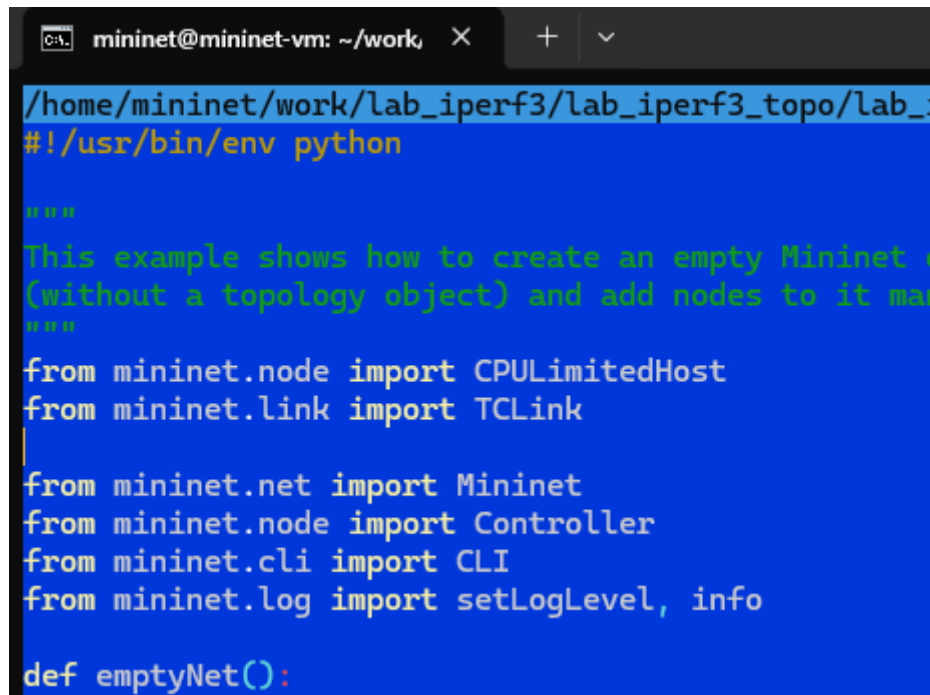
    info( '*** Running CLI\n' )
    CLI( net )

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

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

```

Рис. 3: Изменение скрипта lab_iperf3_topo.py



```
mininet@mininet-vm: ~/work X + v
/home/mininet/work/lab_iperf3/lab_iperf3_topo/lab_
#!/usr/bin/env python

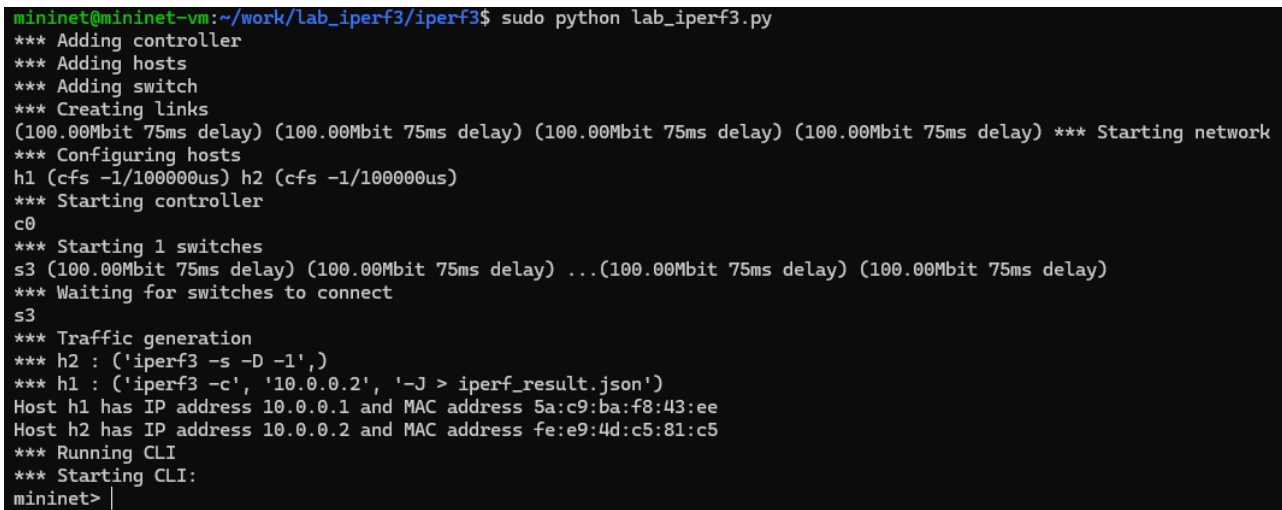
"""
This example shows how to create an empty Mininet
(without a topology object) and add nodes to it manually.
"""

from mininet.node import CPULimitedHost
from mininet.link import TCLink

from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def emptyNet():
```

Рис. 4: Изменение скрипта lab_iperf3_topo2.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) (100.00Mbit 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 5a:c9:ba:f8:43:ee
Host h2 has IP address 10.0.0.2 and MAC address fe:e9:4d:c5:81:c5
*** Running CLI
*** Starting CLI:
mininet> |
```

Рис. 5: Запуск на обработку скрипт lab_iperf3.py

```
mininet@mininet-vm: ~/work, X + v
iperf.csv iperf_result.json lab_iperf3.py results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ vim makefile
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
s 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) ...(100.0
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 22:f8:a3:3b:
Host h2 has IP address 10.0.0.2 and MAC address d2:86:3d:aa:
*** Running CLI
*** Starting CLI:
mininet> |
```

Проверка корректности отработки Makefile

Выводы

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