Лабораторная работа № 3

Измерение и тестирование пропускной способности сети. Воспроизводимый эксперимент

Старовойтов Егор Сергеевич

Содержание

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 про- пускной способности с помощью iPerf3.
2. Построить графики по проведённому эксперименту

Теоретическое введение

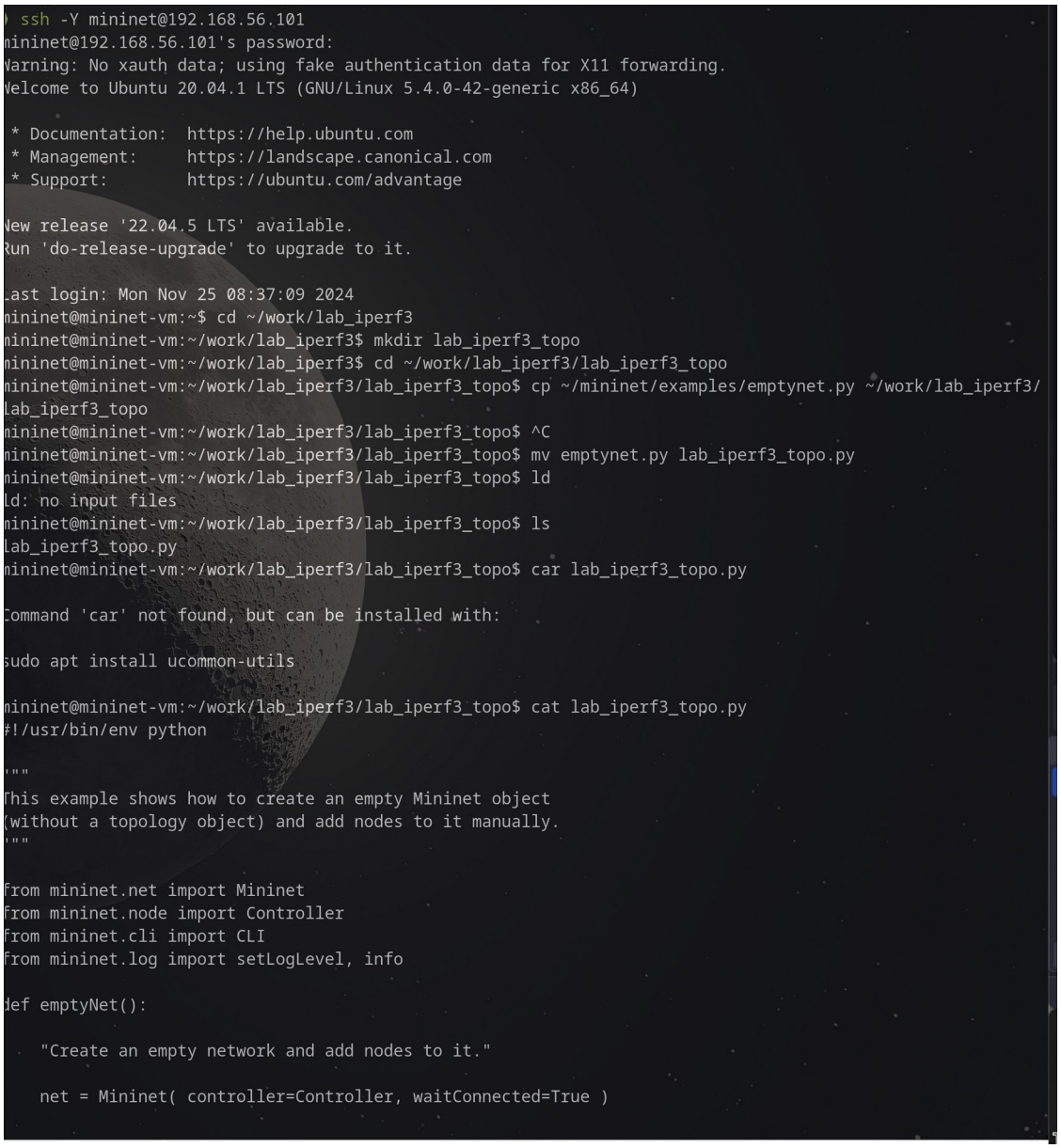
Application Programming Interface (API) — программный интерфейс прило- жения, или интерфейс программирования приложений) представляет собой специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого. API Mininet построен на трех основных уровнях: – Низкоуровневый API состоит из базовых узлов и классов ссылок (таких как Host, Switch, Link и их подклассы), которые на самом деле могут быть созданы по отдельности и использоваться для создания сети, но это немного громоздко. – API среднего уровня добавляет объект Mininet, который служит контейнером для узлов и ссылок. Он предоставляет ряд методов (addHost(), addSwitch(), addLink()) для добавления узлов и ссылок в сеть, а также настройки сети, запуска и завершения работы (start(), stop()). – Высокоуровневый API добавляет абстракцию шаблона топологии (класс Topo), который предоставляет возможность создавать повторно используемые пара- метризованные шаблоны топологии. Эти шаблоны можно передать команде mn (через

параметр –custom) и использовать из командной строки. Низкоуровневый API используется, когда требуется управлять узлами и ком- мутаторами напрямую. API среднего уровня применяют при запуске и остановке сети (в частности используется класс Mininet). Полноценные сети могут быть созданы с использованием любого из уров- ней API, но обычно для создания сетей выбирают либо API среднего уровня (например, Mininet.add*()), либо API высокого уровня (Topo. add*())

Выполнение лабораторной работы

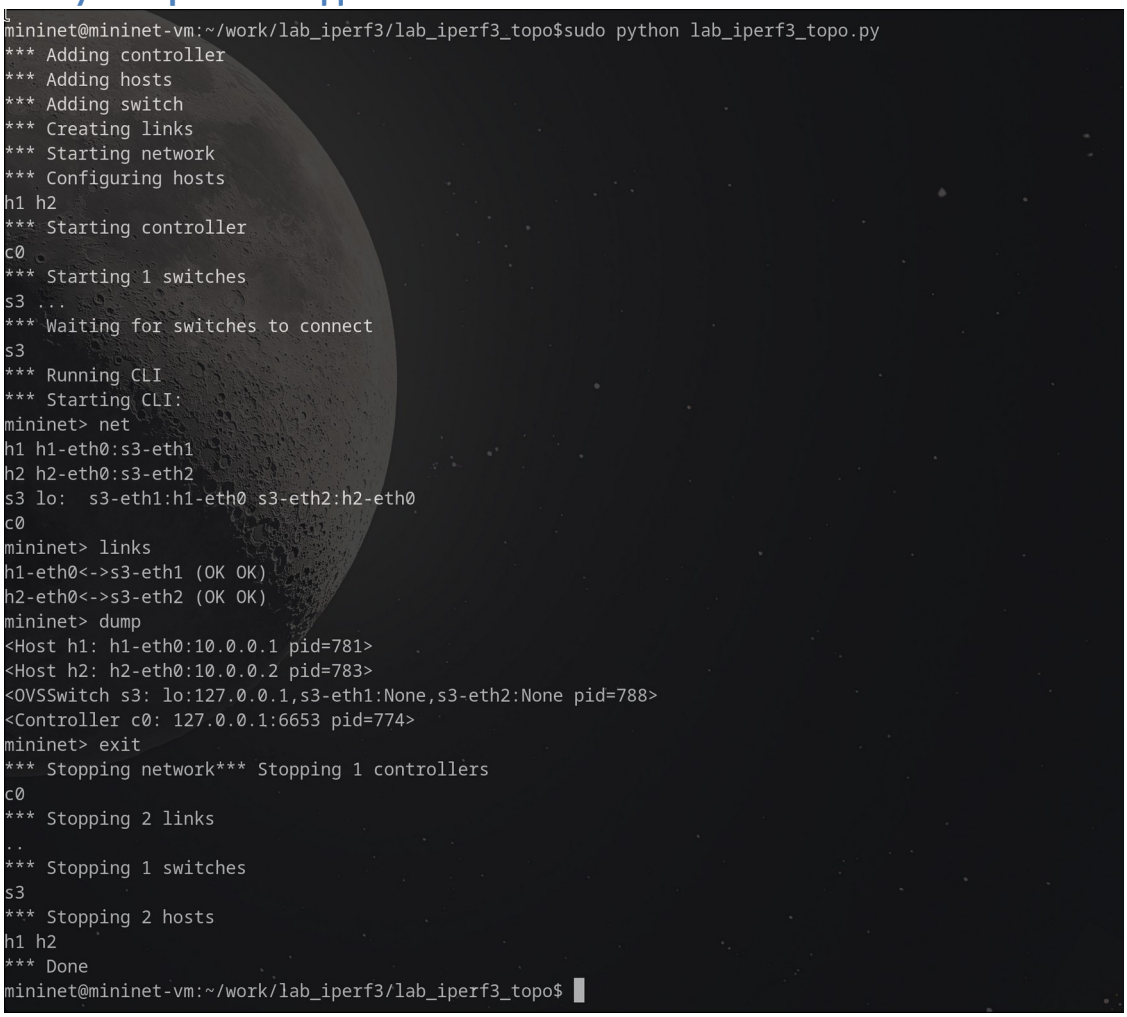
1. Создание файла скрипта

В каталоге /work/lab_iperf3 для работы над проектом я создал под- каталог lab_iperf3_topo и скопировал в него файл с примером скрипта mininet/examples/emptynet.py, описывающий стандартную простую топологию сети mininet.



Создание скрипта

2. Запуск скрипта создания топологии



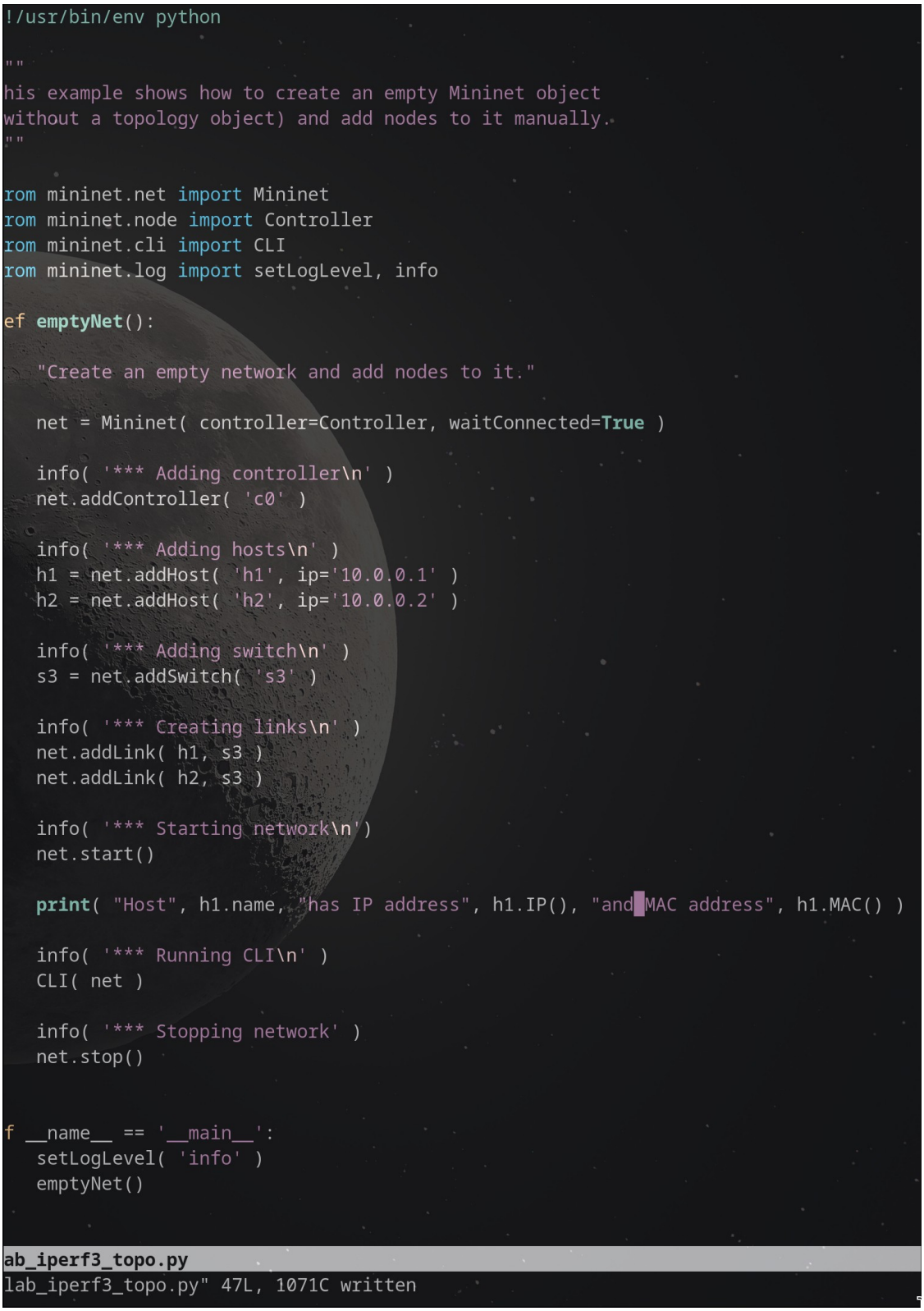
Запуск топологии

3. Вывод информации о хосте h1

Я добавил строку

```
print( "Host", h1.name, "has IP address", h1.IP(), "andMAC address", h1.MAC() )
```

в скрипт.


!/usr/bin/env python
""
his example shows how to create an empty Mininet object
without a topology object) and add nodes to it manually.
""
rom mininet.net import Mininet
rom mininet.node import Controller
rom mininet.cli import CLI
rom mininet.log import setLogLevel, info
ef 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()
f __name__ == '__main__':
setLogLevel('info')
emptyNet()
ab_iperf3_topo.py
lab_iperf3_topo.py" 47L, 1071C written

Скрипт

4. Проверка работы скрипта

```
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 0e:ad:dc:e1:ef:27
*** Running CLI
*** Starting CLI:
mininet>
```

Mininet

5. Вывод информации о втором хосте

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nvim 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
Host h1 has IP address 10.0.0.1 and MAC address 06:5c:78:d9:bd:a8
Host h2 has IP address 10.0.0.2 and MAC address 32:e1:ea:26:48:3d
*** Running CLI
*** Starting CLI:
mininet>
```

Запуск скрипта

6. Настройки параметров производительности

```
"""
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' )
lab_iperf3_topo2.py                                    1,1           To
```

Модифицированный скрипт lab_iperf3_topo2.py

Можно увидеть, что были ограниченыя процессорные ресурсы на хотсах, а также установлены гранциы размера очереди и задержки на канале между первым хостом и коммутатором.

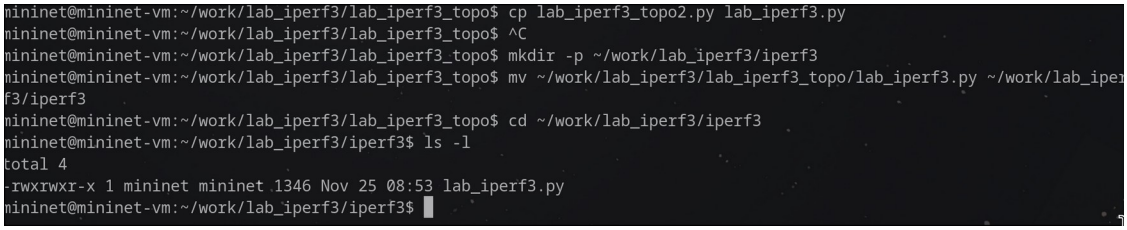
7. Сравнение работы двух скриптов

```
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 d2:0f:a5:af:de:ee
Host h2 has IP address 10.0.0.2 and MAC address 96:ac:de:ae:e8:72
*** 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
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 f2:94:5c:87:da:55
Host h2 has IP address 10.0.0.2 and MAC address 16:7f:c1:ce:8b:1a
*** 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
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$
```

Запуск двух скриптов

8. Создание копии lab_iperf3_topo2.py

Я сделал копию скрипта lab_iperf3_topo2.py и поместил его в подка- талог iperf.



Копирование скрипта

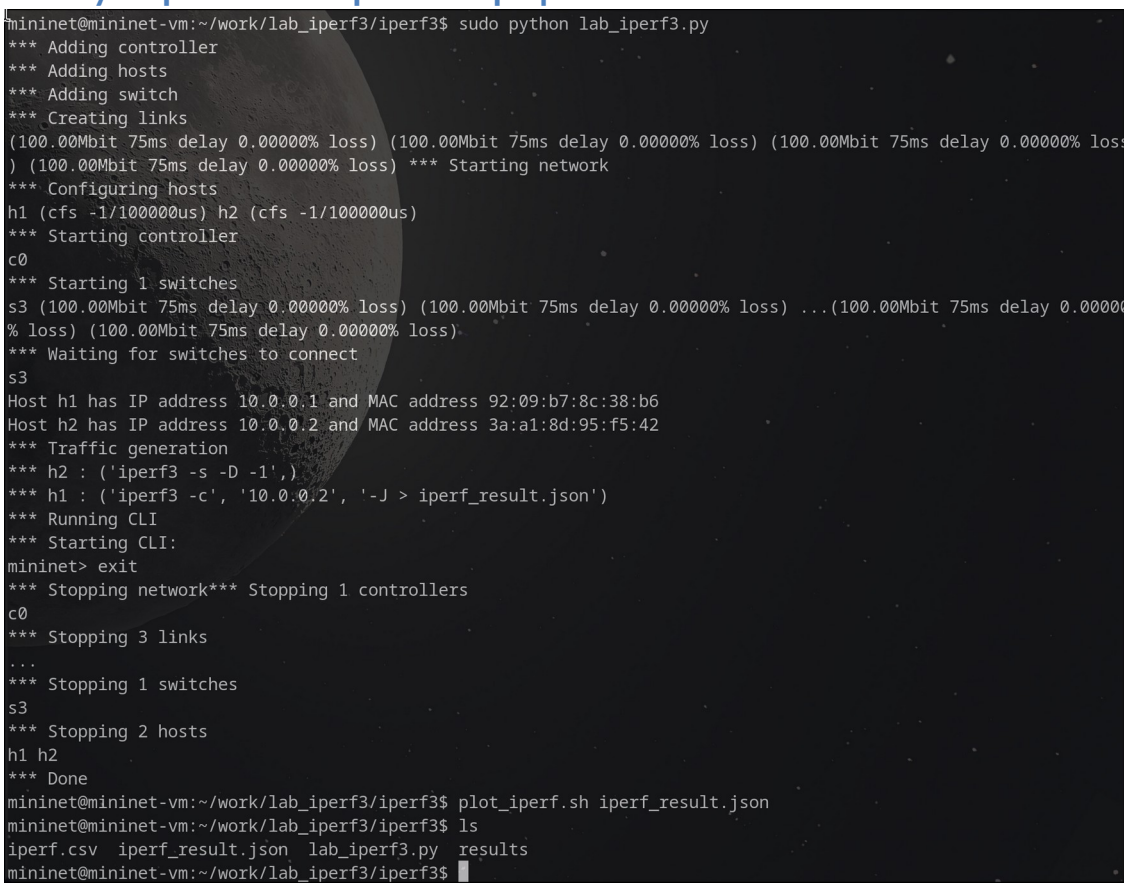
9. Модификация lab_iperf3.py

Я изменил код так, чтобы на хостах не было ограничений по ресурсам процессора, задержка между хостами и коммутаторами была 75мс, а пропускная способность 100Мбит/с без ограничений.

```
"""
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', loss=0)
    net.addLink( h2, s3, bw=100, delay='75ms', loss=0)

    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( '*** 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' )

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

    info( '*** Stopping network' )
    net.stop()
lab_iperf3.py                                                    47,4          16%
"lab_iperf3.py" 58L, 1563C written
```

lab_iperf3.py

10. Запуск iperf3 и построение графиков



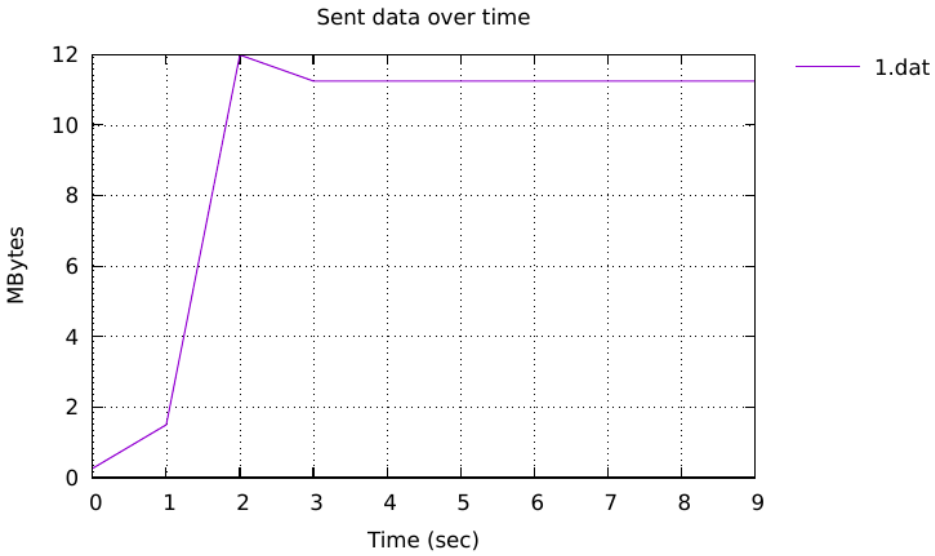
Run

12. Создание и проверка Makefile

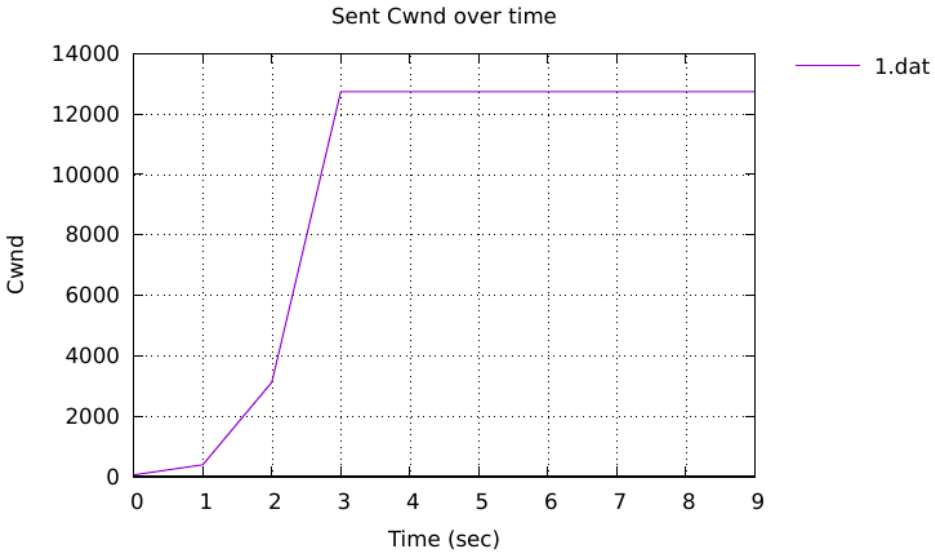
```
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ nvim Makefile
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make clean
rm -f *.json *.csv
rm -rf results
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ make
sudo python lab_iperf3.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
(100.00Mbit 75ms delay 0.00000% loss) (100.00Mbit 75ms delay 0.00000% loss) (100.00Mbit 75ms delay 0.00000% loss
) (100.00Mbit 75ms delay 0.00000% loss) *** Starting network
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay 0.00000% loss) (100.00Mbit 75ms delay 0.00000% loss) ...(100.00Mbit 75ms delay 0.00000
% loss) (100.00Mbit 75ms delay 0.00000% loss)
*** Waiting for switches to connect
s3
Host h1 has IP address 10.0.0.1 and MAC address e2:b8:29:df:a1:74
Host h2 has IP address 10.0.0.2 and MAC address be:0d:7f:4a:9d:b9
*** Traffic generation
*** h2 : ('iperf3 -s -D -1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
*** Running CLI
*** Starting CLI:
mininet> cat Makefile
*** Unknown command: cat Makefile
mininet> exit
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
plot_iperf.sh iperf_result.json
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ cat Makefile
8all: 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
mininet@mininet-vm:~/work/lab_iperf3/iperf3$
```

Создание и проверка Makefile

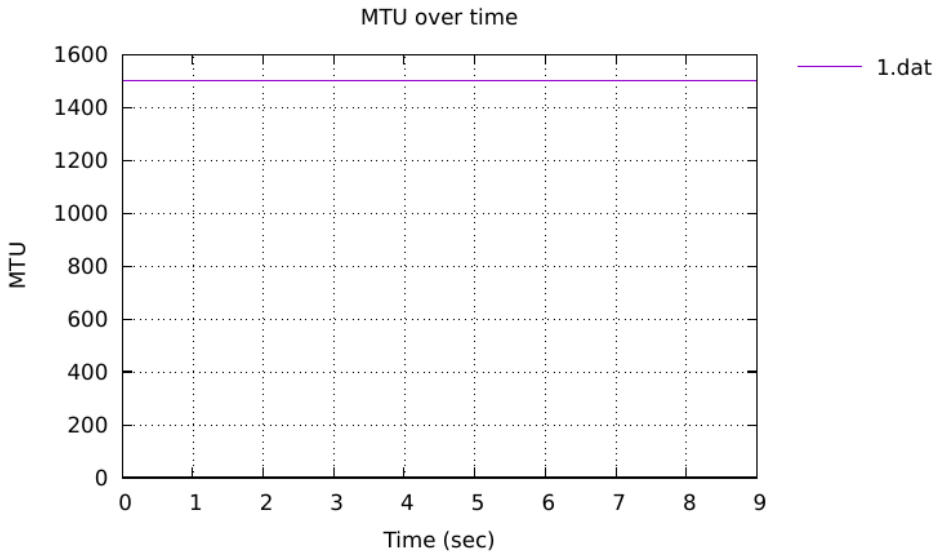
Графики



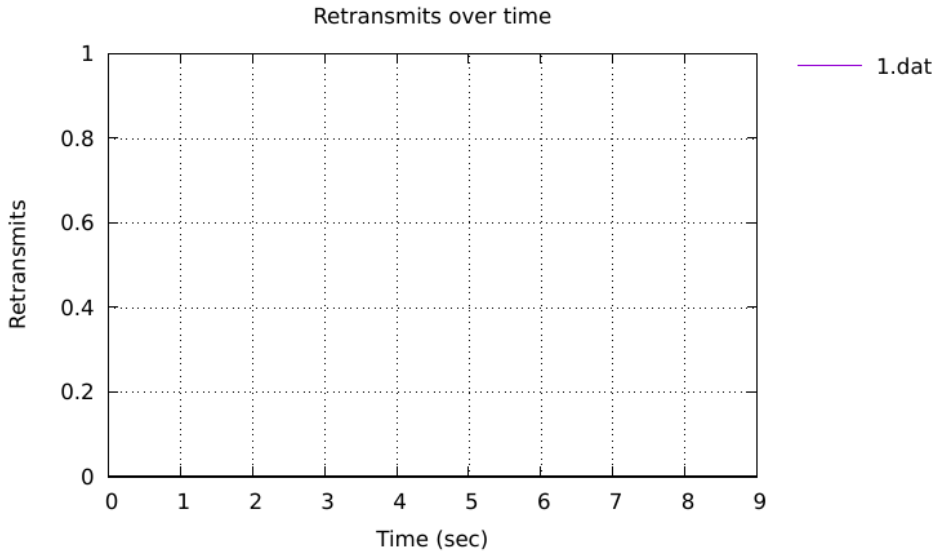
bytes



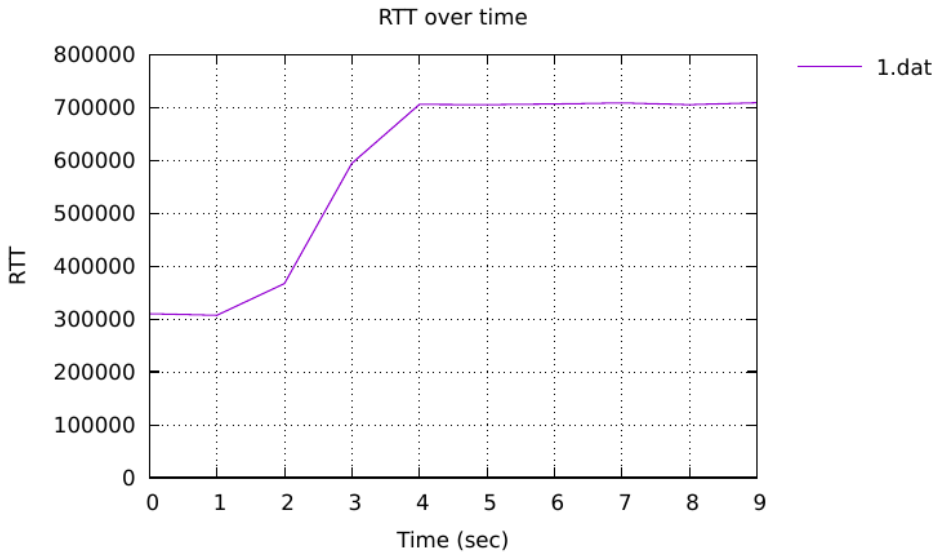
cwnd



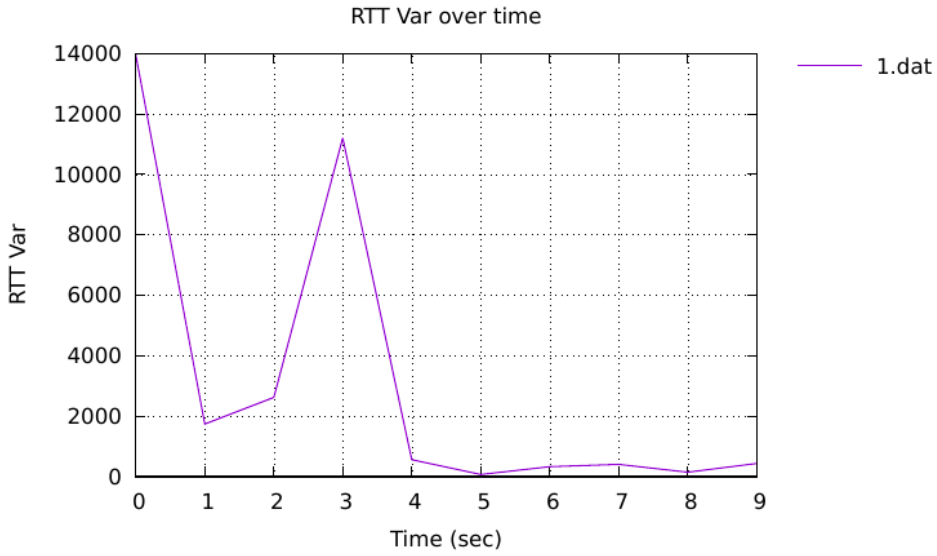
MTU



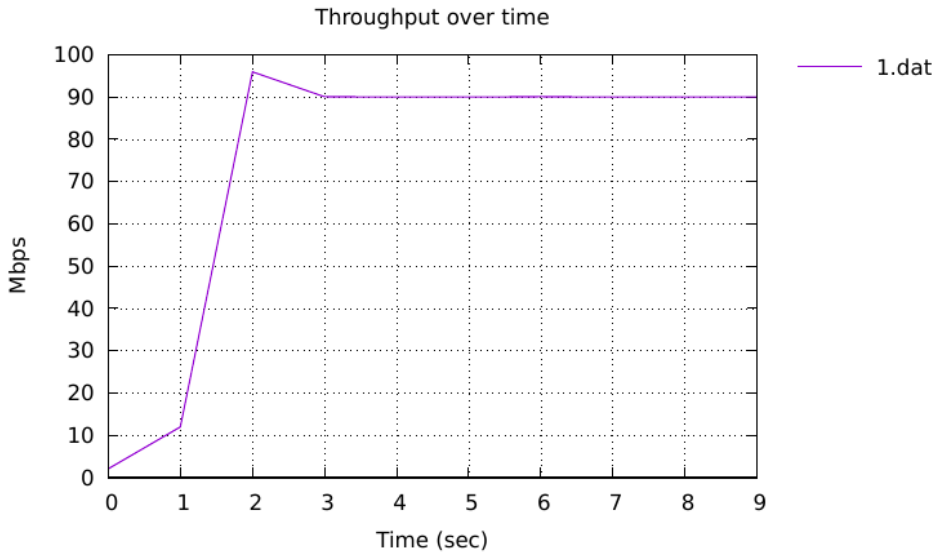
retransmits


RTT over time
800000
700000
600000
500000
400000
300000
200000
100000
0
RTT
0
1
2
3
4
5
6
7
8
9
Time (sec)
1.dat

RTT


RTT Var over time
14000
12000
10000
8000
6000
4000
2000
0
RTT Var
0
1
2
3
4
5
6
7
8
9
Time (sec)
1.dat

RTT_Var



throughput

Выводы

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.

В процессе выполнения работы были решены поставленные задачи, а именно: 1. Воспроизвести посредством API Mininet эксперименты по измерению про- пускной способности с помощью iPerf3. 2. Построить графики по проведённому эксперименту

Список литературы