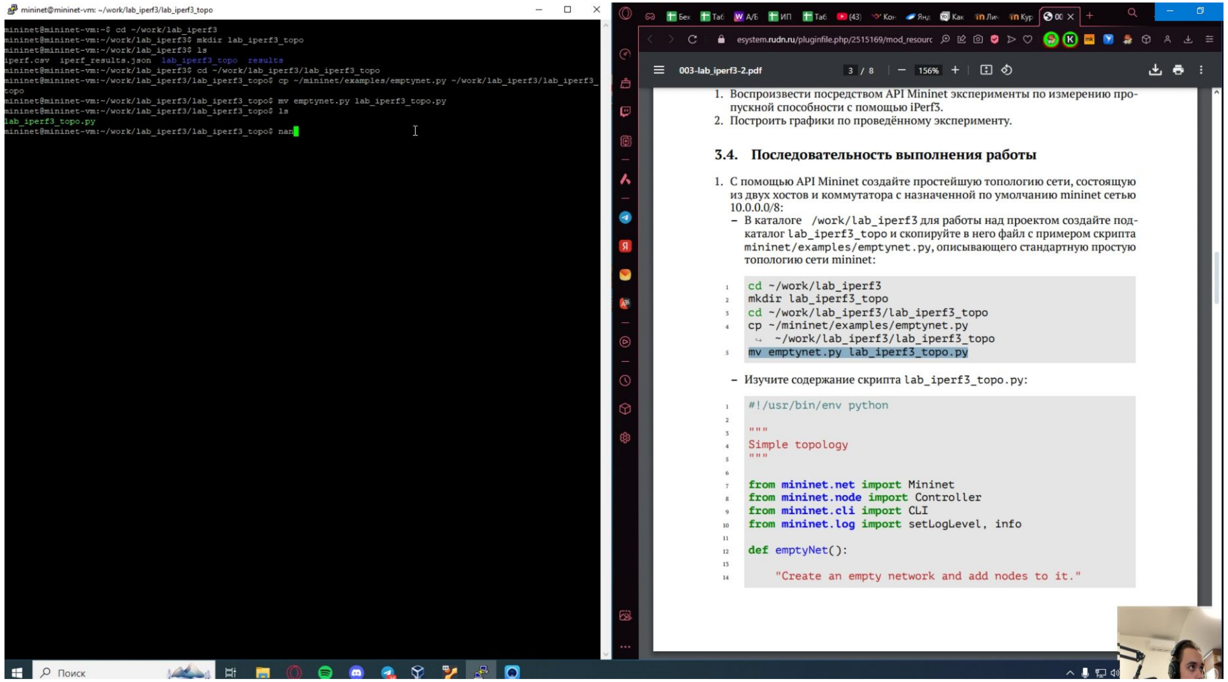
Лабораторная работа №3

по дисциплине Моделирование сетей передачи данных

Ход работы



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

```
mininet@mininet-vm:~$ cd ~/work/lab_iperf3
mininet@mininet-vm:~/work/lab_iperf3$ mkdir 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 ~/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$ ls
lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nan
```

003-lab_iperf3-2.pdf

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

3.4. Последовательность выполнения работы

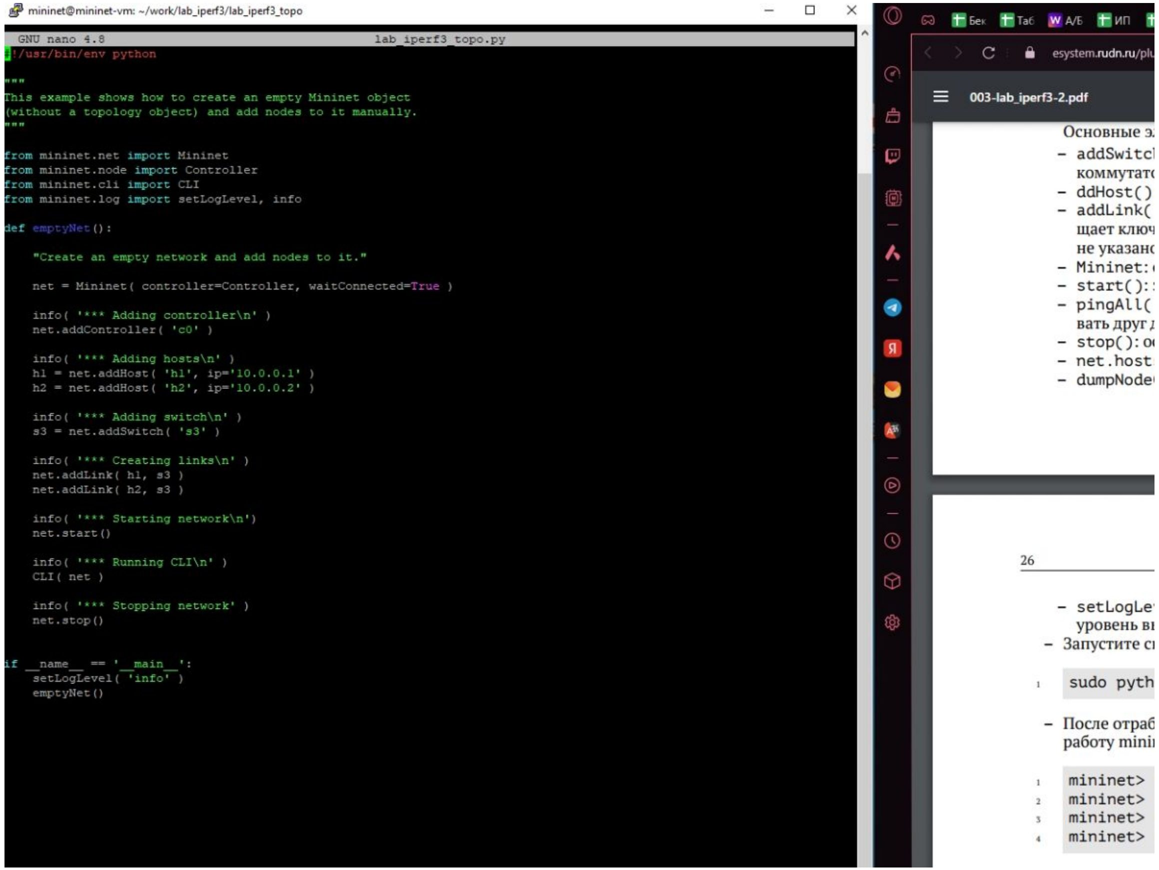
1. С помощью API Mininet создайте простейшую топологию сети, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8:
   – В каталоге /work/lab_iperf3 для работы над проектом создайте подкаталог lab_iperf3_topo и скопируйте в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet:

```
cd ~/work/lab_iperf3
mkdir lab_iperf3_topo
cd ~/work/lab_iperf3/lab_iperf3_topo
cp ~/mininet/examples/emptynet.py
   ~/work/lab_iperf3/lab_iperf3_topo
mv emptynet.py lab_iperf3_topo.py
```

   – Изучите содержание скрипта lab_iperf3_topo.py:

```
#!/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

def emptyNet():

    "Create an empty network and add nodes to it."
```

Ход работы


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

```
  GNU nano 4.8                                    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' )
    net.stop()


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

esystem.rudn.ru/plu

003-lab_iperf3-2.pdf

Основные э
- addSwitc
  коммутат
- ddHost()
- addLink(
  щает ключ
  не указанс
- Mininet:
- start():
- pingAll(
  вать друг
- stop(): о
- net.host
- dumpNode

26

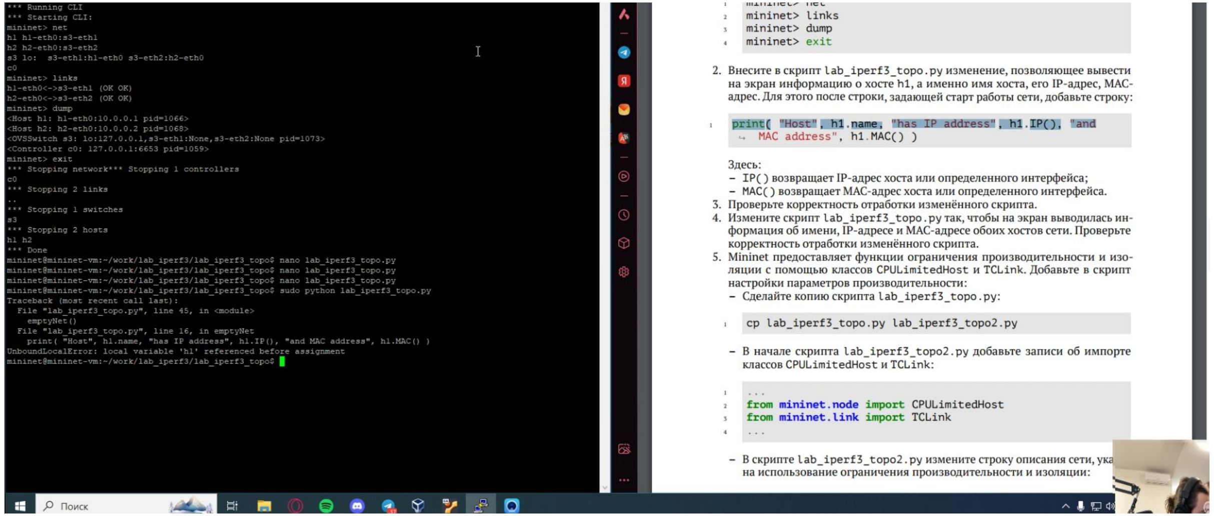
- setLogLe
  уровень в
- Запустите с

```
sudo pyth
```

- После отраб
  работу mini

```
mininet>
mininet>
mininet>
mininet>
```

Ход работы



```
*** 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=1066>
<Host h2: h2-eth0:10.0.0.2 pid=1068>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=1073>
<Controller c0: 127.0.0.1:6653 pid=1059>
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$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ sudo python lab_iperf3_topo.py
Traceback (most recent call last):
  File "lab_iperf3_topo.py", line 45, in <module>
    emptyNet()
  File "lab_iperf3_topo.py", line 16, in emptyNet
    print( "Host", h1.name, "has IP address", h1.IP(), "and MAC address", h1.MAC() )
UnboundLocalError: local variable 'h1' referenced before assignment
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$
```

```
mininet> net
mininet> links
mininet> dump
mininet> exit
```

2. Внесите 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, добавьте строку:

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

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

3. Проверьте корректность отработки изменённого скрипта.
4. Измените скрипт lab_iperf3_topo.py так, чтобы на экран выводилась информация об имени, IP-адресе и MAC-адресе обоих хостов сети. Проверьте корректность отработки изменённого скрипта.
5. Mininet предоставляет функции ограничения производительности и изоляции с помощью классов CPULimitedHost и TCLink. Добавьте в скрипт настройки параметров производительности:
   - Сделайте копию скрипта lab_iperf3_topo.py:

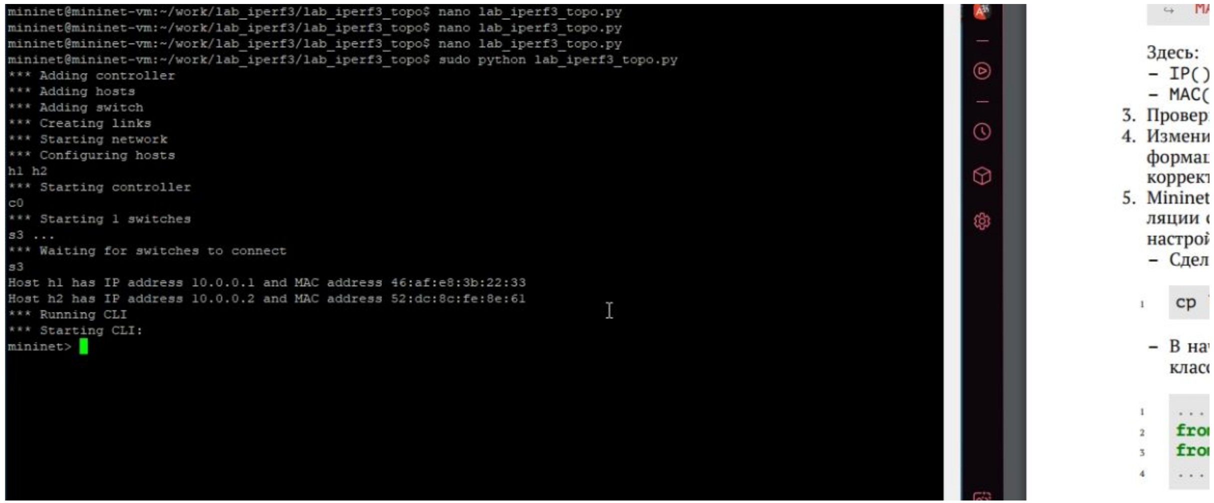
```
cp lab_iperf3_topo.py lab_iperf3_topo2.py
```

   - В начале скрипта lab_iperf3_topo2.py добавьте записи об импорте классов CPULimitedHost и TCLink:

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

   - В скрипте lab_iperf3_topo2.py измените строку описания сети, ука на использование ограничения производительности и изоляции:

Ход работы



```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano 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 46:af:e8:3b:22:33
Host h2 has IP address 10.0.0.2 and MAC address 52:dc:8c:fe:8e:61
*** Running CLI
*** Starting CLI:
mininet>
```

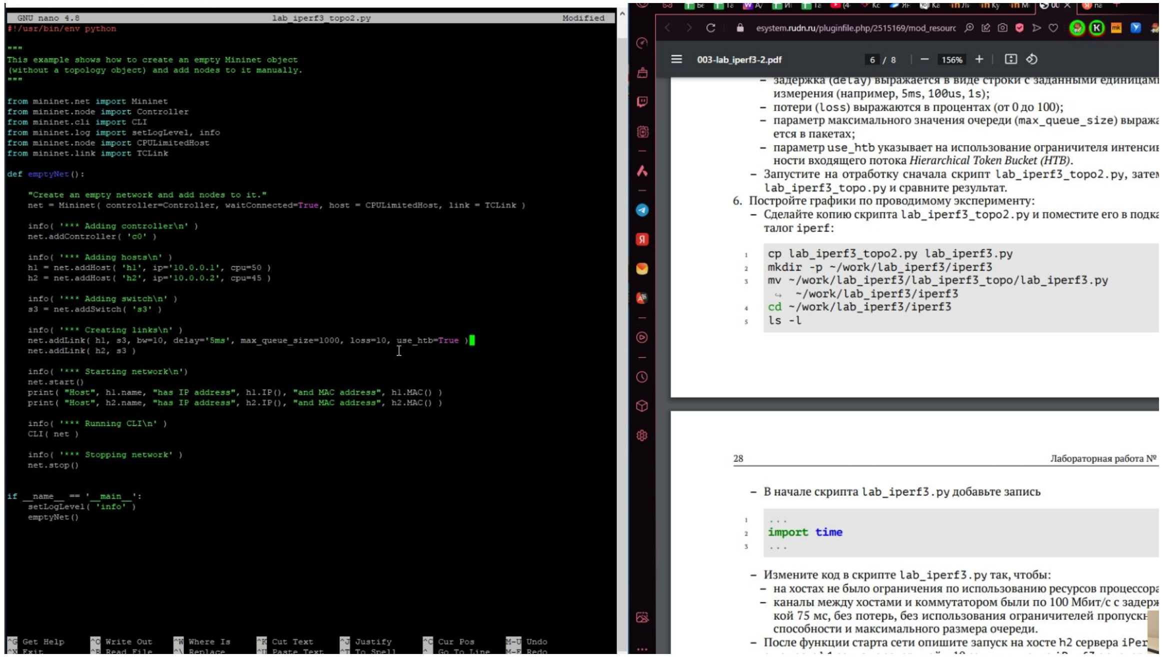
Здесь:
– IP()
– MAC(
3. Провер
4. Измени
формац
коррект
5. Mininet
ляции
настрой
– Сдел

```
cp
```

– В на
класс

```
...
fro
fro
...
```

Ход работы



```
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.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()

^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
```

– задержка (delay) выражается в виде строки с заданными единицам
измерения (например, 5ms, 100us, 1s);
– потери (loss) выражаются в процентах (от 0 до 100);
– параметр максимального значения очереди (max_queue_size) выража
ется в пакетах;
– параметр use_htb указывает на использование ограничителя интенсив
ности входящего потока *Hierarchical Token Bucket (HTB)*.

– Запустите на отработку сначала скрипт lab_iperf3_topo2.py, затем
lab_iperf3_topo.py и сравните результат.

6. Постройте графики по проводимому эксперименту:
– Сделайте копию скрипта lab_iperf3_topo2.py и поместите его в подка
талог iperf:

```
cp lab_iperf3_topo2.py lab_iperf3.py
mkdir -p ~/work/lab_iperf3/iperf3
mv ~/work/lab_iperf3/lab_iperf3_topo/lab_iperf3.py
 ↪  ~/work/lab_iperf3/iperf3
cd ~/work/lab_iperf3/iperf3
ls -l
```

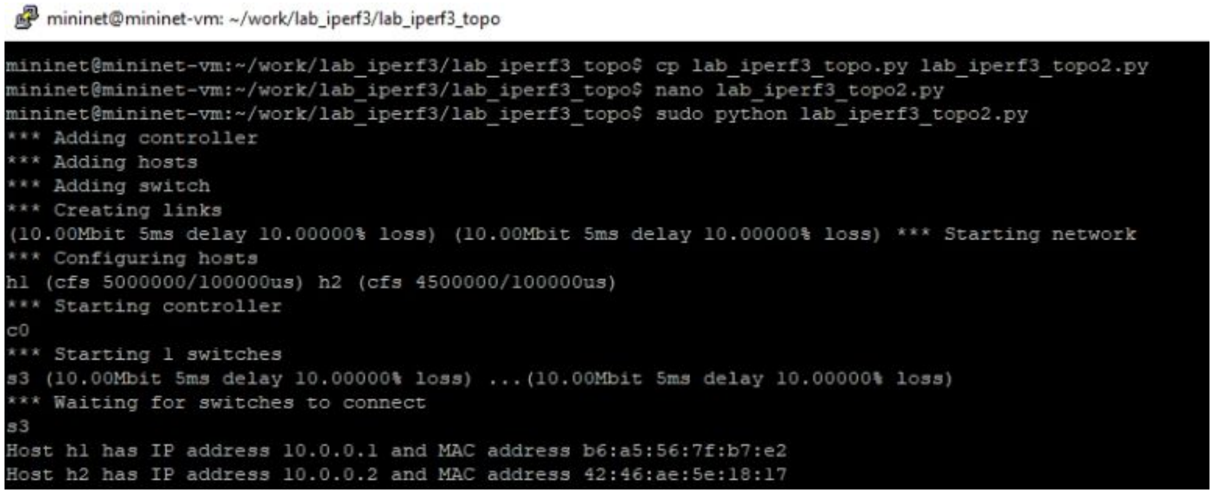
28 Лабораторная работа №

– В начале скрипта lab_iperf3.py добавьте запись

```
...
import time
...
```

– Измените 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 задерж
кой 75 мс, без потерь, без использования ограничителей пропускн
способности и максимального размера очереди.
– После функции старта сети опишите запуск на хосте h2 сервера iPer

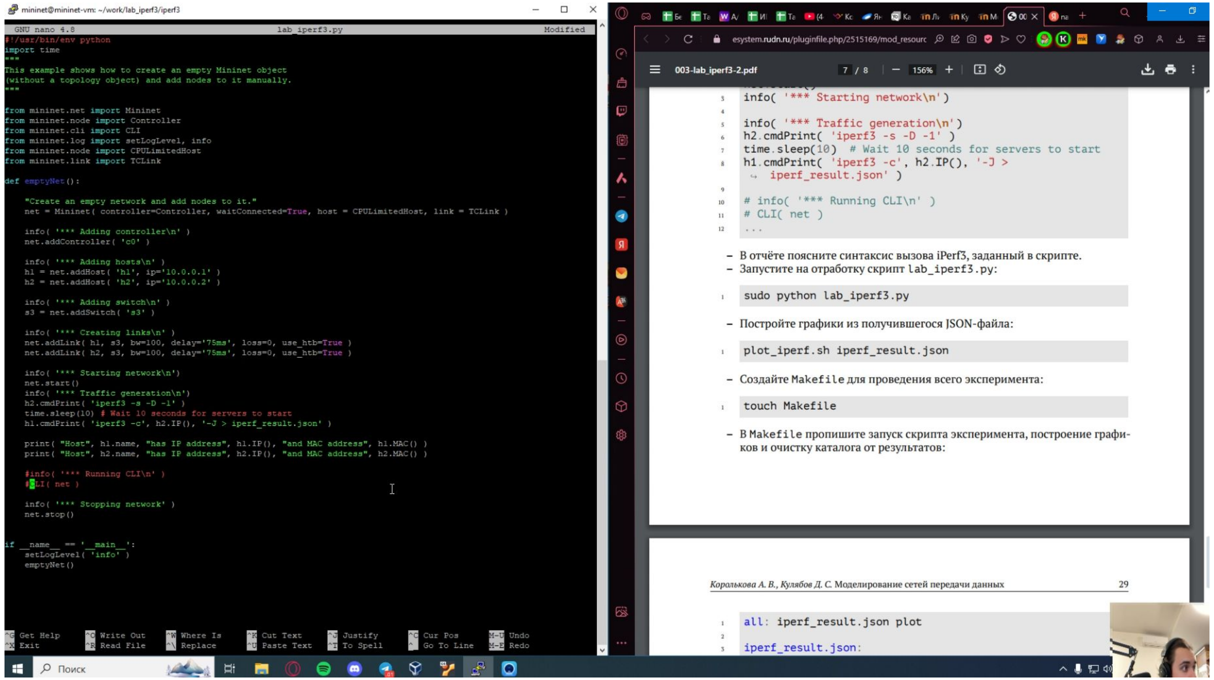
Ход работы



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

```
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ cp lab_iperf3_topo.py lab_iperf3_topo2.py
mininet@mininet-vm:~/work/lab_iperf3/lab_iperf3_topo$ nano lab_iperf3_topo2.py
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 b6:a5:56:7f:b7:e2
Host h2 has IP address 10.0.0.2 and MAC address 42:46:ae:5e:18:17
```

Ход работы



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

```
  GNU nano 4.8                    lab_iperf3.py                    Modified
#!/usr/bin/env python
import time
"""
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' )
    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, use_htb=True )
    net.addLink( h2, s3, bw=100, delay='75ms', loss=0, use_htb=True )

    info( '*** Starting network\n')
    net.start()
    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 )

    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   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

esystem.rudn.ru/pluginfile.php/2515169/mod_resourc

003-lab_iperf3-2.pdf 7 / 8 156%

```
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' )

# info( '*** Running CLI\n' )
# CLI( net )
...
```

– В отчёте поясните синтаксис вызова iPerf3, заданный в скрипте.
– Запустите на отработку скрипт `lab_iperf3.py`:

```
sudo python lab_iperf3.py
```

– Постройте графики из получившегося JSON-файла:

```
plot_iperf.sh iperf_result.json
```

– Создайте `Makefile` для проведения всего эксперимента:

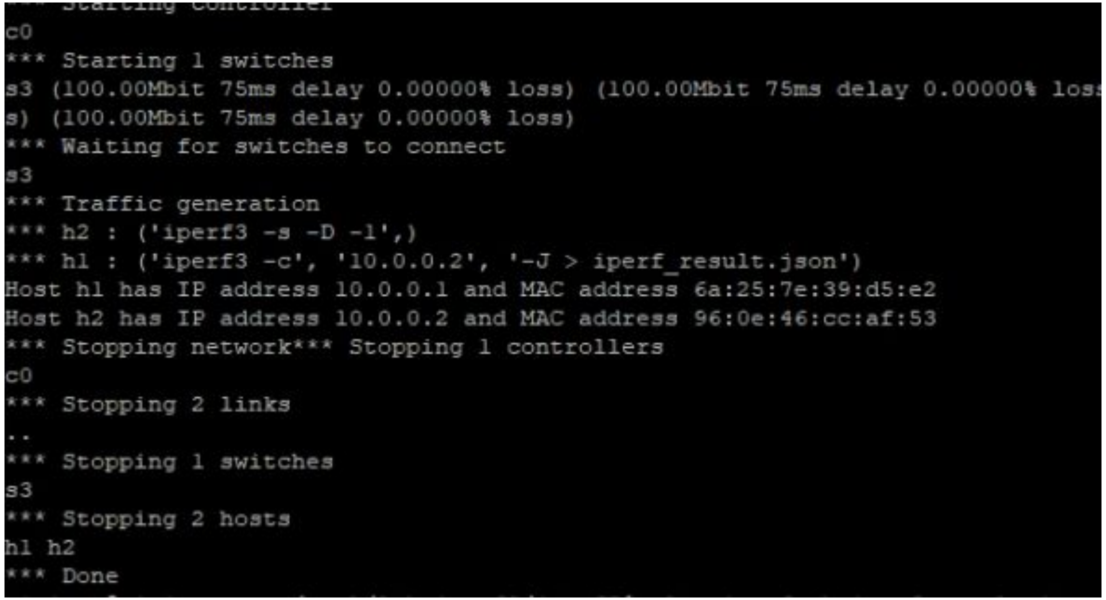
```
touch Makefile
```

– В `Makefile` пропишите запуск скрипта эксперимента, построение графиков и очистку каталога от результатов:

Королькова А. В., Кулябов Д. С. Моделирование сетей передачи данных 29

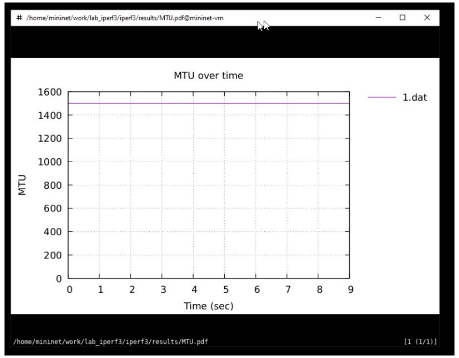
```
all: iperf_result.json plot

iperf_result.json:
```

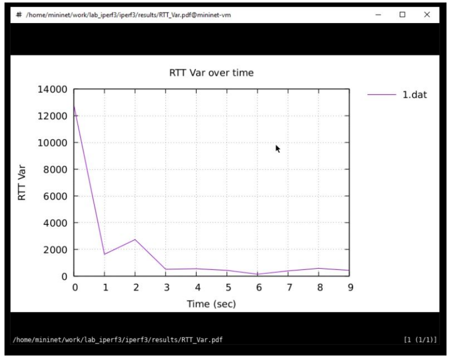
Ход работы

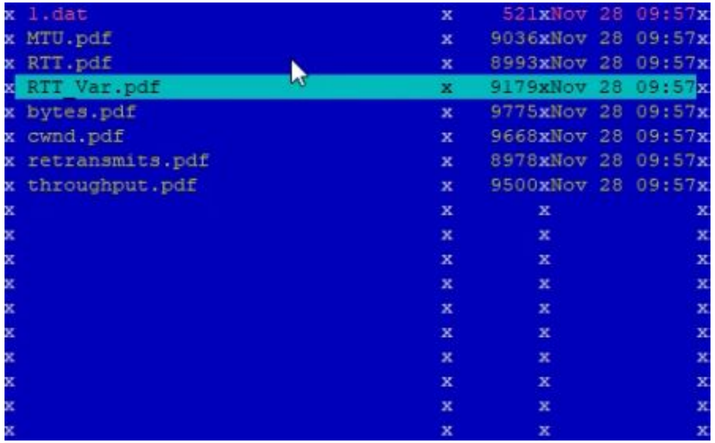


```
*** Starting controller
c0
*** Starting 1 switches
s3 (100.00Mbit 75ms delay 0.00000% loss) (100.00Mbit 75ms delay 0.00000% los
s) (100.00Mbit 75ms delay 0.00000% loss)
*** Waiting for switches to connect
s3
*** Traffic generation
*** h2 : ('iperf3 -s -D -1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
Host h1 has IP address 10.0.0.1 and MAC address 6a:25:7e:39:d5:e2
Host h2 has IP address 10.0.0.2 and MAC address 96:0e:46:cc:af:53
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s3
*** Stopping 2 hosts
h1 h2
*** Done
```

Ход работы

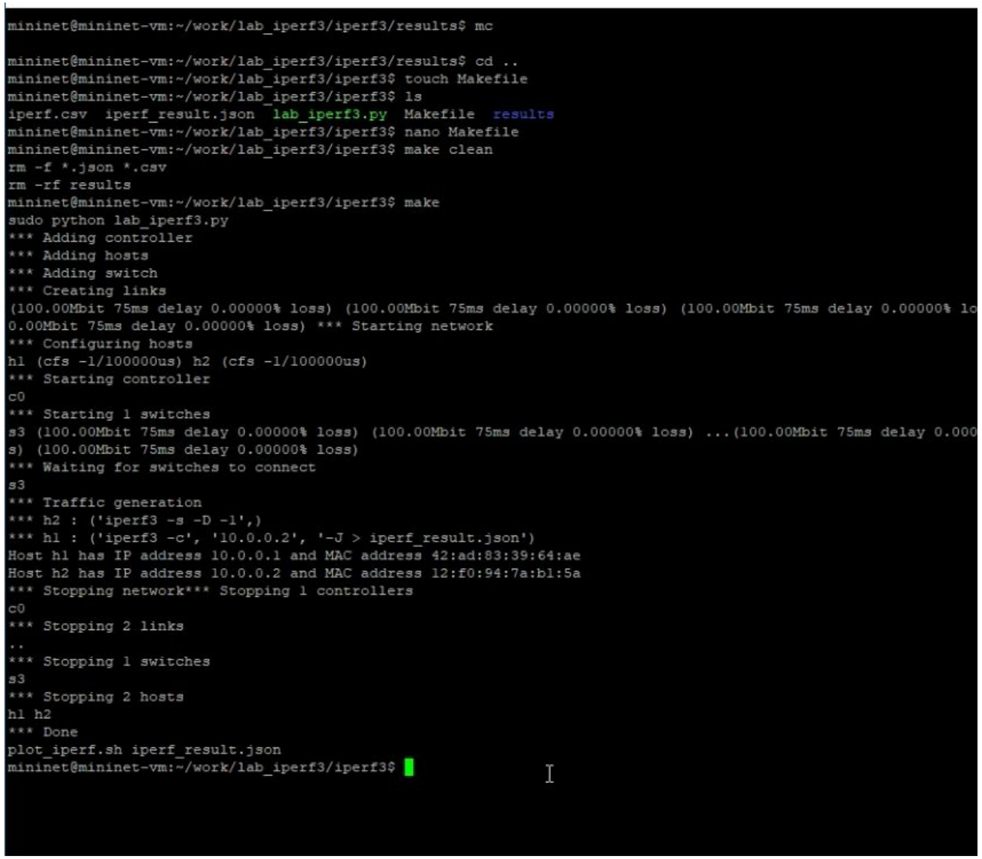






```
x 1.dat                x    521xNov 28 09:57x
x MTU.pdf              x   9036xNov 28 09:57x
x RTT.pdf              x   8993xNov 28 09:57x
x RTT_Var.pdf          x   9179xNov 28 09:57x
x bytes.pdf            x   9775xNov 28 09:57x
x cwnd.pdf             x   9668xNov 28 09:57x
x retransmits.pdf      x   8978xNov 28 09:57x
x throughput.pdf       x   9500xNov 28 09:57x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
x                      x       x            x
```

Ход работы



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

mininet@mininet-vm:~/work/lab_iperf3/iperf3/results$ cd ..
mininet@mininet-vm:~/work/lab_iperf3/iperf3$ touch Makefile
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$ nano 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% lo
0.00Mbit 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 0.000
s) (100.00Mbit 75ms delay 0.00000% loss)
*** 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 42:ad:83:39:64:ae
Host h2 has IP address 10.0.0.2 and MAC address 12:f0:94:7a:b1:5a
*** 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$
```

Вывод

Я ознакомился 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**.