

Лабораторная работа №6

**Настройка пропускной способности глобальной сети с помощью
Token Bucket Filter**

Тазаева Анастасия Анатольевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Запуск лабораторной топологии	7
3.2	Интерактивные эксперименты	9
3.2.1	Ограничение скорости на конечных хостах	9
3.2.2	Ограничение скорости на коммутаторах	11
3.2.3	Объединение NETEM и TBF	12
3.3	Воспроизведение экспериментов	13
4	Выводы	18

Список иллюстраций

2.1	Топология моделируемой сети с Token Bucket Filter	6
3.1	Подключение к mininet. Исправление прав запуска X-соединения	7
3.2	ifconfig h1	8
3.3	ifconfig h2	8
3.4	ifconfig s1	8
3.5	Проверка подключения ping h1 -> h2	8
3.6	Результат отработки iPerf3, когда отсутствуют ограничения скорости передачи данных	9
3.7	Установка пропускной способности на инт.h1-eth0	9
3.8	Проверка значения пропускной способности с помощью iperf3 . .	10
3.9	Установка ограничения скорости на коммутаторах	11
3.10	Результат отработки iPerf3, с ограничением скорости на коммутаторах	11
3.11	Удаление конфигурации на коммутаторе	11
3.12	Объединение NETEM и TBF. ч1	12
3.13	Проверка задержки между h1 h2	12
3.14	Объединение NETEM и TBF. ч2	13
3.15	Результат отработки iPerf3, с ограничением скорости на коммутаторах, введении задержки (netem+tbfb)	13
3.16	samost.py	15
3.17	Makefile	16
3.18	ping.dat	17
3.19	ping.plot	17

Список таблиц

1 Цель работы

Основной целью работы является знакомство с принципами работы дисциплины очереди Token Bucket Filter, которая формирует входящий/исходящий трафик для ограничения пропускной способности, а также получение навыков моделирования и исследования поведения трафика посредством проведения интерактивного и воспроизводимого экспериментов в Mininet.

2 Задание

1. Задайте топологию (рис. 6.3), состоящую из двух хостов и двух коммутаторов с назначенной по умолчанию mininet сетью 10.0.0.0/8.

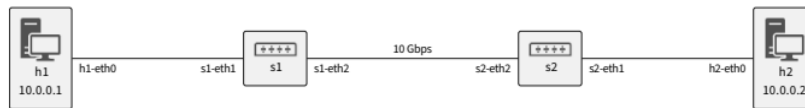


Рис. 2.1: Топология моделируемой сети с Token Bucket Filter

2. Проведите интерактивные эксперименты по ограничению пропускной способности сети с помощью TBF в эмулируемой глобальной сети.
3. Самостоятельно реализуйте воспроизводимые эксперимент по применению TBF для ограничения пропускной способности. Постройте соответствующие графики.

3 Выполнение лабораторной работы

3.1 Запуск лабораторной топологии

1. Запустила виртуальную среду с mininet. Из основной ОС подключилась к виртуальной машине (рис. 3.1). Исправила права запуска X-соединения (рис. 3.1)

```
[aatazaeva@aatazaeva ~]$ ssh -Y mininet@192.168.56.101
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Last login: Wed Dec 11 00:08:28 2024 from 192.168.56.1
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 ff3623ba66e163f75586fe76d3b2d70c
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 ff3623ba66e163f75586fe76d3b2d70c
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 ff3623ba66e163f75586fe76d3b2d70c
root@mininet-vm:~# exit
logout
```

Рис. 3.1: Подключение к mininet. Исправление прав запуска X-соединения

2. Создала простейшую топологию, состоящую из двух хостов и двух коммутаторов с назначенной по умолчанию mininet сетью 10.0.0.0/8.

```
sudo mn --topo=linear,2 -x
```

3. На хостах h1 и h2, а также на коммутаторе ввела команду ifconfig (рис. 3.2 - 3.4), чтобы отобразить информацию, относящуюся к их сетевым интерфейсам и назначенным им IP-адресам. В дальнейшем при работе с NETEM и командой tc будут использоваться интерфейсы h1-eth0 и h2-eth0 .

```

"host: h1" (на mininet-vm)
root@mininet-vm:/home/mininet# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
        ether 2a:20:cd:7c:83:59 txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Рис. 3.2: ifconfig h1

```

"host: h2" (на mininet-vm)
root@mininet-vm:/home/mininet# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
        ether 06:30:2e:3a:29:ff txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Рис. 3.3: ifconfig h2

```

"switch: s2" (root) (на mininet-vm)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 4a:2f:03:ff:22:69 txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Рис. 3.4: ifconfig s1

4. Проверила подключение между хостами h1 и h2 с помощью команды ping с параметром -c 4 (рис. 3.5).

```

root@mininet-vm:/home/mininet# ping -c 4 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.046 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.033/0.044/0.054/0.007 ms

```

Рис. 3.5: Проверка подключения ping h1 -> h2

7. В терминале хоста h2 запустила iPerf3 в режиме сервера:


```
iperf3 -s
```

8. В терминале хоста h1 запустила iPerf3 в режиме клиента (рис. 3.6):

```
iperf3 -c 10.0.0.2
```

```
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 47382 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer    Bitrate      Retr  Cwnd
[ 7]  0.00-1.00    sec  3.90 GBytes  33.5 Gbits/sec  9    23.7 MBytes
[ 7]  1.00-2.00    sec  3.84 GBytes  32.9 Gbits/sec  0    23.7 MBytes
[ 7]  2.00-3.00    sec  3.87 GBytes  33.3 Gbits/sec  0    23.7 MBytes
[ 7]  3.00-4.00    sec  3.89 GBytes  33.4 Gbits/sec  0    23.7 MBytes
[ 7]  4.00-5.00    sec  3.90 GBytes  33.5 Gbits/sec  0    23.7 MBytes
[ 7]  5.00-6.00    sec  3.87 GBytes  33.2 Gbits/sec  0    23.7 MBytes
[ 7]  6.00-7.00    sec  3.78 GBytes  32.5 Gbits/sec  0    23.7 MBytes
[ 7]  7.00-8.00    sec  3.88 GBytes  33.4 Gbits/sec  0    23.7 MBytes
[ 7]  8.00-9.00    sec  3.84 GBytes  33.0 Gbits/sec  0    23.7 MBytes
[ 7]  9.00-10.00   sec  3.85 GBytes  33.1 Gbits/sec  0    23.7 MBytes
- - - - -
[ ID] Interval      Transfer    Bitrate      Retr
[ 7]  0.00-10.00   sec  38.6 GBytes  33.2 Gbits/sec  9
[ 7]  0.00-10.00   sec  38.6 GBytes  33.2 Gbits/sec
                                sender
                                receiver

iperf Done.
-
```

Рис. 3.6: Результат отработки iPerf3, когда отсутствуют ограничения скорости передачи данных

3.2 Интерактивные эксперименты

3.2.1 Ограничение скорости на конечных хостах

Команду tc можно применить к сетевому интерфейсу устройства для формирования исходящего трафика. Требуется ограничить скорость отправки данных с конечного хоста с помощью фильтра Token Bucket Filter (tbf).

1. Изменила пропускную способность хоста h1 , установив пропускную способность на 10 Гбит/с на интерфейсе h1-eth0 и параметры TBF-фильтра: (рис. 3.7):

```
sudo tc qdisc add dev h1-eth0 root tbf rate 10gbit burst 5000000 limit 15000000
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root tbf rate 10gb
it burst 5000000 limit 15000000_
```

Рис. 3.7: Установка пропускной способности на инт.h1-eth0

Здесь: - sudo : включить выполнение команды с более высокими привилегиями безопасности; - tc : вызвать управление трафиком Linux; - qdisc : изменить дисциплину очередей сетевого планировщика; - add (добавить): создать новое правило; - dev h1-eth0 root : интерфейс, на котором будет применяться правило; - tbf : использовать алгоритм Token Bucket Filter; - rate : указать скорость передачи (10 Гбит/с); - burst : количество байтов, которое может поместиться в корзину (5 000 000); - limit : размер очереди в байтах (15 000 000).

2. Фильтр tbf требует установки значения всплеска при ограничении скорости.

Это значение должно быть достаточно высоким, чтобы обеспечить установленную скорость. Она должна быть не ниже указанной частоты, делённой на HZ, где HZ — тактовая частота, настроенная как параметр ядра, и может быть извлечена с помощью следующей команды:

```
egrep '^CONFIG_HZ_[0-9]+' /boot/config-`uname -r`
```

Для расчёта значения всплеска (burst) необходимо скорость передачи (10 Гбит/с или 10 Gbps = 10,000,000,000 bps) разделить на полученное таким образом значение HZ (на хосте h1 HZ = 250): $Burst = 10,000,000,000 / 250 = 40,000,000 \text{ bits} = 40,000,000 / 8 \text{ bytes} = 5,000,000 \text{ bytes}$.

3. С помощью iPerf3 проверила, что значение пропускной способности изменилось (рис. 3.8)

```
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 47386 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer    Bitrate      Retr  Cwnd
[ 7] 0.00-1.00 sec  1.12 GBytes 9.65 Gbits/sec  9  13.8 MBytes
[ 7] 1.00-2.00 sec  1.10 GBytes 9.49 Gbits/sec  0  13.8 MBytes
[ 7] 2.00-3.00 sec  1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
[ 7] 3.00-4.00 sec  1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
[ 7] 4.00-5.00 sec  1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
[ 7] 5.00-6.00 sec  1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
[ 7] 6.00-7.00 sec  1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
[ 7] 7.00-8.00 sec  1.11 GBytes 9.50 Gbits/sec  0  13.8 MBytes
[ 7] 8.00-9.00 sec  1.12 GBytes 9.60 Gbits/sec  0  13.8 MBytes
[ 7] 9.00-10.00 sec 1.11 GBytes 9.56 Gbits/sec  0  13.8 MBytes
- - - - -
[ ID] Interval      Transfer    Bitrate      Retr
[ 7] 0.00-10.00 sec 11.1 GBytes 9.56 Gbits/sec  9      sender
[ 7] 0.00-10.01 sec 11.1 GBytes 9.55 Gbits/sec                receiver
```

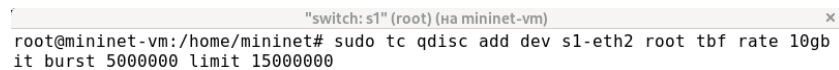
Рис. 3.8: Проверка значения пропускной способности с помощью iPerf3

3.2.2 Ограничение скорости на коммутаторах

При ограничении скорости на интерфейсе s1-eth2 коммутатора s1 все сеансы связи между коммутатором s1 и коммутатором s2 будут фильтроваться в соответствии с применяемыми правилами.

1. Применила правило ограничения скорости tbf с параметрами rate = 10gbit, burst = 5,000,000, limit = 15,000,000 к интерфейсу s1-eth2 коммутатора s1, который соединяет его с коммутатором s2 (рис. 3.9):

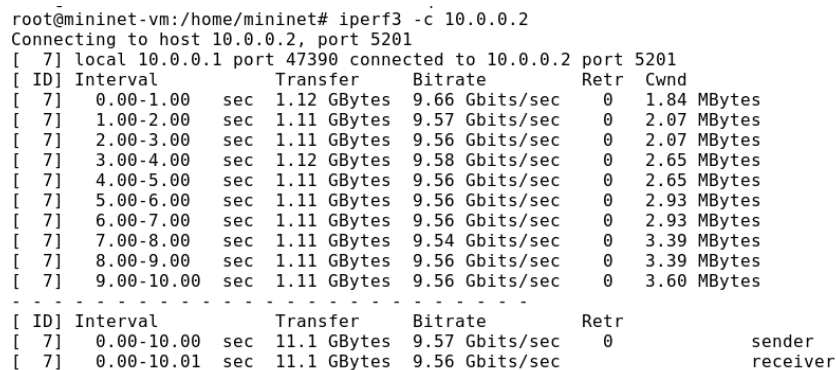
```
sudo tc qdisc add dev s1-eth2 root tbf rate 10gbit burst 5000000 limit 15000000
```



```
"switch: s1" (root) (на mininet-vm) x
root@mininet-vm:/home/mininet# sudo tc qdisc add dev s1-eth2 root tbf rate 10gb
it burst 5000000 limit 15000000
```

Рис. 3.9: Установка ограничения скорости на коммутаторах

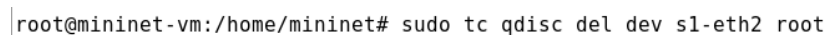
2. Проверила конфигурацию с помощью инструмента iperf3 для измерения пропускной способности(рис. 3.10).



```
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 47390 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer    Bitrate    Retr  Cwnd
[ 7]  0.00-1.00  sec  1.12 GBytes  9.66 Gbits/sec    0   1.84 MBytes
[ 7]  1.00-2.00  sec  1.11 GBytes  9.57 Gbits/sec    0   2.07 MBytes
[ 7]  2.00-3.00  sec  1.11 GBytes  9.56 Gbits/sec    0   2.07 MBytes
[ 7]  3.00-4.00  sec  1.12 GBytes  9.58 Gbits/sec    0   2.65 MBytes
[ 7]  4.00-5.00  sec  1.11 GBytes  9.56 Gbits/sec    0   2.65 MBytes
[ 7]  5.00-6.00  sec  1.11 GBytes  9.56 Gbits/sec    0   2.93 MBytes
[ 7]  6.00-7.00  sec  1.11 GBytes  9.56 Gbits/sec    0   2.93 MBytes
[ 7]  7.00-8.00  sec  1.11 GBytes  9.54 Gbits/sec    0   3.39 MBytes
[ 7]  8.00-9.00  sec  1.11 GBytes  9.56 Gbits/sec    0   3.39 MBytes
[ 7]  9.00-10.00 sec  1.11 GBytes  9.56 Gbits/sec    0   3.60 MBytes
-----
[ ID] Interval      Transfer    Bitrate    Retr
[ 7]  0.00-10.00  sec  11.1 GBytes  9.57 Gbits/sec    0
[ 7]  0.00-10.01  sec  11.1 GBytes  9.56 Gbits/sec    0
sender
receiver
```

Рис. 3.10: Результат отработки iPerf3, с ограничением скорости на коммутаторах

3. Удалила модифицированную конфигурацию на коммутаторе s1 (рис. 3.11).



```
root@mininet-vm:/home/mininet# sudo tc qdisc del dev s1-eth2 root
```

Рис. 3.11: Удаление конфигурации на коммутаторе

3.2.3 Объединение NETEM и TBF

NETEM используется для изменения задержки, джиттера, повреждения пакетов и т.д. TBF может использоваться для ограничения скорости. Утилита tc позволяет комбинировать несколько модулей. При этом первая дисциплина очереди (qdisc1) присоединяется к корневой метке, последующие дисциплины очереди можно прикрепить к своим родителям, указав правильную метку.

1. Объединила NETEM и TBF, введя на интерфейсе s1-eth2 коммутатора s1 задержку, джиттер, повреждение пакетов и указав скорость (рис. 3.12):

```
sudo tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms
```

Рис. 3.12: Объединение NETEM и TBF. ч1

Здесь ключевое слово handle задаёт дескриптор подключения, имеющий смысл очерёдности подключения разных дисциплин qdisc .

2. Убедилась, что соединение от хоста h1 к хосту h2 имеет заданную задержку. Для этого запустила команду ping с параметром -c 4 с терминала хоста h1 (рис. 3.13).

```
root@mininet-vm:/home/mininet# ping -c 4 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=13.2 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=10.8 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 10.792/11.436/13.234/1.038 ms
```

Рис. 3.13: Проверка задержки между h1 h2

3. Добавила второе правило на коммутаторе s1 , которое задаёт ограничение скорости с помощью tbf с параметрами rate =2gbit, burst =1,000,000, limit =2,000,000 (рис. 3.14):

```
sudo tc qdisc add dev s1-eth2 parent 1: handle 2: tbf rate 2gbit burst 1000000 limit 2
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev s1-eth2 parent 1: handle 2
: tbf rate 2gbit burst 1000000 limit 2000000
```

Рис. 3.14: Объединение NETEM и TBF. ч2

4. Проверила конфигурацию с помощью инструмента iperf3 для измерения пропускной способности (рис. 3.15).

```
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 47394 connected to 10.0.0.2 port 5201
```

[ID]	Interval		Transfer	Bitrate	Retr	Cwnd
[7]	0.00-1.00	sec	199 MBytes	1.67 Gbits/sec	707	2.34 MBytes
[7]	1.00-2.00	sec	220 MBytes	1.84 Gbits/sec	0	2.45 MBytes
[7]	2.00-3.00	sec	228 MBytes	1.91 Gbits/sec	0	2.54 MBytes
[7]	3.00-4.00	sec	229 MBytes	1.92 Gbits/sec	0	2.60 MBytes
[7]	4.00-5.00	sec	228 MBytes	1.91 Gbits/sec	0	2.65 MBytes
[7]	5.00-6.00	sec	229 MBytes	1.92 Gbits/sec	0	2.68 MBytes
[7]	6.00-7.00	sec	228 MBytes	1.91 Gbits/sec	0	2.70 MBytes
[7]	7.00-8.00	sec	229 MBytes	1.92 Gbits/sec	0	2.73 MBytes
[7]	8.00-9.00	sec	228 MBytes	1.91 Gbits/sec	0	2.79 MBytes
[7]	9.00-10.00	sec	228 MBytes	1.91 Gbits/sec	0	2.85 MBytes

```

- - - - -
[ ID] Interval      Transfer  Bitrate      Retr
[ 7]  0.00-10.00  sec  2.19 GBytes  1.88 Gbits/sec  707
[ 7]  0.00-10.01  sec  2.18 GBytes  1.87 Gbits/sec

sender
receiver
```

Рис. 3.15: Результат отработки iPerf3, с ограничением скорости на коммутаторах, введении задержки (netem+tbfb)

3.3 Воспроизведение экспериментов

1. В виртуальной среде mininet в своём рабочем каталоге с проектами создала каталог tbf и перешла в него. Создала скрипт для эксперимента samost.py (рис. 3.16).

```
#!/usr/bin/env python
```

```
"""
```

```
Simple experiment.
```

```
Output: ping.dat
```

```
"""
```

```

from mininet.net import Mininet
from mininet.node import Controller
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 )

    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' )
    s1 = net.addSwitch( 's1' )

    info('***creating links\n' )
    net.addLink( h1, s1)
    net.addLink( h2, s1)

    info('***starting network' )
    net.start()

    info('***set loss\n' )

```

```

h1.cmdPrint( 'tc qdisc add dev h1-eth0 root tbf rate 10gbit burst 5000000 limit 15

time.sleep(10)

info('***proverka propysknoi sposobnosti')
h2.cmdPrint('iperf3 -s &')
time.sleep(10)
h1.cmdPrint('iperf3 -c', h2.IP(), ' | grep "MBytes" | awk \'{print $7}\'' > ping.da

info('***stopping network' )
net.stop()

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

24     h1 = net.addHost( 'h1', ip='10.0.0.1' )
25     h2 = net.addHost( 'h2', ip='10.0.0.2' )
26     info('*** adding switch\n' )
27     s1 = net.addSwitch( 's1' )
28
29     info('***creating links\n' )
30     net.addLink( h1, s1)
31     net.addLink( h2, s1)
32
33     info('***starting network' )
34     net.start()
35
36     info('***set loss\n' )
37     h1.cmdPrint( 'tc qdisc add dev h1-eth0 root tbf rate 10gbit burst
5000000 limit 15000000' )
38
39     time.sleep(10)
40
41     info('***proverka propysknoi sposobnosti')
42     h2.cmdPrint('iperf3 -s &')
43     time.sleep(10)
44     h1.cmdPrint('iperf3 -c', h2.IP(), ' | grep "MBytes" | awk \'{print $7}\''
> ping.dat')
45
46     info('***stopping network' )
47     net.stop()
48
49 if __name__ == '__main__':
50     setLogLevel( 'info' )
51     emptyNet()

```

Рис. 3.16: samost.py

2. Создала Makefile для управления процессом проведения эксперимента (рис.

3.17).

```
all: ping.dat ping.png
```

```
ping.dat:
```

```
    sudo python samost.py
```

```
    sudo chown mininet:mininet ping.dat
```

```
ping.png: ping.dat
```

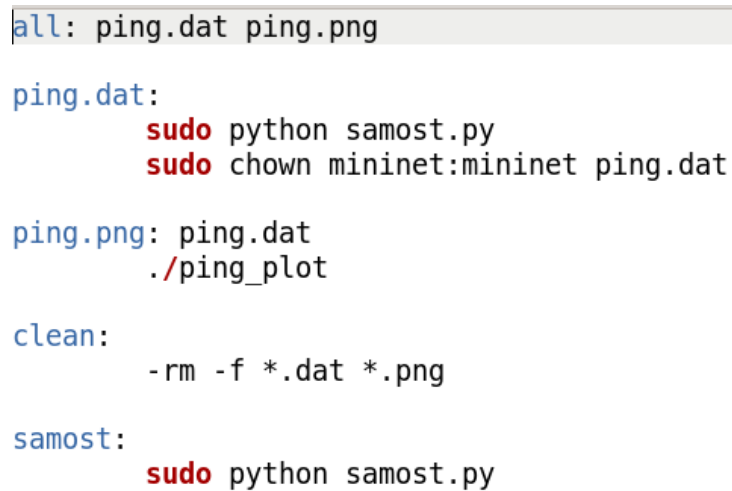
```
    ./ping_plot
```

```
clean:
```

```
    -rm -f *.dat *.png
```

```
samost:
```

```
    sudo python samost.py
```



```
all: ping.dat ping.png

ping.dat:
    sudo python samost.py
    sudo chown mininet:mininet ping.dat

ping.png: ping.dat
    ./ping_plot

clean:
    -rm -f *.dat *.png

samost:
    sudo python samost.py
```

Рис. 3.17: Makefile

3. В файл ping.dat вывелась информация о скорости (bitrate) (рис. 3.18).



Рис. 3.18: ping.dat

4. По результатам ping.dat создала график (рис. 3.19).

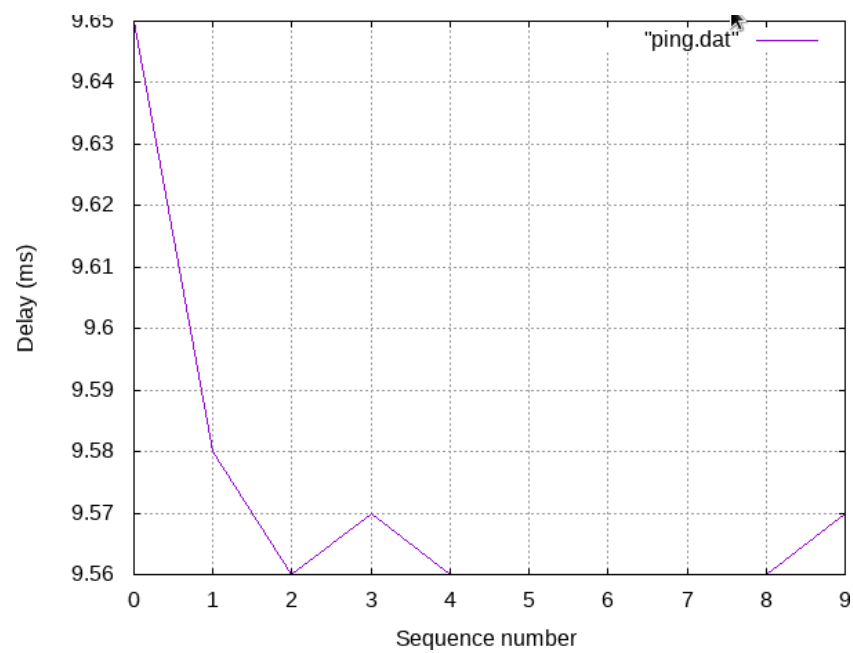


Рис. 3.19: ping.plot

4 Выводы

В ходе лабораторной работы я знакомилась с принципами работы дисциплины очереди Token Bucket Filter, которая формирует входящий/исходящий трафик для ограничения пропускной способности, а также получила навыки моделирования и исследования поведения трафика посредством проведения интерактивного и воспроизводимого экспериментов в Mininet.