

Лабораторная работа №6

Дисциплина: Моделирование сетей передачи данных

Хрусталев Влад Николаевич

Содержание

1 Цель работы	4
2 Теоретическое введение	5
3 Задание	6
4 Выполнение лабораторной работы	7
4.1 Запуск лабораторной топологии	7
4.2 Интерактивные эксперименты	9
4.2.1 Ограничение скорости на конечных хостах	9
4.2.2 Ограничение скорости на коммутаторах	12
4.2.3 Объединение NETEM и TBF	13
4.3 Воспроизведение экспериментов	15
5 Выводы	25
Список литературы	26

Список иллюстраций

4.1	Настройка X-соединения	7
4.2	Запуск простейшей топологии	7
4.3	Информация на хостах и коммутаторах об интерфесах	8
4.4	Тестирование соединения между h1 и h2	8
4.5	Проверка изначальной пропускной способности	9
4.6	Ограничение скорости на конечных хостах и тест	11
4.7	Ограничение скорости на коммутаторе и тест	12
4.8	Удаление модифицированной конфигурации на коммутаторе s1 .	13
4.9	Добавление NETEM правила задержки пакетов	14
4.10	Добавление 2ого правила ограничения скорости и тест	15
4.11	Удаление модифицированной конфигурации на коммутаторе s1 .	15
4.12	Создание необходимых файлов	16
4.13	Листинг exp1/lab_tbf_i.py	16
4.14	Листинг Makefile и ping_plot	17
4.15	Запуск exp1	17
4.16	График iperf3.png из exp1	18
4.17	График ping.png из exp1	18
4.18	Листинг exp2/lab_tbf_i.py	19
4.19	Запуск exp2	20
4.20	График iperf3.png из exp2	20
4.21	График ping.png из exp2	21
4.22	Листинг exp3/lab_tbf_i.py	22
4.23	Запуск exp3	23
4.24	График iperf3.png из exp3	24
4.25	График ping.png из exp1	24

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 Теоретическое введение

Mininet[1] – 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.

3 Задание

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

4 Выполнение лабораторной работы

4.1 Запуск лабораторной топологии

Запустим виртуальную машину и настроим права запуска X-соединения(рис. 4.1).

```
Last login: Sat Nov 22 02:29:37 2020 from 192.168.56.1
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 25dbbe81293c4843b339fb714283e440
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 25dbbe81293c4843b339fb714283e440
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 25dbbe81293c4843b339fb714283e440
root@mininet-vm:~# exit
logout
mininet@mininet-vm:~$
```

Рис. 4.1: Настройка X-соединения

Зададим простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 4.2).

```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo mn --topo=linear,2 -x
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Running terms on localhost:10.0
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

Рис. 4.2: Запуск простейшей топологии

На хостах h1 и h2 и на коммутаторах s1, s2 введем команду ifconfig, 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, h2-eth0 и s1-eth2 (рис. 4.3)

```

root@mininet-vm:/home/mininet# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST mtu 1500
ether 08:00:27:e8:34:b9 txqueuelen 1000 (Ethernet)
RX packets 1946 bytes 496074 (496.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1939 bytes 985673 (985.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST mtu 1500
inet 10.0.0.2 brd 255.255.255.255 broadcast 10.0.0.2.255
ether 08:00:27:e8:34:b9 txqueuelen 1000 (Ethernet)
RX packets 591 bytes 53261 (53.2 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 591 bytes 569102 (569.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING mtu 65536
inet 127.0.0.1 brd 127.0.0.1 netmask 255.255.255.0
loop txqueuelen 1000 (Local Loopback)
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

root@mininet-vm:/home/mininet# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST mtu 1500
ether 12:3a:33:83:65:34 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

lo: flags=73<UP,LOOPBACK,RUNNING mtu 65536
inet 127.0.0.1 brd 127.0.0.1 netmask 255.255.255.0
loop txqueuelen 1000 (Local Loopback)
RX packets 1126 bytes 260532 (260.5 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1126 bytes 260532 (260.5 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST mtu 1500
inet 10.0.0.1 brd 255.255.255.255 broadcast 10.0.0.1.255
ether 46:45:a6:f4:e3:8d 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

lo: flags=73<UP,LOOPBACK,RUNNING mtu 65536
inet 127.0.0.1 brd 127.0.0.1 netmask 255.255.255.0
loop txqueuelen 1000 (Local Loopback)
RX packets 1243 bytes 266844 (266.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1243 bytes 266844 (266.8 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Рис. 4.3: Информация на хостах и коммутаторах об интерфесах

Проверим подключение между хостами h1 и h2 с помощью команды ping с параметром -c 6(рис. 4.4)

```

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=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.116 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.057/0.098/0.116/0.024 ms
root@mininet-vm:/home/mininet# 

root@mininet-vm:/home/mininet# ping -c 4 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=4.46 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.155 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.148 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.108 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.108/1.217/4.460/1.872 ms
root@mininet-vm:/home/mininet#

```

Рис. 4.4: Тестирование соединения между h1 и h2

В терминале хоста h2 запустим iPerf3 в режиме сервера: iperf3 -s. В терми-

нале хоста h1 запустим iPerf3 в режиме клиента: iperf3 -c 10.0.0.2. После завершения работы iPerf3 на хосте h1 остановите iPerf3 на хосте h2, нажав Ctrl + c. (рис. 4.5). Как видно скорости 7-11 Gbits/sec

The screenshot shows two terminal windows side-by-side. The left window, titled "host: h1@mininet-vm", displays the client's perspective of the iPerf3 test. It shows a series of data transfers from h1 to h2, with intervals of 1 second. The transfer rates fluctuate between 1.29 and 8.67 Gbytes/sec. The right window, titled "host: h2@mininet-vm", displays the server's perspective. It shows an accepted connection from h1 and a series of data transfers to h1, with intervals of 1 second. The transfer rates are consistently around 7-8 Gbytes/sec. Both windows include command-line interfaces at the bottom.

```

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 50864 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer     Bitrate   Retr Cwnd
[ 7]  0.00-1.00    sec  1.29 GBytes  11.1 Gbytes/sec  0  8.22 MBytes
[ 7]  1.00-2.00    sec  1.23 GBytes  10.6 Gbytes/sec  0  8.22 MBytes
[ 7]  2.00-3.00    sec  792 MBytes  6.65 Gbytes/sec  0  8.22 MBytes
[ 7]  3.00-4.00    sec  726 MBytes  6.11 Gbytes/sec  0  8.22 MBytes
[ 7]  4.00-5.01    sec  702 MBytes  5.82 Gbytes/sec  0  8.22 MBytes
[ 7]  5.01-6.00    sec  762 MBytes  6.48 Gbytes/sec  0  8.22 MBytes
[ 7]  6.00-7.00    sec  726 MBytes  6.07 Gbytes/sec  0  8.22 MBytes
[ 7]  7.00-8.02    sec  838 MBytes  6.95 Gbytes/sec  1  8.22 MBytes
[ 7]  8.02-9.00    sec  906 MBytes  7.70 Gbytes/sec  0  8.22 MBytes
[ 7]  9.00-10.00   sec  839 MBytes  7.04 Gbytes/sec  1  8.22 MBytes
[ ID] Interval      Transfer     Bitrate   Retr
[ 7]  0.00-10.00   sec  8.67 GBytes  7.45 Gbytes/sec  2
[ 7]  0.00-10.01   sec  8.67 GBytes  7.44 Gbytes/sec

iperf Done.
root@mininet-vm:/home/mininet# 

-----[host: h2@mininet-vm]
root@mininet-vm:/home/mininet# iperf3 -s
warning: this system does not seem to support IPv6 - trying IPv4
-----
Server listening on 5201
-----
Accepted connection from 10.0.0.1, port 50862
[ 7] local 10.0.0.2 port 5201 connected to 10.0.0.1 port 50864
[ ID] Interval      Transfer     Bitrate
[ 7]  0.00-1.00    sec  1.29 GBytes  11.1 Gbytes/sec
[ 7]  1.00-2.02    sec  1.23 GBytes  10.3 Gbytes/sec
[ 7]  2.02-3.00    sec  796 MBytes  6.82 Gbytes/sec
[ 7]  3.00-4.01    sec  721 MBytes  6.01 Gbytes/sec
[ 7]  4.01-5.00    sec  696 MBytes  5.88 Gbytes/sec
[ 7]  5.00-6.00    sec  773 MBytes  6.50 Gbytes/sec
[ 7]  6.00-7.00    sec  727 MBytes  6.10 Gbytes/sec
[ 7]  7.00-8.00    sec  827 MBytes  6.92 Gbytes/sec
[ 7]  8.00-9.00    sec  917 MBytes  7.71 Gbytes/sec
[ 7]  9.00-10.00   sec  840 MBytes  7.05 Gbytes/sec
[ 7]  10.00-10.01   sec  64.4 KBytes  89.3 Mbytes/sec
[ ID] Interval      Transfer     Bitrate
[ 7]  0.00-10.01   sec  8.67 GBytes  7.44 Gbytes/sec

-----Server listening on 5201

```

Рис. 4.5: Проверка изначальной пропускной способности

4.2 Интерактивные эксперименты

4.2.1 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. Измените пропускную способность хоста h1, установив пропускную способность на 10 Гбит/с на интерфейсе h1-eth0 и параметры TBF-фильтра:
`sudo tc qdisc add dev h1-eth0 root tbf rate 10gbit burst 5000000 limit 15000000.`

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 (5000000); - limit: размер очереди в байтах (15000000).

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 bits = 40,000,000 / 8 bytes = 5,000,000 bytes.

3. С помощью Iperf3 проверим, что значение пропускной способности изменилось

- В терминале хоста h2 запустим iPerf3 в режиме сервера: `iperf3 -s`
- В терминале хоста h2 запустим iPerf3 в режиме клиента: `iperf3 -c 10.0.0.2`

- После завершения работы iPerf3 на хосте h1 остановим iPerf3 на хосте h2, нажав Ctrl + c.

Получилась скорость передачи 7-9 Gbits/sec что соответствует заданным ограничениям.

4. Удалим модифицированную конфигурацию на хосте h1: sudo tc qdisc del dev h1-eth0 root (рис. 4.6)

```

X "host: h1@mininet-vm"
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root tbf rate 10gbit b
urst 5000000 limit 15000000
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 50868 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer     Bitrate    Retr  Cwnd
[ 7]  0.00-1.00   sec  1.06 GBytes  9.09 Gbits/sec  0  8.11 MBytes
[ 7]  1.00-2.01   sec  1.07 GBytes  9.17 Gbits/sec  0  8.11 MBytes
[ 7]  2.01-3.00   sec  800 MBytes  6.74 Gbits/sec  0  8.11 MBytes
[ 7]  3.00-4.01   sec  778 MBytes  6.49 Gbits/sec  0  8.11 MBytes
[ 7]  4.01-5.00   sec  815 MBytes  6.87 Gbits/sec  1  8.11 MBytes
[ 7]  5.00-6.00   sec  789 MBytes  6.64 Gbits/sec  0  8.11 MBytes
[ 7]  6.00-7.00   sec  664 MBytes  5.57 Gbits/sec  0  8.11 MBytes
[ 7]  7.00-8.00   sec  761 MBytes  6.39 Gbits/sec  0  8.11 MBytes
[ 7]  8.00-9.00   sec  831 MBytes  6.95 Gbits/sec  0  8.11 MBytes
[ 7]  9.00-10.01  sec  785 MBytes  6.52 Gbits/sec  0  8.11 MBytes
[-----]
[ ID] Interval      Transfer     Bitrate    Retr
[ 7]  0.00-10.01  sec  8.21 GBytes  7.04 Gbits/sec  1
[ 7]  0.00-10.02  sec  8.21 GBytes  7.04 Gbits/sec
                                         sender
                                         receiver
iperf Done.
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root
root@mininet-vm:/home/mininet# ■

X "host: h2@mininet-vm"
Server listening on 5201
-----
Accepted connection from 10.0.0.1, port 50866
[ 7] local 10.0.0.2 port 5201 connected to 10.0.0.1 port 50868
[ ID] Interval      Transfer     Bitrate
[ 7]  0.00-1.00   sec  1.06 GBytes  9.11 Gbits/sec
[ 7]  1.00-2.00   sec  1.05 GBytes  9.05 Gbits/sec
[ 7]  2.00-3.00   sec  818 MBytes  6.87 Gbits/sec
[ 7]  3.00-4.00   sec  769 MBytes  6.45 Gbits/sec
[ 7]  4.00-5.00   sec  824 MBytes  6.91 Gbits/sec
[ 7]  5.00-6.01   sec  782 MBytes  6.53 Gbits/sec
[ 7]  6.01-7.01   sec  662 MBytes  5.56 Gbits/sec
[ 7]  7.01-8.01   sec  758 MBytes  6.36 Gbits/sec
[ 7]  8.01-9.02   sec  840 MBytes  6.93 Gbits/sec
[ 7]  9.02-10.00  sec  778 MBytes  6.64 Gbits/sec
[ 7]  10.00-10.02  sec  9.38 MBytes  7.11 Gbits/sec
[-----]
[ ID] Interval      Transfer     Bitrate
[ 7]  0.00-10.02  sec  8.21 GBytes  7.04 Gbits/sec
                                         receiver
-----
Server listening on 5201
-----
^Ciperf3: interrupt - the server has terminated
root@mininet-vm:/home/mininet# ■

```

Рис. 4.6: Ограничение скорости на конечных хостах и тест

4.2.2 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. Примените 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: `sudo tc qdisc add dev s1-eth2 root tbf rate 10gbit burst 5000000 limit 15000000.`
2. Проверьте конфигурацию с помощью инструмента iperf3 для измерения пропускной способности(4.7)
 - В терминале хоста h2 запустите iPerf3 в режиме сервера: `iperf3 -s`
 - В терминале хоста h2 запустите iPerf3 в режиме клиента: `iperf3 -c 10.0.0.2`
 - После завершения работы iPerf3 на хосте h1 остановите iPerf3 на хосте h2, нажав **Ctrl + c**.

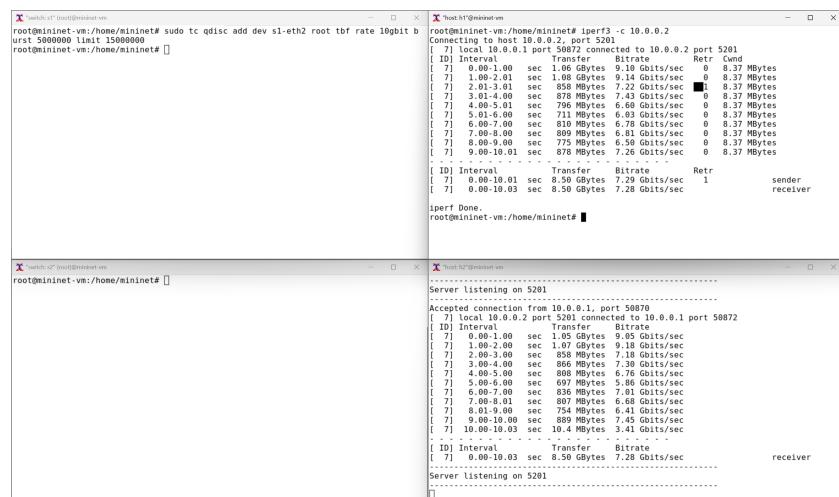
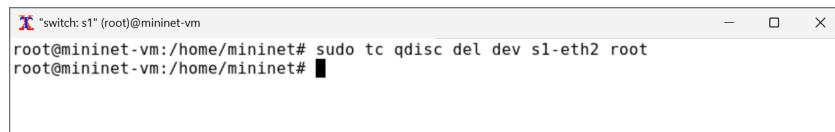


Рис. 4.7: Ограничение скорости на коммутаторе и тест

Скорость передачи в пределах 6-9 Gbits/sec что соответствует правилу.

3. Удалите модифицированную конфигурацию на коммутаторе s1: `sudo tc qdisc del dev s1-eth2 root` (рис. 4.8)



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

Рис. 4.8: Удаление модифицированной конфигурации на коммутаторе s1

4.2.3 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. Объедините NETEM и TBF, введя на интерфейсе `s1-eth2` коммутатора `s1` задержку, джиттер, повреждение пакетов и указав скорость: `sudo tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms`.

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

2. Убедитесь, что соединение от хоста `h1` к хосту `h2` имеет заданную задержку. Для этого запустите команду `ping` с параметром `-c 4` с терминала хоста `h1` (рис. 4.9)

The screenshot shows two terminal windows. The top window, titled "switch: s1" (root)@mininet-vm, runs the command `sudo tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms`. The bottom window, titled "host: h1" (root)@mininet-vm, runs `ping -c 4 10.0.0.2` and displays the output of four ping packets to host 10.0.0.2, showing times around 11-14 ms and a round-trip time of 3007 ms.

```

switch: s1" (root)@mininet-vm
root@mininet-vm:/home/mininet# sudo tc qdisc add dev s1-eth2 root handle 1: netem d
elay 10ms
root@mininet-vm:/home/mininet# 

host: h1" @mininet-vm
root@mininet-vm:/home/mininet# ping -c 4 10.0.0.2
ping: invalid argument: '10.0.0.2'
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=14.7 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=11.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=11.7 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=11.2 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 11.183/12.296/14.686/1.393 ms
root@mininet-vm:/home/mininet#

```

Рис. 4.9: Добавление NETEM правила задержки пакетов

Как мы видим что задержка от 10ms что вызвало наше правило

3. Добавьте второе правило на коммутаторе s1, которое задаёт ограничение скорости с помощью tbf с параметрами rate=2gbit, burst=1,000,000, limit=2,000,000: `sudo tc qdisc add dev s1-eth2 parent 1: handle 2: tbf rate 2gbit burst 1000000 limit 2000000`.
4. Проверьте конфигурацию с помощью инструмента iperf3 для измерения пропускной способности(рис. 4.10)
 - В терминале хоста h2 запустите iPerf3 в режиме сервера: `iperf3 -s`
 - В терминале хоста h2 запустите iPerf3 в режиме клиента: `iperf3 -c 10.0.0.2`
 - После завершения работы iPerf3 на хосте h1 остановите iPerf3 на хосте h2, нажав Ctrl + c. В отчёте зафиксируйте результат работы iPerf3 на данном этапе проведения эксперимента.

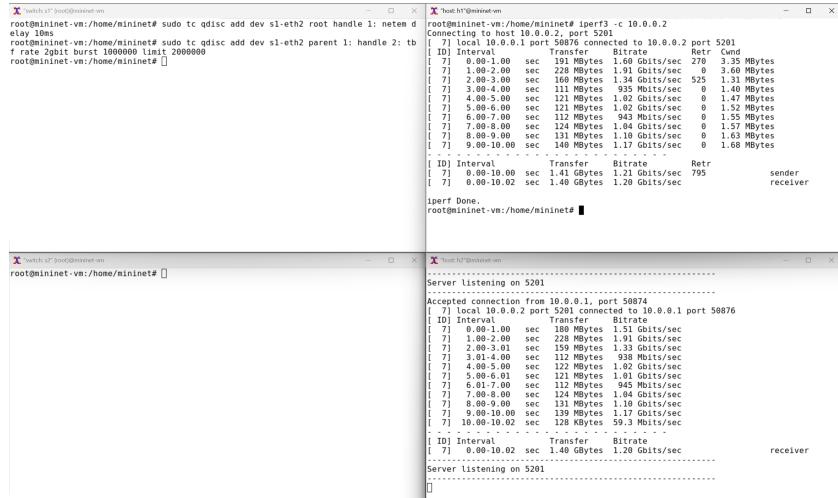


Рис. 4.10: Добавление 2ого правила ограничения скорости и тест

получился битрейт 0.9 - 1.9 Gbits/sec что также соответствует заданным правилом ограничения скорости.

- Удалите модифицированную конфигурацию на коммутаторе s1: sudo tc qdisc del dev s1-eth2 root (рис. 4.11)

The terminal window shows the command to delete the tc configuration on interface s1-eth2:

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

Рис. 4.11: Удаление модифицированной конфигурации на коммутаторе s1

4.3 Воспроизведение экспериментов

- Для воспроизводимого эксперимента создам каталог ~ /work / lab _ tbf _ i , в нём создам файлы для воспроизводимого эксперимента (рис. 4.12).
- Далее перенёс полученную структуру по папкам ~ /work / lab _ tbf _ i / exp1 , ~ /work / lab _ tbf _ i / exp2 , ~ /work / lab _ tbf _ i / exp3 соответствующим нашим экспериментам.

```

mininet@mininet-vm: ~/work/lab_tbf_i
mininet@mininet-vm: $ cd work/
mininet@mininet-vm: ~/work$ ls
lab6  lab_iperf3  lab_netem_i  lab_netem_ii  lesson1.mn
mininet@mininet-vm: ~/work$ rm -rf lab6
mininet@mininet-vm: ~/work$ mkdir -p lab_tbf_i
mininet@mininet-vm: ~/work$ cd lab_tbf_i
mininet@mininet-vm: ~/work/lab_tbf_i$ touch lab_tbf_i.py
mininet@mininet-vm: ~/work/lab_tbf_i$ touch Makefile
mininet@mininet-vm: ~/work/lab_tbf_i$ touch ping_plot
mininet@mininet-vm: ~/work/lab_tbf_i$ 

```

Рис. 4.12: Создание необходимых файлов

Для начала воспроизведём эксперимент “Ограничение скорости на конечных хоста”. Создадим и заполним файл `exp1/lab_tbf_i.py`(рис. 4.13), заполним файлы `Makefile` для выполнения эксперимента и `ping_plot` для построения графиков (рис. 4.14)

```

mininet@mininet-vm: ~/work/lab_tbf_i/exp1
GNU nano 4.8                                     lab_tbf_i.py
#!/usr/bin/env python

"""
Simple experiment.
Output: ping.dat
"""

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

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')
    s2 = net.addSwitch('s2')
    info('*** Creating links\n')
    net.addLink(h1, s1)
    net.addLink(h2, s2)
    net.addLink(s2, s1)
    net.start()
    info('*** Set rate\n')
    h1.cmdPrint('tc qdisc add dev h1-eth0 root tbf rate 10gbps burst 5000000 limit 15000000')
    info('*** Start iperf server on h2\n')
    h2.cmdPrint('iperf3 -s &')
    time.sleep(10)
    info('*** Run iperf3 client from h1 to h2\n')
    h1.cmdPrint('iperf3 -c', h2.IP(), '-J > iperf_result.json')
    h1.cmdPrint('iperf3 -c', h2.IP(), '| grep "MBytes" | awk \'{print \$7}\' > iperf3.dat')
    info('*** Run ping from h1 to h2\n')
    h1.cmdPrint('ping -c 15', h2.IP(), '| grep "time=" | awk \'{print \$5, \$7}\' | sed -e \'s/time=/g\'')
    info('*** Stopping network')
    net.stop()

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

```

Рис. 4.13: Листинг `exp1/lab_tbf_i.py`

```

mininet@mininet-vm:~/work/lab_tbf_i/exp1$ cat Makefile
all: ping.dat ping.png

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

ping.png:
    ./ping_plot

clean:
    -rm -f *.dat *.png
mininet@mininet-vm:~/work/lab_tbf_i/exp1$ cat ping_plot
#!/usr/bin/gnuplot --persist

set terminal png crop
set output 'ping.png'
set xlabel "Sequence number"
set ylabel "Delay (ms)"
set grid
plot "ping.dat" with lines

set terminal png crop
set output 'iperf3.png'
set xlabel "Packet number"
set ylabel "Rate (Gbytes/sec)"
set grid
plot "iperf3.dat" with lines

```

Рис. 4.14: Листинг Makefile и ping_plot

Запустим эксперемент exp1 командой make(рис. 4.15)

```

mininet@mininet-vm:~/work/lab_tbf_i/exp1$ make
sudo python lab_tbf_i.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Waiting for switches to connect
s1 s2
*** Set rate
*** h1 : ('tc qdisc add dev h1-eth0 root tbft rate 10gburst 5000000 limit 15000000',)
*** Start iperf server on h2
*** h2 : ('iperf3 -s &')
*** Run iperf3 client from h1 to h2*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
*** h1 : ('iperf3 -c', '10.0.0.2', '| grep "MBytes" | awk \'(print $7)\' > iperf3.dat')
*** Run ping from h1 to h2
*** h1 : ('ping -c 15', '10.0.0.2', '| grep "time=" | awk \'(print $5, $7)\' | sed -e \'s/time=/g\' -e \'s/icmp_seq=/g\' > ping.dat')
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
./ping_plot
mininet@mininet-vm:~/work/lab_tbf_i/exp1$ 

```

Рис. 4.15: Запуск exp1

Просмотрим полученные графики iperf3.png (рис. 4.16) и ping.png(рис. 4.17)

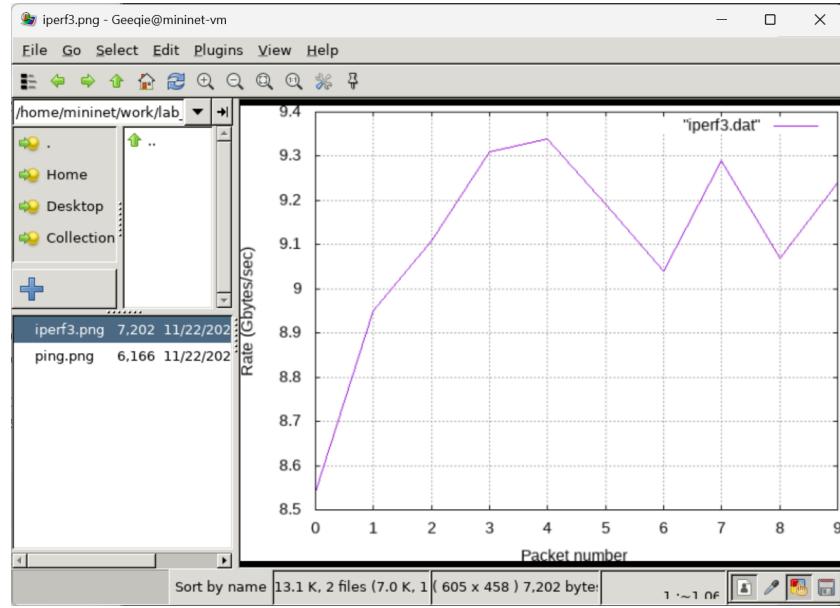


Рис. 4.16: График iperf3.png из exp1

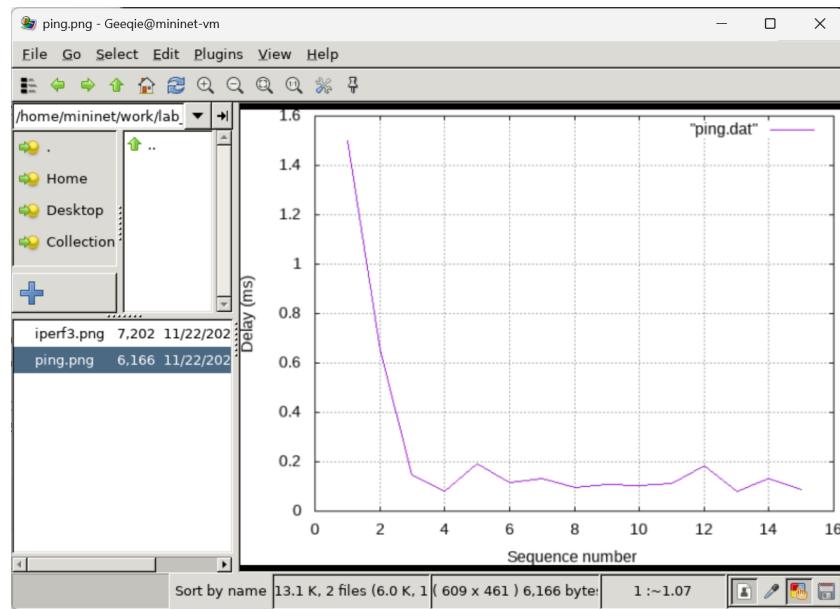


Рис. 4.17: График ping.png из exp1

По аналогии отредактируем файл `exp2/lab_tbf_i.py` для эксперимента "Ограничение скорости на коммутаторах" (рис. 4.18). Остальные файлы оставим такими же как в первом эксперименте.

```
GNU nano 4.8          lab_tbf_i.py          Modified
#!/usr/bin/env python

"""
Simple experiment.
Output: ping.dat
"""

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

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')
    s2 = net.addSwitch('s2')
    info('*** Creating links\n')
    net.addLink(h1, s1)
    net.addLink(h2, s2)
    net.addLink(s2, s1)
    info('*** Starting network\n')
    net.start()

    info('*** Set rate\n')
    s1.cmdPrint('tc qdisc add dev s1-eth2 root tbf rate 10gbit burst 5000000 lim 15000000')
    info('*** Start iperf server on h2\n')
    h2.cmdPrint('iperf3 -s &')
    time.sleep(10)

    info('*** Run iperf3 client from h1 to h2')
    h1.cmdPrint('iperf3 -c', h2.IP(), '-J > iperf_result.json')
    h1.cmdPrint('iperf3 -c', h2.IP(), '| grep "MBytes" | awk \'{print $5, $7}\' > iperf3.dat')

    info('*** Run ping from h1 to h2')
    h1.cmdPrint('ping -c 15', h2.IP(), '| grep "time=" | awk \'{print $5, $7}\' | sed -e \'s/time=/g\' > ping.dat')

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

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

Рис. 4.18: Листинг exp2/lab_tbf_i.py

Запустим эксперемент exp2 командой make(рис. 4.19)

```

mininet@mininet-vm:~/work/lab_tbf_i/exp2$ make
sudo python lab_tbf_i.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Waiting for switches to connect
s1 s2
*** Set rate
*** s1 : ('tc qdisc add dev s1-eth2 root tbf rate 10gbit burst 500000 limit 15000000',)
*** Start iperf server on h2
*** h2 : ('iperf3 -s &')
*** Run iperf3 client from h1 to h2*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
*** h1 : ('iperf3 -c', '10.0.0.2', '| grep "MBytes" | awk \'(print $7)\' > iperf3.dat')
*** Run ping from h1 to h2
*** h1 : ('ping -c 15', '10.0.0.2', '| grep "time=" | awk \'(print $5, $7)\' | sed -e \'s/time=/g\' -e \'s/icmp_seq=/g\' > ping.dat')
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
./ping_plot
mininet@mininet-vm:~/work/lab_tbf_i/exp2$ ls
iperf3.dat iperf3.png iperf_result.json lab_tbf_i.py Makefile ping.dat ping_plot ping.png
mininet@mininet-vm:~/work/lab_tbf_i/exp2$ █

```

Рис. 4.19: Запуск exp2

Просмотрим полученные графики iperf3.png (рис. 4.20) и ping.png(рис. 4.21)

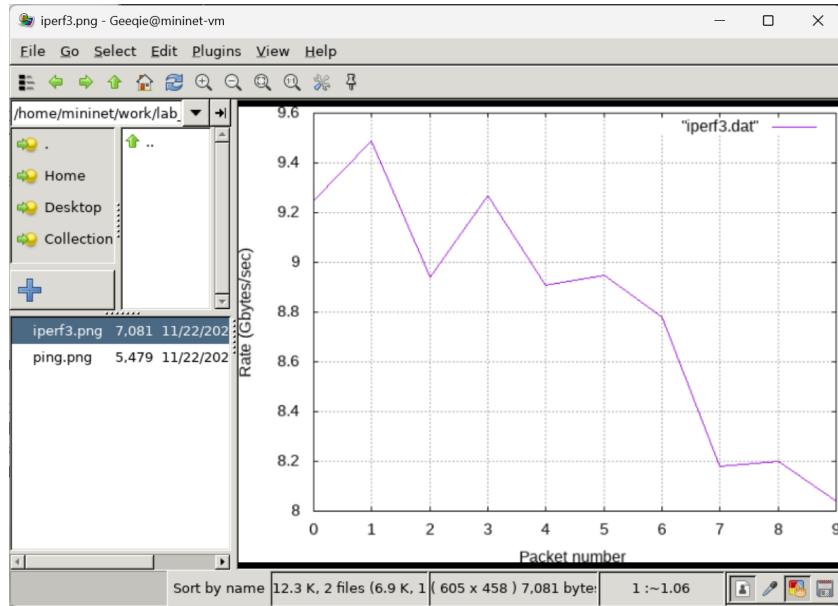


Рис. 4.20: График iperf3.png из exp2

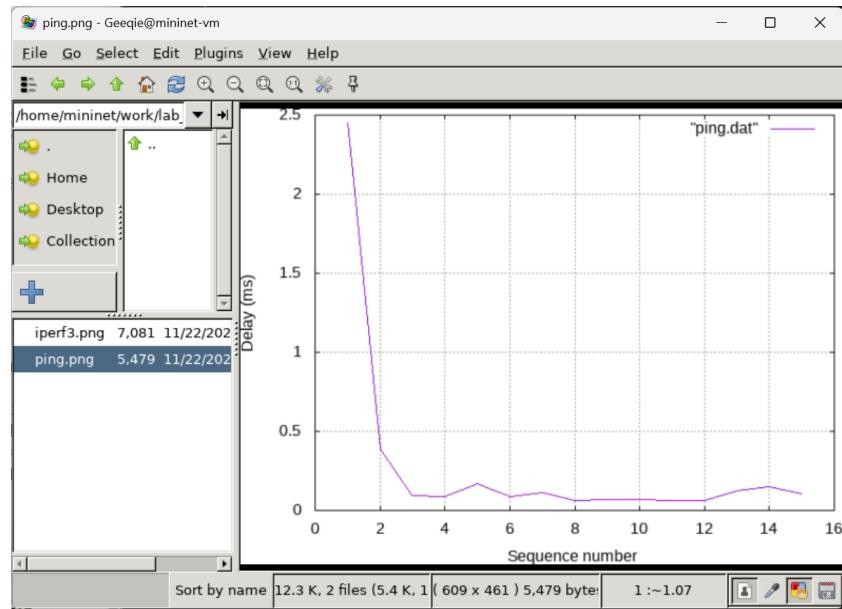
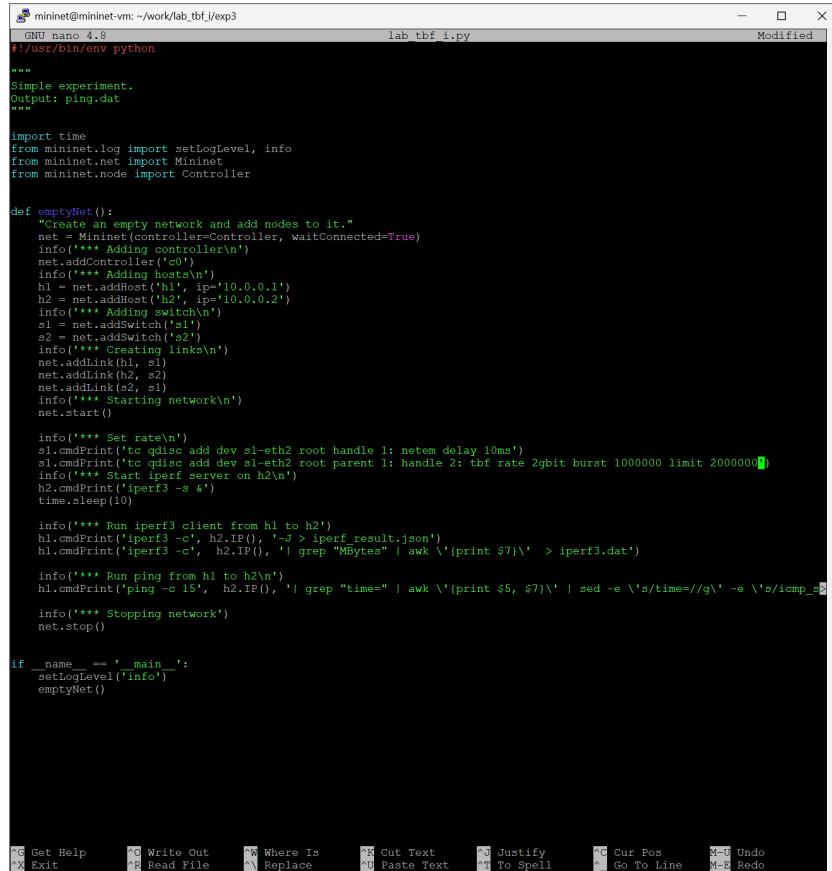


Рис. 4.21: График ping.png из exp2

Таким же образом проведём эксперимент “Объединение NETEM и TBF”. Отредактируем `exp3/lab_tbf_i.py`(рис. 4.22)



```
GNU nano 4.8          lab_tbf_i.py
#!/usr/bin/env python
"""

Simple experiment.
Output: ping.dat
"""

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

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')
    s2 = net.addSwitch('s2')
    info('*** Creating links\n')
    net.addLink(h1, s1)
    net.addLink(h2, s2)
    net.addLink(s2, s1)
    info('*** Starting network\n')
    net.start()

    info('*** Set rate\n')
    s1.cmdPrint('tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms')
    s1.cmdPrint('tc qdisc add dev s1-eth2 root parent 1: tbf rate 2gbit burst 1000000 limit 2000000')
    info('*** Start iperf server on h2\n')
    h2.cmdPrint('iperf3 -s &')
    time.sleep(10)

    info('*** Run iperf3 client from h1 to h2')
    h1.cmdPrint('iperf3 -c', h2.IP(), '-J > iperf_result.json')
    h1.cmdPrint('iperf3 -c', h2.IP(), '| grep "MBytes" | awk \'(print $7)\' > iperf3.dat')

    info('*** Run ping from h1 to h2')
    h1.cmdPrint('ping -c 15', h2.IP(), '| grep "time=' | awk \'(print $5, $7)\' | sed -e \'s/time=/g\' -e \'s/icmp_s/latency\'')

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

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

Рис. 4.22: Листинг exp3/lab_tbf_i.py

Запустим эксперемент exp3 командой make(рис. 4.23)

```
mininet@mininet-vm:~/work/lab_tbf_i/exp3$ nano lab_tbf_i.py
mininet@mininet-vm:~/work/lab_tbf_i/exp3$ make
sudo python lab_tbf_i.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Waiting for switches to connect
s1 s2
*** Set rate
*** s1 : ('tc qdisc add dev sl-eth2 root handle 1: netem delay 10ms',)
*** s1 : ('tc qdisc add dev sl-eth2 root parent 1: handle 2: tbf rate 2gbit burst 1000000 limit 2000000',)
Error: duplicate "parent": "1:" is the second value.
*** Start iperf server on h2
*** h2 : ('iperf3 -s &',)
*** Run iperf3 client from h1 to h2*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
*** h1 : ('iperf3 -c', '10.0.0.2', '| grep "MBytes" | awk \'(print $7)\' > iperf3.dat')
*** Run ping from h1 to h2
*** h1 : ('ping -c 15', '10.0.0.2', '| grep "time=" | awk \'(print $5, $7)\' | sed -e \'s/time=/g\' -e \'s/icmp_seq=/g\' > ping.dat')
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
./ping_plot
mininet@mininet-vm:~/work/lab_tbf_i/exp3$ ls
iperf3.dat  iperf3.png  iperf_result.json  lab_tbf_i.py  Makefile  ping.dat  ping_plot  ping.png
mininet@mininet-vm:~/work/lab_tbf_i/exp3$
```

Рис. 4.23: Запуск exp3

Просмотрим полученные графики iperf3.png (рис. 4.24) и ping.png(рис. 4.25)

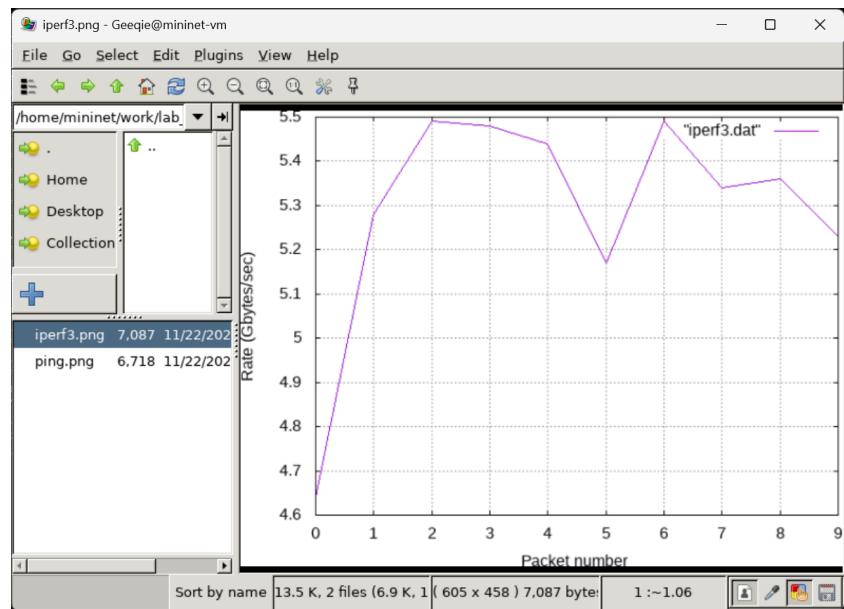


Рис. 4.24: График iperf3.png из exp3

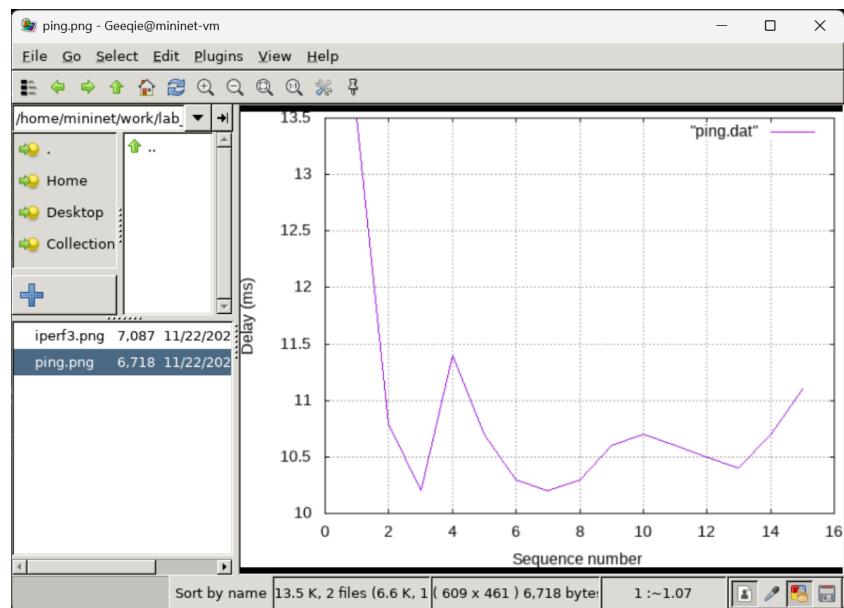


Рис. 4.25: График ping.png из exp1

5 Выводы

В результате выполнения работы я познакомился с принципами работы Token Bucket Filter, а также получили навыки моделирования и исследования поведения трафика посредством проведения интерактивного и воспроизводимого экспериментов в Mininet.

Список литературы

1. Mininet [Электронный ресурс]. Mininet Project Contributors. URL: <http://mininet.org/> (дата обращения: 07.10.2025).