# Отчёт по лабораторной работе №6
# Моделирование сетей передачи данных

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

Выполнил: Исаев Булат Абубакарович,
НПИбд-01-22, 1132227131

## Содержание

1 Цель работы ..... 1
2 Выполнение лабораторной работы ..... 1
3 Вывод ..... 24
4 Список литературы. Библиография ..... 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 Выполнение лабораторной работы

В виртуальной машине mininet исправим права запуска X-соединения (рис. 1):

```
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  bced047e0c3bcbb01e84e95f729f6f92
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  bced047e0c3bcbb01e84e95f729f6f9
2
root@mininet-vm:~# logout
mininet@mininet-vm:~$
```

*Рис. 1: Исправление прав запуска X-соединения в виртуальной машине mininet*

Зададим топологию сети, состоящую из двух хостов и двух коммутаторов с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 2):

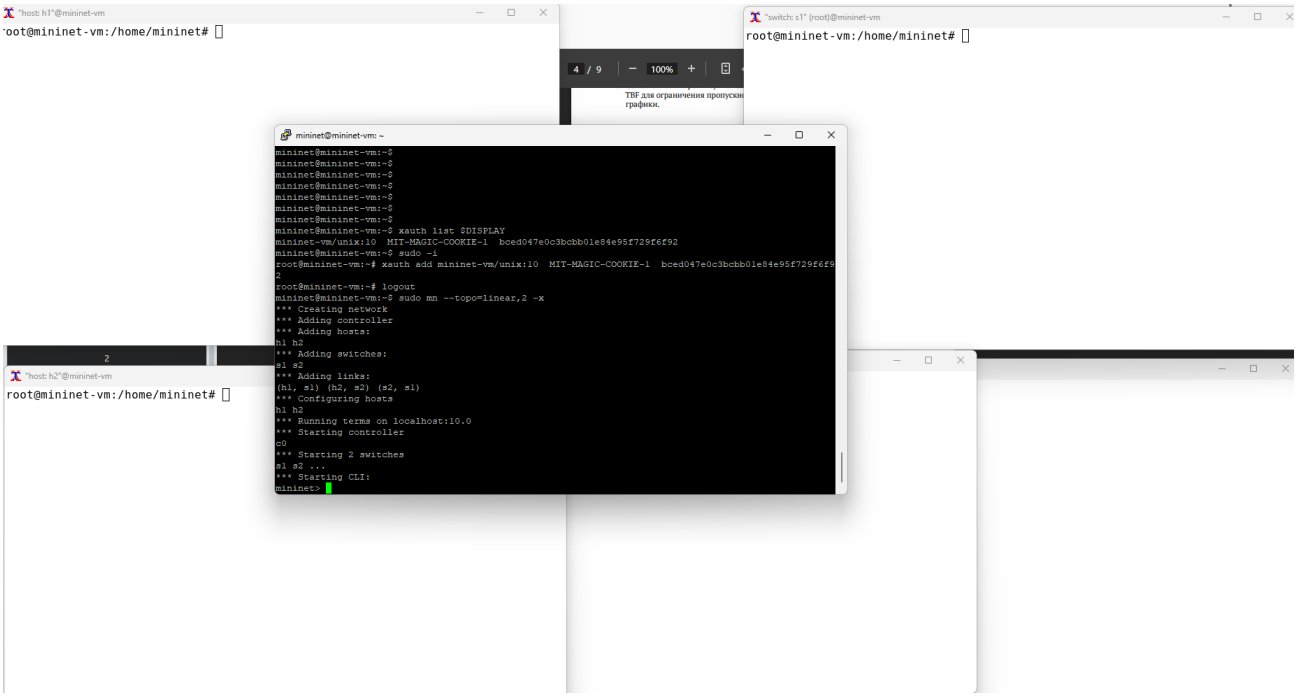


*Рис. 2: Создание топологии с двумя хостами и двумя коммутаторами*

На хостах h1, h2 и на коммутаторах s1, s2 введём команду ifconfig, чтобы отобразить информацию, относящуюся к их сетевым интерфейсам и назначенным им IP-адресам. В дальнейшем при работе с NETEM и командой tc будем использовать интерфейсы h1-eth0, h2-eth0, s1-eth2 (рис. 3):

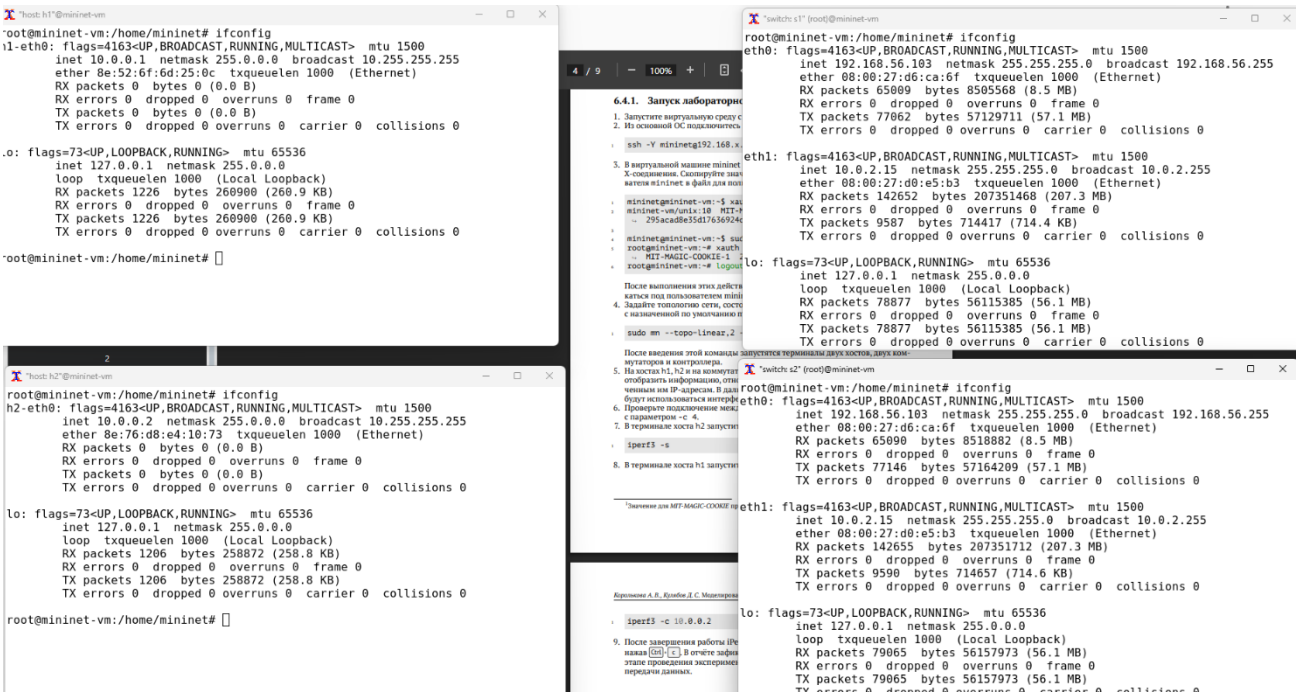


*Рис. 3: Отображение информации сетевых интерфейсов и IP-адресов*

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

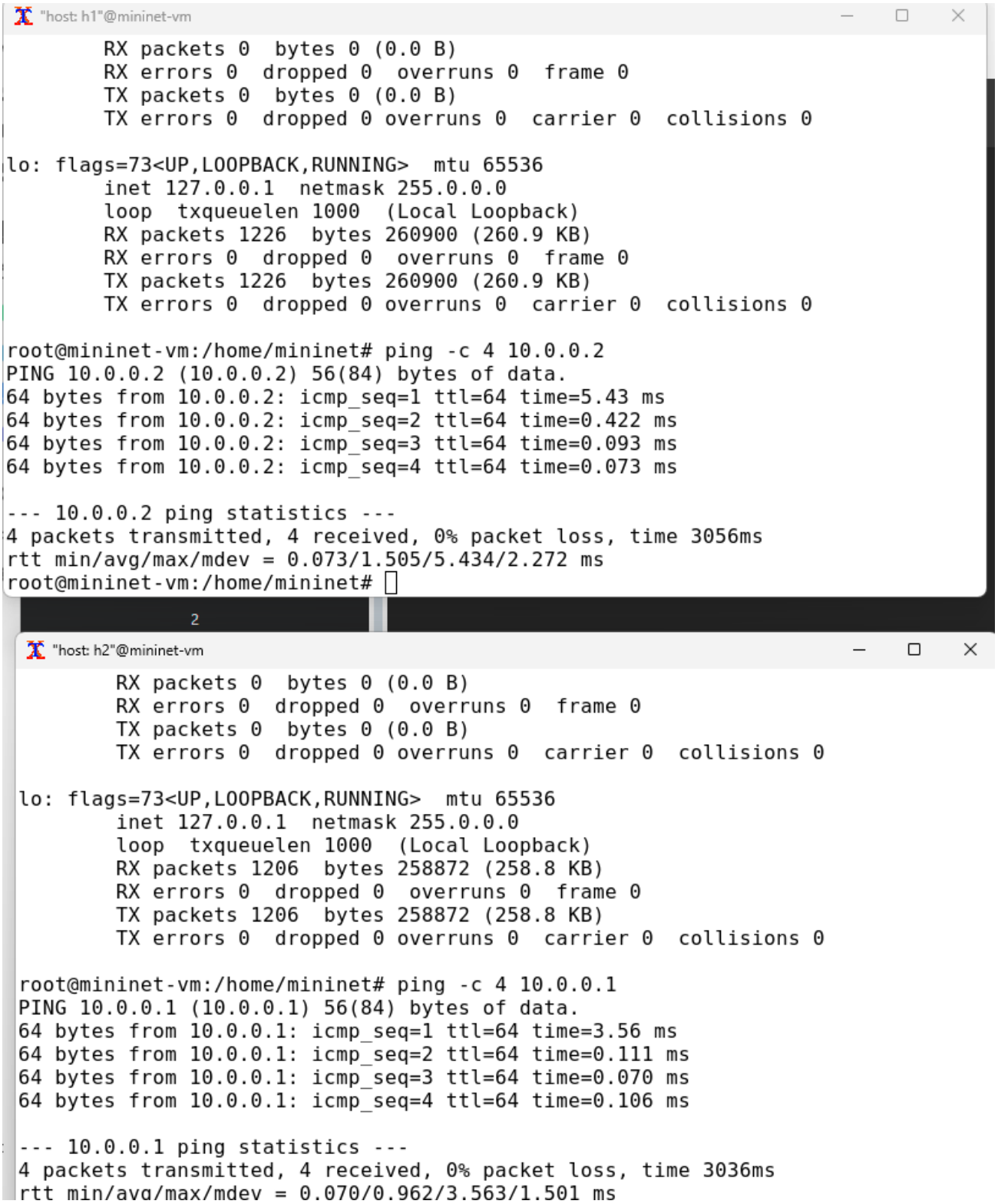


*Рис. 4: Проверка подключения между хостами h1 и h2*

В терминале хоста h2 запустим iPerf3 в режиме сервера (рис. 5):

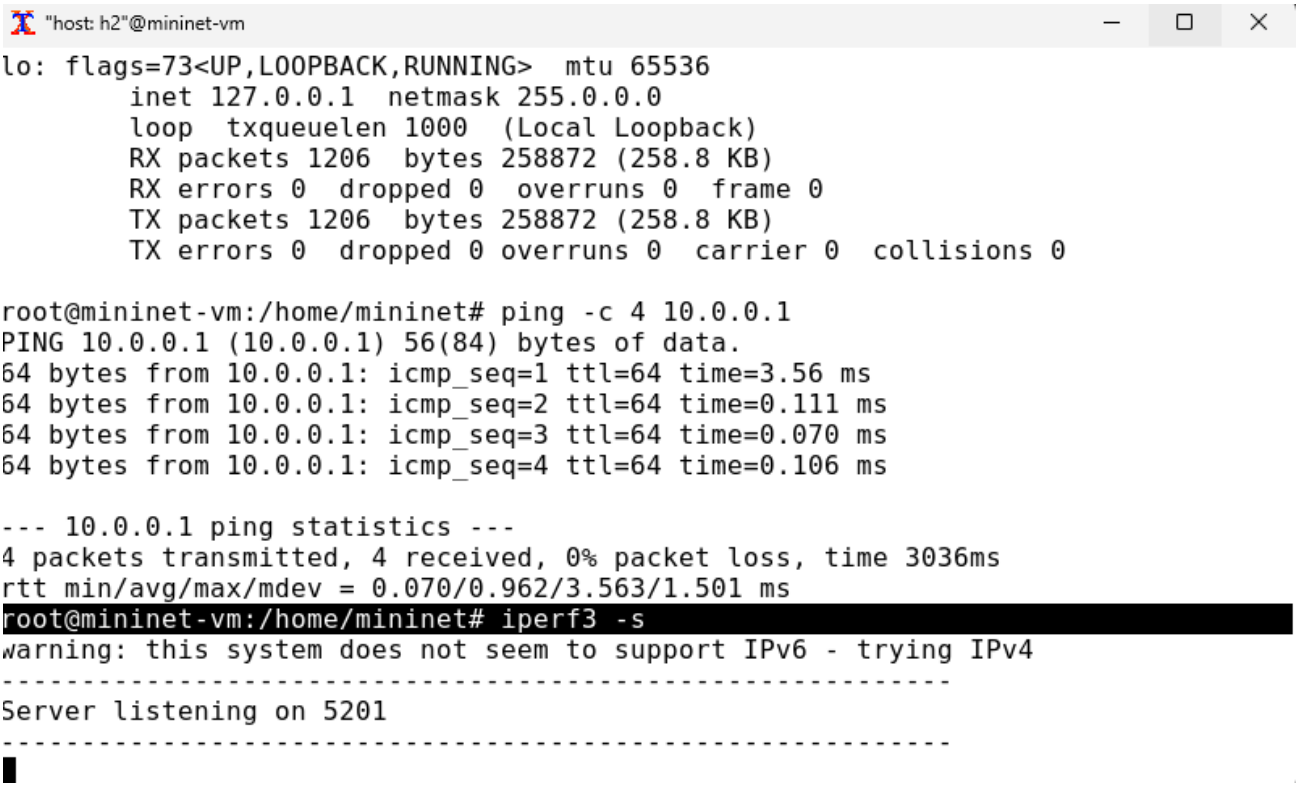


*Рис. 5: Запуск iPerf3 в режиме сервера на хосте h2*

В терминале хоста h1 запустим iPerf3 в режиме клиента (рис. 6):

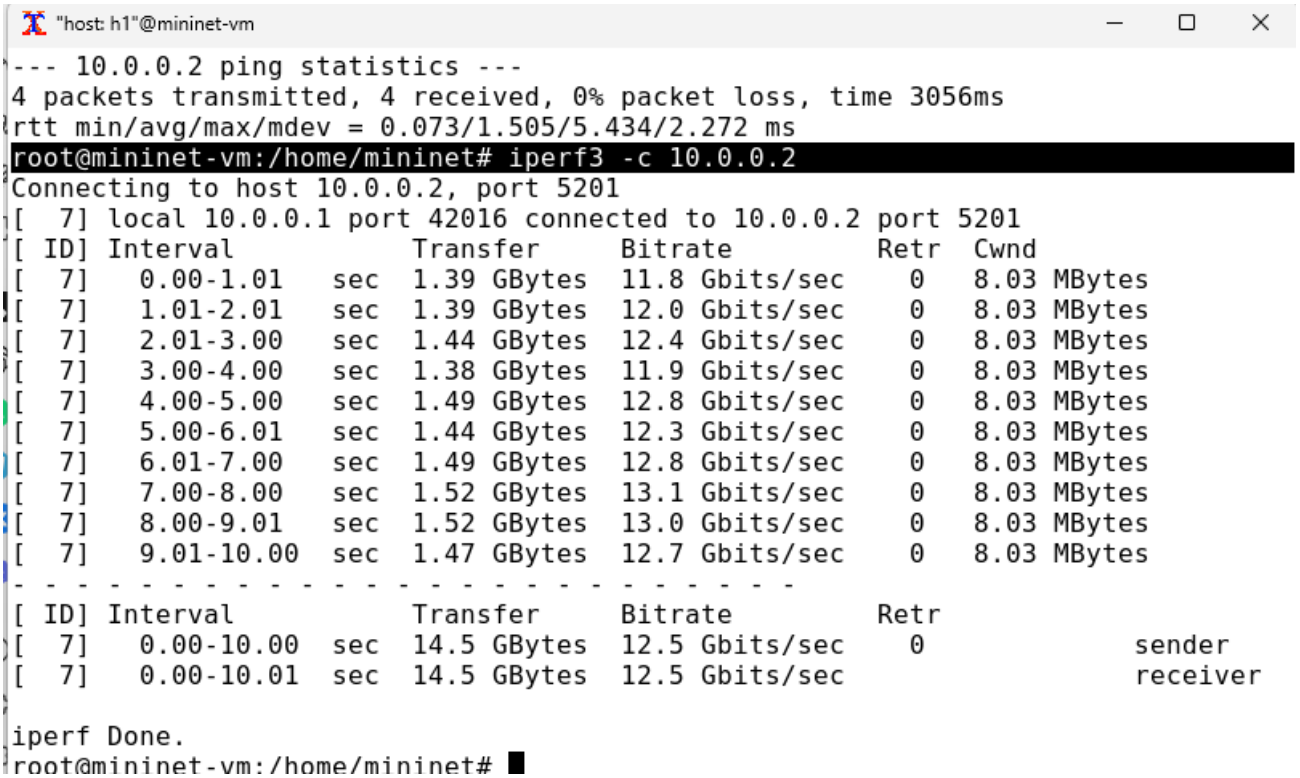


*Рис. 6: Запуск iPerf3 в режиме клиента на хосте h1*

После завершения работы iPerf3 на хосте h1 остановим iPerf3 на хосте h2, нажав Ctrl + c (рис. 7):

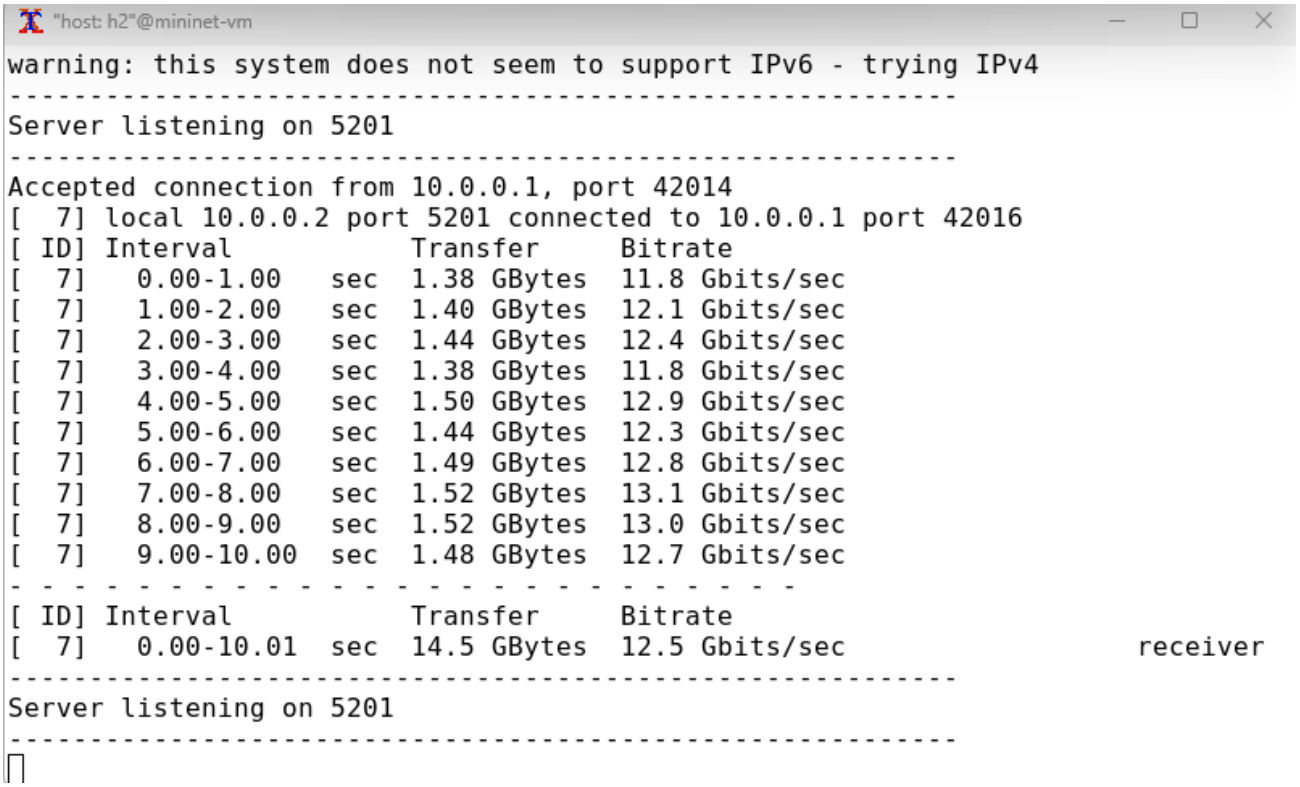


*Рис. 7: Остановка iPerf3*

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).

Изменим пропускную способность хоста h1, установив пропускную способность на 10 Гбит/с на интерфейсе h1-eth0 и параметры TBF-фильтра (рис. 8):

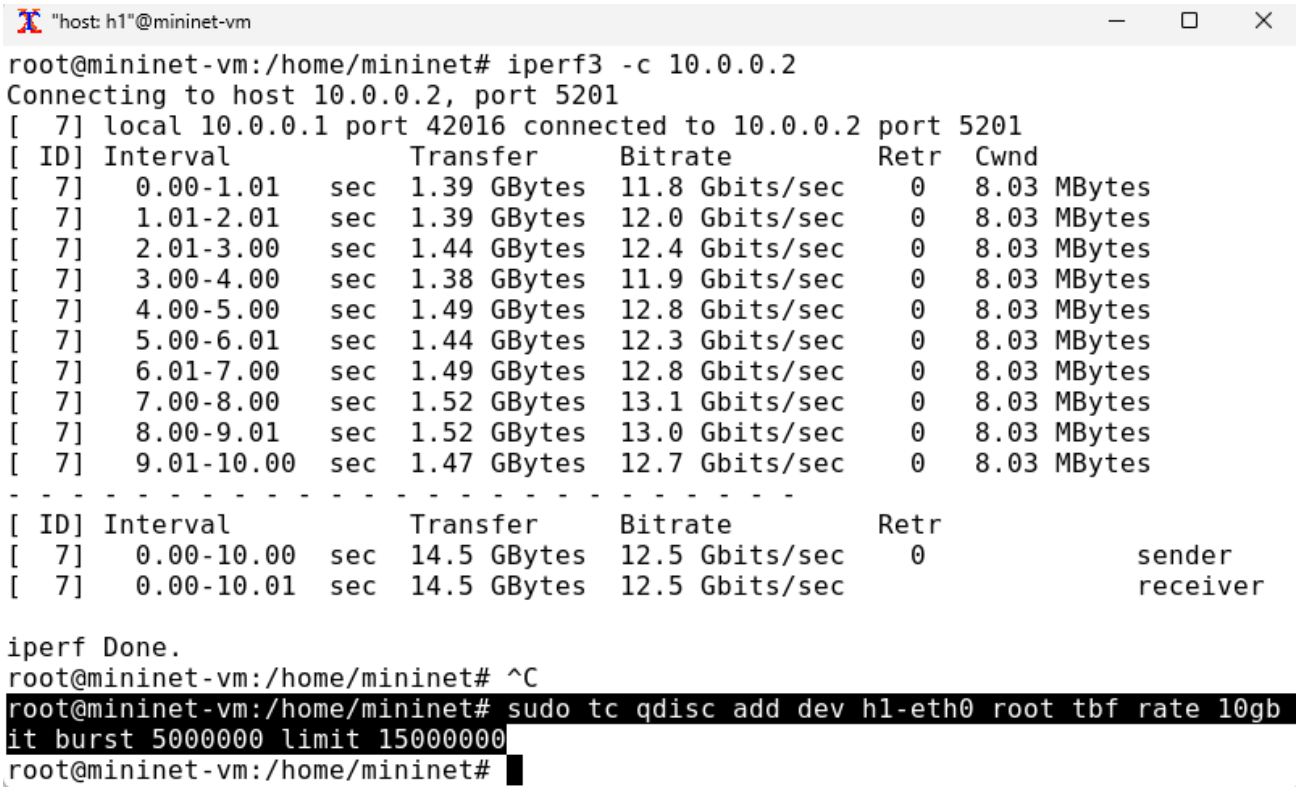


*Рис. 8: Изменение пропускной способности хоста h1*

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 (рис. 9):

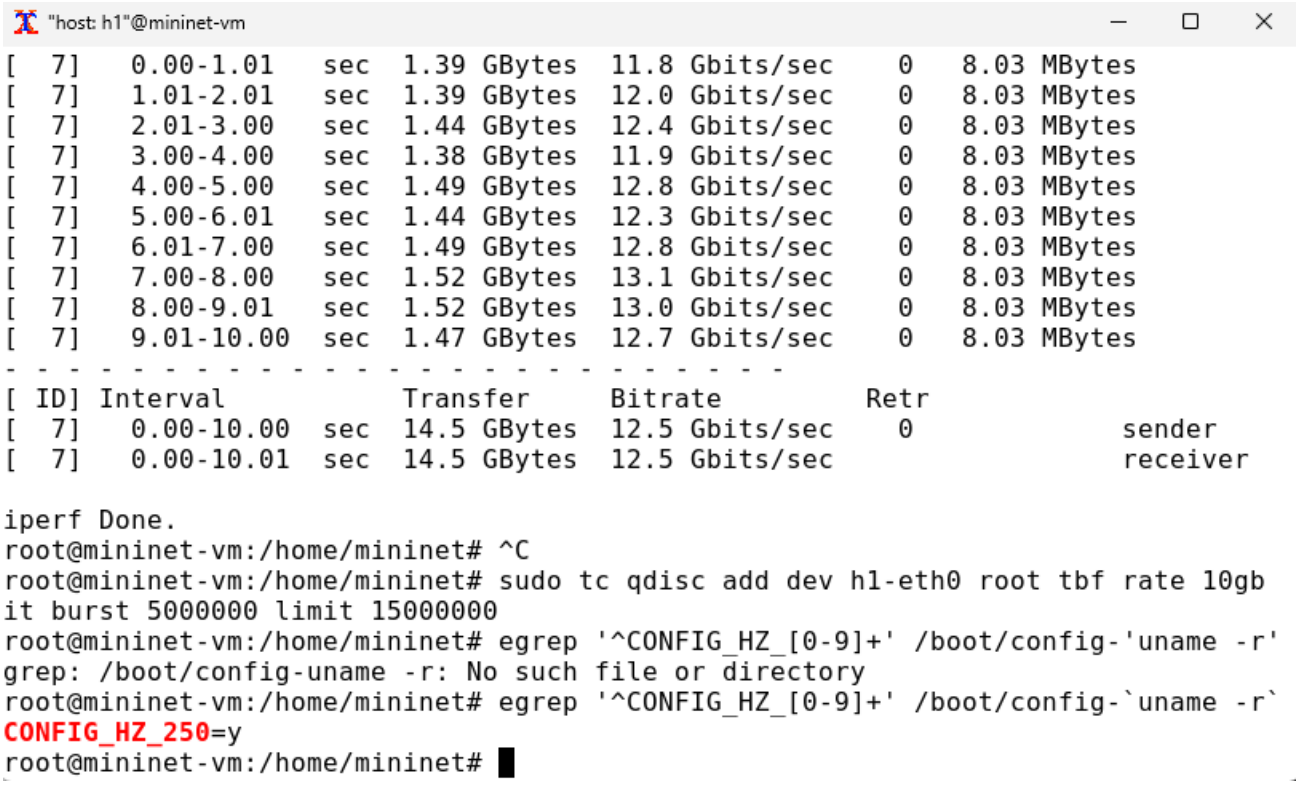


*Рис. 9: Установка значения всплеска при ограничении скорости для фильтра tbf*

С помощью iPerf3 проверим, что значение пропускной способности изменилось.

В терминале хоста h2 запустим iPerf3 в режиме сервера (рис. 10):

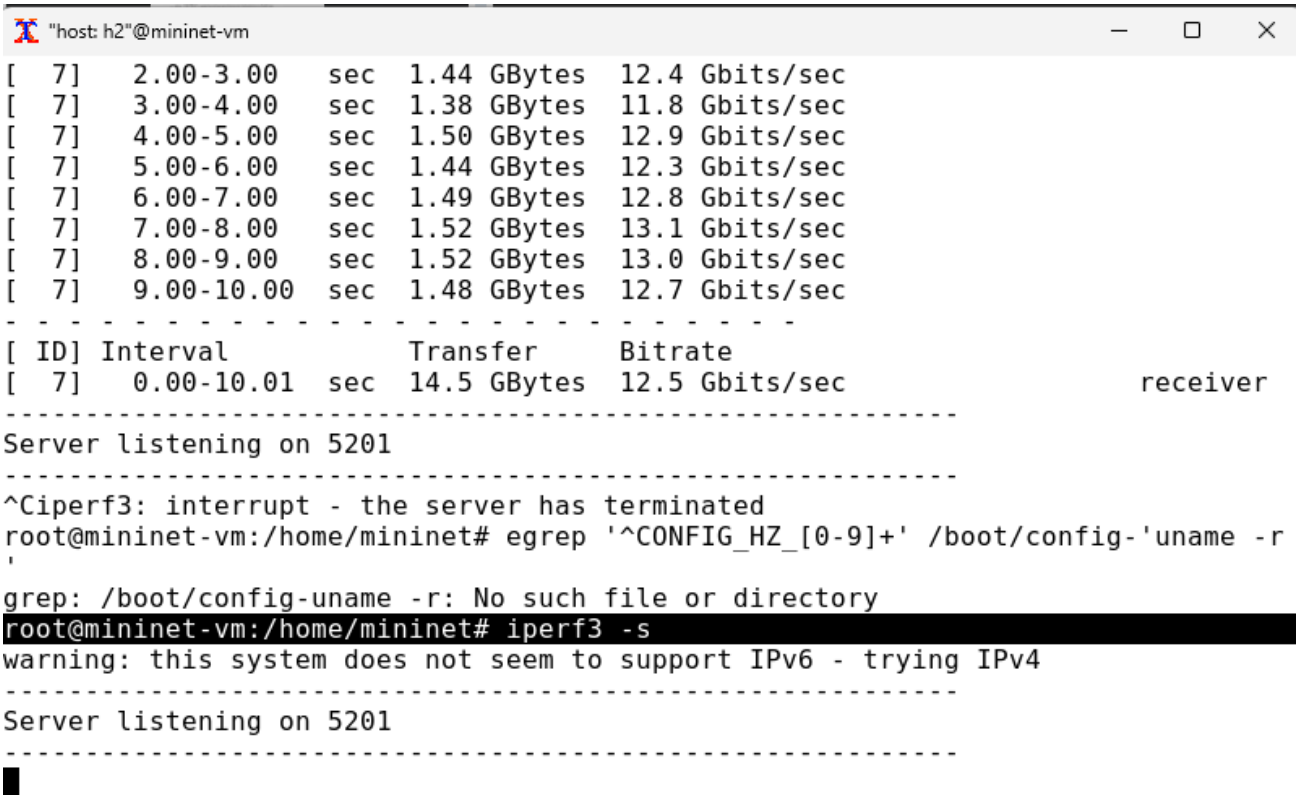


*Рис. 10: Запуск iPerf3 в режиме сервера на хосте h2*

В терминале хоста h1 запустим iPerf3 в режиме клиента (рис. 11):

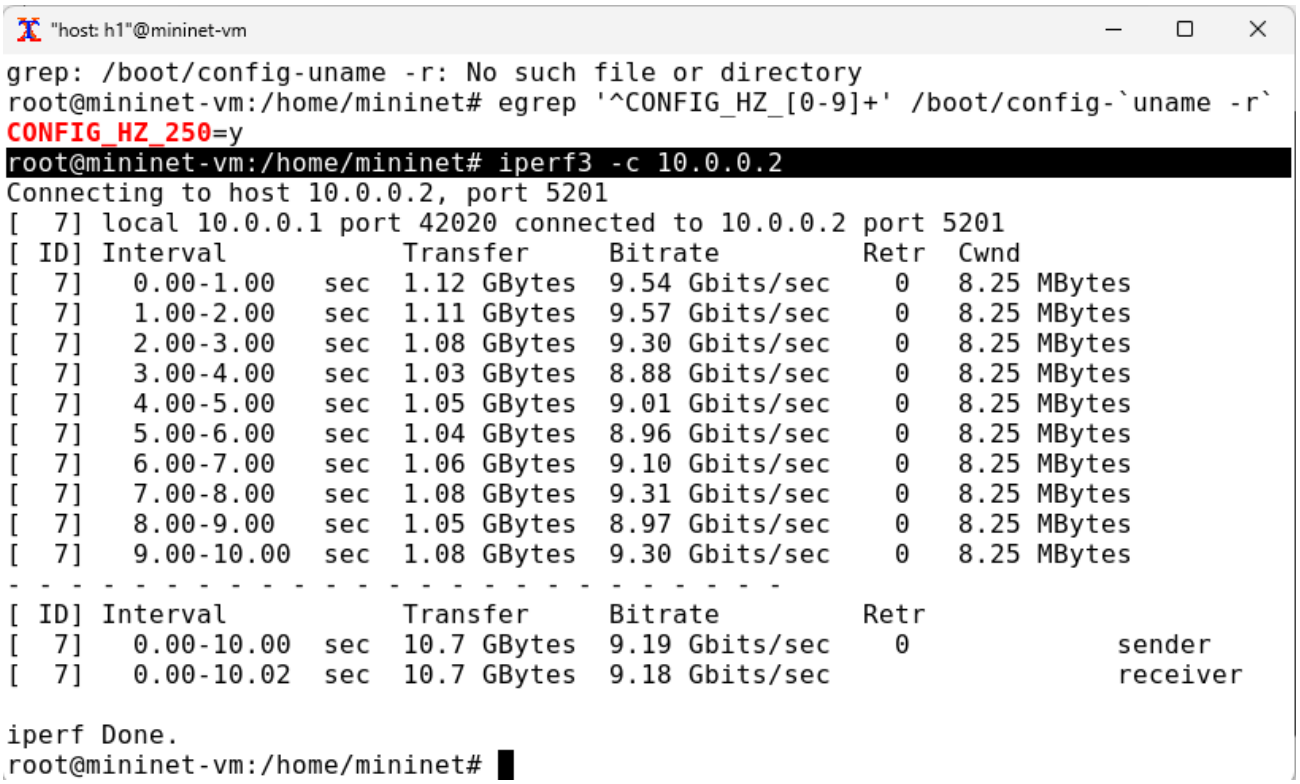


*Рис. 11: Запуск iPerf3 в режиме клиента на хосте h1*

После завершения работы iPerf3 на хосте h1 остановим iPerf3 на хосте h2, нажав Ctrl + c (рис. 12):

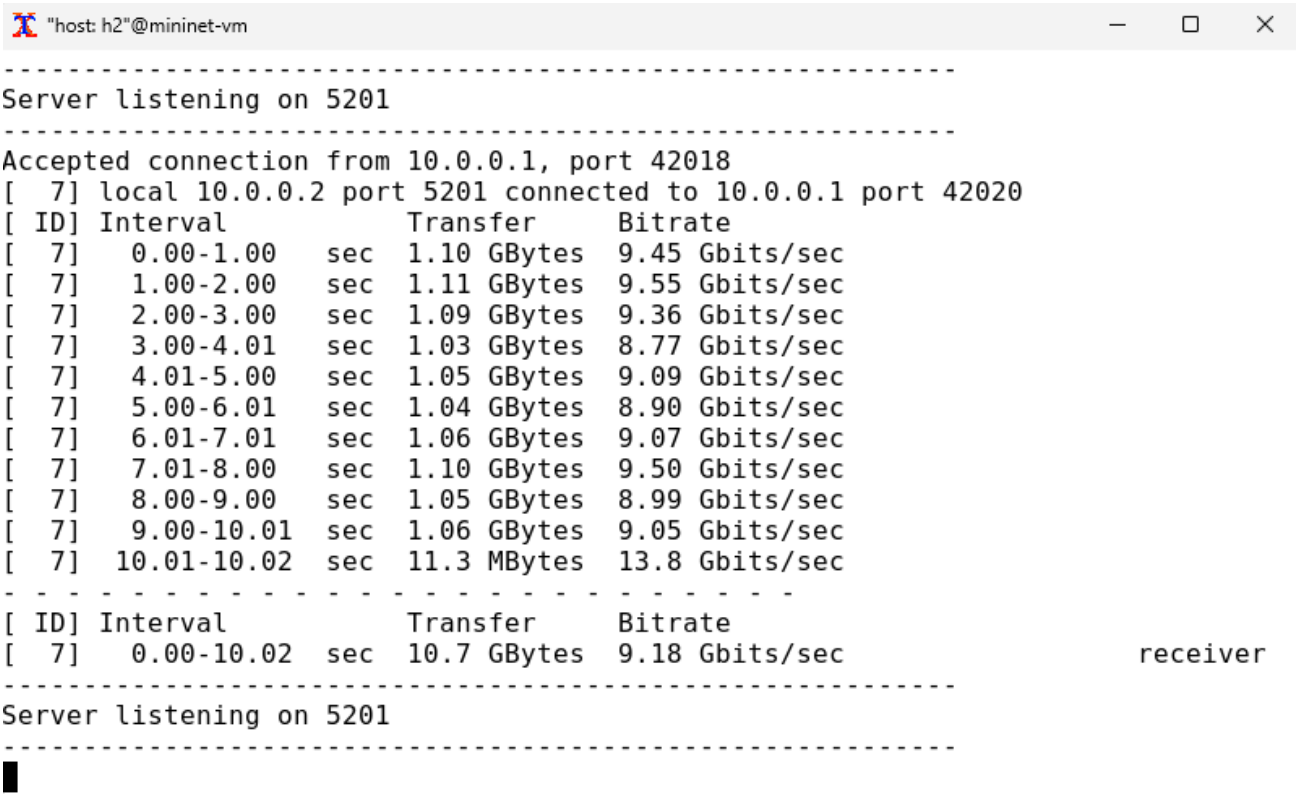


*Рис. 12: Остановка iPerf3*

Удалим модифицированную конфигурацию на хосте h1 (рис. 13):

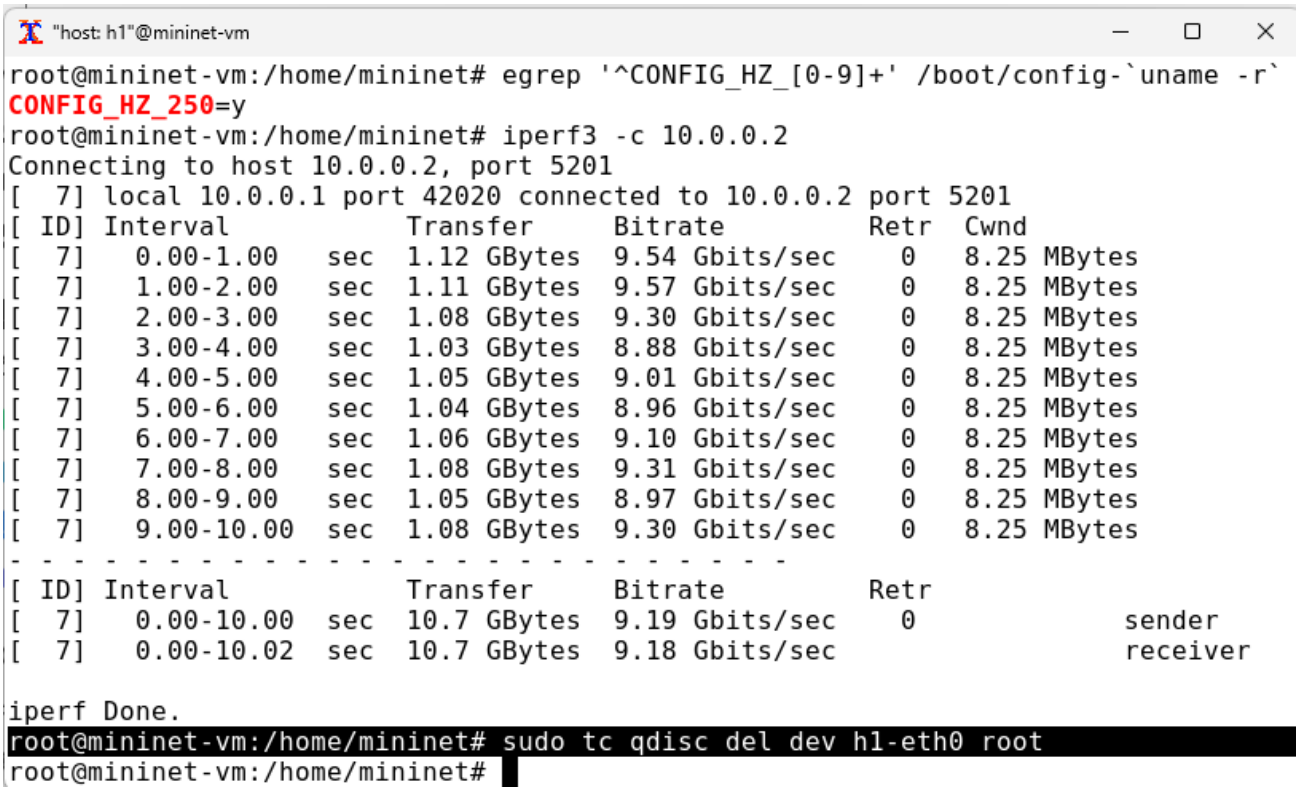


*Рис. 13: Удаление модифицированной конфигурации на хосте h1*

Применим 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. 14):

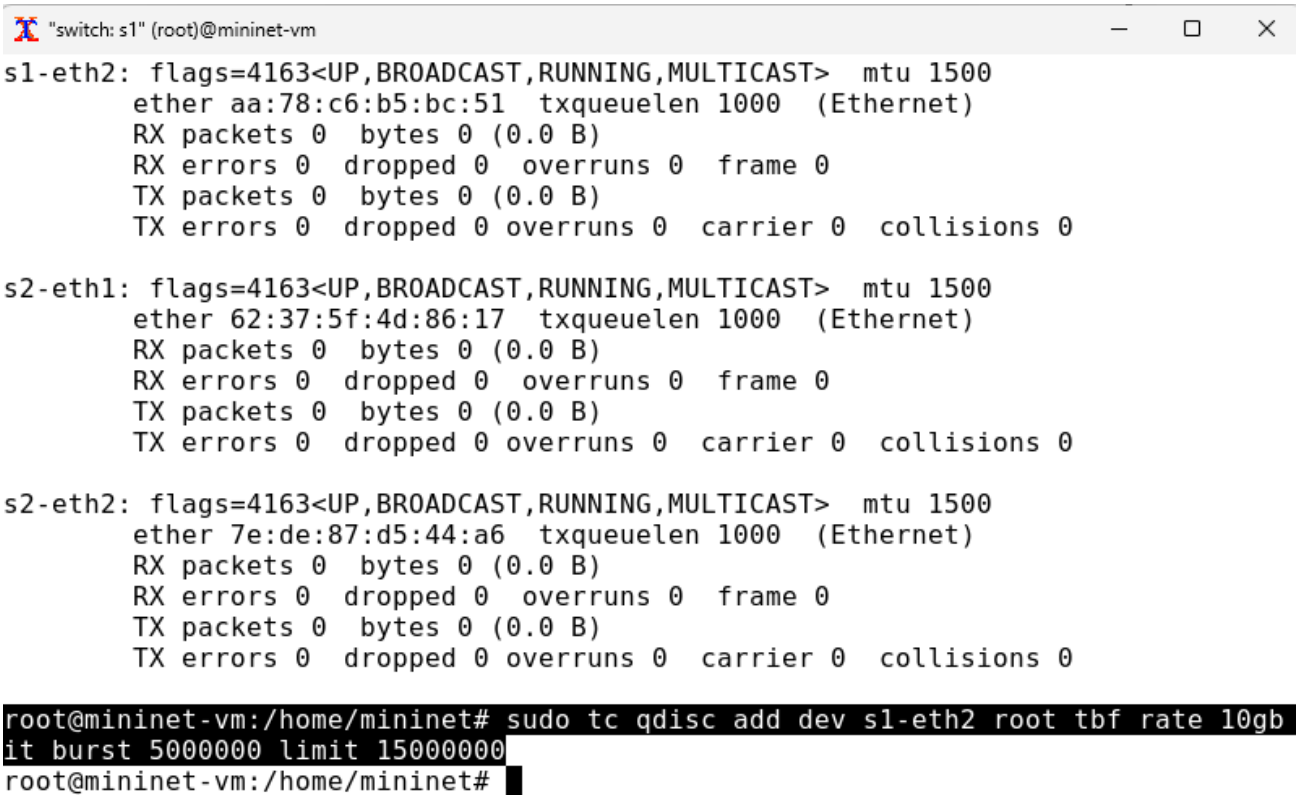


*Рис. 14: Применение правила ограничения скорости tbf*

Проверим конфигурацию с помощью инструмента iperf3 для измерения пропускной способности.

В терминале хоста h2 запустим iPerf3 в режиме сервера (рис. 15):

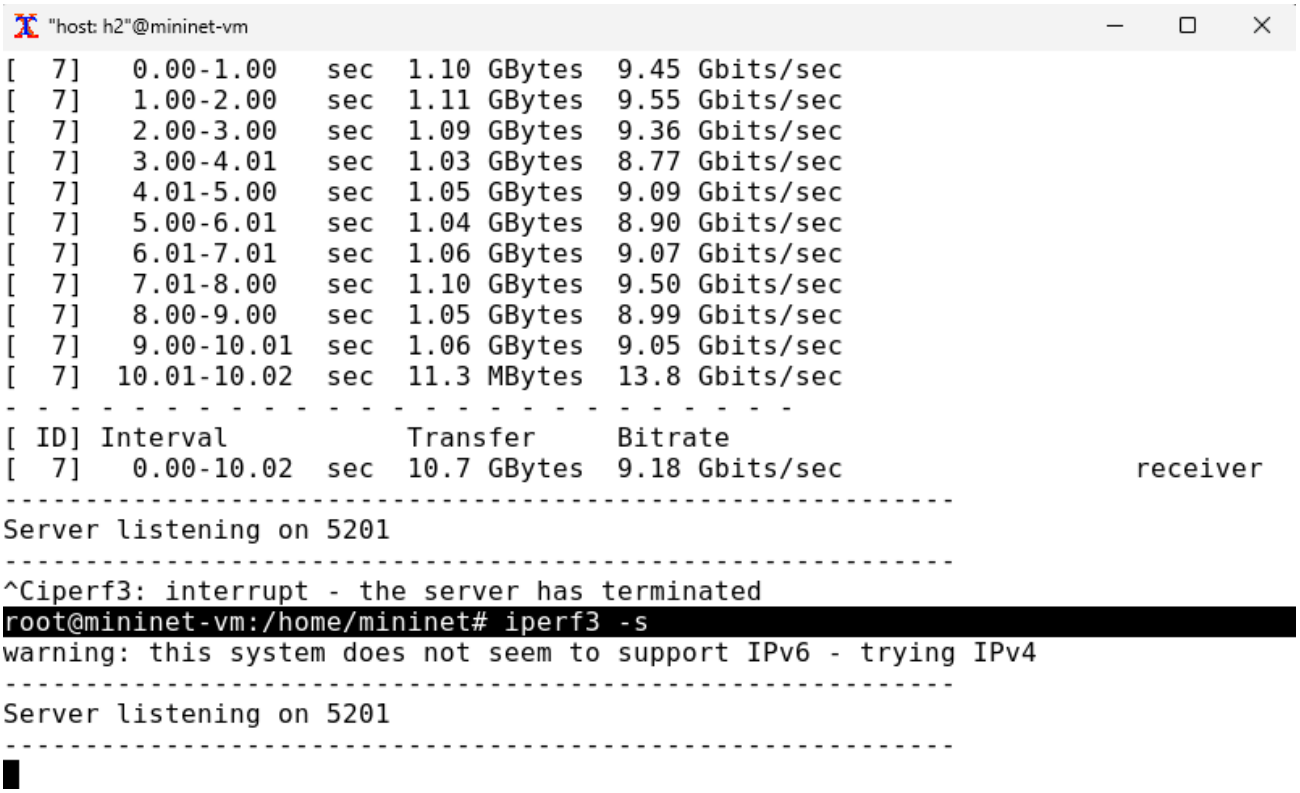


*Рис. 15: Запуск iPerf3 в режиме сервера на хосте h2*

В терминале хоста h1 запустим iPerf3 в режиме клиента (рис. 16):

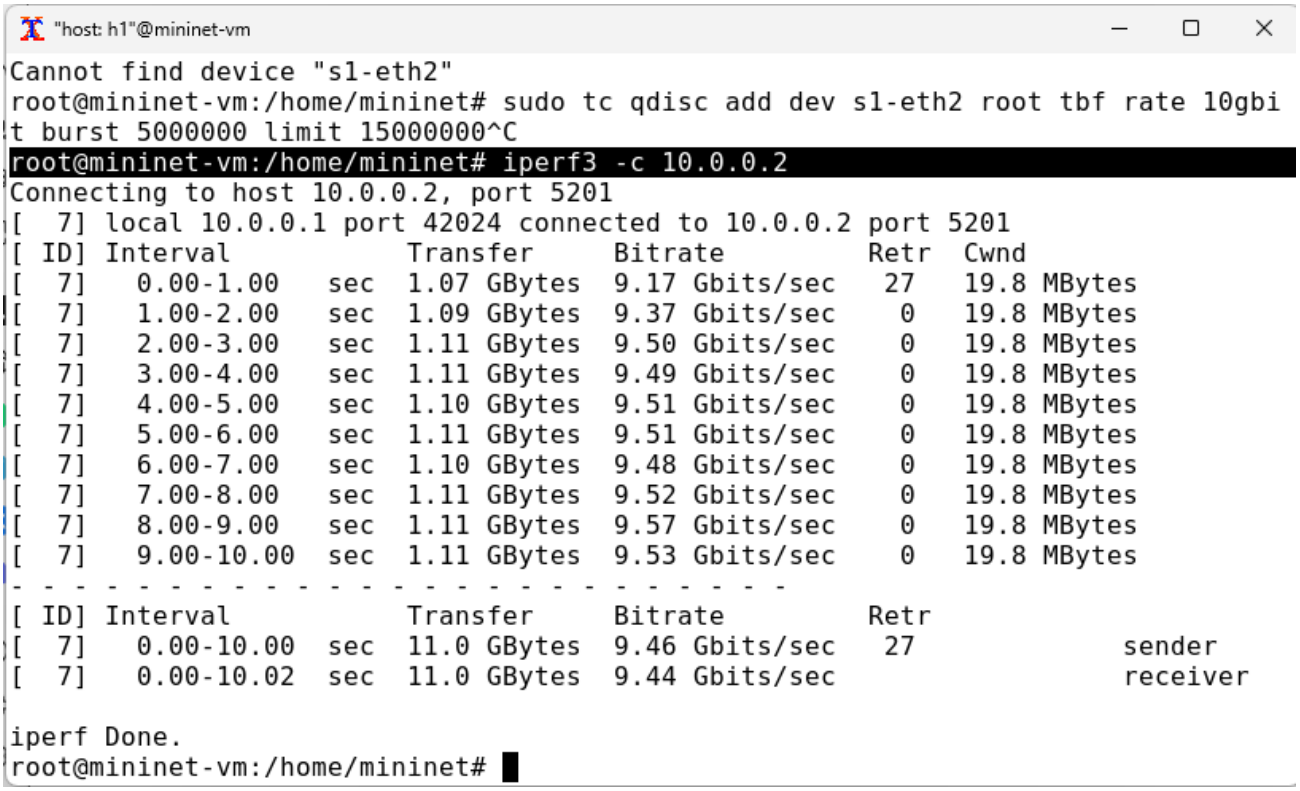


*Рис. 16: Запуск iPerf3 в режиме клиента на хосте h1*

После завершения работы iPerf3 на хосте h1 остановим iPerf3 на хосте h2, нажав Ctrl + c (рис. 17):

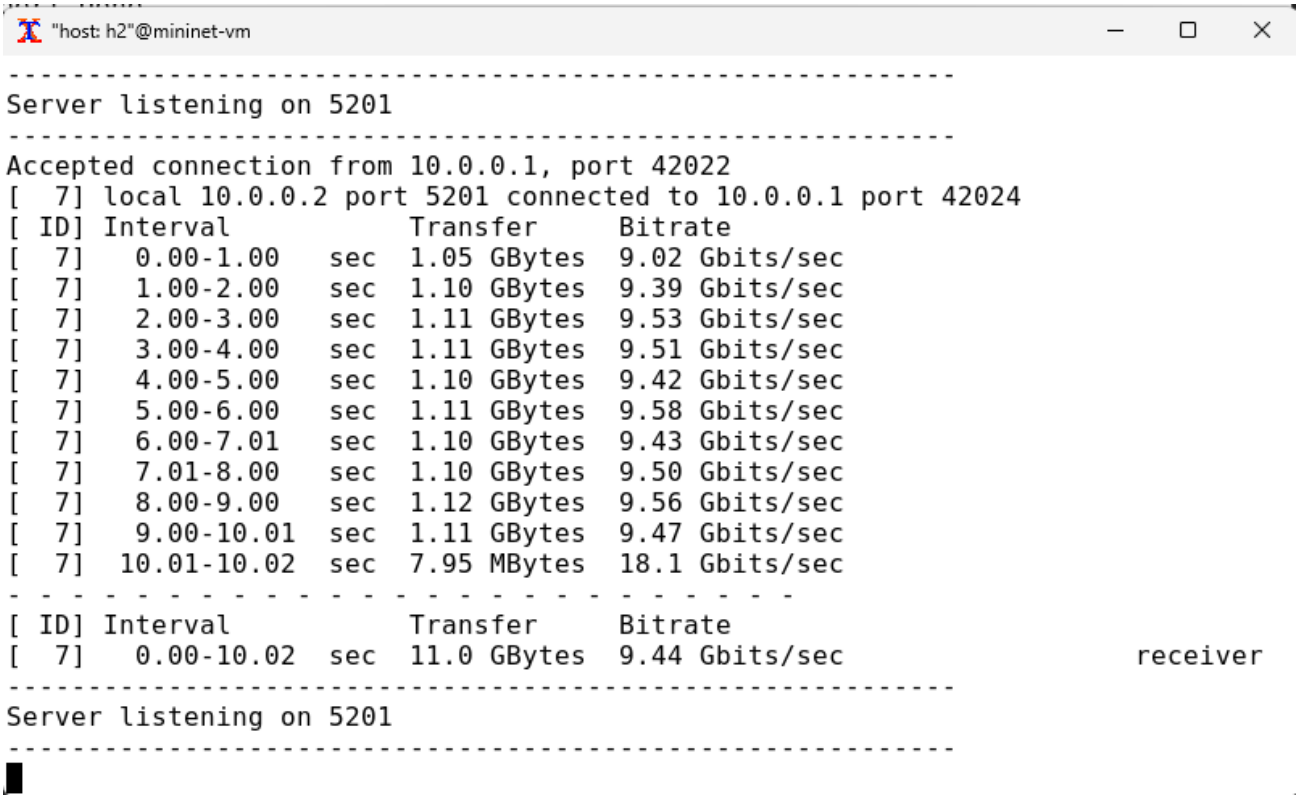


*Рис. 17: Остановка iPerf3*

Удалим модифицированную конфигурацию на коммутаторе s1 (рис. 18):

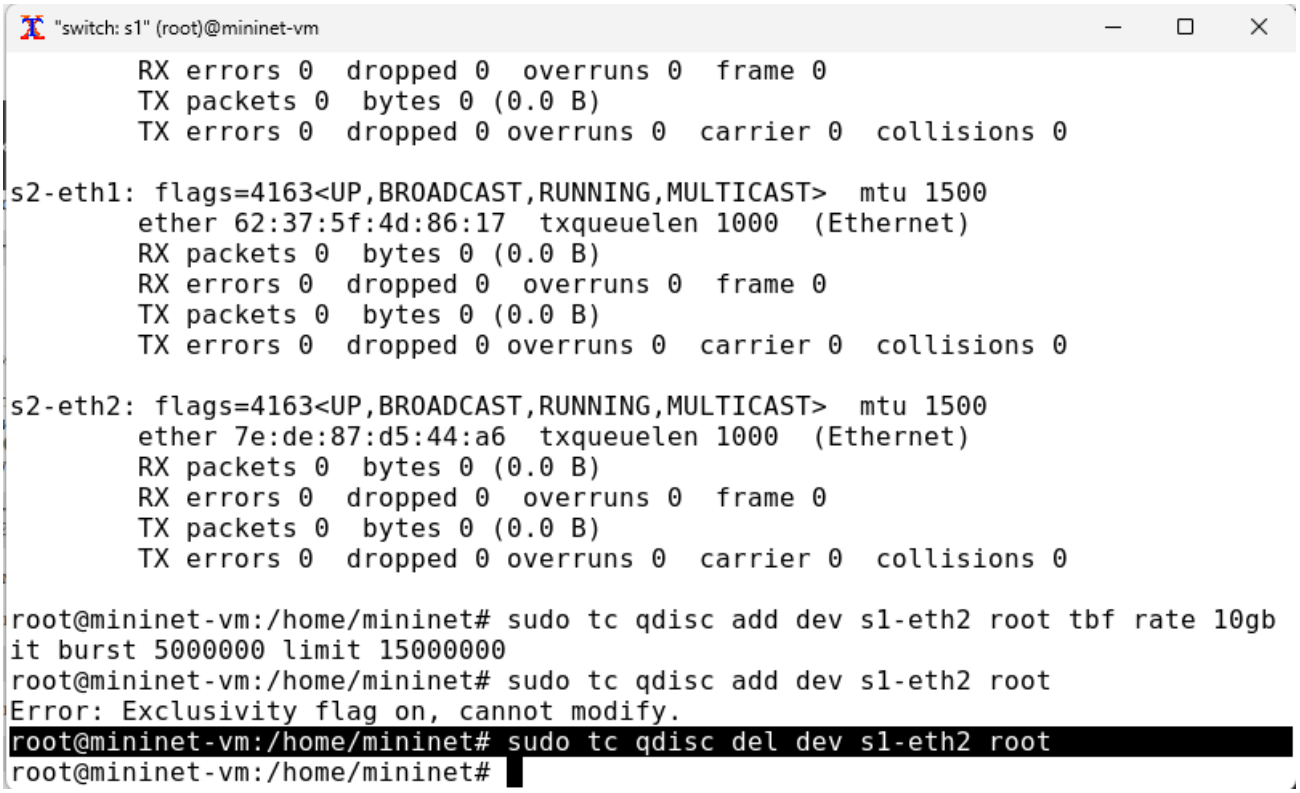


*Рис. 18: Удаление модифицированной конфигурации на коммутаторе s1*

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.

Объединим NETEM и TBF, введя на интерфейсе s1-eth2 коммутатора s1 задержку, джиттер, повреждение пакетов и указав скорость (рис. 19):

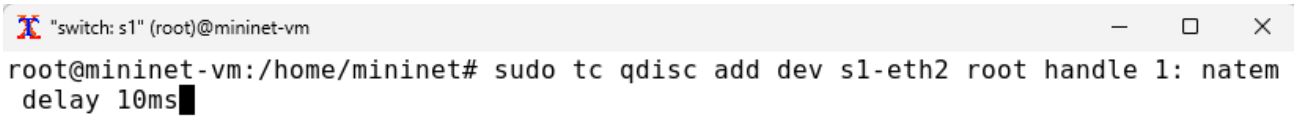


*Рис. 19: Объединение NETEM и TBF*

Убедимся, что соединение от хоста h1 к хосту h2 имеет заданную задержку. Для этого запустим команду ping с параметром -c 4 с терминала хоста h1 (рис. 20):

```
"host: h1"@mininet-vm
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=5.22 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.839 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.130 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 0.092/1.569/5.218/2.127 ms
root@mininet-vm:/home/mininet#
```

*Рис. 20: Проверка задержки*

Добавим 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. 21):

```
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#
```

*Рис. 21: Добавление второго правила на коммутаторе s1*

Проверим конфигурацию с помощью инструмента iperf3 для измерения пропускной способности.

В терминале хоста h2 запустим iPerf3 в режиме сервера (рис. 22):

```
"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
-----------------------------------------------------------
^Ciperf3: interrupt - the server has terminated
root@mininet-vm:/home/mininet# iperf3 -s
warning: this system does not seem to support IPv6 - trying IPv4
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

*Рис. 22: Запуск iPerf3 в режиме сервера на хосте h2*

В терминале хоста h1 запустим iPerf3 в режиме клиента (рис. 23):

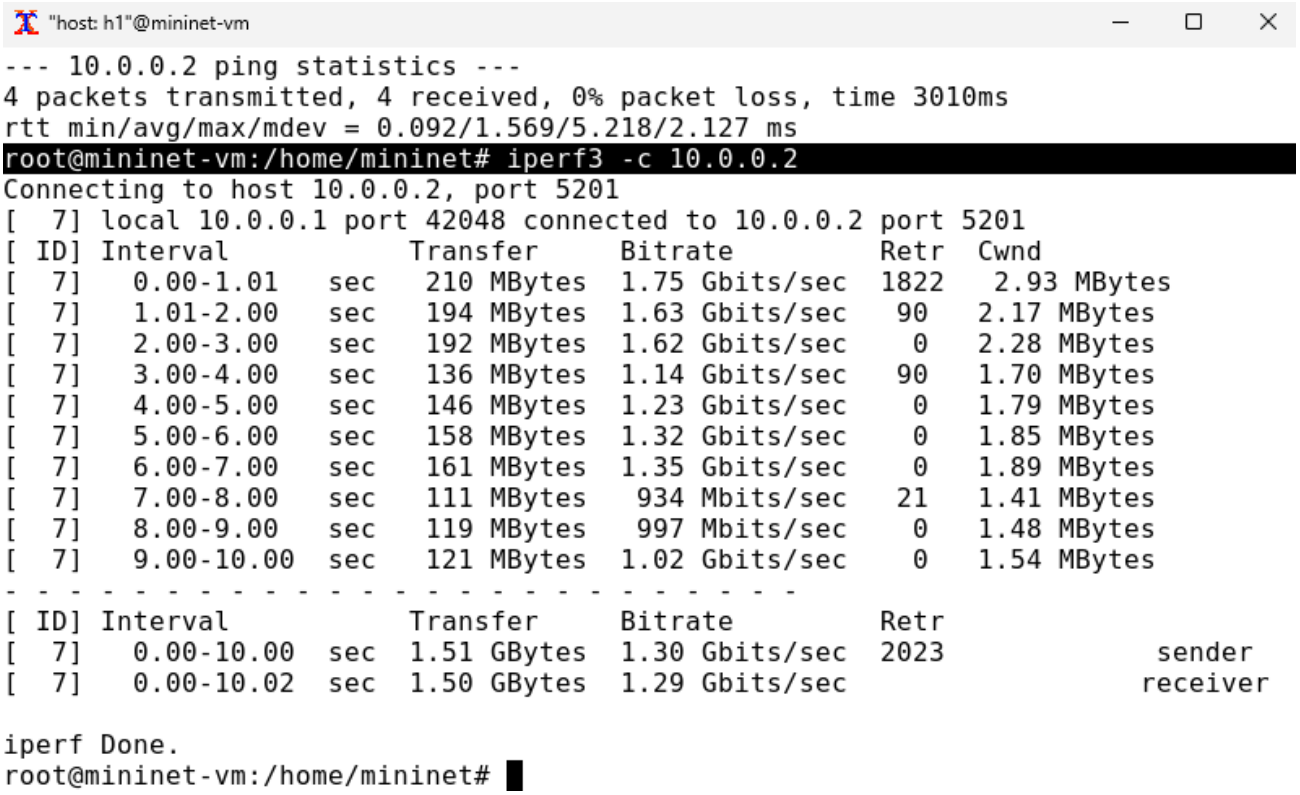


*Рис. 23: Запуск iPerf3 в режиме клиента на хосте h1*

После завершения работы iPerf3 на хосте h1 остановим iPerf3 на хосте h2, нажав Ctrl + c (рис. 24):

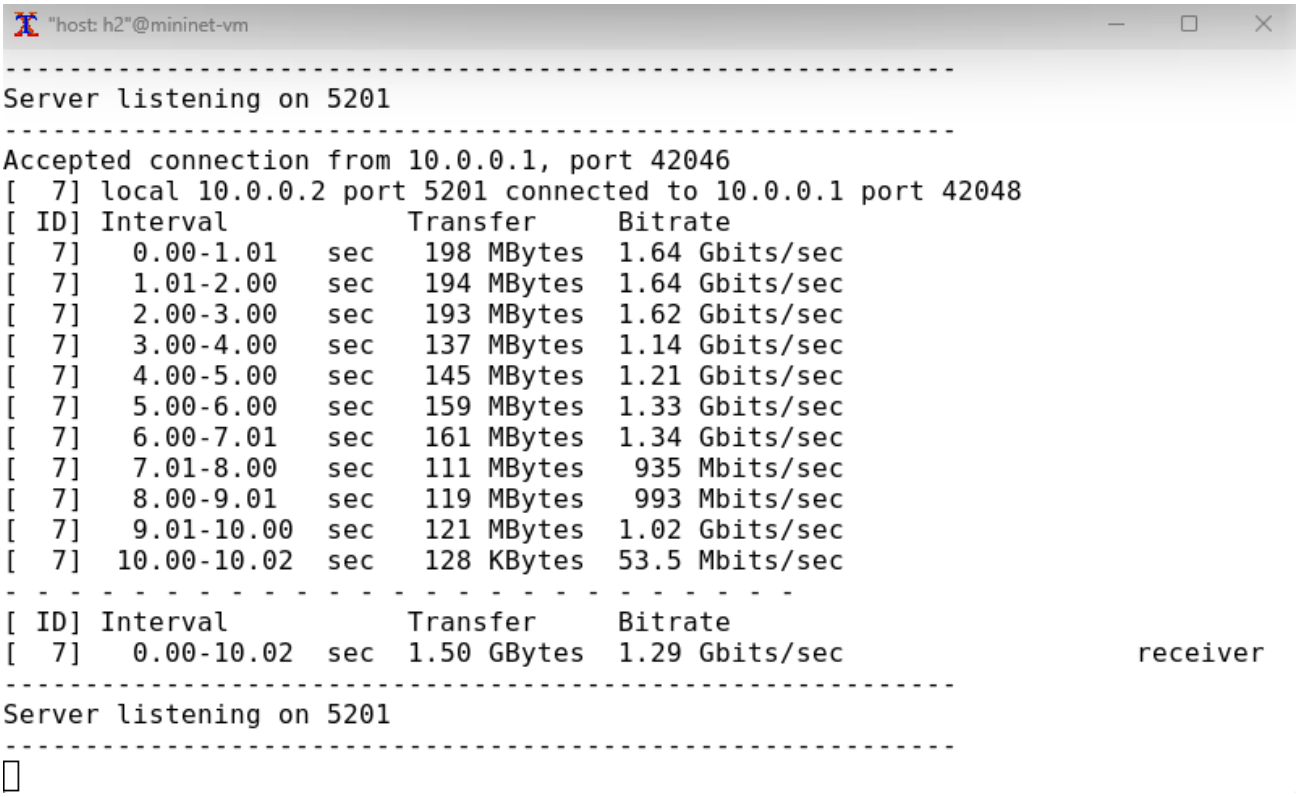


*Рис. 24: Остановка iPerf3*

Удалим модифицированную конфигурацию на коммутаторе s1 (рис. 25):

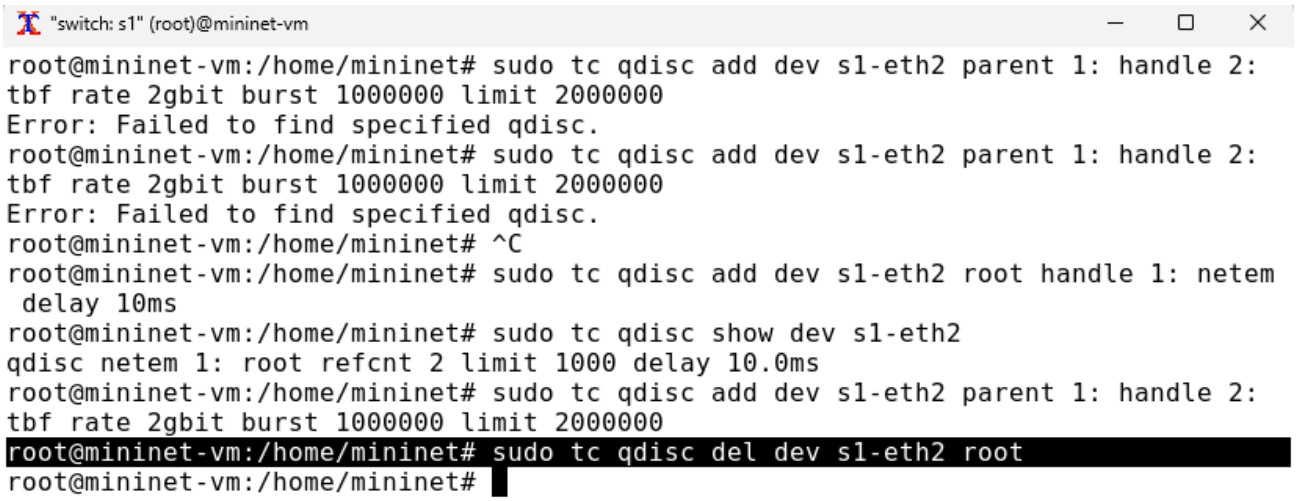


*Рис. 25: Удаление модифицированной конфигурации на коммутаторе s1*

Для самостоятельного задания создадим необходимые каталоги (рис. 26):

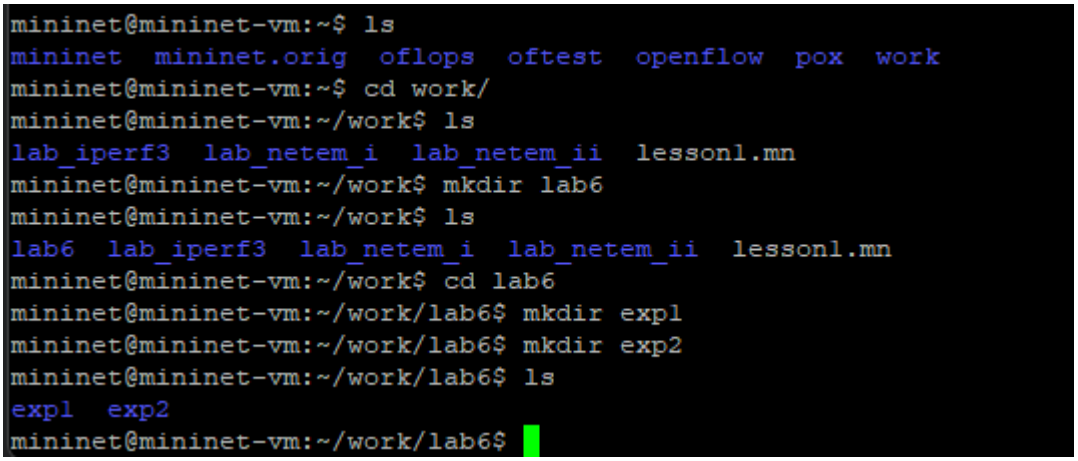


*Рис. 26: Создание необходимых каталогов*

Затем напишем скрипты по примеру из прошлых лабораторных работ (рис. 27 - рис. 29):

mininet@mininet-vm: ~/work/lab6/exp1

GNU nano 4.8

```
#!/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' )
    s1 = net.addSwitch( 's1' )
    s2 = net.addSwitch( 's2' )

    s1.cmd('ip link del s1-eth2')
    s2.cmd('ip link del s2-eth1')

    info( '*** Creating links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s1 )
    net.addLink( s1, s2 )

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

    s1.cmdPrint('sudo tc qdisc add dev s1-eth2 root handle 1: netem delay 10ms')
    s2.cmdPrint('sudo tc qdisc add dev s1-eth2 parent 1: handle 2: tbf rate 2gbit burst 1000000 limit 2000000')

    info( '*** Running CLI\n' )
    h2.cmdPrint('iperf3 -s -D -1')
    time.sleep(10) # Wait 10 seconds
    h1.cmdPrint('iperf3 -s -D -1')
    h1.cmdPrint('iperf3 -s -D -1')

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


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

*Рис. 27: Написание 1 скрипта*

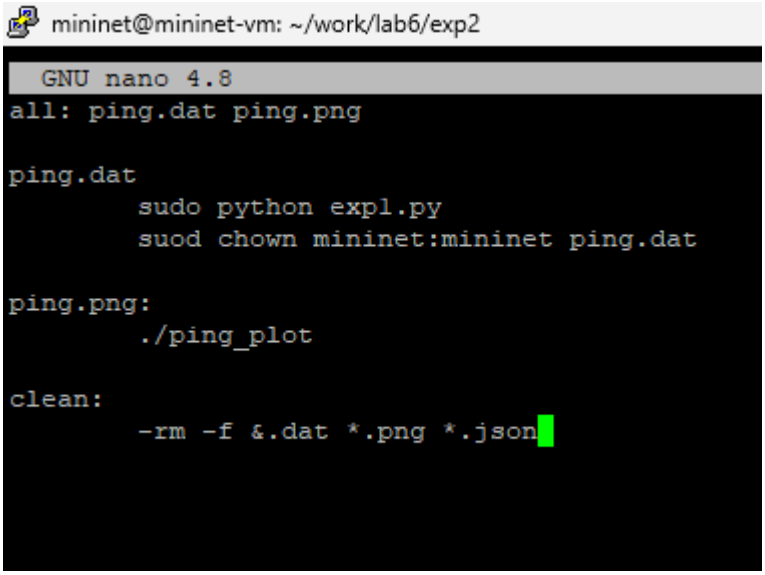


*Рис. 28: Написание 2 скрпита*

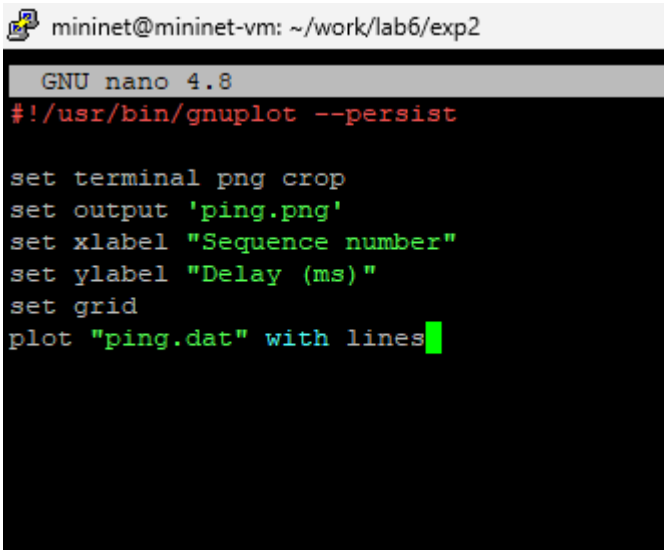


*Рис. 29: Написание 3 скрпита*

Запустим на выполнение скрипты для первой части самостоятельного задания (рис. 30):

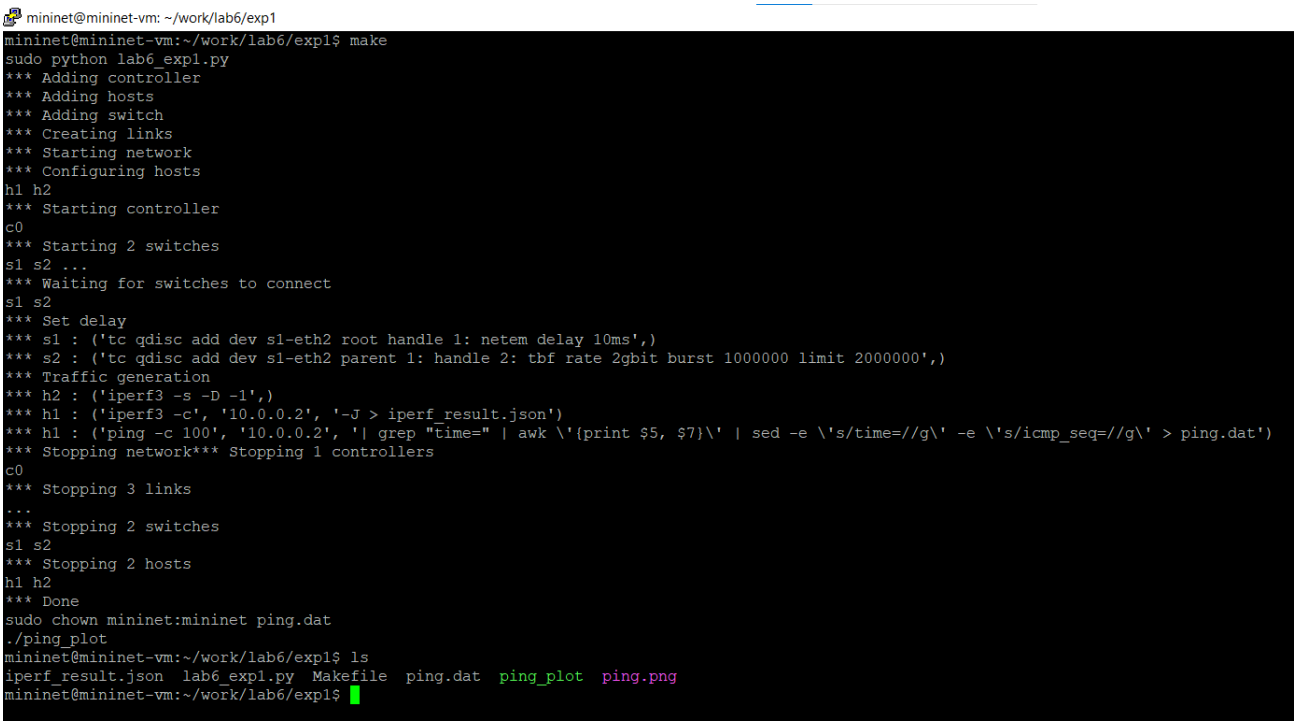


*Рис. 30: Выполнение скриптов*

Изменим параметры в скрипте для первого задания и запустим на выполнение (рис. 31 - рис. 32):

mininet@mininet-vm: ~/work/lab6/exp1

```
/home/mininet/work/lab6/exp1/lab6_exp1.py   [-M--] 92 L:[ 12+32  44/ 58] *(1029/1469b)   39 0x027
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' )
    s2 = net.addSwitch( 's2' )

    s1.cmd('ip link del s1-eth2')
    s2.cmd('ip link del s2-eth1')

    info( '*** Creating links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s1 )
    net.addLink( s1, s2 )

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

    info( '*** Set delay\n')
    s1.cmdPrint( 'tc qdisc add dev h1-eth0 root tbf rate 10gbit burst 5000000 limit 15000000' )


    info( '*** Traffic generation\n')
    h2.cmdPrint('iperf3 -s -D -1')
    time.sleep(10) # Wait 10 seconds
    h1.cmdPrint( 'iperf3 -c', h2.IP(), '-J > iperf_result.json')
    h1.cmdPrint('ping -c 100', h2.IP(), '| grep "time=" | awk \'{print $5, $7}\' | sed -e \'s/time=//g\' -e \'s/icmp_seq=//g\' > ping.dat')

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

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

*Рис. 31: Изменение параметров*

mininet@mininet-vm: ~/work/lab6/exp1

```
mininet@mininet-vm:~/work/lab6/exp1$ make
sudo python lab6_exp1.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 delay
*** s1 : ('tc qdisc add dev h1-eth0 root tbf rate 10gbit burst 5000000 limit 15000000',)
Cannot find device "h1-eth0"
*** Traffic generation
*** h2 : ('iperf3 -s -D -1',)
*** h1 : ('iperf3 -c', '10.0.0.2', '-J > iperf_result.json')
*** h1 : ('ping -c 100', '10.0.0.2', '| grep "time=" | awk \'{print $5, $7}\' | sed -e \'s/time=//g\' -e \'s/icmp_seq=//g\' > ping.dat')
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
./ping_plot
mininet@mininet-vm:~/work/lab6/exp1$ ls
iperf_result.json  lab6_exp1.py  Makefile  ping.dat  ping_plot  ping.png
mininet@mininet-vm:~/work/lab6/exp1$
```

*Рис. 32: Выполнение скриптов*

Просмотрим полученные графики (рис. 33 - рис. 34):

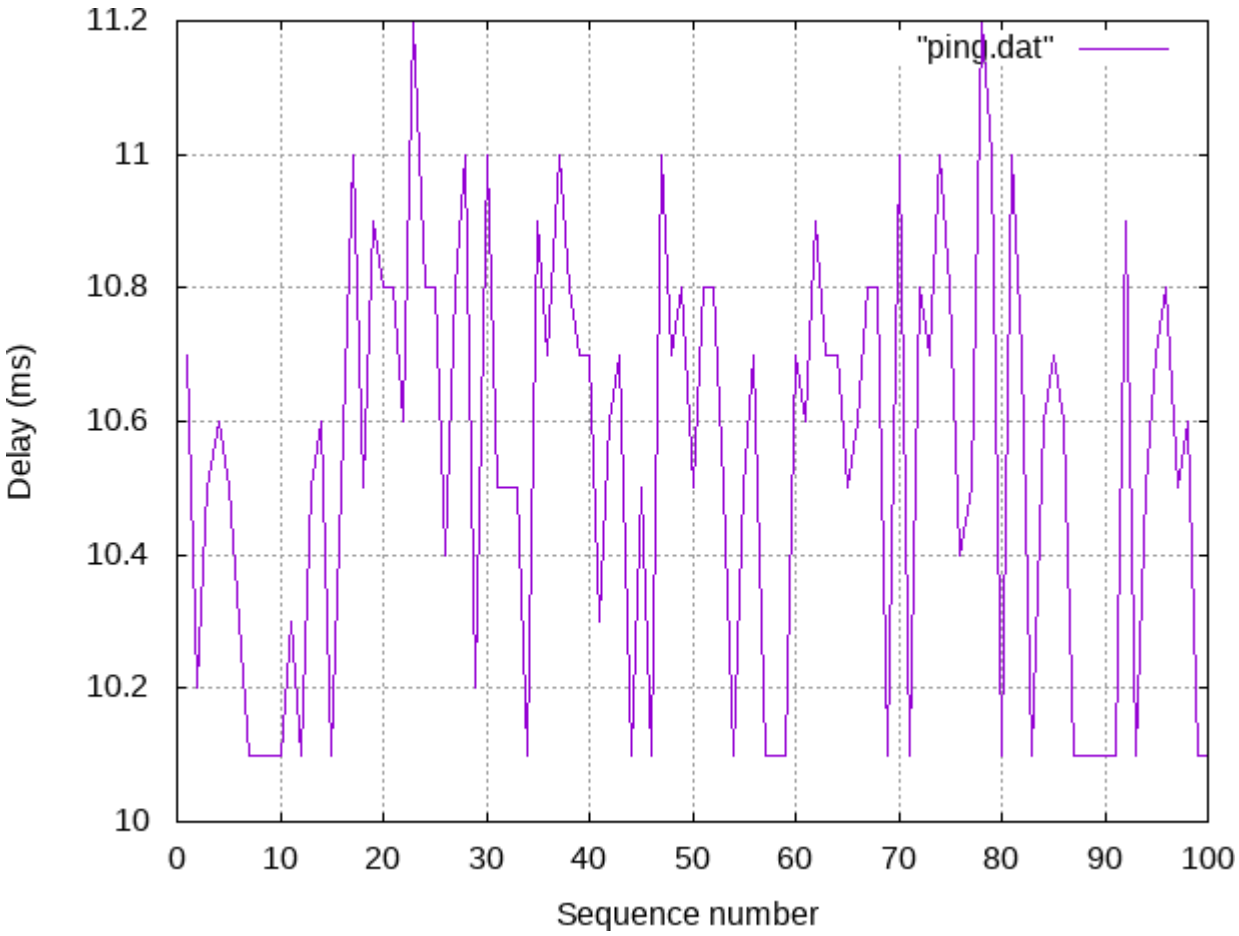


*Рис. 33: График №1*

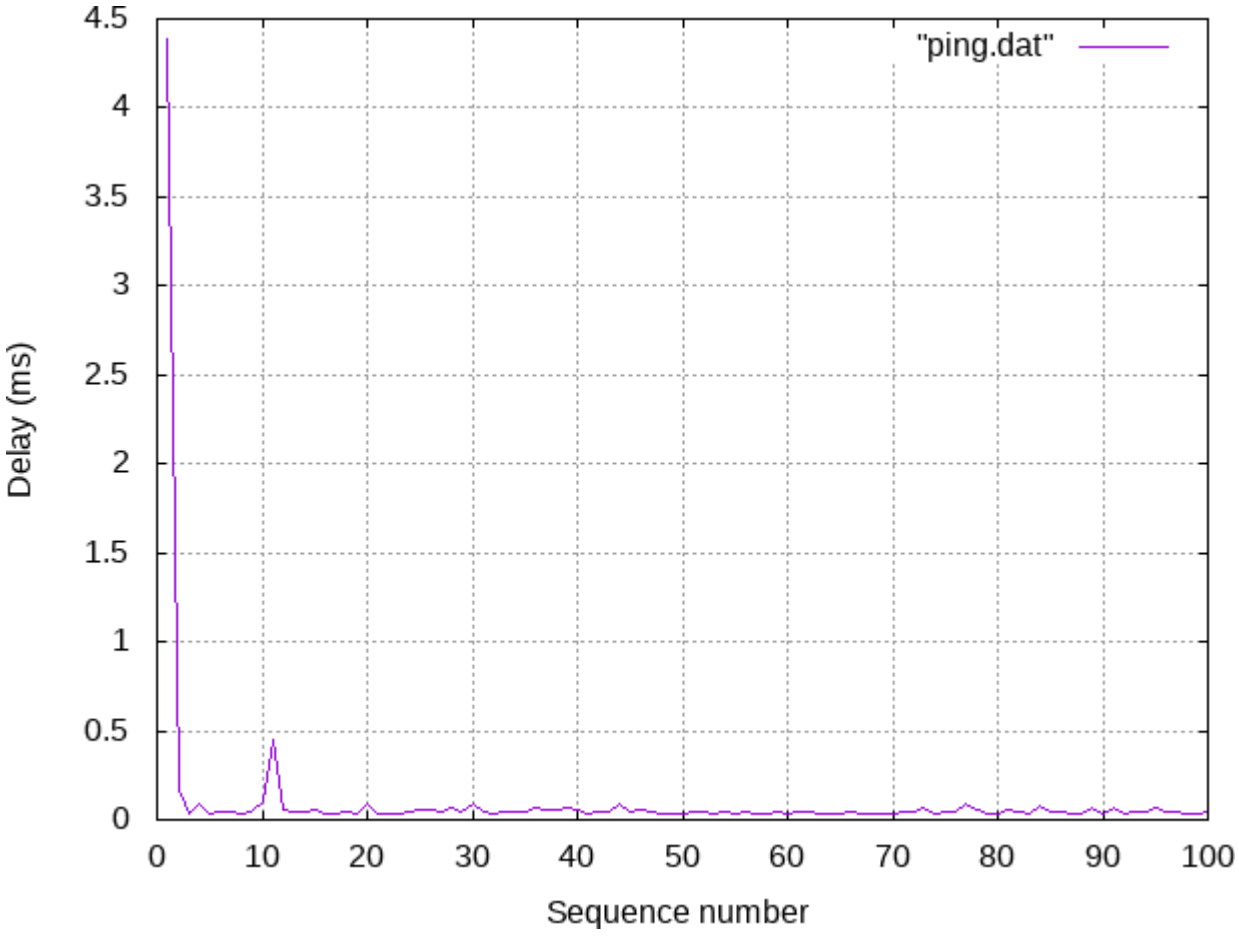


*Рис. 34: График №2*

## 3 Вывод

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

## 4 Список литературы. Библиография

[1] Mininet: https://mininet.org/