

# **Отчет по лабораторной работе №4**

**Дисциплина: Моделирование сетей передачи данных**

Амуничников Антон Игоревич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Запуск лабораторной топологии . . . . .	8
4.2	Интерактивные эксперименты . . . . .	11
4.3	Воспроизведение экспериментов . . . . .	18
<b>5</b>	<b>Выводы</b>	<b>25</b>
	<b>Список литературы</b>	<b>26</b>

## Список иллюстраций

4.1	Исправление прав запуска X-соединения . . . . .	8
4.2	Простейшая топология . . . . .	9
4.3	ifconfig на хостах h1 и h2 . . . . .	10
4.4	Проверка подключения между хостами . . . . .	11
4.5	Добавление задержки в 100мс . . . . .	12
4.6	Двунаправленная задержка соединения . . . . .	13
4.7	Изменение задержки на 50мс . . . . .	14
4.8	Восстановление исходных значений задержки . . . . .	15
4.9	Добавление значения дрожания задержки в интерфейс подключения . . . . .	16
4.10	Добавление значения корреляции для джиттера и задержки в интерфейс подключения . . . . .	17
4.11	Распределение задержки в интерфейсе подключения . . . . .	18
4.12	Скрипт для визуализации ping_plot . . . . .	21
4.13	Создание каталогов, права к файлу скрипта . . . . .	21
4.14	Makefile для управления процессом проведения эксперимента . .	22
4.15	Результат выполнения скрипта . . . . .	22
4.16	Результат выполнения скрипта . . . . .	23
4.17	Скрипт rtt.py . . . . .	23
4.18	Результат работы скрипта rtt.py . . . . .	24
4.19	Добавление правила запуска скрипта в Makefile . . . . .	24

## **Список таблиц**

# 1 Цель работы

Основной целью работы является знакомство с NETEM — инструментом для тестирования производительности приложений в виртуальной сети, а также получение навыков проведения интерактивного и воспроизводимого экспериментов по измерению задержки и её дрожания (jitter) в моделируемой сети в среде Mininet.

## 2 Задание

1. Задайте простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8.
2. Проведите интерактивные эксперименты по добавлению/изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети.
3. Реализуйте воспроизводимый эксперимент по заданию значения задержки в эмулируемой глобальной сети. Постройте график.
4. Самостоятельно реализуйте воспроизводимые эксперименты по изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети. Постройте графики.

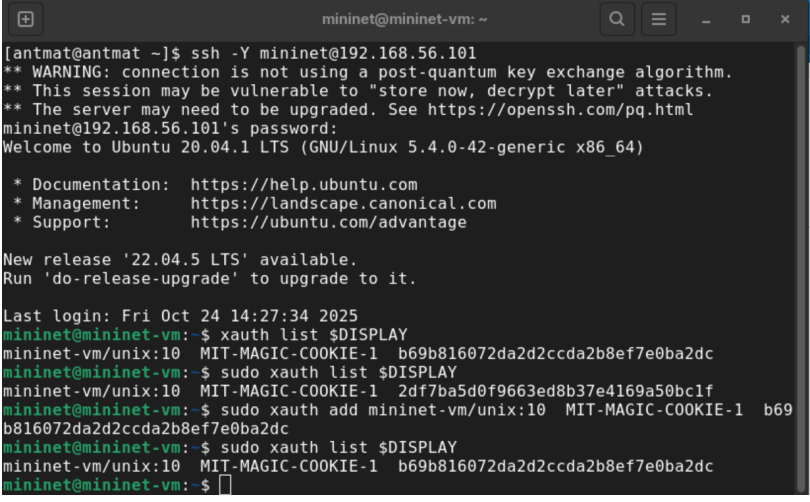
## 3 Теоретическое введение

Mininet[**mininet**] – 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.

## 4 Выполнение лабораторной работы

### 4.1 Запуск лабораторной топологии

Запустим виртуальную среду с mininet. Из основной ОС подключимся к виртуальной машине. В виртуальной машине mininet при необходимости исправим права запуска X-соединения. Скопируем значение куки (MIT magic cookie) своего пользователя mininet в файл для пользователя root (рис. 4.1).



```
mininet@mininet-vm: ~  
[antmat@antmat ~]$ ssh -Y mininet@192.168.56.101  
** WARNING: connection is not using a post-quantum key exchange algorithm.  
** This session may be vulnerable to "store now, decrypt later" attacks.  
** The server may need to be upgraded. See https://openssh.com/pq.html  
mininet@192.168.56.101's password:  
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  
  
New release '22.04.5 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Fri Oct 24 14:27:34 2025  
mininet@mininet-vm:~$ xauth list $DISPLAY  
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 b69b816072da2d2ccda2b8ef7e0ba2dc  
mininet@mininet-vm:~$ sudo xauth list $DISPLAY  
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 2df7ba5d0f9663ed8b37e4169a50bc1f  
mininet@mininet-vm:~$ sudo xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 b69  
b816072da2d2ccda2b8ef7e0ba2dc  
mininet@mininet-vm:~$ sudo xauth list $DISPLAY  
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 b69b816072da2d2ccda2b8ef7e0ba2dc  
mininet@mininet-vm:~$
```

Рисунок 4.1: Исправление прав запуска X-соединения

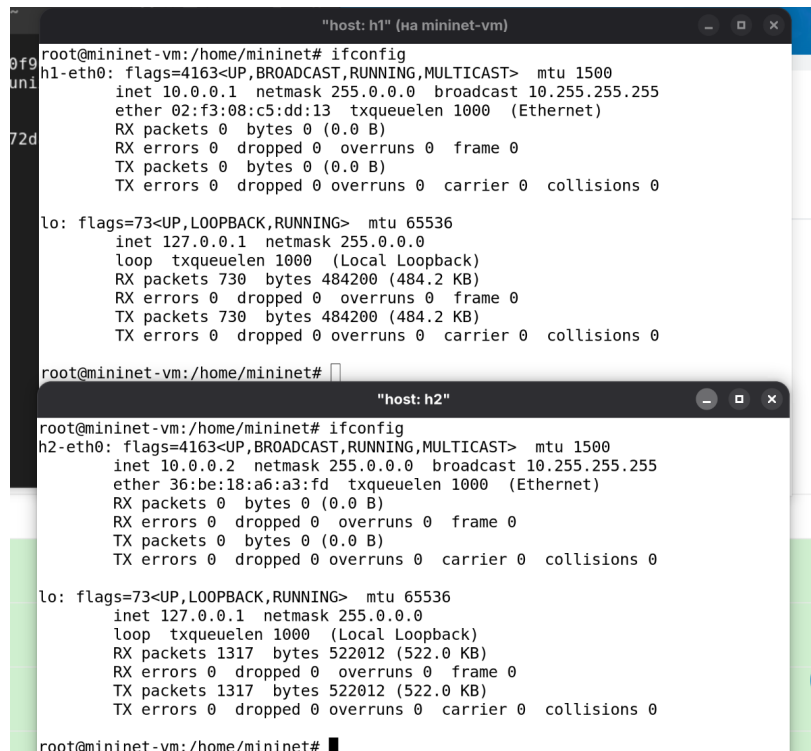
Зададим простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 4.2).



```
mininet@mininet-vm: ~  
mininet-vm:/unix:10 MIT-MAGIC-COOKIE-1 2df7ba5d0f9663ed8b37e4169a50bc1f  
mininet@mininet-vm:~$ sudo xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 b69  
b816072da2d2ccda2b8ef7e0ba2dc  
mininet@mininet-vm:~$ sudo xauth list $DISPLAY  
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 b69b816072da2d2ccda2b8ef7e0ba2dc  
mininet@mininet-vm:~$ sudo mn --topo=single,2 -x  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Running terms on localhost:10.0  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

Рисунок 4.2: Простейшая топология

На хостах h1 и h2 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 и h2-eth0 (рис. 4.3).



```
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 02:f3:08:c5:dd:13 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 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 730 bytes 484200 (484.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 730 bytes 484200 (484.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet#

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 36:be:18:a6:a3:fd 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 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1317 bytes 522012 (522.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1317 bytes 522012 (522.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet#
```

Рисунок 4.3: ifconfig на хостах h1 и h2

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 6 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=1.80 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.207 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.041 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5095ms
rtt min/avg/max/mdev = 0.041/0.365/1.797/0.643 ms
root@mininet-vm:/home/mininet#

"host: h2"
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 netmask 255.0.0.0
loop txqueuelen 1000 (Local Loopback)
RX packets 1317 bytes 522012 (522.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1317 bytes 522012 (522.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet# ping -c 6 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=2.19 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.047 ms
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.039/0.406/2.191/0.797 ms
root@mininet-vm:/home/mininet#
```

Рисунок 4.4: Проверка подключения между хостами

## 4.2 Интерактивные эксперименты

### 4.2.1 Добавление/изменение задержки в эмулируемой глобальной сети

Сетевые эмуляторы задают задержки на интерфейсе. Например, задержка, вносимая в интерфейс коммутатора А, который подключён к интерфейсу коммутатора В, может представлять собой задержку распространения WAN, соединяющей оба коммутатора.

На хосте h1 добавим задержку в 100 мс к выходному интерфейсу.

```
sudo tc qdisc add dev h1-eth0 root netem delay 100ms
```

- sudo: выполнить команду с более высокими привилегиями;
- tc: вызвать управление трафиком Linux;

- qdisc: изменить дисциплину очередей сетевого планировщика;
- add: создать новое правило;
- dev h1-eth0: указать интерфейс, на котором будет применяться правило;
- netem: использовать эмулятор сети;
- delay 100ms: задержка ввода 100 мс.

Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду ping с параметром -c 6 с хоста h1 (рис. 4.5). Минимальное - (100.467), среднее - (101.597), максимальное - (102.384), стандартное отклонение времени (0.656) приема-передачи.

```

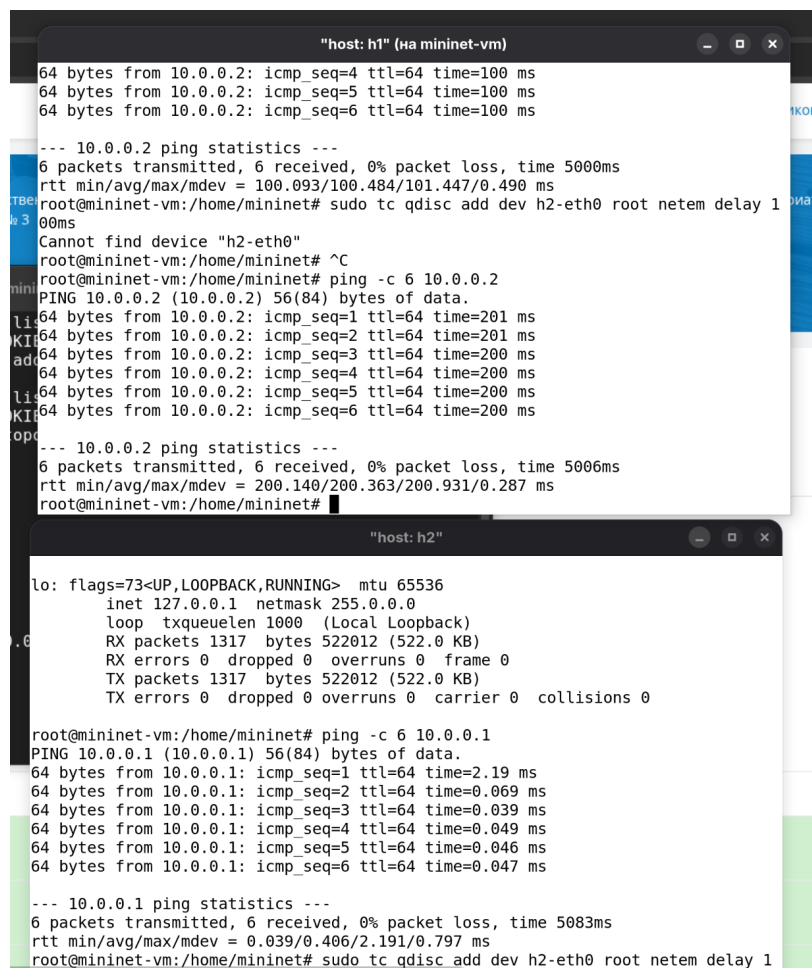
"host: h1" (на mininet-vm)
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.207 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.041 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5095ms
rtt min/avg/max/mdev = 0.041/0.365/1.797/0.643 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# ping -c 6 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=101 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=100 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 100.093/100.484/101.447/0.490 ms
root@mininet-vm:/home/mininet#

```

Рисунок 4.5: Добавление задержки в 100мс

Для эмуляции глобальной сети с двунаправленной задержкой необходимо к соответствующему интерфейсу на хосте h2 также добавим задержку в 100 миллисекунд (рис. 4.6).

Проверим, что соединение между хостом h1 и хостом h2 имеет RTT в 200 мс (100 мс от хоста h1 к хосту h2 и 100 мс от хоста h2 к хосту h1), повторив команду ping с параметром -c 6 на терминале хоста h1. Минимальное - (202.068), среднее - (202.884), максимальное - (204.777), стандартное отклонение времени (0.943) приема-передачи.



```
"host: h1" (на mininet-vm)
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=100 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 100.093/100.484/101.447/0.490 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 100ms
Cannot find device "h2-eth0"
root@mininet-vm:/home/mininet# ^C
root@mininet-vm:/home/mininet# ping -c 6 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=201 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=201 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=200 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 200.140/200.363/200.931/0.287 ms
root@mininet-vm:/home/mininet#

"host: h2"
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1317 bytes 522012 (522.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1317 bytes 522012 (522.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet# ping -c 6 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=2.19 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.047 ms

--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.039/0.406/2.191/0.797 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 1
```

Рисунок 4.6: Двухнаправленная задержка соединения

#### 4.2.2 Изменение задержки в эмулируемой глобальной сети

Изменим задержку со 100 мс до 50 мс для отправителя h1 и для получателя h2 (рис. 4.7).

Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду ping с параметром -c 6 с терминала хоста h1. Минимальное - (101.050), среднее - (101.253), максимальное - (108.223), стандартное отклонение времени (2.475) приема-передачи.

```
"host: h1" (на mininet-vm)
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=151 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=151 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=150 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=150 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=150 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=150 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 150.132/150.341/150.887/0.276 ms
root@mininet-vm:/home/mininet# ping -c 6 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=100 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=100 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 100.134/100.228/100.443/0.108 ms
root@mininet-vm:/home/mininet#

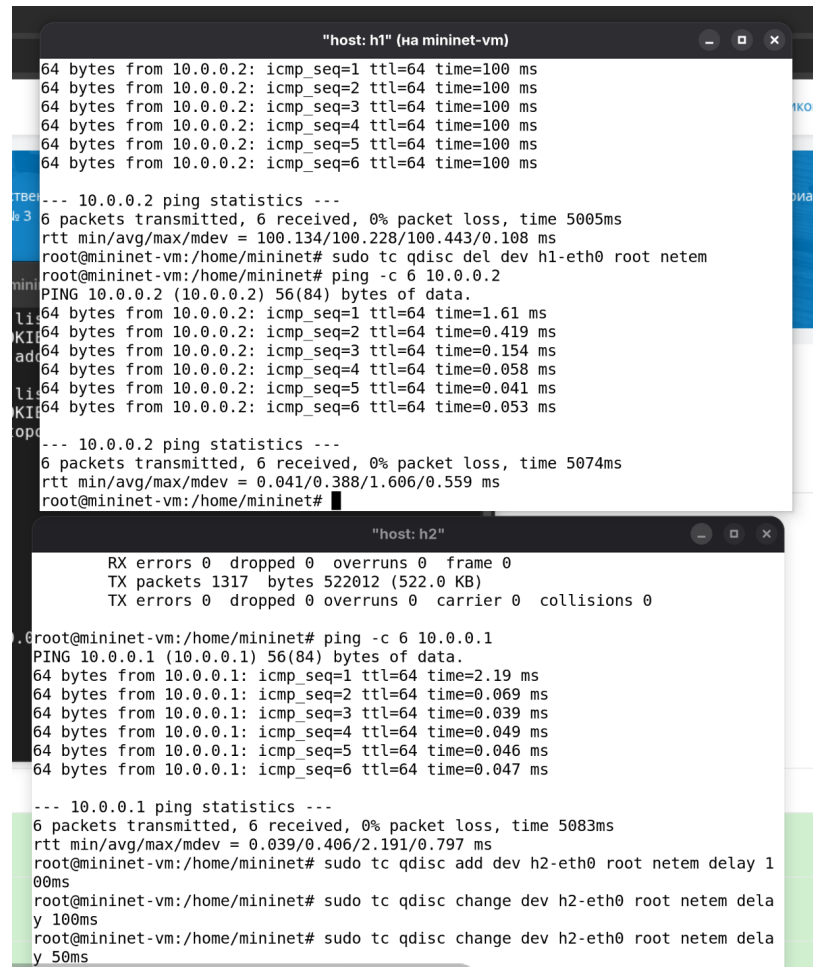
"host: h2"
RX packets 1317 bytes 522012 (522.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1317 bytes 522012 (522.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@mininet-vm:/home/mininet# ping -c 6 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.19 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.047 ms
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.039/0.406/2.191/0.797 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h2-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h2-eth0 root netem dela
```

Рисунок 4.7: Изменение задержки на 50мс

### 4.2.3 Восстановление исходных значений (удаление правил) задержки в эмулируемой глобальной сети

Восстановим конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса. Для отправителя h1: `sudo tc qdisc del dev h1-eth0 root netem`. Для получателя h2: `sudo tc qdisc del dev h2-eth0 root netem`. Проверим, что соединение между хостом h1 и хостом h2 не имеет явно установленной задержки, используя команду `ping` с параметром `-c 6` с терминала хоста h1. Минимальное - (0.145), среднее - (1.256), максимальное - (4.875), стандартное отклонение

времени (1.655) приема-передачи (рис. 4.8).



```
"host: h1" (на mininet-vm)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=100 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 100.134/100.228/100.443/0.108 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet# ping -c 6 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=1.61 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.154 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.053 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5074ms
rtt min/avg/max/mdev = 0.041/0.388/1.606/0.559 ms
root@mininet-vm:/home/mininet#

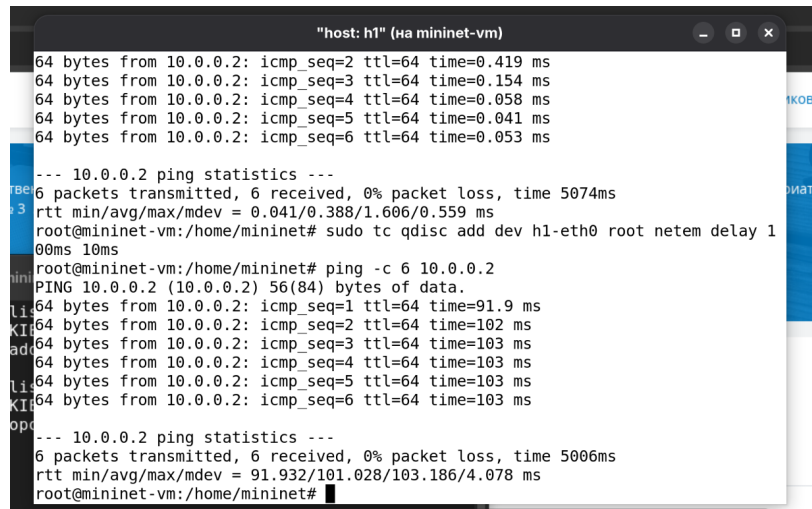
"host: h2"
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1317 bytes 522012 (522.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@mininet-vm:/home/mininet# ping -c 6 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.19 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.047 ms
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.039/0.406/2.191/0.797 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h2-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h2-eth0 root netem delay 50ms
```

Рисунок 4.8: Восстановление исходных значений задержки

#### 4.2.4 Добавление значения дрожания задержки в интерфейс подключения к эмулируемой глобальной сети

Добавим на узле h1 задержку в 100 мс со случайным отклонением 10 мс. Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс со случайным отклонением  $\pm 10$  мс, используя в терминале хоста h1 команду ping с параметром -c 6. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (91.961), среднее - (99.758), максимальное - (106.319),

стандартное отклонение времени (5.130) приема-передачи (рис. 4.9).



```
host: h1 (на mininet-vm)
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.154 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.053 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5074ms
rtt min/avg/max/mdev = 0.041/0.388/1.606/0.559 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms
root@mininet-vm:/home/mininet# ping -c 6 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=91.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=103 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 91.932/101.028/103.186/4.078 ms
root@mininet-vm:/home/mininet#
```

Рисунок 4.9: Добавление значения дрожания задержки в интерфейс подключения

#### 4.2.5 Добавление значения корреляции для джиттера и задержки в интерфейс подключения к эмулируемой глобальной сети

Добавим на интерфейсе хоста h1 задержку в 100 мс с вариацией  $\pm 10$  мс и значением корреляции в 25%. Убедимся, что все пакеты, покидающие устройство h1 на интерфейсе h1-eth0, будут иметь время задержки 100 мс со случайным отклонением  $\pm 10$  мс, при этом время передачи следующего пакета зависит от предыдущего значения на 25%. Используем для этого в терминале хоста h1 команду `ping` с параметром `-c 20`. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (91.887), среднее - (102.565), максимальное - (111.050), стандартное отклонение времени (5.730) приема-передачи (рис. 4.10).

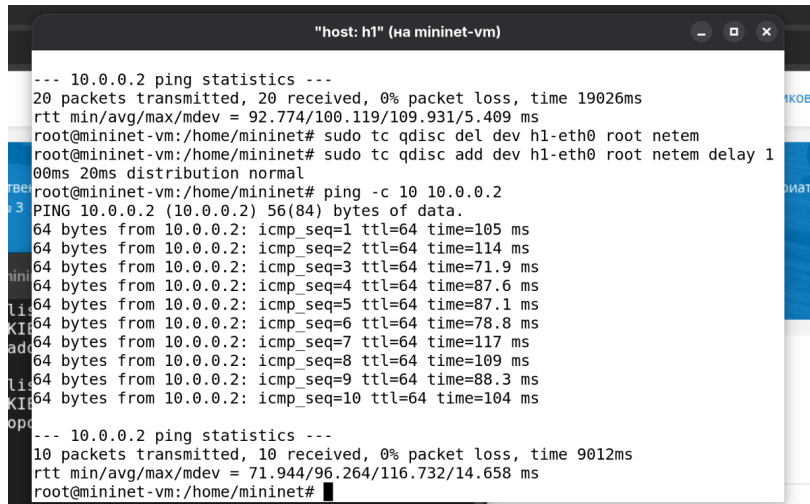


```
"host: h1" (на mininet-vm)
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=94.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=99.8 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=93.1 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=99.1 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=97.0 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=97.6 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=96.4 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=92.8 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=96.7 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=93.8 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=104 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=92.8 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=110 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=106 ms
--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19026ms
rtt min/avg/max/mdev = 92.774/100.119/109.931/5.409 ms
root@mininet-vm:/home/mininet#
```

Рисунок 4.10: Добавление значения корреляции для джиттера и задержки в интерфейс подключения

#### 4.2.6 Распределение задержки в интерфейсе подключения к эмулируемой глобальной сети

Зададим нормальное распределение задержки на узле h1 в эмулируемой сети. Убедимся, что все пакеты, покидающие хост h1 на интерфейсе h1-eth0, будут иметь время задержки, которое распределено в диапазоне 100 мс  $\pm$  20 мс. Используем для этого команду ping на терминале хоста h1 с параметром -с 10. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (74.585), среднее - (105.367), максимальное - (126.328), стандартное отклонение времени (16.340) приема-передачи. Завершим работу mininet в интерактивном режиме (рис. 4.11).



```
--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19026ms
rtt min/avg/max/mdev = 92.774/100.119/109.931/5.409 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 1
00ms 20ms distribution normal
root@mininet-vm:/home/mininet# ping -c 10 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=105 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=114 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=71.9 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=87.6 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=87.1 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=78.8 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=117 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=109 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=88.3 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=104 ms
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 71.944/96.264/116.732/14.658 ms
root@mininet-vm:/home/mininet#
```

Рисунок 4.11: Распределение задержки в интерфейсе подключения

## 4.3 Воспроизведение экспериментов

### 4.3.1 Предварительная подготовка

Обновим репозитории программного обеспечения на виртуальной машине: `sudo apt-get update`. Установим пакет `geeie` — понадобится для просмотра файлов `png`: `sudo apt install geeie`.

Для каждого воспроизводимого эксперимента `exrname` создадим свой каталог, в котором будут размещаться файлы эксперимента: `mkdir -p ~/work/lab_netem_i/exrname`. Здесь `exrname` может принимать значения `simple-delay`, `change-delay`, `jitter-delay`, `correlation-delay` и т.п. Для каждого случая создадим скрипт для проведения эксперимента `lab_netem_i.py` и скрипт для визуализации результатов `ping_plot`.

### 4.3.2 Добавление задержки для интерфейса, подключающегося к эмулируемой глобальной сети

С помощью API Mininet воспроизведем эксперимент по добавлению задержки для интерфейса хоста, подключающегося к эмулируемой глобальной сети.

В виртуальной среде mininet в своём рабочем каталоге с проектами создадим каталог simple-delay и перейдем в него.

```
mkdir -p ~/work/lab_netem_i/simple-delay
cd ~/work/lab_netem_i/simple-delay
```

Создадим скрипт для эксперимента lab\_netem\_i.py.

```
#!/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\n' )
net.start()

info( '*** Set delay\n' )
h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem delay 100ms' )
h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem delay 100ms' )

time.sleep(10) # Wait 10 seconds

info( '*** Ping\n' )
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' )

```

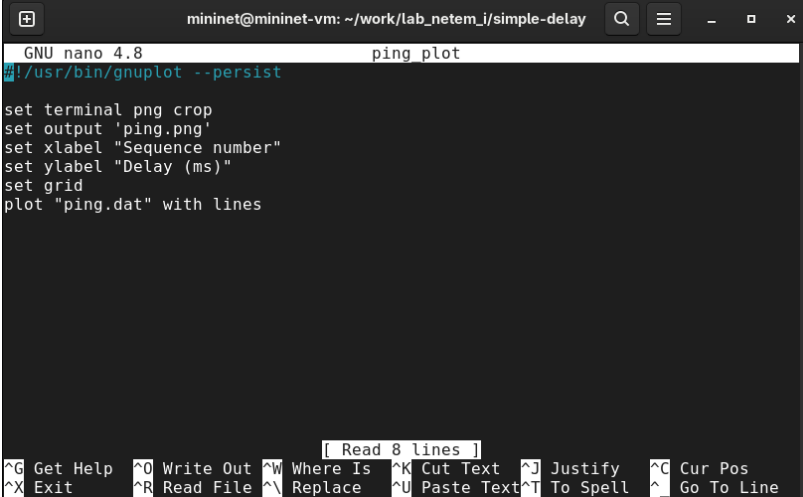
```

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

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

```

Создадим скрипт для визуализации ping\_plot результатов эксперимента (рис. 4.12).



```

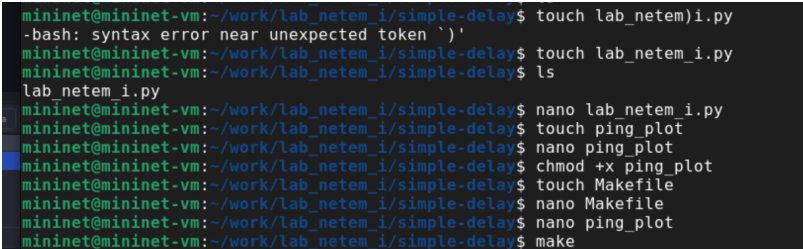
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay
GNU nano 4.8 ping_plot
#!/usr/bin/gnuplot --persist

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

```

Рисунок 4.12: Скрипт для визуализации ping\_plot

Зададим права доступа к файлу скрипта: `chmod +x ping_plot` (рис. 4.13).



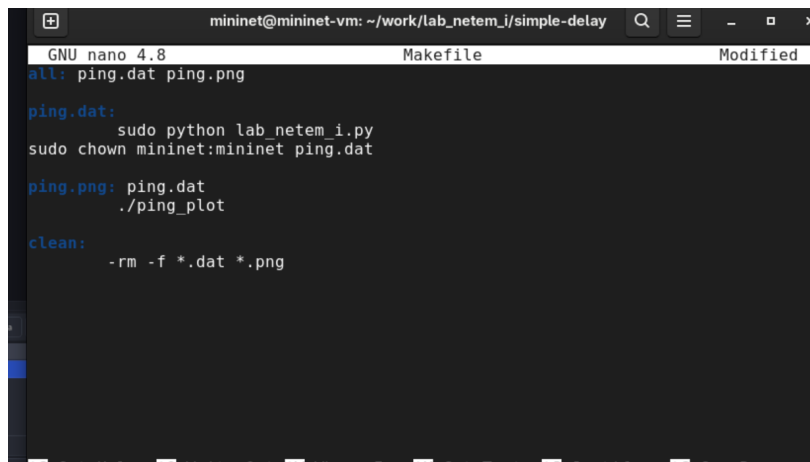
```

mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ touch lab_netem_i.py
-bash: syntax error near unexpected token `)'
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ touch lab_netem_i.py
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ ls
lab_netem_i.py
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ nano lab_netem_i.py
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ touch ping_plot
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ nano ping_plot
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ chmod +x ping_plot
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ touch Makefile
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ nano Makefile
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ nano ping_plot
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay$ make

```

Рисунок 4.13: Создание каталогов, права к файлу скрипта

Создадим Makefile для управления процессом проведения эксперимента (рис. 4.14).



```
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay
GNU nano 4.8 Makefile Modified
all: ping.dat ping.png

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

ping.png: ping.dat
    ./ping_plot

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

Рисунок 4.14: Makefile для управления процессом проведения эксперимента

Выполним эксперимент. Продемонстрируем построенный в результате выполнения скриптов график (рис. 4.15).

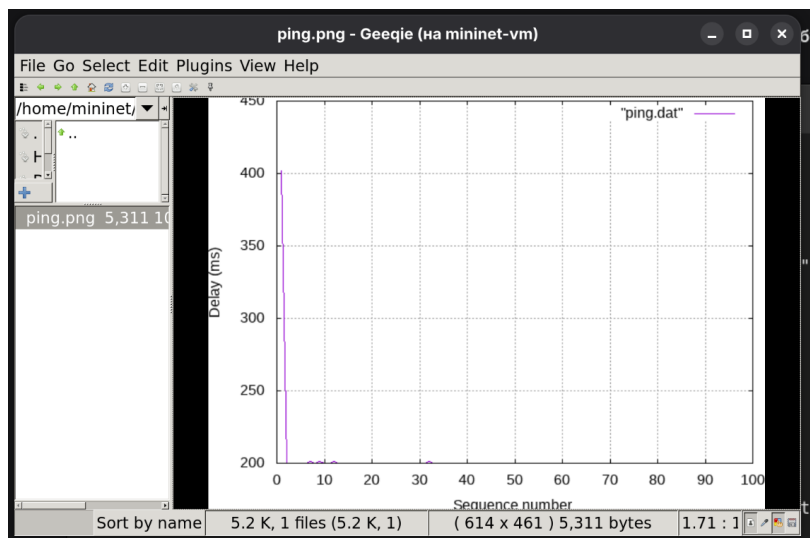


Рисунок 4.15: Результат выполнения скрипта

Из файла ping.dat удалим первую строку и заново построим график. Продемонстрируем построенный в результате график (рис. 4.16).

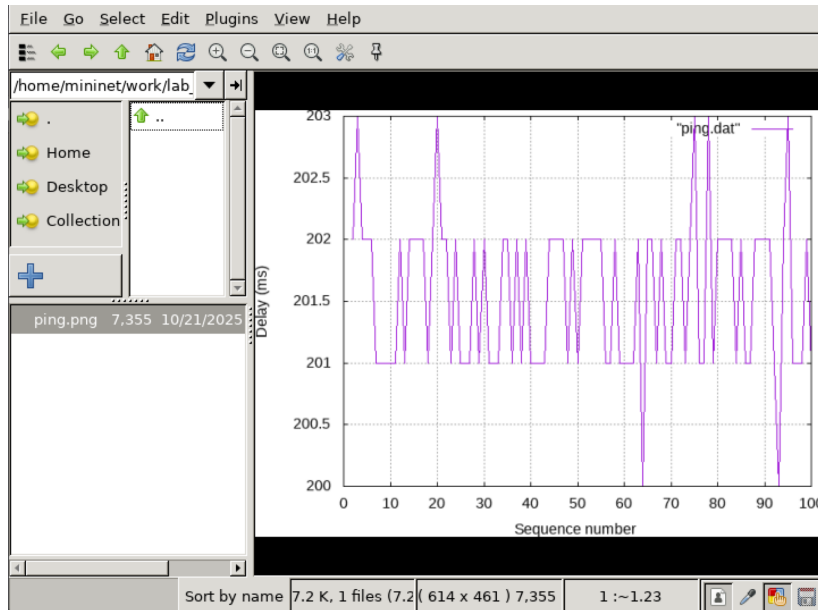


Рисунок 4.16: Результат выполнения скрипта

Разработаем скрипт для вычисления на основе данных файла ping.dat минимального, среднего, максимального и стандартного отклонения времени приёма-передачи (рис. 4.17).

```

mininet@mininet-vm: ~/work/lab_netem_l/simple-delay
GNU nano 4.8 rtt.py Modified
with open('ping.dat', 'r') as f:
    s = []
    for line in f.readlines():
        if '\n' in line:
            line.replace('\n', '')
            s.append([int(j) for j in line.split(" ")])
    s = [j[1] for j in s]
    std = (sum([(1-(sum(s)/len(s))**2 for i in s))/(len(s)-1)**0.5)
    print (f'min: {(min(s))} \max: {(max(s))} \n avg: {(sum(s)/len(s))} \n std: {(std)}')
  
```

Рисунок 4.17: Скрипт rtt.py

Продemonстрируем работу скрипта с выводом значений на экран (рис. 4.18).

```
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$ make rtt
sudo python rtt.py
min: 200
max: 203
avg: 201.56565656565655
std: 0.6253426189325271
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$
```

Рисунок 4.18: Результат работы скрипта rtt.py

Продemonстрируем работу скрипта с выводом значений на экран. Добавим правило запуска скрипта в Makefile (рис. 4.19).

```
GNU nano 4.8 Makefile
all: ping.dat ping.png

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

ping.png: ping.dat
    ./ping_plot

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

^G Get Help  ^O Write Out  ^W Where Is  | Read 11 lines |
^X Exit      ^R Read File  ^\ Replace   ^K Cut Text   ^J Justify    ^C Cur Pos   M-U Undo
^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo

4. Проверка Makefile:
```

Рисунок 4.19: Добавление правила запуска скрипта в Makefile



## 5 Выводы

В результате выполнения данной лабораторной работы я познакомился с NETEM – инструментом для тестирования производительности приложений в виртуальной сети, а также получил навыки проведения интерактивного и воспроизводимого экспериментов по измерению задержки и её дрожания (jitter) в моделируемой сети в среде Mininet.

## **Список литературы**