

Лабораторная работа №4

Эмуляция и измерение задержек в глобальных сетях

Ланцова Яна Игоревна

Содержание

1	Цель работы	5
2	Задачи	6
3	Выполнение лабораторной работы	7
3.1	Запуск лабораторной топологии	7
3.2	Добавление/изменение задержки в эмулируемой глобальной сети	9
3.3	Изменение задержки в эмулируемой глобальной сети	11
3.4	Восстановление исходных значений (удаление правил) задержки в эмулируемой глобальной сети	12
3.5	Добавление значения дрожания задержки в интерфейс подключения к эмулируемой глобальной сети	12
3.6	Добавление значения корреляции для джиттера и задержки в интерфейс подключения к эмулируемой глобальной сети	13
3.7	Распределение задержки в интерфейсе подключения к эмулируемой глобальной сети	14
3.8	Воспроизведение экспериментов. Добавление задержки для интерфейса, подключающегося к эмулируемой глобальной сети	14
3.9	Задание для самостоятельной работы	19
4	Выводы	21

Список иллюстраций

3.1	Исправление MIT magic cookie	7
3.2	Простейшая топология	8
3.3	ifconfig на хостах h1 и h2	9
3.4	Проверка подключения между хостами	9
3.5	Добавление задержки в 100мс	10
3.6	Двунаправленная задержка соединения	11
3.7	Изменение задержки на 50мс	11
3.8	Восстановление исходных значений задержки	12
3.9	Добавление значения дрожания задержки в интерфейс подключения	13
3.10	Добавление значения корреляции для джиттера и задержки в ин- терфейс подключения	13
3.11	Распределение задержки в интерфейсе подключения	14
3.12	Установка пакета и создание каталога	15
3.13	Скрипт на Python для эксперимента	15
3.14	Скрипт для визуализации ping_plot	16
3.15	Makefile для управления процессом проведения эксперимента . .	16
3.16	Запуск эксперимента	16
3.17	Визуализация эксперимента	17
3.18	Удаление строки из файла .dat	17
3.19	Визуализация эксперимента	18
3.20	Скрипт script_1.py	18
3.21	Результат работы скрипта script_1.py	18
3.22	Изменение файла lab_netem_i.py	19
3.23	Визуализация эксперимента	20
3.24	Результат работы скрипта script_1.py	20

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.

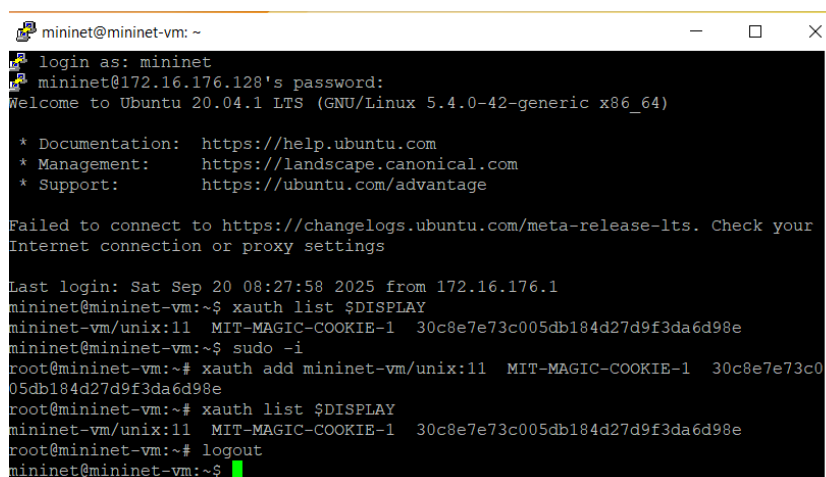
2 Задачи

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 Выполнение лабораторной работы

3.1 Запуск лабораторной топологии

Из основной ОС подключимся к виртуальной машине и исправим права запуска X-соединения. Скопируем значение куки (MIT magic cookie) своего пользователя mininet в файл для пользователя root (рис. 3.1).



```
mininet@mininet-vm: ~  
login as: mininet  
mininet@172.16.176.128'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  
  
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your  
Internet connection or proxy settings  
  
Last login: Sat Sep 20 08:27:58 2025 from 172.16.176.1  
mininet@mininet-vm:~$ xauth list $DISPLAY  
mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 30c8e7e73c005db184d27d9f3da6d98e  
mininet@mininet-vm:~$ sudo -i  
root@mininet-vm:~# xauth add mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 30c8e7e73c0  
05db184d27d9f3da6d98e  
root@mininet-vm:~# xauth list $DISPLAY  
mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 30c8e7e73c005db184d27d9f3da6d98e  
root@mininet-vm:~# logout  
mininet@mininet-vm:~$
```

Рис. 3.1: Исправление MIT magic cookie

Зададим простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 3.2). После введения этой команды запустятся терминалы двух хостов, коммутатора и контроллера. Терминалы коммутатора и контроллера можно закрыть.

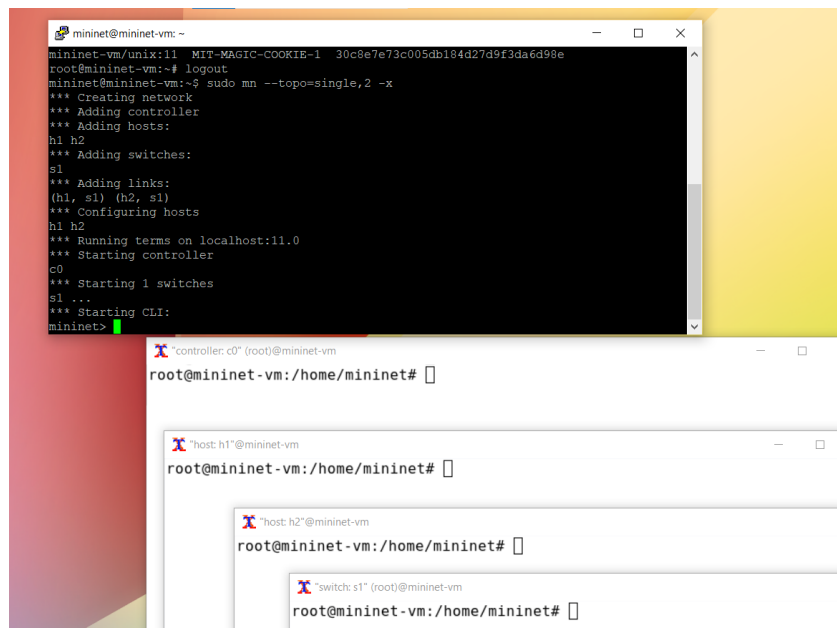


Рис. 3.2: Простейшая топология

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. 3.3).


```
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 82:42:2f:d3:18:6b 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 977 bytes 275008 (275.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 977 bytes 275008 (275.0 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 56:cf:4e:22:6a:61 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 955 bytes 297892 (297.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 955 bytes 297892 (297.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Рис. 3.3: ifconfig на хостах h1 и h2

Проверим подключение между хостами h1 и h2 с помощью команды ping с параметром -c 6 (рис. 3.4).

```
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=23.8 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.904 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.104 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.152 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.211 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.220 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5079ms
rtt min/avg/max/mdev = 0.104/4.234/23.814/8.760 ms
```

Рис. 3.4: Проверка подключения между хостами

3.2 Добавление/изменение задержки в эмулируемой глобальной сети

На хосте h1 добавим задержку в 100 мс к выходному интерфейсу. Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду

ping с параметром -с 6 с хоста h1 (рис. 3.5).

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=103 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=101 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 5008ms
rtt min/avg/max/mdev = 100.418/101.560/103.122/1.147 ms
```

Рис. 3.5: Добавление задержки в 100мс

В введенной команде:

- `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.

Для эмуляции глобальной сети с двунаправленной задержкой необходимо к соответствующему интерфейсу на хосте h2 также добавить задержку в 100 миллисекунд. 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 -с 6 на терминале хоста h1 (рис. 3.6).

```

root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# █

root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=206 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=204 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=202 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5011ms
rtt min/avg/max/mdev = 201.505/202.907/205.568/1.391 ms
root@mininet-vm:/home/mininet# █

```

Рис. 3.6: Двухнаправленная задержка соединения

3.3 Изменение задержки в эмулируемой глобальной сети

Изменим задержку со 100 мс до 50 мс для отправителя h1 и для получателя h2. Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду ping с параметром -c 6 с терминала хоста h1. (рис. 3.7).

```

root@mininet-vm:/home/mininet# sudo tc qdisc change dev h2-eth0 root netem delay 50ms
root@mininet-vm:/home/mininet# █

root@mininet-vm:/home/mininet# █
lay 50ms
1: command not found
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h1-eth0 root netem delay 50ms
Error: Qdisc not found. To create specify NLM_F_CREATE flag.
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h1-eth0 root netem delay 50ms
Error: Qdisc not found. To create specify NLM_F_CREATE flag.
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h1-eth0 root netem delay 50ms
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=210 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=102 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 5013ms
rtt min/avg/max/mdev = 101.434/120.245/209.959/40.126 ms

```

Рис. 3.7: Изменение задержки на 50мс

3.4 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 и для получателя h2. 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 (рис. 3.8).

```
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h2-eth0 root netem
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=102 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 5013ms
rtt min/avg/max/mdev = 101.434/120.245/209.959/40.126 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=3.60 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.57 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.352 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.159 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.151 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5057ms
rtt min/avg/max/mdev = 0.115/1.157/3.600/1.396 ms
```

Рис. 3.8: Восстановление исходных значений задержки

3.5 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 ± 10 мс, используя в терминале хоста h1 команду `ping` с параметром `-c 6`. Восстановим конфигурацию интерфейса по умолчанию на узле h1 (рис. 3.9). Увидим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.

```

root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 6
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=114 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=98.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=94.1 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=111 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=99.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=109 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5012ms
rtt min/avg/max/mdev = 94.050/104.215/113.511/7.245 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet# █

```

Рис. 3.9: Добавление значения дрожания задержки в интерфейс подключения

3.6 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 ± 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 ± 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(рис. 3.10).

```

root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms 25%
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 20
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=95.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=109 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=96.7 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=105 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=97.4 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=103 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=98.0 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=99.6 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=99.0 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=111 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=109 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=102 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=108 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=99.0 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=104 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=91.3 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=93.7 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19047ms
rtt min/avg/max/mdev = 91.287/101.498/111.086/5.185 ms
root@mininet-vm:/home/mininet# █

```

Рис. 3.10: Добавление значения корреляции для джиттера и задержки в интерфейс подключения

3.7 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 `-c 10`. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Завершим работу `mininet` в интерактивном режиме (рис. 3.11):

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 20ms distribution normal
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 10
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=79.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=116 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=61.3 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=108 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=109 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=98.7 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=115 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=130 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=96.1 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=114 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 902lms
rtt min/avg/max/mdev = 61.331/102.699/129.986/18.905 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet#
```

Рис. 3.11: Распределение задержки в интерфейсе подключения

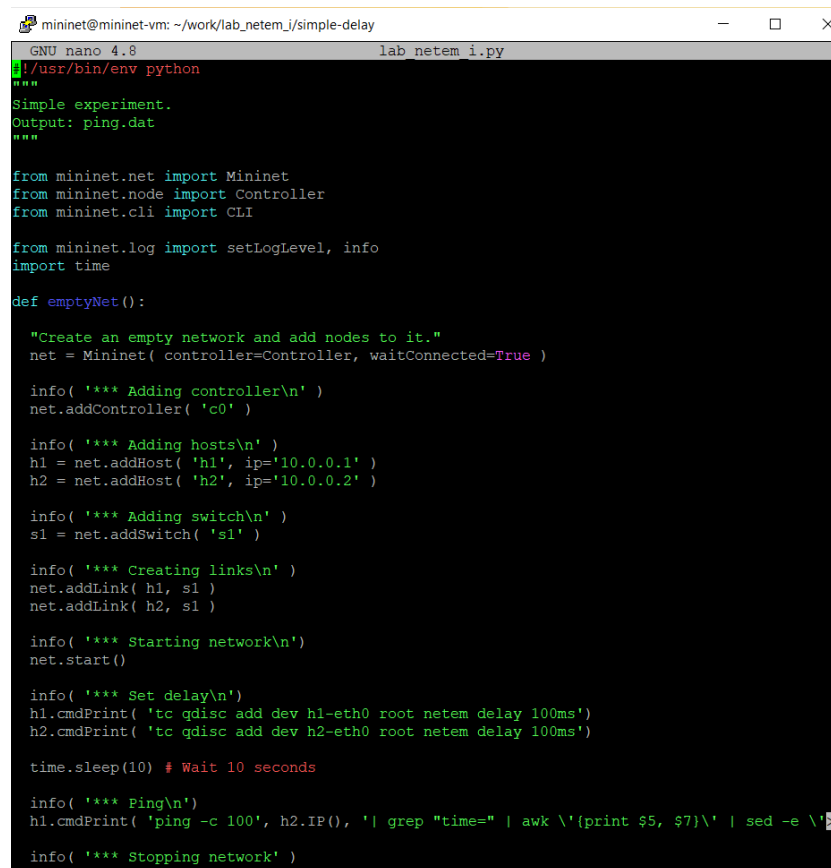
3.8 Воспроизведение экспериментов. Добавление задержки для интерфейса, подключающегося к эмулируемой глобальной сети

С помощью API Mininet воспроизведем эксперимент по добавлению задержки для интерфейса хоста, подключающегося к эмулируемой глобальной сети. В виртуальной среде `mininet` установим пакет `gheeie` (понадобится для просмотра файлов `png`), в своём рабочем каталоге с проектами создадим каталог `simple-delay` и перейдем в него (рис. 3.12).

```
mininet@mininet-vm:~$ mkdir -p ~/work/lab_netem_i/simple-delay
mininet@mininet-vm:~$ sudo apt install geqqie
Reading package lists... Done
Building dependency tree
Reading state information... Done
geqqie is already the newest version (1:1.5.1-8build1).
0 upgraded, 0 newly installed, 0 to remove and 374 not upgraded.
mininet@mininet-vm:~$
```

Рис. 3.12: Установка пакета и создание каталога

Создадим скрипт для эксперимента lab_netem_i.py (рис. 3.13).



```
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay
GNU nano 4.8 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 \'
    info( '*** Stopping network' )
```

Рис. 3.13: Скрипт на Python для эксперимента

В этом скрипте создается простейшая топология сети, затем с помощью команд, использованных нами ранее задается задержка в 100 мс для обоих хостов, после чего пингуется второй хост (100 сообщений отправляется), при этом из сообщений при пинге вытаскиваются номер сообщения и значение времени, которые записываются в файл с данными.

Создаём скрипт для визуализации ping_plot результатов эксперимента (рис. 3.14):

```
GNU nano 4.8 ping_plot Modified
#!/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
```

Рис. 3.14: Скрипт для визуализации ping_plot

Зададим права доступа к файлу скрипта: `chmod +x ping_plot`.

Создадим Makefile для управления процессом проведения эксперимента (рис. 3.15).

```
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
```

Рис. 3.15: Makefile для управления процессом проведения эксперимента

Выполним эксперимент, написав команду `make` (рис. 3.16).

```
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$ make
sudo python lab_netem_i.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Set delay
*** h1 : ('tc qdisc add dev h1-eth0 root netem delay 100ms',)
*** h2 : ('tc qdisc add dev h2-eth0 root netem delay 100ms',)
*** Ping
*** 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')
```

Рис. 3.16: Запуск эксперимента

Продemonстрируем построенный в результате выполнения скриптов график (рис. 3.17).

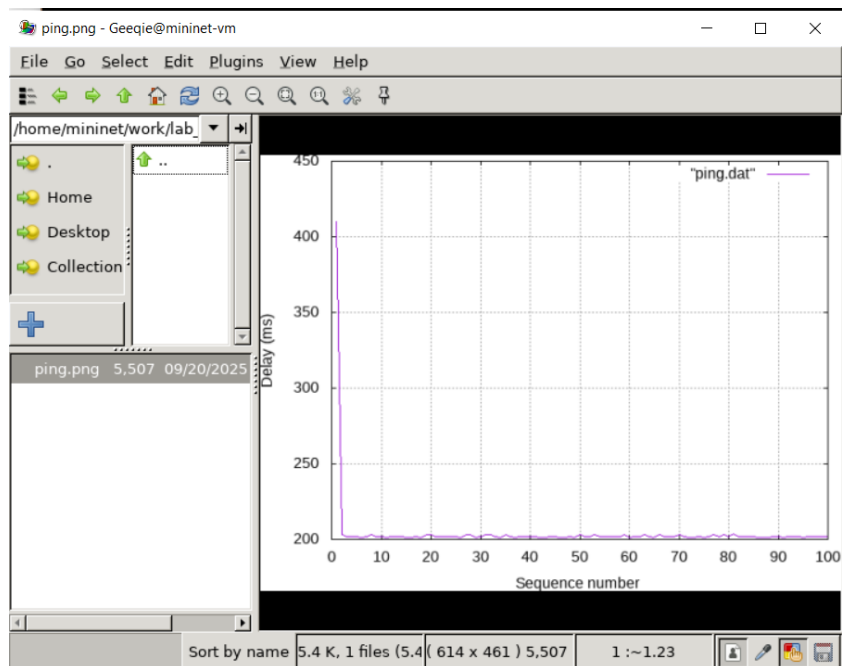


Рис. 3.17: Визуализация эксперимента

Из файла ping.dat удалим первую строку (рис. 3.18).

A screenshot of a terminal window titled 'mininet@mininet-vm: ~/work/lab_netem_i/simple-delay'. The terminal shows the output of the 'cat ping.dat' command. The output is a list of 11 lines, each containing a sequence number followed by a delay value in milliseconds. The first line is '203', and the subsequent lines are '202', '202', '202', '201', '202', '203', '202', '202', '202', '202', and '201'. The terminal prompt is 'GNU nano 4.8'.

Рис. 3.18: Удаление строки из файла .dat

Из файла ping.dat удалим первую строку и заново построим график (рис. 3.19).

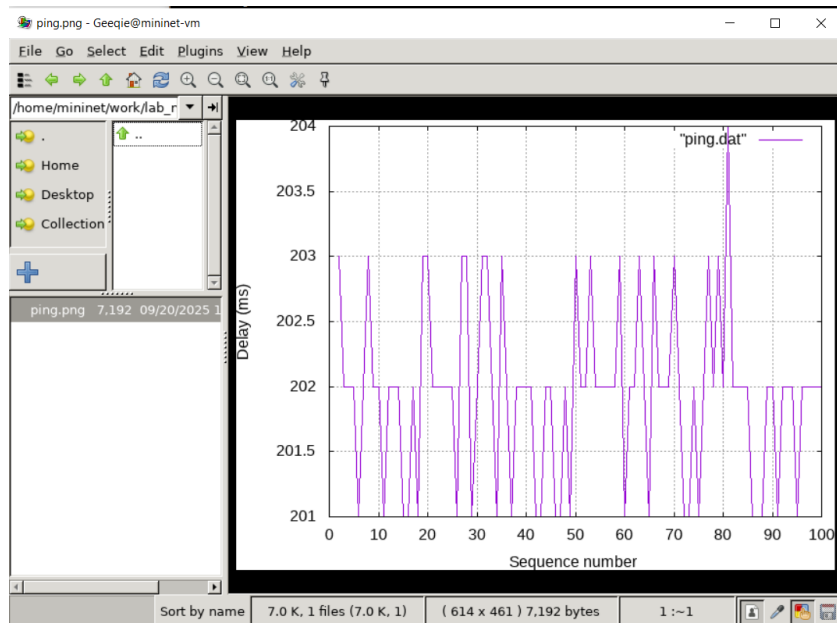


Рис. 3.19: Визуализация эксперимента

Разработаем скрипт для вычисления на основе данных файла ping.dat минимального, среднего, максимального и стандартного отклонения времени приёма-передачи (рис. fig. 3.20).

```
GNU nano 4.8 script_1.py
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([(i-(sum(s)/len(s)))**2 for i in s])/(len(s)-1))**0.5
    print(f"min: {min(s)} \nmax: {max(s)} \navg: {sum(s)/len(s)} \nstd: {std}")
```

Рис. 3.20: Скрипт script_1.py

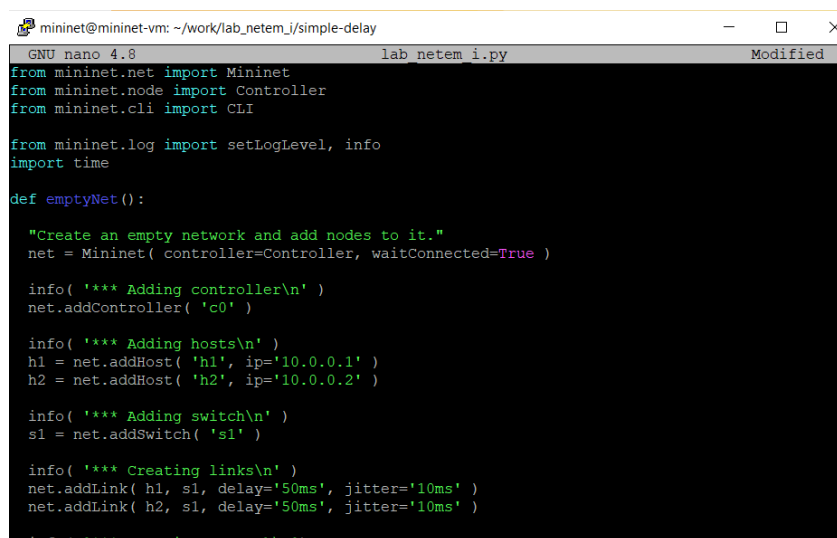
Продemonстрируем работу скрипта с выводом значений на экран или в отдельный файл (рис. fig. 3.21).

```
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$ sudo python script_1.py
min: 201
max: 204
avg: 201.94949494949495
std: 0.6757269309257353
```

Рис. 3.21: Результат работы скрипта script_1.py

3.9 Задание для самостоятельной работы

Самостоятельно реализуем воспроизводимые эксперименты по изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети (рис. fig. 3.22).



```
mininet@mininet-vm: ~/work/lab_netem_i/simple-delay
GNU nano 4.8 lab_netem_i.py Modified
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, delay='50ms', jitter='10ms' )
    net.addLink( h2, s1, delay='50ms', jitter='10ms' )

    info( '*** Starting network\n' )
```

Рис. 3.22: Изменение файла lab_netem_i.py

Продemonстрируем построенный в результате выполнения скриптов график (рис. 3.23).

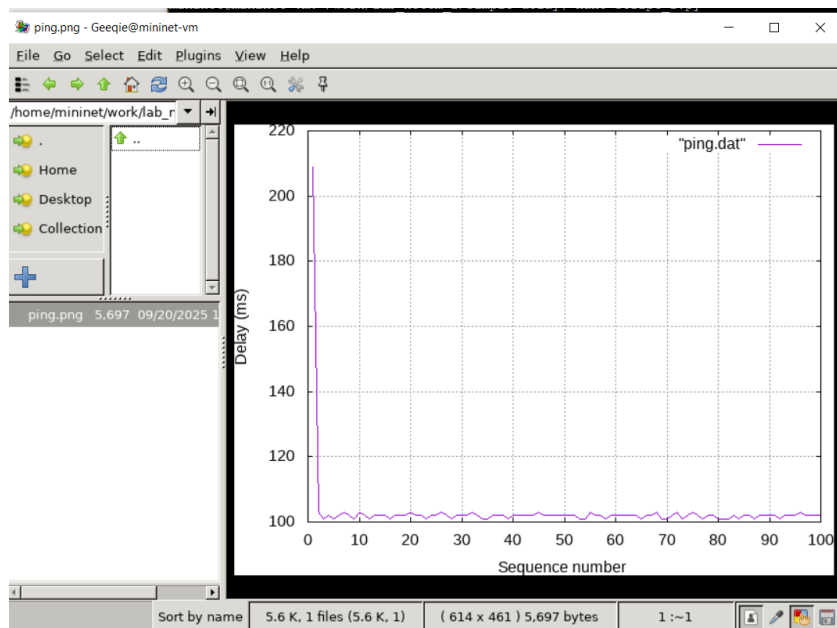


Рис. 3.23: Визуализация эксперимента

Вычислим минимальное, среднее, максимальное и стандартное отклонение времени приёма-передачи для каждого случая (рис. 3.24).

```
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$ sudo python script_1.py
min: 101
max: 209
avg: 102.95
std: 10.72839619181194
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$
```

Рис. 3.24: Результат работы скрипта script_1.py

4 Выводы

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