

Лабораторная работа № 4

Эмуляция и измерение задержек в глобальных сетях

Беличева Дарья Михайловна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Добавление/изменение задержки в эмулируемой глобальной сети	10
4.2	Изменение задержки в эмулируемой глобальной сети	12
4.3	Восстановление исходных значений (удаление правил) задержки в эмулируемой глобальной сети	12
4.4	Добавление значения дрожания задержки в интерфейс подключения к эмулируемой глобальной сети	13
4.5	Добавление значения корреляции для джиттера и задержки в интерфейс подключения к эмулируемой глобальной сети	14
4.6	Распределение задержки в интерфейсе подключения к эмулируемой глобальной сети	14
4.7	Воспроизведение экспериментов. Добавление задержки для интерфейса, подключающегося к эмулируемой глобальной сети	15
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Исправление прав запуска X-соединения	7
4.2	Простейшая топология	8
4.3	ifconfig на хостах h1 и h2	9
4.4	Проверка подключения между хостами	10
4.5	Добавление задержки в 100мс	11
4.6	Двунаправленная задержка соединения	11
4.7	Изменение задержки на 50мс	12
4.8	Восстановление исходных значений задержки	13
4.9	Добавление значения дрожания задержки в интерфейс подключения	13
4.10	Добавление значения корреляции для джиттера и задержки в ин- терфейс подключения	14
4.11	Распределение задержки в интерфейсе подключения	15
4.12	скрипт для визуализации ping_plot	17
4.13	Makefile для управления процессом проведения эксперимента . .	18
4.14	результате выполнения скриптов	18
4.15	результате выполнения скрипта	19
4.16	Скрипт rtt.py	19
4.17	Результат работы скрипта rtt.py	19

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[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.

4 Выполнение лабораторной работы

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:~$ xauth list $DISPLAY
mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 bba0fc98025a4656ec10bf5957fc1c6b
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 bba0fc98025a4656ec10bf5957fc1c6b
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:11 MIT-MAGIC-COOKIE-1 bba0fc98025a4656ec10bf5957fc1c6b
root@mininet-vm:~# logout
mininet@mininet-vm:~$
```

Рис. 4.1: Исправление прав запуска X-соединения

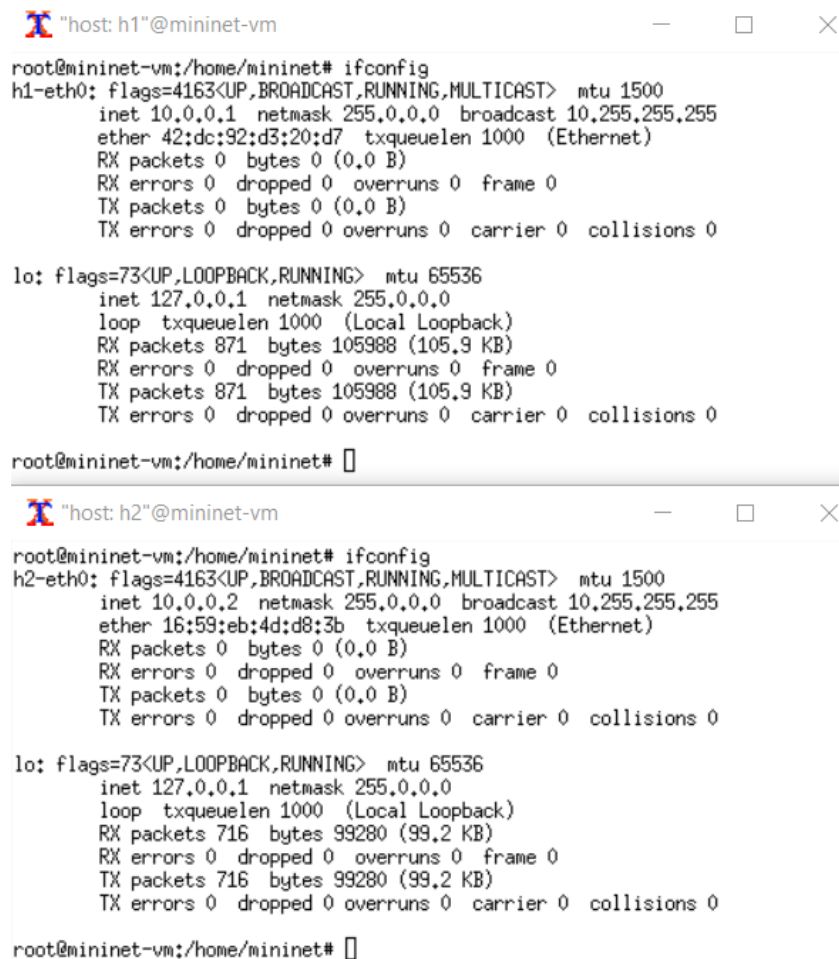
Задайте простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 4.2).

После введения этой команды запустятся терминалы двух хостов, коммутатора и контроллера. Терминалы коммутатора и контроллера можно закрыть.

```
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:11.0
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Рис. 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).



The image shows two terminal windows from a Mininet VM. The top window is titled "host: h1"@mininet-vm and shows the output of the 'ifconfig' command. It displays details for the h1-eth0 interface (10.0.0.1) and the loopback interface lo (127.0.0.1). The bottom window is titled "host: h2"@mininet-vm and shows the output of 'ifconfig' for host h2, displaying details for h2-eth0 (10.0.0.2) and lo (127.0.0.1).

```
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 42:dc:92:d3:20:d7 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 871 bytes 105988 (105.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 871 bytes 105988 (105.9 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 16:59:eb:4d:d8:3b 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 716 bytes 99280 (99.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 716 bytes 99280 (99.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

Рис. 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=2.89 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.168 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.157 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.113 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5100ms
rtt min/avg/max/mdev = 0.089/0.586/2.893/1.031 ms
root@mininet-vm:/home/mininet# █

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=4.42 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.611 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.104 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.091 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.094 ms

--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5083ms
rtt min/avg/max/mdev = 0.091/0.901/4.416/1.583 ms
root@mininet-vm:/home/mininet# █

```

Рис. 4.4: Проверка подключения между хостами

4.1 Добавление/изменение задержки в эмулируемой глобальной сети

На хосте h1 добавим задержку в 100 мс к выходному интерфейсу (рис. 4.5).

```
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

```

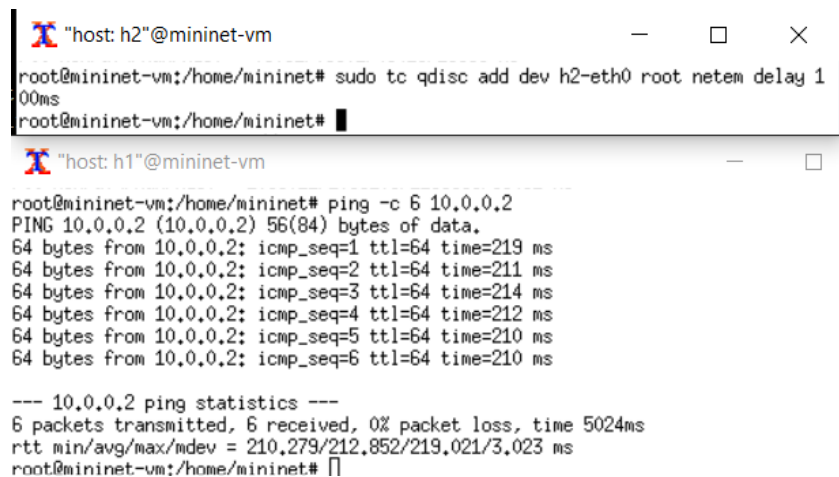
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 1
00ms
root@mininet-vm:/home/mininet# ping -c 6 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=116 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=107 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5038ms
rtt min/avg/max/mdev = 106.012/108.105/115.859/3.482 ms
root@mininet-vm:/home/mininet# █

```

Рис. 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.



```

"host: h2"@mininet-vm
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem delay 1
00ms
root@mininet-vm:/home/mininet# █

"host: h1"@mininet-vm
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=219 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=211 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=214 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=212 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=210 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=210 ms
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5024ms
rtt min/avg/max/mdev = 210.279/212.852/219.021/3.023 ms
root@mininet-vm:/home/mininet# █

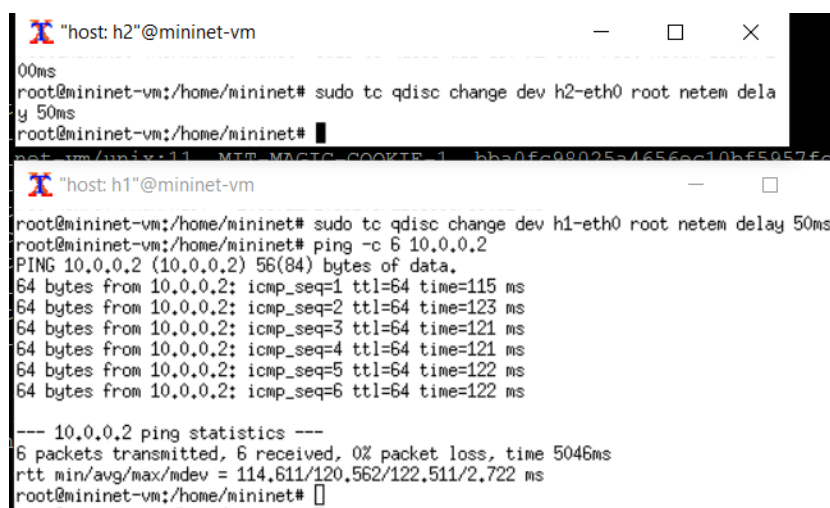
```

Рис. 4.6: Двунаправленная задержка соединения

4.2 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.



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

"host: h1"@mininet-vm
root@mininet-vm:/home/mininet# sudo tc qdisc change dev h1-eth0 root netem delay 50ms
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=115 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=123 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=121 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=121 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=122 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=122 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5046ms
rtt min/avg/max/mdev = 114.611/120.562/122.511/2.722 ms
root@mininet-vm:/home/mininet#
```

Рис. 4.7: Изменение задержки на 50мс

4.3 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 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. 4.8).

```
"host: h2"@mininet-vm
y 50ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h2-eth0 root netem
root@mininet-vm:/home/mininet#

"host: h1"@mininet-vm
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=1.80 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.963 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.299 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.117 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5077ms
rtt min/avg/max/mdev = 0.091/0.560/1.801/0.633 ms
root@mininet-vm:/home/mininet#
```

Рис. 4.8: Восстановление исходных значений задержки

4.4 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 ± 10 мс, используя в терминале хоста h1 команду ping с параметром -c 6. Восстановим конфигурацию интерфейса по умолчанию на узле h1 (рис. 4.9).

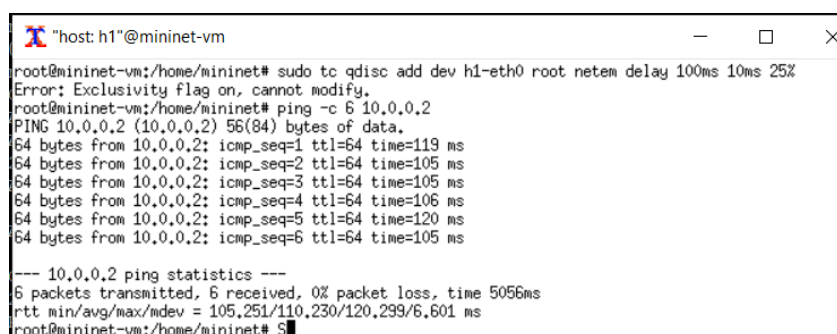
```
"host: h1"@mininet-vm
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms
root@mininet-vm:/home/mininet# ping -c 6 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=117 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=121 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=92.1 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5045ms
rtt min/avg/max/mdev = 92.057/108.532/121.221/9.314 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet#
```

Рис. 4.9: Добавление значения дрожания задержки в интерфейс подключения

4.5 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 ± 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. 4.10).



```
"host: h1"@mininet-vm
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms 25%
Error: Exclusivity flag on, cannot modify.
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=119 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=120 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=105 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5056ms
rtt min/avg/max/mdev = 105.251/110.230/120.299/6.601 ms
root@mininet-vm:/home/mininet# S
```

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

4.6 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 ± 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. 4.11).

```
"host: h1" @mininet-vm
rtt min/avg/max/mdev = 75,856/109,943/150,204/27,527 ms
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 20ms distribution normal
Error: Exclusivity flag on, cannot modify.
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=129 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=107 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=138 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=90,8 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=91,6 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=90,7 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=89,9 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9106ms
rtt min/avg/max/mdev = 89,893/105,350/138,031/15,730 ms
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:/home/mininet#
```

Рис. 4.11: Распределение задержки в интерфейсе подключения

4.7 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 в своём рабочем каталоге с проектами создадим каталог simple-delay и перейдем в него.

Создадим скрипт для эксперимента lab_netem_i.py:

```
1 #!/usr/bin/env python
2
3 """
4 Simple experiment.
5 Output: ping.dat
6 """
7
8 from mininet.net import Mininet
```

```

9 from mininet.node import Controller
10 from mininet.cli import CLI
11 from mininet.log import setLogLevel, info
12 import time
13
14 def emptyNet():
15
16     "Create an empty network and add nodes to it."
17
18     net = Mininet( controller=Controller,
19 ↪ waitConnected=True )
20
21     info( '*** Adding controller\n' )
22     net.addController( 'c0' )
23
24     info( '*** Adding hosts\n' )
25     h1 = net.addHost( 'h1', ip='10.0.0.1' )
26     h2 = net.addHost( 'h2', ip='10.0.0.2' )
27
28     info( '*** Adding switch\n' )
29     s1 = net.addSwitch( 's1' )
30
31     info( '*** Creating links\n' )
32     net.addLink( h1, s1 )
33     net.addLink( h2, s1 )
34
35     info( '*** Starting network\n' )
36     net.start()

```

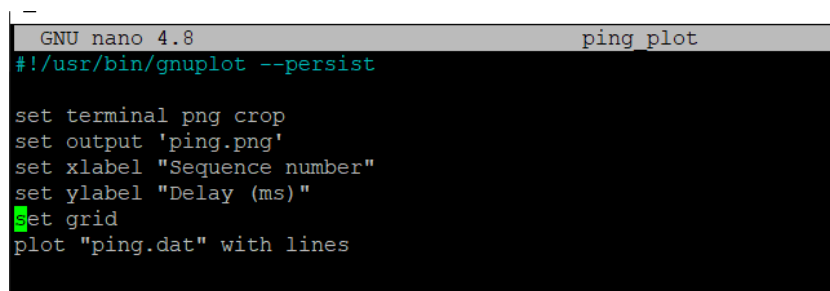


```

37 info( '*** Set delay\n')
38 h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem delay 100ms' )
39 h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem delay 100ms' )
40
41 time.sleep(10) # Wait 10 seconds
42
43 info( '*** Ping\n')
44 h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "time=" | awk \'{print $5, $7}\'' | sed
45
46 info( '*** Stopping network' )
47 net.stop()
48
49 if __name__ == '__main__':
50 setLogLevel( 'info' )
51 emptyNet()

```

Создаём скрипт для визуализации ping_plot результатов эксперимента (рис. 4.12).



```

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

```

Рис. 4.12: скрипт для визуализации ping_plot

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

Создадим Makefile для управления процессом проведения эксперимента (рис. 4.13).

```

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

```

Рис. 4.13: Makefile для управления процессом проведения эксперимента

Выполним эксперимент. Продемонстрируем построенный в результате выполнения скриптов график (рис. 4.14).

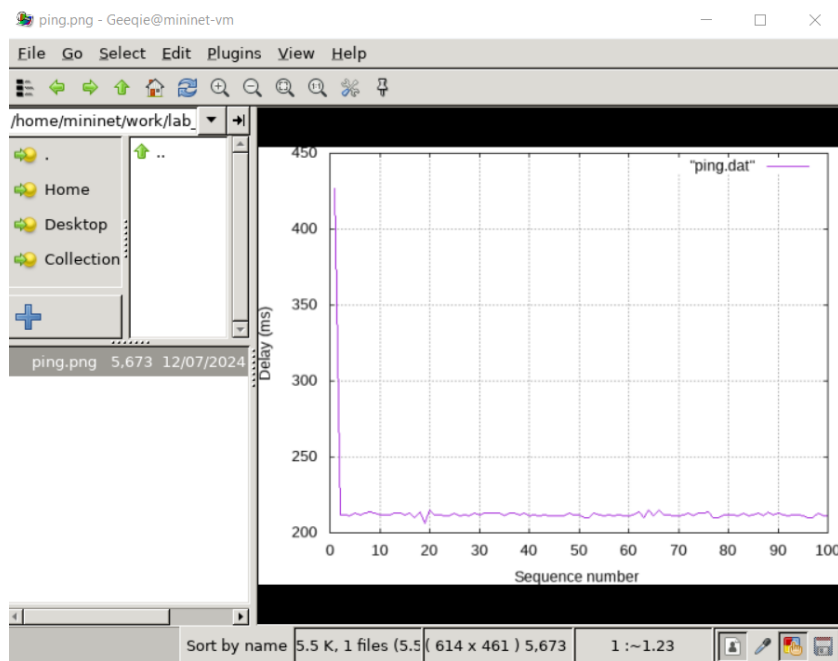


Рис. 4.14: результате выполнения скриптов

Из файла ping.dat удалим первую строку и заново постройте график. Продемонстрируем построенный в результате график (рис. 4.15).

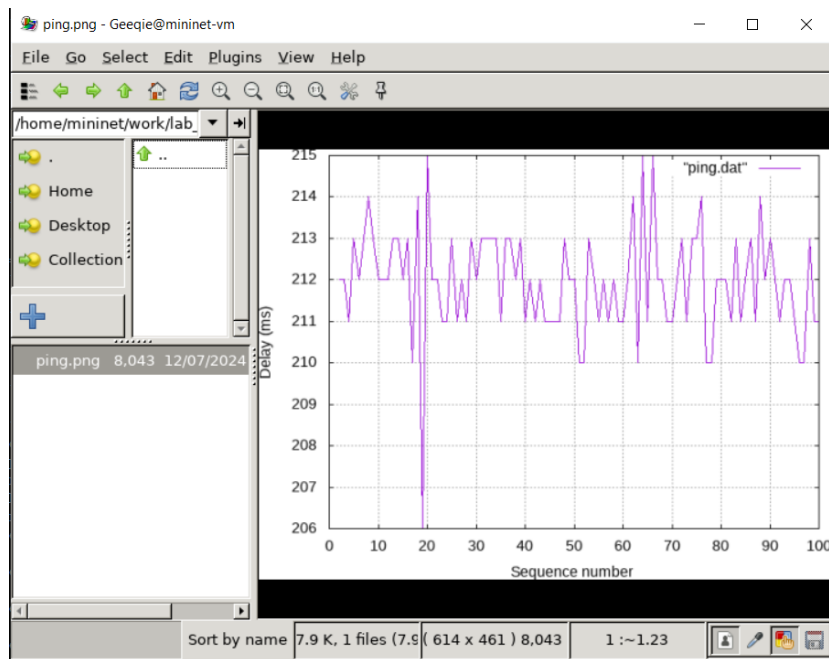


Рис. 4.15: результате выполнения скрипта

Разработайте скрипт для вычисления на основе данных файла ping.dat минимального, среднего, максимального и стандартного отклонения времени приёма-передачи (рис. 4.16).

```

GNU nano 4.8                                rtt.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))} \max: {(max(s))} \n avg: {(sum(s)/len(s))} \n std: {(std)}")

```

Рис. 4.16: Скрипт rtt.py

Продemonстрируем работу скрипта с выводом значений на экран или в отдельный файл (рис. 4.17).

```

mininet@mininet-vm:~/work/lab_netem_i/simple-delay$ sudo python rtt.py
min: 206
max: 215
avg: 211.93939393939394
std: 1.3000642199065626
mininet@mininet-vm:~/work/lab_netem_i/simple-delay$

```

Рис. 4.17: Результат работы скрипта rtt.py

5 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.

Список литературы

1. Mininet [Электронный ресурс]. Mininet Project Contributors. URL: <http://mininet.org/> (дата обращения: 17.11.2024).