

Лабораторная работа №5

Эмуляция и измерение потерь пакетов в глобальных сетях

Тазаева Анастасия Анатольевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Запуск лабораторной топологии	7
3.2	Интерактивные эксперименты	9
3.2.1	Добавление потери пакетов на интерфейс, подключённый к эмулируемой глобальной сети	9
3.2.2	Добавление значения корреляции для потери пакетов в эмулируемой глобальной сети	12
3.2.3	Добавление повреждения пакетов в эмулируемой глобальной сети	12
3.2.4	Добавление переупорядочивания пакетов в интерфейс подключения к эмулируемой глобальной сети	14
3.2.5	Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети	15
3.3	Воспроизведение экспериментов	16
3.3.1	Предварительная подготовка	16
3.3.2	Добавление потери пакетов на интерфейс, подключённый к эмулируемой глобальной сети	16
3.4	Самостоятельная работа	20
3.4.1	Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети	20
3.4.2	Добавление повреждения пакетов в эмулируемой глобальной сети	22
4	Выводы	26

Список иллюстраций

3.1	Подключение к mininet. Исправление прав запуска X-соединения	7
3.2	Создание топологии	8
3.3	ifconfig h1	8
3.4	ifconfig h2	8
3.5	Проверка подключения ping h1 -> h2	9
3.6	Потери пакетов для хоста h1	9
3.7	Потери пакетов для хоста h1. Проверка	10
3.8	Потери пакетов для хоста h1. Проверка. Результат	10
3.9	Потери пакетов для хоста h2	10
3.10	Потеря пакетов для двух хостов. Проверка	11
3.11	Удаление задержки для хоста h1. Проверка	11
3.12	Удаление задержки для хоста h2	11
3.13	Задержка для хоста h1 после удаления задержки. Результат	11
3.14	Значение корреляции для потери пакетов для хоста h1	12
3.15	Значение корреляции для потери пакетов для хоста h1. Проверка	12
3.16	Повреждение пакетов для хоста h1	13
3.17	Повреждение пакетов для хоста h1. Проверка. Запуск сервера	13
3.18	Повреждение пакетов для хоста h1. Проверка. Запуск клиента	13
3.19	Переупорядочивание пакетов для хоста h1	14
3.20	Переупорядочивание пакетов для хоста h1. Проверка	14
3.21	Дублирование пакетов для хоста h1	15
3.22	Дублирование пакетов для хоста h1. Проверка	15
3.23	Каталог для размещения файлов эксперимента	16
3.24	lab_netem_ii.py	18
3.25	Makefile	19
3.26	Make	20
3.27	Результат скрипта	20
3.28	Результат скрипта /dup/lab_netem_ii.py	22
3.29	Результат скрипта /dup/lab_netem_ii.py	25

Список таблиц

1 Цель работы

Освоить моделирование следующих параметров сети: потеря пакетов, дублирование пакетов, изменение порядка и повреждение пакетов.

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 исследованию параметров сети, связанных с потерей, дублированием, изменением порядка и повреждением пакетов при передаче данных.
3. Реализуйте воспроизводимый эксперимент по добавлению правила отбрасывания пакетов в эмулируемой глобальной сети. На экран выведите сводную информацию о потерянных пакетах.
4. Самостоятельно реализуйте воспроизводимые эксперименты по исследованию параметров сети, связанных с потерей, изменением порядка и повреждением пакетов при передаче данных. На экран выведите сводную информацию о потерянных пакетах.

3 Выполнение лабораторной работы

3.1 Запуск лабораторной топологии

1. Запустила виртуальную среду с mininet. Из основной ОС подключилась к виртуальной машине (рис. 3.1). Исправила права запуска X-соединения (рис. 3.1)

```
[aatazaeva@aatazaeva ~]$ ssh -Y mininet@192.168.56.101
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

Last login: Thu Dec  5 05:39:51 2024 from 192.168.56.1
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  2983e8a223c75f24435b1c7215b3ccea6
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  354e0b0b8205dd1dad6e57cd32216f64
root@mininet-vm:~# xauth add mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  2983e8a223c75f24435b1c7215b3ccea6
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  2983e8a223c75f24435b1c7215b3ccea6
root@mininet-vm:~# ^C
root@mininet-vm:~# exit
logout
```

Рис. 3.1: Подключение к mininet. Исправление прав запуска X-соединения

2. Создала простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 (рис. 3.2). Терминалы коммутатора и контроллера закрыла.

```
sudo mn --topo=single,2 -x
```

```

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:

```

Рис. 3.2: Создание топологии

3. На хостах h1 и h2 ввела команду ifconfig (рис. 3.3 и 3.4), 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 .

```

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 d6:83:e8:df:fe:a5 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

```

Рис. 3.3: ifconfig h1

```

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 ae:bd:12:08:6a:63 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

```

Рис. 3.4: ifconfig h2

4. Проверила подключение между хостами h1 и h2 с помощью команды ping с параметром -c 6 (рис. 3.5). На этом же рисунке выделила цветом значения

всеё возможных отклонений времени приёма-передачи.

```
root@mininet-vm:/home/mininet# ping 10.0.0.1 -c 6
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=2.91 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.047 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.053 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.052 ms

--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5089ms
rtt min/avg/max/mdev = 0.044/0.551/2.912/1.057 ms
```

Рис. 3.5: Проверка подключения ping h1 -> h2

3.2 Интерактивные эксперименты

3.2.1 Добавление потери пакетов на интерфейс, подключённый к эмулируемой глобальной сети

1. На хосте h1 добавила 10% потерь пакетов к интерфейсу h1-eth0 (рис. 3.6):

```
sudo tc qdisc add dev h1-eth0 root netem delay loss 10%
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem loss 10%
%
```

Рис. 3.6: Потери пакетов для хоста h1

Здесь: - 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; - loss 10% : 10% потерь пакетов.

2. Проверила, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 100 с хоста h1 (рис. 3.7). Некоторые номера оследовательности отсутствуют из-за потери пакетов

- выделены синим (рис. 3.7). В сводном отчёте ping сообщает о проценте потерянных пакетов после завершения передачи - 12% потерянных пакетов (рис. 3.8).

```
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 100
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.625 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.308 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.143 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.055 ms
```

Рис. 3.7: Потери пакетов для хоста h1. Проверка

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 88 received, 12% packet loss, time 101379ms
rtt min/avg/max/mdev = 0.042/0.059/0.625/0.067 ms
```

Рис. 3.8: Потери пакетов для хоста h1. Проверка. Результат

3. Для эмуляции глобальной сети с потерей пакетов в обоих направлениях необходимо к соответствующему интерфейсу на хосте h2 также добавить 10% потерь пакетов (рис. 3.9):

```
sudo tc qdisc add dev h2-eth0 root netem loss 10%
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h2-eth0 root netem loss 10%
%
```

Рис. 3.9: Потери пакетов для хоста h2

4. Проверила, что соединение между хостом h1 и хостом h2 имеет больший процент потерянных данных (10% от хоста h1 к хосту h2 и 10% от

хоста h2 к хосту h1), повторив команду ping с параметром -c 100 на терминале хоста h1 (рис. 3.10). Отсутствующие из-за потери пакетов номера последовательности (4, 13, 18, 24, 27, 28, 34, 36, 45, 59, 67, 72, 74, 77, 96, 100), процент потерянных пакетов после завершения передачи - 16%.

```
100 packets transmitted, 84 received, 16% packet loss, time 101373ms
rtt min/avg/max/mdev = 0.039/0.066/0.543/0.077 ms
```

Рис. 3.10: Потеря пакетов для двух хостов. Проверка

5. Восстановила конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса. Для отправителя h1 (рис. 3.11):

```
sudo tc qdisc del dev h1-eth0 root netem
```

Для получателя h2 (рис. 3.12):

```
sudo tc qdisc del dev h2-eth0 root netem
```

```
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 50
```

Рис. 3.11: Удаление задержки для хоста h1. Проверка

```
root@mininet-vm:/home/mininet# sudo tc qdisc del dev h2-eth0 root netem
```

Рис. 3.12: Удаление задержки для хоста h2

6. Проверила, что соединение между хостом h1 и хостом h2 не имеет явноустановленной задержки, используя команду ping с параметром -c 6 с терминала хоста h1 (рис. 3.13 и 3.12).

```
--- 10.0.0.2 ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 50165ms
rtt min/avg/max/mdev = 0.042/0.066/0.487/0.074 ms
```

Рис. 3.13: Задержка для хоста h1 после удаления задержки. Результат

3.2.2 Добавление значения корреляции для потери пакетов в эмулируемой глобальной сети

1. Добавила на интерфейсе узла h1 коэффициент потери пакетов 50% (такой высокий уровень потери пакетов маловероятен), и каждая последующая вероятность зависит на 50% от последней (рис. 3.14):

```
sudo tc qdisc add dev h1-eth0 root netem loss 50% 50%
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem loss 50  
% 50%  
root@mininet-vm:/home/mininet# ping 10.0.0.2 -c 50
```

Рис. 3.14: Значение корреляции для потери пакетов для хоста h1

2. Проверила, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 50 с хоста h1 . Отсутствующие из-за потери пакетов номера последовательности (17, 27, 29, 30, 31, 32, 33, 36, 42, 43, 44, 50), процент потерянных пакетов после завершения передачи - 24% (рис. 3.15).

```
--- 10.0.0.2 ping statistics ---  
50 packets transmitted, 38 received, 24% packet loss, time 50155ms  
rtt min/avg/max/mdev = 0.034/0.080/0.491/0.082 ms
```

Рис. 3.15: Значение корреляции для потери пакетов для хоста h1. Проверка

3. Восстановила для узла h1 конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса:

```
sudo tc qdisc del dev h1-eth0 root netem
```

3.2.3 Добавление повреждения пакетов в эмулируемой глобальной сети

1. Добавила на интерфейсе узла h1 0,01% повреждения пакетов (рис. 3.16):

```
sudo tc qdisc add dev h1-eth0 root netem corrupt 0.01%
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem corrupt 0.01%
```

Рис. 3.16: Повреждение пакетов для хоста h1

2. Проверила конфигурацию с помощью инструмента iPerf3 для проверки повторных передач. Для этого:

- запустила iPerf3 в режиме сервера в терминале хоста h2 (рис. 3.17):

```
iperf3 -s
```

- запустила iPerf3 в клиентском режиме в терминале хоста h1 (рис. 3.18):

```
iperf3 -c 10.0.0.2
```

```
"host: h2" (на mininet-vm)
root@mininet-vm:/home/mininet# iperf3 -s
```

Рис. 3.17: Повреждение пакетов для хоста h1. Проверка. Запуск сервера

```
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 53038 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer    Bitrate      Retr  Cwnd
[ 7]  0.00-1.00    sec  4.19 GBytes 35.9 Gbits/sec 26  1.96 MBytes
[ 7]  1.00-2.00    sec  4.20 GBytes 36.2 Gbits/sec 11  1.48 MBytes
[ 7]  2.00-3.00    sec  4.27 GBytes 36.7 Gbits/sec 11  1.34 MBytes
[ 7]  3.00-4.00    sec  4.28 GBytes 36.8 Gbits/sec 9   1.63 MBytes
[ 7]  4.00-5.00    sec  4.17 GBytes 35.8 Gbits/sec 15  1.12 MBytes
[ 7]  5.00-6.00    sec  4.29 GBytes 36.9 Gbits/sec 8   1.73 MBytes
[ 7]  6.00-7.00    sec  4.24 GBytes 36.4 Gbits/sec 9   1.52 MBytes
[ 7]  7.00-8.00    sec  4.11 GBytes 35.3 Gbits/sec 6   1.72 MBytes
[ 7]  8.00-9.00    sec  4.17 GBytes 35.8 Gbits/sec 7   1.48 MBytes
[ 7]  9.00-10.00   sec  4.26 GBytes 36.6 Gbits/sec 7   1.49 MBytes
- - - - -
[ ID] Interval      Transfer    Bitrate      Retr
[ 7]  0.00-10.00   sec  42.2 GBytes 36.2 Gbits/sec 109
[ 7]  0.00-10.00   sec  42.2 GBytes 36.2 Gbits/sec
iperf Done.
```

Рис. 3.18: Повреждение пакетов для хоста h1. Проверка. Запуск клиента

- общее количество повторно переданных пакетов указано в поле Retr (рис. 3.18).

3. Восстановила для узла h1 конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса.

3.2.4 Добавление переупорядочивания пакетов в интерфейс подключения к эмулируемой глобальной сети

1. Добавила на интерфейсе узла h1 следующее правило (рис. 3.19):

```
sudo tc qdisc add dev h1-eth0 root netem delay 10ms reorder 25% 50%
```

Здесь 25% пакетов (со значением корреляции 50%) будут отправлены немедленно, а остальные 75% будут задержаны на 10 мс.

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 10ms reorder 25% 50%
```

Рис. 3.19: Переупорядочивание пакетов для хоста h1

2. Проверила, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 20 с хоста h1 . Часть пакетов не имеют задержки, а последующие несколько пакетов будут иметь задержку около 10 миллисекунд, выделила цветом (рис. 3.20).

```
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=10.2 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=10.6 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=10.6 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=10.8 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=10.8 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19071ms
rtt min/avg/max/mdev = 0.045/9.642/10.910/3.201 ms
```

Рис. 3.20: Переупорядочивание пакетов для хоста h1. Проверка

3. Восстановила для узла h1 конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса.

3.2.5 Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети

1. Для интерфейса узла h1 задала правило с дублированием 50% пакетов (т.е. 50% пакетов должны быть получены дважды) (рис. 3.21):

```
sudo tc qdisc add dev h1-eth0 root netem duplicate 50%
```

```
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem duplicate 50%
```

Рис. 3.21: Дублирование пакетов для хоста h1

2. Проверила, что на соединении от хоста h1 к хосту h2 имеются дублированные пакеты, используя команду ping с параметром -c 20 с хоста h1 (рис. 3.22). Дубликаты пакетов помечаются как DUP! . Измеренная скорость дублирования пакетов будет приближаться к настроенной скорости по мере выполнения большего количества попыток.

```
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=0.656 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.351 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.135 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.407 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.054 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.054 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.054 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.051 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, +4 duplicates, 0% packet loss, time 19445ms
rtt min/avg/max/mdev = 0.044/0.106/0.656/0.146 ms
```

Рис. 3.22: Дублирование пакетов для хоста h1. Проверка

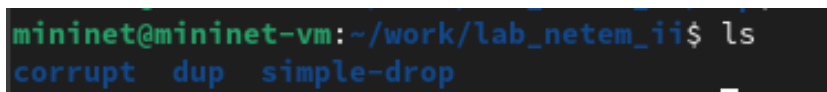
3. Восстановила для узла h1 конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса.

3.3 Воспроизведение экспериментов

3.3.1 Предварительная подготовка

1. Для каждого воспроизводимого эксперимента `exrname` создала свой каталог, в котором будут размещаться файлы эксперимента (рис. 3.23):

```
mkdir -p ~/work/lab_netem_ii/exrname
```



```
mininet@mininet-vm:~/work/lab_netem_ii$ ls
corrupt  dup  simple-drop
```

Рис. 3.23: Каталог для размещения файлов эксперимента

3.3.2 Добавление потери пакетов на интерфейс, подключённый к эмулируемой глобальной сети

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

1. В виртуальной среде mininet в своём рабочем каталоге с проектами создала каталог `simple-drop` и перешла в него. Создала скрипт для эксперимента `lab_netem_ii.py` (рис. 3.24). Скорректировала скрипт так, чтобы в отдельный файл выводилась информация о потерях пакетов.

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

    info('***set loss\n' )

```

```

h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem loss 10%' )
h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem loss 10%' )

time.sleep(10)

info('***ping\n' )
h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "packet loss" | awk \'{print $6, $7,

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

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

```

```

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' )
    net.start()

    info('***set loss\n' )
    h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem loss 10%' )
    h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem loss 10%' )

    time.sleep(10)

    info('***ping\n' )
    h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "packet loss" | awk \'{print $6, $7, $8}\'' > ping.dat' )

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

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

```

Рис. 3.24: lab_netem_ii.py

-В каких строках скрипта задается значение потери пакетов для интерфейса хоста?

```
h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem loss 10%' )  
h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem loss 10%' )
```

2. Создала Makefile для управления процессом проведения эксперимента (рис. 3.25).

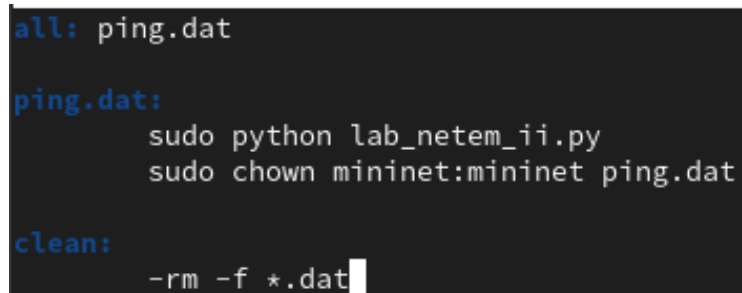
```
all: ping.dat
```

```
ping.dat:
```

```
    sudo python lab_netem_ii.py  
    sudo chown mininet:mininet ping.dat
```

```
clean:
```

```
    -rm -f *.dat
```

A screenshot of a terminal window with a dark background. It displays the content of a Makefile. The text is as follows:

```
all: ping.dat  
  
ping.dat:  
    sudo python lab_netem_ii.py  
    sudo chown mininet:mininet ping.dat  
  
clean:  
    -rm -f *.dat
```

Рис. 3.25: Makefile

3. Выполнила эксперимент (рис. 3.26).

```
make
```

```

mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$ make
sudo python lab_netem_ii.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 loss
*** h1 : ('tc qdisc add dev h1-eth0 root netem loss 10%,')
*** h2 : ('tc qdisc add dev h2-eth0 root netem loss 10%,')
***ping
*** h1 : ('ping -c 100', '10.0.0.2', '| grep "packet loss" | awk \'{print $6, $7, $8}\'} > ping.dat')
***stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat

```

Рис. 3.26: Make

4. В файл ping.dat вывелась информация о потерянных пакетах в % (рис. 3.27).

```

mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$ cat ping.dat
17% packet loss,

```

Рис. 3.27: Результат скрипта

5. Очистила каталог от результатов проведения экспериментов:

make clean

3.4 Самостоятельная работа

3.4.1 Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети

1. Создала скрипт /dup/lab_netem_ii.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' )
net.start()

info('***set loss\n' )
h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem duplicate 50%' )

time.sleep(10)

info('***ping\n' )
h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "duplicates" | awk \'{print "for ", $

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

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

```

2. Результат скрипта выводится в файл /dup/ping.dat (рис. 3.28).

```
1|for 100 packets - +36 duplacates
```

Рис. 3.28: Результат скрипта /dup/lab_netem_ii.py

3.4.2 Добавление повреждения пакетов в эмулируемой глобальной сети

1. Создала скрипт /corrupt/lab_netem_ii.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' )
net.start()

info('***set loss\n' )
h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem corrupt 0.01%' )
time.sleep(10)

info('***ping\n' )
h2.cmdPrint( 'iperf3 -s -D -1' )
time.sleep(10)
h1.cmdPrint( 'iperf3 -c ', h2.IP(), ' -J > res.json' )

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

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

```

2. Результат скрипта выводится в файл `/corrupt/res.json`. Также получены графики, расположенные в каталоге `/dup/results` (рис. 3.29).

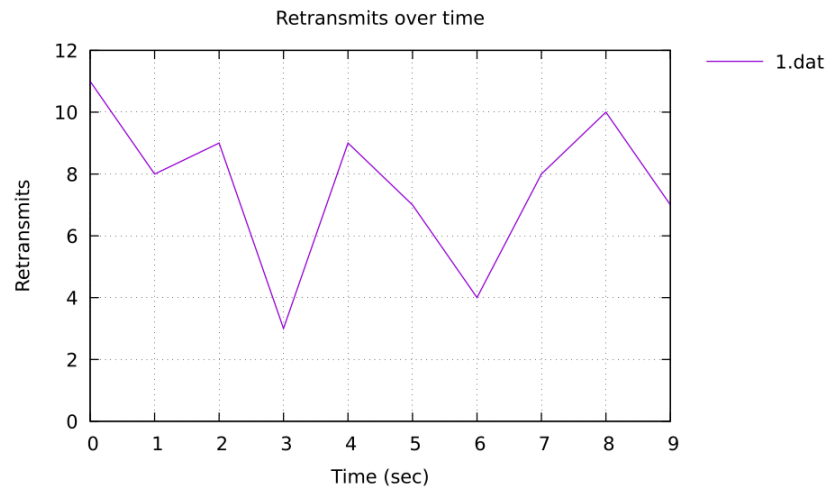


Рис. 3.29: Результат скрипта /dup/lab_netem_ii.py

4 Выводы

В ходе лабораторной работы мною было освоено моделирование следующих параметров сети: потеря пакетов, дублирование пакетов, изменение порядка и повреждение пакетов.