

Лабораторная работа №5

Эмуляция и измерение потерь пакетов в глобальных сетях

Хрусталеv Влад Николаевич

Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Задание	6
4	Выполнение лабораторной работы	7
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Исправление прав запуска X-соединения в виртуальной машине mininet	7
4.2	Информацию о сетевых интерфейсах и IP-адресах хостов	8
4.3	Проверка подключения между хостами h1 и h2	9
4.4	Добавление 10% потерь пакетов на хосте h1	10
4.5	Проверка потерь пакетов 1	10
4.6	Проверка потерь пакетов 2	11
4.7	Восстановление конфигурации по умолчанию для хоста h1 и хоста h2 и проверка	12
4.8	Добавление значения корреляции для потери пакетов и тест	13
4.9	Добавление на узле h1 0.01% повреждения пакетов и проверка через iperf	14
4.10	Добавление переупорядочивания пакетов и тест	15
4.11	Добавление дублирования пакетов и тест	16
4.12	Листинг lab_netem_ii для simple-drop из лабораторно	17
4.13	Листинг lab_netem_ii для simple-drop с анализом	18
4.14	Листинг Makefile для simple-drop	18
4.15	Выполнение эксперимента и последующая очистка каталога	19
4.16	Создание каталогов для самостоятельной работы	19
4.17	Листинг программы для эксперимента по добавлению потери и коэффициента корреляции	20
4.18	Выполнение эксперимента по добавлению потери и коэффициента корреляции	21
4.19	Листинг программы для эксперимента по повреждению пакетов	21
4.20	Выполнение эксперимента по повреждению пакетов	22
4.21	Листинг программы для эксперимента по изменению порядка пакетов	22
4.22	Выполнение эксперимента по изменению порядка пакетов	23
4.23	Листинг программы для эксперимента по дублированию пакетов	23
4.24	Выполнение эксперимента по дублированию пакетов	24

1 Цель работы

Основной целью работы является получение навыков проведения интерактивных экспериментов в среде Mininet по исследованию параметров сети, связанных с потерей, дублированием, изменением порядка и повреждением пакетов при передаче данных. Эти параметры влияют на производительность протоколов и сетей.

2 Теоретическое введение

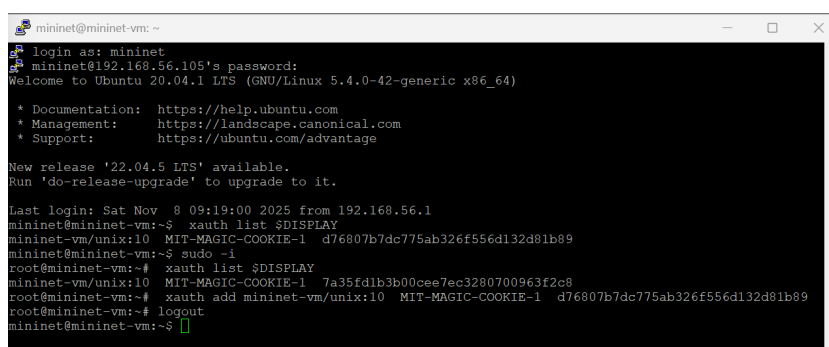
Mininet[1] – это эмулятор компьютерной сети. Под компьютерной сетью подразумеваются простые компьютеры — хосты, коммутаторы, а так же OpenFlow-контроллеры. С помощью простейшего синтаксиса в примитивном интерпретаторе команд можно разворачивать сети из произвольного количества хостов, коммутаторов в 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.

3 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.

4 Выполнение лабораторной работы

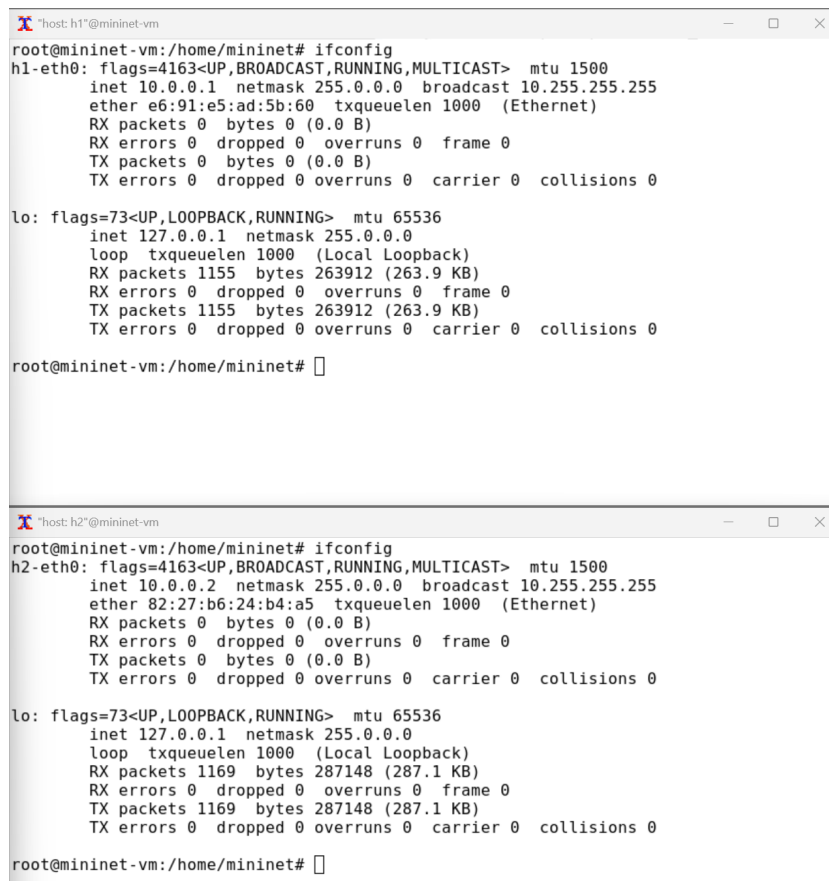
В виртуальной машине mininet исправим права запуска X-соединения (рис. 4.1).



```
mininet@mininet-vm: ~  
$ login as: mininet  
$ mininet@192.168.56.105'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: Sat Nov  8 09:19:00 2025 from 192.168.56.1  
mininet@mininet-vm:~$ xauth list $DISPLAY  
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  d76807b7dc775ab326f556d132d81b89  
mininet@mininet-vm:~$ sudo -i  
root@mininet-vm:~# xauth list $DISPLAY  
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  7a35fd1b3b00cee7ec3280700963f2c8  
root@mininet-vm:~# xauth add mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  d76807b7dc775ab326f556d132d81b89  
root@mininet-vm:~# logout  
mininet@mininet-vm:~$
```

Рис. 4.1: Исправление прав запуска X-соединения в виртуальной машине mininet

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



The image displays two terminal windows from a Mininet virtual network environment. The top window is for host h1, and the bottom window is for host h2. Both windows show the output of the 'ifconfig' command, detailing the configuration of the Ethernet interface (eth0) and the loopback interface (lo).

Host h1 Configuration:

```
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 e6:91:e5:ad:5b:60 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 1155 bytes 263912 (263.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1155 bytes 263912 (263.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

Host h2 Configuration:

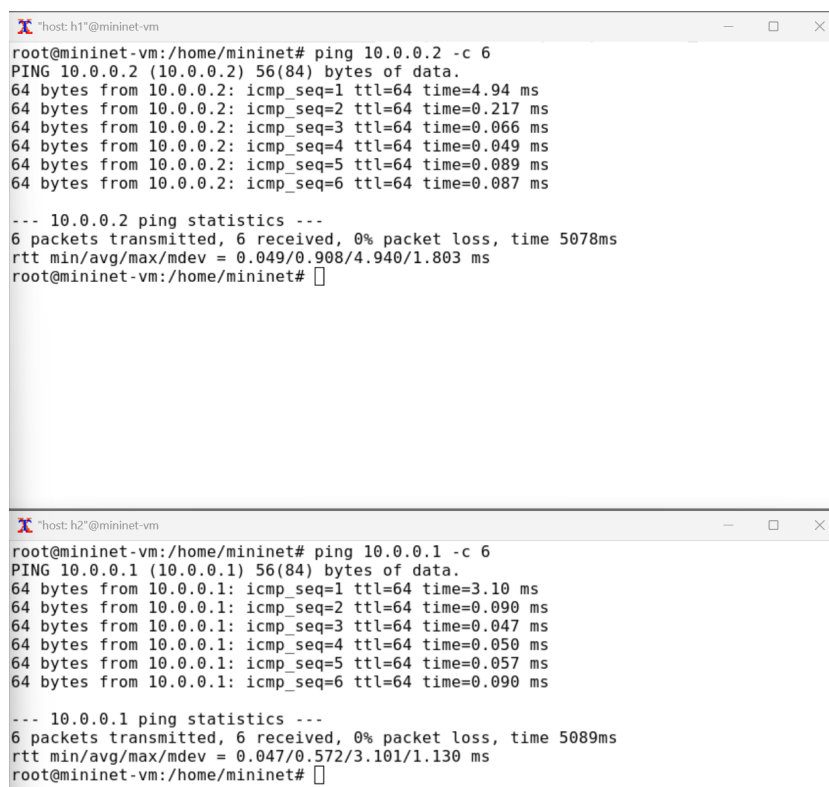
```
root@mininet-vm:/home/mininet# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 82:27:b6:24:b4:a5 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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 1169 bytes 287148 (287.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1169 bytes 287148 (287.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

Рис. 4.2: Информацию о сетевых интерфейсах и IP-адресах хостов

Проверим подключение между хостами h1 и h2 с помощью команды ping с параметром -c 6 (рис. 4.3).



The image shows two terminal windows from a Mininet VM. The top window, titled "host: h1", shows a ping command from 10.0.0.2 to 10.0.0.2. It displays six successful ping responses with times ranging from 0.066 ms to 4.94 ms, and a summary showing 0% packet loss and a total time of 5078ms. The bottom window, titled "host: h2", shows a ping command from 10.0.0.1 to 10.0.0.1. It displays six successful ping responses with times ranging from 0.050 ms to 3.10 ms, and a summary showing 0% packet loss and a total time of 5089ms.

```
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=4.94 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.217 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.087 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5078ms
rtt min/avg/max/mdev = 0.049/0.908/4.940/1.803 ms
root@mininet-vm:/home/mininet#

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=3.10 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.090 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=0.050 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.090 ms

--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5089ms
rtt min/avg/max/mdev = 0.047/0.572/3.101/1.130 ms
root@mininet-vm:/home/mininet#
```

Рис. 4.3: Проверка подключения между хостами h1 и h2

Пакеты могут быть потеряны в процессе передачи из-за таких факторов, как битовые ошибки и перегрузка сети. Скорость потери данных часто измеряется как процентная доля потерянных пакетов по отношению к количеству отправленных пакетов. На хосте h1 добавим 10% потерь пакетов к интерфейсу h1-eth0 (рис. 4.4):

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

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.

```

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

```

Рис. 4.4: Добавление 10% потерь пакетов на хосте h1

Проверим, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 100 с хоста h1.(рис. 4.5).

```

"host: h1"@mininet-vm
64 bytes from 10.0.0.2: icmp_seq=80 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.079 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.077 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 91 received, 9% packet loss, time 101335ms
rtt min/avg/max/mdev = 0.049/0.107/2.993/0.309 ms
root@mininet-vm:/home/mininet#

```

Рис. 4.5: Проверка потерь пакетов 1

Для эмуляции глобальной сети с потерей пакетов в обоих направлениях необходимо к соответствующему интерфейсу на хосте h2 также добавить 10% потерь пакетов. Проверим, 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. 4.6).

```
host: h1@mininet-vm
64 bytes from 10.0.0.2: icmp_seq=79 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=80 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=0.062 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 80 received, 20% packet loss, time 101335ms
rtt min/avg/max/mdev = 0.051/0.106/2.294/0.251 ms
root@mininet-vm:/home/mininet#

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

Рис. 4.6: Проверка потерь пакетов 2

Пропущенны номера `icmp_seq` (возможны значения от 1 до 100): [2, 9, 14, 19, 22, 25, 31, 33, 42, 44, 49, 55, 59, 67, 69, 73, 78, 82, 85, 91].

Восстановим конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса. Убедимся, что соединение от хоста `h1` к хосту `h2` не имеет явной потери пакетов, запустив команду `ping` с терминала хоста `h1` и затем нажав `Ctrl + c`, чтобы остановить тест(рис. 4.7).

```
"host: h1"@mininet-vm
command 'sudo' from deb sudo (1.8.31-lubuntu1.5)
command 'sudo' from deb sudo-ldap (1.8.31-lubuntu1.5)

Try: apt install <deb name>

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 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=3.12 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.732 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.282 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.060 ms
^C
--- 10.0.0.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10220ms
rtt min/avg/max/mdev = 0.052/0.429/3.116/0.870 ms
root@mininet-vm:/home/mininet# █

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

Рис. 4.7: Восстановление конфигурации по умолчанию для хоста h1 и хоста h2 и проверка

Добавим на интерфейсе узла h1 коэффициент потери пакетов 50% (такой высокий уровень потери пакетов маловероятен), и каждая последующая вероятность зависит на 50% от последней: Проверим, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду `ping` с параметром `-c 50` с хоста h1(рис. 4.8).

```
host: h1@mininet-vm
root@mininet-vm:/home/mininet#
root@mininet-vm:/home/mininet#
root@mininet-vm:/home/mininet#
root@mininet-vm:/home/mininet#
root@mininet-vm:/home/mininet#
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
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=2.82 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.432 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.190 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=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.077 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.077 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=26 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=42 ttl=64 time=0.206 ms
64 bytes from 10.0.0.2: icmp_seq=47 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=50 ttl=64 time=0.070 ms

--- 10.0.0.2 ping statistics ---
50 packets transmitted, 19 received, 62% packet loss, time 50146ms
rtt min/avg/max/mdev = 0.053/0.251/2.815/0.610 ms
root@mininet-vm:/home/mininet#
```

Рис. 4.8: Добавление значения корреляции для потери пакетов и тест

Восстановим конфигурацию интерфейса по умолчанию на узле h1. Добавим на интерфейсе узла h1 0,01% повреждения пакетов. Проверим конфигурацию с помощью инструмента iPerf3 для проверки повторных передач (рис. 4.9).

```

"host: h1"@mininet-vm
root@mininet-vm:/home/mininet#
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem corrupt 0.0
1%
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
iperf3: error - unable to connect to server: Connection refused
root@mininet-vm:/home/mininet# iperf3 -c 10.0.0.2
iperf3: error - unable to connect to server: Connection refused
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 35450 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer      Bitrate      Retr  Cwnd
[ 7] 0.00-1.01 sec  1.16 GBytes  9.86 Gbits/sec  1    5.69 MBytes
[ 7] 1.01-2.01 sec  1.30 GBytes  11.2 Gbits/sec  2    2.92 MBytes
[ 7] 2.01-3.01 sec  1.29 GBytes  11.2 Gbits/sec  4    1.38 MBytes
[ 7] 3.01-4.00 sec  1.88 GBytes  16.3 Gbits/sec  1    2.36 MBytes
[ 7] 4.00-5.00 sec  1.45 GBytes  12.4 Gbits/sec  3    1.19 MBytes
[ 7] 5.00-6.00 sec  1.16 GBytes  10.0 Gbits/sec  1    1.17 MBytes
[ 7] 6.00-7.00 sec  1.20 GBytes  10.3 Gbits/sec  1    1.31 MBytes
[ 7] 7.00-8.00 sec  1.30 GBytes  11.2 Gbits/sec  6    831 KBytes
[ 7] 8.00-9.00 sec  1.27 GBytes  10.9 Gbits/sec  4    1.02 MBytes
[ 7] 9.00-10.01 sec 1.48 GBytes  12.6 Gbits/sec  3    1.36 MBytes
-----
[ ID] Interval      Transfer      Bitrate      Retr
[ 7] 0.00-10.01 sec 13.5 GBytes  11.6 Gbits/sec  26
[ 7] 0.00-10.01 sec 13.5 GBytes  11.6 Gbits/sec
sender
receiver

iperf Done.
root@mininet-vm:/home/mininet#
"host: h2"@mininet-vm
Accepted connection from 10.0.0.1, port 35448
[ 7] local 10.0.0.2 port 5201 connected to 10.0.0.1 port 35450
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-1.00 sec  1.15 GBytes  9.86 Gbits/sec
[ 7] 1.00-2.01 sec  1.30 GBytes  11.1 Gbits/sec
[ 7] 2.01-3.00 sec  1.30 GBytes  11.3 Gbits/sec
[ 7] 3.00-4.00 sec  1.89 GBytes  16.2 Gbits/sec
[ 7] 4.00-5.00 sec  1.44 GBytes  12.4 Gbits/sec
[ 7] 5.00-6.01 sec  1.15 GBytes  9.79 Gbits/sec
[ 7] 6.01-7.00 sec  1.23 GBytes  10.6 Gbits/sec
[ 7] 7.00-8.00 sec  1.28 GBytes  11.0 Gbits/sec
[ 7] 8.00-9.00 sec  1.28 GBytes  11.1 Gbits/sec
[ 7] 9.00-10.00 sec 1.47 GBytes  12.6 Gbits/sec
[ 7] 10.00-10.01 sec 11.1 MBytes  7.27 Gbits/sec
-----
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-10.01 sec 13.5 GBytes  11.6 Gbits/sec
receiver

Server listening on 5201
-----
^Ciperf3: interrupt - the server has terminated
root@mininet-vm:/home/mininet#

```

Рис. 4.9: Добавление на узле h1 0.01% повреждения пакетов и проверка через iperf

Восстановим конфигурацию интерфейса по умолчанию на узле h1. Добавим на интерфейсе узла h1 следующее правило: 25% пакетов (со значением корреляции 50%) будут отправлены немедленно, а остальные 75% будут задержаны на 10 мс. Проверим, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 20 с хоста h1. Убедимся, что часть пакетов не будут иметь задержки (один из четырех, или 25%), а последующие несколько пакетов будут иметь задержку около 10 миллисекунд (три из четырех, или 75%)(рис. 4.10).

```

root@mininet-vm:/home/mininet# sudo tc qdisc del dev h1-eth0 root netem
Error: Invalid qdisc name.
root@mininet-vm:/home/mininet# sudo tc qdisc add dev h1-eth0 root netem delay 10ms
reorder 25% 50%
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=11.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.13 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.241 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=10.2 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=0.152 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=11.0 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=10.5 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=10.3 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=10.6 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=11.1 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=10.4 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=37.0 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=11.2 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=10.6 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.075 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19074ms
rtt min/avg/max/mdev = 0.075/9.470/37.048/7.760 ms
root@mininet-vm:/home/mininet#

```

Рис. 4.10: Добавление переупорядочивания пакетов и тест

Восстановим конфигурацию интерфейса по умолчанию на узле h1. Для интерфейса узла h1 зададим правило с дублированием 50% пакетов (т.е. 50% пакетов должны быть получены дважды): Проверим, что на соединении от хоста h1 к хосту h2 имеются дублированные пакеты, используя команду ping с параметром -с 20 с хоста h1. Дубликаты пакетов помечаются как DUP!(рис. 4.11).

```
host: h1" @mininet-vm
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 duplicate 5
0%
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=4.73 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.19 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.485 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=3 ttl=64 time=0.776 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.055 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.073 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.060 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.069 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.065 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.067 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.065 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, +7 duplicates, 0% packet loss, time 19436ms
rtt min/avg/max/mdev = 0.049/0.516/6.193/1.421 ms
root@mininet-vm:/home/mininet#
```

Рис. 4.11: Добавление дублирования пакетов и тест

Для каждого воспроизводимого эксперимента exрname создадим свой каталог, в котором будут размещаться файлы эксперимента. В виртуальной среде mininet в своём рабочем каталоге с проектами создадим каталог simple-drop и перейдём в него. Создадим скрипт для эксперимента lab_netem_ii.py и внесём в него листинг программы(рис. 4.12).


```
mininet@mininet-vm: ~/work/lab_netem_i/simple-drop
GNU nano 4.8 lab_netem_i.py Modified
#!/usr/bin/env python

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():

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

    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%' )

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

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

Рис. 4.12: Листинг lab_netem_i для simple-drop из лабораторно

Скорректируем скрипт так, чтобы на экран или в отдельный файл выводилась информация о потерях пакетов(рис. 4.13).

```

mininet@mininet-vm: ~/work/lab_netem_i/simple-drop
GNU nano 4.8 lab_netem ii.py Modified
#!/usr/bin/env python

from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info
import time

def analyze_ping_data(file_path='ping.dat', total_packets=100):
    if total_packets <= 0:
        print("ERROR")
        return

    received_packets = set()

    with open(file_path, 'r') as f:
        for line in f:
            parts = line.split()
            if parts:
                try:
                    packet_n = int(parts[0])
                    received_packets.add(packet_n)
                except ValueError:
                    pass

    lost_packets = {i for i in range(1, total_packets + 1) if i not in received_packets}
    lost_packets_count = len(lost_packets)
    loss_percent = (lost_packets_count / total_packets) * 100

    print(f'Total packets: {total_packets}')
    print(f'Received packets count: {len(received_packets)}')
    print(f'Lost packets count: {lost_packets_count}')
    print(f'Lost packets: {sorted(lost_packets)}')
    print(f'Loss packets percent: {loss_percent:.2f}%')

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

    time.sleep(10) # Wait 10 seconds

    info( '*** Ping\n' )

```

Рис. 4.13: Листинг lab_netem_ii для simple-drop с анализом

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

```

mininet@mininet-vm: ~/work/lab_netem_i/simple-drop
GNU nano 4.8 Makefile
all: ping.dat

ping.dat:
    sudo python lab_netem ii.py
    sudo chown mininet:mininet ping.dat

clean:
    -rm -f *.dat

```

Рис. 4.14: Листинг Makefile для simple-drop

Выполним эксперимент и далее очистим каталог от результатов проведения экспериментов(рис. 4.15).

```
mininet@mininet-vm: ~/work/lab_netem_ii/simple-drop
mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$ make
sudo python lab_netem_ii.py
File "lab_netem_ii.py", line 17
    lost_packets = set([for i in range(1, total_packets+1) if i not in received_packets])
                        ^
SyntaxError: invalid syntax
make: *** [Makefile:4: ping.dat] Error 1
mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$ nano lab_netem_ii.py
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 delay
*** 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 "time=" | awk \'{print $5, $7}\'' | sed -e \'s/time=//g\' -e
\'s/icmp_seq=//g\' > ping.dat')
Total packets: 100
Received packets count: 74
Lost packets count: 26
Lost packets: [1, 3, 4, 5, 6, 10, 23, 29, 32, 33, 34, 38, 46, 50, 55, 56, 63, 65, 67, 70, 73, 82, 84, 90,
94, 95]
Loss packets percent: 26.00%
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$ make clean
rm -f *.dat
mininet@mininet-vm:~/work/lab_netem_ii/simple-drop$
```

Рис. 4.15: Выполнение эксперимента и последующая очистка каталога

Теперь создадим каталоги основываясь на simple-drop под выполнение самостоятельной работы(рис. 4.16).

```
mininet@mininet-vm: ~/work/lab_netem_ii
mininet@mininet-vm:~/work/lab_netem_ii$ ls
simple-drop
mininet@mininet-vm:~/work/lab_netem_ii$ cp -R simple-drop corr_drop
mininet@mininet-vm:~/work/lab_netem_ii$ cp -R simple-drop simple_corrupt
mininet@mininet-vm:~/work/lab_netem_ii$ cp -R simple-drop delay_reorder
mininet@mininet-vm:~/work/lab_netem_ii$ cp -R simple-drop simple_duplicate
mininet@mininet-vm:~/work/lab_netem_ii$
```

Рис. 4.16: Создание каталогов для самостоятельной работы

Далее реализуем воспроизводимые эксперименты по исследованию параметров сети, связанных с потерей, изменением порядка и повреждением пакетов при передаче данных.

Скорректируем lab_netem_ii.py для эксперимента по добавлению потери и коэффициента корреляции.(рис. 4.17)

```

mininet@mininet-vm: ~/work/lab_netem_ii/corr_drop
GNU nano 4.8 lab_netem_ii.py Modified
return

received_packets = set()

with open(file_path, 'r') as f:
    for line in f:
        parts = line.split()
        if parts:
            try:
                packet_n = int(parts[0])
                received_packets.add(packet_n)
            except ValueError:
                pass # если не число

lost_packets = (i for i in range(1, total_packets + 1) if i not in received_packets)
lost_packets_count = len(lost_packets)
loss_percent = (lost_packets_count / total_packets) * 100

print(f'Total packets: {total_packets}')
print(f'Received packets count: {len(received_packets)}')
print(f'Lost packets count: {lost_packets_count}')
print(f'Lost packets: {sorted(lost_packets)}')
print(f'Loss packets percent: {loss_percent:.2f}%')

def emptyNet():
    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 loss 50% 50%' )
    h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem loss 10%' )

    time.sleep(10) # Wait 10 seconds

    info( '*** Ping\n' )
    h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "time=" | awk \'{print $5, $7}\'' | sed -e \'/time=
analyze_ping_data()

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

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

```

Рис. 4.17: Листинг программы для эксперимента по добавлению потери и коэффициента корреляции

Запустим скрипт.(рис. 4.18)

```

mininet@mininet-vm: ~/work/lab_netem_ii/corr_drop
mininet@mininet-vm:~/work/lab_netem_ii$ ls
corr_drop delay_reorder simple_corrupt simple_drop simple_duplicate
mininet@mininet-vm:~/work/lab_netem_ii$ cd corr_drop
mininet@mininet-vm:~/work/lab_netem_ii/corr_drop$ nano lab_netem_ii.py
mininet@mininet-vm:~/work/lab_netem_ii/corr_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 delay
*** h1 : ('tc qdisc add dev h1-eth0 root netem loss 50% 50%,')
*** Ping
*** h1 : ('ping -c 100', '10.0.0.2', '| grep "time=" | awk \'{print $5, $7}\'' | sed -e \'s/time=//g\' -e \'s/icmp_seq=//g\' > ping.dat')
Total packets: 100
Received packets count: 39
Lost packets count: 61
Lost packets: [1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 14, 16, 17, 18, 19, 20, 23, 24, 25, 26, 27, 28, 30, 31, 33, 34, 37, 41, 42, 43, 46, 47, 51, 52, 53, 54, 61, 65, 66, 67, 68, 69, 70, 71, 72, 74, 75, 77, 78, 80, 81, 82, 83, 85, 86, 88, 90, 91, 94, 95, 99]
Loss packets percent: 61.00%
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet ping.dat
mininet@mininet-vm:~/work/lab_netem_ii/corr_drop$

```

Рис. 4.18: Выполнение эксперимента по добавлению потери и коэффициента корреляции

Скорректируем lab_netem_ii.py для эксперимента по повреждению пакетов.(рис. 4.19)

```

mininet@mininet-vm: ~/work/lab_netem_ii/simple_corrupt
GNU nano 4.8 lab_netem_ii.py
#!/usr/bin/env python
from mininet.net import Mininet
from mininet.node import Controller
from mininet.log import setLogLevel
import time

def analyze_corrupt(file='iperf_result.txt'):
    print("=== Packet Corruption Analysis ===")
    total_retr = 0
    with open(file) as f:
        for line in f:
            parts = line.split()
            if len(parts) >= 9 and parts[-3].isdigit():
                total_retr += int(parts[-3])
    print(f"Total TCP retransmissions: {total_retr}")

def experiment():
    net = Mininet(controller=Controller, waitConnected=True)
    net.addController('c0')

    h1 = net.addHost('h1', ip='10.0.0.1')
    h2 = net.addHost('h2', ip='10.0.0.2')
    s1 = net.addSwitch('s1')
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.start()

    h1.cmd('tc qdisc add dev h1-eth0 root netem corrupt 0.01%')
    h2.cmd('iperf3 -s > /tmp/iperf_srv.log 2>&1 &')
    time.sleep(1)

    h1.cmd(f'iperf3 -c {h2.IP()} > iperf_result.txt')

    analyze_corrupt()
    net.stop()

if __name__ == "__main__":
    setLogLevel("info")
    experiment()

```

Рис. 4.19: Листинг программы для эксперимента по повреждению пакетов

Запустим скрипт.(рис. 4.20)

```
mininet@mininet-vm:~/work/lab_netem_ii/simple_corrupt$ make
sudo python lab_netem_ii.py
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
=== Packet Corruption Analysis ===
Total TCP retransmissions: 32
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
```

Рис. 4.20: Выполнение эксперимента по повреждению пакетов

Скорректируем lab_netem_ii.py для эксперимента по изменению порядка пакетов.(рис. 4.21)

```
mininet@mininet-vm: ~/work/lab_netem_ii/delay_reorder
GNU nano 4.8 lab_netem_ii.py
#!/usr/bin/env python
from mininet.net import Mininet
from mininet.node import Controller
from mininet.log import setLogLevel
import time

def analyze_reorder(file='reorder.dat'):
    delays = []
    with open(file, 'r') as f:
        for line in f:
            try:
                rtt = float(line.strip())
                delays.append(rtt)
            except:
                pass
    if not delays:
        print("No data found.")
        return
    threshold = min(delays) + 5
    fast = [d for d in delays if d < threshold]
    slow = [d for d in delays if d >= threshold]

    print("=== Packet Reordering Analysis ===")
    print(f"Total packets: {len(delays)}")
    print(f"Fast path (~no delay): {len(fast)}")
    print(f"Slow path (~delayed): {len(slow)}")
    print(f"Avg fast RTT: {sum(fast)/len(fast):.3f} ms")
    print(f"Avg slow RTT: {sum(slow)/len(slow):.3f} ms")

def experiment():
    net = Mininet(controller=Controller, waitConnected=True)
    net.addController('c0')
    h1 = net.addHost('h1', ip='10.0.0.1')
    h2 = net.addHost('h2', ip='10.0.0.2')
    s1 = net.addSwitch('s1')
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.start()

    h1.cmd('tc qdisc add dev h1-eth0 root netem delay 10ms reorder 25% 50%')
    h1.cmd(f'ping -c 20 {h2.IP()} | grep "time=" | sed "s/.time=/" | sed "s/ ms/" > reorder.dat')

    analyze_reorder()
    net.stop()

if __name__ == "__main__":
    setLogLevel("info")
    experiment()
```

Рис. 4.21: Листинг программы для эксперимента по изменению порядка пакетов

Запустим скрипт.(рис. 4.22)

```

mininet@mininet-vm:~/work/lab_netem_i/delay_reorder$ nano lab_netem_i.py
mininet@mininet-vm:~/work/lab_netem_i/delay_reorder$ make
sudo python lab_netem_i.py
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
=== Packet Reordering Analysis ===
Total packets: 20
Fast path (-no delay): 2
Slow path (-delayed): 18
Avg fast RTT: 0.060 ms
Avg slow RTT: 11.339 ms
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet reordero.dat

```

Рис. 4.22: Выполнение эксперимента по изменению порядка пакетов

Скорректируем lab_netem_i.py для эксперимента по дублированию пакетов.(рис. 4.23)

```

mininet@mininet-vm: ~/work/lab_netem_i/simple_duplicate
GNU nano 4.8 lab_netem_i.py
#!/usr/bin/env python
from mininet.net import Mininet
from mininet.node import Controller
from mininet.log import setLogLevel

def analyze_dup(file='dup.txt'):
    with open(file) as f:
        lines = f.readlines()
        duplicates = [l for l in lines if "DUP!" in l]
        print("=== Packet Duplication Analysis ===")
        print(f"Total packets: {len(lines)}")
        print(f"Duplicated packets: {len(duplicates)}")
        print(f"Duplication percent: {(len(duplicates)/len(lines))*100:.2f}%")

def experiment():
    net = Mininet(controller=Controller, waitConnected=True)
    net.addController('c0')
    h1 = net.addHost('h1', ip='10.0.0.1')
    h2 = net.addHost('h2', ip='10.0.0.2')
    s1 = net.addSwitch('s1')
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.start()

    h1.cmd('tc qdisc add dev h1-eth0 root netem duplicate 50%')
    h1.cmd(f'ping -c 30 {h2.IP()} > dup.txt')

    analyze_dup()
    net.stop()

if __name__ == "__main__":
    setLogLevel("info")
    experiment()

```

Рис. 4.23: Листинг программы для эксперимента по дублированию пакетов

Запустим скрипт.(рис. 4.24)

```
mininet@mininet-vm:~/work/lab_netem_ii/simple_duplicates$ make
sudo python lab_netem_ii.py
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
=== Packet Duplication Analysis ===
Total packets: 49
Duplicated packets: 14
Duplication percent: 28.57%
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
sudo chown mininet:mininet dup.txt
```

Рис. 4.24: Выполнение эксперимента по дублированию пакетов

5 Выводы

В результате выполнения данной лабораторной работы я получил навыки проведения интерактивных экспериментов в среде Mininet по исследованию параметров сети, связанных с потерей, дублированием, изменением порядка и повреждением пакетов при передаче данных.

Список литературы

1. Mininet [Электронный ресурс]. Mininet Project Contributors. URL: <http://mininet.org/> (дата обращения: 07.10.2025).