

Лабораторная работа №5

Эмуляция и измерение потерь пакетов в глобальных сетях

Ланцова Яна Игоревна

Содержание

1 Цель работы	5
2 Задачи	6
3 Выполнение лабораторной работы	7
3.1 Запуск лабораторной топологии	7
3.2 Добавление потери пакетов на интерфейс, подключённый к эмулируемой глобальной сети	9
3.3 Добавление значения корреляции для потери пакетов в эмулируемой глобальной сети	12
3.4 Добавление повреждения пакетов в эмулируемой глобальной сети	12
3.5 Добавление переупорядочивания пакетов в интерфейс подключения к эмулируемой глобальной сети	13
3.6 Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети	14
3.7 Воспроизведение экспериментов	15
3.8 Задание для самостоятельного выполнения	19
4 Выводы	21

Список иллюстраций

3.1 Исправление MIT magic cookie	7
3.2 Простейшая топология	8
3.3 ifconfig на хостах h1 и h2	8
3.4 Проверка подключения между хостами	9
3.5 Добавление потери в 10%	9
3.6 Просмотр сводного отчета	10
3.7 Добавление потери в 10% на второй хост	11
3.8 Восстановление исходных значений потерь	11
3.9 Добавление значения корреляции для потери пакетов	12
3.10 Добавление повреждения пакетов и проверка	13
3.11 Добавление переупорядочивания пакетов	14
3.12 Добавление дублирования пакетов	15
3.13 Создание рабочего каталога	15
3.14 Скрипт на Python для эксперимента	16
3.15 Makefile для управления процессом проведения эксперимента	18
3.16 Запуск эксперимента	19
3.17 Изменение файла lab_netem_ii.py	19
3.18 Запуск эксперимента	20
3.19 Просмотр информации	20

Список таблиц

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.

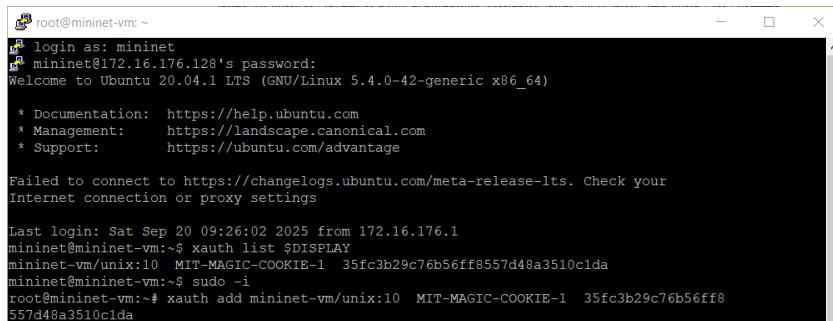
2 Задачи

1. Задайте простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8.
2. Проведите интерактивные эксперименты по 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

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



```
root@mininet-vm: ~
login as: mininet
mininet@172.16.176.128's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Sat Sep 20 09:26:02 2025 from 172.16.176.1
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 35fc3b29c76b56ff8557d48a3510c1da
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 35fc3b29c76b56ff8
557d48a3510c1da
```

Рис. 3.1: Исправление MIT magic cookie

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

```

root@mininet-vm:~# sudo mn --topo=single,2 -x
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Running terms on localhost:10.0
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
X "controller: c0" (root)@mininet-vm
root@mininet-vm:~#

```

Рис. 3.2: Простейшая топология

На хостах h1 и h2 введем команду ifconfig, чтобы отобразить информацию, относящуюся к их сетевым интерфейсам и назначенным им IP-адресам. В дальнейшем при работе с NETEM и командой tc будут использоваться интерфейсы h1-eth0 и h2-eth0(рис. 3.3).

```

X "host: h1" @mininet-vm
root@mininet-vm:~# 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 2e:bc:d9:2e:1b:a6 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 1208 bytes 290656 (290.6 KB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 1208 bytes 290656 (290.6 KB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

X "host: h2" @mininet-vm
root@mininet-vm:~# 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 92:90:be:9c:0e:5b 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 1221 bytes 334108 (334.1 KB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 1221 bytes 334108 (334.1 KB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Рис. 3.3: ifconfig на хостах h1 и h2

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

```
root@mininet-vm:~# 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=0.166 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.167 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.188 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.159 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.122 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.117 ms
...
-- 10.0.0.1 ping statistics --
6 packets transmitted, 6 received, 0% packet loss, time 5077ms
rtt min/avg/max/mdev = 0.117/0.779/3.894/1.393 ms

root@mininet-vm:~# 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=0.95 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.537 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.302 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.184 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.109 ms
...
-- 10.0.0.2 ping statistics --
6 packets transmitted, 6 received, 0% packet loss, time 5087ms
rtt min/avg/max/mdev = 0.109/0.709/8.959/3.245 ms
```

Рис. 3.4: Проверка подключения между хостами

3.2 Добавление потери пакетов на интерфейс,

подключённый к эмулируемой глобальной сети

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. 3.5).

```
root@mininet-vm:~# sudo tc qdisc add dev h1-eth0 root netem loss 10%
root@mininet-vm:~# 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=4.36 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.45 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.425 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.242 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.243 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.129 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.194 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.179 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.173 ms
```

Рис. 3.5: Добавление потери в 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.

Проверим, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 100 с хоста h1. Параметр -c указывает общее количество пакетов для отправки. Обратите внимание на значения icmp_seq. Некоторые номера последовательности отсутствуют из-за потери пакетов. В сводном отчёте ping сообщает о проценте потерянных пакетов после завершения передачи (рис. 3.6).

```
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.134 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.139 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=0.126 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.127 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.255 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.196 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.123 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=0.131 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 96 received, 4% packet loss, time 101329ms
rtt min/avg/max/mdev = 0.097/0.245/4.363/0.445 ms
root@mininet-vm:~# █
```

Рис. 3.6: Просмотр сводного отчета

Потеряно 4% пакетов.

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. 3.7).

```

root@mininet-vm:~# sudo tc qdisc add dev h2-eth0 root netem loss 10%
root@mininet-vm:~# 
  "host: h1" @mininet-vm
64 bytes from 10.0.0.2: icmp_seq=74 ttl=64 time=0.236 ms
64 bytes from 10.0.0.2: icmp_seq=75 ttl=64 time=0.308 ms
64 bytes from 10.0.0.2: icmp_seq=76 ttl=64 time=0.189 ms
64 bytes from 10.0.0.2: icmp_seq=77 ttl=64 time=0.232 ms
64 bytes from 10.0.0.2: icmp_seq=78 ttl=64 time=0.227 ms
64 bytes from 10.0.0.2: icmp_seq=79 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.137 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.161 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=0.174 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.224 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.154 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=0.324 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=0.318 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.356 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.382 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.153 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.330 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.327 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.229 ms
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 82 received, 18% packet loss, time 101320ms
rtt min/avg/max/mdev = 0.118/0.310/3.676/0.426 ms

```

Рис. 3.7: Добавление потери в 10% на второй хост

Потеряно 18% пакетов.

Восстановим конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса. Для отправителяй h1 и h2 (рис. 3.8). Также убедимся, что соединение от хоста h1 к хосту h2 не имеет явной потери пакетов, запустив команду ping с терминала хоста h1.

```

root@mininet-vm:~# sudo tc qdisc del dev h2-eth0 root netem
root@mininet-vm:~# 
root@mininet-vm:~# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:~# ping 10.0.0.2 -c 5
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=3.55 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.663 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.198 ms
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4039ms
rtt min/avg/max/mdev = 0.116/1.189/3.550/1.267 ms
root@mininet-vm:~# 

```

Рис. 3.8: Восстановление исходных значений потерь

3.3 Добавление значения корреляции для потери пакетов в эмулируемой глобальной сети

Добавим на интерфейсе узла h1 коэффициент потери пакетов 50% (такой высокий уровень потери пакетов маловероятен), и каждая последующая вероятность зависит на 50% от предыдущей: Проверим, что на соединении от хоста h1 к хосту h2 имеются потери пакетов, используя команду ping с параметром -c 50 с хоста h1 (рис. 3.9).

```
root@mininet-vm:~# sudo tc qdisc add dev h1-eth0 root netem loss 50% 50%
root@mininet-vm:~# 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=5 ttl=64 time=2.27 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=1.94 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.690 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.144 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=2.61 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.245 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.282 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.237 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.194 ms
64 bytes from 10.0.0.2: icmp_seq=24 ttl=64 time=0.296 ms
64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=0.236 ms
64 bytes from 10.0.0.2: icmp_seq=28 ttl=64 time=0.246 ms
64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=32 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=33 ttl=64 time=0.296 ms
64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=0.136 ms
64 bytes from 10.0.0.2: icmp_seq=36 ttl=64 time=0.127 ms
64 bytes from 10.0.0.2: icmp_seq=37 ttl=64 time=0.180 ms
64 bytes from 10.0.0.2: icmp_seq=38 ttl=64 time=0.249 ms
64 bytes from 10.0.0.2: icmp_seq=40 ttl=64 time=0.284 ms
64 bytes from 10.0.0.2: icmp_seq=41 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=42 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=46 ttl=64 time=0.131 ms
64 bytes from 10.0.0.2: icmp_seq=47 ttl=64 time=0.282 ms
64 bytes from 10.0.0.2: icmp_seq=48 ttl=64 time=0.191 ms
64 bytes from 10.0.0.2: icmp_seq=50 ttl=64 time=0.140 ms
...
--- 10.0.0.2 ping statistics ---
50 packets transmitted, 27 received, 46% packet loss, time 50068ms
rtt min/avg/max/mdev = 0.099/0.443/2.606/0.661 ms
root@mininet-vm:~# sudo tc qdisc del dev h1-eth0 root netem
root@mininet-vm:~#
```

Рис. 3.9: Добавление значения корреляции для потери пакетов

3.4 Добавление повреждения пакетов в эмулируемой глобальной сети

Добавим на интерфейсе узла h1 0,01% повреждения пакетов с помощью команды sudo tc qdisc add dev h1-eth0 root netem corrupt 0.01%. Проверим конфи-

гурацию с помощью инструмента iPerf3 для проверки повторных передач (рис. 3.10).

```
root@mininet-vm:~# iperf3 -s
warning: this system does not seem to support IPv6 - trying IPv4
-----
Server listening on 5201
-----
Accepted connection from 10.0.0.1, port 34558
[ 7] local 10.0.0.2 port 5201 connected to 10.0.0.1 port 34560
[ ID] Interval      Transfer     Bitrate
[ 7]  0.00-1.00   sec   176 MBytes  1.47 Gbits/sec
[ 7]  1.00-2.00   sec   175 MBytes  1.47 Gbits/sec
[ 7]  2.00-3.00   sec   168 MBytes  1.41 Gbits/sec
[ 7]  3.00-4.00   sec   180 MBytes  1.51 Gbits/sec
[ 7]  4.00-5.00   sec   150 MBytes  1.26 Gbits/sec
[ 7]  5.00-6.00   sec   172 MBytes  1.44 Gbits/sec
[ 7]  6.00-7.00   sec   185 MBytes  1.55 Gbits/sec
-----
root@mininet-vm:~# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 7] local 10.0.0.1 port 34560 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer     Bitrate     Retr Cwnd
[ 7]  0.00-1.01   sec   176 MBytes  1.47 Gbits/sec  0    782 KBytes
[ 7]  1.01-2.01   sec   175 MBytes  1.47 Gbits/sec  0    1.18 MBytes
[ 7]  2.01-3.01   sec   169 MBytes  1.41 Gbits/sec  0    1.18 MBytes
[ 7]  3.01-4.00   sec   179 MBytes  1.51 Gbits/sec  1    1.56 MBytes
[ 7]  4.00-5.00   sec   150 MBytes  1.26 Gbits/sec  0    1.56 MBytes
[ 7]  5.00-6.00   sec   171 MBytes  1.44 Gbits/sec  0    1.76 MBytes
[ 7]  6.00-7.00   sec   185 MBytes  1.55 Gbits/sec  0    1.85 MBytes
[ 7]  7.00-8.00   sec   441 MBytes  3.70 Gbits/sec  1    1.38 MBytes
[ 7]  8.00-9.00   sec   466 MBytes  3.91 Gbits/sec  0    1.38 MBytes
[ 7]  9.00-10.00  sec   461 MBytes  3.87 Gbits/sec  1    991 KBytes
-----
[ ID] Interval      Transfer     Bitrate     Retr
[ 7]  0.00-10.00  sec  2.51 GBytes  2.16 Gbits/sec  3
[ 7]  0.00-10.00  sec  2.49 GBytes  2.14 Gbits/sec
                                         sender
                                         receiver
iperf Done.
root@mininet-vm:~# sudo tc qdisc del dev h1-eth0 root netem
```

Рис. 3.10: Добавление повреждения пакетов и проверка

Значения повторно переданных пакетов можно увидеть в колонке Retr. Общее число 3.

3.5 Добавление переупорядочивания пакетов в интерфейс подключения к эмулируемой глобальной сети

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. 3.11):

```
root@mininet-vm:~# sudo tc qdisc add dev h1-eth0 root netem delay 10ms reorder
25% 50%
root@mininet-vm:~# 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=13.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=12.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=12.0 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=11.1 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=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=11.4 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=10.5 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=0.203 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=10.7 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=10.9 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=11.2 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=11.0 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=11.3 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=10.6 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=10.8 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=11.3 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19070ms
rtt min/avg/max/mdev = 0.203/10.695/13.897/2.528 ms
root@mininet-vm:~# sudo tc qdisc del dev h1-eth0 root netem
```

Рис. 3.11: Добавление переупорядочивания пакетов

3.6 Добавление дублирования пакетов в интерфейс подключения к эмулируемой глобальной сети

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 -c 20 с хоста h1. Дубликаты пакетов помечаются как DUP! (рис. 3.12).

```

root@mininet-vm:~# sudo tc qdisc add dev h1-eth0 root netem duplicate 50%
root@mininet-vm:~# 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=3.74 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.21 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.966 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.209 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.279 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.289 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.285 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.182 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.351 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.361 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.290 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.321 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.331 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.172 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.182 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.458 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.486 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.299 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.341 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.308 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.318 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.428 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.457 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.224 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.234 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.237 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.267 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.277 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.269 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, +9 duplicates, 0% packet loss, time 19418ms
rtt min/avg/max/mdev = 0.172/0.509/3.740/0.715 ms
root@mininet-vm:~# sudo tc qdisc del dev h1-eth0 root netem

```

Рис. 3.12: Добавление дублирования пакетов

3.7 Воспроизведение экспериментов

Для каждого воспроизводимого эксперимента `expname` создадим свой каталог, в котором будут размещаться файлы эксперимента. В виртуальной среде `mininet` в своём рабочем каталоге с проектами создадим каталог `simple-drop` и перейдем в него (рис. 3.13).

```

root@mininet-vm:~# mkdir -p ~/work/lab_netem_ii/simple-drop
root@mininet-vm:~# cd ~/work/lab_netem_ii/simple-drop
root@mininet-vm:~/work/lab_netem_ii/simple-drop# touch lab_netem_ii.py
root@mininet-vm:~/work/lab_netem_ii/simple-drop# ls
lab_netem_ii.py
root@mininet-vm:~/work/lab_netem_ii/simple-drop#

```

Рис. 3.13: Создание рабочего каталога

Создадим скрипт для эксперимента `lab_netem_ii.py` (рис. 3.14). В этом скрипте создается простейшая топология сети, затем с помощью команд, использованных нами ранее задается потеря в 10% для обоих хостов, после чего пингуется второй хост (100 сообщений отправляется), при этом из сообщений при пинге

вытаскиваются номер сообщения и значение времени, которые записываются в файл с данными.

```
GNU nano 4.8          lab_netem_ii.py          Modified
#!/usr/bin/env python
"""
Simple experiment.
Output: ping.dat
"""

from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI

from mininet.log import setLogLevel, info
import time

def emptyNet():

    "Create an empty network and add nodes to it."
    net = Mininet( controller=Controller, waitConnected=True )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1' )
    h2 = net.addHost( 'h2', ip='10.0.0.2' )

    info( '*** Adding switch\n' )
    s1 = net.addSwitch( 's1' )

    info( '*** Creating links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s1 )

    info( '*** Starting network\n' )
    net.start()

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

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

Рис. 3.14: Скрипт на Python для эксперимента

Скорректируем скрипт так, чтобы на экран или 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\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 "packet loss" | awk \'{print $6, $7, $8}\'')

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

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

```

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

```

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. 3.15: Makefile для управления процессом проведения эксперимента

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

```

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 "packet loss" | awk \'{print $6, $7, $8}\'')
23% packet loss,
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done

```

Рис. 3.16: Запуск эксперимента

3.8 Задание для самостоятельного выполнения

Самостоятельно реализуем воспроизводимые эксперименты по исследованию параметров сети, связанных с потерей, изменением порядка и повреждением пакетов при передаче данных (рис. 3.17).

```

GNU nano 4.8
                lab_netem_ii.py
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 5% duplicate 50%' )
h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem loss 5% delay 40ms reorder 40% 10%' )

time.sleep(10) # Wait 10 seconds

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()

```

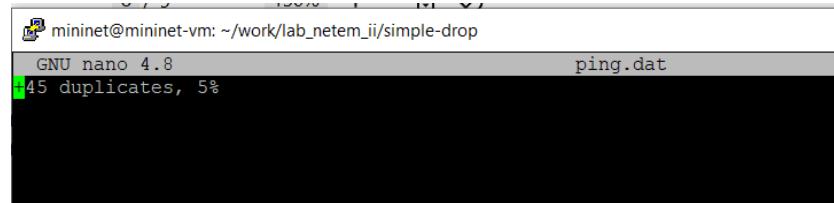
Рис. 3.17: Изменение файла lab_netem_ii.py

Выполним эксперимент, написав команду make (рис. 3.18).

```
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 5% duplicate 50%',)
*** h2 : ('tc qdisc add dev h2-eth0 root netem loss 5% delay 40ms reorder 40% 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
```

Рис. 3.18: Запуск эксперимента

Посмотрим информацию, записанную в ping.dat (рис. 3.19).



```
mininet@mininet-vm: ~/work/lab_netem_ii/simple-drop
GNU nano 4.8                               ping.dat
45 duplicates, 5%
```

Рис. 3.19: Просмотр информации

4 Выводы

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