Отчет по лабораторной работе №4

Дисциплина: Моделирование сетей передачи данных

Амуничников Антон Игоревич

Содержание

# 1. Цель работы

Основной целью работы является знакомство с NETEM — инструментом для тестирования производительности приложений в виртуальной сети, а также получение навыков проведения интерактивного и воспроизводимого экспериментов по измерению задержки и её дрожания (jitter) в моделируемой сети в среде Mininet.

# 2. Задание

1. Задайте простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8.
2. Проведите интерактивные эксперименты по добавлению/изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети.
3. Реализуйте воспроизводимый эксперимент по заданию значения задержки в эмулируемой глобальной сети. Постройте график.
4. Самостоятельно реализуйте воспроизводимые эксперименты по изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети. Постройте графики.

# 3. Теоретическое введение

Mininet[**mininet?**] – это эмулятор компьютерной сети. Под компьютерной сетью подразумеваются простые компьютеры — хосты, коммутаторы, а так же OpenFlow-контроллеры. С помощью простейшего синтаксиса в примитивном интерпретаторе команд можно разворачивать сети из произвольного количества хостов, коммутаторов в различных топологиях и все это в рамках одной виртуальной машины(ВМ). На всех хостах можно изменять сетевую конфигурацию, пользоваться стандартными утилитами(ifconfig, ping) и даже получать доступ к терминалу. На коммутаторы можно добавлять различные правила и маршрутизировать трафик.

# 4. Выполнение лабораторной работы

## 4.1 Запуск лабораторной топологии

Запустим виртуальную среду с mininet. Из основной ОС подключимся к виртуальной машине. В виртуальной машине mininet при необходимости исправим права запуска X-соединения. Скопируем значение куки (MIT magic cookie) своего пользователя mininet в файл для пользователя root ([рис. 1](#fig-001)).

|  |
| --- |
| Рисунок 1: Исправление прав запуска X-соединения |

Зададим простейшую топологию, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8 ([рис. 2](#fig-002)).

|  |
| --- |
| Рисунок 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](#fig-003)).

|  |
| --- |
| Рисунок 3: ifconfig на хостах h1 и h2 |

Проверим подключение между хостами h1 и h2 с помощью команды ping с параметром -c 6 ([рис. 4](#fig-004)).

|  |
| --- |
| Рисунок 4: Проверка подключения между хостами |

## 4.2 Интерактивные эксперименты

### 4.2.1 Добавление/изменение задержки в эмулируемой глобальной сети

Сетевые эмуляторы задают задержки на интерфейсе. Например, задержка, вносимая в интерфейс коммутатора A, который подключён к интерфейсу коммутатора B, может представлять собой задержку распространения WAN, соединяющей оба коммутатора.

На хосте h1 добавим задержку в 100 мс к выходному интерфейсу.

sudo tc qdisc add dev h1-eth0 root netem delay 100ms

* sudo: выполнить команду с более высокими привилегиями;
* tc: вызвать управление трафиком Linux;
* qdisc: изменить дисциплину очередей сетевого планировщика;
* add: создать новое правило;
* dev h1-eth0: указать интерфейс, на котором будет применяться правило;
* netem: использовать эмулятор сети;
* delay 100ms: задержка ввода 100 мс.

Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду ping с параметром -c 6 с хоста h1 ([рис. 5](#fig-005)). Минимальное - (100.467), среднее - (101.597), максимальное - (102.384), стандартное отклонение времени (0.656) приема-передачи.

|  |
| --- |
| Рисунок 5: Добавление задержки в 100мс |

Для эмуляции глобальной сети с двунаправленной задержкой необходимо к соответствующему интерфейсу на хосте h2 также добавим задержку в 100 миллисекунд ([рис. 6](#fig-006)).

Проверим, что соединение между хостом h1 и хостом h2 имеет RTT в 200 мс (100 мс от хоста h1 к хосту h2 и 100 мс от хоста h2 к хосту h1), повторив команду ping с параметром -c 6 на терминале хоста h1. Минимальное - (202.068), среднее - (202.884), максимальное - (204.777), стандартное отклонение времени (0.943) приема-передачи.

|  |
| --- |
| Рисунок 6: Двунаправленная задержка соединения |

### 4.2.2 Изменение задержки в эмулируемой глобальной сети

Изменим задержку со 100 мс до 50 мс для отправителя h1 и для получателя h2 ([рис. 7](#fig-007)).

Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс, используя команду ping с параметром -c 6 с терминала хоста h1. Минимальное - (101.050), среднее - (101.253), максимальное - (108.223), стандартное отклонение времени (2.475) приема-передачи.

|  |
| --- |
| Рисунок 7: Изменение задержки на 50мс |

### 4.2.3 Восстановление исходных значений (удаление правил) задержки в эмулируемой глобальной сети

Восстановим конфигурацию по умолчанию, удалив все правила, применённые к сетевому планировщику соответствующего интерфейса. Для отправителя h1: sudo tc qdisc del dev h1-eth0 root netem. Для получателя h2: sudo tc qdisc del dev h2-eth0 root netem. Проверим, что соединение между хостом h1 и хостом h2 не имеет явно установленной задержки, используя команду ping с параметром -c 6 с терминала хоста h1. Минимальное - (0.145), среднее - (1.256), максимальное - (4.875), стандартное отклонение времени (1.655) приема-передачи ([рис. 8](#fig-008)).

|  |
| --- |
| Рисунок 8: Восстановление исходных значений задержки |

### 4.2.4 Добавление значения дрожания задержки в интерфейс подключения к эмулируемой глобальной сети

Добавим на узле h1 задержку в 100 мс со случайным отклонением 10 мс. Проверим, что соединение от хоста h1 к хосту h2 имеет задержку 100 мс со случайным отклонением ±10 мс, используя в терминале хоста h1 команду ping с параметром -c 6. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (91.961), среднее - (99.758), максимальное - (106.319), стандартное отклонение времени (5.130) приема-передачи ([рис. 9](#fig-009)).

|  |
| --- |
| Рисунок 9: Добавление значения дрожания задержки в интерфейс подключения |

### 4.2.5 Добавление значения корреляции для джиттера и задержки в интерфейс подключения к эмулируемой глобальной сети

Добавим на интерфейсе хоста h1 задержку в 100 мс с вариацией ±10 мс и значением корреляции в 25%. Убедимся, что все пакеты, покидающие устройство h1 на интерфейсе h1-eth0, будут иметь время задержки 100 мс со случайным отклонением ±10 мс, при этом время передачи следующего пакета зависит от предыдущего значения на 25%. Используем для этого в терминале хоста h1 команду ping с параметром -c 20. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (91.887), среднее - (102.565), максимальное - (111.050), стандартное отклонение времени (5.730) приема-передачи ([рис. 10](#fig-010)).

|  |
| --- |
| Рисунок 10: Добавление значения корреляции для джиттера и задержки в интерфейс подключения |

### 4.2.6 Распределение задержки в интерфейсе подключения к эмулируемой глобальной сети

Зададим нормальное распределение задержки на узле h1 в эмулируемой сети. Убедимся, что все пакеты, покидающие хост h1 на интерфейсе h1-eth0, будут иметь время задержки, которое распределено в диапазоне 100 мс ±20 мс. Используем для этого команду ping на терминале хоста h1 с параметром -c 10. Восстановим конфигурацию интерфейса по умолчанию на узле h1. Минимальное - (74.585), среднее - (105.367), максимальное - (126.328), стандартное отклонение времени (16.340) приема-передачи. Завершим работу mininet в интерактивном режиме ([рис. 11](#fig-011)).

|  |
| --- |
| Рисунок 11: Распределение задержки в интерфейсе подключения |

## 4.3 Воспроизведение экспериментов

### 4.3.1 Предварительная подготовка

Обновим репозитории программного обеспечения на виртуальной машине: sudo apt-get update. Установим пакет geeqie — понадобится для просмотра файлов png: sudo apt install geeqie.

Для каждого воспроизводимого эксперимента expname создадим свой каталог, в котором будут размещаться файлы эксперимента: mkdir -p ~/work/lab\_netem\_i/expname. Здесь expname может принимать значения simple-delay, change-delay, jitter-delay, correlation-delay и т.п. Для каждого случая создадим скрипт для проведения эксперимента lab\_netem\_i.py и скрипт для визуализации результатов ping\_plot.

### 4.3.2 Добавление задержки для интерфейса, подключающегося к эмулируемой глобальной сети

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

В виртуальной среде mininet в своём рабочем каталоге с проектами создадим каталог simple-delay и перейдем в него.

mkdir -p ~/work/lab\_netem\_i/simple-delay  
cd ~/work/lab\_netem\_i/simple-delay

Создадим скрипт для эксперимента lab\_netem\_i.py.

#!/usr/bin/env python  
  
"""  
Simple experiment.  
Output: ping.dat  
"""  
  
from mininet.net import Mininet  
from mininet.node import Controller  
from mininet.cli import CLI  
  
from mininet.log import setLogLevel, info  
import time  
  
def emptyNet():  
  
 "Create an empty network and add nodes to it."  
  
 net = Mininet( controller=Controller, waitConnected=True )  
  
 info( '\*\*\* Adding controller\n' )  
 net.addController( 'c0' )  
  
 info( '\*\*\* Adding hosts\n' )  
 h1 = net.addHost( 'h1', ip='10.0.0.1' )  
 h2 = net.addHost( 'h2', ip='10.0.0.2' )  
  
 info( '\*\*\* Adding switch\n' )  
 s1 = net.addSwitch( 's1' )  
  
 info( '\*\*\* Creating links\n' )  
 net.addLink( h1, s1 )  
 net.addLink( h2, s1 )  
  
 info( '\*\*\* Starting network\n')  
 net.start()  
  
 info( '\*\*\* Set delay\n')  
 h1.cmdPrint( 'tc qdisc add dev h1-eth0 root netem delay 100ms' )  
 h2.cmdPrint( 'tc qdisc add dev h2-eth0 root netem delay 100ms' )  
  
 time.sleep(10) # Wait 10 seconds  
  
 info( '\*\*\* Ping\n')  
 h1.cmdPrint( 'ping -c 100', h2.IP(), '| grep "time=" | awk \'{print $5, $7}\' | sed -e \'s/time=//g\' -e \'s/icmp\_seq=//g\' > ping.dat' )  
  
 info( '\*\*\* Stopping network' )  
 net.stop()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 setLogLevel( 'info' )  
 emptyNet()

Создадим скрипт для визуализации ping\_plot результатов эксперимента ([рис. 12](#fig-012)).

|  |
| --- |
| Рисунок 12: Скрипт для визуализации ping\_plot |

Зададим права доступа к файлу скрипта: chmod +x ping\_plot ([рис. 13](#fig-013)).

|  |
| --- |
| Рисунок 13: Создание каталогов, права к файлу скрипта |

Создадим Makefile для управления процессом проведения эксперимента ([рис. 14](#fig-014)).

|  |
| --- |
| Рисунок 14: Makefile для управления процессом проведения эксперимента |

Выполним эксперимент. Продемонстрируем построенный в результате выполнения скриптов график ([рис. 15](#fig-015)).

|  |
| --- |
| Рисунок 15: Результат выполнения скрипта |

Из файла ping.dat удалим первую строку и заново постройте график. Продемонстрируем построенный в результате график ([рис. 16](#fig-016)).

|  |
| --- |
| Рисунок 16: Результат выполнения скрипта |

Разработаем скрипт для вычисления на основе данных файла ping.dat минимального, среднего, максимального и стандартного отклонения времени приёма-передачи ([рис. 17](#fig-017)).

|  |
| --- |
| Рисунок 17: Cкрипт rtt.py |

Продемонстрируем работу скрипта с выводом значений на экран ([рис. 18](#fig-018)).

|  |
| --- |
| Рисунок 18: Результат работы скрипта rtt.py |

Продемонстрируем работу скрипта с выводом значений на экран. Добавим правило запуска скрипта в Makefile ([рис. 19](#fig-019)).

|  |
| --- |
| Рисунок 19: Добавление правила запуска скрипта в Makefile |

# 5. Выводы

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

# Список литературы