Моделирование сетей передачи данных

Лабораторная работа № 4. Эмуляция и измерение задержек в глобальных сетях

Демидова Екатерина Алексеевна

Содержание

# 1 Введение

**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.

**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 добавлению/изменению задержки, джиттера, значения корреляции для джиттера и задержки, распределения времени задержки в эмулируемой глобальной сети.
3. Реализуйте воспроизводимый эксперимент по заданию значения задержкив 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.

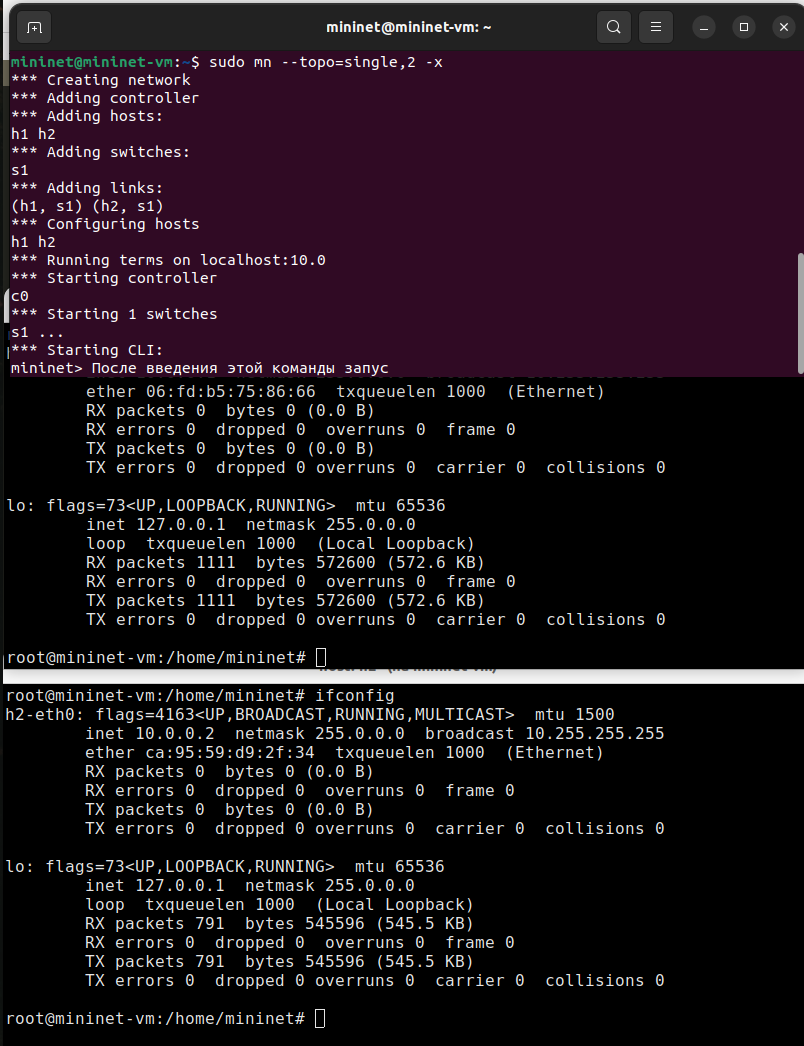
# 2 Теоретическое введение

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(ipconfig, ping) и даже получать доступ к терминалу. На коммутаторы можно добавлять различные правила и маршрутизировать трафик.

# 3 Выполнение лабораторной работы

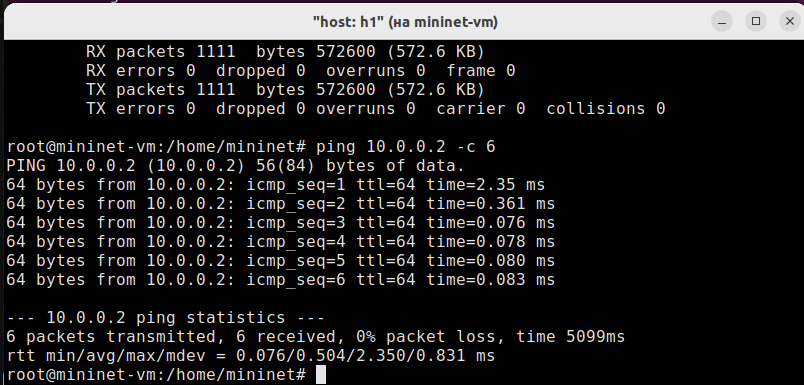
## 3.1 Запуск лабораторной топологии

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(рис. ??)



Простейшая сеть

Проверим подключение между хостами сети(рис. ??).

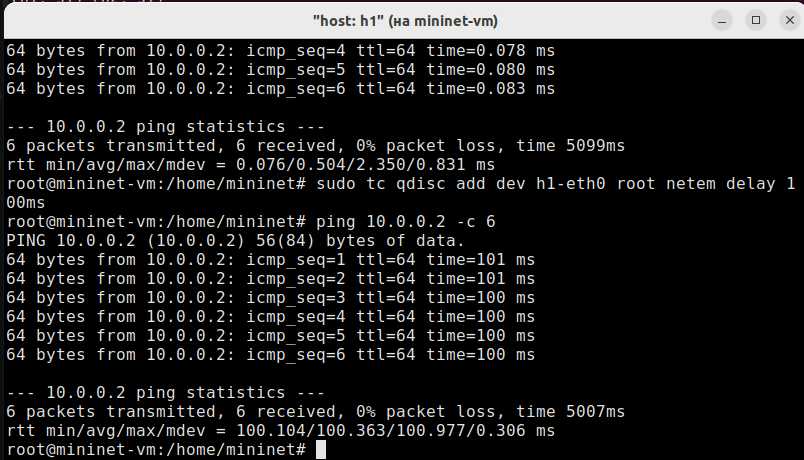


Проверка подключения

## 3.2 Добавление/изменение задержки в эмулируемой глобальной сети

В конце написано минимальное, среднее, максимальное и стандартное отклонение времени приёма-передачи (RTT): rtt min/avg/max/mdev = 0.076/0.504/2.350/0.831 ms.

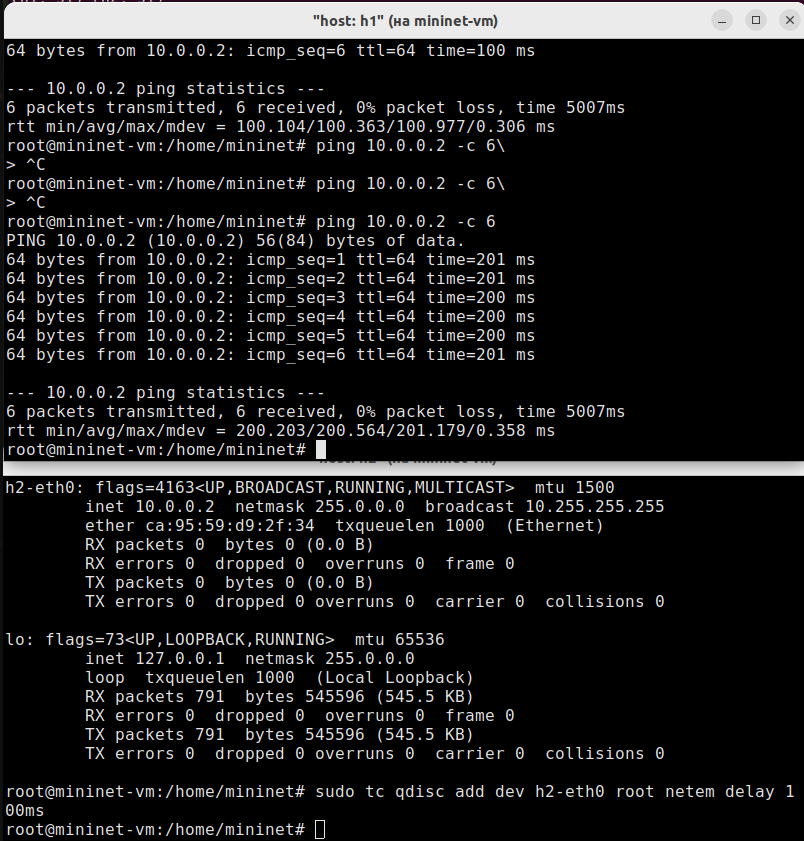
Зададим задержку на хосте h1 и проверим ее наличие(рис. ??).



Изменение задержки на хосте h1

Действительно, теперь rtt около 100: rtt min/avg/max/mdev = 100.104/100.363/100.977/0.306 ms.

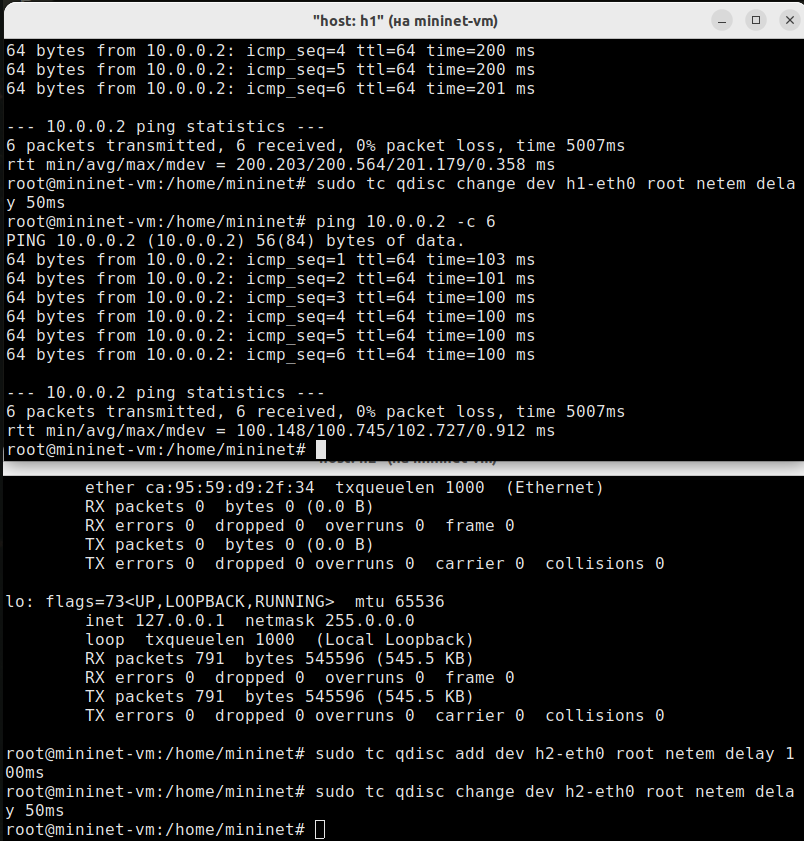
Добавим задержку на хост h2 и увидим, что теперь rtt около 200 rtt min/avg/max/mdev = 200.203/200.564/201.179/0.358 ms(рис. ??).



Задержка на обоих хостах

## 3.3 Изменение задержки в эмулируемой глобальной сети

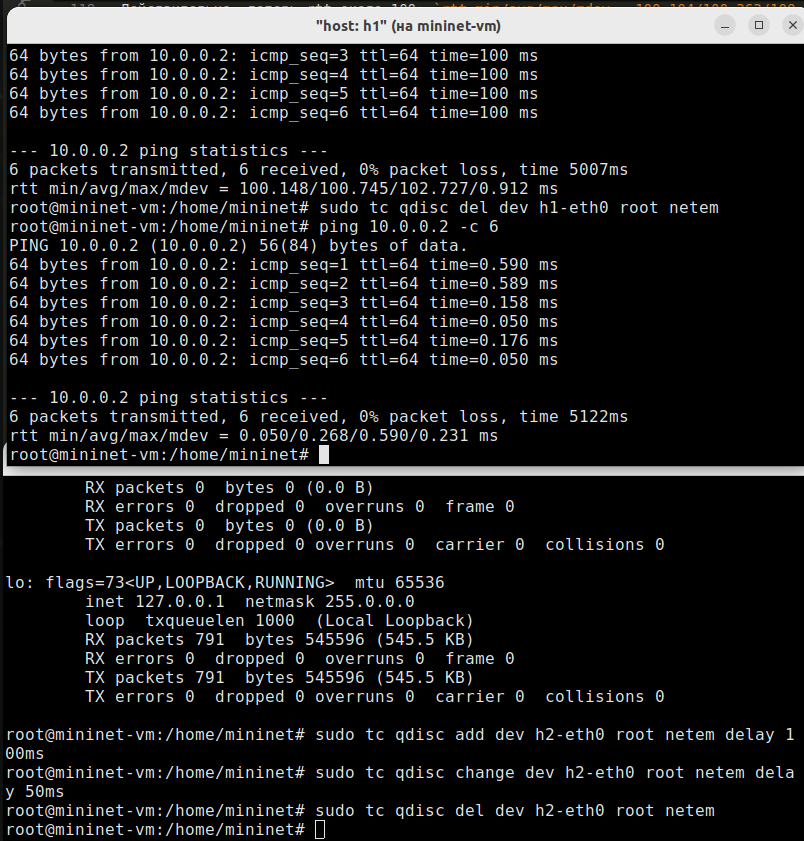
Изменим задержку на хостах на 50 мс, при этом у нас rtt min/avg/max/mdev = 100.148/100.745/102.727/0.912 ms(рис. ??)



Изменение задержки

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

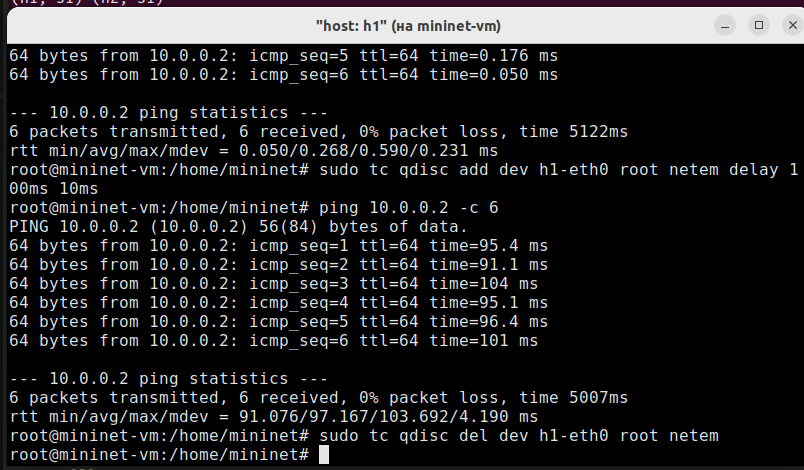
Вернем изначальное значение задержки и проверим корректность изменений rtt min/avg/max/mdev = 0.050/0.268/0.590/0.231 ms(рис. ??).



Восстановление исходных значений (удаление правил) задержки

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

Добавим на узле h1 задержку в 100 мс со случайным отклонением 10 мс и проверим изменение(рис. ??).

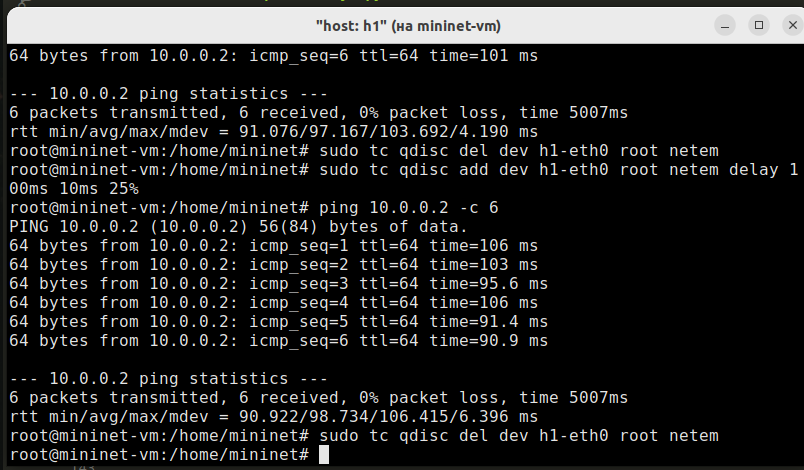


Добавление значения дрожания задержки в интерфейс

ВИдно, что rtt равняется ‘rtt min/avg/max/mdev = 91.076/97.167/103.692/4.190 m’.

## 3.6 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%(рис. ??).

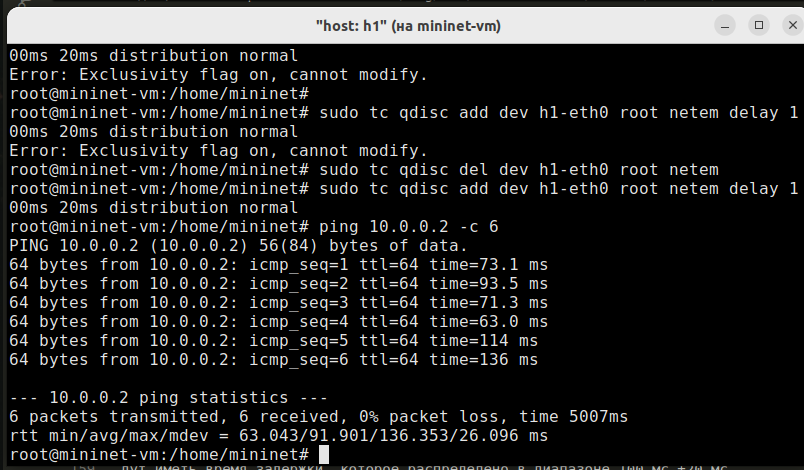


Добавление значения корреляции для джиттера и задержки

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%: rtt min/avg/max/mdev = 90.922/98.734/106.415/6.396 ms.

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

Задайте нормальное распределение задержки на узле h1 в эмулируемой сети(рис. ??).



Распределение задержки в интерфейсе подключения к эмулируемой глобальной сети

Убедимся, что все пакеты, покидающие хост h1 на интерфейсе h1-eth0, будут иметь время задержки, которое распределено в диапазоне 100 мс ±20 мс.

## 3.8 Воспроизведение экспериментов

В виртуальной среде mininet в своём рабочем каталоге с проектами создадим каталог simple-delay и перейдем в него. Создадим скрипт для эксперимента lab\_netem\_i.py:

#!/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()

В этом скрипте создается простейщая топология сети, затем с помощью комнанд, использованных нами ранее задается задержка в 100 мс 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.

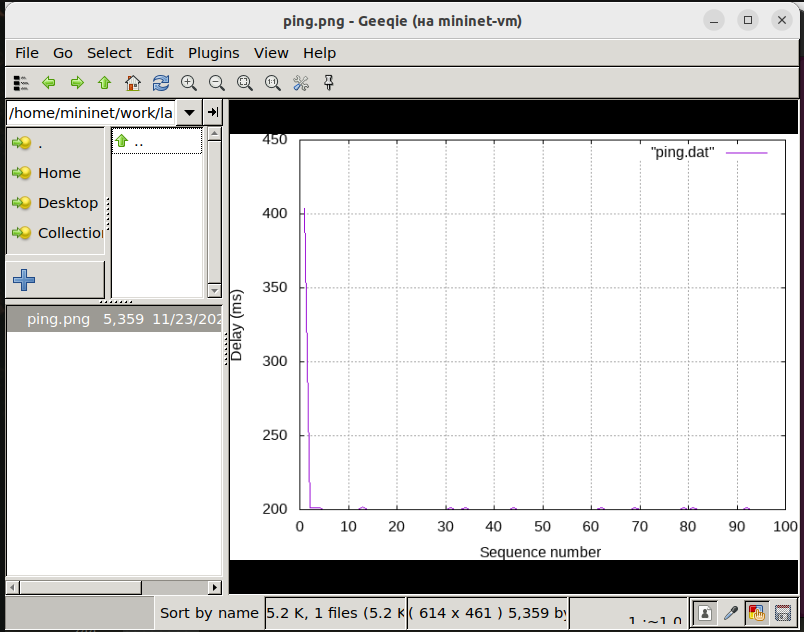
Создадим также скрипт для визуализации ping\_plot результатов эксперимента:

#!/usr/bin/gnuplot --persist  
set terminal png crop  
set output 'ping.png'  
set xlabel "Sequence number"  
set ylabel "Delay (ms)"  
set grid  
plot "ping.dat" with lines

Создадим Makefile для управления процессом проведения эксперимента:

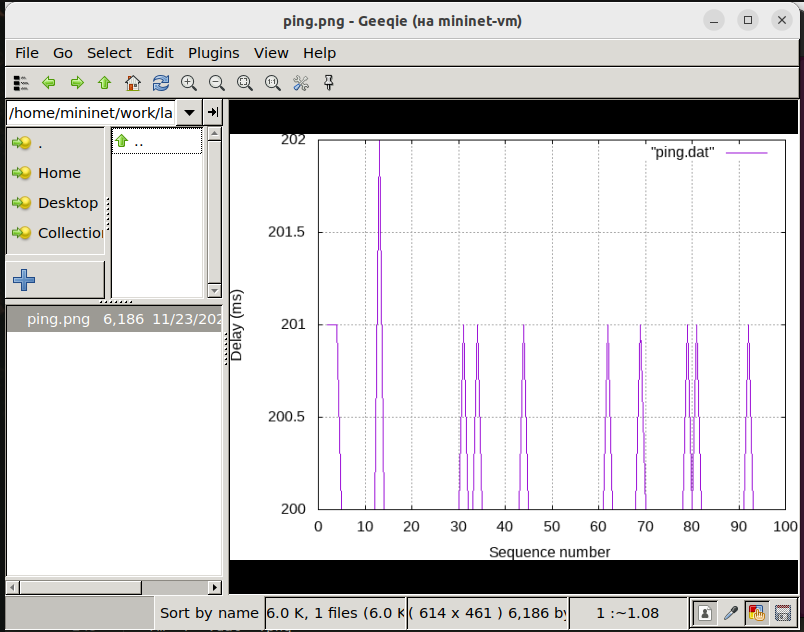
all: ping.dat ping.png  
  
ping.dat:  
 sudo python lab\_netem\_i.py  
 sudo chown mininet:mininet ping.dat  
  
ping.png: ping.dat  
 ./ping\_plot  
  
clean:  
 -rm -f \*.dat \*.png

В реузльтате получим график(рис. ??).



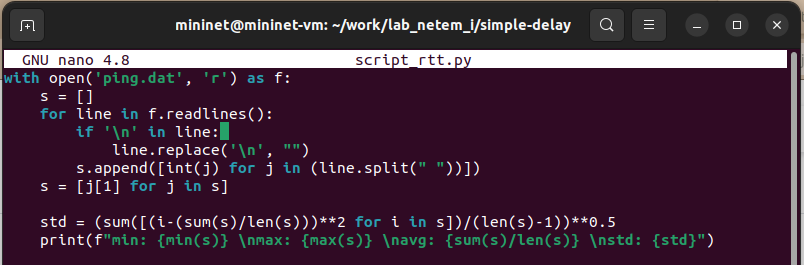
Визуализация эксперимента

Из файла ping.dat удалим первую строку и заново построим график(рис. ??).



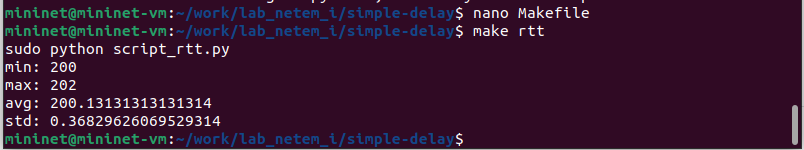
Визуализация эксперимента

Разработаем скрипт для вычисления на основе данных файла ping.dat минимального, среднего, максимального и стандартного отклонения времени приёма-передачи. Добавьте правило запуска скрипта в Makefile(рис. ??).



Скрипт

Запустим скрипт с помощью Makefile(рис. ??).



Вывод скрипта

# 4 Выводы

В результате выполнения работы познакомились с NETEM – инструментом для тестирования производительности приложений в виртуальной сети, а также получbkb навыкb проведения интерактивного и воспроизводимого экспериментов по измерению задержки и её дрожания (jitter) в моделируемой сети в среде Mininet.

# Список литературы

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