Лабораторная работа № 3

Измерение и тестирование пропускной способности сети. Воспроизводимый эксперимент

Беличева Дарья Михайловна

Содержание

# 1 Цель работы

Основной целью работы является знакомство с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получение навыков проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.

# 2 Задание

1. Воспроизвести посредством API Mininet эксперименты по измерению пропускной способности с помощью iPerf3.
2. Построить графики по проведённому эксперименту.

# 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.

iPerf3[**iperf?**] представляет собой кроссплатформенное клиент-серверное приложение с открытым исходным кодом, которое можно использовать для измерения пропускной способности между двумя конечными устройствами. iPerf3 может работать с транспортными протоколами TCP, UDP и SCTP:

* TCP и SCTP:
  + измеряет пропускную способность;
  + позволяет задать размер MSS/MTU;
  + отслеживает размер окна перегрузки TCP (CWnd).
* UDP:
  + измеряет пропускную способность;
  + измеряет потери пакетов;
  + измеряет колебания задержки (jitter);
  + поддерживает групповую рассылку пакетов (multicast).

# 4 Выполнение лабораторной работы

С помощью API Mininet создадим простейшую топологию сети, состоящую из двух хостов и коммутатора с назначенной по умолчанию mininet сетью 10.0.0.0/8.

В каталоге /work/lab\_iperf3 для работы над проектом создадим подкаталог lab\_iperf3\_topo и скопируем в него файл с примером скрипта mininet/examples/emptynet.py, описывающего стандартную простую топологию сети mininet (рис. 1).

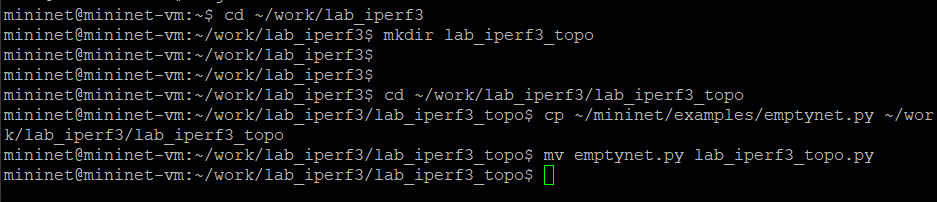


Рис. 1: Копирование файла emptynet.py

Изучим содержание скрипта lab\_iperf3\_topo.py (рис. 2).

В нем написан скрипт по созданию простейшей топологии из двух хостов h1 и h2, а также коммутатора s3 и контроллера c0. В начале файла видим импорт необходимых библиотек.

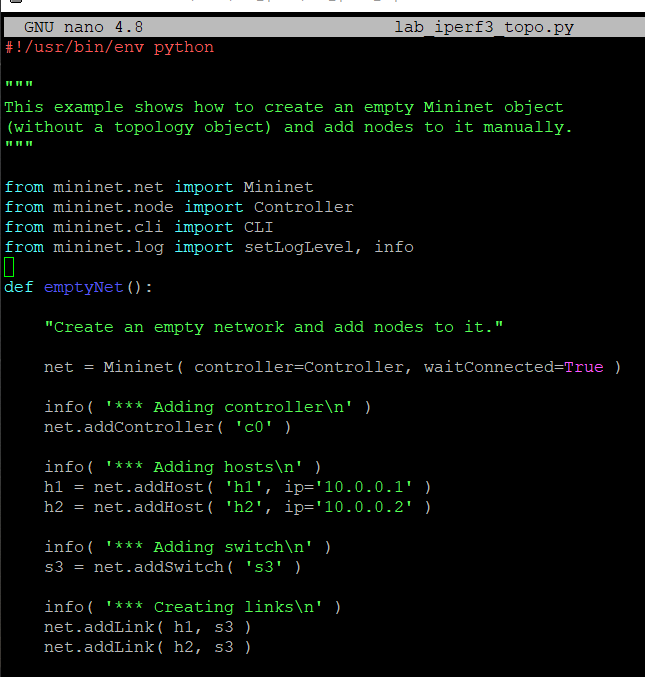


Рис. 2: Содержание файла lab\_iperf3\_topo.py

Основные элементы:

* addSwitch(): добавляет коммутатор в топологию и возвращает имя коммутатора;
* ddHost(): добавляет хост в топологию и возвращает имя хоста;
* addLink(): добавляет двунаправленную ссылку в топологию (и возвращает ключ ссылки; ссылки в Mininet являются двунаправленными, если не указано иное);
* Mininet: основной класс для создания и управления сетью;
* start(): запускает сеть;
* pingAll(): проверяет подключение, пытаясь заставить все узлы пинговать друг друга;
* stop(): останавливает сеть;
* net.hosts: все хосты в сети;
* dumpNodeConnections(): сбрасывает подключения к/от набора узлов;
* setLogLevel( ‘info’ | ‘debug’ | ‘output’ ): устанавливает уровень вывода Mininet по умолчанию; рекомендуется info.

Запустим скрипт создания топологии lab\_iperf3\_topo.py и посмотрим ее основные параметры (рис. 3).

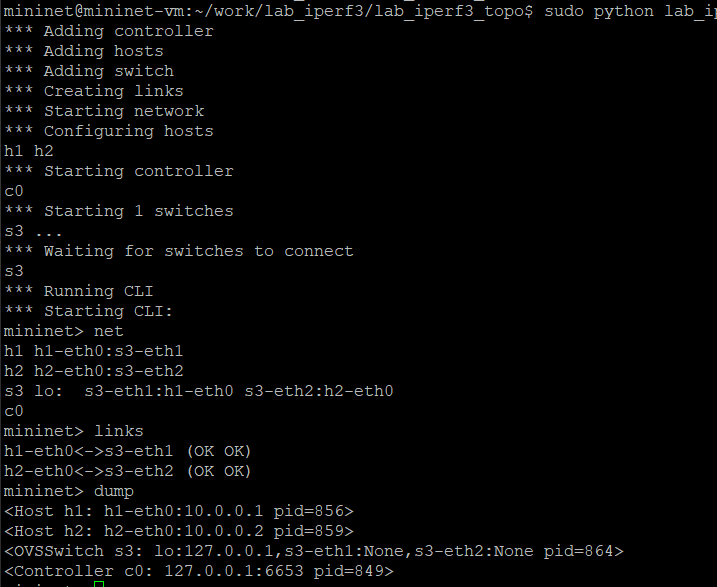


Рис. 3: Создание топологии и ее основные параметры

Внесем в скрипт lab\_iperf3\_topo.py изменение, позволяющее вывести на экран информацию обоих хостов сети, а именно имя хоста, его IP-адрес, MAC-адрес (рис. 4).

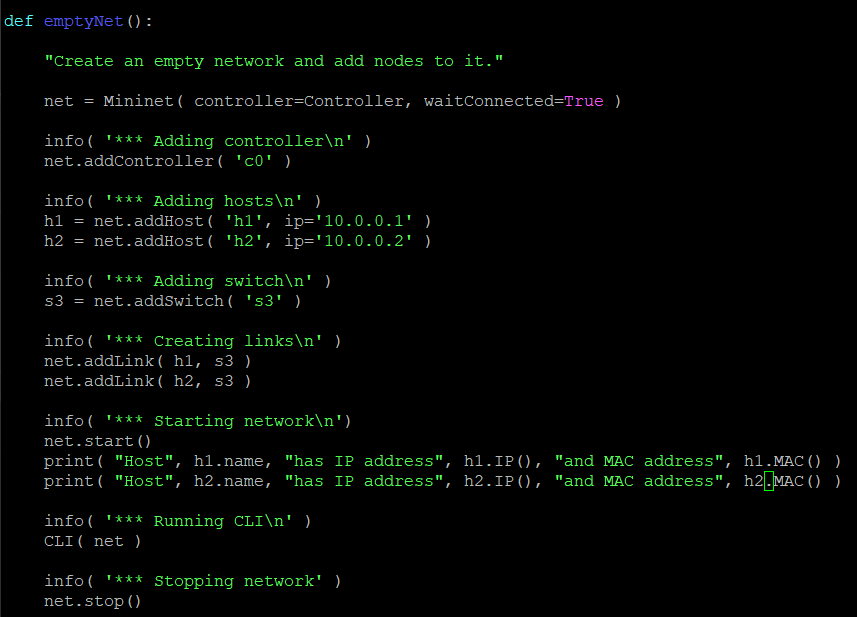


Рис. 4: Изменение скрипта lab\_iperf3\_topo.py

Здесь:

* IP() возвращает IP-адрес хоста или определенного интерфейса;
* MAC() возвращает MAC-адрес хоста или определенного интерфейса.

Проверим корректность отработки изменённого скрипта (рис. 5).

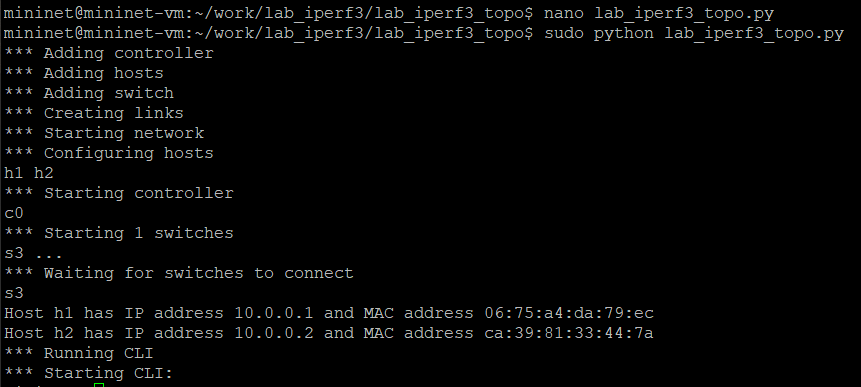


Рис. 5: Проверка работы внесенных изменений

Действительно, нам вывелась информация об IP и mac адресах хостов.

Mininet предоставляет функции ограничения производительности и изоляции с помощью классов CPULimitedHost и TCLink. Добавим в скрипт настройки параметров производительности (рис. 6).

В скрипте lab\_iperf3\_topo2.py изменим строку описания сети, указав на использование ограничения производительности и изоляции. Также измении функцию задания параметров виртуального хоста h1, указав, что ему будет выделено 50% от общих ресурсов процессора системы. Аналогичным образом для хоста h2 зададим долю выделения ресурсов процессора в 45%. В скрипте изменим функцию параметров соединения между хостом h1 и коммутатором s3. А именно добавим двунаправленный канал с характеристиками пропускной способности, задержки и потерь:

* параметр пропускной способности (bw) выражается числом в Мбит;
* задержка (delay) выражается в виде строки с заданными единицами измерения (например, 5ms, 100us, 1s);
* потери (loss) выражаются в процентах (от 0 до 100);
* параметр максимального значения очереди (max\_queue\_size) выражается в пакетах;
* параметр use\_htb указывает на использование ограничителя интенсивности входящего потока Hierarchical Token Bucket (HTB)

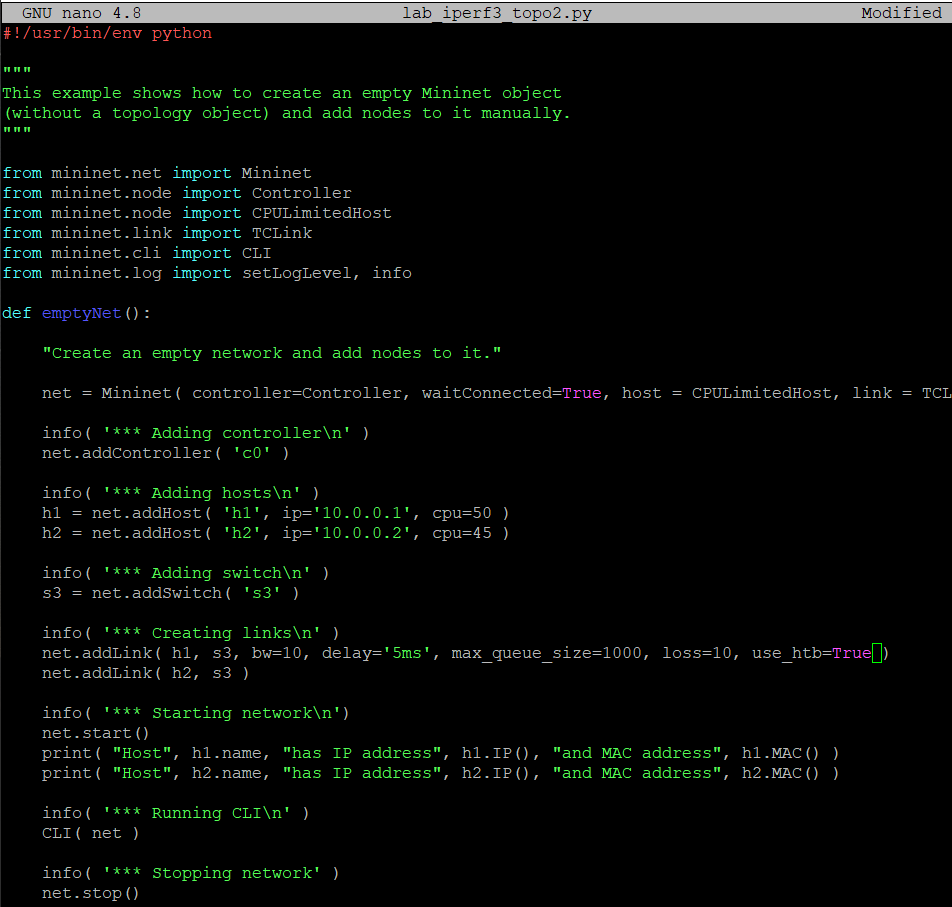


Рис. 6: Настройка параметров производительности

Запустим на отработку сначала скрипт lab\_iperf3\_topo2.py, затем lab\_iperf3\_topo.py и сравним результат (рис. 7). Увидим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.

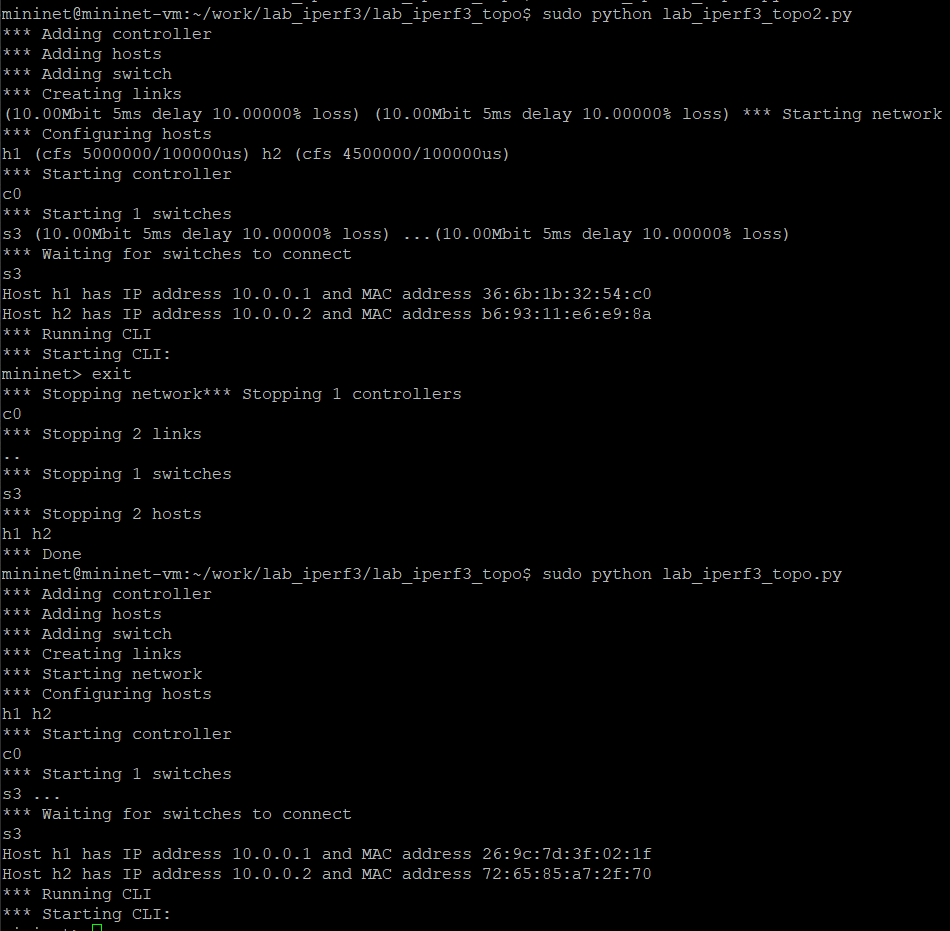


Рис. 7: Запуск скрипта с настройкой параметров производительности и без нее

Построим графики по проводимому эксперименту.

Сделаем копию скрипта lab\_iperf3\_topo2.py и поместим его в подкаталог iperf (рис. 8).

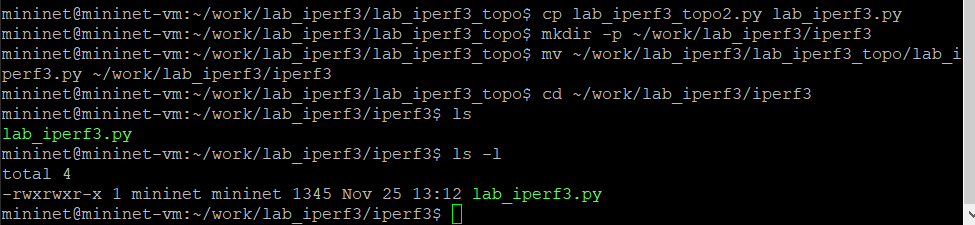


Рис. 8: Создание копии скрипта lab\_iperf3\_topo2.py

Изменим код в скрипте lab\_iperf3.py так, чтобы (рис. 9).:

* на хостах не было ограничения по использованию ресурсов процессора;
* каналы между хостами и коммутатором были по 100 Мбит/с с задержкой 75 мс, без потерь, без использования ограничителей пропускной способности и максимального размера очереди.
* После функции старта сети опишем запуск на хосте h2 сервера iPerf3, а на хосте h1 запуск с задержкой в 10 секунд клиента iPerf3 с экспортом результатов в JSON-файл, закомментируем строки, отвечающие за запуск CLI-интерфейса:

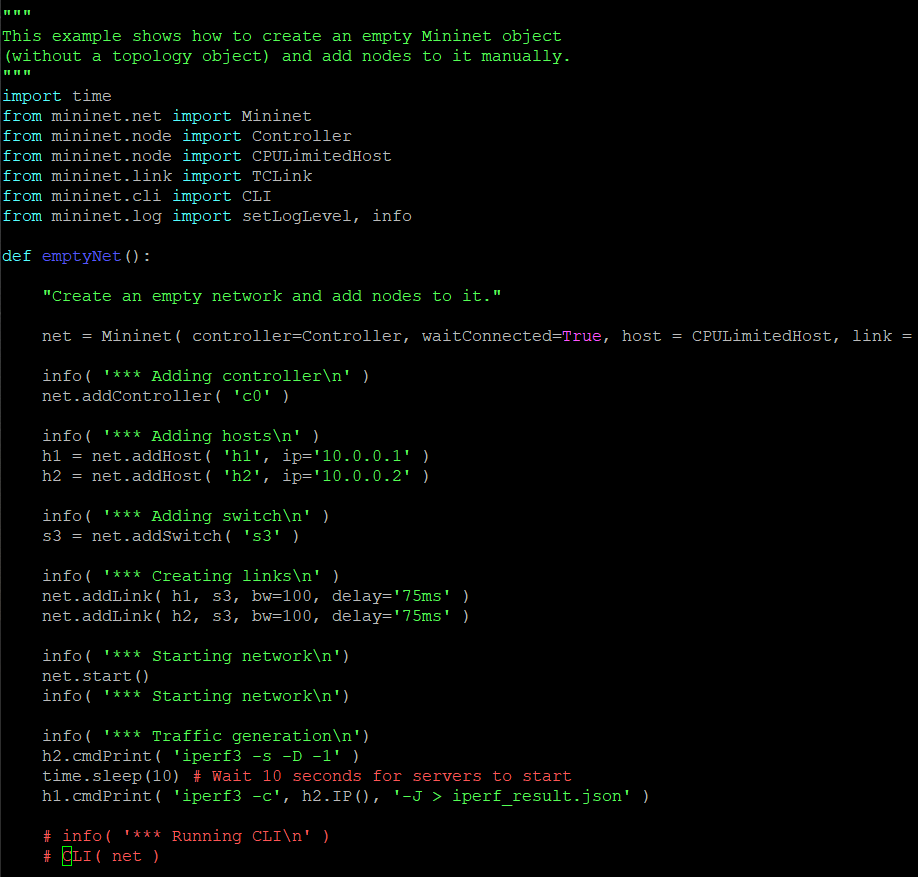


Рис. 9: Изменен ия кода в скрипте lab\_iperf3.py

Запустим на отработку скрипт lab\_iperf3.py (рис. 10).

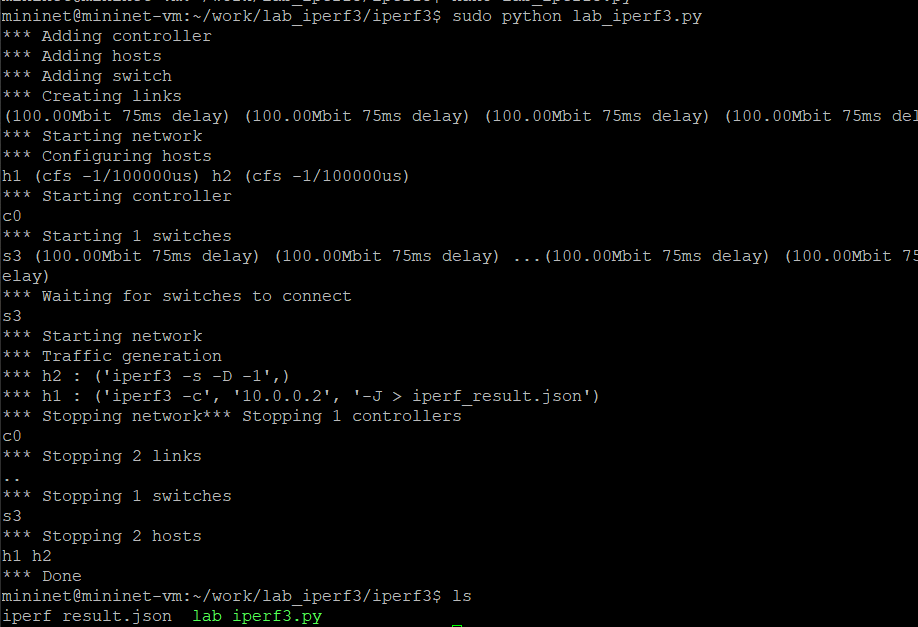


Рис. 10: Запуск скрипта lab\_iperf3.py

Построим графики из получившегося JSON-файла. Создадим Makefile для проведения всего эксперимента. В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (рис. 11).

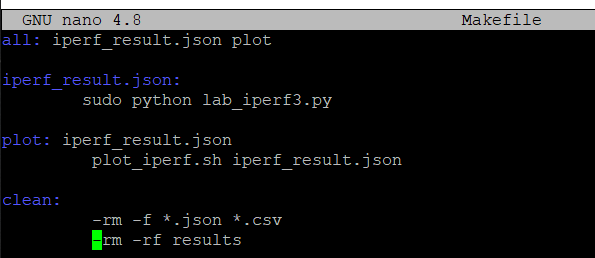


Рис. 11: Создание Makefile

Проверьте корректность отработки Makefile (рис. 12).

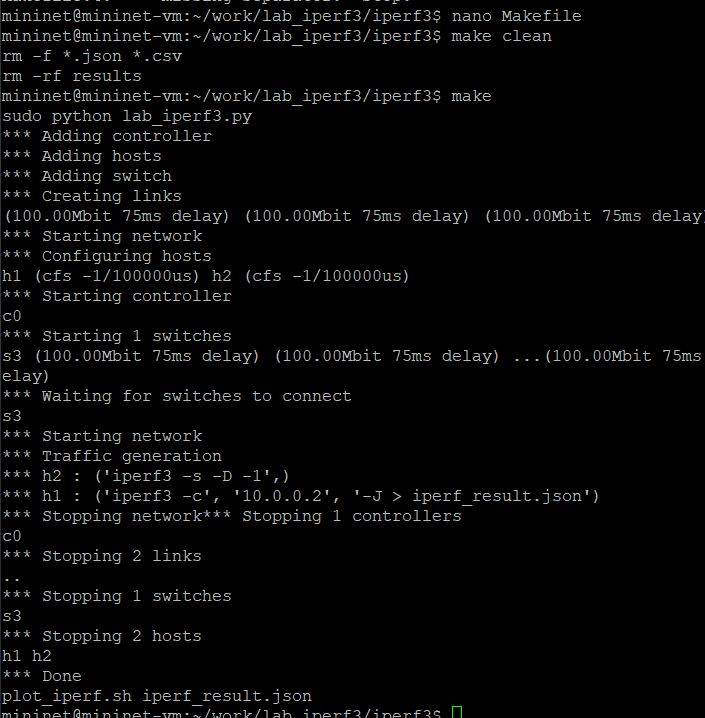


Рис. 12: Проверка работы Makefile

# 5 Выводы

В результате выполнения данной лабораторной работы я познакомилась с инструментом для измерения пропускной способности сети в режиме реального времени — iPerf3, а также получила навыки проведения воспроизводимого эксперимента по измерению пропускной способности моделируемой сети в среде Mininet.

# Список литературы