Отчёт по лабораторной работе №3

Моделирование сетей передачи данных

Ищенко Ирина Олеговна

Содержание

1 Цель работы **4**

2 Задание **5**

3 Выполнение лабораторной работы **6**

4 Выводы **16**

Список иллюстраций

3.1 Содержание файла lab_iperf3_topo.py 7
3.2 Создание топологии и ее основные параметры 8
3.3 Изменение скрипта lab_iperf3_topo.py 9
3.4 Проверка работы внесенных изменений 10
3.5 Настройка параметров производительности 11
3.6 Запуск скрипта с настройкой параметров производительности и без нее 12
3.7 Создание копии скрипта lab_iperf3_topo2.py 12
3.8 Изменен ия кода в скрипте lab_iperf3.py 13
3.9 Запуск скрипта lab_iperf3.py 14
3.10 Создание Makefile 14
3.11 Создание Makefile 15
3.12 Проверка работы Makefile 15

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 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.

Изучим содержание скрипта lab_iperf3_topo.py (Рисунок 3.1).

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.

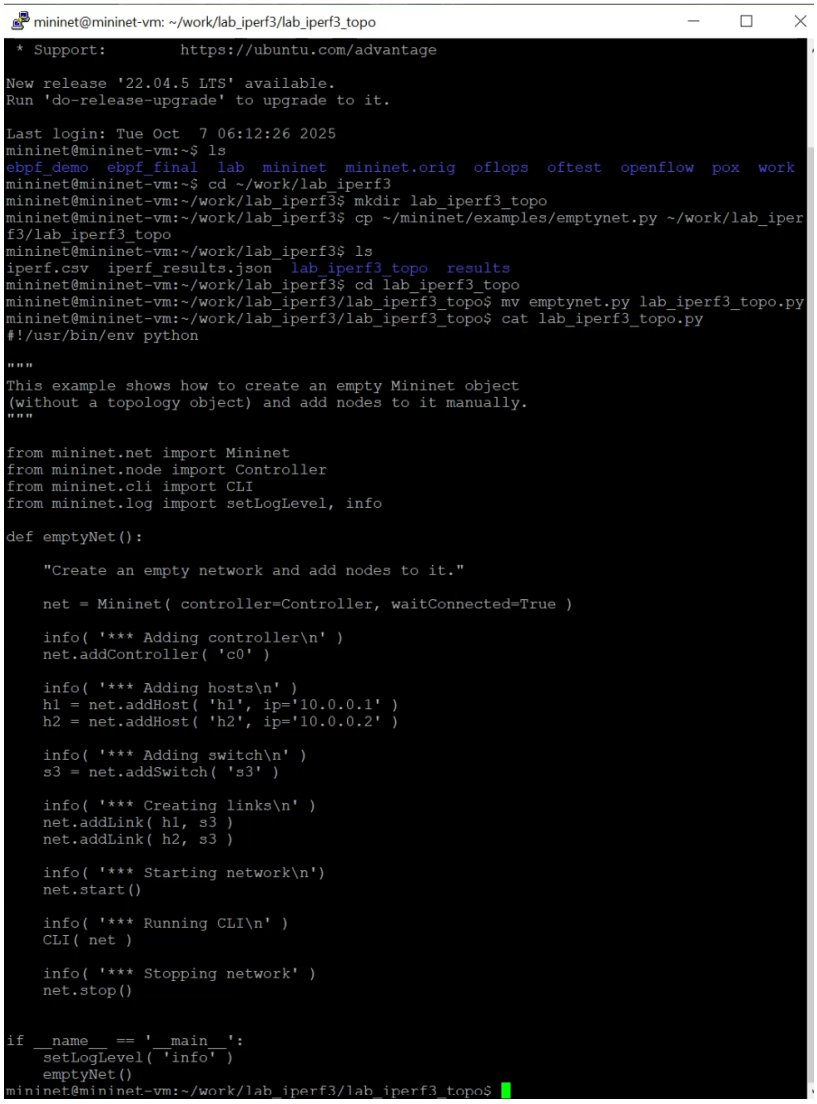

Рисунок 3.1: Содержание файла lab_iperf3_topo.py

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 (Рисунок 3.2).

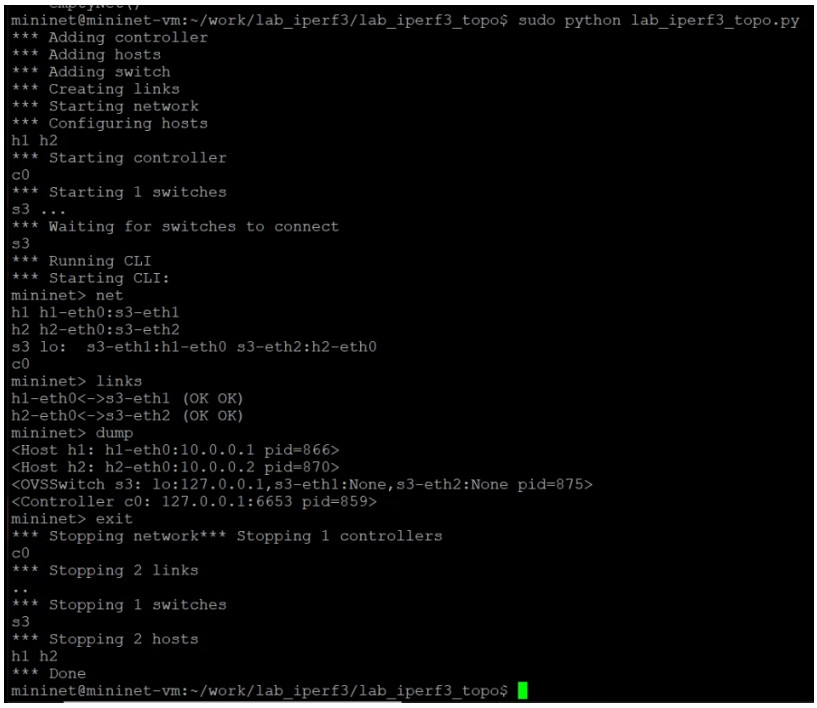

Рисунок 3.2: Создание топологии и ее основные параметры

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 (Рисунок 3.3).

Рисунок 3.3: 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 (Рисунок 3.4).

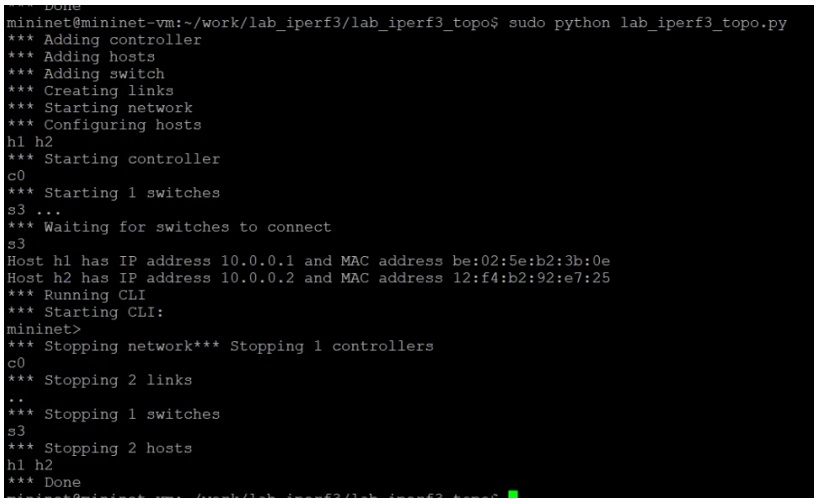

Рисунок 3.4: Проверка работы внесенных изменений

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 (Рисунок 3.5).

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)

Рисунок 3.5: Настройка параметров производительности

Запустим на отработку сначала скрипт lab_iperf3_topo2.py, затем lab_iperf3_topo.py и сравним результат (Рисунок 3.6). Увидим, что в первом случае у нас создалась сеть с настроенными параметрами, а во втором случае дефолтная сеть без этих параметров.

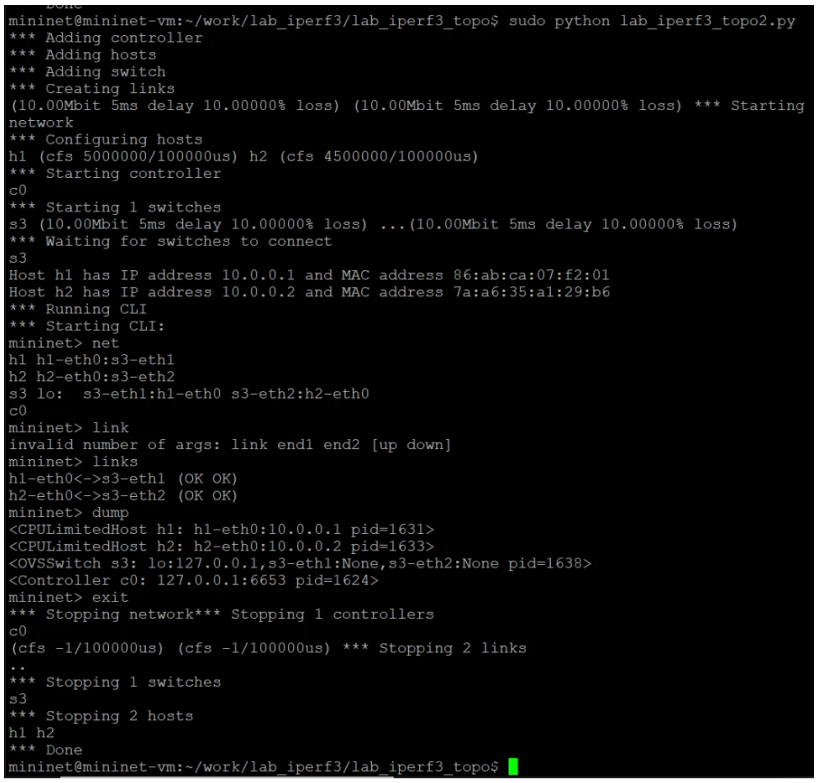

Рисунок 3.6: Запуск скрипта с настройкой параметров производительности и без нее

Построим графики по проводимому эксперименту.

Сделаем копию скрипта lab_iperf3_topo2.py и поместим его в подкаталог iperf (Рисунок 3.7).

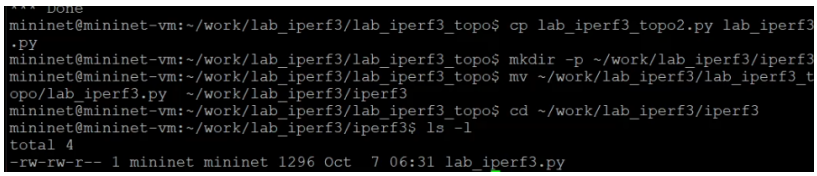

Рисунок 3.7: Создание копии скрипта lab_iperf3_topo2.py

Изменим код в скрипте lab_iperf3.py так, чтобы (Рисунок 3.8):

- 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:

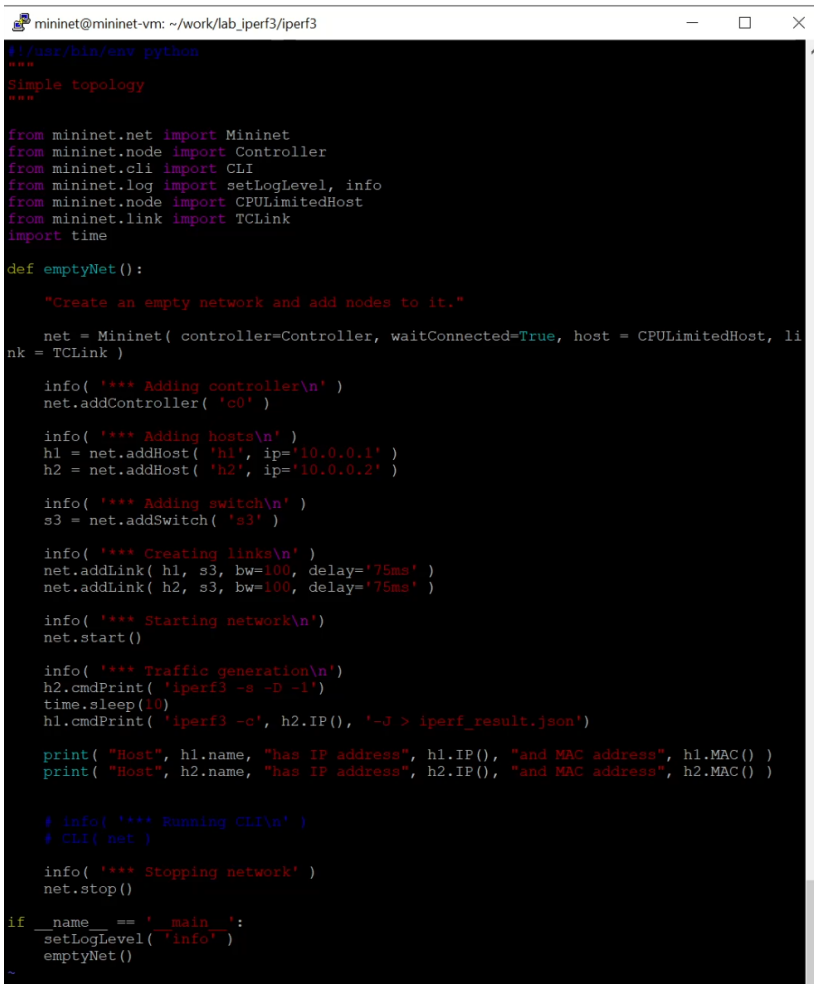

Рисунок 3.8: Изменен ия кода в скрипте lab_iperf3.py

Запустим на отработку скрипт lab_iperf3.py (Рисунок 3.9).

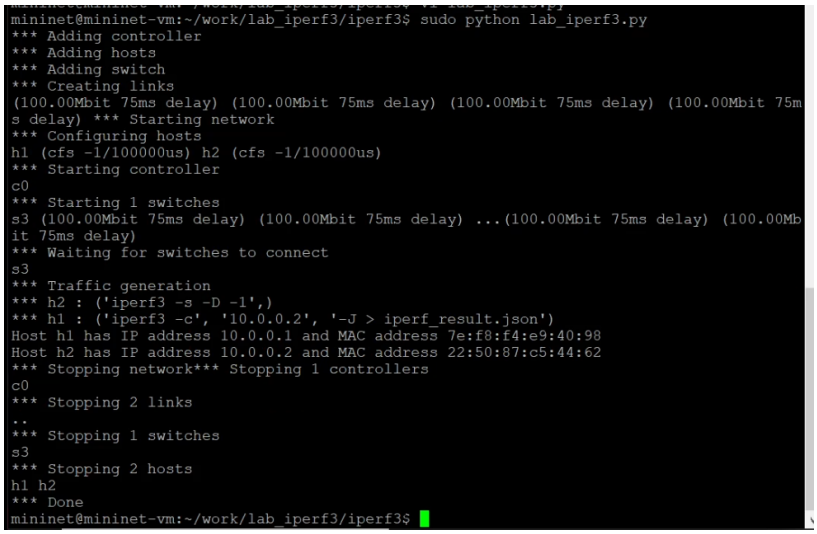

Рисунок 3.9: Запуск скрипта lab_iperf3.py

Построим графики из получившегося JSON-файла. Создадим Makefile для проведения всего эксперимента (Рисунок 3.10).

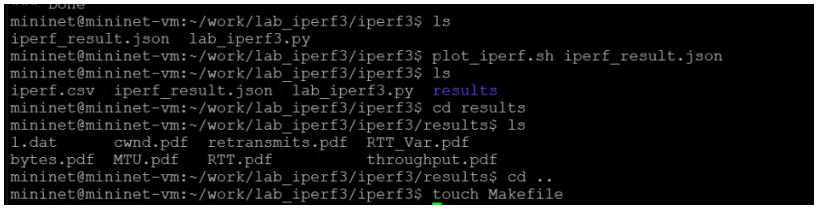

Рисунок 3.10: Создание Makefile

В Makefile пропишем запуск скрипта эксперимента, построение графиков и очистку каталога от результатов (Рисунок 3.11).

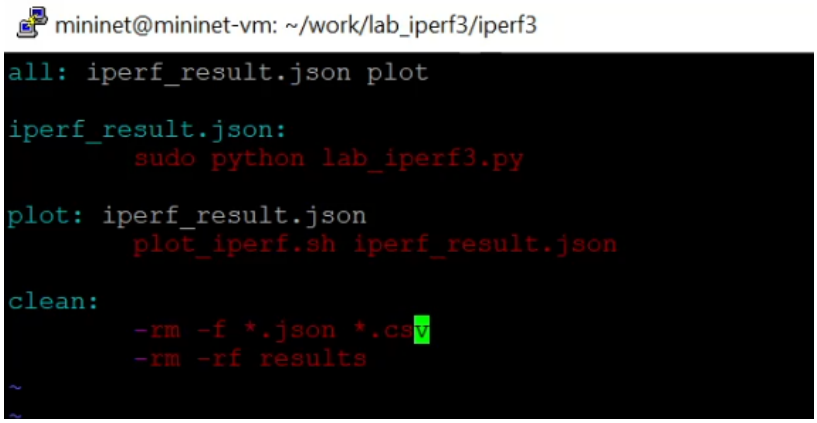

Рисунок 3.11: Создание Makefile

Проверьте корректность отработки Makefile (Рисунок 3.12).

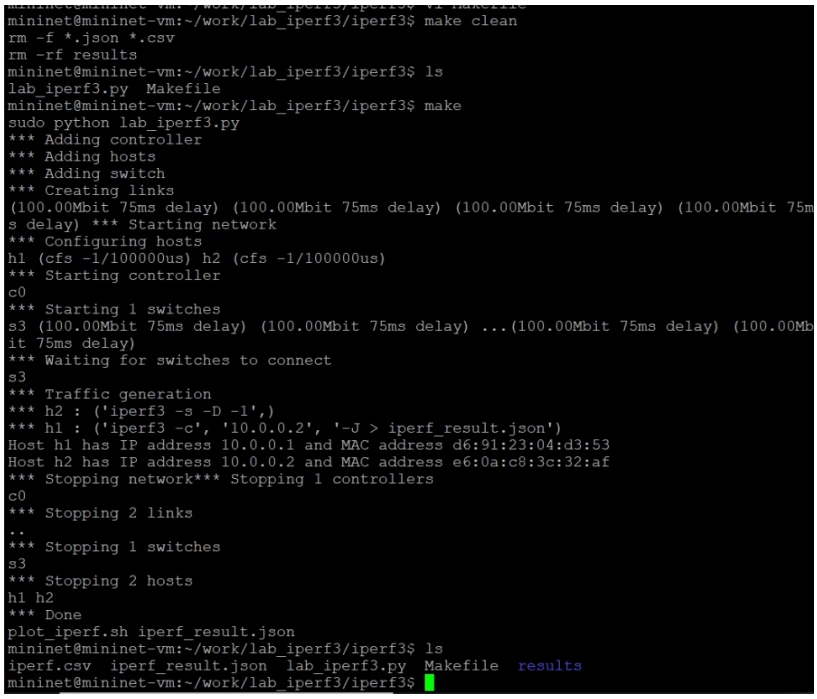

Рисунок 3.12: Проверка работы Makefile

4 Выводы

В ходе выполнения 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.