УДК 004.94

Автоматизация процессов моделирования и измерения сетевых характеристик в Mininet

С. М. Наливайко

Кафедра прикладной информатики и теории вероятностей
Российский университет дружбы народов
ул. Миклухо-Маклая, д.6, Москва, 117198, Россия

Email: 1032183644@rudn.ru

Среда виртуального моделирования Mininet позволяет использовать реальные сетевые приложения, сетевые протоколы и ядро Unix/Linux для тестирования и анализа характеристик моделируемых в ней компьютерных сетей и сетевых протоколов. Процесс моделирования в Mininet, указание на используемые при моделировании протоколы, задание параметров сети и протоколов и т. п. проводится с помощью команд терминала. Это достаточно рутинный процесс. В данной работе предложено описание разработанной программы для автоматизации как процесса задания параметров моделируемой сети, так и процесса обработки полученных результатов. Программа представляет собой набор модулей на языке программирования Python, причем каждый из модулей отвечает за свой круг задач, будь то модуль для отрисовки графиков или модуль проведения замеров на интерфейсе. Каждый из модулей самодостаточен и может быть использован отдельно.

Ключевые слова: информационные технологии, моделирование сетей, Mininet, автоматизация, измерение сетевых характеристик.

1. Введение

Создание сети не может обходиться без предварительного анализа и прототипирования. Для таких задач могут использоваться современные программы, которые позволяют создавать и испытывать сети без реальных сетевых компонентов. Такое решение дешево в построении, а сбор данных сетевых характеристик заметно ускоряется и упрощается. Есть несколько программных средств для моделирования сети, например gns3 [1], ns-3 [2], Cisco Packet Tracer [3] и др. Для измерения характеристик сети используются другие средства, например, Wireshark [4], Total Network Monitor 2 [5], Zabbix [6] и др. Выбрав средство моделирования для подготовки испытательного стенда и средство мониторинга сетевых характеристик можно приступать к анализу и прототипированию сети с заданными параметрами. Настройка сетевых компонентов вручную в среде симуляции представляет собой достаточно рутинный процесс.

Целью данной работы является построение средства автоматизации процессов моделирования и измерения сетевых характеристик передачи данных. В качестве среды моделирования была выбрана программа Mininet [7], так как она позволяет использовать реальные сетевые приложения, сетевые протоколы и ядро Unix/Linux для тестирования и анализа характеристик моделируемых в ней компьютерных сетей и сетевых протоколов, а в качестве программ, которые позволяют измерять сетевые характеристики, были выбраны iproute2 [8] и iperf3 [9].

2. Решение задачи автоматизации построения виртуальной сети и измерения сетевых характеристик

2.1. Недостаток процесса построения сети и ее тестирования

Mininet предоставляет исчерпывающий API [10] для создания виртуальной сети. API предоставлен для языка программирования Python, поэтому, чтобы приступить к работе, требуется с помощью программы pip установить пакет mininet. Установив библиотеку, ее можно импортировать в рабочий файл.

Сначала требуется создать топологию сети. Пример простой топологии, состоящей из двух хостов h1 и h2, а также соединяющих их двух коммутаторов s1 и s2, представлен на рис. 1.

Рис. 1. Пример топологии

Предполагается, что в данной сети будут действовать следующие правила:

— скорость передачи данных ограничена;
— максимальная пропускная способность соединения h1-s1 равна 100 Мбит/с;
— максимальная пропускная способность соединения s2-h2 равна 50 Мбит/с;
— на коммутаторе s2 стоит дисциплина обработки очередей FIFO с максимальным количеством пакетов, равным 30;
— потери в сети составляют 0.001%;
— задержка имеет нормальное распределение с математическим ожиданием в 30 мс и с дисперсией в 7 мс.
— в сети работает алгоритм для работы с перегрузками TCP Reno.

Тогда процесс создания и использования сети с данной топологией с помощью Python и Mininet-API будет иметь несколько этапов:

— описание класса топологии в файле;
— создание сети с помощью описанной топологии;
— старт сети;
— настройка сетевых компонентов из терминала;
— тестирование сети;
— остановка работы сети.

Предположим, что нам потребовалось поменять количество узлов, коммутаторов, сетевые характеристики или еще что-либо. Нам потребуется снова править код? Да, однако, можно автоматизировать данный процесс.

2.2. Автоматизация процесса создания сети

Метод автоматизации основан на создании конфигурационного toml-файла [11], из которого программа на языке программирования Python будет считывать данные и собственно учитывать изменения при настройке параметров топологии, мониторинговых характеристик.

Приведём пример toml-файла (см. листинг 1).

Листинг 1: Пример toml-файла с описанием топологии и настроек сети

```
1  # device settings
2  [devices]
3
4  [devices.h1]
5  name = "h1"
6  ip = "10.0.0.1"
7  cmd = [
```

```
8  "sysctl -w net.ipv4.tcp_congestion_control=reno"
9  ]
10
11 [devices.h2]
12 name = "h2"
13 ip = "10.0.0.2"
14 cmd = [
15 "sysctl -w net.ipv4.tcp_congestion_control=reno"
16 ]
17
18 # switch settings
19 [switches]
20
21 [switches.s1]
22 name = "s1"
23 [switches.s2]
24 name = "s2"
25
26 # link settings
27 [links]
28
29 pairs = [
30 ["h1", "s1"],
31 ["s1", "s2"],
32 ["s2", "h2"]
33 ]
34
35 cmd = [
36 "tc qdisc replace dev s1-eth2 root handle 10: tbf rate 100mbit burst
       50000 limit 150000",
37 "tc qdisc add dev s1-eth2 parent 10: handle 20: netem loss 0.001%
       delay 30ms 7ms distribution normal",
38 "tc qdisc replace dev s1-eth1 root handle 10: tbf rate 100mbit burst
       50000 limit 150000",
39 "tc qdisc add dev s1-eth1 parent 10: handle 20: netem loss 0.001%
       delay 30ms 7ms distribution normal",
40 "tc qdisc replace dev s2-eth2 root handle 10: tbf rate 50mbit burst
       25000 limit 75000",
41 "tc qdisc add dev s2-eth2 parent 10: handle 15: pfifo limit 30",
42 "tc qdisc replace dev s2-eth1 root handle 10: tbf rate 50mbit burst
       25000 limit 75000"
43 ]
```

В представленном toml-файле раздел **devices** отвечает за настройку конечных узлов сети, раздел **switches** — за настройку коммутаторов, а раздел **links** — за настройку соединений узлов сети и конфигурацию интерфейсов коммутаторов.

На хостах указывается ip-адрес, имя хоста и алгоритм работы с перегрузками. На коммутаторах прописывается только имя, однако, список настроек можно расширить, изменив программную логику в классе с топологией. В разделе **links** явно указывается, какие пары сетевых устройств соединяются, и команды, которые настраивают дисциплину очередей на интерфейсах. Подробнее о дисциплинах очередей можно прочесть в [12].

Имея подобный конфигурационный toml-файл, его можно прочесть с помощью средств Python, обработать и положить требуемые значения в объекты. Такой подход позволяет строить сколь угодно большие топологии без правки логики приложения.

2.3. Автоматизация процесса мониторинга

Вернемся к toml-файлу и добавим параметры для класса мониторинга (см. листинг 2).

Листинг 2: Пример toml-файла с описанием мониторинга сетевых характеристик

```
[monitoring]
monitoring_time = 30
monitoring_interval = 0.1
host_client = "h1"
host_server = "h2"
interface = "s2-eth2"
iperf_file_name = "iperf.json"
iperf_flags = ""
queue_data_file_name = "qlen.data"
plots_dir = "plots_dir"
plots_format = "pdf"
```

В представленном коде указаны все параметры, которые могут понадобиться для задачи анализа характеристик сети:

- monitoring_time — время мониторинга сети;
- monitoring_interval — интервалы между замерами длины очереди;
- host_client — узел, который будет отправлять данные;
- host_server — узел, который будет принимать данные;
- interface — интерфейс, на котором будет мониторится размер очереди;
- iperf_file_name — имя файла с отчетом мониторинга iperf;
- iperf_flags — iperf-флаги клиента;
- queue_data_file_name — имя файла с отчетом мониторинга длины очереди;
- plots_dir – директория со всеми графиками сетевых характеристик.

Мониторинг запускается после старта сети, а так как у нас отслеживаются как сетевые характеристики (пропускная способность, задержки, повторная отправка пакетов и т. д.), так и размер длины очереди на интерфейсе коммутатора, то следует пустить 2 параллельных потока под эти нужды. Эти и прочие детали реализации класса мониторинга не рассмотрены в данной работе, однако, полный исходный код можно найти по адресу github.com/nalivaykosergey/net_automatic_monitoring.

2.4. Объединение автоматизированных модулей

Имея готовый класс топологии и класс мониторинга, можно создать объекты этих классов, запустить сеть, включить мониторинг сети, построить графики и так далее. Диаграмма активностей для данного приложения представлена на рис. 2.

Диаграмма классов разработанного комплекса программ представлена на рис. 3.

Главным классом, включающим в себя все остальные, является CustomModel. В методе **simulation** создаются объекты классов Monitor, mininet.net.Mininet, CustomTopology и NetStatsPlotter.

Рассмотрим подробнее классы, создаваемые в методе **simulation**:

- mininet.net.Minet — предоставляемый Mininet API класс, отвечающий за создание сети с топологией, указанной в CustomTopology;
- CustomTopology — класс топологии сети;
- Monitor — класс, в котором происходят замеры сетевых характеристик исследуемой сети;
- NetStatsPlotter — класс, объект которого занимается построением графиков сетевых характеристик.

Точка входа в комплекс программ находится в файле **main.py**. Запустить данный скрипт можно с помощью команды

```
sudo ./main.py -c config/pfifo_config.toml
```

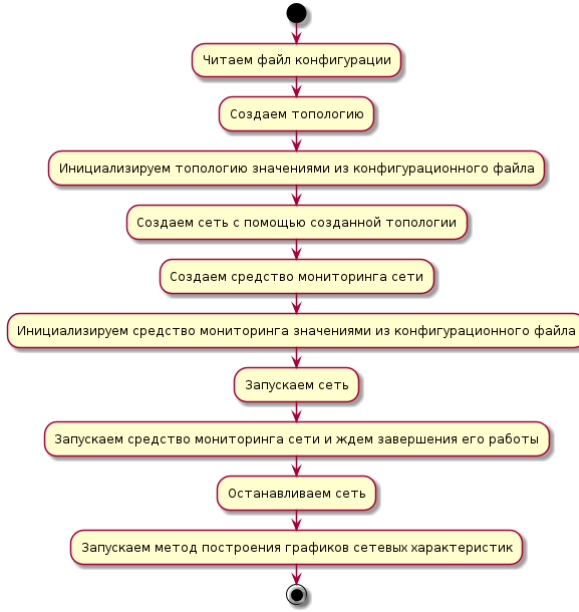

Рис. 2. Диаграмма активностей для программы

Параметр **-c** отвечает за местоположение конфигурационного файла.

Естественно, перед запуском скрипта требуется задать право на исполнение файла. Делается это с помощью команды

```
chmod +x main.py
```

После запуска в каталоге приложения появится директория с именем, которое было указано в toml-файле. В ней содержатся графики изменения сетевых характеристик и сырые данные, которые были обработаны объектом класса NetStatsPlotter. Например, график изменение длины очереди на интерфейсе s2-eth2 приведен на рис. 4. Видно, что длина очереди не превышает размера 30 пакетов, что явно было указано в настройках сети.

Процесс создания сети и мониторинга сетевых характеристик полностью автоматизирован и не требует более вмешательств в программный код.

3. Заключение

Анализ работоспособности и производительности сети — неотъемлемая часть работы сетевых инженеров. Современные программы, такие как Mininet, предоставляют разработчикам быстрый и дешевый в построении испытательный полигон, в котором можно проектировать и отлаживать сетевые программы.

Решение по автоматизации, которое было предложено в ходе исследования, ускоряет процесс перехода с создания сети на анализ ее производительности. Имея несколько конфигурационных файлов, мы можем быстро менять настройки сети и анализировать ее производительность одной командой, не исправляя исходный код программы.

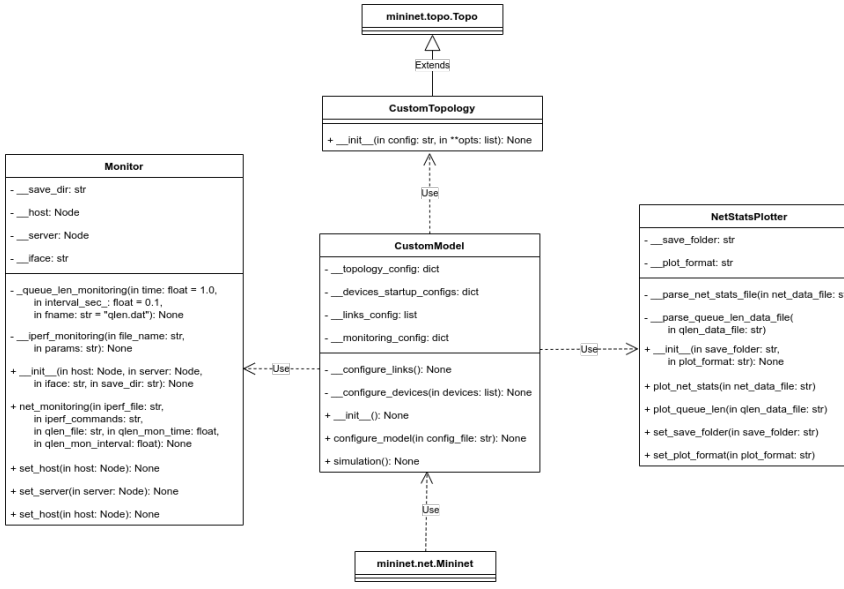

Рис. 3. Диаграмма классов приложения

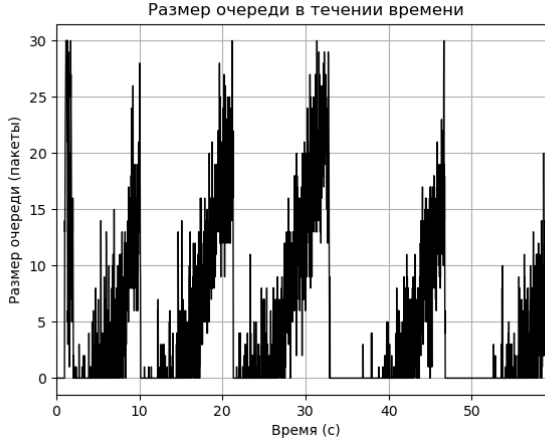

Рис. 4. График изменения длины очереди на интерфейсе s2-eth2

Литература

1. gns3. — URL: https://www.gns3.com.
2. ns-3. — URL: https://www.nsnam.org.
3. Cisco Packet Tracer. — URL: https://www.netacad.com/courses/packet-tracer.
4. Wireshark. — URL: https://www.wireshark.org.
5. Total Network Monitor 2. — URL: https://www.total-network-monitor.ru.
6. Zabbix. — URL: https://www.zabbix.com.
7. Mininet. — URL: http://mininet.org/.
8. iproute2. — URL: https://en.wikipedia.org/wiki/Iproute2.
9. iPerf — the ultimate speed test tool for TCP, UDP and SCTP. — URL: https://iperf.fr/iperf-doc.php.
10. Mininet Python API Reference Manual. — URL: http://mininet.org/api/annotated.html.
11. TOML. — URL: https://toml.io/en/.
12. tc(8) — Linux manual page. — URL: https://man7.org/linux/man-pages/man8/tc.8.html.

UDC 004.94

Automation of Mininet modeling and network performance measurement processes

S. M. Nalyvaiko

Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
Miklukho-Maklaya St., 6, Moscow, 117198, Russian Federation

Email: 1032183644@rudn.ru

The Mininet virtual simulation environment allows to use some real network applications, network protocols, and the Unix/Linux kernel to test and analyze characteristics of computer networks and network protocols modeled in it. The modeling process in Mininet, indication of the protocols used in the simulation, setting network and protocol parameters, etc., is carried out by using terminal commands. It's a routine process. In this paper, the description of the program for automating both the process of setting the parameters of the simulated network and the process of processing the obtained results is proposed. The program is the set of modules in the Python programming language, and each of the modules is responsible for its own range of tasks, whether it is a module for drawing graphs or a module for taking measurements on the interface. Each of the modules is self-sufficient and can be used separately.

Key words and phrases: information technology, network modeling, Mininet, automation, performance measurement.