Автоматизация процессов моделирования и измерения

сетевых характеристик в Mininet

Здравствуйте, уважаемые участники конференции. Меня зовут Наливайко Сергей. Я студент 4 курса Российского Университета дружбы народов, кафедры прикладной информатики и теории вероятностей. Тема моего доклада: Автоматизация процессов моделирования и измерения сетевых характеристик в Mininet

**Слайд 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.

Целью данной работы является построение средства автоматизации процессов моделирования и измерения сетевых характеристик передачи данных. В качестве среды моделирования была выбрана программа Mininet [1], 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 [2] и iperf3 [3].

**Слайд 2. Недостаток процесса построения сети и ее тестирования**

Mininet предоставляет исчерпывающий API [4] 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.

**Слайд 4. Недостаток процесса построения сети и ее тестирования**

Характеристики создаваемой сети отображены на слайде

**Слайд 3. Недостаток процесса построения сети и ее тестирования**

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.

Рисунок 1. Топология

**Слайд 5. Недостаток процесса построения сети и ее тестирования**

Мы можем написать код программы, который будет включать в себя топологию сети, способы взаимодействия с сетью, некоторые манипуляции с сетевыми элементами внутри сети. Если в ходе работы потребуется произвести изменения сети, то без правки исходного кода не обойтись. Однако, мы можем автоматизировать процесс создания сети и измерения сетевых характеристик.

**Слайд 6. Автоматизация процесса создания сети**

Метод автоматизации основан на создании конфигурационного toml-файла [5], из которого программа на языке программирования Python будет считывать данные и учитывать изменения при настройке параметров топологии, мониторинговых характеристик.

**Слайд 7. Автоматизация процесса создания сети**

Листинг конфигурационного файла.

В листинге 1 приведен пример создаваемого toml-файла, из которого считываются данные конфигурации сети и мониторинговых модулей.

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.

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.

**Слайд 8. Автоматизация процесса мониторинга**

Пример toml-файла с описанием мониторинга сетевых характеристик

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.

**Слайд 9. 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.

**Слайд 10. Объединение автоматизированных модулей**

РИС. 2

**Слайд 11. Объединение автоматизированных модулей**

РИС. 3

**Слайд 12. Объединение автоматизированных модулей**

Главным классом, включающим в себя все остальные, является CustomModel. В

методе simulation создаются объекты классов Monitor, mininet.net.Mininet, Custom-

Topology и NetStatsPlotter.

Рассмотрим подробнее классы, создаваемые в методе simulation:

— mininet.net.Minet — предоставляемый Mininet API класс, отвечающий за со-

здание сети с топологией, указанной в CustomTopology;

— CustomTopology — класс топологии сети;

— Monitor — класс, в котором происходят замеры сетевых характеристик иссле-

дуемой 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

**Слайд 13. Объединение автоматизированных модулей**

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.

**Слайд 14. Объединение автоматизированных модулей**

РИС. 4

Видно, что длина очереди не превышает размера 30 пакетов, что явно было указано в настройках сети.

**Слайд 15. 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.

**Слайд 16. Литература**

1. Mininet. —URL: <http://mininet.org/>.
2. iproute2. —URL: <https://en.wikipedia.org/wiki/Iproute2>.
3. iPerf — the ultimate speed test tool for TCP, UDP and SCTP.—URL: <https://iperf.fr/iperf-doc.php>.
4. Mininet Python API Reference Manual. — URL: <http://mininet.org/api/annotated.html>.
5. TOML.—URL: <https://toml.io/en/>.

На этом у меня все, спасибо за внимание.