## Моделирование в Mininet

Качество работы сетевого приложения определяется, в значительной степени, качествами линии передачи данных, сетевых устройств и работающих в них алгоритмах обработки потока данных. Современные сети передачи данных не могут обеспечить нужный уровень качества обслуживания по ряду причин. Одна из них – слабая взаимосвязь физических сетевых элементов разных уровней. Контроль передачи данных лежит на каждом устройстве отдельно, что, в некоторых случаях, ведет к многочисленным потерям данных и задержкам. Консорциум ONF (Open Networking Foundation) [1] предложил решение данной проблемы: отделить уровень контроля передачи данных от сетевого устройства и перенести все эти заботы на некоторый контроллер. Управление данным контроллером ведется централизовано с помощью протокола OpenFlow [2], что помогает установить нужный уровень предоставления услуг для конкретного приложения. Такое устройство сети консорциум ONF назвал SDN (Software-defined Networking) [3]. Для изучения производительности сетей SDN методом моделирования организация ONF разработала эмулятор сети Mininet.

В работе [4] речь идет об имитационном моделировании сети SDN с помощью средств Mininet и измерении производительности сети. Были затронуты следующие показатели производительности: RTT (Round Trip Time) [5] для каждого направления связи, пропускная способность ветвей и направлений связи, величина задержки на сетевых элементах, загрузка портов OpenFlow Switch, элементы сети с наибольшей задержкой, число обслуженных и потерянных пакетов. Mininet хорошо подходит для задач имитационного моделирования и исследований общей эффективности работы сети, однако, для создания сети требуется умение писать программы и знание Mininet API.

Работа [6] представляет собой подробное руководство по взаимодействию с Mininet, как с помощью CLI [7], так и с помощью написания программ на языке программирования Python. Авторы показывают простоту создания сетевых топологий различной сложности, такие как Minimal, Single, Linear, Tree, Reversed. Также, в работе представлена программа, которая создает собственную сеть внутри Mininet и проводит простой анализ ее производительности. Созданная сеть детерминирована и состоит всего из 3 узлов, соединенных коммутатором. Анализ работы сети не имеет смысла, так как сеть создана, скорее, для демонстрации принципов работы с Mininet, а не для исследований. Из данной работы видно, что Mininet отлично подходит для исследований производительности сети и сетевых компонентов, благодаря своей простоте и настройке.

Идея автоматизации процесса создания топологий сети возникает сама собой, после написания двух-трех программ. Авторы работы [8] описывают способ такой автоматизации. Способ основан на создании конфигурационного файла, в котором описаны основные правила адресации, создания сетевых устройств и соединений между ними. Чтение конфигурационного файла происходит в программе, написанной на языке программирования Python. Данный способ заметно ускоряет развертывание сети, избавляя исследователя от постоянный модификаций программного кода. Помимо прочего, в данной работе поверхностно рассмотрены и другие среды моделирования сети передачи данных: NS2 [9], NS3 [10], OPNET [11], OMNeT++ [12]. Исследователь волен выбирать среду моделирования исходя из своих предпочтений, учитывая их особенность.

## Анализ производительности сети в Mininet

Простое моделирование сети не дает информации об ее производительности. Нужно провести качественный анализ работы сетевых компонентов, рассчитать пропускную способность соединения, количество потерянных и доставленных пакетов и т. д. Работа [15] является хорошим началом для изучения методов анализа производительности сети. Авторы исследуют задержки в сети при передаче данных и сравнивают показатели в HDN и SDN. HDN (Hardware Defined Network) – привычные нам сети, где управление передачи данных лежит на устройствах сетевого и канального уровней. Работа показала, что SDN справляется с работой лучше и средняя задержка передачи данных в ней ниже (3.891 мс против 8.277 мс). Однако, стоит заметить, что здесь речь идет о передаче простого ICMP пакета, а не потока TCP/UDP трафика.

Работа [13] является более интересным примером анализа производительности сети в Mininet. В ней рассматриваются вопросы оценки производительности механизмов для эффективной работы с перегрузки в SDN. Авторы оценивают общую производительность сети, сравнивая ее с производительностью сети, которая использует LLDP (Link Layer Discovery Protocol) [14]. Данный протокол канального уровня позволяет сетевому оборудованию оповещать оборудование, работающее в локальной сети, о своём существовании и передавать ему свои характеристики. Протокол отлично подходит для новой концепции построения сетей SDN, которую мы рассматривали ранее, так как позволяет SDN-контроллеру знать характеристики элементов сети и в зависимости от этого управлять потоками трафика. Оценка производительности основывается на трех пунктах:

1. Уровень потери пакетов;
2. Уровень доставки пакетов;
3. Общая пропускная способность.

Данные оценки определяются для каждой итерации исследования. Всего таких итераций 4, и они имеют следующие сетевые топологии:

1. 1 SDN-контроллер, 4 коммутатора, 1 хост, 4 соединения;
2. 1 SDN-контроллер, 15 коммутаторов, 16 хостов, 30 соединений;
3. 1 SDN-контроллер, 40 коммутаторов,81 хост, 120 соединений;
4. 1 SDN-контроллер, 85 коммутаторов, 256 хостов, 340 соединений.

В ходе работы были построен график, которые отображают зависимость сетевой характеристики от сетевой топологии для обычной сети и сети, которая использует LLDP. На графиках видно, что сеть второго типа показывает лучший уровень производительности для каждой сетевой характеристики передачи данных: ниже уровень потерь пакетов, выше уровень доставки пакетов, выше общая пропускная способность.

Mininet отлично подходит для проведения исследований поведенческих особенностей сетевых компонентов. Однако, можно также провести исследование работоспособности и производительности сетевых протоколов и приложений. Авторы статьи [16] исследуют производительность алгоритма для эффективной работы с перегрузками BBRv2. Анализ работы алгоритма, презентацию и исходный код можно найти в [17]. BBRv1 – нестандартный алгоритм управления перегрузками разработанный в Google, который не использует потерю пакетов как маркет для снижения скорости отправки. Алгоритм BBRv1 [21], в сравнении с предшествующими алгоритмами, выдает большую пропускную способность для потоков данных при равных условиях. Одним из минусов алгоритма является его слабая совместимость с более старыми алгоритмами, используемыми в сети, и алгоритм BBRv2 создан как раз для исправления данной бреши.

В исследовании средствами Mininet создается сеть, имеющая 100 узлов отправителей и 100 узлов получателей. Отправители и получатели связаны между собой сетью из 3-х коммутаторов, каждый из который имеет свою задачу:

- коммутатор 1: является точкой входа для узлов-отправителей, эмулирует потери и задержки данных с помощью средств NetEm [18], соединен с коммутатором 2;

- коммутатор 2: эмулирует «узкое горлышко» между отправителями и получателями, ограничивая скорость передачи данных до 1 Гбит/с. Ограничивание передачи осуществляется с помощью дисциплины очередей TBF [19].

- коммутатор 3: соединяет коммутатор 2 и хосты получатели.

В ходе работы были исследованы и сравнены алгоритмы CUBIC [22], BBRv1, BBRv2 на такие сетевые характеристики: пропускная способность, индекс справедливости [20], сосуществование. Результаты показывают, что BBRv2 обеспечивает лучшее сосуществование с потоками, использующие алгоритм CUBIC, по сравнению со своим предшественником. Кроме того, BBRv2 способен обеспечить более справедливую долю пропускной способности по сравнению с BBRv1, когда сетевые условия, такие как пропускная способность и задержка, динамически изменяются.

Литература

1. https://en.wikipedia.org/wiki/Open\_Networking\_Foundation

2. https://en.wikipedia.org/wiki/OpenFlow

3. <https://en.wikipedia.org/wiki/Software-defined_networking>

4. RESEARCH OF SDN NETWORK PERFORMANCE PARAMETERS USING MININET NETWORK EMULATOR

5. https://en.wikipedia.org/wiki/Round-trip\_delay <https://en.wikipedia.org/wiki/Round-trip_delay>

6. Mininet as Software Defined Networking Testing Platform

7. <https://en.wikipedia.org/wiki/Command-line_interface>

8. Implementation of simplified custom topology framework in Mininet

9. <https://www.isi.edu/nsnam/ns/>

10. <https://www.nsnam.org/>

11. <https://opnetprojects.com/opnet-network-simulator/>

12. <https://omnetpp.org/>

13. Performance Analysis of Congestion Control Mechanism in Software Defined Network

14. <https://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol>

15. Performance Analysis of a Software Defined Network Using Mininet

16. A Performance Evaluation of TCP BBRv2 Alpha

17. https://github.com/google/bbr/blob/v2alpha/README.md

18. <https://man7.org/linux/man-pages/man8/tc-netem.8.html>

19. <https://man7.org/linux/man-pages/man8/tc-tbf.8.html>

20. https://datatracker.ietf.org/doc/rfc5166/

21. <https://github.com/google/bbr/blob/master/Documentation/bbr-quick-start.md>

22. https://datatracker.ietf.org/doc/rfc8312/