электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Автоматизированная программная система моделирования SDN-сетей

 # 03, март 2015

Сурков Л. В.[1,*], Коломеец А. Е.[2]

УДК: 004.72

[1]Россия, МГТУ им. Н.Э. Баумана
[2]ОАО ОРКК
[*]leovsur@gmail.com

Введение

В процессе создания SDN-сети острым стоит вопрос об удобстве моделирования SDN-сетей. Немаловажным является время, затраченное на процесс построения топологии и конфигурации SDN-коммутаторов. Поэтому перспективным представляется разработка автоматизированной системы для быстрого построения подобных сетей с удобным пользовательским интерфейсом.

Для тестирования приложений контроллера можно использовать: физическое оборудование, виртуальное оборудование (эмуляция), моделирование. Первый подход обладает высокой степенью доверия, но при этом является крайне не удобным при экспериментировании (изменении топологии, кода, настроек). Эмуляция покрывает этот недостаток и при этом позволяет сэкономить на покупке оборудования. Но из-за того, что каждое устройство представляет из себя виртуальную машину, при больших сетевых топологиях требуются значительные вычислительные мощности. Самым оптимальным инструментом для тестирования является моделирование, которое использует низкое требование к ресурсам. В ходе исследования современных подходов моделирования SDN-сетей самой актуальной является среда моделирования MiniNet, разработанной Стэнфордским университетом. Однако, данная разработка требует высокой квалификации пользователя в области программирования, так как основной процесс моделирования выполняется в командной оболочке с невозможностью доступа к визуальной информации. Данный подход хоть и является лучшим на сегодняшний день, он используется узкоспециализированными специалистами. Для расширения предела пользования средой моделирования требуется обеспечить удобный пользовательский интерфейс с возможностью доступа к визуальной информации.

Анализ существующих методов создания SDN-сетей показывает, что у проприетарных сетей имеются существенные барьеры для экспериментирования и создания новых сервисов, поэтому самым актуальным решением является использование открытого протокола OpenFlow, созданного по инициативе Open Networking Foundation (открытого фонда сетевых технологий). В OpenFlow управление массивами данных происходит не на уровне отдельных пакетов, а на уровне потоков ими образуемых, тем самым повышается быстродействие сети.

Таким образом, на сегодняшний день перспективной является задача разработки автоматизированной программной системы для построения, моделирования и тестирования SDN-сетей на основе протокола OpenFlow.

Этапы разработки программной системы

В качестве архитектуры системы моделирования была выбрана клиент – серверная модель, так как пользователям для работы с системой моделирования достаточно иметь лишь выход в интернет и веб-браузер.

В качестве инструментария для разработки был использован фреймворк для веб – приложений Django, обладающий следующими достоинствами:

- высокая скорость разработки;
- шаблон проектирования MVC (модель-представление-контроллер);
- наличие большого количества готовых решений;
- тесная интеграция с Python-приложениями (в том числе с контроллером POX, и средой MiniNet, написанными также на Python).

За осуществление моделирования, хранения и модификации данных коммутаторов и контроллера отвечают модели (Model). Они предоставляют остальным компонентам приложения объектно-ориентированное отображение данных. Объекты модели осуществляют загрузку и сохранение данных в реляционной базе данных (SQLite). Контроллер (Controller) – это набор логики, отвечающей за вызов методов модели и запускающей формирование представления. Помимо этого, он реализует переходы между экранами системы, реагируя на соответствующие запросы пользователя. Основной задачей представления (View) является создание пользовательского интерфейса с использованием данных, полученных от контроллера. Представление также передает запросы клиента на изменение данных обратно в контроллер.

Создание автоматизированной системы включает в себя следующие этапы: проектирование структуры системы в целом и её отдельных компонентов, проектирование базы данных, создание пользовательского интерфейса и комплексное тестирование системы. В соответствии с выбранной архитектурой программного обеспечения, была спроектирована структурная схема ПО (рис. 1).

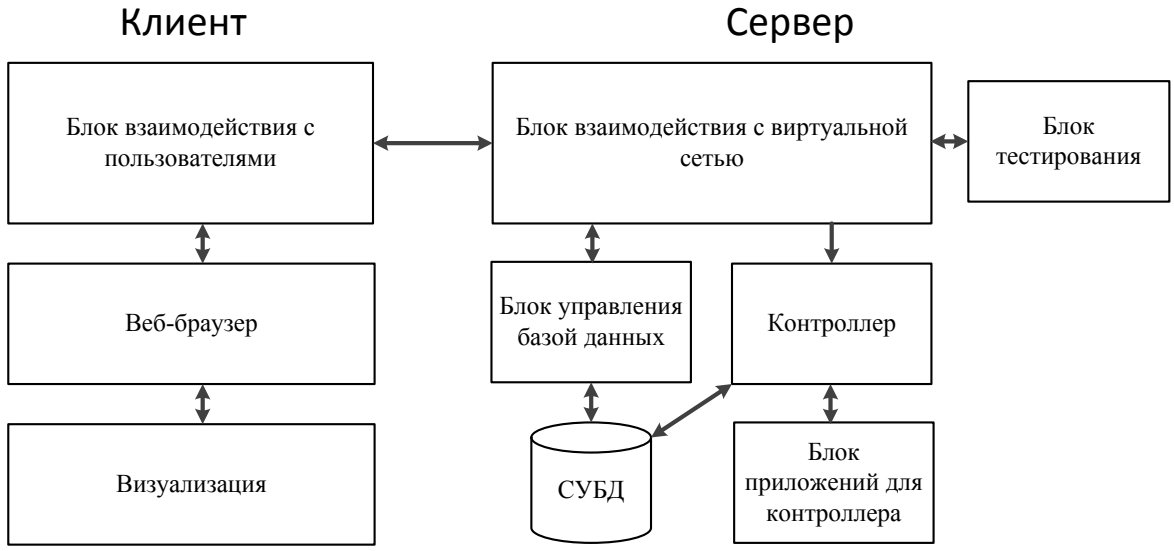

Рисунок 1 - Структурная схема программной системы

Основную функцию выполнения моделирования и хранения данных берёт на себя сервер. Применение такого подхода минимизирует требования к аппаратной составляющей клиентских рабочих машин, допуская применение даже «тонких» клиентов, однако быстродействие настоящей системы напрямую зависит от размерности моделируемой SDN-сети, так как топология генерируется на клиентской машине, используя данные, полученные с сервера. Таким образом, чем мощнее аппаратная часть рабочей машины пользователя – тем быстрее и удобнее будет происходить работа с системой.

Блок взаимодействия с виртуальной сетью служит важной частью системы в целом: он связывает контроллер с пользовательским интерфейсом. При старте моделирования данный блок посылает конфигурационные файлы среде моделирования MiniNet и запускает контроллер POX исходя из заданных пользователем параметров. Также он передает пользователю текущую информацию о сети через блок взаимодействия с пользователями.

За отображение текстовой и графической информации отвечает веб-браузер. Для взаимодействия с элементами управления, топологии сети, создания динамического содержимого на стороне клиента используется HTML5 и JavaScript. Данные инструменты разработаны специально для улучшения уровня поддержки мультимедиа-технологий.

На этапе проектирования базы данных было проведено проектирование диаграмм потоков данных (рис. 2).

Основные информационные потоки формируют моделируемую пользователем виртуальную сеть: ее топологию и конфигурацию коммутаторов и контроллера.

Способ хранения и доступа к данным является одним из основных, влияющих на производительность программной системы. Задачу хранения данных можно решить несколькими способами, используя, например, файлы специального формата, под каждую сущность.

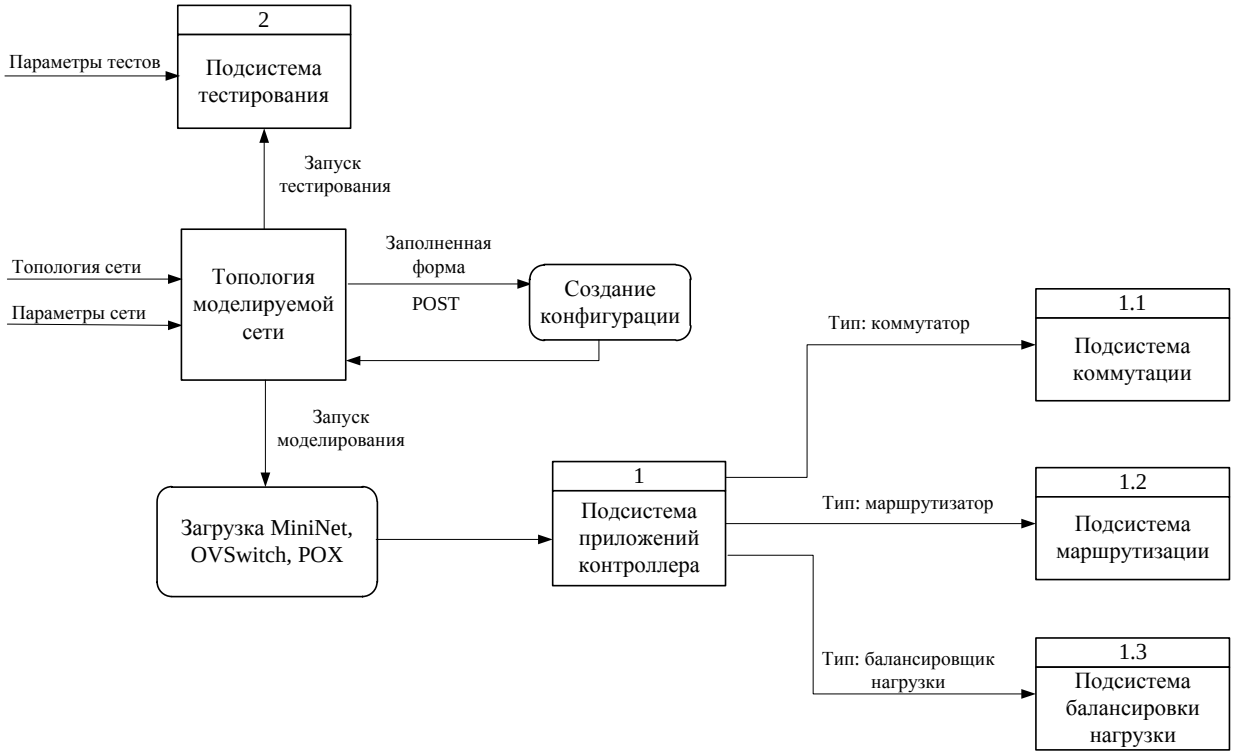

Рисунок 2 –Диаграмма потоков данных для работы с топологией сети

Однако, такой подход несколько не эффективен. Гораздо более верным выбором, согласно принципу проектирования веб – систем, является использование реляционных БД, характеризующихся высокой производительностью и широкими возможностями репликации. База данных содержит учётные записи пользователей, хранит настройки системы, используемые при выполнении моделирования, таблицы потоков коммутаторов, а так же сами топологии, включая множество их конфигураций (рисунок 3).

За моделирование сети отвечает таблица «Топология», содержащая все необходимые для её описания поля. Таблица «Коммутатор» применяется для хранения данных по всем существующим в системе типам приложений (коммутатор, маршрутизатор, балансировщик нагрузки).

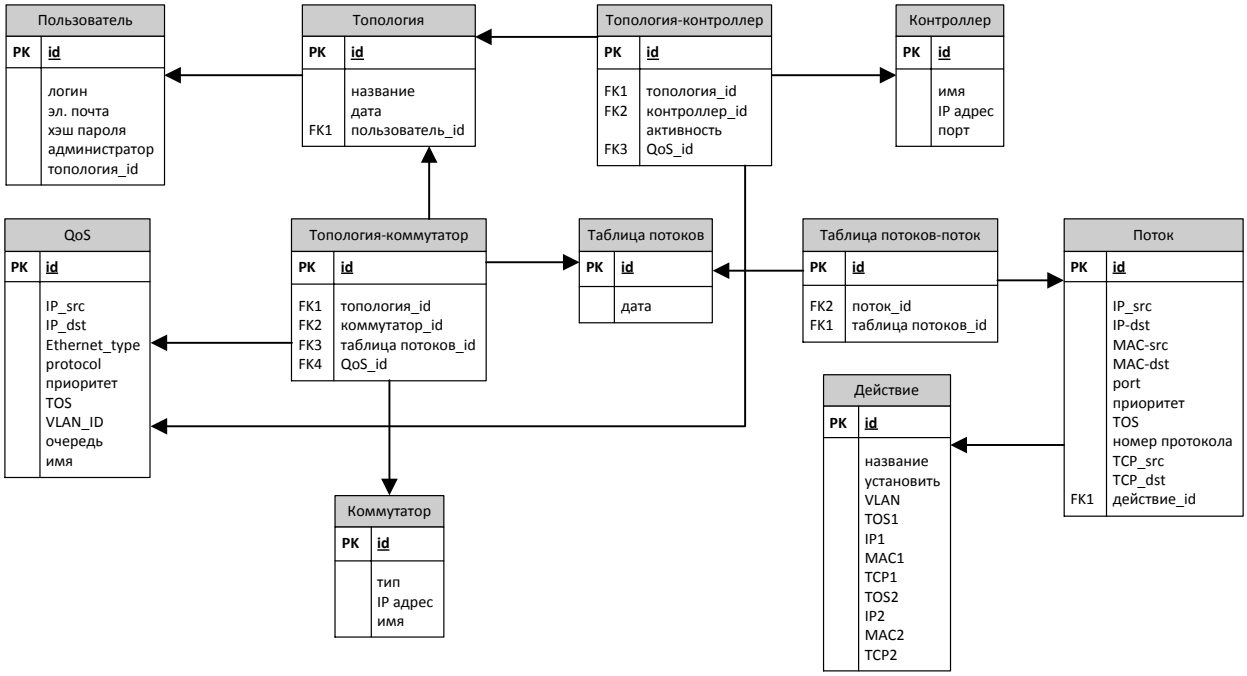

Рисунок 3 – Структурная схема базы данных

Диаграмма размещения программных компонентов, представленная на рисунке 4, показывает как выглядит программное обеспечение на физическом уровне: - из каких частей оно состоит и как эти части связаны между собой.

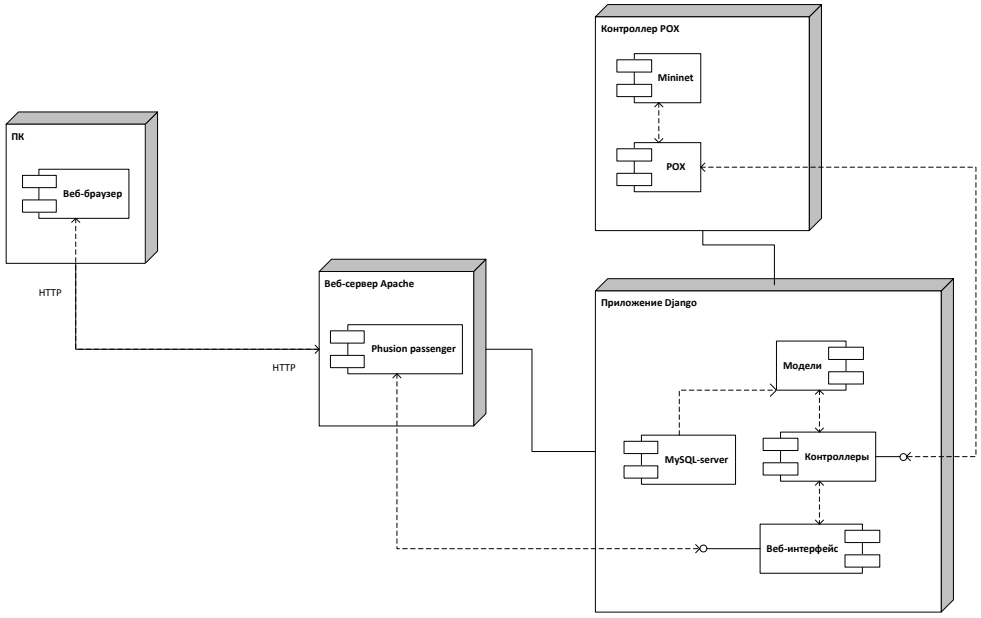

Рисунок 4 –Диаграмма размещения программных компонентов

Веб – сервер, контроллер POX и приложение располагаются на одном сервере, но для наглядности они разделены. В качестве сервера может выступать nginx или, например,

Apache. Phusion passenger – это модуль для вышеупомянутых серверов, позволяющий им работать с приложением Django. База данных системы показана на диаграмме в том же узле, что и само приложение, однако, на деле, она может быть вынесена на отдельный сервер, специально для неё предназначенный. В этом случае, скорость доступа к данным значительно повышается. Приложение Django, веб – серверы и контроллер POX могут быть развёрнуты на Unix – подобных системах, например, на Ubuntu Server. Данная операционная система предназначена для использования по большому счёту на серверах, но может использоваться и на рабочих станциях, и является бесплатной с открытым кодом. Расположение сервера SQLite и контроллера POX зависит непосредственно от того, где система будет работать.

Использование автоматизированной системы

Система моделирования позволяет создавать общие сетевые сценарии экспериментов для дальнейшего моделирования на различных инструментах OpenFlow, таких, например, как MiniNet. Для этой цели система экспортирует скрипты Python для описания правил и таблиц потока для выполнения их на контроллерах OpenFlow.

Первым шагом в моделировании SDN-сетей на основе протокола OpenFlow является создание новой топологии, объединяющей в себе сколь угодно много коммутаторов с различными параметрами (рисунок 5).

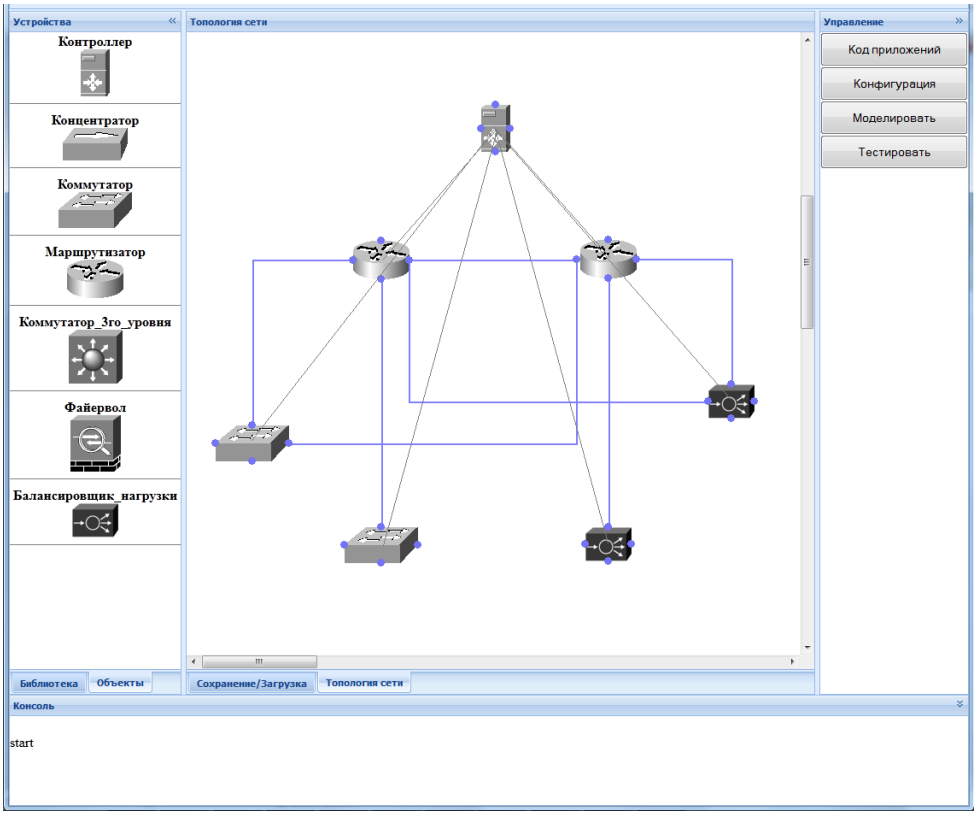

Рисунок 5 –Экран построения топологии SDN-сети

Для редактирования таблицы потоков в каждом коммутаторе предусмотрен вывод информации в виде диалогового окна, появляющегося при выборе соответствующей иконки оборудования (рисунок 6).

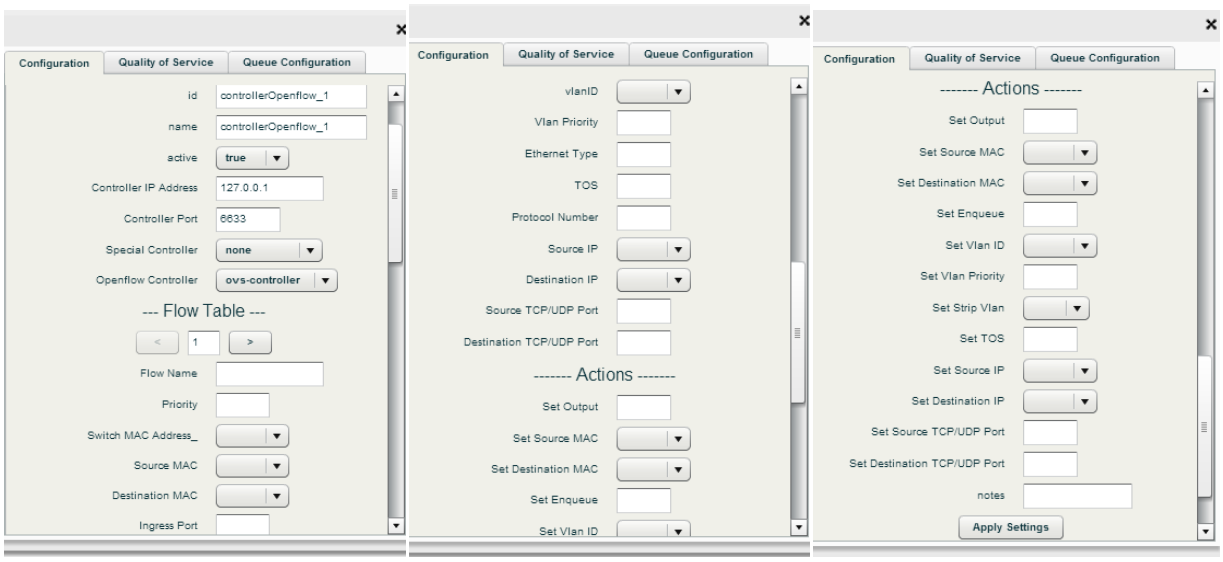

Рисунок 6 – Диалоговое окно таблицы потоков

Эта функция позволяет разработчику быстро просматривать и изменять «на лету» таблицы потоков коммутатора. При этом будет использована интеллектуальная система предотвращения дублирования записей в таблице потоков. Она создана для обеспечения целостности работы сети и выявления логических ошибок в действиях пользователя. Верификация написанных приложений для контроллера происходит с помощью утилиты ping и ПО Wireshark, запущенного на каждом узле, а также скрипта для тестирования виртуальных SDN-сетей, который позволяет генерировать произвольные сообщения OpenFlow и передавать их через всю моделируемую сеть. В результате этого строятся графики по производительности и задержкам, как представлено на рисунке 7.

С помощью Python_скрипта производится отправка сообщений между выбранными точками сети. При этом поток случайным образом меняется через 100 пакетов. Из данного графика можно сделать вывод о том, что в момент изменения заголовка пакета происходит единовременная задержка. В остальное время передача данных происходит в квазистационарном виде, что является отражением протекающих в SDN-сетях процессов.

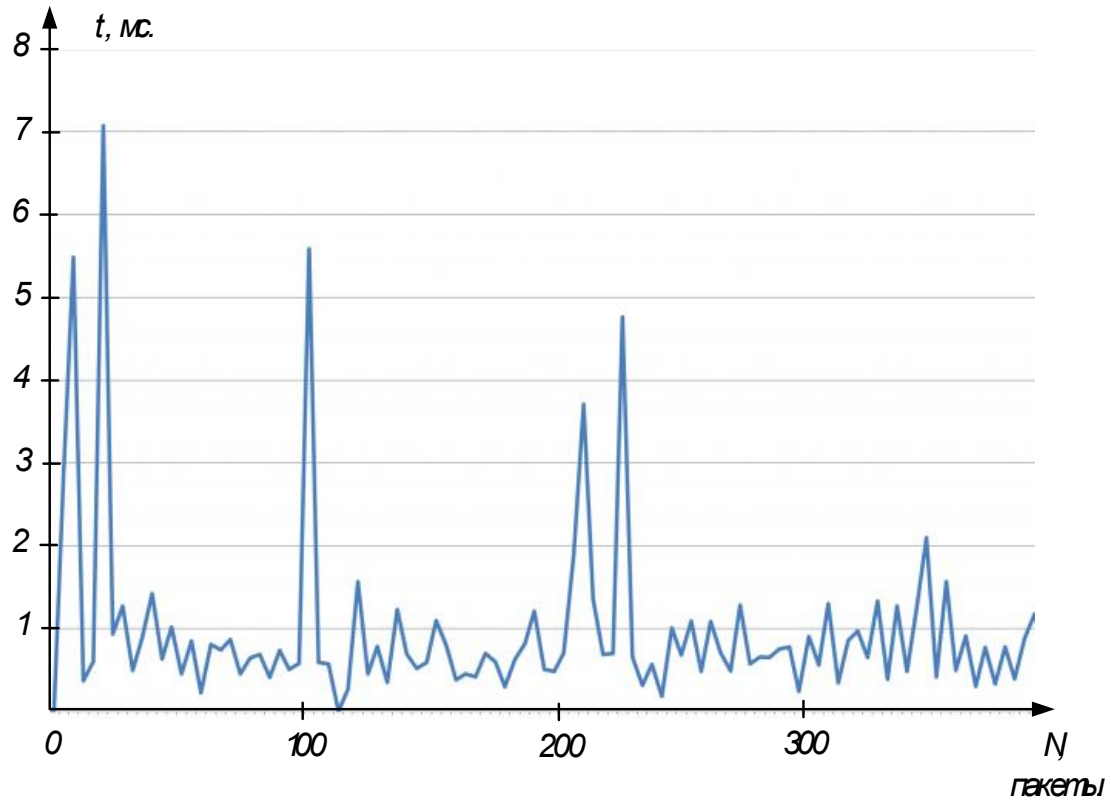

Рисунок 7 – График задержек пакетов в виртуальной SDN-сети

Заключение

Задача быстрого способа построения виртуальных SDN-сетей, а также их удобного моделирования, в настоящее время является одной из значимых в данной области. Отметим потенциальные достоинства разработанной автоматизированной программной системы моделирования относительно существующих подходов.

- *Уменьшение времени развертывания сетей.* Проектирование и построение виртуальных сетей с использованием удобного пользовательского интерфейса.
- *Возможность использования практически любых ОС*. Автоматизированная система размещается на веб-сервере Apache, что предоставляет пользователю универсальный подход при работе с системой (пользователю необходимо только наличие веб-браузера).
- *Контроль над использованием контроллера*. Система моделирования позволяет получить информативную статистику о нагрузке контроллера.
- *Открытость протокола OpenFlow*, позволяющая не зависеть от производителей сетевых устройств.
- *Удобство администрирования и отладки.*

Список литературы

1. Коломеец А.Е, Сурков Л.В. Программно-конфигурируемые сети на базе протокола OpenFlow // Электронный научно-технический журнал «Инженерный вестник» МГТУ им. Н.Э. Баумана. 2014. №5. 9 с. / URL:http://engbul.bmstu.ru/doc/711486.html
2. Коломеец А.Е, Сурков Л.В. Моделирование сетей SDN в среде Nicira // Электронный научно-технический журнал «Инженерный вестник» МГТУ им. Н.Э. Баумана».. 2014. №6. 6 с. / URL: http://engbul.bmstu.ru/doc/714126.html
3. Смелянский Р.Л. Программно-конфигурируемые сети // Открытые системы. 2012. №9. URL: http://www.osp.ru/os/2012/09/13032491/ (дата обращения: 01.11.2014).
4. Thomas D. Nadeau, Ken Gray, SDN: Software Defined Networks, O'Reilly, 2013. pp 10-25.
5. OpenFlow Tutorial // OpenFlow.2013.URL: http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial (Дата обращения: 07.11.2014).
6. ONF Specification // Open network foundation.2014. / URL: https://www.opennetworking.org/ja/sdn-resources-ja/onf-specifications (дата обращения: 07.11.2014).
7. Koponen T., Casado M., Gude N., Stribling J., Poutievski L., Zhu M., Ramanathan R., Iwata Y., Inoue H., Hama T., Shenker S. Onix: A distributed control platform for large-scale production networks / T. Koponen [et al.], Proc. of the 9th USENIX Conf. on OSDI'10. Pp. 351–364, Berkeley, Oct. 2010. USENIX Association.