Моделирование и анализ информационных систем. Т. 22, №6 (2015), с. 852–861
Modeling and Analysis of Information Systems. Vol. 22, No 6 (2015), pp. 852–861

DOI: 10.18255/1818-1015-2015-6-852-861

УДК 004.415.25

Разработка активного внешнего модуля сетевой топологии для контроллера программно-конфигурируемой сети Floodlight

Носков А. А., Никитинский М. А., Алексеев И. В.

получена 15 декабря 2015

Традиционная архитектура сети передачи данных является негибкой и сложной. Данное обстоятельство привело к появлению парадигмы программно-конфигурируемой сети (ПКС), в которой уровень управления сетью отделен от уровня передачи данных. Это стало возможно за счет переноса плоскости управления с коммутационного оборудования в программные модули, которые работают на выделенном сервере, называемом контроллером (или сетевой операционной системой), или в сетевые приложения, которые работают с этим контроллером. Способы представления, хранения и интерфейсы взаимодействия с элементами сетевой топологии, доступные пользователям контроллера ПКС, являются одними из наиболее важных аспектов сетевых операционных систем. Данное обстоятельство обусловлено тем, что функционирование некоторых ключевых модулей контроллера в существенной степени основано на внутреннем представлении сетевой топологии. Такими модулями, к примеру, являются модуль firewall, модуль маршрутизации и т.д. В данной статье рассмотрены применяемые способы представления и хранения сетевой топологии, а также интерфейсы взаимодействия с соответствующими модулями контроллера Floodlight. Предложен и разработан альтернативный алгоритм обмена сообщениями об изменении сетевой топологии между контроллером и сетевыми приложениями, позволяющий реализовать оповещение на основе подписки на соответствующие события. Разработан API для модуля взаимодействия с прикладными программами контроллера программно-конфигурируемой сети. На основе данного алгоритма и API разработан модуль Topology Tracker, способный в активном режиме сообщать сетевым приложениям о произошедших изменениях в топологии сети и хранящий ее компактное представление для ускорения процесса взаимодействия.

Ключевые слова: программно-конфигурируемая сеть, контроллер Floodlight, внешний модуль, сервис, ПКС, сетевая топология, Topology Tracker, DEventBus, Link Discovery

Для цитирования: Носков А. А., Никитинский М. А., Алексеев И. В., "Разработка активного внешнего модуля сетевой топологии для контроллера программно-конфигурируемой сети Floodlight", *Моделирование и анализ информационных систем*, **22**:6 (2015), 852–861.

Об авторах:
Носков Андрей Александрович, orcid.org/0000-0002-2268-4912, инженер,
ООО «Энергия-Инфо», ул. Союзная, 144, г. Ярославль, 150008 Россия, e-mail: naa@a-real.ru

Никитинский Михаил Александрович, orcid.org/0000-0001-8830-8613, программист-аналитик,
ООО «Энергия-Инфо», ул. Союзная, 144, г. Ярославль, 150008 Россия, e-mail: man@a-real.ru

Алексеев Игорь Вадимович, orcid.org/0000-0001-8321-2399, канд. физ.-мат. наук, директор центра Интернет,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: aiv@yars.free.net

Благодарности:

Работа выполнена в ФГБОУ ВПО "Ярославский государственный университет им. П.Г. Демидова" при финансовой поддержке Министерства образования и науки Российской Федерации в рамках Соглашения о предоставлении субсидии (*ID RFMEFI57414X0036*).

Введение

Одной из наиболее заметных тенденций в развитии сетевых технологий в наши дни является подход разделения уровней управления сетью и передачи данных за счет переноса управляющих воздействий с коммутационного оборудования (gateway, switch, router, hub и т.д.) в программные модули, которые работают на выделенном сервере, называемом контроллером, или в сетевые приложения, которые работают с контроллером. Разделение уровней управления и передачи в коммутационном оборудовании связано с тем, что оборудование, выпускаемое различными компаниями, является закрытым с точки зрения реализации и организации. Практически каждое устройство — это определенная система, в которой есть физическая составляющая, есть конфигурационная система и есть набор приложений. Данная организация сетевых устройств усложняет их взаимодействие, а введение нового устройства или приложения в сетевую инфраструктуру порой является весьма нетривиальной задачей [1]. Для решения данной проблемы был предложен новый подход к организации взаимодействия сетевого оборудования — подход ПКС [2]. Основная идея такого подхода была сформулирована специалистами университетов Стэнфорда и Беркли в 2006 году. Произведенные ими исследования нашли поддержку не только в университетах по всему миру, но и были позитивно восприняты более чем четырьмя десятками ведущих производителей сетевого оборудования, такими как CISCO SISTEMS, International Business Machines, Hewlett-Packard, Nippon Electronics Corporation.

Коммуникационная сеть называется ПКС, если содержит как минимум один контроллер сети, с установленной сетевой операционной системой, и как минимум один аппаратный или виртуальный OpenFlow-коммутатор, при этом в состав сети могут входить другие коммутационные устройства. В большинстве случаев при организации ПКС разработчики связывают контроллер с коммутационным оборудованием с помощью протокола OpenFlow [3], перенося логическую составляющую с коммутационных устройств на контроллер. Контроллер управляет OpenFlow-коммутаторами по выделенному защищенному каналу связи. На него возлагается ответственность за решения по маршрутизации сетевого трафика, управление безопасностью, разведку топологии сети и прочее. Данный функционал реализуется при помощи внутренних модулей контроллера или сетевых приложений, устанавливаемых на него.

Что касается сетевых приложений, то разработчики отмечают здесь проблему отсутствия единого стандартизированного *API* (application programming interface [4]) между сетевыми приложениями и контроллером, а также невозможность использовать приложение одного контроллера на другом [5]. Это связано с тем, что контроллеры написаны на различных языках программирования и имеют различный базовый функционал. При этом приложения, запущенные на одном контроллере, работают консистентно — они располагают одинаковой информацией о состоянии сети. Таким образом, задача управления сетью теперь возложена на разработчиков контроллеров и приложений для них, что фактически создает новый рынок — рынок сетевых приложений для контроллеров. Например, ряду приложений требуется решать задачу прокладки оптимальных маршрутов между конечными узлами в сети. Так как сеть может быть представлена в виде графа, в котором ребра — соединение между сетевыми узлами, а вершины соответствуют этим узлам, то

для поиска оптимального пути между конечными узлами может быть применена теория графов. Для построения оптимального маршрута может быть использован алгоритм Дейкстры [6], но если веса ребер изменяются часто или необходимо учитывать дополнительные параметры сети (такие как политики *Quality of Service* [7] или требования безопасности), то использование алгоритма Дейкстры для полного перерасчета кратчайших путей может быть нецелесообразным [8]. Пример различий в реализациях сетевых операционных систем можно найти и в способах обмена информацией с сетевыми приложениями. Классический подход в организации взаимосвязи такого рода предполагает использование протокола TCP [9], в качестве альтернативного подхода было предложено использование асимметричного протокола [10, 11].

Одними из наиболее востребованных модулей контроллера, в любом существующем на сегодняшний день контроллере ПКС, являются модули построения сетевой топологии. Данные модули собирают информацию о коммутаторах, хостах и связи между ними, а также отвечают за периодическое обновление информации о топологии. Обычно разработчики контроллеров называют эти модули *discovery* и *topology*, реже они объединены в один модуль. Сетевые приложения, работающие с контроллером ПКС, могут запрашивать информацию о текущем состоянии сети, но, как правило, не имеют возможности подписываться на получение сообщений о динамическом изменении топологии. Данное обстоятельство приводит либо к некорректной работе приложений, либо к необходимости периодически обращаться сетевым приложениям к контроллеру.

1. Контроллер Floodlight

Одним из наиболее динамично развивающихся контроллеров, поддерживаемых сообществом разработчиков во всем мире, является контроллер *Floodlight* [12]. FloodLight — контроллер корпоративного уровня, написан на языке Java, имеет лицензию Apache, разрабатывается компанией Big Switch Networks и является ядром для платного контроллера Big Cloud Fabric SDN controller. На момент написания статьи текущая версия контроллера FloodLigth 1.1. Контроллер имеет модульную структуру, за счет которой облегчается процесс расширения и внесения изменений, поддерживает широкий спектр виртуальных и физических коммутаторов, способен поддерживать смешанные OpenFlow сети и сети традиционной архитектуры. При описании архитектуры контроллера используют четыре основных понятия: сервисы, внутренние и внешние модули, сетевые приложения. Сервис — это интерфейс, который экспортирует состояние и генерирует события. Потребители сервиса могут получать/устанавливать состояние и подписываться или отписываться от событий. При этом допускается множество реализаций одного и того же сервиса. Каждый модуль может использовать некоторый набор сервисов для реализации некоторой функциональности. Модуль может предоставлять соответственно ноль или более сервисов. Все модули в FloodLight имеют минимальное количество зависимостей между собой, что упрощает разработку приложений. Различие между внешними и внутренними модулями заключается в том, что без внешних модулей контроллер сможет работать, а без внутренних нет. Сетевое приложение — это программа, ко-

торая по *REST API* [13] взаимодействует с внутренними и внешними модулями, может устанавливаться как на самом контроллере, так и удаленно. Общая структура контроллера Floodlight представлена на рис. 1.

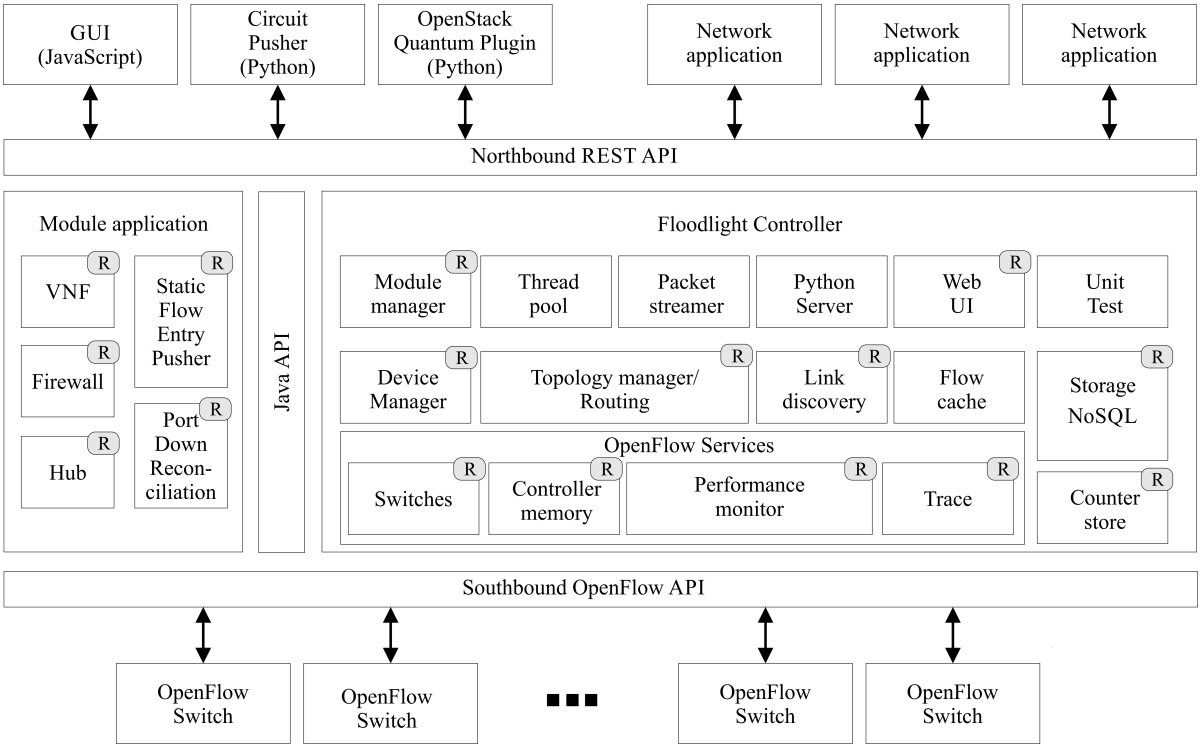

Рис. 1. Общая структура контроллера Floodlight

Контроллер Floodlight использует для разведки топологии модуль *Link Discovery*. Данный модуль работает с протоколом *LLDP (Link Layer Discovery Protocol)* [14]. LLDP является протоколом канального уровня, который позволяет сетевому оборудованию уведомлять о своем существовании другие узлы сети, а также обрабатывать подобные сообщения, приходящие от других устройств.

Информация, полученная контроллером Floodlight из LLDP Data Unit сообщений, сохраняется на устройстве в виде *management information base (MIB)* [15–17]. Таким образом, общая топология сети получается путем сбора и объединения информации со всех устройств. Собираемая информация может содержать следующие данные:

- имя устройства и его описание,
- имя порта и его описание,
- имя VLAN,
- IP-адрес управления устройством через протокол SNMP,
- функции, которые может выполнять устройство,
- информация о MAC/PHY,

- информация о питании через Ethernet (Power over Ethernet),
- информация об объединении каналов.

Используя LLDP, контроллер Floodlight, при включении получает информацию обо всех OF-коммутаторах [18], присутствующих в сети, и при этом не имеет информации о конечных узлах сети. Эта информация появляется по мере работы конечных узлов, как только они начинают пересылать какой-либо трафик через сеть, информация о них записывается в MIB.

Однако модуль Link Discovery не может отправлять динамические изменения, происходящие в сети, сетевым приложениям. Данная особенность оказывает серьезное влияние на работу сетевых приложений. Например, за графическое отображение в web-браузере топологии ПКС в контроллере Floodlight отвечает сетевое приложение GUI. Для отображения актуального состояния ПКС в приложении GUI необходимо либо дать команду актуализировать информацию на данный момент, либо установить требование постоянно обновлять информацию о состоянии сети, при этом время обращения к контроллеру фиксировано, неизменно и равно трем секундам.

2. Принципы функционирования внешнего модуля Topology Tracker

При изменении сетевой структуры ПКС в модуль Link Discovery поступают сообщения, содержащие информацию об источнике и параметрах произошедшего изменения. Внутренние модули контроллера обрабатывают их, соответствующим образом изменяя внутреннее представление сети. Затем через Java API подписанным внешним модулям контроллера передается информация о произошедших изменениях. Внешние модули могут формировать некоторую реакцию на эти события.

В статье [19] авторами был предложен механизм взаимосвязи сетевых приложений с контроллером ПКС. Данный механизм обеспечивает сетевые приложения возможностью осуществлять подписку на предоставляемые сервисы как контроллером, так и другими сетевыми приложениями.

Так как внешние модули могут быть подписаны на изменения, происходящие в сетевой инфраструктуре, а изменение ядра контроллера Floodlight может привести к критическим ошибкам, был разработан внешний модуль Topology Tracker. Основными функциями данного модуля является собственное представление сетевой топологии и передача информации сетевым приложениям, которые подписаны на произошедшие изменения в ПКС. Topology Tracker формирует представление сетевой топологии в виде трех ассоциативных массивов для хранения информации о маршрутизаторах, конечных устройствах и соединениях соответственно. В первых двух случаях ключом является идентификатор устройства (в простейшем случае – MAC-адрес устройства). Ключом для массива соединений является строка, образованная конкатенацией строковых представлений идентификаторов конечных устройств. Значениями, во всех случаях, являются специальные классы представлений, организованные таким образом, чтобы уменьшить время поиска запрашиваемо-

го устройства и его параметров. Минусом такой структуры является относительно медленное добавление и удаление элементов массивов.

Модуль Topology Tracker может взаимодействовать с внутренними модулями контроллера Floodlight через Java API для восстановления информации о состоянии сети при обнаружении некорректных данных, а также в случае остановки и возобновления работы. Когда происходит изменение в топологии сети, контроллер по очереди предоставляет данную информацию внутренним модулям. Модуль Topology Tracker получает данные после обработки их модулем Link Discovery. Обработав поступившую информацию, модуль Topology Tracker отправляет сформированные данные на внутренний сокет контроллера Floodlight. Данный сокет прослушивает локальное сетевое приложение DEventBus, которое выступает в роли сетевой шины с механизмом регистрации событий от модулей и приложений, а также с возможностью подписки на данные события. Приложение DEventBus было разработано в соответствии с логикой, описанной в [19]. Если приложению DEventBus необходимо передать данные на удаленное устройство, на котором находится подписанное на событие сетевое приложение, приложение DEventBus связывается с аналогичным приложением на стороне приема и через него передает данные. Если сетевое приложение находится локально, то от модуля DEventBus ему передаются данные. Схема обмена сообщениями об изменениях сетевой топологии ПКС представлена на рис. 2. Сплошной линией обозначен предлагаемый метод обмена сообщениями, пунктирной линией – метод обмена сообщениями, применяемый в контроллере Floodlight в настоящее время. Штрих-пунктирной линией обозначена защищенная схема обмена сообщениями при запросе, адресованном через REST API разработанному модулю. Во всех случаях стрелками обозначено направление передачи сообщений.

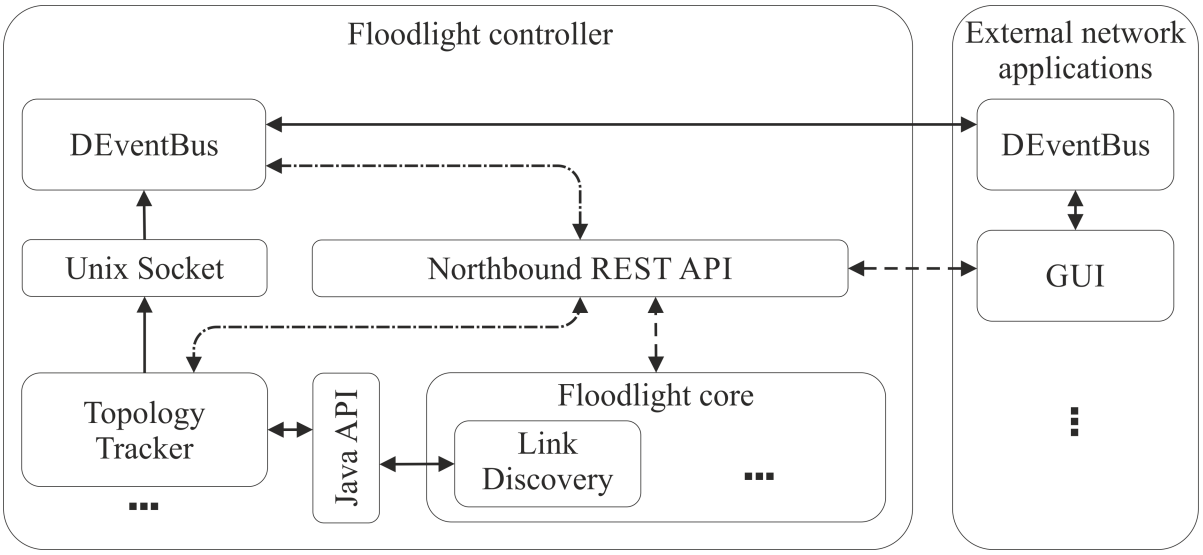

Рис. 2. Схема обмена сообщениями об изменениях сетевой топологии ПКС

3. Сравнение модулей Topology Tracker и Link Discovery

Анализ разработанного внешнего модуля Topology Tracker позволяет провести частичное сравнение по ключевым параметрам с аналогичным модулем Link Discovery, который в данный момент используются в контроллере Floodlight.

С точки зрения обеспечения безопасности контроллера. Обращение к внутреннему модулю контроллера Floodlight происходит по REST API запросу с любого устройства сети, при этом он может происходить как с указанием логина и пароля, так и без них. Взаимодействие с разработанным модулем могут осуществлять сетевые приложения, зарегистрированные в DEventBus. Ответственность за регистрацию сетевых приложений возложена на администратора сети.

С точки зрения используемых ресурсов контроллера. Разработанная система программных решений использует собственную базу данных, которая формируется в оперативной памяти по мере работы контроллера Floodlight параллельно с его собственной. Такой подход позволяет быстро формировать ответ на запрос о предоставлении полной или частичной информации о текущем состоянии сетевой топологии. Кроме того, разработанная структура имеет меньший размер, чем собственная база контроллера (расположенная *$floodlight_path$ /net/floodlightcontroller/storage*), так как содержит только информацию о хостах, сетевом оборудовании и связи между ними. Технически созданная база данных является java-коллекцией типа HashMap, что обеспечивает в лучшем случае константное ($O(1)$) время поиска, вставки и удаления элемента. В худшем случае эти операции выполняются за линейное время ($O(n)$). Так как HashMap в Java использует метод цепочек для разрешения коллизий, среднее время поиска элемента составляет $\Theta(1+\alpha)$, где α – коэффициент заполнения таблицы [20]. Стоит отметить, что собственная база данных контроллера Floodlight также располагается в оперативной памяти и имеет ряд вложенных структур типа HashMap, что в худшем случае дает сложность основных операций $\Theta(n^k)$, где k – уровень вложенности. Таким образом, при небольшой или простой топологии времена выполнения запросов будут сопоставимы, а при сложной или большой топологии очевидно преимущество разработанного программного решения.

С точки зрения загруженности каналов связи при условии, что контроллер и сетевые приложения находятся на различных физических устройствах. В данном случае при использовании разработанного механизма загрузка входящего канала на контроллер от сетевых приложений становится равна нулю.

Заключение

Качественным отличием разработанной системы является возможность своевременной доставки информации о произошедших в сетевой топологии изменениях от контроллера Floodlight к сетевым приложениям. Кроме того, высвобождаются сетевые и процессорные ресурсы, которые, использовались для обработки входящих REST-

запросов. В качестве недостатка следует отметить узкую специализацию разработанной базы данных.

Разработанные алгоритмы и созданная на их основе система обмена сообщениями, является универсальным средством взаимодействия контроллера Floodlight и сетевых приложений. Она может быть использована для разработки иных сетевых приложений, для которых критичным фактором является своевременность поступления информации от контроллера Floodlight.

По описанным в статье алгоритмам были разработаны сетевые приложения *Topology Tracker* и *DEventBus* для контроллера Floodlight, а также были получены свидетельства о государственной регистрации программ для ЭВМ Российской Федерации *№2015618629* и *№2015660314* соответственно.

Список литературы / References

[1] V. Sokolov et al., "A network analytics system in the SDN", SDN&NFV: The Next Generation of Computational Infrastructure: 2014 International Science and Technology Conference "Modern Networking Technologies (MoNeTec)" (Moscow, October 27–29, 2014), 160–162.

[2] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, **38**:2 (2008), 69–74.

[3] *Open Networking Foundation*, https://www.opennetworking.org/.

[4] David Orenstein, "Application Programming Interface", *Computerworld*, 2010, http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html.

[5] Sally Johnson, "Do SDN northbound APIs need standards?", SearchSDN (January, 2013), http://searchnetworking.techtarget.com/feature/Do-SDN-northbound-APIs-need-standards.

[6] Dijkstra E. W., "A note on two problems in connexion with graphs", *Numer. Math*, **1**:1 (1959), 269–271.

[7] Ryan Wallner and Robert Cannistra, "An SDN Approach: Quality of Service using Big Switch's Floodlight Open-source Controller", Proceedings of the Asia-Pacific Advanced Network, **35** (2013), 14–19.

[8] C. Demetrescu, G.F. Italiano, "A new approach to dynamic all pairs shortest paths", **51**:6 (2004), 968–992.

[9] *Transmission Control Protocol. DARPA Internet Program. Protocol Specification*, RFC793,September, 1981, www.rfc-editor.org.

[10] M. Nikitinskiy, I. Alekseev., "A stateless transport protocol in software defined networks", SDN&NFV: The Next Generation of Computational Infrastructure: 2014 International Science and Technology Conference "Modern Networking Technologies (MoNeTec)" (Moscow, October 27–29, 2014), 108–113.

[11] M.A. Nikitinskiy and I.V. Alekseev, "Analyzing the Possibility of Applying Asymmetric Transport Protocols in Terms of Software Defined Networks", *Automatic Control and Computer Sciences*, **49**:2 (2015), 94–102.

[12] *Floodlight SDN OpenFlow Controller*, https://github.com/floodlight/floodlight.

[13] Pautasso C., Wilde E., Alarcon R., *REST: Advanced Research Topics and Practical Applications*, Springer-Verlag New York, 2014, ISBN: 978-1-4614-9298-6.

[14] *IEEE 802.1AB (LLDP) Specification*, http://standards.ieee.org/getieee802/download/802.1AB-2005.pdf.

[15] M. Rose and K. McCloghrie, *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC1155, May, 1990, www.rfc-editor.org.

[16] M. Rose and K. McCloghrie, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, RFC1213, March, 1991, www.rfc-editor.org.

[17] J. Case et al., *A Simple Network Management Protocol (SNMP)*, RFC1157, May, 1990, www.rfc-editor.org.

[18] *OpenFlow Switch Specification, Version 1.3.4*, March, 2014, OF switch v.1.3.4.

[19] Alekseev I. and Nikitinskiy M., "EvenetBus Module for Distributed OpenFlow Controllers", Proceedings of the 17th Conference of Open Innovations Association FRUCT (Yaroslavl, Russia, 20-24 April 2015), 3–8.

[20] Thomas H. Cormen et al., *Introduction to Algorithms*, 3rd., MIT Press, 2009, ISBN: 0-262-03384-4.

DOI: 10.18255/1818-1015-2015-6-852-861

Development of Active External Network Topology Module for Floodlight SDN Controller

Noskov A. A., Nikitinskiy M. A., Alekseev I. V.

Received December 15, 2015

Traditional network architecture is inflexible and complicated. This observation has led to a paradigm shift towards software-defined networking (SDN), where network management level is separated from data forwarding level. This change was made possible by control plane transfer from the switching equipment to software modules that run on a dedicated server, called the controller (or network operating system), or network applications, that work with this controller. Methods of representation, storage and communication interfaces with network topology elements are the most important aspects of network operating systems available to SDN user because performance of some key controller modules is heavily dependent on internal representation of the network topology. Notably, firewall and routing modules are examples of such modules. This article describes the methods used for presentation and storage of network topologies, as well as interface to the corresponding Floodlight modules. An alternative algorithm has been suggested and developed for message exchange conveying network topology alterations between the controller and network applications. Proposed algorithm makes implementation of module alerting based on subscription to the relevant events. API for interaction between controller and network applications has been developed. This algorithm and API formed the base for Topology Tracker module capable to inform network applications about the changes that had occurred in the network topology and also stores compact representation of the network to speed up the interaction process.

Keywords: software-defined network, Floodlight controller, external module, service, SDN, network topology, Topology Tracker, DEventBus, Link Discovery

For citation: Noskov A. A., Nikitinskiy M. A., Alekseev I. V., "Development of Active External Network Topology Module for Floodlight SDN Controller", *Modeling and Analysis of Information Systems*, **22**:6 (2015), 852–861.

On the authors:
Noskov Andrey Aleksandrovich, orcid.org/0000-0002-2268-4912, engineer,
A-Real Group, Energiya-Info Inc., Souznaya str., 144, Yaroslavl, 150008, Russia, e-mail: naa@a-real.ru

Nikitinskiy Mikhail Aleksandrovich, orcid.org/0000-0001-8830-8613, system analyst, programmer,
A-Real Group, Energiya-Info Inc., Souznaya str., 144, Yaroslavl, 150008, Russia, e-mail: man@a-real.ru

Alekseev Igor Vadimovich, orcid.org/0000-0001-8321-2399, Director of the Internet Center, Ph.D.,
P.G. Demidov Yaroslavl State University, Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: aiv@yars.free.net

Acknowledgments:

This work was performed in the Yaroslavl State University with support of the Ministry of Education and Science of the Russian Federation in the framework of the Subsidy Agreement (*ID RFMEFI57414X0036*).