ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Июнь 2016 Том 4 № 2 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
June 2016 Vol. 4 Iss. 2 ISSN 2307-1303 http://www.itt.sut.ru

МЕТОД УПРАВЛЕНИЯ ТРАФИКОМ В ГИБРИДНОЙ ПРОГРАММНО-ОПРЕДЕЛЯЕМОЙ СЕТИ

А. В. Красов[1], М. В. Левин[1], А. Ю. Цветков[1]

[1]СПбГУТ, Санкт-Петербург, 193232, Российская Федерация
Адрес для переписки: krasov@inbox.ru

Информация о статье
УДК 004.72
Язык статьи – русский.
Поступила в редакцию 11.02.16, принята к печати 11.03.16.

Ссылка для цитирования: Красов А. В., Левин М. В., Цветков А. Ю. Метод управления трафиком в гибридной программно-определяемой сети // Информационные технологии и телекоммуникации. 2016. Том 4. № 2. С. 53–63.

Аннотация
Предмет исследования. Стать посвящена методу управления трафиком в высоконагруженной гибридной программно-определяемой сети. **Метод.** В качестве метода исследования применяется эксперимент. **Основные результаты.** В статье приведены универсальные программные модели реализации предложенного метода управления трафиком. **Практическая значимость.** Приведена формализованная модель предложенного метода управления трафиком, а также его универсальная программная модель реализации.
Ключевые слова
доступность, безопасность, программно-определяемая сеть, контроллер, EIGRP, нагрузка.

THE METHOD OF THE TRAFFIC CONTROL IN HYBRID SOFTWARE-DEFINED NETWORK

A. Krasov[1], M. Levin[1], A. Tcvetkov[1]

[1]SPbSUT, St. Petersburg, 193232, Russian Federation
Corresponding author: krasov@inbox.ru

Article info
Article in Russian.
Received 11.02.16, accepted 11.03.16.

For citation: Krasov A., Levin M., Tcvetkov A.: The Method of the Traffic Control in Hybrid Software-defined Network // Telecom IT. 2016. Vol. 4. Iss. 2. pp. 53–63 (in Russian).

Abstract
Research subject. The article is devoted to the method of managing traffic highly loaded hybrid software-defined network. **Method.** As a method of study used experiment. **Core results.** The article presents the universal software model of the proposed traffic control method. **Practical relevance.** Shows formalized model of the proposed traffic control method, as well as its versatile programming model implementation.
Keywords
availability, security, software-defined network, controller, EIGRP, load.

Введение

В настоящее время, концепция SDN (*Software Defined Network* или программно-определяемая сеть) стремительно завоевывает мир сетевых технологий. Быстрый рост объема трафика приводит к росту инфраструктуры, которая позволит его обработать. Это в свою очередь приводит к тому, что управление любыми сетями становится громоздким, малоэффективным и сложным.

В работах [1, 2, 3, 4, 5] предложены решения по обеспечению механизма маршрутизации в SDN с учетом выполнения требований к качеству обслуживания (QoS) для максимально возможного числа потоков и в условиях изменяющейся нагрузки.

На основе выводов, полученных в [4] и [5], рассмотрим гибридную модель SDN на основе протокола маршрутизации EIGRP.

Данный протокол относится к типу distance-vector, однако, в отличие от RIP и IGRP, использует в качестве основы своей работы алгоритм диффузионных вычислений. Подробности работы данного алгоритма приведены в [6] и [7].

Маршрутизаторы, участвующие в работе EIGRP обмениваются информацией о префиксах, находящихся в таблице маршрутизации каждого из них. Информация о префиксах содержит:

- IP-адрес сети;
- маску подсети;
- полосу пропускания сегмента данной сети;
- задержку на интерфейсе маршрутизатора, который представляет данный сегмент сети;
- нагрузку, которая действует на интерфейсе, представляющем данный сегмент сети;
- надежность, рассчитанная на интерфейсе данного сегмента сети;
- размер MTU.

В качестве полосы пропускания сегмента сети передается минимальная полоса пропускания по сетевому пути, через который проходит маршрут до заданной сети. В качестве задержки передается суммарное значение задержки для передачи пакета по всем каналам на маршруте до заданной сети. В качестве нагрузки передается максимальное значение параметра txload по всем каналам на маршруте до заданной сети, который автоматически рассчитывается на каждом интерфейсе маршрутизатора. При этом важно учитывать, что передается значение нагрузки, генерируемой трафиком, передающимся в сторону префикса-назначения. В качестве надежности передается минимальное значе-

ние отношения количества верно принятых пакетов к общему количеству принятых пакетов по всем каналам на маршруте до заданной сети. Значения нагрузки и надежности вычисляются каждым маршрутизатором в отдельности на интерфейсе, подключенным к каналу, который является частью маршрута до заданной сети. Процесс передачи информации о префиксе представлен на рис. 1.

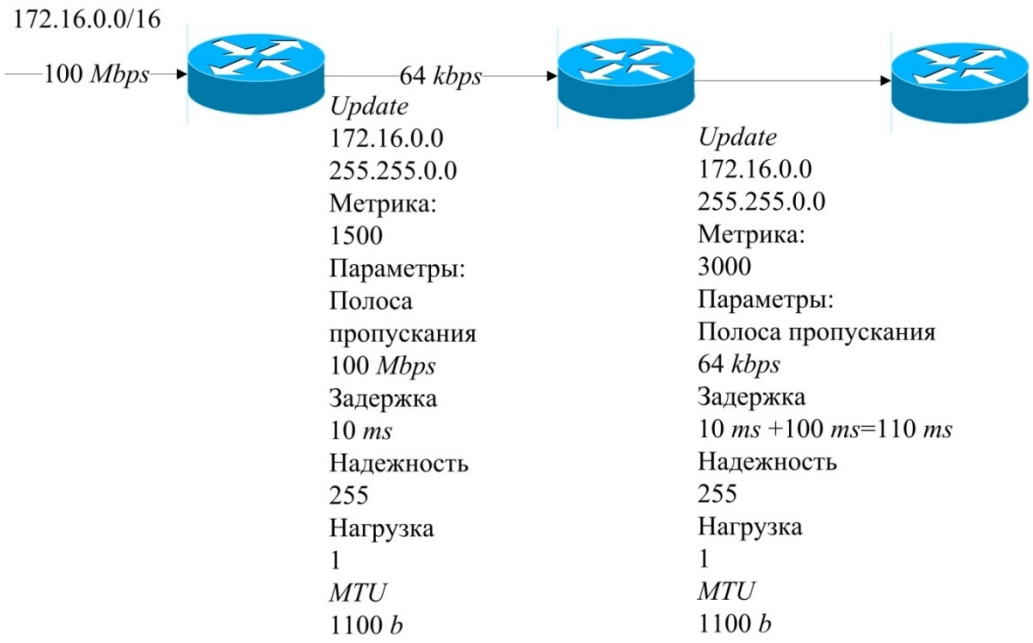

Рис. 1. Передача маршрутной информации протоколом EIGRP

На основе полученной информации о префиксе маршрутизатор вычисляет метрику [8, 9, 10]. В стандарте, описывающим работу EIGRP, определено два варианта метрики. Стандартная и расширенная. Стандартная метрика рассчитывается маршрутизатором по формуле (1).

$$CM = \left(K_1 \cdot BW_S + K_2 \cdot \frac{BW_S}{256 - Lo_{\max}} + K_3 \cdot D_S \right) \cdot \left(\frac{K_5}{K_4 + R_{\min}} \right), \quad (1)$$

где $BW_S = \frac{256 \cdot 10^7}{Bandwidth_{\min}}$, $Lo_{\max} - Load$, $D_S = 256 \cdot Delay_{summed}$, $R - Reliability$.

Как видно из формулы (1), при расчете метрики для конкретного префикса используется параметр *Load* (нагрузка), максимальное его значение по всем каналам на маршруте до заданного префикса.

Коэффициенты K_1, K_2, K_3, K_4, K_5 представляют собой весовые коэффициенты, изменяющиеся от 0 до 255, они служат для того, чтобы при расчете метрики тот или иной параметр оказывал большее или меньшее влияние на конечный результат расчета. Верхний порог – 255 обусловлен необходимостью масштабирования метрики до размера в 32 бита – максимального размера значения метрики в таблице маршрутизации. Во всех текущих реализациях *EIGRP* значения коэффициентов: K_1, K_2, K_3, K_4, K_5, таким образом реальную роль при расчете метрики играют только задержка и пропускная способность.

Кроме того, возможно использовать также расширенную метрику, которая рассчитывается по формуле (2).

$$WM = \left(K_1 \cdot T_{\min} + K_2 \cdot \frac{T_{\min}}{256 - Lo_{\max}} + K_3 \cdot La_{summed} + K_6 \cdot ExtM \right) \cdot \left(\frac{K_5}{K_4 + R_{\min}} \right), \quad (2)$$

где $T_{\min} = \frac{65536 \cdot 10^7}{Bandwidth_{\min}}$, $La_{summed} = \sum_{1}^{Hop\ Count} \frac{65536 \cdot Delay_{interface}}{10^6}$.

Использование расширенной метрики обусловлено использование в настоящее время каналов с пропускной способность 1 Гбит/с и выше [8, 9, 10]. Из формулы (1) видно, что при использовании таких каналов различий в метрики не будет, поэтому при расчете расширенной метрики используются большие значения числителей.

При включении в расчет параметра Load, т. е. при установлении значения коэффициента $K_2 > 1$ этот параметр учитывается только при первоначальном расчете метрики для маршрута. После первого расчета и получения значения метрики даже при изменении параметра Load эти изменения не учитываются и перерасчет метрики не производится. Данная особенность связана со следующими сложностями. Как известно из положений теории массового обслуживания, нагрузка (трафик), генерируемая подключенными к сети устройствами, имеет вероятностную природу, т. е. значение нагрузки, присутствующей в сети не постоянно и изменяется во времени случайным образом. Распределение вероятностей действия нагрузки трафика при этом различается, в зависимости от типа сервисов, предоставляемых сетью и типа устройств, подключенных к сети. В результате, изменение параметра Load, который отражает изменение нагрузки, действующей на интерфейсе маршрутизатора, имеет нелинейную природу; изменение также происходит в соответствии с действующим в данной сети законом распределения вероятностей, из-за чего значение параметра может принимать сильно различающиеся по абсолютной величине значения на коротком интервале времени. В результате, частый перерасчет метрики с сильно различающимися значениями параметра Load может приводить к частому изменению маршрута для прохождения трафика до заданного префикса, что:

- увеличивает задержку при передаче трафика;
- приводит к потере некоторых пакетов в моменты нестабильности сети;
- вызывает изменения в последовательности передаваемых пакетов.

Иллюстрация данной ситуации приведена на рис. 2.

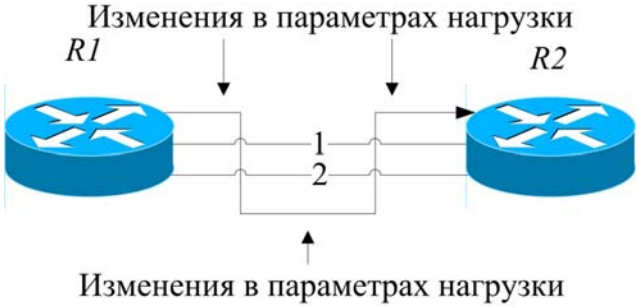

Рис. 2. Проблема с нестабильностью маршрутов в EIGRP

Кроме того, изменение метрики маршрута вызывает работу механизмов диффузионных вычислений, заложенные в EIGRP, из-за чего, информация о частых изменениях метрики, и, соответственно, маршрута, передается к другим маршрутизаторам, участвующим в работе EIGRP, которые также пересчитывают метрику и изменяют решения о маршрутизации, что, в итоге, приводит к постоянному частому изменению всех маршрутов и нестабильности сети [9, 10], в результате которых негативные факторы, перечисленные выше, усиливаются и увеличивается количество отказов в обслуживании, что сказывается на доступности информации, передающейся по сети.

По всем указанным причинам, как было отмечено выше, после первого расчета и получения значения метрики даже при изменении параметра Load эти изменения не учитываются и перерасчет метрики не производится.

Алгоритм пересчета нагрузки

Текущие реализации протокола не используют параметры *Load* для расчета метрики. Для того, чтобы обойти эти ограничения, можно использовать адаптивный алгоритм реагирования на изменения нагрузки, который задается разностным уравнением (3):

$$Load = \alpha \cdot Load + (1-\alpha) \cdot Load_{new} \qquad 0 \le \alpha \le 1 \qquad (3)$$

Согласно особенностям разностного уравнения, значение параметра Load, вычисляемое за текущий интервал на контроллере, будет зависеть от значений параметра Load, вычисленного на предыдущем интервале и значений параметра Load, полученного за текущий интервал от маршрутизатора. При этом вес последнего в общем результате вычислений текущего такта будет зависеть от значения коэффициента α [11]. С увеличением данного коэффициента уменьшается чувствительность данного алгоритма к изменениям в нагрузке, с уменьшением данного коэффициента увеличивается чувствительность алгоритма к изменениям в нагрузке. Вопрос о том, какое значение коэффициента выбрать в случае конкретных топологии и модели трафика (закона распределения вероятностей нагрузки) является открытым и требует дальнейшего исследования.

На уровне контроллера задается пороговое значение, которое обеспечивает условие реакции на изменения в нагрузке, совместно с заданием коэффициента α определяется общая реакция алгоритма на изменения нагрузки в сети.

Поскольку при вычислении метрики для маршрута маршрутизатором используются целые значения параметров (пропускной способности, задержки, нагрузки, надежности), то и вычисления на уровне контроллера имеет смысл производить с целыми числами, тогда результат вычислений следует округлять до ближайшего целого значения, чтобы впоследствии передавать маршрутизатору.

Вычисления производится для каждого интерфейса маршрутизатора, параметры которого может получить контроллер.

Кроме вопроса о выборе коэффициента α также встает вопрос о применении предложенного механизма в случае конкретной сетевой топологии. Он также требует отдельного исследования, но можно сказать о крайних случаях та-

кого вопроса. Не имеет смысла использовать предложенной механизм в сети с топологией, изображенной на рис. 3.

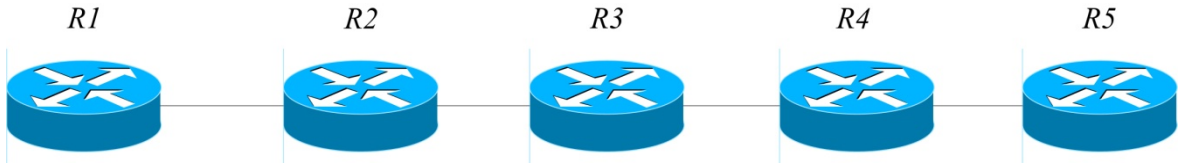

Рис. 3. Сетевая топология

Поскольку такая топология представляет собой последовательность маршрутизаторов, соединенных между собой последовательно, существует единственный маршрут до любой из сетей-назначения. Вследствие этого перерасчет метрики и вычисления алгоритма не имеют смысла и занимают процессорное время контроллера и маршрутизаторов сети.

После того, как вследствие изменения значения нагрузки было достигнуто определенное (заданный администратором) пороговое значение этого параметра, контроллер передает на маршрутизаторы (один или несколько) решение о перерасчете маршрута. Поскольку маршрутизатору, участвующему в работе EIGRP, известны текущие значения параметра нагрузки, то передача этих параметров не требуется. После получения решения о перерасчете маршрутов маршрутизатор запускает вычисления метрики для всех маршрутов, или для тех маршрутов, которые затрагиваются изменившимся значением метрики согласно текущим спецификациям EIGRP безо всяких изменений. Такая конечная реализация предложенного метода задействует уже имеющиеся на сетевых устройствах механизмы и алгоритмы и не требует изменений ни в аппаратной, ни в программной реализации сетевых устройств. Предложенный алгоритм вместе со всеми вычислениями развертывается на контроллере, в роли которого может выступать любая платформа, аппаратная или программная с поддержкой соответствующих программных интерфейсов. Архитектура предлагаемого решения приведена на рис. 4.

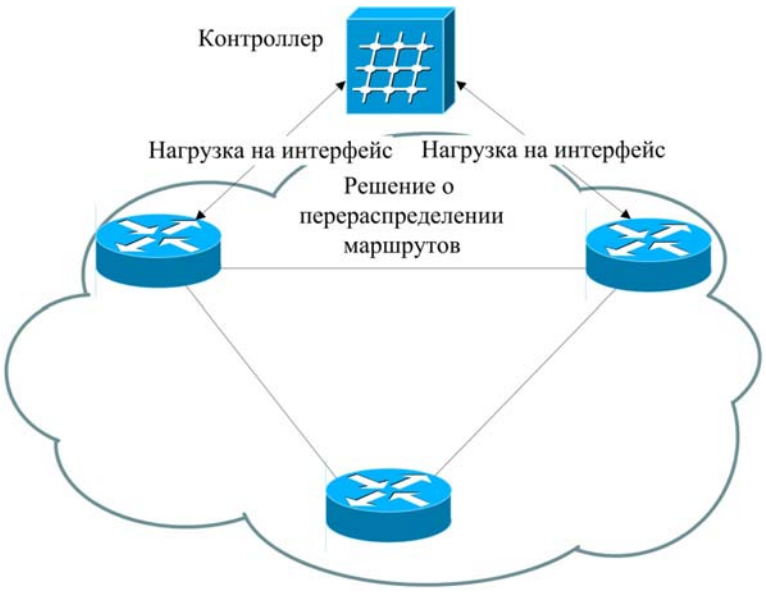

Рис. 4. Предлагаемой решение по учету нагрузки в сети ПД

Важным является вопрос о тех маршрутизаторах, которым контроллер должен передать решение о перемаршрутизации. Если высокая нагрузка действует на всех каналах в сети, то перерасчет метрики не позволит перенаправить часть трафика по менее загруженному каналу, поскольку последних в сети нет.

Механизм сбора информации с узла сети

Рассмотренный алгоритм предлагается использовать в рамках гибридной реализации программно-определяемой сети.

При реализации предлагаемого метода на практике использовалось:

- Оборудование Cisco Systems Inc.
- Программные компоненты Cisco Systems Inc., которые состоят из операционной системы для сетевого оборудования с установленными API, а также программных библиотек, совместимых с указанными API (Cisco IOS 15.4.2, ONE Platform Kit (onePK)).

Cisco® Open Network Environment (ONE) представляет собой комплексное решение, которое позволяет внедрить в существующую сетевую инфраструктуру элементы программно-определяемой сети, в частности, элементы гибридной модели программно-определяемой сети. По сути, данное решение включает в себя набор взаимосвязанных технологий и механизмов:

- API для различных платформ Cisco Systems Inc;
- Контроллер;
- приложения агенты для взаимодействия с контроллером.

Возможности onePK представляют собой реализацию гибридной модели SDN. Однако, с помощью этих возможностей также может быть реализована классическая модель SDN, в частности, с помощью onePK на сетевых устройствах могут быть внедрены OpenFlow-агенты, каждый из которых является часть архитектуры классической модели SDN.

Пакет разработки onePK совместим со всеми основными платформами Cisco. Гибкая среда разработки, включенная в состав onePK, предоставляет программный уровень представления сетевых элементов с помощью API, поддерживаемых объектно-ориентированными языками программирования [12, 13].

Центральным элементом архитектуры onePK является единый набор API библиотек для всех основных платформ Cisco.

По сути, этот уровень инфраструктуры (платформы с установленными библиотеками API) предоставляет уровень абстракции для специфических платформенных решений. Такая реализация позволяет разработчикам приложений не концентрироваться на специфических особенностях отдельных платформ или всей инфраструктуры, что значительно упрощает разработку и увеличивает масштабируемость приложений. Разработчики могут использовать одни и те же библиотеки API во всей инфраструктуре, даже если отдельные сетевые устройства работают под управление разных операционных систем.

Между уровнем представлений и уровнем сетевой инфраструктуры строится канал для обеспечения их взаимодействия, как правило, в рамках клиент-серверной модели взаимодействия. Вся описанная выше архитектура *onePK* представлена на рис. 5.

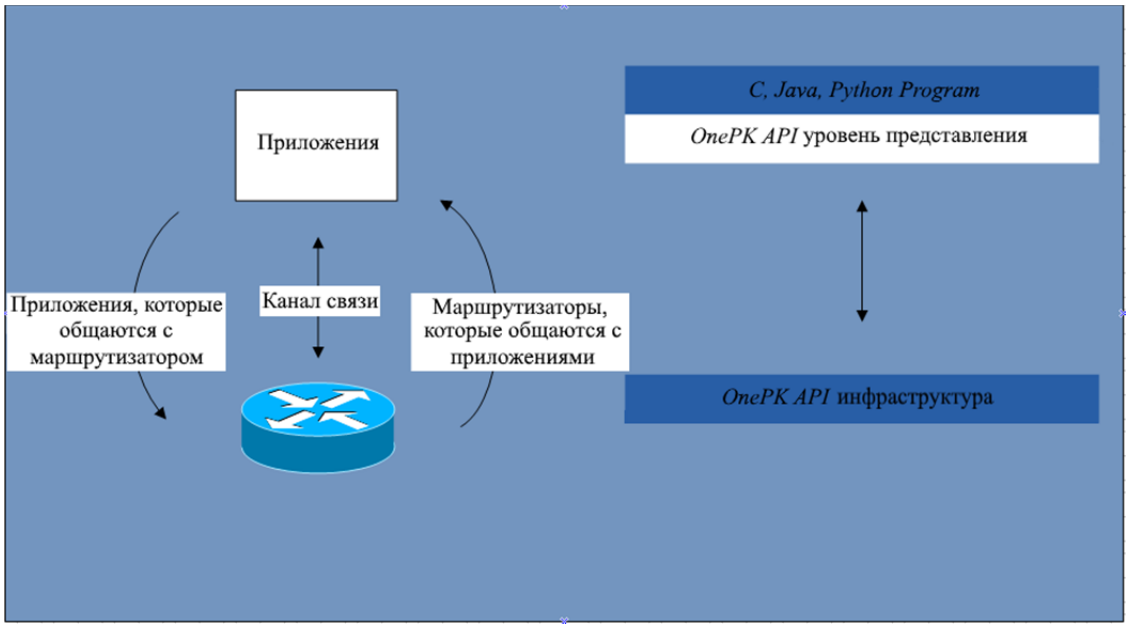

Рис. 5. Архитектура onePK

Существует несколько моделей развертывания onePK приложений. Приложения onePK могут быть развернуты:

- на устройствах *Cisco* (коммутаторах, маршрутизаторах);
- на интегрированных в устройства *Cisco* вычислительных элементах;
- на внешних серверах.

При реализации механизма была выбрана модель развертывания приложений на внешнем сервере приложения onePK, так как запускаются на внешних серверах, связанных по IP с сетевыми устройствами. Принцип работы данной модели развертывания проиллюстрирован на рис. 6.

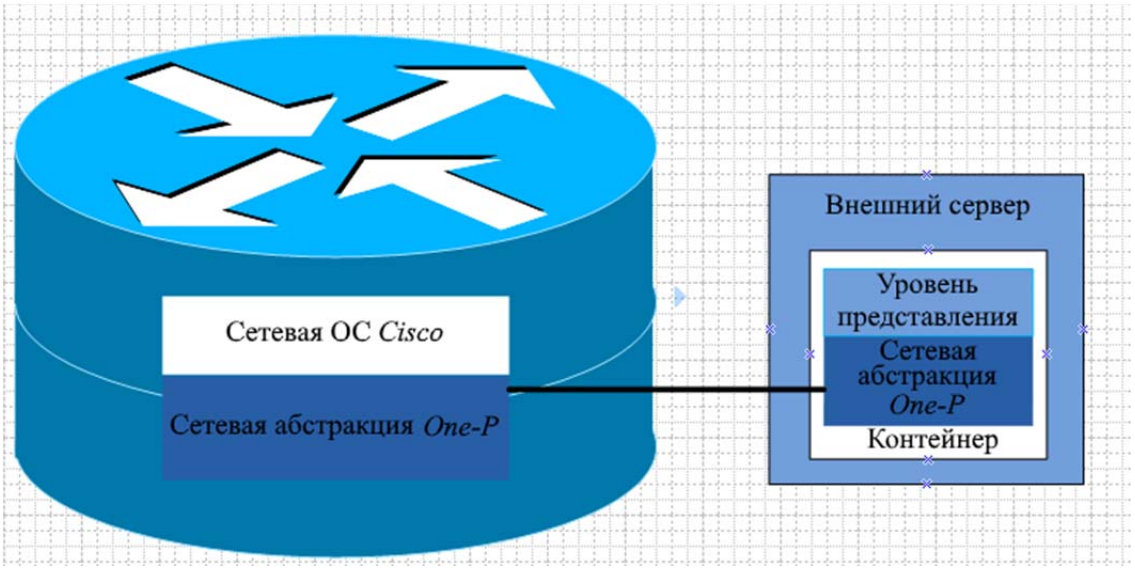

Рис. 6. Модель развертывания приложений на внешнем сервере

Данная модель позволяет разработчикам и администраторам сетевой инфраструктуры выбирать платформу для работы по собственным критериям. Платформы (серверные) могут быть использованы следующие:

– серверы под управление операционной системы GNU/Linux;
– серверы под управлением операционной системы Windows;
– мобильные устройства под управлением операционных систем Android или iOS (Apple).

Эти платформы могут запускать выполнение onePK либо в отдельных программных контейнерах, либо в пространстве операционной системы сервера. Такой подход обеспечивает наибольший уровень изоляции onePK приложений.

Реализация механизма пересчета нагрузки

Программа, реализующая предложенный алгоритм учета нагрузки, написана на объектно-ориентированном языке программирования Java. Выбор в качестве языка программирования Java обусловлен его многоплатформенностью [12], а также большим количеством материалов и готовых библиотек. При этом использованы специальные библиотеки API для onePK, созданные специально для языка Java. Структура программы представлена на рис. 7.

Программа состоит из трех основных компонентов:
– DeviceSetup – модуль собственно реализующий соединение к API onePK устройства;
– PinningHandler – модуль верификации TLS сессии;
– Recalculation – модуль считывающий параметры входящей и передающей нагрузке, а так же осуществляющей ее пересчет по формуле (3) и передающей новое значение на устройство.

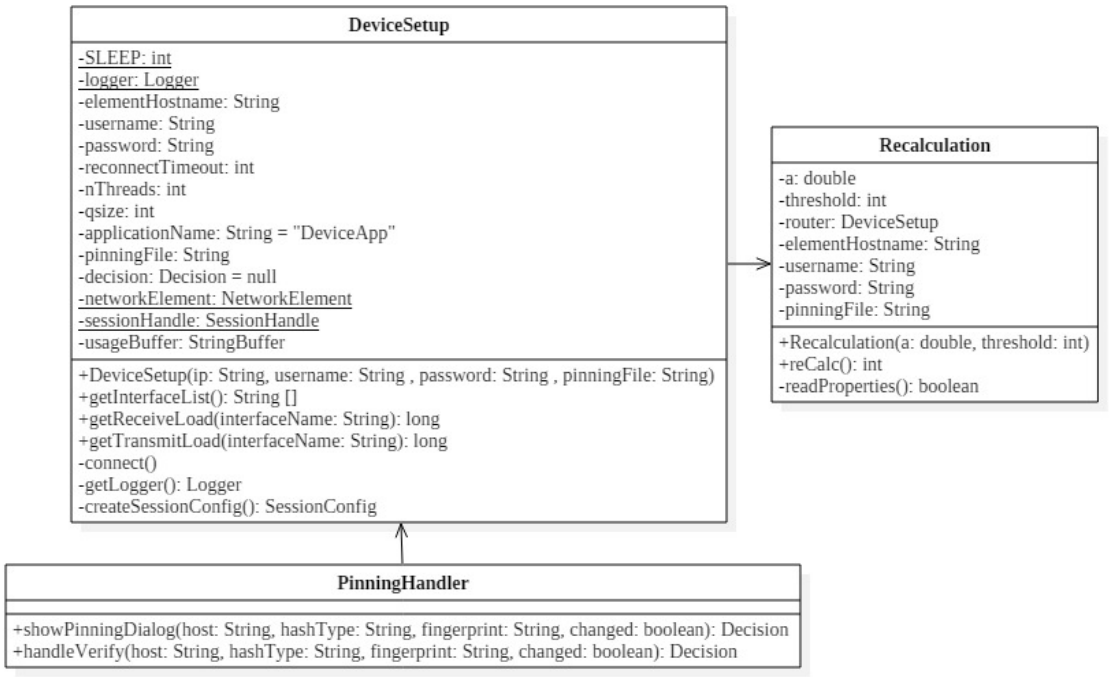

Рис. 7. UML диаграмма классов

Заключение

Рассмотренное решение предлагается в рамках концепции программно-определяемой сети. Эта концепция выносит плоскость управления сетью на но-

вый архитектурный уровень, что позволяет принимать решения на основе большого количества параметров, относящихся к сети в целом, что значительно увеличивает возможности применения алгоритма, а также повышает качество и целесообразность его решений. Таким образом, представленное решение обладает следующими характеристиками:

– способность получать данные от сетевых устройств о нагрузке в сети и ее изменении;

– централизованный анализ и обработка полученных данных на контроллере;

– применение адаптивных алгоритмов для принятия решений, в соответствии с изменениями в нагрузке;

– концентрация процессорной нагрузки на контроллере.

Применение описанного в статье метода, позволяет, во-первых, стабилизировать работу EIGRP, и, во-вторых, обеспечить больший управляющий контроль над IP-сетью передачи данных, что, в совокупности, позволяет предотвратить отказы в обслуживании и обеспечить свойство доступность

Литература

1. Алейников А. А., Билятдинов К. З., Красов А. В., Левин М. В. Контроль, измерение и интеллектуальное управление трафиком. СПб. : Астерион", 2016.

2. Vladyko A., Letenko I., Lezhepekov A., Buinevich M. Fuzzy Model of Dynamic Traffic Management in Software-Defined Mobile Networks // Lecture Notes in Computer Science. 2016. Vol. 9870. pp. 561–570.

3. Летенко И. Д. Нечеткая модель динамического управления трафиком в программируемых сетях // Системы управления и информационные технологии. 2015. Т. 62. № 4.1. С. 179–184.

4. Красов А. В., Левин М. В., Цветков А. Ю. Управления сетями передачи данных с изменяющейся нагрузкой // Всероссийская научная конференция по проблемам управления в технических системах. Санкт-Петербург, 2015. С. 141–146.

5. Красов А. В., Левин М. В. Возможности управления трафиком в рамках концепции SDN // IV Международная научно-техническая и научно-методическая конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании»: сборник научных статей в 2 т. 2015. С. 350–354.

6. Doyle D., Carroll J. Routing TCP/IP Vol. 1. Cisco Press, 2005. 936 p.

7. Doyle D., Carroll J. Routing TCP/IP Vol. 2. Cisco Press, 2001. 976 p.

8. White R., Slice D., Retana A. Optimal Routing Design. – Cisco press, 2005. 504 p.

9. Pepelnjak I. EIGRP network design solutions. Cisco press, 2000. 384 p.

10. Zinin A. Cisco IP Routing: packet forwarding and Intra-domain routing. Addison Wesley Professional, 2001. 656 p.

11. Каменский Г. А. Лекции по теории функций комплексного переменного, операционному исчислению и теории разностных уравнений: учебное пособие для вузов. М.: ВШ, 2008. 160 с.

12. Weisfeld M. The Object-Oriented thought process. Addison Wesley, 2009. 309 p.

13. Scratch S. R. Object-Oriented and classical software engineering. McGraw Hill, 2007. 654 p.

References

1. Aleynikov A. A., Bilyatdinov K. Z., Krasov A. V., Levin M. V. Monitoring, measurement and intelligent traffic management. SPb. : Asterion, 2016.

2. Vladyko A., Letenko I., Lezhepekov A., Buinevich M. Fuzzy Model of Dynamic Traffic Management in Software-Defined Mobile Networks // Lecture Notes in Computer Science. 2016. Vol. 9870. pp. 561–570.

3. Letenko I. D. The Fuzzy Model of Dynamic Traffic Management in Programmable Networks // Sistemy upravlenija i informacionnye tehnologii. 2015. T. 62. No. 4.1. S. 179–184.
4. Krasov A. V., Levin M. V., Tcvetkov A. Y. (2015) Control of data networks with dynamic changeable load // Scientific Conference on Control in technical systems. Saint Petersburg, 2015. pp. 141–146.
5. Krasov A. V., Levin M. V. Traffic Management Capabilities within the SDN concept // IV International Conference "Actual Problems of Information and Telecommunications in Science and Education" Collection of Scientific Articles in 2 vol. SPb: SPBGUT. 2015. pp. 350–354.
6. Doyle D., Carroll J. Routing TCP/IP Vol. 1. Cisco Press, 2005, pp. -936.
7. Doyle D., Carroll J. Routing TCP/IP Vol. 2. Cisco Press, 2001,pp. 976.
8. White R., Slice D., Retana A. Optimal Routing Design. Cisco press, 2005, pp. 504.
9. Pepelnjak I. EIGRP network design solutions. Cisco press, 2000, pp. 384.
10. Zinin A. Cisco IP Routing: packet forwarding and Intra-domain routing. Addison Wesley Professional, 2001,pp. 656.
11. Kamensky G. A. Lectures on the Theory of Functions of a Complex Variable, Operational Calculus and the Theory of Differential Equations: Textbook for High Schools. M.: VSH. 2008. 160 p.
12. Weisfeld M. The Object-Oriented thought process. Addison Wesley, 2009, pp. 309.
13. Scratch S. R. Object-Oriented and classical software engineering. McGraw Hill, 2007, pp. 654.

Красов Андрей Владимирович – кандидат технических наук, доцент, заведующий кафедрой, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, krasov@pisem.ru

Левин Марк Вадимович – ассистент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, m.va.levin@gmail.com

Цветков Александр Юрьевич – старший преподаватель, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, alexander.tsvetkov@gmail.com

Krasov Andrey – Ph.D, assistant professor, head of the Department, SPbSUT, St. Petersburg, 193232, Russian Federation, krasov@pisem.net

Levin Mark – assistant, SPbSUT, St. Petersburg, 193232, Russian Federation, m.va.levin@gmail.com

Tcvetkov Aleksandr – senior lecturer, SPbSUT, St. Petersburg, 193232, Russian Federation, alexander.tsvetkov89@gmail.com