УДК 004.722
DOI: 10.25559/SITITO.14.201802.355-367

МЕТОД ВЫБОРА КОНФИГУРАЦИИ РАСПРЕДЕЛЕННОЙ ПЛАТФОРМЫ УПРАВЛЕНИЯ ВЫСОКОЙ ГОТОВНОСТИ ДЛЯ ТРАНСПОРТНЫХ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

В.Н. Пашков
Московский государственный университет имени М.В. Ломоносова, г. Москва, Россия

METHOD OF CHOOSING A CONFIGURATION OF A HIGH AVAILABILITY DISTRIBUTED CONTROL PLATFORM FOR TRANSPORT SOFTWARE-DEFINED NETWORKS

Vasily N. Pashkov
Lomonosov Moscow State University, Moscow, Russia

Ключевые слова

Программно-конфигурируемые сети (ПКС); протокол OpenFlow; распределенный контроллер; распределенная платформа управления; архитектура сети; готовность; отказоустойчивость; отказ контроллера; резервный контроллер, размещение контроллеров.

Аннотация

В работе рассматривается задача выбора конфигурации распределенной платформы управления (РПУ) с высокой степенью готовности для транспортных программно-конфигурируемых (ПКС) сетей. Для обеспечения устойчивости РПУ к единичным отказам контроллеров в рамках данной проблемы затрагиваются следующие аспекты: выбор размещения контроллеров РПУ в узлах транспортной ПКС сети, выбор основных и резервных контроллеров для каждого коммутатора сети с использованием наименьшего количества активных контроллеров распределенной платформы управления. В качестве критерия оптимизации размещения контроллеров рассматривается минимизация задержек на передачу управляющих сообщений между контроллером и коммутатором. В случае единичного отказа контроллера РПУ управление коммутаторами распределяется между исправными контроллерами таким образом, чтобы количество коммутаторов, управляемое одним контроллером, не превышало заданного максимального числа.
Предлагается комплексный метод решения данной задачи, включающий в себя алгоритмы на графах, алгоритмы кластеризации и алгоритм решения задачи булевого линейного программирования, позволяющий выбрать размещение контроллеров РПУ в сети и конфигурацию основного и резервного контроллера для каждого коммутатора сети. Предложенный метод реализован в виде программного средства. Проведено экспериментальное исследование разработанного метода на топологиях реальных транспортных сетей.

Keywords

Software defined networks (SDN); OpenFlow protocol; distributed controller; distributed control platform; network architecture; high availability; fault tolerance; controller failure; backup controller, controller placement.

Abstract

The method of choosing a configuration of a high availability distributed control platform (HA DCP) for transport software-defined networks is considered. To provide fault tolerance to single controller failures the following aspects are affected in this problem: the choice of the controller instance location in the nodes of the transport SDN, the choosing of the primary and backup controllers for each switch, determining the optimal number of active controllers for distributed control platform. The minimization of delays in the transmission of control messages between the controller and the switch is a criterion for optimizing the controller placement. In the case of a single controller failure, the management of the switches is distributed among the active backup controllers.
A complex method for solving this problem is proposed, including algorithms on graphs, clustering algorithms, and an algorithm for solving the Boolean linear programming problem, which allows selecting the location of the controllers in the network and the configuration of the primary and backup controller for each switch. The proposed method is implemented as a software tool. An experimental investigation of the developed method on the topologies of real transport networks is carried out.

Об авторе:

Пашков Василий Николаевич, программист, лаборатория вычислительных комплексов, кафедра автоматизации систем и вычислительных комплексов, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1); ORCID: http://orcid.org/0000-0001-5783-4557, pashkov@lvk.cs.msu.su

Введение

В последние годы наблюдаются новые тенденции в развитии сетевых сервисов: получили широкое распространение мобильные устройства и приложения, развивается направление облачных сервисов, в частности, Infrastructure-as-a-Service, постоянно растут объемы передаваемых в сетях данных. Однако используемые на сегодняшний день компьютерные сети используют технологии, основы которых были заложены более двадцати лет назад. Несмотря на активное развитие различных сетевых технологий, основные концепции и архитектура построения компьютерных сетей остались неизменными. Поддержка больших сетей с динамически изменяющейся топологией требует работы высококвалифицированных специалистов и времени на изменение конфигурации сетевых устройств. При этом внесение изменений в процесс построения сетей и управления ими является трудоемким и дорогостоящим и зачастую требует участия производителей сетевого оборудования, так как многие технологии являются проприетарными. Поэтому специалистами университетов Стэнфорда и Беркли была предложена новая архитектура компьютерных сетей — программно-конфигурируемые сети (ПКС, Software Defined Networks, SDN) [1].

Можно выделить следующие особенности технологии ПКС [2]:

1. Разделение функций управления сетью и передачи данных;
2. Централизация управления в сети;
3. Использование открытого протокола для управления сетевыми устройствами.

В традиционной архитектуре компьютерных сетей функции управления и передачи данных выполняются одним устройством, что усложняет контроль и управление. Сетевое оборудование поддерживает работу с множеством протоколов и самостоятельно принимает решение об обработке и маршрутизации трафика в сети в соответствии с тем или иным алгоритмом. Настройка всего сетевого оборудования сети в соответствии с заданными требованиями также является нетривиальной и трудоемкой задачей.

ПКС предполагает разделение функций управления сетью и передачи данных. Реализация функции передачи данных остается в сетевом оборудовании, а управление сетью осуществляется сетевой операционной системой (или контроллером). Таким образом, предлагается перейти от управления отдельными сетевыми устройствами к управлению сетью в целом.

Управление передачей данных в сети сводится к написанию приложений, реализующих требуемую функциональность, которые взаимодействуют с сетью посредством сетевых сервисов, предоставляемых контроллером. Контроллер представляет собой программное обеспечение, работающее на некотором сервере. Контроллер предоставляет приложениям программный интерфейс для управления сетевым оборудованием. Таким образом, приложения могут работать с некоторым виртуальным представлением сети, абстрагируясь от ее реальных физических характеристик. Взаимодействие между контроллерами и сетевым оборудованием происходит по открытому протоколу. Использование открытого протокола позволяет не зависеть от характеристик и внутреннего устройства сетевого оборудования при написании управляющих приложений, а также использовать оборудование различных производителей, если оно поддерживает протокол взаимодействия с контроллером.

Таким образом, управление сетью в ПКС логически централизованно. С одной стороны, это упрощает задачу управления сетью, позволяя целиком сосредоточить управление в контроллере. С другой стороны, это неизбежно ведет к появлению узкого места в сети — контроллера.

В настоящее время известны около двух десятков реализаций сетевых ОС для программно-конфигурируемых OpenFlow сетей: NOX [3], POX, Beacon [4], FloodLight [5], RUNOS [6] и другие. По результатам исследования производительности контроллеров [7] было показано, что одного контроллера может быть недостаточно для управления транспортной сетью региона, континента или центра обработки данных. Отказ контроллера или сервера, на котором он работает, может привести к сбою в работе всего управляемого им оборудования и, следовательно, к отказу всей сети. Отказ физического канала между сервером контроллера и управляемой сетью также приведет к нарушению работы сети.

Другая проблема, связанная с централизацией управления — размещение контроллеров в сетях большого масштаба (городских, масштаба страны или континента), в которых с увеличением расстояний растут задержки на передачу данных между узлами. Особенность технологии ПКС состоит в том, что для принятия решения о маршрутизации нового вида трафика сетевому оборудованию необходимо обратиться с запросом к контроллеру, что приводит к задержкам при установлении соединений между конечными узлами сети.

Одним из основных способов преодоления данных проблем производительности, масштабируемости и надежности является резервировании контроллеров и построение распределенной платформы управления с использованием множества контроллеров, размещенных в разных узлах сети. В этом случае при отказе одного контроллера функции управления сетевым устройством будут переданы резервному контроллеру. В настоящее время известны распределенные контроллеры:

- с одноуровневой архитектурой: закрытые реализации HyperFlow [8], Onix [9], ElastiCon [10], Beehive[11] и открытая реализация ONOS [12].
- с иерархической (многоуровневой) архитектурой: закрытые реализации Kandoo [13] и Espresso [14].

Таким образом, возникает задача выбора конфигурации распределенной платформы управления с высокой степенью готовности, для решения которой необходимо разрешить следующие вопросы:

1. Какое количество контроллеров необходимо для обеспечения надежности управления транспортной ПКС сетью?
2. Где разместить контроллеры и как распределить управление коммутаторами между ними?
3. Как перераспределить управление коммутаторами в случае одиночного отказа контроллера РПУ?

В статье рассматриваются особенности программно-конфигурируемых сетей, подходы к обеспечению устойчивости РПУ ПКС к единичным отказам контроллеров, приводится формальная постановка задачи и обзор работ по теме исследования, предлагается метод решения задачи выбора конфигурации распределенной платформы управления с высокой степенью готовности, приводится описание реализации и результатов экспериментального исследования.

1 Программно-конфигурируемые сети

В основе ПКС лежит идея разделения уровня передачи данных и уровня управления. Архитектура ПКС состоит из следующих логических уровней [15]:

1. Уровень передачи данных (Data Plane). Это уровень сетевого оборудования (как физического, так и виртуального), которое предоставляет единый интерфейс управления вышестоящему уровню.
2. Уровень управления (Control Plane). На этом уровне работает сетевая операционная система (контроллер) и приложения, управляющие сетью.
   (a) Контроллер управляет уровнем передачи данных через стандартный протокол взаимодействия. Он обеспечивает базовую функциональность для взаимодействия с сетевым оборудованием, например, установление и поддержание соединения, и реализует некоторые служебные сервисы, такие как сбор информации о топологии сети. Также контроллер предоставляет приложениям программный интерфейс для управления сетью и обеспечивает взаимодействие между различными приложениями и сервисами контроллера.
   (b) Приложения реализуют произвольную функциональность, необходимую администратору. На контроллере может работать набор приложений, которые могут взаимодействовать друг с другом и использовать сервисы контроллера, при этом каждое приложение может иметь свое собственное абстрактное видение сети.

Архитектура ПКС имеет следующие преимущества по сравнению с традиционной сетевой архитектурой:

1. Простота управления. За счет централизации управления упрощается контроль сети. Например, в сетях центров обработки данных необходимо быстро реагировать на появление новых узлов сети и на их перемещение, поэтому ручное управление сетевым оборудованием может быть неэффективно.
2. Сокращение расходов. Сетевое оборудование в ПКС выполняет только функцию передачи данных, что ведет к упрощению оборудования и, следовательно, к снижению его стоимости. Также за счет централизации управления сокращаются расходы на обслуживание сетевой инфраструктуры.
3. Гибкость управления. Использование приложений для управления сетью открывает широкие возможности для задания гибких политик в сети и реализации необходимого функционала. Реализация функций управления сетью в виде программного обеспечения позволяет сократить срок внедрения новых технологий, так как отсутствует необходимость внесения изменений в физическую структуру сети.
4. Независимость от производителей сетевого оборудования. Использование открытого протокола для взаимодействия между контроллером и сетевым оборудованием позволяет не зависеть от производителей сетевого оборудования. В одной сети может быть использовано оборудование различных производителей, управляемое контроллером по единому протоколу. Все приложения контроллеров будут иметь единое абстрактное представление сети, не зависящее от физических характеристик оборудования.

2 Обзор подходов к обеспечению отказоустойчивости в ПКС

Недостатком централизации управления сетью является то, что контроллер становится единой точкой отказа. В случае отказа контроллера или потери соединения между контроллером и коммутаторами правила, внесенные в таблицу, будут удалены после истечения срока их действия. Коммутаторы не смогут получить новые правила для установления новых потоков в сети, и, как следствие, функционирование сети будет нарушено.

Возможны следующие причины потери соединения между контроллером и коммутатором:

- Программные ошибки в контроллере или в работающих на нем приложениях.
- Отказы физического оборудования (на сервере, на котором запущен контроллер).
- Отказы в сети управления: отказ соединения или коммутаторов, через который сервер контроллера соединен с сетью.

Можно выделить следующие подходы к восстановлению работы программно-конфигурируемой сети в случае отказа контроллера.

Первый подход, описанный в спецификации OpenFlow [17], предполагает, что в случае потери соединения с контроллером коммутатор переходит в аварийный режим работы, в котором обработка пакетов осуществляется согласно установленным на коммутаторе аварийным правилам (например, осуществляется обычная L2 коммутация). Этот подход, во-первых, требует поддержки коммутатором возможности работы без связи с контроллером, а во-вторых, очевидно, не может гарантировать корректного функционирования сети в случае отказа контроллера. Приложения, работающие на контроллере, могут реализовывать довольно сложную логику управления сетью или важные политики безопасности. Например, если на контроллере было запущено приложение, выполняющее функцию firewall, при отказе контроллера запрещенный ранее трафик может проникнуть в сеть. Таким образом, данный подход нельзя считать корректной схемой восстановления работы сети.

Другой подход — резервирование контроллеров. Этот подход используется, например, в работе [18], а также механизм резервирования реализован во всех коммерческих контроллерах ПКС. Резервирование является стандартным методом обеспечения отказоустойчивости вычислительных систем. Предполагается, что используются дополнительные (резервные) контроллеры, работающие на отдельных серверах. Резервные контроллеры идентичны основным контроллерам. В случае отказа основного контроллера управление передается одному из резервных контроллеров.

Традиционно резервирование аппаратных компонент разделяют на активное (или горячее) резервирование, теплое резервирование и резервирование с замещением (или холодное) [18].

При резервировании с замещением в системе функционирует только один (основной) компонент, резервный компонент при этом находится в отключенном состоянии. В случае отказа основного компонента резервный компонент переводится в ра-

бочее состояние, и ему передаются функции основного компонента.

При использовании активного резервирования в системе одновременно работают два (или более) идентичных компонента, при этом одни и те же входные данные обрабатываются компонентами параллельно. Таким образом, в случае отказа одного компонента второй может немедленно принять на себя все управление.

Для транспортных ПКС приемлемой является только схема с активным резервированием контроллеров, так как необходимо сократить время передачи управления в случае отказа основного контроллера, а дополнительные задержки на приведение резервного контроллера в работающее состояние могут привести к потере пакетов в сети и задержкам при установлении новых соединений.

Для обеспечения корректной передачи управления коммутаторами основной и резервный контроллеры должны использовать одинаковую сетевую ОС, иметь одинаковый набор запущенных приложений и постоянно синхронизировать данные о состоянии сети.

Использование активного резервирования контроллеров имеет следующие преимущества:

1. Корректность работы. В случае отказа основного контроллера логика управления сетью не меняется, так как на резервном контроллере работают те же самые приложения, что работали и на основном.
2. Быстрое восстановление. Так как резервный контроллер имеет актуальную информацию о сети, необходимый набор приложений и постоянно отслеживает состояние основного контроллера, время обнаружения отказа и перехвата управления невелико.

Однако использование активного резервирования имеет свои недостатки:

1. Дополнительные требования к контроллерам и коммутаторам. Для использования активного резервирования должны быть реализованы механизмы обнаружения отказа основного контроллера и перехвата управления резервным контроллером, а также поддержания актуальной информации о сети на резервном контроллере. В зависимости от выбранных механизмов, их поддержка может требоваться только от контроллеров или как от контроллеров, так и от коммутаторов.
2. Дополнительное оборудование. Для работы резервного контроллера требуется использование дополнительного сервера, что приводит к увеличению стоимости решения в целом.

Также желательно обеспечить рациональное использование имеющихся ресурсов при использовании резервирования. Например, вместо использования отдельного резервного контроллера для каждого из основных контроллеров сети можно распределить управление коммутаторами между всеми исправными контроллерами, а в случае отказа одного из контроллеров перераспределять управляемые им коммутаторы между оставшимися контроллерами.

Пример схемы распределения управления коммутаторами в случае отказа контроллера представлен на рисунке 1. Каждый контроллер управляет своим сегментом сети. В сегмент, управляемый контроллером 2, входят коммутаторы 1, 2 и 3. В случае отказа контроллера 2 управление коммутатором 1 будет передано контроллеру 1, а управление коммутаторами 2 и 3 — контроллеру 3.

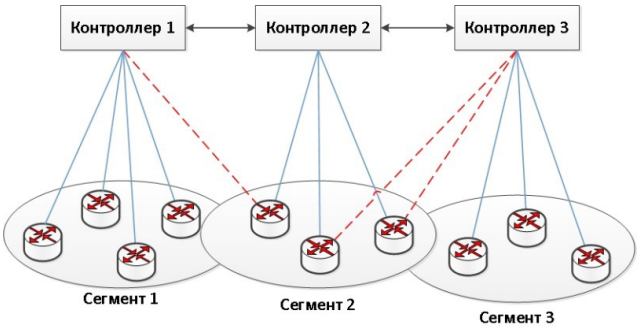

Рис. 1. Распределение управления коммутаторами между резервными контроллерами при отказе Контроллера 2

Fig. 1. Distribution of switch control between spare controllers in the event of Controller 2 failure

3. Постановка задачи

Пусть дана транспортная программно-конфигурируемая сеть, в узлах которой расположены OpenFlow-коммутаторы. Топология сети зафиксирована, для каждого канала связи задана задержка на этом канале. Данная сеть может управляться несколькими контроллерами, образующими РПУ ПКС.

Управление коммутаторами может осуществляться как out-of-band (с использованием выделенной сети управления), так и in-band (то есть сеть передачи данных используется также для передачи управляющих сообщений между коммутаторами и контроллером).

Пусть задан тип контроллера, который будет использоваться для управления сетью. Тип контроллера характеризуется максимальным числом коммутаторов, которыми может управлять один контроллер. Контроллер имеет механизм обнаружения отказов контроллеров и перехвата управления управляемыми ими коммутаторами, а также необходимые механизмы синхронизации с другими контроллерами. Контроллер может располагаться в любом узле данной сети.

Для каждого коммутатора сети могут быть заданы основной и резервный контроллеры. При штатной работе сети коммутатором управляет его основной контроллер, в случае отказа основного контроллера управление коммутатором передается резервному контроллеру. Каждый контроллер может являться для одних коммутаторов основным контроллером, а для других — резервным.

Будем называть сегментом сети все коммутаторы, имеющие один и тот же основной контроллер (т. е. в штатной ситуации все они управляются одним контроллером). Заметим, что коммутаторы из одного сегмента могут иметь разные резервные контроллеры.

Сеть должна быть устойчивой к отказу одного контроллера. Это значит, что в случае отказа любого из контроллеров после передачи управления коммутаторами из управляемого им сегмента сети их резервным контроллерам число коммутаторов, управляемых каждым исправным контроллером, не должно превышать максимального допустимого количества коммутаторов, которым может управлять один контроллер.

Следовательно, требуется определить количество кон-

троллеров заданного типа, необходимых для управления данной сетью, с учетом требования устойчивости сети к отказу одного контроллера.

Далее необходимо решить задачу выбора узлов сети, в которые следует поместить контроллеры, а также для каждого коммутатора определить основной и резервный контроллер. При этом следует учитывать следующее:

Принимая во внимание то, что рассматривается транспортная сеть большого масштаба, критерием оптимальности размещения контроллеров является минимизация задержек на передачу управляющих сообщений между контроллером и коммутаторами (как в штатном режиме работы сети, так и в случае отказа одного контроллера). При этом под минимизацией задержек может пониматься как минимизация средней задержки, так и минимизация наибольшей задержки между всеми контроллерами и коммутаторами.

Так как контроллерам сети необходимо поддерживать согласованное состояние (например, иметь одинаковое представление сети и полный набор установленных всеми контроллерами правил), а также своевременно определять отказ другого контроллера для перехвата управления коммутаторами, требуется обеспечить надежные соединения между контроллерами [18]. В теории надежности сетей в данном случае используется понятие структурной живучести, которое подразумевает сохранение возможности передачи информации между узлами даже в случае отказа части узлов или соединений. Для обеспечения живучести обычно используется принцип двухсвязности [19] (рис. 2): если между двумя узлами должно существовать соединение, обладающее свойством живучести, то между этими узлами должны существовать два независимых пути (т. е. не имеющих общих промежуточных узлов и соединений). Таким образом, в случае отказа какого-либо узла или соединения из одного пути между узлами информация может быть передана по другому пути. Следовательно, контроллеры в сети требуется разместить таким образом, чтобы между любой парой контроллеров существовало не менее двух независимых путей (предполагается, что хотя бы один вариант такого размещения в сети существует).

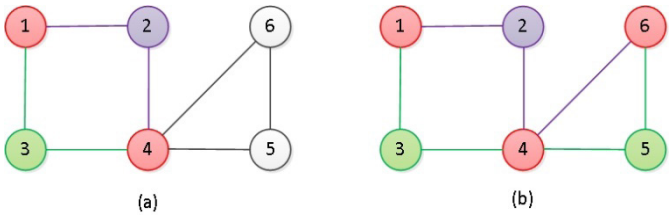

Рис. 2. Условие двухсвязности
Fig. 2. The condition of two-bin

Между узлами 1 и 4 выполнено условие двухсвязности(a): существуют два независимых пути через вершины 2 и 3 соответственно. Между узлами 1 и 5 условие двухсвязности не выполняется(b): вершина 4 будет общей для всех путей от 1 к 6.

Решением поставленной задачи будет являться исходная сеть, в которой определены узлы размещения контроллеров и для каждого коммутатора заданны основной и резервный контроллеры. При этом количество контроллеров должно быть минимальным необходимым для управления данной сетью, а размещение контроллеров будет удовлетворять указанным выше требованиям.

Для формализации постановки задачи введем следующие обозначения:

- $G = (S, E)$ — граф, описывающий сеть.
- S — множество коммутаторов в сети (каждый коммутатор является узлом сети, в узле, помимо коммутатора, может быть размещен контроллер).
- $|S|$ — число коммутаторов в сети. Все коммутаторы пронумерованы от 1 до $|S|$.
- $E = \{e_{ij} | e_{ij} = (s_i, s_j)\}$ — множество ребер графа, соответствующее множеству соединений между коммутаторами.
- $d^0_{i,j}$, где $i, j \in V$ — вес ребра между соседними вершинами i и j, пропорциональный задержке между соответствующими узлами.
- d^j_i, где $i, j \in V$ — длина кратчайшего пути в графе между вершинами i и j, пропорциональная величине задержки при передаче данных между соответствующими узлами сети по этому пути. Длина пути равна сумме входящих в этот путь ребер.
- C — множество контроллеров, управляющих сетью. Все контроллеры пронумерованы, номер контроллера совпадает с номером узла, в котором он расположен.
- $|C|$ — число контроллеров, управляющих сетью.
- $S' = \{S'_1 \dots S'_{|C|}\}$ — разбиение сети на сегменты, то есть коммутатор $s_i \in S$, $s_i \in S'_j$ управляется контроллером $c_j \in C$.
- $S'_{primary}$ — разбиение сети, в котором каждый сегмент S'_c содержит коммутаторы, для которых контроллер с является основным.
- S'_{backup} — разбиение сети, в котором каждый сегмент S'_c содержит коммутаторы, для которых контроллер с является резервным.
- N_{max} — максимальное число коммутаторов, которые могут управляться одним контроллером.

Также введем обозначения для двух метрик минимизации задержки:

1. Величина средней задержки между контроллером и коммутатором в сети (average-case latency):

$$L_{avg}(S') = \frac{1}{|S|}\sum_{i \in S} d^j_i, \quad c_j \in C, \quad s_i \in S'_j \quad (1)$$

2. Величина максимальной задержки между контроллером и коммутатором в сети (worst-case latency):

$$L_{wc}(S') = \max_{i \in S} d^j_i, \quad c_j \in C, \quad s_i \in S'_j \quad (2)$$

С использованием введенных обозначений задачу выбора конфигурации распределенной платформы управления ПКС с высокой степенью готовности можно сформулировать следующим образом.

Заданы:

1. сеть $G = (S, E)$;
2. величина задержки между каждой парой коммутаторов i и j: $d^0_{i,j}$;
3. N_{max};
4. метрика минимизации задержки L, равная L_{avg} или L_{wc}.

Требуется найти такие C, $S'_{primary}$ и S'_{backup}, что:

1. для любых двух контроллеров $c_1, c_2 \in C$ существуют два независимых пути;
2. $\min_{S'_{primary}} L$, где минимум берется по всем возможным разбиениям сети на сегменты;
3. $\forall s \in S$: $s \in S'_i$, $s \in S'_j$, где $S'_i \in S'_{primary}$, $S'_j \in S'_{backup}$: $c_i \neq c_j$;
4. $\min_{S'_{backup}} L$, где минимум берется по всем возможным разбиениям сети на сегменты, таким образом, чтобы разбиение удовлетворяло условию 3 при фиксированном согласно условию 2 размещении контроллеров;
5. $\min|C|$: $\forall S'_i \quad |S'_i| \leq N_{max}$, где $S'_i \in S'_{primary}$ или $S'_i \in S'_{backup}$.

4 Обзор подходов к выбору конфигурации распределенного управления в ПКС сетях

Задача размещения контроллеров в сети ПКС рассматривается в статье [20]. Авторы рассматривают проблему размещения контроллеров в транспортных сетях крупного масштаба (в качестве примера рассматривается сеть Internet2). В статье ставятся два вопроса:

1. Какое количество контроллеров необходимо для управления сетью и как число контроллеров влияет на время передачи сообщений между контроллером и коммутаторами?
2. Как выбрать место для размещения контроллера в сети с учетом минимизации времени передачи управляющих сообщений?

Вопрос минимизации задержек является важным при проектировании программно-конфигурируемых сетей большого масштаба, так как если контроллер и коммутатор находятся на значительном расстоянии друг от друга, время на передачу управляющих сообщений негативно сказывается на задержках при установлении новых потоков в сети.

Авторы вводят две метрики для оценки оптимальности размещения контроллеров в сети относительно задержек на передачу сообщений: средняя задержка для всех контроллеров в сети и максимальная задержка. В статье приведено исследование взаимосвязи этих метрик и показано, что при выборе одной из метрик в качестве критерия оптимальности во многих случаях решение не будет являться оптимальным по второй метрике.

В данной работе не предлагается каких-либо алгоритмов для оптимизации поиска места размещения контроллера, для поиска оптимального решения использовался перебор всех возможных вариантов, что требует много процессорного времени. Цель статьи — показать актуальность задачи выбора размещения контроллера в сети, так как использование нескольких контроллеров и выбор для них оптимального места позволяют значительно сократить задержки по сравнению со случайным размещением контроллеров.

Задача размещения контроллера с учетом требований отказоустойчивости рассматривается в статьях [21] и [22]. В данных работах решается задача повышения устойчивости сети к отказам физических каналов между коммутаторами и контроллерами.

Статья [21] посвящена решению задачи размещения нескольких контроллеров в сети таким образом, чтобы увеличить ее отказоустойчивость. Первая задача, рассматриваемая авторами, следующая: требуется разбить сеть на два сегмента таким образом, чтобы между каждым контроллером и коммутатором, которым он управляет, существовало как можно больше независимых путей в сети (т. е. путей, не имеющих общих соединений и узлов). Для этого предлагается использовать алгоритм кластеризации на графах, основанный на поиске минимального разреза, т. е. такого разбиения графа, при котором количество связей между двумя кластерами будет минимальным. Утверждается, что подобное разбиение позволяет максимизировать количество соединений между узлами одного сегмента сети.

Авторы приводят результаты исследований, показывающие, что увеличение длины пути между контроллером и коммутатором ведет к увеличению вероятности отказа некоторого элемента этого пути и, следовательно, потери связи между коммутатором и контроллером. Поэтому вторая задача, рассматриваемая в статье, это выбор такого места для размещения контроллера внутри заданного сегмента, чтобы среднее расстояние между контроллером и коммутатором было минимальным. Для этого контроллер помещается в центр графа, описывающего данный сегмент.

Следует заметить, что данный подход к разбиению графа на сегменты нацелен, прежде всего, на повышение отказоустойчивости соединений между контроллером и коммутатором. Минимизация длины пути между контроллером и коммутатором является вторичной задачей, поэтому полученное разбиение может не являться оптимальным с точки зрения минимизации задержек.

Статья [22] является продолжением работы [21]. В ней авторы предлагают следующий подход к организации соединения между контроллером и коммутатором: для каждого коммутатора задается основной и резервный физический порт, через который будут передаваться все управляющие сообщения (как между ним и контроллером, так и сообщения других коммутаторов). По умолчанию обмен сообщениями ведется через основной порт, однако если коммутатор обнаруживает отсутствие соединения с контроллером по данному порту, он переключается на передачу сообщений через резервный порт. При этом дополнительные соединения в существующую топологию не добавляются.

Авторы предлагают размещать контроллер в сети таким образом, чтобы максимизировать количество коммутаторов, для которых возможно использование основного и резервного портов, так, чтобы это не приводило к зацикливанию сообщений в сети (такие узлы сети называются защищенными). Рекурсивно водится понятие веса узла: вес узла равен количеству защищенных соседей плюс сумма весов всех соседей. Для поиска оптимального решения предлагается аппроксимационный алгоритм, который рекурсивно просматривает и помечает все узлы сети, вычисляя их вес, до тех пор, пока не останутся не помеченные узлы, не имеющие не помеченных соседей. Вес таких узлов берется равным 1, если они защищенные, и 0 — в противном случае.

В статье предлагается жадный алгоритм для построения дерева маршрутизации управляющих сообщений при заданном расположении контроллера. Данный алгоритм также направлен на максимизацию числа защищенных контроллеров в сети.

Достоинством данного подхода является быстрое восстановление соединения с контроллером через резервный порт в случае отказа основного соединения, так как коммутатор не тратит время на поиск альтернативного маршрута до контроллера. Однако отказ от поиска альтернативного пути приводит к неполному использованию ресурсов сети, так как может возникнуть ситуация, когда коммутатор, применяющий данную схему обеспечения отказоустойчивости, не сможет использовать резервный порт (это приведет к зацикливанию управляющего трафика), несмотря на то, что фактически в сети будут существовать альтернативные маршруты между ним и контроллером.

В работах [23] и [24] для каждого физического соединения, каждого коммутатора и каждого контроллера определена его надёжность. Требуется разместить контроллеры в сети так, чтобы для каждого коммутатора вероятность потери соединения со всеми контроллерами не превышала заданного значения, а количество контроллеров было минимально.

В работе [25] авторы дополняют метрики, введённые в работе [24] четырьмя метриками:

- максимальная задержка между коммутатором и управляющим контроллером среди всех сценариев отказа *x* контроллеров,
- максимальное число бесконтроллерных коммутаторов среди всех сценариев отказа *x* контроллеров,
- максимальный дисбаланс среди всех сценариев отказа *x* контроллеров, где для каждого сценария дисбалансом называется максимальная разница в количестве управляемых коммутаторов среди всех контроллеров,
- максимальная задержка между контроллерами.

Авторы разработали фреймворк, который ищет решение задачи размещения контроллеров на границе Парето для вышеупомянутых метрик, однако алгоритм работы фрейморка не публикуется.

Проведенный обзор существующих статей, посвященных вопросам размещения контроллеров в ПКС и резервирования контроллеров, показывает, что на сегодняшний день не существует работ, в которых при решении задачи размещения контроллеров учитывается возможность отказа контроллеров. Ни одна из статей по размещению контроллеров не подразумевает передачи управления коммутатором другому контроллеру в случае отказа основного контроллера. С другой стороны, в работах по резервированию контроллеров не поднимается вопрос выбора места для контроллеров с точки зрения рационального использования сетевых ресурсов и равномерного распределения нагрузки между несколькими контроллерами.

Также следует отметить отсутствие работ по обеспечению отказоустойчивости в транспортных программно-конфигурируемых сетях большого масштаба, в которых наряду с необходимостью обеспечить возможности для резервирования и передачи управления при отказе контроллера требуется также минимизировать задержки на передачу управляющих сообщений, величина которых может сильно повлиять на время установления новых соединений.

5. Разработка метода выбора конфигурации РПУ ПКС высокой готовности

Входными данными для метода выбора конфигурации распределенной платформы управления ПКС сети являются:

$D^0 = (d^0_{i,j})$ — матрица весов ребер между смежными вершинами графа , описывающего сеть (матрица смежности).

N_{max} — максимальное число коммутаторов, которыми может управлять один контроллер.

Метод включает в себя следующие основные этапы:

1. Определение потенциальных узлов сети, в которых могут быть размещены контроллеры РПУ. Для этого необходимо найти такие множества вершин графа , чтобы при любом размещении в этом множестве контроллеров для любой пары контроллеров выполнялось условие двухсвязности. Для нахождения таких множеств используется алгоритм поиска двухсвязных компонент в графе, основанный на обходе графа в глубину.

2. Определение количество контроллеров заданного типа, которое необходимо для управления данной сетью с учетом резервирования отказа одного контроллера.

3. Построение матрицы кратчайших путей для сети при помощи алгоритма Флойда-Уоршелла.

4. Разбиение сети на сегменты и выбор места расположения контроллера для каждого сегмента. Для разбиения сети на сегменты используются методы k-means или k-medians, в зависимости от выбранной метрики минимизации задержек. При расчете центров кластеров центры выбираются так, чтобы все они располагались в одной компоненте двухсвязности, найденной на шаге 1. Выбор начальных центров кластеров для методов k-means/k-medians осуществляется при помощи алгоритма k-means++. Сегменты сети выбираются таким образом, чтобы в каждом сегменте было не более коммутаторов.

5. Выбор для каждого коммутатора резервного контроллера. Данная задача сводится к задаче булевого линейного программирования. Для решения полученной задачи булевого ЛП используется аддитивный алгоритм (метод Балаша).

Результатом работы метода выбора конфигурации распределенной платформы управления ПКС сети являются являются:

- C — множество вершин, в которых располагаются контроллеры;
- для каждого коммутатора s∈S: основной контроллер $c_s^{primary}$ и резервный контроллер c_s^{backup}.

В рамках данной работы предложенный метод был реализован в виде программного средства на языке программирования C++ с использованием набора библиотек Qt. Программное средство включает в себя следующие основные модули:

- **Модуль обработки и анализа входных данных.** Модуль предназначен для преобразования входных данных в формате GraphML (на основе XML) во внутреннее представление программы.
- **Модуль алгоритмов**, содержащий реализации алгоритмов, которые применяются на разных этапах решения задачи:
  - Graphs. Алгоритмы на графах: реализация алгоритмов Флойда-Уоршелла и поиска двухсвязных компонент.
  - Clustering. Реализация алгоритмов k-means/k-medians и алгоритма выбора начальных центров кластеров k-means++. Настройки алгоритмов k-means/k-medians (Clustering Settings): максимальное количество итераций, условие останова (количество итераций без улучшения или с улучшением не больше, чем в раз).
  - BLP. Реализация аддитивного алгоритма для решения задачи булевского ЛП.
- **Основной модуль.** Реализует общий метод решения задачи, используя реализации соответствующих алгоритмов для решения отдельных подзадач. При помощи опций командной строки задаются входные данные и требования к решению (Global Settings).

7. Экспериментальная часть

В качестве входных данных для экспериментальных исследований была выбрана библиотека Internet Topology Zoo [26], содержащая данные о реальных транспортных сетях (масштаба страны и более). Данные о сетях представлены в формате GraphML.

Так как описания сетей из данной библиотеки содержат лишь географические координаты узлов сети, но не содержат информации о задержках в сети, задержка на соединениях между узлами бралась пропорциональной расстоянию между этими узлами.

Применение метода выбора конфигурации РПУ ПКС позволяет получать точно решение размещение контроллеров и распределения управления коммутаторами между контроллерами. Пример использования метода приведен на рисунке и в таблице.

Для сети Integra при минимизации метрики средней задержки L_{avg} от коммутатора до контроллера достаточно трех контроллеров и они размещаются в узлах 3,6 и 25. При минимизации метрики L_{wc} контроллеры размещаются в узлах 0, 8, 20.

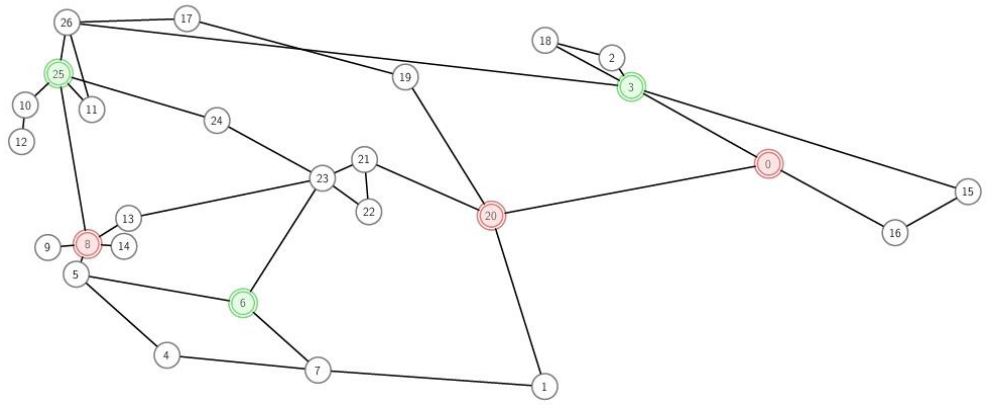

Рис. 3. Размещение контроллеров в сети Integra

Fig. 3. Placing controllers in the Integra network

Таблица 1. Основной и резервный контроллер для коммутаторов в сети Integra

Table 1. The main and spare controller for commutators in the Integra network

ID коммутатора	L_{avg}		L_{wc}	
	основной контроллер	резервный контроллер	основной контроллер	резервный контроллер
0	3	6	0	20
1	6	3	20	0
2	3	25	0	20
3	3	25	0	20
4	6	25	8	20
5	25	6	8	20
6	6	25	8	0
7	6	25	8	20
8	25	6	8	20
9	25	6	8	20
10	25	6	8	20
11	25	6	8	20
12	25	6	8	20
13	25	6	8	20
14	25	6	8	20
15	3	6	0	20
16	3	6	0	20
17	25	3	8	20
18	3	25	0	20
19	25	6	20	0
20	6	25	20	0
21	6	25	20	0
22	6	25	20	0
23	6	25	20	0
24	25	6	20	0
25	25	6	8	20
26	25	6	8	20

Для каждой сети, для которой было получено решение, вычислялось ухудшение выбранной метрики минимизации задержки при отказе каждого из контроллеров. Далее для различных сетей и для разных метрик сравнивались минимальное, максимальное и среднее ухудшение задержки.

В таблицах 2 и 3 приведены результаты оценки влияния отказа контроллеров на задержку в сети. Для всех сетей библиотеки Internet Topology Zoo приведены данные об ухудшении метрик L_{avg} и L_{wc} при отказе одного контроллера. В среднем отказ одного контроллера оказывает большее влияние на среднюю задержку в сети, чем на максимальную. Для 15 из 33 сетей отказ хотя бы одного из контроллеров вообще не приводит к увеличению максимальной задержки в сети.

Таблица 2. Увеличение средней задержки в сети (L_{avg}) при отказе одного контроллера

Table 2. Increase in the average network delay (L_{avg}) when one controller fails

Сеть	|S|	|C|	min, %	max, %	среднее, %
HiberniaCanada	10	2	60,08	61,82	60,95
Abilene	11	2	117,02	218,67	167,85
Compuserve	11	2	43,97	188,98	116,48
Navigata	13	2	103,28	193,78	148,53
Nsfnet	13	2	65,53	189,65	127,59
Claranet	15	2	77,48	186,82	132,15
Garr199901	16	2	88,72	107,90	98,31
Peer1	16	2	45,95	120,17	83,06
Ernet	16	2	764,00	1427,25	1095,63
Goodnet	17	2	102,56	212,38	157,47
Arpanet19719	18	2	4,48	40,82	22,65
Ibm	18	2	19,38	20,14	19,76
Internetmci	19	2	31,52	51,45	41,49
GtsRomania	19	2	69,68	577,23	323,45
Quest	20	2	147,99	526,47	337,23
BtEurope	22	3	11,50	34,46	20,54
York	23	3	45,11	98,10	78,85
Funet	24	3	147,71	233,42	187,62
Psinet	24	3	1,89	21,11	11,45
Agis	25	3	12,79	31,26	24,58
Integra	27	3	45,17	91,87	69,95
Biznet	28	3	102,10	745,80	519,16
Darkstrand	28	3	22,05	39,01	28,96
Digex	31	3	27,97	84,29	59,05
Bics	33	3	22,06	73,46	39,63
BtNorthAmerica	33	3	16,38	57,86	36,16
Grnet	34	3	0,18	14,59	5,68
NetworkUsa	35	3	6,65	24,51	13,67
Geant2012	37	3	3,73	7,77	6,13
Renater2010	37	3	25,95	191,51	127,11
Cesnet200706	38	3	36,10	126,26	66,29
Chinanet	38	3	16,67	21,41	19,01
Garr200912	42	4	4,73	14,76	10,53
Garr201101	44	4	8,00	41,67	20,16

Таблица 3. Увеличение максимальной задержки в сети (L_{wc}) при отказе одного контроллера

Table 3. Increase in the maximum network delay (L_{wc}) when one controller fails

Сеть	|S|	|C|	min, %	max, %	среднее, %
HiberniaCanada	10	2	0	30,78	15,39
Abilene	11	2	156,70	204,02	180,36
Compuserve	11	2	86,38	105,66	96,02
Navigata	13	2	132,4	153,8	143,1
Nsfnet	13	2	70,9	105,76	88,33
Claranet	15	2	83,49	113,98	98,74
Garr199901	16	2	71,99	87,33	79,66
Peer1	16	2	10,72	31,37	21,04
Ernet	16	2	335,68	435,68	385,68
Goodnet	17	2	116,16	160,11	138,13
Arpanet19719	18	2	0	0,04	0,02
Ibm	18	2	0	33,474	16,74
Internetmci	19	2	0	33,58	16,792
GtsRomania	19	2	93,57	97,796	95,68
Quest	20	2	304,38	318,8	311,59
BtEurope	22	3	0	15,22	5,08
York	23	3	97,83	151,43	115,7
Funet	24	3	364,52	393,05	376,67
Psinet	24	3	0	10,42	4,67
Agis	25	3	0	9,72	3,24
Integra	27	3	109,99	121,23	114,77
Biznet	28	3	537,52	565,87	550,04
Darkstrand	28	3	0	21,55	11,96
Digex	31	3	15,38	115,38	76,07
BtNorthAmerica	33	3	51,37	151,37	85,31
Bics	33	3	0	57,21	20,97
Grnet	34	3	0	5,75	1,92
NetworkUsa	35	3	0	0,24	0,08
Geant2012	37	3	0	2,28	0,76
Cesnet200706	38	3	4,93	99,03	67,33
Uunet	42	4	0	19,46	8
Garr201101	44	4	0	6,1	1,53
Surfnet	50	4	0	55,9	27,73

Значительное ухудшение метрик для некоторых сетей обусловлено особенностями используемого метода решения задачи размещения контроллеров. Предложенный метод, согласно постановке задачи, ориентирован, в первую очередь, на уменьшение задержек в сети в условиях нормальной работы сети, т. е. когда все контроллеры исправны.

Для сетей, для которых были получены решения с учетом обоих критериев минимизации задержки, исследовалось, как часто совпадают оптимальные решения для разных критериев минимизации задержки (Таблица 4). Для сетей, имеющих отличающиеся решения для разных метрик, исследовалось влияние выбора одной метрики в качестве критерия минимизации задержки на значения другой метрики минимизации задержки (т. е. насколько оптимизация с учетом метрики приводит к ухудшению метрики , и наоборот).

Таблица 4. Зависимость выбора размещения контроллеров от выбранного критерия минимизации задержки: идентификаторы узлов сети, в которых расположены контроллеры

Table 4. Dependence of the choice of the controllers' location from the chosen criterion for minimizing the delay: the identifiers of the network nodes where the controllers are located

Сеть	|S|	|C|	L_{avg}	L_{wc}
Gray HiberniaCanada	10	2	4,7	4,7
Gray Abilene	11	2	4,9	4,9
Compuserve	11	2	1,9	0,10
Navigata	13	2	4,8	4,5
Gray Nsfnet	13	2	6,11	6,11
Claranet	15	2	3,12	1,12
Garr199901	16	2	4,7	7,10
Peer1	16	2	3,9	3,10
Gray Ernet	16	2	7,8	7,8
Gray Goodnet	17	2	9,15	9,15
Arpanet19719	18	2	0,7	7,8
Ibm	18	2	2,5	4,5
Gray Internetmci	19	2	2,3	2,3
Gray GtsRomania	19	2	5,9	5,9
Quest	20	2	2,4	2,6
BtEurope	22	3	5,15,19	7,15,19
York	23	3	5,21,22	5,17,22
Gray Funet	24	3	12,14,23	12,14,23
Gray Psinet	24	3	1,2,3	1,2,3
Agis	25	3	3,6,19	3,19,22
Integra	27	3	3,6,25	0,8,20
Biznet	28	3	19,22,24	19,22,23
Gray Darkstrand	28	3	1,6,16	1,6,16
Digex	31	3	2,16,25	22,23,27
Bics	33	3	5,14,15	0,2,22
BtNorthAmerica	33	3	1,13,23	1,12,13
Gray Grnet	34	3	0,2,17	0,2,17
NetworkUsa	35	3	5,6,7	30,31,32
Geant2012	37	3	2,4,6	6,4,27
Cesnet200706	38	3	0,5,34	3,7,34
Garr201101	44	4	0,1,7,23	0,1,7,42

Данные приведены только для тех сетей, для которых были получены решения с учетом обоих критериев. Для 11 из 31 сетей места размещения контроллеров с учетом разных критериев минимизации задержки совпали. Для остальных сетей решение различается в зависимости от выбранной метрики, тем не менее, для большинства сетей хотя бы одно место размещения контроллера совпадает для обеих метрик.

Таким образом, можно сделать вывод, что если нет четкого требования минимизации максимальной задержки в сети, то для выбора оптимального размещения контроллеров предпочтительнее использовать минимизацию средней задержки в качестве критерия оптимизации. В этом случае полученное для оптимальное решение для многих сетей совпадет с оптимальным для или будет близким к нему.

По результатам проведенных экспериментов можно сделать следующие выводы:

- Количество итераций аддитивного алгоритма решения задачи булевского линейного программирования зависит от топологии сети и, в меньшей степени, от ее размера. Данный алгоритм может применяться для сетей среднего размера (до 50 узлов). Для сетей, содержащих большое число узлов, а также в тех случаях, когда при помощи аддитивного алгоритма решение не может быть получено за приемлемое время, предпочтительнее использовать приближенные методы для решения поставленной оптимизационной задачи.
- Предложенный метод решения задачи ориентирован, в первую очередь, на минимизацию задержек при условии исправной работы всех контроллеров. Поэтому для некоторых сетей отказ контроллера может приводить к значительному увеличению задержек в сети.
- Для поиска оптимального размещения контроллеров в большинстве случаев можно использовать критерий минимизации средней задержки в сети. Для многих сетей полученное решение будет близким к оптимальному и с точки зрения минимизации максимальной задержки в сети.

Заключение

В данной статье приводится метод, позволяющий выбрать, как организовать управление с помощью контроллеров РПУ ПКС для заданной транспортной сети с учетом требований отказоустойчивости и минимизации средней задержки между коммутаторами и контроллерами. Метод позволяет определить оптимальное количество контроллеров РПУ для данной сети, оптимальные места размещения контроллеров с учетом задержки и определить основной и резервный контроллеры для каждого коммутатора сети. Размещение контроллеров производится с помощью разбиения сети на области двусвязности и применением методов кластеризации k-means или k-medians для каждой области двусвязности, а определение для каждого коммутатора резервного контроллера осуществляется с помощью метода Балаша.

Список использованных источников

[1] *Смелянский Р.Л.* Программно-конфигурируемые сети // Открытые системы. СУБД. 2012. № 9. С. 15-26. URL: https://www.osp.ru/os/2012/09/13032491 (дата обращения: 12.04.2018).

[2] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. ONF White Paper. April 13, 2012. 12 p. URL: http://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (дата обращения: 12.04.2018).

[3] *Gude N., Koponen T., Pettit J., Pfaff B., Casado M., McKeown N., Shenker S.* NOX: towards an operating system for networks // ACM SIGCOMM Computer Communication Review. 2008.

Vol. 38, issue 3. Pp. 105-110. DOI: 10.1145/1384609.1384625

[4] *Erickson D.* The beacon openflow controller // Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13). ACM, New York, NY, USA, 2013. Pp. 13-18. DOI: 10.1145/2491185.2491189

[5] Floodlight OpenFlow Controller [Электронный ресурс]. URL: http://www.projectfloodlight.org (дата обращения: 12.04.2018).

[6] *Shalimov A.* et al. The Runos OpenFlow Controller // Proceedings of 2015 Fourth European Workshop on Software Defined Networks (9 30 Sept.-2 Oct. 2015). Bilbao, Spain, 2015. Pp. 103-104. DOI: 10.1109/EWSDN.2015.69

[7] *Shalimov A., Zuikov D., Zimarina D., Pashkov V., Smeliansky R.* Advanced study of SDN/OpenFlow controllers // Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13). ACM, New York, NY, USA, 2013. Article 1, 6 pages. DOI: 10.1145/2556610.2556621

[8] *Tootoocian A., Ganjali Y.* HyperFlow: A distribute control plane for OpenFlow // Proceedings of the 2010 INM conference / WREN workshop. 2010. Pp. 1 - 6. URL: https://pdfs.semanticscholar.org/f7bd/dc08b9d9e2993b363972b89e08e67dd8518b.pdf (дата обращения: 12.04.2018).

[9] *Koponen T., Casado M., Gude N., Stribling J.,* et al., Onix: A distributed control platform for large-scale production networks // Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10). USENIX, 2010. Vol. 10. URL: https://www.usenix.org/event/osdi10/tech/full_papers/Koponen.pdf (дата обращения: 12.04.2018).

[10] *Dixit A., Hao F., Mukherjee S., Lakshman T.V., Kompella R.* Towards an elastic distributed SDN controller // Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13). ACM, New York, NY, USA, 2013. Pp. 7-12. DOI: 10.1145/2491185.2491193

[11] *Yeganeh S.H., Ganjali Y.* Beehive: Simple Distributed Programming in Software-Defined Networks // Proceedings of the Symposium on SDN Research (SOSR '16). ACM, New York, NY, USA, 2016. Article 4, 12 pages. DOI: 10.1145/2890955.2890958

[12] *Berde P., Gerola M., Hart J., Higuchi Y., Kobayashi M., Koide T., Lantz B., O'Connor B., Radoslavov P., Snow W., Parulkar G.* ONOS: towards an open, distributed SDN OS // Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14). ACM, New York, NY, USA, 2016. Pp. 1-6. DOI: 10.1145/2620728.2620744

[13] *Yeganeh S.H., Kandoo Y.G.* A Framework for Efficient and Scalable Offloading of Control Applications // Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN12). ACM, New York, NY, USA, 2012. Pp. 19–24. URL: http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p19.pdf (дата обращения: 12.04.2018).

[14] *Yap K.-K., Motiwala M., Rahe J., Padgett S., Holliman M., Baldus G., Hines M., Kim T., Narayanan A., Jain A., Lin V., Rice C., Rogan B., Singh A., Tanaka B., Verma M., Sood P., Tariq M., Tierney M., Trumic D., V. Valancius, Ying C., Kallahalla M., Koley B., Vahdat A.* Taking the edge off with espresso: Scale, reliability and programmability for global internet peering // Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17). ACM, New York, NY, USA, 2017. Pp. 432–445. DOI: 10.1145/3098822.3098854

[15] Open Networking Foundation TR-521, SDN Architecture 1.1 (Technical Reference), non-normative, type 2, issue 1.1, 2016. 59 p. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf (дата обращения: 12.04.2018).

[16] Open Networking Foundation. OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06). March 26, 2015. 283 p. URL: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (дата обращения: 12.04.2018).

[17] *Pashkov V., Shalimov A., Smeliansky R.* Controller failover for SDN enterprise networks // Proceedings of 2014 IEEE International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC) (28-29 Oct. 2014). Moscow, Russia, 2014. Pp. 1-6. DOI: 10.1109/MoNeTeC.2014.6995594

[18] *Пашков В.Н.* Разработка высокодоступной платформы управления для программно-конфигурируемых сетей // Материалы 19-ой международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС'2015), 24-31 мая 2015 г. Алушта: Изд-во МАИ М, 2015. С. 169–171. URL: https://istina.msu.ru/download/45107482/1fn02v:vzOM-U0i3gP-FH9mZ6kE9dpfHIfg/ (дата обращения: 12.04.2018).

[19] *Вишневский В.М.* Теоретические основы проектирования компьютерных сетей. Москва: Техносфера, 2003. 512 с.

[20] *Heller B., Sherwood R., McKeown N.* The controller placement problem // Proceedings of the first workshop on Hot topics in software defined networks. (HotSDN '12). ACM, New York, NY, USA, 2012. Pp. 7-12. DOI: 10.1145/2342441.2342444

[21] *Ying Zh., Neda B., Mallik T.* On Resilience of Split-Architecture Networks // Proceedings of 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011 (5-9 Dec. 2011). Kathmandu, Nepal, 2011. Pp. 1-6. DOI: 10.1109/GLOCOM.2011.6134496

[22] *Neda B., Ying Zh.* Fast failover for control traffic in Software-defined Networks // Proceedings of 2012 IEEE Global Communications Conference (GLOBECOM) (3-7 Dec. 2012). Anaheim, CA, USA, 2012. Pp. 2665-2670. DOI: 10.1109/GLOCOM.2012.6503519

[23] *Ros F.J., Ruiz P.M.* On Reliable Controller Placements in Software-Defined Networks // Computer Communications. 2016. Vol. 77. Pp. 41-51. DOI: 10.1016/j.comcom.2015.09.008

[24] *Ros F.J., Ruiz P.M.* Five Nines of Southbound Reliability in Software-Defined Networks // Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14). ACM, New York, NY, USA, 2014. Pp. 31-36. DOI: 10.1145/2620728.2620752

[25] *Hock D., Hartmann M., Gebert S., Jarschel M., Zinner T., Tran-Gia P.* Pareto-optimal resilient controller placement in SDN-based core networks // Proceedings of the 2013 25th International Teletraffic Congress (ITC) (10-12 Sept. 2013). Shanghai, China, 2013. Pp. 1-9. DOI: 10.1109/ITC.2013.6662939

[26] *Knight S., Nguyen H.X., Falkner N., Bowden R., Roughan M.* The Internet Topology Zoo // IEEE Journal on Selected Areas in Communications. 2011. Vol. 29, no. 9. Pp. 1765-1775. DOI: 10.1109/JSAC.2011.111002

Поступила 12.04.2018; принята в печать 10.06.2018;
опубликована онлайн 30.06.2018.

References

[1] Smelyanskiy R.L. Software defined networks. *Open Systems. DBMS* 2012; 9:15-26. Available at: https://www.osp.ru/os/2012/09/13032491 (accessed 12.04.2018). (In Russian)

[2] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. ONF White Paper. April 13, 2012. 12 p. Available at: http://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (accessed 12.04.2018).

[3] Gude N., Koponen T., Pettit J., Pfaff B., Casado M., McKeown N., Shenker S. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*. 2008; 38(3):105-110. DOI: 10.1145/1384609.1384625

[4] Erickson D. The beacon openflow controller. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13)*. ACM, New York, NY, USA, 2013. pp. 13-18. DOI: 10.1145/2491185.2491189

[5] Floodlight OpenFlow Controller. Available at: http://www.projectfloodlight.org (accessed 12.04.2018).

[6] Shalimov A. et al. The Runos OpenFlow Controller. *Proceedings of 2015 Fourth European Workshop on Software Defined Networks* (9 30 Sept.-2 Oct. 2015). Bilbao, Spain, 2015. pp. 103-104. DOI: 10.1109/EWSDN.2015.69

[7] Shalimov A., Zuikov D., Zimarina D., Pashkov V., Smeliansky R. Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13)*. ACM, New York, NY, USA, 2013. Article 1, 6 pages. DOI: 10.1145/2556610.2556621

[8] Tootoocian A., Ganjali Y. HyperFlow: A distribute control plane for OpenFlow. *Proceedings of the 2010 INM conference / WREN workshop. 2010*. Pp. 1 - 6. Available at: https://pdfs.semanticscholar.org/f7bd/dc08b9d9e2993b363972b89e08e67dd8518b.pdf (accessed 12.04.2018).

[9] Koponen T., Casado M., Gude N., Stribling J., et al., Onix: A distributed control platform for large-scale production networks. *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10)*. USENIX, 2010. Vol. 10. Available at: https://www.usenix.org/event/osdi10/tech/full_papers/Koponen.pdf (accessed 12.04.2018).

[10] Dixit A., Hao F., Mukherjee S., Lakshman T.V., Kompella R. Towards an elastic distributed SDN controller. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13)*. ACM, New York, NY, USA, 2013. pp. 7-12. DOI: 10.1145/2491185.2491193

[11] Yeganeh S.H., Ganjali Y. Beehive: Simple Distributed Programming in Software-Defined Networks. *Proceedings of the Symposium on SDN Research (SOSR '16)*. ACM, New York, NY, USA, 2016. Article 4, 12 pages. DOI: 10.1145/2890955.2890958

[12] Berde P., Gerola M., Hart J., Higuchi Y., Kobayashi M., Koide T., Lantz B., O'Connor B., Radoslavov P., Snow W., Parulkar G. ONOS: towards an open, distributed SDN OS. *Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14)*. ACM, New York, NY, USA, 2016. pp. 1-6. DOI: 10.1145/2620728.2620744

[13] Yeganeh S.H., Kandoo Y.G. A Framework for Efficient and Scalable Offloading of Control Applications. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN12)*. ACM, New York, NY, USA, 2012. pp. 19–24. Available at: http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p19.pdf (accessed 12.04.2018).

[14] Yap K.-K., Motiwala M., Rahe J., Padgett S., Holliman M., Baldus G., Hines M., Kim T., Narayanan A., Jain A., Lin V., Rice C., Rogan B., Singh A., Tanaka B., Verma M., Sood P., Tariq M., Tierney M., Trumic D., V. Valancius, Ying C., Kallahalla M., Koley B., Vahdat A. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 2017. pp. 432–445. DOI: 10.1145/3098822.3098854

[15] Open Networking Foundation TR-521, SDN Architecture 1.1 (Technical Reference), non-normative, type 2, issue 1.1, 2016. 59 p. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf (accessed 12.04.2018).

[16] Open Networking Foundation. OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06). March 26, 2015. 283 p. Available at: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed 12.04.2018).

[17] Pashkov V., Shalimov A., Smeliansky R. Controller failover for SDN enterprise networks. *Proceedings of 2014 IEEE International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)* (28-29 Oct. 2014). Moscow, Russia, 2014. pp. 1-6. DOI: 10.1109/MoNeTeC.2014.6995594

[18] Pashkov V. N. Development of a highly available control platform for software-defined networks. *Proceedings of XIX International Conference on Computational mechanics and modern applied software systems (CMMASS'2015)* (24-31 May, 2015). Alushta: Izd-vo MAI M, 2015. pp. 169–171. Available at: https://istina.msu.ru/download/45107482/1fn02v:v-zOM-U0i3gPFH9mZ6kE9dpfHIfg/ (accessed 12.04.2018). (In Russian)

[19] Vishnevskiy V. Theoretical foundations of computer network design. Moscow: The Technosphere, 2003. 512 p. (In Russian)

[20] Heller B., Sherwood R., McKeown N. The controller placement problem. *Proceedings of the first workshop on Hot topics in software defined networks. (HotSDN '12)*. ACM, New York, NY, USA, 2012. p. 7-12. DOI: 10.1145/2342441.2342444

[21] Ying Zh., Neda B., Mallik T. On Resilience of Split-Architecture Networks. *Proceedings of 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011* (5-9 Dec. 2011). Kathmandu, Nepal, 2011. pp. 1-6. DOI: 10.1109/GLOCOM.2011.6134496

[22] Neda B., Ying Zh. Fast failover for control traffic in Software-defined Networks. *Proceedings of 2012 IEEE Global Communications Conference (GLOBECOM)* (3-7 Dec. 2012). Anaheim, CA, USA, 2012. pp. 2665-2670. DOI: 10.1109/GLOCOM.2012.6503519

[23] Ros F.J., Ruiz P.M. On Reliable Controller Placements in Software-Defined Networks. *Computer Communications*. 2016; 77:41-51. DOI: 10.1016/j.comcom.2015.09.008

[24] Ros F.J., Ruiz P.M. Five Nines of Southbound Reliability in Software-Defined Networks. *Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14)*. ACM, New York, NY, USA, 2014. pp. 31-36. DOI: 10.1145/2620728.2620752

[25] Hock D., Hartmann M., Gebert S., Jarschel M., Zinner T., Tran-Gia P. Pareto-optimal resilient controller placement in SDN-based core networks. *Proceedings of the 2013 25th Interna-*

tional Teletraffic Congress (ITC) (10-12 Sept. 2013). Shanghai, China, 2013. pp. 1-9. DOI: 10.1109/ITC.2013.6662939

[26] Knight S., Nguyen H.X., Falkner N., Bowden R., Roughan M. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*. 2011; 29(9):1765-1775. DOI: 10.1109/JSAC.2011.111002

Submitted 12.04.2018; revised 10.06.2018;
published online 30.06.2018.

About the author:

Vasily N. Pashkov, Software developer, Computer Systems Laboratory, Department of Computing Systems and Automation, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia); ORCID: http://orcid.org/0000-0001-5783-4557, pashkov@lvk.cs.msu.su