

РАЗРАБОТКА АЛГОРИТМА МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ СВЯЗИ

DOI: 10.36724/2072-8735-2021-15-9-17-23

Волков Алексей Станиславович,
Национальный исследовательский университет
"МИЭТ", Москва, Зеленоград, Россия,
leshvol@mail.ru

Баскаков Александр Евгеньевич,
Национальный исследовательский университет
"МИЭТ", Москва, Зеленоград, Россия,
9999924816@ya.ru

Manuscript received 12 April 2021;
Accepted 27 May 2021

Ключевые слова: программно-конфигурируемые сети, алгоритм маршрутизации, многопутевая маршрутизация

В статье приведена разработка алгоритма маршрутизации в программно-конфигурируемых сетях связи с использованием принципа многопутевой доставки сообщений. Использование протокола OpenFlow в качестве основного для связи устройств плоскости данных и плоскости управления, то есть программно-конфигурируемых коммутаторов с контроллером, позволяет принять в качестве исходных данных для алгоритма топологию сети, представленную в виде неориентированного взвешенного графа. Существуют известные решения задачи поиска путей передачи данных в сети связи, как правило, с использованием протокола резервирования сетевых ресурсов, но, на сеть накладываются дополнительные ограничения, поскольку RSVP имеет низкую степень масштабируемости, соответственно, нецелевой расход вычислительных ресурсов и ресурсов системы хранения отдельных маршрутизаторов. С учетом вышесказанного, разработан алгоритм поиска множества путей на графике с построением вспомогательного графа на основе исходного. Приведены условия, при выполнении которых может быть построен вспомогательный график от исходного. В алгоритме учтена возможность построения нескольких путей, проходящих через одну вершину, при выполнении требований задержки входного потока данных. Для расширения функционала и возможных областей применения алгоритма поиска множества путей введен критерий необходимой суммарной пропускной способности множеством путей передачи данных.. Приведены условия построения путей от вершины к множеству вершин. Представленный в работе алгоритм имеет на порядок меньшую временную сложность, что позволяет оперативно реагировать на изменения в сети передачи данных, при этом наиболее значимые различия в затрачиваемом на построение множества путей времени заметны при увеличении узлов в сети передачи данных и количестве возможных путей.

Информация об авторах:

Волков Алексей Станиславович, к.т.н., доцент кафедры "Телекоммуникационные системы" федерального государственного автономного образовательного учреждения высшего образования "Национальный исследовательский университет "Московский институт электронной техники", Москва, Зеленоград, Россия

Баскаков Александр Евгеньевич, преподаватель кафедры "Телекоммуникационные системы" федерального государственного автономного образовательного учреждения высшего образования "Национальный исследовательский университет "Московский институт электронной техники", Москва, Зеленоград, Россия

Для цитирования:

Волков А.С., Баскаков А.Е. Разработка алгоритма многопутевой маршрутизации в программно-конфигурируемых сетях связи // T-Comm: Телекоммуникации и транспорт. 2021. Том 15. №9. С. 17-23.

For citation:

Volkov A.S., Baskakov A.E. (2021) Development of a multipath routing algorithm in software-defined communication networks. *T-Comm*, vol. 15, no.9, pp. 17-23. (in Russian)

Введение

В настоящее время существуют предпосылки к экспоненциальному росту передаваемого сетевого трафика в глобальной сети Интернет, как следствие, рост объемов передаваемых данных в магистральных и транспортных сетях связи [1]. Для обеспечения потребностей пользователей глобальных сетей связи существуют механизмы обеспечения качества обслуживания, но при росте интенсивности входного потока на сетевых устройствах могут возникать очереди, вносящие дополнительные задержки при передаче данных, что может быть критично в ряде случаев.

Для решения описанных проблем используются технологии и методы программно-определяемых сетей связи, позволяющие снизить нагрузку на сетевые устройства путем переноса управляющих функций в отдельное устройство – ПКС контроллер. В такой сети применимы методы динамического отслеживания состояния отдельных сетевых устройств и подконтрольной сети связи в целом с использованием централизованного управления, что позволяет проводить пересчет маршрутов с последующей отправкой актуальной схемы маршрутизации непосредственно на устройства, участвующие в передаче пользовательских данных, фактически без использования их вычислительных мощностей.

Вышеописанное решение является актуальным, но использует стандартные алгоритмы поиска оптимальных путей передачи данных, как правило, без применения многопутевой маршрутизации. Таким образом, актуальной задачей в области транспортных программно-конфигурируемых сетей является организация многопутевой маршрутизации, а именно, построения путей передачи данных на центральном устройстве управления.

Протокол OpenFlow

Для обеспечения функционала централизованного управления сетью связи в ПКС используются специализированные протоколы, обеспечивающие передачу служебной информации от ПКС-коммутаторов к центральному устройству – ПКС-контроллеру. Наиболее популярным решением является использование протокола OpenFlow [2-3].

При использовании ПКС, коммутаторы выполняют функцию перенаправления трафика по предварительно установленным правилам, основанным на имеющихся таблицах потоков и групп. При этом сами коммутаторы не участвуют в составлении этих таблиц, а получают их от ПКС контроллера. Для поддержания актуальной информации на центральном устройстве, равно как и добавлении нового устройства в обслуживаемую сеть, используются сообщения установленных форматов, далее рассмотрим некоторые из них.

Сообщения «Features» необходимо для запроса состояния ПКС коммутатора с центрального устройства. Определены два формата сообщения для запроса и ответа, соответственно сообщений контроллер-коммутатор и обратного. На основе ответа, а именно поля «capabilities» (возможности) ПКС коммутатора, контроллер не только строит карту сети (поля PORT_STATS, STP), но и может провести оценку загруженности определенного устройства по полям PORT_STATS и QUEUE_STATS [4-5].

После установления соединения с ПКС коммутатором и получения его доступного функционала, контроллер может инициировать изменение конфигурации отдельных параметров устройства, например статуса определенного порта, ограничение очереди ПКС коммутатора, добавление или удаление правил передачи данных, изменение таблицы потоков и т.д. Для возможности использования вышеописанного функционала применяются сообщения типа Configuration и Modify-State (конфигурация и изменение состояния), причем этими типами описано подмножество сообщений OpenFlow, как от контроллера к коммутатору, как правило, сообщения формата запрос или Request, так и от коммутатора к контроллеру – Response или ответ.

Отдельно стоит выделить группу сообщений Read-State, позволяющую запрашивать статистические данные с ПКС коммутатора [6-7]. Так, используя сообщения установленного формата можно получить информацию о текущих передаваемых потоках, а именно: продолжительность активного соединения для передачи потока данных в секундах или наносекундах, для более точных вычислений, идентификационные данные обслуживаемых потоков данных, приоритет обслуживаемых потоков, время простоя коммутатора, то есть время работы без наличия активных потоков, общее количество переданных сетевых пакетов, общее количество переданных байт информации, таблицу соответствия полей и действий ПКС коммутатора и его уникальный идентификатор. Вышеописанные параметры могут быть получены в рамках запроса информации об обслуживаемых потоках, но, кроме этого, могут быть также получены данные о группах, интерфейсах, пакетах, очередях или таблицах ПКС коммутатора, а также сводная таблица, включающая в себя все приведенные параметры.

При запросе статистических данных о группах, контроллером может быть получена информация об уникальном идентификаторе коммутатора, идентификаторе группы, в которую включено устройство, количеству обслуживаемых группой потоков данных, данных о переданных пакетах и количеству байт (данные в этом случае разделены на секции, соответствующие сконфигурированным на коммутаторе группам), количестве отложенных пакетов или байт.

Ответ на запрос статистических данных об интерфейсе может содержать информацию о номере порта ПКС коммутатора, количестве принятых и переданных пакетов или байт данных, количестве ошибок при приеме и передачи данных, количестве отброшенных пакетов при приеме и передаче, количестве ошибок приема сетевого кадра, несоответствии контрольной суммы и количестве коллизий, возникших в ходе передачи данных.

При получении первого сетевого пакета OpenFlow-коммутатор извлекает из него поля, отвечающие за заголовки канального, сетевого и транспортного уровней. На основе информации из извлеченных полей, а именно битовой строки, идентичной этим полям, происходит поиск записи в имеющихся таблицах потоков коммутатора, как правило, поиск начинается с таблицы под номером 0. Если соответствующая запись в таблице не найдена, то происходит обновление счетчиков и переход к следующей таблице, при ее наличии, где процедура поиска повторяется до момента, пока не произойдет нахождение соответствующего правила или не кончатся таблицы.

Если соответствие не найдено ни в одной таблице, то выполняется одно из трех действий на основе конфигурации коммутатора:

- отправка заголовка исходного сообщения на контроллер;
- отбрасывание сетевого пакета без последующей обработки;
- поиск соответствующей записи в групповой таблице.

Записи в групповой таблице называются групповыми правилами, они используются для обработки пакетов с групповой адресацией или в широковещательной рассылке. При этом запись в таблице потоков может описывать пересылку сетевых пакетов обслуживаемого потока на порт коммутатора с заданным уникальным идентификатором, как физический, так и виртуальный.

Механизм групповых таблиц предполагает сокращение используемых правил в таблицах, путем объединения их в группы с описанием общих действий для определенных потоков [8-9]. Например, при наличии активных правил пересылки потока на устройстве программно-конфигурируемого коммутатора от устройства с уникальным идентификатором 1 (далее К1), до устройства с идентификатором 3 через устройство К2 имеем правило перенаправления потока с входного порта К1 на порт, непосредственно связанный соединением с К2.

Предположим, что на вход К1 поступает второй поток данных с адресом назначения, находящимся за устройством К2, тогда, в таблицу потоков К1 следует добавить запись действия пересылки данных К1-К2 как для первого, так и для второго потоков данных. Это же действие можно выполнить, добавив оба правила в групповую таблицу, описывающую рассмотренные случаи одинаковым набором действий для К1. Объединение правил обработки потоков в групповые таблицы дает выигрыш в количестве таблиц потоков и общем количестве правил, а именно, в их сокращении. Поскольку этот этап может быть проделан после нахождения путей передачи данных, в работе не будут рассматриваться методы объединения правил обработки и путей передачи данных в группы.

Исходя из вышесказанного следует, что в работающей программно-конфигурируемой сети связи на основе протокола OpenFlow, с учетом уже установленных логических соединений коммутатор-контроллер для каждого из используемых устройств, ПКС контроллер имеет актуальную топологию обслуживаемой сети связи, с указанием пропускных способностей портов устройств, задержках передачи данных на каналах между коммутаторами и информацию о загруженности каждого устройства сети. Таким образом, возможно рассматривать исходную программно-конфигурируемую сеть как неориентированный взвешенный граф, для реализации алгоритма многопутевой маршрутизации.

Алгоритм поиска путей передачи данных.

Для описания алгоритма поиска множества путей передачи данных опишем используемую сетевую топологию в виде взвешенного неориентированного графа $G(V, E)$, где V – множество вершин графа, E – множество ребер между существующими вершинами. Поскольку граф является взвешен-

ным, то для каждого из ребер графа существуют параметры пропускной способности и задержки, описываемые как $b(e) \geq 0$ и $d(e) \geq 0$.

При этом для каждого пути $P = (s, 1, \dots, k)$ между двумя вершинами s и k графа G , при условии, что s и k принадлежат множеству вершин, также существуют параметры пропускной способности пути и его задержки, выраженные через $b(p) = \min b(e)$ и $d(p) = \sum d(e)$, где $b(e)$ – пропускная способность ребра графа G , $d(e)$ – значение задержки передачи данных на одном из ребер графа G .

Существующая проблема передачи данных с заданными параметрами пропускной способности и задержки, соответствующая требованиям архитектуры управления ресурсами IntServ, заключается в необходимости поиска пути передачи данных, удовлетворяющего входным требованиям потока данных.

Существуют известные решения указанной проблемы, как правило, с использованием протокола RSVP (протокол резервирования сетевых ресурсов), но ввиду использования последнего, на сеть накладываются дополнительные ограничения, поскольку RSVP не лишен недостатков. Основным является низкая степень масштабируемости протокола RSVP, что приводит к нецелевому расходу вычислительных ресурсов и ресурсов системы хранения отдельных маршрутизаторов, поскольку с ростом количества входящих потоков с заданными параметрами качества обслуживания (QoS), прямо пропорционально растет и количество информации, необходимое для обслуживания потоков данных. В таких случаях, как правило, задача сводится к поиску пути с удовлетворяющими входному потоку требованиями.

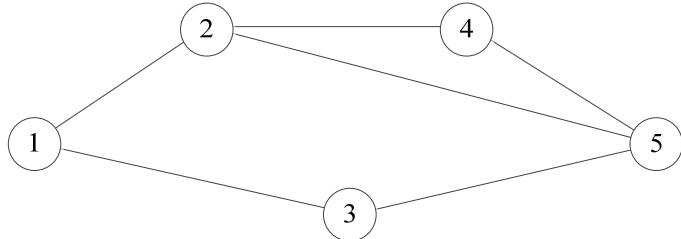


Рис. 1. Исходная топология сети

На рисунке 1 изображен граф $G(V, E)$, описывающий топологию сети передачи данных. Введем в исходный граф параметр задержки передачи данных для каждого из ребер исходного графа, тогда получим граф $G(V, E, d_e)$.

Тогда, для решения задачи поиска необходимого маршрута, удовлетворяющего входному потоку данных с требованием задержки $d(t)$ введем понятие вспомогательного графа $G_d(V_d, E_d, d_u)$, где d_u – значение задержки для искомого пути P [10]. Граф G_d может быть построен, если выполнены условия:

1. Для каждой вершины v , принадлежащей множеству V и не являющейся входной, добавляются d_u вершин к множеству $V_d : v^0, v^1, \dots, v^{du}$;
2. Для входной вершины s добавляется вершина s^0 в множество V_d ;
3. Для каждого ребра между вершинами $e(v, r)$, не включая входную, добавляется ребро в множество E_d ;

4. Для каждой вершины v множества V_d , не являющейся входной, не включать вершину v , если ее степень равно нулю.

Добавляемое в п.2 ребро имеет вид $(v^i, k^{i+d(v,r)})$, где $i=0,1,\dots,\partial_u - d(v,r)-1$, для ребра, первой вершиной которого является входная, добавляется вершина $(s^0, v^{d(s,v)})$.

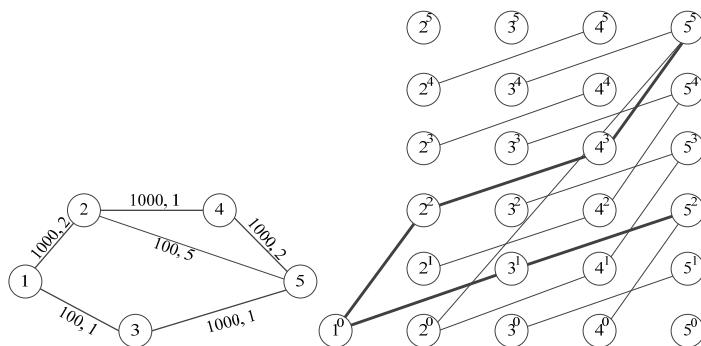


Рис. 2. Вспомогательный граф $G_o(V_o, E_o, d_u)$

На рисунке 2 изображен пример построения вспомогательного графа для решения задачи поиска множества путей передачи данных в исходном графе $G(V, E, d_e)$ для искомого значения задержки, равной 5мс.

На первом шаге выполняется добавление вершин вспомогательного графа в соответствии с полученным значением искомой задержки, соответственно, максимальная степень вершины, указывающая на искомый параметр, равна 5. Вышесказанное справедливо для всех вершин, исключая входную, ровно как описано в условиях построения вспомогательного графа.

На втором шаге происходит добавление ребер между вершинами, стоит отметить, что этот шаг при визуализации процесса отражен в копировании ребер между вершинами по горизонтали, например, ребро между вершинами 2^0 и 4^1 также будет скопировано и на вершины 2^1 и 4^2 .

На третьем шаге выполняется поиск удовлетворяющих первоначальным условиям маршрутов. Результатом работы алгоритма является построение двух путей передачи данных: P1 = (1, 3, 5) и P2 = (1, 2, 4, 5), с суммарной задержкой передачи данных равной 2мс и 5мс соответственно. При этом путь P3 = (1, 2, 5) отброшен как неудовлетворяющий условиям, поскольку имеет большее чем искомое значение суммарной задержки передачи данных, а именно, 7мс.

Основное отличие при таком подходе к построению множества путей заключено в итерационном отбрасывании путей, не отвечающих требованиям по задержке передачи данных. Также стоит отметить, что используемый подход предоставляет возможность поиска как непересекающихся путей передачи данных, так и имеющих общие вершины или ребра.

Поскольку в программно-конфигурируемых сетях существует концепция централизованного управления сетевыми устройствами, то ПКС контроллер, как центральное устройство сети связи, имеет актуальную информацию о загруженности каждого из устройств сети. Это позволяет наиболее точно находить удовлетворяющие входным требованиям

пути передачи данных, поскольку параметры исходного графа $G(V, E, d_e)$ могут быть изменены, в работе будем считать, что количество вершин и ребер графа остается неизменным, то есть все устройства и линии связи между ними функционируют в нормальном режиме, не происходит реконфигурирования топологии сети, а значение задержки для определенного ребра, то есть его вес, может быть изменено. Тогда, для поиска требуемых путей достаточно будет построить или изменить существующий вспомогательный граф с учетом изменившихся параметров, остальные пункты выполняются без изменений.

Недостатком используемого алгоритма является использование одного критерия выбора маршрута передачи данных – задержки, пропускная способность каналов связи между сетевыми устройствами не учитывается. Для решения указанной проблемы введем в используемую модель дополнительный параметр пропускной способности ребра между двумя вершинами, тогда, передаваемый поток данных может рассматриваться как набор потоков данных, сумма которых равна исходному входному потоку $\sum_i \lambda_i = \lambda$, где i – количество

подпотоков. Тогда, при условии наличия очередей на сетевых устройствах, задержка передачи одного подпотока равна $D_{pi} = d_{pi} + \frac{\lambda_i}{b_{pi}}$, где d_{pi} – задержка передачи для используемого пути, b_{pi} – используемая пропускная способность пути.

Обязательным условием ввода параметра пропускной способности является использование переменной b_{pi} , такой, что $b_{pi} \leq \min_{1 \dots i} (b_e)$. При этом задержка передачи данных может быть рассчитана как $\max_{1 \dots i} (D_{pi})$.

На основе вышесказанного, изменим вводимый вспомогательный граф, а именно, заменим переменную d_i на задержку передачи подпотока D_{pi} . Тогда получим вспомогательный граф $G_o(V_o, E_o, D_{pi})$. После введения описанной переменной имеем алгоритм поиска множества путей передачи данных с заданными параметрами пропускной способности и задержки, позволяющий вести поиск как непересекающихся, так и смежных путей передачи данных. Важным фактором является непрерывное обновление данных от ПКС-коммутаторов в реальной сети связи через протокол OpenFlow, рассмотренный ранее. При наличии трафика в сети связи, узел может быть задействован в передаче данных или быть свободным. В первом случае, объем данных, который может быть обслужен конкретным узлом, может быть уменьшен ввиду уже имеющихся данных на входе устройства.

Непрерывное обновление информации о загруженности сетевых устройств позволяет проводить объективную оценку приоритета маршрута передачи данных в зависимости от загруженности устройств, задействованных в этом маршруте. Ввод параметра остаточной пропускной способности устройства позволит точнее определять возможность использования конкретного пути передачи данных, для этого изменим требования к переменной, указывающей на используемую пропускную способность пути:

$$b_{pi} \leq \min_{1..i}(b_{e_{ocm}}) \leq \min_{1..i}(b_e),$$

где $b_{e_{ocm}}$ – остаточная (нездействованная) пропускная способность.

Результатом выполнения приведенных в работе алгоритмов является получение набора путей передачи данных $P = (P_1, \dots, P_n)$, удовлетворяющих требованиям входного потока данных. Для реализации многопутевой маршрутизации поверх найденных маршрутов необходима интеграция программного модуля, обеспечивающего возможность динамического управления подпотоками, сумма которых равна исходному входному потоку. Описанный принцип управления имеет существующие реализации и, как правило, является составной частью программно-алгоритмического комплекса балансировки трафика или управления ресурсами программно-конфигурируемой сети.

Из известных алгоритмов управления ресурсами ПКС стоит отметить GNMSP, Split, SDN-VN, CNMMCF, как наиболее гибкие решения, в полной мере поддерживающие необходимый для полноценной работы многопутевой маршрутизации функционал [11-13].

Основной принцип работы приведенных алгоритмов выражен в балансировке нагрузки на устройства программно-конфигурируемой сети связи с одновременной минимизацией нагрузки на виртуальную сеть или виртуальные сетевые функции (при их использовании).

Использование приведенного алгоритма также возможно для формирования множества путей из исходной вершины к множеству вершин. Поскольку при выполнении этапов алгоритма поиска путей от исходной вершины до конечной затрагивают и вершины между ними, то путь до промежуточной вершины v , с учетом требований пропускной способности и задержки передачи данных. То есть аналогичной вершины в графе $G_o(V_o, E_o, D_{pi})$, сводится к поиску кратчайших путей до вершины v^i в исходном дереве вспомогательного графа, построенного для задачи поиска маршрутов до вершины k^i , где $i \leq D_{pi}$.

Тогда, временная сложность представленного алгоритма составит $O(qS_mD\log(S_mD))$, где q – общее количество найденных путей, S_m – средняя степень вершин пути, D – длина максимального найденного пути, логарифмическая сложность объясняется итерационным выполнением операций над парой вершин.

Существующие алгоритмы поиска множества путей, описаны алгоритмами формирования виртуальных каналов связи поверх физических, такой подход позволяет повысить эффективность процедур управления трафиком для обеспечения заданных параметров качества обслуживания при изменении параметров сети.

В общем случае, временная сложность таких алгоритмов может быть описана как $O(qN^2)$, что объясняется формированием одного виртуального канала поверх q физических каналов с использованием алгоритма поиска путей передачи данных на основе алгоритма Дейкстры. Для полно связных топологий будет справедлива формула временной сложности алгоритма, описанная через $O(M^2N^2)$.

При этом, при учете формирования виртуального канала из исключительно множества непересекающихся путей, временная сложность составит $O(N^2 + \sum_2^n (N - \sum_1^{n-1} N_v)^2)$, где

N – общее количество узлов в рассматриваемой сети связи, n – количество существующих непересекающихся путей передачи данных, N_v – количество вершин существующего пути передачи данных.

Представленный в работе алгоритм имеет на порядок меньшую временную сложность, что позволяет наиболее оперативно реагировать на изменения в сети передачи данных, при этом наиболее значимые различия в затрачиваемом на построение множества путей времени заметны при увеличении узлов в сети передачи данных и количестве возможных путей.

Заключение

В работе приведен анализ принципа работы программно-конфигурируемой сети связи, а также результаты исследования, полученные при разработке алгоритма многопутевой маршрутизации в программно-конфигурируемых сетях.

Предложено решение задачи поиска множества путей передачи данных, удовлетворяющих требуемым параметрам входного потока данных, с использованием вспомогательного графа, полученного на основе исходного графа, описывающего топологию сети. Предложенное решение может быть применено в ПКС для поиска путей передачи данных с требуемой задержкой передачи меньшей, или равной, входному значению.

Предложено решение, позволяющее расширить функционал разработанного алгоритма поиска путей передачи данных путем введения параметра пропускной способности каналов связи, а далее – остаточной пропускной способности при наличии одного или нескольких активных потоков данных на установленном пути.

Проведена оценка временной сложности разработанного алгоритма поиска путей передачи данных, равная $O(qS_mD\log(S_mD))$, что на порядок меньше известных решений поиска множества путей передачи данных.

Предложено решение, позволяющее модифицировать разработанный алгоритм для поиска путей передачи данных от входной вершины до множества вершин.

Описанный в работе алгоритм может быть дополнен программным модулем балансировки трафика или управления ресурсами обслуживаемой программно-конфигурируемой сети связи для обеспечения полнофункционального протокола маршрутизации, а также, может быть модифицирован под условия работы специализированных реализаций ПКС, например, SDWSN (программно-конфигурируемые беспроводные сенсорные сети).

Литература

1. Garrich M. et al. Open-source network optimization software in the open SDN/NFV transport ecosystem // Journal of Lightwave Technology. 2018. Т. 37. №. 1. С. 75-88.
2. Alsaeedi M., Mohamad M. M., Al-Roubaiey A. A. Toward adaptive and scalable OpenFlow-SDN flow control: A survey // IEEE Access. 2019. Т. 7. С. 107346-107379.

3. Bholebawa I. Z., Jha R. K., Dalal U. D. Performance analysis of proposed OpenFlow-based network architecture using mininet // Wireless Personal Communications. 2016. T. 86. №. 2. C. 943-958.
4. Kim W. et al. OFMon: OpenFlow monitoring system in ONOS controllers // 2016 IEEE NetSoft Conference and Workshops (NetSoft). IEEE, 2016. C. 397-402.
5. Jero S. et al. Beads: automated attack discovery in openflow-based sdn systems // International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, Cham, 2017. C. 311-333.
6. Bakhshi T., Ghita B. OpenFlow-enabled user traffic profiling in campus software defined networks // 2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, 2016. C. 1-8.
7. Li Y. et al. SDN components and OpenFlow // Big Data and Software Defined Networks. IET, 2018. C. 49-67.
8. Qiao S. et al. Taming the flow table overflow in openflow switch //Proceedings of the 2016 ACM SIGCOMM Conference. 2016. C. 591-592.
9. Al-Najjar A., Layeghy S., Portmann M. Pushing SDN to the end-host, network load balancing using OpenFlow // 2016 IEEE international conference on pervasive computing and communication workshops (percom workshops). IEEE, 2016. C. 1-6.
10. Dong K. et al. Auxiliary Graph Based Routing, Wavelength and Time-slot Assignment in Metro Quantum Optical Networks // 2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC). IEEE, 2019. C. 1-3.
11. Khan M. A. et al. A Novel Technique of Dynamic Resource Allocation in Software Defined Network // 2019 15th International Conference on Emerging Technologies (ICET). IEEE, 2019. C. 1-5.
12. Guck J. W., Reisslein M., Kellerer W. Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks //IEEE Transactions on Industrial Informatics. 2016. T. 12. №. 6. C. 2050-2061.
13. Javadpour A., Wang G., Xing X. Managing heterogeneous substrate resources by mapping and visualization based on software-defined network // 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE, 2018. C. 316-321.

ЗАО "НПЦ ИРС" НИУ "МИЭТ" ООО "ИНТЕХ"



<http://intech-spb.com/conferences/>
konferencia_asu_vka@mail.ru

ВСЕРОССИЙСКАЯ МЕЖВЕДОМСТВЕННАЯ НАУЧНО-ТЕХНИЧЕСКАЯ КОНФЕРЕНЦИЯ

по теоретическим и прикладным проблемам
развития и совершенствования АСУ
и связи специального назначения

DEVELOPMENT OF A MULTIPATH ROUTING ALGORITHM IN SOFTWARE-DEFINED COMMUNICATION NETWORKS

Alexey S. Volkov, National Research University of Electronic Technology, Moscow, Zelenograd, Russia, leshvol@mail.ru

Aleksandr E. Baskakov, National Research University of Electronic Technology, Moscow, Zelenograd, Russia, 9999924816@ya.ru

Abstract

The paper describes the development of routing algorithm in software-defined communication networks using the principle of multipath message delivery. The use of the OpenFlow protocol as the main one for connecting data- and control-plane devices between each other, that is, programmable switches with the controller, allows us to take the network topology presented in undirected weighted graph form as the initial data for the algorithm. There are known solutions to the problem of finding ways to transmit data in a communication network, as a rule, using the network resource reservation protocol, but additional restrictions are imposed on the network, since RSVP has a low degree of scalability, respectively, inappropriate consumption of computing resources and storage system resources of individual routers. In view of the above, an algorithm has been developed for finding a set of paths on a graph with the construction of an auxiliary graph based on the original one. Conditions are given under which an auxiliary graph can be constructed from the initial one. The algorithm takes into account the possibility of constructing several paths passing through one vertex, while meeting the requirements for the delay of the input data stream. To expand the functionality and possible areas of application of the algorithm for finding a set of paths, a criterion for the required total throughput by a set of data transmission paths is introduced. Conditions for constructing paths from a vertex to set of vertices are given. The algorithm presented in the work has an order of magnitude less time complexity, which allows you to quickly respond to changes in the data transmission network, while the most significant differences in the time spent on building a set of paths are noticeable with an increase in nodes in the data transmission network and the number of possible paths.

Keywords: software-defined networks, routing algorithm, multipath routing.

References

1. M.Garrich, F.J. Moreno-Muro, M.V.B. Delgado, and P.P. Marino (2018). Open-source network optimization software in the open SDN/NFV transport ecosystem. *Journal of Lightwave Technology*, 37(1), pp. 75-88.
2. M. Alsaeedi, M.M. Mohamad, and A.A., Al-Roubaiey (2019). Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE Access*, 7, pp. 107346-107379.
3. I.Z. Bholebawa, R.K. Jha, and U.D. Dala (2016). Performance analysis of proposed OpenFlow-based network architecture using mininet. *Wireless Personal Communications*, 86(2), pp. 943-958.
4. W. Kim, J. Li, J.W.K. Hong and Y.J. Suh (2016), June. OFMon: OpenFlow monitoring system in ONOS controllers. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 397-402.
5. S. Jero, X. Bu, C. Nita-Rotaru, H. Okhravi, R. Skowyra and S. Fahmy (2017), September. Beads: automated attack discovery in openflow-based sdn systems. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, Cham, pp. 311-333.
6. T. Bakhshi and B. Ghita (2016), October. OpenFlow-enabled user traffic profiling in campus software defined networks. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1-8.
7. Y. Li, D. Zhang, J. Taheri and K. Li (2018). SDN components and OpenFlow. In *Big Data and Software Defined Networks*. IET, pp. 49-67.
8. S. Qiao, C. Hu, X. Guan, and J. Zou (2016), August. Taming the flow table overflow in openflow switch. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 591-592.
9. A. Al-Najjar, S. Layeghy, and M. Portmann (2016), March. Pushing SDN to the end-host, network load balancing using OpenFlow. In *2016 IEEE international conference on pervasive computing and communication workshops (percom workshops)*, pp. 1-6.
10. K. Dong, Y. Zhao, X. Yu, J. Zhang, H. Yu, and Z. Li (2019), July. Auxiliary Graph Based Routing, Wavelength and Time-slot Assignment in Metro Quantum Optical Networks. In *2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*, pp. 1-3.
11. M.A. Khan, M.A. Shah, F.Z. Raja and H.A. Khattak (2019), December. A Novel Technique of Dynamic Resource Allocation in Software Defined Network. In *2019 15th International Conference on Emerging Technologies (ICET)*, pp. 1-5.
12. J.W Guck, M. Reisslein, and W. Kellerer (2016). Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks. *IEEE Transactions on Industrial Informatics*, 12(6), pp. 2050-2061.
13. A. Javadpour, G. Wang, and X. Xing (2018), December. Managing heterogeneous substrate resources by mapping and visualization based on software-defined network. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 316-321.

Information about authors:

Alexey S. Volkov, assistant professor of the Department of Telecommunications, National Research University of Electronic Technology, Moscow, Zelenograd, Russia
Aleksandr E. Baskakov, graduate student of the Department of Telecommunications, National Research University of Electronic Technology, Moscow, Zelenograd, Russia