DOI: 10.15514/ISPRAS-2024-36(1)-2

Исследование вопросов учёта нагрузок в программно-конфигурируемых сетях

[1] *И. Б. Бурдонов,* ORCID: 0000-0001-9539-7853 <*igor@ispras.ru*>
[1,2] *Н. В. Евтушенко,* ORCID: 0000-0002-4006-1161 <*evtushenko@ispras.ru*>
[1] *А. С. Косачев,* ORCID: 0000-0001-5316-3813 <*kos@ispras.ru*>

[1] *Институт системного программирования РАН им. В.П. Иванникова, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20.*

Аннотация. Описывается исследование возможности реализации виртуальных сетей с учётом различных параметров и их корректировки в программно-конфигурируемых структурах, моделируемых взвешенным графом плоскости данных. В работе исследуются параметры двух типов: «ресурс» и «стоимость». Для параметра типа «ресурс» с ребром ассоциируется его «ёмкость», и число путей, проходящих через ребро, не должно превышать ёмкость ребра. Для параметра типа «стоимость» с ребром ассоциируется его «цена», «цена» пути есть сумма «цен» его рёбер, и ставится задача минимизации суммарной «цены» всех путей. Для реализации на взвешенном графе плоскости данных предложены алгоритм корректировки виртуальной сети с учётом параметров типа «ресурс» и два алгоритма построения виртуальной сети с учётом параметров типа «стоимость». В последнем случае один алгоритм строит для каждого хоста один путь из него в один хост из заданного подмножества целевых хостов; другой алгоритм строит для каждого хоста множество путей: по одному пути в один хост из каждого множества из семейства множеств целевых хостов.

Ключевые слова: распределённые и параллельные вычисления; программно-конфигурируемые сети; маршрутизация пакетов; взвешенный граф.

Для цитирования: Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Исследование вопросов учёта нагрузок в программно-конфигурируемых сетях. Труды ИСП РАН, том 36, вып. 1, 2024 г., стр. 23–34. DOI: 10.15514/ISPRAS–2024–36(1)–2.

Studying load issues in software-defined networks

[1] *I.B. Burdonov* ORCID: 0000-0001-9539-7853 <*igor@ispras.ru*>
[1,2] *N.V. Yevtushenko* ORCID: 0000-0002-4006-1161 <*evtushenko@ispras.ru*>
[1] *A.S. Kossatchev* ORCID: 0000-0001-5316-3813 <*kos@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *National Research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia.*

Abstract. The purpose of the work is to study the possibility of implementing virtual networks taking into account various parameters and their adjustments in software-configurable structures modeled by a weighted data plane graph. The work examines parameters of "resource" and "cost" types. For a resource type parameter, an edge is augmented with its "capacity," and the number of paths passing through the edge must not exceed the edge's capacity. For a parameter of the "cost" type, the path weight is the sum of the weights of the edges and the task of minimizing the weight of the path is set. For implementing a virtual network on a weighted graph, an algorithm for adjusting a virtual network taking into account parameters of the "resource" type and two algorithms for constructing a virtual network taking into account parameters of the "cost" type are proposed. In the latter case, one algorithm builds one path from each host to one host of the given subset of target hosts; another algorithm builds a set of paths for each host: one path to one host from each set of a family of sets of target hosts.

Keywords: distributed and parallel computations; software-defined networks (SDN); packet routing; weighted graph.

For citation: Burdonov I.B., Yevtushenko N.V., Kossatchev A.S. Studying load issues in the software-defined networks. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 1, 2024. pp. 23-34 (in Russian). DOI: 10.15514/ISPRAS-2024-36(1)-2.

1. Введение

В программно-конфигурируемых сетях (SDN) с разделёнными плоскостями данных и управления (см., например, [1-5]) пакеты между хостами пересылаются на плоскости данных через промежуточные коммутаторы, настройка которых осуществляется SDN-контроллерами. Настройка коммутаторов определяет множество путей, по которым между хостами пересылаются пакеты с заданными идентификаторами. Набор реализуемых путей моделирует виртуальную сеть на плоскости данных, которая в работах [4-6] рассматривается как невзвешенный граф связей. При исследовании вопросов надёжности, безопасности и оптимизации нагрузки на хосты и коммутаторы необходимо рассматривать дополнительные параметры сети, которые игнорируются в предыдущих работах авторов [4-7]. Для учёта параметров типа «ресурс» и «стоимость» можно использовать взвешенный граф связей для плоскости данных сети, тем самым налагаются ограничения на маршрутизацию пакетов.

Для параметра типа «ресурс» с ребром ассоциируется его «ёмкость», и число путей, реализующих виртуальную сеть и проходящих через ребро, не должно превышать ёмкости ребра. В разделе 3 предлагается алгоритм корректировки виртуальной сети с учётом параметров типа «ресурс» при реализации на взвешенном графе плоскости данных. Предлагаемый алгоритм достаточно просто расширяется на случай реализации нескольких виртуальных сетей.

Для параметра типа «стоимость» ребро имеет «цену», и «цена» пути есть сумма «цен» его рёбер. Ставится задача минимизации суммарной «цены» путей, реализующих виртуальную сеть. В разделе 4 предлагаются два алгоритма построения виртуальной сети с учётом параметров типа «стоимость» при реализации на взвешенном графе плоскости данных. Один алгоритм строит замкнутое по дугам множество путей без зацикливания и дублирования и такое, что из каждого хоста ведёт ровно один путь в один из хостов заданного подмножества

целевых хостов. Другой алгоритм строит замкнутое по дугам множество путей без зацикливания и дублирования и такое, что из каждого хоста ведёт ровно один путь в хост каждого множества из заданного семейства множеств целевых хостов.

2. Определения и обозначения

Графом физических связей (далее просто графом связей) будем называть связный неориентированный граф $G = \{V, E\}$ без кратных рёбер и петель, где V – множество коммутаторов и хостов, $E \subseteq V \times V$ – множество рёбер, моделирующих физические связи между коммутаторами и между коммутаторами и хостами. Поскольку ребро, соединяющее вершины a и b, неориентированное и нет кратных рёбер, его можно обозначать как ab, так и ba. Поскольку нет петель, рёбер вида aa в E нет. Поскольку нет кратных рёбер, путь как последовательность смежных рёбер однозначно задаётся последовательностью вершин $a_1 \ldots a_n$, через которые он проходит. Если путь проходит по ребру ab из a в b, то будем говорить, что он проходит дугу ab. Если x и y хосты, то путь из x в y, в котором все остальные вершины коммутаторы, будем называть *полным* и обозначать как xy-путь. Путь, в котором вершины (дуги) не повторяются, называется *вершинно-простым* (*рёберно-простым*[1]). Вершины графа будем обозначать строчными буквами a, b, c, …, x, y, z, пути – жирными строчными буквами $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$,…, а множества путей – прописными буквами: P, Q, R, … На рисунках коммутаторам соответствуют белые кружки, а хостам – чёрные кружки; если безразлично, является вершина графа коммутатором или хостом, рисуются серые кружки.

Мы будем предполагать, что каждый хост x соединён в точности с одним коммутатором [4], и, как показано в [4-5], в этом случае достаточно рассматривать графы, в которых терминальные вершины суть хосты. Множества хостов и коммутаторов обозначается через H и S, соответственно; $H \cup S = V$, $H \cap S = \varnothing$, и под виртуальной сетью понимается конечное множество P полных путей между парами различных хостов, которое контроллер должен реализовать на плоскости данных, задавая нужные правила в коммутаторах.

В общем случае правило коммутатора b имеет вид σabc, где a и c соседи b, а σ вектор значений параметров заголовка пакета, которые используются в правилах. Такое правило означает, что коммутатор b, получив пакет с вектором σ от соседа a, пересылает его соседу c. В настоящей работе предполагается, что коммутатор не меняет σ. Тем самым, для вектора σ порождаются полные пути вида $a_1 \ldots a_n$, где для $i = 2..n - 1$ в коммутаторе a_i есть правило $\sigma a_{i-1} a_i a_{i+1}$, хост a_1 соединён с коммутатором a_2 и хост a_n соединён с коммутатором a_{n-1}. Если в коммутаторе есть два правила σabc и $\sigma abc`$, где $c \neq c`$, говорят, что пакет *клонируется*, то есть пересылается обоим соседям c и $c`$.

Вектора σ и $\sigma`$ считаются эквивалентными, если они определяют одни и те же пути, то есть для каждого правила $\sigma abc$ есть правило $\sigma`abc$ и, наоборот, для каждого правила $\sigma`abc$ есть правило σabc. В модели для множества всех векторов задаётся его разбиение на непересекающиеся подмножества, каждый из которых является подмножеством класса эквивалентных векторов. Каждому такому подмножеству ставится во взаимно-однозначное соответствие идентификатор пакета. Таким образом, каждому идентификатору соответствует множество путей, определяемых правилами коммутаторов, хотя разным идентификаторам может соответствовать одно и то же множество путей, если эти идентификаторы определяют подмножества векторов из одного и того же класса эквивалентных векторов. Если вектору σ соответствует идентификатор d, то вместо правила σabc рассматривается правило $dabc$. В [6]

[1] В литературе часто даётся другое определение: "путь рёберно-простой, если он не проходит дважды по одному ребру" (а не дуге), то есть путь, проходящий дважды по ребру, но в противоположных направлениях, не считается рёберно-простым. Однако для нашего определения «замкнутости по дугам» (см. ниже) удобнее использовать наше определение рёберно-простого пути.

показано, каким образом результаты по построению виртуальной сети можно распространить на реализацию конечного множества виртуальных сетей.

Заданное множество P полных путей однозначно определяет необходимый набор правил коммутаторов, порождающий все пути из P, однако при этом могут порождаться и новые пути, не принадлежащие P [4,5], которые не обязательно являются рёберно-простыми, то есть некоторые пакеты могут находиться в сети сколь угодно долго. В работах показано, что для точной реализации на плоскости данных множество путей должно быть замкнутым по дугам.

Множество рёберно-простых полных путей, то есть путей между хостами на плоскости данных через коммутаторы, называется *замкнутым по дугам*, если для любых двух путей множества с различными начальными и конечными хостами выполняется следующее: если после слияния на некоторой дуге эти пути разделяются (после этой же или другой дуги), то все четыре пути должны принадлежать исходному множеству [4,5]. Если множество P полных рёберно-простых путей замкнуто по дугам, то на плоскости данных дополнительных путей, а следовательно (для конечного P) и зацикливаний, не появляется; в [3] предложен алгоритм проверки, является ли заданное множество путей замкнутым по дугам.

Для учёта различных параметров при реализации виртуальных сетей и их корректировки в программно-конфигурируемых структурах плоскость данных моделируется взвешенным графом. Таким образом, мы полагаем, что задан граф связей, в котором каждому ребру приписано натуральное число («вес» ребра), то есть граф является взвешенным. В работе исследуются параметры двух типов: «ресурс» и «стоимость». Для параметра типа «ресурс» под весом ребра понимается его «ёмкость», и число путей, проходящих через ребро, не должно превышать ёмкость ребра. Для параметра типа «стоимость» под весом ребра понимается «цена» прохождения пути через ребро, «цена» пути — это сумма «цен» его рёбер, и ставится задача минимизации суммарной «цены» путей, реализующих виртуальную сеть.

3. Алгоритм корректировки виртуальной сети с учётом параметров типа «ресурс» при реализации на взвешенном графе плоскости данных

В этом разделе мы полагаем, что во взвешенном графе связей каждому ребру приписан некоторый вес, соответствующий параметру типа «ресурс», значение которого равно натуральному числу. Превышение допустимого числа путей, проходящих по ребру, может привести к потере пакетов или задержке при их отправке. Замкнутое по дугам множество путей P, то есть путей, не порождающих циклы в графе, называется *ресурсно-допустимым*, если число путей из P, проходящих по каждому ребру, не превышает вес ребра.

Мы предлагаем осуществлять построение ресурсно-допустимого множества путей в два этапа. На первом этапе строится замкнутое по дугам множество рёберно-простых путей, соединяющих требуемые пары хостов [5]. Если построенное множество не является ресурсно-допустимым, то предлагается простой способ для изменения дуг пути (рис. 1), используя рёбра взвешенного графа связей, которые отсутствуют в путях P (множество N в алгоритме 1). Замена ребра происходит, если число путей, проходящих через ребро, больше веса, сопоставленного ребру.

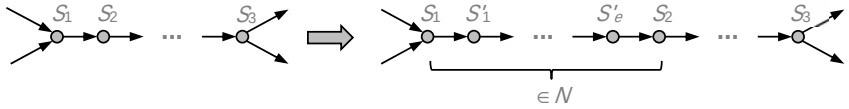

Рис. 1. Замена дуги s_1s_2 в путях множества P
Fig. 1. Replacing the arc s_1s_2 in the paths of P

Алгоритм 1: Модификация пути посредством изменения ребра или последовательности рёбер, при этом сохраняя головной и хвостовой узлы исходного пути.

***Вход**: непустое множество P рёберно-простых полных путей, замкнутое по дугам, непустое множество N рёбер между коммутаторами графа связей, которые не используются в путях из множества P.*

***Выход**: сообщение «множество P путей не может быть модифицировано» или модифицированное замкнутое по дугам множество Q, где головная и хвостовая вершины каждого изменённого пути совпадают с таковыми из исходного множества.*

while существует ребро (s_1, s_2) такое, что суммарное число k дуг s_1s_2 и s_2s_1, входящих в пути из множества P, больше ёмкости ребра (s_1, s_2)
do
  if в N существует подмножество рёбер $\{(s_1, s_1'), (s_1', s_2'), \ldots, (s_l', s_2)\}$, в котором ёмкость каждого ребра не меньше k,
  then
    В каждом пути из множества P заменить дугу s_1s_2 на путь $s_1s_1's_2' \ldots s_l's_2$,
    а дугу s_2s_1 – на путь $s_2s_l' \ldots s_2's_1's_1$;
    Удалить рёбра (s_1, s_1'), (s_1', s_2'), …, (s_l', s_2) из N;
  else
    модификация невозможна;

Можно показать, что если алгоритм 1 возвращает набор путей Q, то Q есть ресурсно-допустимое замкнутое по дугам множество рёберно-простых полных пути, причём для каждого модифицированного пути головная и хвостовая вершины совпадают с таковыми из исходного множества P. Действительно, рассмотрим два пути из множества P, которые пересекаются по дуге s_1s_2, и суммарное число k дуг s_1s_2 и s_2s_1, входящих в пути из множества P, больше ёмкости соответствующего ребра. Если в N существует подмножество рёбер $\{(s_1, s_1'), (s_1', s_2'), \ldots, (s_l', s_2)\}$, такое, что ёмкость каждого ребра не меньше k, то в каждом пути из множества P дуга s_1s_2 заменяется на путь $s_1s_1's_2' \ldots s_l's_2$, а дуга s_2s_1 – на путь $s_2s_l' \ldots s_2's_1's_1$. Поскольку каждый модифицируемый путь является рёберно-простым, и ни в каком другом месте модифицируемого пути дуг s_1s_1', $s_1's_2'$, …, $s_l's_2$ и s_2s_l', …, $s_1's_1s_1$ нет, то получим множество рёберно-простых путей Q, в котором для каждого модифицированного пути головная и хвостовая вершины совпадают с таковыми из исходного множества P. Более того, если до модификации для любых двух путей, для которых имелось слияние по некоторой дуге, а потом расхождение, все четыре пути принадлежали P, то и после модификации это свойство охраняется, то есть модифицированное множество путей является замкнутым по дугам. Сложность алгоритма $\boldsymbol{O}(m^2)$, где m число рёбер графа.

В алгоритме 1 мы рассматриваем достаточно простую эвристику для модификации путей, и результат существенно зависит от порядка перебора рёбер путей в P. Необходимы дополнительные исследования, чтобы детально исследовать вопрос о порядке такого перебора. Заметим также, что параметр типа «ресурс» можно ввести для дуг, заменив ребро в графе связей двумя соответствующими дугами, каждая из которых имеет свой ресурс, а также ввести такой параметр для коммутаторов. В обоих случаях можно использовать алгоритм 1 с небольшими модификациями.

4. Построение виртуальной сети на взвешенном графе физических связей с учётом параметров типа «стоимость»

Учёт параметров типа «стоимость» – это задача снижения нагрузки на сеть, когда нагрузка на сеть понимается как суммарное число прохождений пакетов по рёбрам графа: пара (пакет, ребро) учитывается, когда пакет проходит по ребру. Число пакетов, одновременно циркулирующих в сети SDN, зависит от поведения хостов, формирующих эти пакеты, и

потому является величиной переменной. Более грубая оценка характеризует виртуальную сеть независимо от движения пакетов по ней, она учитывает суммарное число не пакетов, а путей, по которым пакеты движутся. Пара (путь, ребро) учитывается, если путь проходит по ребру. Множество путей, реализуемых SDN, определяется настройкой коммутаторов, которая меняется по командам с верхнего уровня (SDN-контроллеры), что происходит значительно реже, и потому это множество путей в рамках рассматриваемой задачи может считаться статическим. Прохождение пути по разным рёбрам вносит в суммарную нагрузку слагаемые разной величины, что моделируется понятием веса ребра, понимаемым как «цена» его прохождения.

В нашей работе [7] рассматривалось выполнение на плоскости данных SDN программы задания, которая понималась в духе парадигмы объектно-ориентированного программирования как состоящая из объектов и сообщений, которыми объекты могут обмениваться. Объекты реализуется в хостах, причём в одном хосте может быть реализовано несколько разных объектов, а один и тот же объект может быть реализован в нескольких хостах. Сообщения между объектами, реализованными в разных хостах, помещаются в пакеты, двигающиеся по сети из начального хоста в конечный хост. В работе решались две задачи: 1) минимизация числа идентификаторов пакетов, соответствующих объектам, реализованным в хостах, 2) настройка коммутаторов (реализации путей, которые должны проходить пакеты) для выбранного соответствия идентификаторов пакетов и объектов. Эти задачи решались в двух случаях: A) без клонирования пакетов: пакет, предназначенный для некоторого объекта, должен попасть ровно в один хост, и в этом хосте должен быть реализован нужный объект, B) с клонированием пакетов: пакет может попадать в несколько хостов, но в одном и только одном из них должен быть реализован нужный объект.

Решение задачи 1 зависит от распределения объектов по хостам, реализующим эти объекты, и не зависит от графа физических связей. Решение задачи 2 использует результаты решения задачи 1 и зависит от графа физических связей. В случае A решением задачи 1 является непустое множество хостов $U(d)$, определяемое для каждого идентификатора пакета d. Требовалось провести пути из каждого хоста в один и только один хост из $U(d)$. В случае B решением задачи 1 является непустое семейство непустых непересекающихся множеств хостов $U(d)$, определяемое для каждого идентификатора пакета d. Хосты одного множества из семейства $U(d)$ реализуют одно и то же множество объектов, а каждый объект реализован в каждом хосте одного и только одного множества из семейства $U(d)$. Требовалось провести пути так, чтобы для каждого хоста h и каждого множества $U(d)_i$ из семейства $U(d)$ множество путей, начинающихся в хосте h, содержало один и только один путь, заканчивающийся в некотором хосте из множества $U(d)_i$.

В работе [7] нас не интересовали вопросы оптимизации нагрузки на SDN, то есть мы не учитывали веса рёбер (граф считался невзвешенным) при рассмотрении возможности реализации требуемой настройки коммутаторов. Однако такой учёт был специально указан как возможное направление дальнейших исследований, то есть рассмотрение взвешенного графа физических связей.

В данной работе мы корректируем решение задачи 2 для случая взвешенного графа с весами типа «стоимость». В предложенных в [7] решениях задачи 2 для невзвешенного графа строились кратчайшие пути, удовлетворяющие условиям задачи (случай A или B). Для взвешенного графа аналогичная задача 2 формулируется как задача минимизации суммарного веса путей, то есть задача о кратчайших взвешенных путях, то есть о путях с минимальной суммой весов входящих в эти пути рёбер.

4.1. Решение задачи построения путей с минимальной суммарной стоимостью для случая положительных целочисленных весов рёбер

В случае, когда веса рёбер выражаются натуральными числами (для невзвешенного графа веса всех рёбер равны 1), задача построения путей с минимальной суммарной стоимостью сводится к аналогичной задаче для невзвешенного графа с соответствующим увеличением сложности алгоритма. Для этого ребро с весом p преобразуется в вершинно-простой путь длиной p введением p - 1 промежуточных вершин степени 2 (рис. 2). Опишем это формально.

Под взвешенным графом будем понимать граф $G = (V, E, f)$, где V множество вершин, $E \subseteq V \times V$ множество рёбер, f: $E \to N$ функция веса ребра, N множество натуральных чисел. Задача построения кратчайших взвешенных путей в графе G сводится к задаче построения невзвешенных кратчайших путей в невзвешенном графе $G^* = (V^*, E^*)$, который получается из G удалением всех рёбер и добавлением для каждого ребра $ab \in E$ с весом $p = f(ab)$ дополнительных вершин $V(ab) = \{ c_i : i = 1..p - 1 \}$ и, обозначая $c_0 = a$, $c_p = b$, дополнительных рёбер $E(ab) = \{c_i c_{i+1} : i = 0..p\}$, где $c_i \notin V$, $i = 1..p - 1$. Тогда $V^* = V \cup \cup \{ V(ab) : ab \in E \}$ и $E^* = \{ E(ab) : ab \in E \}$.

Рис. 2. Преобразование ребра ab с весом p в путь из a в b длиной p
Fig. 2. Replacing edge ab of weight p with path from a to b of length p

Поскольку сложность алгоритма решения задачи 2A построения кратчайших путей для невзвешенного графа равна $\boldsymbol{O}(m)$, где m число рёбер, сложность соответствующего алгоритма для взвешенного графа равна $\boldsymbol{O}(M)$, где M сумма весов рёбер графа. Сложность алгоритма решения задачи 2B для невзвешенного графа равна $\boldsymbol{O}(km)$, где k мощность семейства U целевых хостов (U результат решения задачи 1B), а m число рёбер графа. Поэтому сложность соответствующего алгоритма для взвешенного графа равна $\boldsymbol{O}(kM)$.

Отсутствие зацикливания, дублирования и замкнутость по дугам строящегося множества путей непосредственно следует из того, что алгоритм для каждой вершины определяет в случае без клонирования единственный кратчайший путь в заданное множество U, а в случае с клонированием – ровно по одному пути в каждое множество из семейства U.

4.2. Оптимизация решения задачи построения путей с минимальной суммарной стоимостью для случая положительных целочисленных весов рёбер

Если сумма весов рёбер графа $M >> nm$, где n число вершин графа, то возможна оптимизация алгоритма со сложностью $\boldsymbol{O}(nm)$ для случая A или, учитывая, что $m = \boldsymbol{O}(n^2)$, со сложностью $\boldsymbol{O}(n^3)$, и со сложностью $\boldsymbol{O}(knm)$ для случая B, или, учитывая, что $m = \boldsymbol{O}(n^2)$, со сложностью $\boldsymbol{O}(kn^3)$.

Для невзвешенного графа алгоритм 2A основан на обходе неориентированного графа в ширину и может рассматриваться как модификация алгоритма Дейкстра, но вместо вычисления длины кратчайшего пути выполняется настройка коммутаторов вдоль пути. Идея алгоритма заключается в следующем. Начиная с множества хостов U, по графу распространяется «волна» построения путей с помощью пометок вершин. Фронт F этой волны состоит из помеченных вершин a с одним и тем же расстоянием до ближайшего хоста из множества U, тогда как остальные помеченные вершины находятся на меньшем расстоянии от U. Для каждой вершины a из фронта волны просматриваются её соседи и для каждого соседа b, который ещё не помечен, выставляется пометка и запоминается его сосед

$p(b) = a$, через которого ведёт кратчайший путь из вершины b до ближайшего хоста из множества U. Если вершина a коммутатор, в нём устанавливается правило $d\,b\,a\,p(a)$, означающее: при получении коммутатором a пакета с идентификатором d от вершины b следует направить этот пакет вершине $p(a)$. Если вершина b коммутатор, она помещается в новый фронт волны F_{next}. Когда просмотрены все вершины фронта F, начинается новый шаг, на котором фронтом волны становится F_{next}. Алгоритм завершает свою работу, когда просмотрены все вершины и фронт F стал пустым.

На каждом шаге алгоритма имеется множество помеченных вершин и его подмножество, фронт волны, содержащее вершины, у которых могут быть не просмотренные инцидентные им рёбра (эти рёбра могут вести в непомеченные вершины). На каждом шаге алгоритма просматриваются вершины фронта волны и не просмотренные инцидентные им рёбра. В новый фронт волны помещаются непомеченные вершины на концах таких рёбер. В случае взвешенного графа мы модифицируем граф G в граф G^*, превращая каждое ребро ab веса p в вершинно-простой путь $E(ab)$ из a в b длиной p - 1, добавляя дополнительные вершины c_i, $i = 1..p - 1$. Из-за этого несколько шагов подряд во фронт волны могут попадать только дополнительные вершины. Оптимизация заключается в том, чтобы объединить несколько шагов в один шаг так, чтобы на каждом объединённом шаге во фронт волны попадала хотя бы одна исходная (не дополнительная) вершина.

Пусть в графе G^* вершина v принадлежит фронту волны, $v \in F$, и ребро vu не просмотрено. Ребро vu лежит на пути $(d_1 = v)(d_2 = u)$, $d_2 d_3$, ..., $d_L(d_{L+1} = w)$, в котором конечная вершина w исходная (не дополнительная), а все промежуточные вершины d_i, $i = 1..k$, дополнительные. Длина этого пути равна L, обозначим её $L(vu)$. Минимум $l = \{ L(vu) \mid v \in F$ & ребро vu не просмотрено $\}$ определяет число шагов алгоритма, которые объединяются в один шаг. На этом объединённом шаге для каждого не просмотренного ребра vu помечается вершина на расстоянии l от вершины v вдоль пути, начинающегося ребром vu, то есть вершина d_{l+1}. На каждом объединённом шаге помечается хотя бы одна непомеченная ранее исходная (не дополнительная) вершина.

Заметим, что для определения минимальной длины $L(vu)$ не обязательно просматривать весь путь от вершины v до исходной (не дополнительной) вершины w, начинающийся ребром vu. Когда ребро ab с весом p превращается в путь $(c_0 = a)c_1$, c_1c_2, ..., $c_{p-1}(c_p = b)$ длиной p, в каждой вершине c_i, $i = 0..p$, можно хранить её расстояние до конца b этого пути, которое равно $p - i$. Тогда, если $v = c_i$, а $u = c_{i+1}$, то $L(vu) = p - i$.

Оценка сложности этой оптимизации. На каждом шаге просматривается не более m рёбер. Также на каждом шаге помечается хотя бы одна исходная (не дополнительная) вершина. Следовательно, сложность алгоритма не превышает $\boldsymbol{O}(nm) = \boldsymbol{O}(n^3)$. В случае B алгоритм делает то же самое, но для каждого из k множеств семейства U. Тем самым, оценка сложности равна $\boldsymbol{O}(knm) = \boldsymbol{O}(kn^3)$.

4.3. Решение задачи построения путей с минимальной суммарной стоимостью для случая произвольных неотрицательных весов рёбер

Это решение основано не на модификации графа, как в подразделах 4.1-4.2, а на модификации самих алгоритмов построения путей.

Поскольку рёбра имеют разные веса, кратчайший путь (путь с минимальным числом рёбер) может не совпадать с кратчайшим взвешенным путём (путь с минимальной суммой весов его рёбер). Поэтому предлагается следующая модификация алгоритма.

Каждый раз, когда из вершины a просматривается ребро ab, не обращаем внимания на то, помечена вершина b или нет, то есть пометки вершин не используются. Вместо этого в каждой вершине v хранится её взвешенное расстояние $L(v)$, как минимальная сумма весов рёбер пути до множества хостов U, вычисленное на данный момент времени. Если вершина

ещё не просматривалась, это расстояние считается бесконечным. Когда из вершины a просматривается ребро ab, проверяется условие $L(b) > L(a) + f(ab)$. Если условие выполнено, $L(b)$ меняется: $L(b) = L(a) + f(ab)$, и вершина b, если она коммутатор, помещается в новый фронт волны. Заметим, что условия $L(b) > L(a) + f(ab)$ и $L(a) > L(b) + f(ab)$ не могут одновременно выполняться, если вес ребра неотрицательный. Тем самым, вершина может несколько раз оказываться во фронте волны со строго убывающей последовательностью её расстояний до целевых хостов.

Оценим сложность алгоритма для случая A. Пусть максимальная невзвешенная длина пути от хоста до хоста из U равна l_{max}. Очевидно, что алгоритм закончит работу через l_{max} шагов. На каждом шаге просматривается не более m рёбер. Поскольку $l_{max} < n$, где n число вершин, оценка сложности равна $\boldsymbol{O}(nm) = \boldsymbol{O}(n^3)$. Для случая B то же самое делается для каждого из k множеств хостов из семейства U: каждый раз, когда в алгоритме 2A ребро просматривается один раз, в алгоритме 2B оно просматривается k раз. Поэтому оценка сложности равна $\boldsymbol{O}(knm) = \boldsymbol{O}(kn^3)$. Ниже приведены формальные описания алгоритмов.

Алгоритм_2A (V, NB, f, d, U, R) /* Настройка коммутаторов для кратчайшего взвешенного доступа (замкнутого по дугам, без зацикливания и дублирования) пакетов с идентификатором d от каждого хоста до хоста из множества U */

***Вход*:**

V – множество вершин графа физических связей;

NB – функция, задающая для каждой вершины a множество $NB(a)$ её соседей;

f – функция, задающая для каждого ребра ab его вес $f(ab)$;

d – идентификатор пакетов;

U – непустое множество целевых хостов, соответствующее d;

R – текущая настройка коммутаторов, задающая для каждого коммутатора s текущее множество правил вида $d'a\,s\,b$, где $d' \neq d$;

***Выход*:** новая настройка коммутаторов R.

$p(a)$ – сосед вершины a на пути к ближайшему (с учётом весов) хосту из U.

$L(a)$ – взвешенное расстояние от вершины a до хоста из U; $L(a) = \infty$ означает, что вершина ещё не просматривалась; вес ребра меньше ∞;

F – фронт (множество вершин) волны от хостов из U до других хостов.

F_{next} – множество F на следующем шаге.

```
F = ∅; F_next = ∅;
for all a ∈ V do L(a) = ∞;                    /* все вершины ещё не просматривались */
for all a ∈ U do                              /* хосты из U помещаются в фронт волны */
    L(a) = 0; F = F ∪ { a };
while F ≠ ∅ do                                /* пока фронт волны не пуст */
    for all a ∈ F do                          /* просмотр вершин a из фронта волны */
        for all b ∈ NB(a) do                  /* просматриваются соседи b вершины a */
            if L(b) > L(a) + f(ab) then       /* расстояние соседа b станет меньше */
                L(b) = L(a) + f(ab); p(b) = a;
                if | NB(a) | > 1 then         /* a коммутатор */
                    R(a) = R(a) ∪ { d b a p(a) };   /* правило коммутатора a */
                if | NB(b) | > 1 then         /* b коммутатор */
                    F_next = F_next ∪ { b };       /* помещаем b в следующий фронт */
return R;
```


Алгоритм_2B (V, NB, f, d, U, R) /* Настройка коммутаторов для кратчайшего взвешенного доступа (замкнутого по дугам, без зацикливания и дублирования) от каждого хоста пакетов с идентификатором d до одного хоста из каждого множества хостов в семействе множеств хостов U */

***Вход*:**

V – множество вершин графа физических связей;

NB – функция, задающая для каждой вершины a множество $NB(a)$ её соседей;

f – функция, задающая для каждого ребра ab его вес $f(ab)$;

d – идентификатор пакетов;

U – непустое семейство непустых множеств хостов, соответствующее d;

$k = | U |$;

R – текущая настройка коммутаторов, задающая для каждого коммутатора s текущее множество правил вида $d'a\,s\,b$, где $d' \neq d$;

***Выход*:** новая настройка коммутаторов R.

$p(a) = \{ p(a)(1), ..., p(a)(k) \}$ – вектор соседей вершины a на путях к ближайшим (с учётом весов) хостам из множеств, входящих в U,

$L(a) = \{ L(a)(1), ..., L(a)(k) \}$ – вектор взвешенных расстояний от вершины a до ближайшего (с учётом весов) хоста из множеств, входящих в U; $L(a)(i) = \infty$ означает, что вершина ещё не просматривалась; вес ребра меньше ∞;

F – фронт волны (множество вершин) от хостов из U до других хостов,

F_{next} – множество F на следующем шаге.

```
F = ∅; F_next = ∅;
for all a ∈ V do
    for all i ∈ { 1, ..., k } do
        L(a)(i) = ∞;                          /* все вершины ещё не просматривались */
for all i ∈ { 1, ..., k } do                  /* индексы i */
    for all a ∈ U(i) do                       /* вершины в U(i)*/
        L(a)(i) = 0; F = F ∪ { a };           /* хосты из U(i) помещаются в фронт волны */
while F ≠ ∅ do                                /* пока фронт волны не пуст */
    for all a ∈ F do                          /* просмотр вершин a из фронта волны */
        for all b ∈ N(a) do                   /* просмотр соседей b вершины a */
            for all i ∈ { 1, ..., k } do      /* индексы i */
                if L(b)(i) > L(a)(i) + f(ab) then   /* расстояние соседа b станет меньше */
                    L(b)(i) = L(a)(i) + f(ab); p(b)(i) = a;
                    if | NB(a) | > 1 then             /* a коммутатор */
                        R(a) = R(a) ∪ { ( d b a p(v)(i) };   /* правило коммутатора a */
                    if | NB(b) | > 1 then             /* b коммутатор */
                        F_next = F_next ∪ { b };      /* помещаем b в следующий фронт */
return R;
```

5. Заключение

В предыдущих работах [4-7] авторов не интересовали вопросы оптимизации нагрузки на SDN, и соответственно не учитывался вес ребра в графе связей, то есть граф считался невзвешенным при рассмотрении возможности реализации требуемой настройки коммутаторов. В настоящей работе рассматривается взвешенный граф связей для плоскости данных сети, в котором веса рёбер соответствуют параметрам типа «ресурс» и «стоимость», что налагает ограничения на маршрутизацию пакетов.

Вообще говоря, набор нефункциональных параметров, используемых для исследования вопросов надежности, безопасности и оптимизации SDN, может быть расширен, а кроме

того, могут использоваться сочетания параметров разных типов. Мы предполагаем продолжить наши исследования различных нефункциональных параметров и их сочетаний, а также их влияния на качество реализуемых SDN.

Список литературы / References

[1]. Sezer. S, Scott-Hayward. S, Chouhan P.K., Fraser B., Lake D., Finnegan J., Viljoen N., Miller M. and Rao N. Are we ready for sdn? Implementation challenges for software-defined networks IEEE Communications Magazine, 2013, 51 (7), pp. 36-43.

[2]. Mohammed, A. H., Khaleefah, R. M., k. Hussein, M., and Amjad Abdulateef, I. A review software defined networking for internet of things. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1–8.

[3]. OpenNetworkingFoundation (2012). Software-defined networking: The new norm for networks. ONF White Paper. 2012.

[4]. Burdonov, I.; Kossachev, A.; Yevtushenko, N.; López, J.; Kushik, N. and Zeghlache, D. (2021). Preventive Model-based Verification and Repairing for SDN Requests. In Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE, ISBN 978-989-758-508-1 ISSN 2184-4895, pages 421-428. DOI: 10.5220/0010494504210428.

[5]. Igor B. Burdonov, Nina Yevtushenko, Alexandre Kossachev: Verifying Multiple Virtual Networks in Software Defined Networks. Proceedings 2021 IEEE East-West Design & Test Symposium (EWDTS2021).

[6]. Igor Burdonov, Nina Yevtushenko and Alexander Kossatchev. Implementing a virtual network on the SDN data plane. Proceedings 2020 IEEE East-West Design & Test Symposium (EWDTS). 2020, pp. 279-283.

[7]. Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Реализация распределенных и параллельных вычислений в сети SDN. Труды института системного программирования. 2022. Т. 34. № 3. С. 159-172.

Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, главный научный сотрудник ИСП РАН. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Igor Borisovich BURDONOV – Dr. Sci. (Phys.-Math.), a Leading Researcher of ISP RAS. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

Нина Владимировна ЕВТУШЕНКО, доктор технических наук, профессор, главный научный сотрудник ИСП РАН, до 1991 года работала научным сотрудником в Сибирском физико-техническом институте. С 1991 г. работала в ТГУ профессором, зав. кафедрой, зав. лабораторией по компьютерным наукам. Её исследовательские интересы включают формальные методы, теорию автоматов, распределённые системы, протоколы и тестирование программного обеспечения.

Nina Vladimirovna YEVTUSHENKO, Dr. Sci. (Tech.), Professor, a Leading Researcher of ISP RAS, worked at the Siberian Scientific Institute of Physics and Technology as a researcher up to 1991. In 1991, she joined Tomsk State University as a professor and then worked as the chair head and the head of Computer Science laboratory. Her research interests include formal methods, automata theory, distributed systems, protocol and software testing.

Александр Сергеевич КОСАЧЕВ – кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН. Научные интересы: формальные спецификации, генерация тестов,

технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Alexander Sergeevitch KOSSATCHEV – Cand. Sci. (Phys.-Math.), a Leading Researcher of ISP RAS. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.