DOI: 10.15514/ISPRAS–2020–32(4)–18

Совершенные множества путей в полном графе коммутаторов SDN-сети

[1] *И.Б. Бурдонов, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[2] *Е.М. Винарский, ORCID: 0000-0002-7328-0942 <vinevg2015@gmail.com>*
[1,3] *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] *Московский государственный университет им. М.В. Ломоносова, 119234, Россия, г. Москва, Ленинские Горы, 1, д. 1*
[3] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В статье исследуется задача виртуализации сети на плоскости данных программно-конфигурируемой сети, моделируемой графом физических связей между узлами сети. Виртуальная сеть задается как множество упорядоченных пар хостов (отправитель, получатель), а реализуется множеством путей хост-хост, однозначно определяющим настройки коммутаторов. Множество путей совершенное, если любое подмножество связываемых им пар хостов связывается соответствующим подмножеством путей без возникновения бесконечного движения пакетов по циклу, без дублирующих путей, когда хост получает один и тот же пакет несколько раз, и без непредусмотренных путей, когда хост получает пакет, ему не предназначенный. Для случая, когда подграф, порождённый коммутаторами, является полным графом, устанавливаются достаточные условия существования наибольшего совершенного множества путей, связывающего все пары различных хостов. Предлагаются алгоритмы построения такого наибольшего совершенного множества и даются оценки их сложности. Приводятся результаты компьютерных экспериментов.

Ключевые слова: программно-конфигурируемые сети; виртуализация сети; совершенные множества путей; полный граф коммутаторов

Для цитирования: Бурдонов И.Б., Винарский Е.М., Евтушенко Н.В., Косачев А.С. Совершенные множества путей в полном графе коммутаторов SDN-сети. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 245–260. DOI: 10.15514/ISPRAS–2020–32(4)–18

Благодарности. Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 20-07-00338-а.

Perfect sets of paths in the full graph of SDN network switches

[1] *I.B. Burdonov, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[2] *E.M. Binarskii, ORCID: 0000-0002-7328-0942 <vinevg2015@gmail.com>*
[1,3] *N.V. Yevtushenko, ORCID: 0000-0002-4006-1161 <evtushenko@ispras.ru>*
[1] *A.S. Kossatchev, ORCID: 0000-0001-5316-3813 <kos@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, 1, Lenin mountains, Moscow, 119234, Russia*
[3] *National Research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia*

Abstract. The paper investigates the implementation of virtual networks on the SDN data plane which is modeled by a graph of physical connections between network nodes. A virtual network is defined as a set of ordered host pairs (sender, receiver), and it is implemented by a set of host-host paths that uniquely determine the switch settings. A set of paths is perfect if any subset of its paths can be loop-free implemented, i.e., can be implemented without the occurrence of an endless movement of packets in a loop, without duplicate paths, when the host receives the same packet several times, and without unintended paths when the host receives the packet that was directed to another host. For the case when the switchgraph is a complete graph, sufficient conditions for the existence of the largest perfect set of paths connecting all pairs of different hosts are established. Algorithms for constructing such a largest perfect set are proposed with the estimates of their complexity. The paper also has the preliminary results of computer experiments which show that proposed sufficient conditions are not necessary conditions.

Keywords: software defined networking; network virtualization; perfect sets of paths; complete graph of switches

For citation: Burdonov I.B., Binarskii E.M., Yevtushenko N.V., Kossatchev A.S. Perfect sets of paths in the full graph of SDN network switches. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 245–260 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–18

Acknowledgments. This work was supported by the Russian Foundation for Basic Research, project 20-07-00338-a.

1. Введение

Одной из основных технологий виртуализации [1-5] в настоящее время являются программно-конфигурируемые сети (SDN) с разделенными плоскостями данных и управления. Пакеты между хостами пересылаются на плоскости данных через промежуточные коммутаторы, система правил которых (настройка) осуществляется специальными SDN-контроллерами. Правило определяет, каким соседним узлам пересылается принятый коммутатором пакет в зависимости от того, откуда пришел пакет, и от вектора параметров в заголовке пакета [6]. Иными словами, настройка коммутаторов определяет множество путей от хоста к хосту, по которым и будут пересылаться пакеты. Ситуация может быть промоделирована с использованием графа физических связей, вершинами которого являются хосты и коммутаторы, а ребра соответствуют физическим связям между ними, при этом каждый хост может быть соединен только с одним коммутатором.

Соответственно в таком графе возникают задачи двух уровней. 1) Возможно ли и если да, то каким образом, реализовать заданное множество путей хост-хост через подходящие настройки коммутаторов? 2) Возможно ли и если да, то каким образом, реализовать заданное множество пар (хост, хост) через подходящие пути хост-хост в графе физических связей?

Известно, что при решении первой задачи возникают три эффекта, которые не позволяют реализовать произвольное множество путей хост-хост через подходящие настройки

коммутаторов. Во-первых, как показано в [4-5], при реализации множества реберно-простых путей хост-хост на панели данных могут появляться пути хост-хост, которых нет в заданном множестве, в том числе, могут появиться циклы, по которым пакеты будут передаваться бесконечно и, соответственно, бесконечно размножаться. Кроме того, могут появиться дублирующие пути, из-за чего хост-адресат получает один и тот же пакет не один, а несколько раз, и этот вопрос с точки зрения его полезности и нежелательных эффектов исследуется в [7].

Вторая задача сводится к первой задаче выбором подходящего множества путей, по возможности избегая описанных выше ситуаций, и таким образом, возникает вопрос (2a), можно ли заданное множество пар хостов реализовать на графе, т.е. выбрать подходящее множество путей, по возможности без нежелательных эффектов? На следующем шаге возникает вопрос (2b), можно ли любое множество пар хостов реализовать без указанных выше эффектов на данном графе физических связей?

В [7] показано, что любое множество пар хостов можно реализовать на связном графе без возникновения циклов и дублирования (ответ на вопрос 2a), однако могут возникать пути, соединяющие непредусмотренные пары хостов. Также показано, что в некоторых случаях реализация заданного множества пар хостов без непредусмотренных путей неизбежно приводит к возникновению дублирования или циклов.

В [7] также устанавливается достаточное условие положительного ответа на вопрос (2b). Это достаточное условие опирается на понятие наибольшего совершенного множества путей. Множество путей хост-хост называется совершенным, если никакие два пути, ведущие из разных начальных хостов в разные конечные хосты, не проходят по одному ребру графа физических связей в одном и том же направлении. Любое подмножество множества пар хостов, связываемых совершенным множеством путей, реализуется соответствующим подмножеством путей без «зацикливания», дублирования и непредусмотренных путей. Наибольшее совершенное множество путей связывает все пары разных хостов и, тем самым, любое множество пар разных хостов реализуется без «зацикливания», дублирования и непредусмотренных путей.

В данной статье исследуется вопрос о существовании наибольшего совершенного множества путей для случая, когда подграф, порождённый коммутаторами, является полным графом, т.е. для любой пары коммутаторов существует физическая связь. Устанавливаются достаточные условия существования наибольшего совершенного множества путей, предлагаются два алгоритма построения такого множества и даются их оценки (раздел 4). Эти алгоритмы опираются на два способа построения совершенного множества путей для случая, когда подграф, порождённый коммутаторами, является графом-звездой (раздел 3). Показано, что максимальное (по числу связываемых пар хостов) совершенное множество путей для графа-звезда строится одним из этих двух алгоритмов, выбираемых в зависимости от соотношения числа коммутаторов и числа хостов, подсоединенных к каждому коммутатору. Приводятся результаты компьютерных экспериментов (раздел 5). В частности, получен следующий результат. Как известно [7], наличие совершенного множества путей является достаточным условием для того, чтобы каждое множество пар различных хостов имело строгую реализацию без циклов и дублирования. Необходимость этого условия принималась как гипотеза, которая должна быть доказана или опровергнута. Эксперименты показали, что эта гипотеза не верна: для SDN-сети максимальное совершенное множество может не существовать, однако каждое множество пар различных хостов имеет строгую реализацию без циклов и дублирования.

2. Основные понятия

Графом физических связей (далее просто графом) будем называть связный неориентированный граф $G = \{V, E\}$ без кратных ребер и петель, где V – множество

коммутаторов и хостов, $E \subseteq V \times V$ – множество ребер, моделирующих физические связи между коммутаторами и между коммутаторами и хостами. Если не оговорено противное, под графом будет пониматься именно такой граф. Поскольку ребро, соединяющее вершины a и b, неориентированное и нет кратных ребер, его можно обозначать как ab, так и ba. Поскольку нет петель, ребер вида aa в E нет. Поскольку нет кратных ребер, путь как последовательность смежных ребер однозначно задается последовательностью вершин $a_1 \dots a_n$, через которые он проходит. Путь, начинающийся в вершине a и заканчивающийся в вершине b, называют ab-путем. Если путь проходит по ребру ab из a в b, то будем говорить, что он проходит дугу ab. Если a и b хосты, ab -путь, в котором все вершины, кроме первой вершины a и последней вершины b, коммутаторы, будем называть *полным*. Путь, в котором вершины (дуги) не повторяются, называется *вершинно-простым* (*реберно-простым*). Вершины графа будем обозначать строчными буквами $a, b, c, \dots, x, y, z$, пути – жирными строчными буквами $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}$, …, а множества путей – прописными буквами – P, Q, R, …

Мы будем предполагать, что каждый хост x подсоединен ровно к одному коммутатору [3]. Поэтому хост – это терминальная вершина графа, т.е. вершина степени 1. Коммутатор, к которому не подсоединены хосты, будем называть *пустым* коммутатором. Если коммутатор a имеет степень 1 и соединен с вершиной b, то любой полный путь, проходящий через a, имеет вид $\dots bab \dots$; удаляя из него все циклы bab, получаем путь, не проходящий через a. Это значит, что такой коммутатор «лишний», и достаточно рассматривать графы, в которых терминальные вершины – это только хосты. Множества хостов и коммутаторов обозначим через H и S, соответственно; $H \cup S = V$, $H \cap S = \varnothing$.

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

Заданное множество P полных путей однозначно определяет минимальный набор правил коммутаторов, порождающий все пути из P [5]. Однако это не значит, что порождаются только пути из P. Будем говорить, что два пути *сливаются* на дуге ab в вершине a, если у них дуга ab общая и не первая, а непосредственно предшествующие ей дуги ca и $c`a$ разные (т.е. $c \neq c`$), и *разделяются* после дуги de в вершине e, если у них дуга de общая и не последняя, а непосредственно следующие дуги ef и $ef`$ разные (т.е. $f \neq f`$).

Цикл порождается, если полный xy-путь проходит через некоторую дугу дважды, т.е. путь имеет вид $\boldsymbol{paqer}(\boldsymbol{aqer})^*\boldsymbol{aqes}$, где отрезок $\boldsymbol{p}$ начинается в хосте $x \neq a$, отрезки $\boldsymbol{p}$ и $\boldsymbol{r}$ не заканчиваются в одной вершине, после этих отрезков в пути следует коммутатор a, отрезок $\boldsymbol{aqer}$ проходится один или несколько раз, после коммутатора e отрезки $\boldsymbol{r}$ и $\boldsymbol{s}$ не начинаются в одной вершине, и отрезок $\boldsymbol{s}$ заканчивается в хосте $y \neq e$. Двигаясь вдоль пути, мы видим, что в вершине a путь сливается сам с собой, потом в вершине e разделяется сам с собой, а затем это слияние и разделение происходит ещё раз (если путь проходит цикл k раз, то будет $k + 1$ раз как разделение после слияния, так и слияние после разделения). Пакеты будут не только бесконечно ходить по циклу $\boldsymbol{aqer}$, но и бесконечно клонироваться в вершине e, так что хост y будет получать бесконечное число клонов пакета.

Путь, который не сливается сам с собой, это реберно-простой путь. Для отсутствия циклов необходимо, чтобы все пути множества P были реберно-простыми. Но этого недостаточно. Если два реберно-простых полных пути из P после слияния на дуге ab разделяются (после этой же или другой дуги), т.е. имеют вид $\boldsymbol{xpabqy}$ и $\boldsymbol{x`p`ab q`y`}$ с разными начальными и конечными хостами $x \neq x`$ и $y \neq y`$, то порождаются и новые пути $\boldsymbol{xpabq`y`}$ и $\boldsymbol{x`p`abqy}$. Эта операция порождения новых путей называется замыканием по дугам, а результат замыкания

по дугам всех пар путей из P обозначается $P{\downarrow}{\uparrow}$[4-5]. Очевидно, $P \subseteq P{\downarrow}{\uparrow}$. Если $P \neq P{\downarrow}{\uparrow}$, т.е. P не замкнуто по дугам, то возникают непредусмотренные пути. В частности, могут возникнуть не реберно-простые пути и, следовательно, циклы. Появление циклов в замыкании по дугам множества полных путей всегда свидетельствует о бесконечности этого замыкания и, тем самым, наличии дублирования. В конечном замкнутом по дугам множестве полных реберно-простых путей циклов нет.

Для множества полных путей P через $H(P) \subseteq H \times H$ обозначим множество пар xy, для которых в P есть xy-путь. Множество пар хостов $D \subseteq H \times H$, не содержащее пар вида xx, будем называть *нормальным*. Будем говорить, что нормальное множество D (*нестрого*) *реализуется* замкнутым по дугам множеством полных путей P, если $D \subseteq H(P)$, и, кроме того, *реализуется без циклов*, если P конечно, *строго реализуется*, если $D = H(P)$, *реализуется без дублирования*, если P содержит ровно один xy-путь для каждой пары $xy \in D$.

Если адрес хоста-отправителя входит в вектор параметров σ, то правила для векторов параметров с разными адресами хоста-отправителя работают независимо друг от друга. Для каждого хоста-отправителя x в графе можно выбрать исходящее из x дерево I_x кратчайших путей, ведущих во все остальные хосты. Для любого нормального множества D пар хостов и любого хоста x выбирается подмножество D_x пар, где первый элемент пары – это хост x, а в дереве I_x – поддерево $I_x(D)$, в котором листовые вершины – это вершины y такие, что $xy \in D_x$. В исходящем дереве все пути реберно-простые (даже вершинно-простые), и нет слияния, тем самым, нет и разделения после слияния. Поэтому $I_x(D)$ замкнуто по дугам и, очевидно, строго реализует D_x без циклов и дублирования, причём используются кратчайшие полные пути. Таким образом, в этом случае нет проблем со строгой реализацией без циклов и дублирования любого нормального множества пар хостов. Более того, реализация любого такого множества оказывается подмножеством одного и того же множества путей – объединения деревьев I_x по всем хостам x. Аналогичная процедура с аналогичным результатом применима тогда, когда в вектор параметров σ входит адрес хоста-получателя. Только здесь строится входящее дерево O_x для каждого хоста x.

Ниже мы рассматриваем случай, когда адрес хоста-отправителя и адрес хоста-получателя не входят в вектор параметров σ. Остальные параметры никак не влияют на перемещение пакетов с данным вектором σ, поэтому мы будем опускать σ в обозначении правила и писать вместо σ*abc* просто *abc*.[1] Вектор σ определяет идентификатор потока, в котором передаются векторы с такими параметрами. Иными словами, правила коммутатора (для данного вектора σ) определяют, кому должен быть послан пакет, только в зависимости от соседа, от которого пакет принят. В этом случае число правил, по которым работает коммутатор, зависит только от числа его соседей и не зависит от числа хостов в сети.

В [7] доказаны следующие утверждения.

1) На связном графе G любое нормальное множество D пар хостов нестрого реализуется без циклов и дублирования.
2) Любое множество D пар хостов, строго реализуемое без циклов, может быть строго реализовано множеством вершинно-простых путей.
3) Строгая реализация не всегда возможна без дублирования: существует граф, на котором некоторое нормальное множество пар хостов строго реализуется только с дублированием.
4) Строгая реализация не всегда возможна без циклов: существует граф, на котором некоторое нормальное множество пар хостов строго реализуемо, но только бесконечными замкнутыми по дугам множествами путей.

[1] В настоящей работе рассматриваются пакеты потока с одним идентификатором, и соответственно, при описании правил и путей опускается вектор параметров σ.

5) Если множество полных путей конечно, то отсутствие разделения после слияния достаточно, но не необходимо для замкнутости по дугам и отсутствия циклов.
6) Для замкнутого по дугам множества P полных путей условие отсутствия слияния после разделения необходимо и достаточно для отсутствия дублирования.

Также в [7] дано определение совершенного множества путей и доказано соответствующее утверждение.

Множество P путей называется *совершенным*, если оно конечно, содержит только полные пути, и в нем нет как разделения после слияния путей, т.е. нет двух путей вида $\boldsymbol{p}ab\boldsymbol{q}$ и $\boldsymbol{p}`ab\boldsymbol{q}`$, где $\boldsymbol{p} \neq \boldsymbol{p}`$ и $\boldsymbol{q} \neq \boldsymbol{q}`$,так и слияния после разделения путей, т.е. нет двух путей вида $\boldsymbol{p}ab\boldsymbol{q}cd\boldsymbol{r}$ и $\boldsymbol{p}`ab\boldsymbol{q}`cd\boldsymbol{r}`$, где $\boldsymbol{q} \neq \boldsymbol{q}`$. Если при этом множество $H(P)$ содержит все пары разных хостов (т.е. является наибольшим нормальным множество пар хостов), то множество P наибольшее совершенное множество. Заметим, что отсутствие разделения после слияния эквивалентно отсутствию дублирующих путей.

7) Совершенное множество путей для каждого нормального множества пар хостов содержит его строгую реализацию без циклов и без дублирования как своё подмножество.

3. Совершенное множество путей в графе-звезда

Граф-звезда – это граф, в котором выделена одна вершина – центр звезды, которая соединена ребрами со всеми остальными вершинами графа, и других ребер в графе нет. В этом разделе мы рассматриваем графы, в которых подграф, порожденный коммутаторами, является графом-звездой, и центр – пустой коммутатор (к нему не подсоединены хосты). Центр звезды обозначим через s_0, а остальные коммутаторы – через s_1, s_2, …, s_k,где k – число коммутаторов, не считая центра звезды. Хосты, подсоединенные к одному коммутатору, будет обозначать 1, 2, … Исследуем вопрос о максимальном совершенном множестве P путей в таком графе, где максимальность понимается как максимальное число связываемых им пар хостов, т.е. максимум $|H(P)|$.

Замечание 1. Пусть множество P путей совершенное. Удалим из P все пути, соединяющие хосты, подсоединенные к одному коммутатору, а потом добавим все пути вида $hs_ih`$, где $h$ и $h`$ разные хосты, подсоединенные к одному коммутатору s_i. Множество P останется совершенным.

С учетом этого замечания, а также утверждения 2 из [7], достаточно рассматривать совершенные множества вершинно-простых путей, соединяющих хосты, подсоединенные к разным коммутаторам. Хосты, подсоединенные к одному коммутатору, всегда можно дополнительно соединить путями, не проходящими по ребрам, соединяющим коммутаторы. Число таких дополнительных путей для k непустых коммутаторов, к каждому из которых подсоединено $n > 1$ хостов, равно $kn(n - 1)$.

Рассмотрим два способа соединения хостов в графе.

Способ 1: в одном непустом коммутаторе выделяем один хост, для определенности в коммутаторе s_1 выделяем хост 1, и соединяем его со всеми хостами h, подсоединенными к другим коммутаторам, путями вида $1s_1s_0s_ih$ и $hs_is_0s_11$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число пар соединяемых хостов $N_1 = 2n(k - 1)$.

Способ 2: в каждом непустом коммутатореs_i выделяем один хост 1 и соединяем путями вида $1s_is_0s_j1$ все выделенные хосты каждый с каждым. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число пар соединяемых хостов $N_2 = k(k - 1)$.

Утверждение 1. Для графа, в котором подграф, порожденный коммутаторами, является графом-звездой, центр звезды – пустой коммутатор, а к каждому из остальных k коммутаторов подсоединено по n хостов, максимальное совершенное множество путей

строится способом 1 или способом 2 в зависимости от того, какая величина больше: $2n(k-1)$ или $k(k-1)$.

Доказательство. Сначала нужно показать, что множество путей, которое строится по способу 1 или 2 совершенное. Но это непосредственно следует из построения: число путей конечно, все пути полные, любые два пути либо только сливаются, либо только разделяются.

Теперь докажем остальную часть утверждения.

В совершенном множестве путей пути, проходящие через одну дугу, либо разделяются, либо сливаются. Поэтому для данных i и j, $i \neq j$, все пути вида $hs_is_0s_jh`$ либо начинаются в одном хосте $h$ (разделение), либо заканчиваются в одном хосте $h`$ (слияние). Пусть число таких путей равно m. Если $m = 0$, $m = 1$ или $m > 1$ и имеет место разделение, то обозначим $\boldsymbol{m}(i,j) = m$, если $m > 1$ и имеет место слияние, то обозначим $\boldsymbol{m}(i,j) = -m$. В случае разделения $\boldsymbol{m}(i,j) > 1$ должно быть $\boldsymbol{m}(i`,j) = 0$ для $i` \neq i$, а в случае слияния $\boldsymbol{m}(i,j) < -1$ должно быть $\boldsymbol{m}(i,j`) = 0$ для $j` \neq j$. Кроме того, в случае разделения $\boldsymbol{m}(i,j) > 1$ мы можем, сохраняя совершенность множества путей, для каждого хоста $h`$, подсоединенного к коммутатору $s_j$, добавить путь $hs_is_0s_jh`$, если такого пути не было. Соответственно, в случае слияния $\boldsymbol{m}(i,j) < -1$ мы можем, сохраняя совершенность множества путей, для каждого хоста h, подсоединенного к коммутатору s_i, добавить путь $hs_is_0s_jh`$, если такого пути не было. Это значит, что в максимальном совершенном множестве путей $\boldsymbol{m}(i,j) > 1 \Rightarrow \boldsymbol{m}(i,j) = n$ и $\boldsymbol{m}(i,j) < -1 \Rightarrow \boldsymbol{m}(i,j) = -n$.

Рассмотрим полученную матрицу $\boldsymbol{m}(i,j)$, где $i = 1, ..., k$ и $j = 1, ..., k$. Она обладает следующими свойствами:

1) На главной диагонали находятся нули, т.е. $\boldsymbol{m}(i,i) = 0$ для $i = 1, ..., k$. Это объясняется тем, что мы не рассматриваем пути, соединяющие хосты, подсоединенные к одному коммутатору.
2) Если $\boldsymbol{m}(i,j) = n$, то в остальных ячейках столбца j находятся нули. Это разделение.
3) Если $\boldsymbol{m}(i,j) = -n$, то в остальных ячейках строки i находятся нули. Это слияние.
4) Если множество совершенных путей максимально, то в остальных ячейках матрицы находятся 1.

Матрицу порядка k с этими свойствами будем называть *правильной*. Число путей в совершенном множестве равно сумме абсолютных значений во всех ячейках матрицы, т.е. $\Sigma|\boldsymbol{m}(i,j)|$. Максимум $\Sigma|\boldsymbol{m}(i,j)|$ на классе всех правильных матриц обозначим M_k. Покажем, что $M_k = max\{2n(k-1), k(k-1)\}$. Для этого сначала докажем несколько лемм. Утверждение теоремы будет непосредственно следовать из лемм 4, 5, 6.

Лемма 1. Всегда существует правильная матрица $\boldsymbol{m}$, в которой $\Sigma|\boldsymbol{m}(i,j)| = 2n(k-1)$.

Доказательство. Рассмотрим матрицу, в которой в первой строке во всех ячейках, кроме первой, размещается "n", а в первом столбце во всех ячейках, кроме первой, размещается "$-n$". Очевидно, $\Sigma\,|\,\boldsymbol{m}(i,j)\,| = 2n(k-1)$.

□

Лемма 2. Если $n > 1$ и $2 < k \leq 2n$, то в правильной матрице, для которой достигается M_k, n и $(-n)$ либо оба присутствуют, либо оба отсутствуют.

Доказательство. По лемме 1, $M_k \geq 2n(k-1)$.

Рассмотрим матрицу, для которой имеет место один из двух случаев: 1) в матрице есть n и нет $(-n)$, 2) в матрице нет n и есть $(-n)$.

Случай 1. Пусть число ячеек с n, равно $x > 0$. Поскольку в столбце, где есть n, в остальных ячейках находится 0, число столбцов, где есть n, равно x. В каждом из остальных столбцов не более $(k-1)$ ячеек с 1, а остальные ячейки с 0. Тогда $\Sigma\,|\,\boldsymbol{m}(i,j)\,| \leq xn + (k-x)(k-1) = k(k-1) + x(n-k+1)$. Если $k \leq n+1$, то $(n-k+1) \geq 0$, и $x \leq k$ влечет $k(k-1) + x(n-k+1) \leq k(k-1) + k(n-k+1) = kn$; $k > 2$ влечет $kn < 2n(k-1)$. Тем самым,

$\Sigma\,|\,\boldsymbol{m}(i,j)\,| < 2n(k-1)$. Если $k > n+1$, то $(n-k+1) < 0$, и, учитывая, что $x > 0$, имеем $k(k-1) + x(n-k+1) < k(k-1)$; $k \leq 2n$ влечет $k(k-1) \leq 2n(k-1)$. Тем самым, $\Sigma\,|\,\boldsymbol{m}(i,j)\,| < 2n(k-1)$.

В случае 2 аналогично случаю 1 доказывается, что $\Sigma\,|\,\boldsymbol{m}(i,j)\,| < 2n(k-1)$.

Таким образом, в обоих случаях для рассматриваемой матрицы не достигается M_k. Тем самым, лемма доказана.

□

Лемма 3. Если $n > 1$ и $2 < k < 2n$, то в матрице, для которой достигается M_k, существует столбец, в которой есть $(-n)$, и строка, в которой есть n.

Доказательство. По лемме 1, $M_k \geq 2n(k-1)$, а по лемме 2 в матрице, для которой достигается M_k, n и $(-n)$ либо 1) оба присутствуют, либо 2) оба отсутствуют. Покажем, что второго случая быть не может. Действительно, если n и $(-n)$ оба отсутствуют, то на главной диагонали стоят 0, в остальных ячейках 0 или 1. Поэтому $\Sigma\,|\,\boldsymbol{m}(i,j)\,| \leq k(k-1)$. Если $k < 2n$, то $k(k-1) < 2n(k-1)$, поэтому $\Sigma\,|\,\boldsymbol{m}(i,j)\,| < 2n(k-1)$. Лемма доказана.

□

Лемма 4. Если $n > 1$и $2 \leq k < 2n$, то $M_k = 2n(k-1)$.

Доказательство. По лемме 1, $M_k \geq 2n(k-1)$. Покажем по индукции, что M_k не может быть больше $2n(k-1)$.

При $k = 2$ утверждение верно, поскольку сумма чисел в ячейках равна 0, 1, 2, n, $n+1$ или $2n$, а $n > 1$ влечет $2n(k-1) = 2n = max\{0, 1, 2, n, n+1, 2n\}$. Предположим, что утверждение верно при $k = m-1$, где $2 < m < 2n$ и покажем, что оно верно при $k = m$. По лемме 3, в матрице, для которой достигается M_m, существует столбец, в которой есть $(-n)$, и строка, в которой есть n. Без ограничения общности будем считать, что это последний столбец и последняя строка, и рассмотрим минор в левом верхнем углу порядка $(m-1)$. Если $M_m > 2n(m-1)$, то для этого минора $M_{m-1} > 2n(m-1) - 2n = 2n(m-2)$, что невозможно по предположению шага индукции.

□

Лемма 5. Если $n > 1$и $k = 2n$, то $M_k = 2n(k-1) = k(k-1)$.

Доказательство. По лемме 1, $M_k \geq 2n(k-1)$. Условия $n > 1$ и $k = 2n$ влекут $k > 2$. Согласно лемме 2, в матрице, для которой достигается M_k, n и $(-n)$ либо 1) оба присутствуют, либо 2) оба отсутствуют. В первом случае существуют столбец, в котором есть $(-n)$, и строка, в которой есть n. Без ограничения общности будем считать, что это последний столбец и последняя строка, и рассмотрим минор в левом верхнем углу порядка $(m-1)$. Заметим, что $2 < k = 2n$ влечет $2 \leq k-1 < 2n$. Поэтому по лемме 4, $M_{k-1} = 2n(k-2)$, и соответственно, $\Sigma\,|\,\boldsymbol{m}(i,j)\,| \leq M_{k-1} + 2n = 2n(k-1)$. Во втором случае $\Sigma\,|\,\boldsymbol{m}(i,j)\,| = k(k-1)$. Поскольку $k = 2n$, имеем $M_k = 2n(k-1) = k(k-1)$. Лемма доказана.

□

Лемма 6. Если $n > 1$, $k \geq 2n$, то $M_k = k(k-1)$.

Доказательство. Известно, как построить матрицу с $M_k = k(k-1)$. Эта матрица содержит все 1, кроме 0 на главной диагонали. Покажем по индукции, что M_k не может быть больше $k(k-1)$. По лемме 5 это верно для $k = 2n$. Предположим, что это так при $2n \leq k < m$ и покажем, что это верно при $k = m$.

Рассмотрим матрицу порядка m, в которой достигается M_m, и минор в левом верхнем углу порядка $(m-1)$. Определим, какие возможности есть для последней строки и столбца матрицы, чтобы получить максимальное значение. Возможны четыре случая: 1) последний столбец содержит n и остальные 0, последняя строка содержит $(-n)$ и остальные 0; 2) последний столбец содержит n и остальные 0, последняя строка содержит $(m-1)$ единиц и 0 в последнем столбце; 3) последняя строка содержит $(-n)$ и остальные 0, последний столбец

содержит (m - 1) единиц и 0 в последней строке; 4) последний столбец содержит (m - 1) единиц и 0 в последней строке, последняя строка содержит (m - 1) единиц и 0 в последнем столбце.

По предположению шага индукции в рассматриваемом миноре сумма абсолютных значений ячеек не превосходит (m - 1)(m - 2).

В случае 1 это верно, поскольку $m \ge 2n$ и $\Sigma \mid \boldsymbol{m}(i, j) \mid \le (m - 1)(m - 2) + 2n \le$

$\le (m - 1)(m - 2) + m < m\ (m - 2) + m = m\ (m - 1)$.

В случаях 2 и 3, условие $m \ge 2n$ влечет $\Sigma \mid \boldsymbol{m}(i, j) \mid \le (m - 1)(m - 2) + (m - 1) + n \le$

$\le (m - 1)(m - 2) + (m - 1) + m / 2 = m\ (m - 1) - (m - 1) + m / 2$. Поскольку $m > 2$, имеем $-(m - 1) + m / 2 < 0$, что влечет $m\ (m - 1) - (m - 1) + m / 2 < m\ (m - 1)$.

В случае 4, поскольку $m \ge 2n$, $\Sigma \mid \boldsymbol{m}(i, j) \mid \le (m - 1)(m - 2) + 2(m - 1) = m\ (m - 1)$.

4. Совершенное множество путей в полном графе

В этом разделе мы исследуем вопрос о существовании наибольшего совершенного множества путей для графа, в котором подграф, порожденный коммутаторами, является полным графом. Пусть имеется k непустых коммутаторов $s_1, s_2, \ldots, s_k$, к которым подсоединено $n_1, n_2, \ldots, n_k$ хостов, соответственно, остальные коммутаторы пустые. Упорядочим непустые коммутаторы по неубыванию числа подсоединенных к ним хостов, т.е. будем считать, что $n_1 \le n_2 \le \ldots \le n_k$. Как отмечено в разделе 3, достаточно рассматривать совершенные множества вершинно-простых путей, соединяющих хосты, подсоединенные к разным коммутаторам, поскольку хосты, подсоединенные к одному коммутатору, всегда можно дополнительно соединить путями, не проходящими по ребрам, соединяющим коммутаторы.

Для хоста x через $s(x)$ будем обозначать коммутатор, к которому подсоединен хост x.

Утверждение 2. Если подграф, порожденный коммутаторами, является полным графом, и существует не более одного коммутатора, к которому подсоединено более одного хоста, то существует наибольшее совершенное множество путей.

Доказательство. Рассмотрим множество P путей, состоящее из путей вида $xs(x)s(y)y$, где x и y хосты, $s(x) \ne s(y)$. Пути из P, очевидно, полные и вершинно-простые. Также очевидно, что для каждой пары хостов x и y, подсоединенных к разным коммутаторам, в P существует ровно один путь из x в y. Разъединение после слияния возможно только для двух путей вида $xs(x)s(y)y$ и $x`s(x`)s(y`)y`$, где $x \ne x`$, $y \ne y`$, $s(x) = s(x`)$, $s(x) \ne s(y)$, $s(y) = s(y`)$. Из этих условий следует, что $s(x)$ и $s(y)$ два разных коммутатора, к каждому из которых подсоединено более одного хоста, что противоречит условию. Таким образом, множество P после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору, является совершенным.

□

Далее рассматривается случай, когда имеется более одного коммутатора, к которому подсоединено более одного хоста. Мы определим два условия на число пустых коммутаторов, которые позволяют построить наибольшее совершенное множество путей, и, соответственно, предложим два алгоритма построения таких множеств путей. Эти алгоритмы основаны на алгоритмах для графа-звезда, рассмотренных в предыдущем разделе.

4.1 Алгоритм 1

Этот алгоритм основан на способе 1 построения максимального совершенного множества путей для графа-звезда.

Алгоритм 1. Для каждого непустого коммутатораs_i, $i = 1, \ldots, k$, выделяем один хост 1, подсоединенный к этому коммутатору. Алгоритм состоит из двух этапов.

Этап 1: Соединяем каждый выделенный хост 1, подсоединенный к коммутатору s_i, со всеми хостами h (не только выделенными), подсоединенными ко всем последующим коммутаторам s_j, где $j>i$, путями вида $1s_is_jh$ и hs_js_i1.

Этап 2: Далее для каждого невыделенного хоста $h \ne 1$, подсоединенного к коммутатору s_i, где $i < k$, выделяем пустой хост $s(i, h)$ и соединяем хост h со всеми хостами $h`$ (не только выделенными), подсоединенными ко всем последующим коммутаторам $s_j$, где $j > i$, путями вида $hs_is(i, h)s_jh`$ и $h`s_js(i, h)s_ih$.

Утверждение 3. Алгоритм 1 строит множество вершинно-простых путей, которое становится наибольшим совершенным множеством путей после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору, при условии, что число пустых коммутаторов не меньше $\Sigma\{ n_i \mid i = 1, \ldots, k \} - max\{ n_i \mid i = 1, \ldots, k \} - (k - 1)$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов $K_1 = (n - 1)(k - 1)$.

Доказательство. На этапе 2 для каждого невыделенного хоста h, подсоединенного к каждому коммутатору s_i, кроме последнего коммутатора s_k, выделяется пустой хост $s(i, h)$. Поскольку для каждого непустого коммутатора выделяется ровно один хост, подсоединенный к нему, число требуемых пустых коммутаторов равно (напомним, что мы упорядочили непустые коммутаторы по неубыванию числа подсоединенных к ним хостов) $\Sigma\{ n_i - 1 \mid i = 1, \ldots, k \} = \Sigma\{ n_i \mid i = 1, \ldots, k \} - max\{ n_i \mid i = 1, \ldots, k \} - (k - 1)$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов равно $(n - 1)(k - 1)$.

Покажем, что, если число пустых коммутаторов достаточно велико, построенное множество путей является наибольшим совершенным множеством вершинно-простых путей (после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору). Действительно, по построению все эти пути являются полными и вершинно-простыми, а их число конечно. Для каждых i и j, $i \ne j$, $h \ne 1$, $h` \ne 1$ строится следующее множество P путей:

	в хост 1		в хост $h` \ne 1$	
	$i < j$	$i > j$	$i < j$	$i > j$
из хоста 1	$1s_is_j1$	$1s_is_j1$	$1s_is_jh$	$1s_is(j, h)s_jh$
из хоста $h \ne 1$	$hs_is(i, h)s_j1$	hs_is_j1	$hs_is(i, h)s_jh`$	$hs_is(j, h`)s_jh`$

Мы видим, что множество P содержит единственный путь из каждого хоста в каждый хост, подсоединенный к другому коммутатору, т.е. $H(P)$ состоит из всех пар хостов, подсоединенных к разным коммутаторам. Пути, проходящие по дуге s_is_j, либо только разделяются, если $i < j$, либо только сливаются, если $i > j$. Пути, проходящие по дугам $s_is(i, h)$ и $s(i, h)s_j$, ведут из хоста h, подсоединенного к коммутатору s_i, и только разделяются, а пути, проходящие по дугам $s_is(j, h`)$ и $s(j, h`)s_j$, ведут в хост $h`$, подсоединенный к коммутатору s_j, и только сливаются. Тем самым, нет разделения после слияния и, поскольку нет дублирующих путей, нет слияния после разделения. Мы доказали, что построенное множество путей (после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору) является наибольшим совершенным множеством путей.

4.2 Алгоритм 2

Этот алгоритм основан на способе 2 построения максимального совершенного множества путей для графа-звезда. В алгоритме используется понятие минимального покрывающего набора глубиной 2 – это такое минимальное семейство $\mathcal{K}$ множеств хостов, что 1) в каждом множестве для каждого непустого коммутатора содержится ровно один хост, подсоединенный к этому коммутатору, 2) для любых двух хостов, подсоединенных к разным коммутаторам, в семействе есть множество, содержащее эти два хоста. Мощность минимального покрывающего набора глубиной 2 обозначается как $CAN(2, k, n_1, \ldots, n_k)$, где

n_i число хостов, подключенных к коммутатору s_i. Если $n_1 = \ldots = n_k = n$, то пишут сокращенно CAN(2, k, n).

Алгоритм 2. Пусть $\mathcal{K}$ минимальный покрывающий набор множества хостов глубиной 2. Выделим один элемент $\mathcal{K}$ и обозначим его L. Пусть число пустых коммутаторов не меньше $|\mathcal{K}|$ - 1. Выделим $|\mathcal{K}|$ - 1 пустых коммутаторов и поставим множество этих коммутаторов во взаимно-однозначное соответствие с множеством $\mathcal{K}\setminus\{L\}$. Коммутатор, соответствующий множеству $K \in \mathcal{K}\setminus\{L\}$ обозначим $t(K)$. Линейно упорядочим семейство $\mathcal{K}$: $K_0 = L$, K_1, …, $K_{|\mathcal{K}|-1}$. Для каждой пары хостов x и y, подсоединенных к разным коммутаторам, обозначим через $K(x, y)$ первое множество из $\mathcal{K}$, содержащее оба этих хоста. Очевидно, для каждой пары разных хостов x и y, принадлежащих L, будет $K(x, y) = L$. Строится множество $P = P_L \cup P_{\mathcal{K}}$, объединяемые множества определены ниже.

Множество P_L – это множество путей, соединяющих все пары разных хостов из L: для каждых $x \in L$, $y \in L$, $x \neq y$ выбирается путь $xs(x)s(y)y$ длиной 3.

Множество $P_{\mathcal{K}}$состоит из путей, соединяющих все остальные пары хостов, подсоединенных к разным коммутаторам. Для каждых хостов x и y таких, что $s(x) \neq s(y)$ и $x \notin L$ или $y \notin L$, выбирается путь $xs(x)t(K(x, y))s(y)y$ длиной 4.

Утверждение 4. Алгоритм 2 строит множество вершинно-простых путей, которое становится наибольшим совершенным множеством путей после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору, при условии, что число пустых коммутаторов не меньше CAN(2, k, n_1, …, n_k) - 1. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов K_2 = CAN(2, k, n) - 1.

Доказательство. По построению для успешного завершения алгоритма требуется число пустых коммутаторов не меньшее чем CAN(2, k, n_1, …, n_k) - 1.

Покажем, что, если число пустых коммутаторов достаточно велико, построенное множество путей является наибольшим совершенным множеством вершинно-простых путей (после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору).

1. Каждый путь из P вершинно-простой.

Вершины этого пути являются хостами тогда и только тогда, когда это начало и конец пути. Поэтому достаточно показать, что эти хосты разные, а все коммутаторы на пути тоже попарно разные. Путь из P_L имеет вид $xs(x)s(y)y$, где $x \in L$, $y \in L$, $x \neq y$. Эти условия влекут $s(x) \neq s(y)$, поэтому этот путь вершинно-простой. Путь из $P_{\mathcal{K}}$имеет вид $xs(x)t(K(x, y))s(y)y$, где $s(x) \neq s(y)$ и $x \notin L$ или $y \notin L$. Условие $s(x) \neq s(y)$ влечет $x \neq y$. Поскольку $t(K(x, y)) \in H_0$, то $s(x) \neq t(K(x, y))$, $s(y) \neq t(K(x, y))$. Поэтому этот путь вершинно-простой.

2. P содержит ровно один путь от каждого хоста x до каждого другого хоста y.

Пусть $x \in L$, $y \in L$, $x \neq y$. Тогда в P_L есть xy-путь и только один, а в $P_{\mathcal{K}}$ нет xy-путей. Пусть $x \notin L$ или $y \notin L$, и $s(x) \neq s(y)$. Тогда в $\mathcal{K}\setminus\{L\}$ есть множество $K(x, y)$, которому принадлежат x и y, и ровно один xy-путь, а в P_L нет xy-путей.

3. Разъединение после слияния.

Для разъединения после слияния двух путей каждый из них должен быть длиной не меньше 3. Рассмотрим 6 теоретически возможных случаев, и покажем, что ни один из этих случаев невозможен для путей из P.

1) Два пути длиной 4 из $P_{\mathcal{K}}$: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)t(K(x`, y`))s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$, $t(K(x, y)) = t(K(x`, y`))$. В этом случае в одном множестве $K(x, y) = K(x`, y`)$ есть два разных хоста $x$ и $x`$, подсоединенных к одному коммутатору $s(x) = s(x`)$, что невозможно по построению.
2) Два пути длиной 4 из $P_{\mathcal{K}}$: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)t(K(x`, y`))s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = t(K(x`, y`))$, $t(K(x, y)) = s(y`)$. В этом случае к каждому из коммутаторов $s(x)$ и $s(y`)$ подсоединено более одного хоста, а $K(x, y)$ и $K(x`, y`)$ должны быть пустыми коммутаторами, что противоречит равенствам $s(x) = t(K(x`, y`))$, $t(K(x, y)) = s(y)$.
3) Два пути длиной 4 из $P_{\mathcal{K}}$: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)t(K(x`, y`))s(y`)y`$, где $x \neq x`$, $y \neq y`$, $t(K(x, y)) = t(K(x`, y`))$, $s(y) = s(y`)$. В этом случае в одном множестве $K(x, y) = K(x`, y`)$ есть два разных хоста $y$ и $y`$, подсоединенных к одному коммутатору $s(y) = s(y`)$, что невозможно по построению.
4) Два пути длиной 3 из P_L: $xs(x)s(y)y$ и $x`s(x`)s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$, $s(y) = s(y`)$. В этом случае два разных хоста x и $x`$подсоединены к одному коммутатору $s(x) = s(x`)$, и два разных хоста y и $y`$подсоединены к одному коммутатору $s(y) = s(y`)$, что невозможно для P_L по построению.
5) Один путь длиной 4 из $P_{\mathcal{K}}$и другой путь длиной 3 из P_L: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)s(y`)y`$, где $x \neq x`$, $y \neq y`$, $t(K(x, y)) = s(x`)$ и $s(y) = s(y`)$. В этом случае к коммутатору $s(x`)$ должно быть подсоединено более одного хоста, а коммутатор $t(K(x, y))$ должен быть пустым коммутатором, что противоречит $t(K(x, y)) = s(x`)$.
6) Один путь длиной 4 из $P_{\mathcal{K}}$и другой путь длиной 3 из P_L: $xs(x)t(K(x,))s(y)y$ и $x`s(x`)s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$ и $t(K(x, y)) = s(y`)$. В этом случае к коммутатору $s(y`)$ должно быть подсоединено более одного хоста, а коммутатор $t(K(x, y))$ должен быть пустым коммутатором, что противоречит $t(K(x, y)) = s(y`)$.

Таким образом, мы доказали, что построенное множество путей (после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору) является наибольшим совершенным множеством путей.

4.3 Сравнение двух алгоритмов

Выбор алгоритма 1 или алгоритма 2 для построения наибольшего совершенного множества путей определяется соотношением числа путей. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов в этих алгоритмах равно $K_1 = (n - 1)(k - 1)$ и K_2 = CAN(2, k, n) - 1.

Для CAN(2, k, n) существуют только асимптотическая оценка сверху CAN(2, k, n) $\leq n^2\log(k)(1 + o(1))$ при $n\to\infty$ и очевидная нижняя граница CAN(2,k,n) $\geq n^2$, основанная на том, что всевозможные комбинации из 2 значений, каждое из которых можно выбрать n способами, в покрывающем наборе глубины 2 должны присутствовать [8].

Если k фиксировано, а $n\to\infty$, то асимптотически $(n - 1)(k - 1) < n^2 - 1$ (меньше нижней границы). Если, наоборот, n фиксировано, а $k\to\infty$, то асимптотически $(n - 1)(k - 1) > n^2 log k - 1$ (больше верхней границы). Поэтому общая оценка для этих двух алгоритмов $min\{(n - 1)(k - 1)$, CAN(2,k, n) - 1$\}$.

К сожалению, более точные оценки CAN(2,k, n) неизвестны. Только для n = 2 доказано, что $k = (x - 1)! / (\lfloor x/2 \rfloor - 1)!\ (x - \lfloor x/2 \rfloor)!$, где x = CAN(2, k, 2) [9]. Уже для n = 3 CAN(2, k, 3) известно только для $k \leq 7$ [10].

5. Экспериментальные результаты

5.1 Формула логики первого порядка для нахождения максимального совершенного множества

Согласно утверждению 2, для любого $k > 0$, если подграф, порожденный коммутаторами $s_1, \ldots, s_k$, является полным графом, и существует не более одного коммутатора, к которому подсоединено более одного хоста, то существует наибольшее совершенное множество путей. Возникает следующий вопрос: остаётся ли утверждение верным, если существует более одного коммутатора, к которому подсоединено более одного хоста. Проведенные

компьютерные эксперименты показывают, что это не так. Чтобы доказать этот факт, мы свели задачу поиска совершенного множества путей к задаче проверки выполнимости формулы логики первого порядка [11], и для проверки выполнимости такой формулы использовали солвер Z3 [12] для языка программирования python 2.7. Все исходные коды экспериментов доступны по ссылке [13].

Без потери общности рассуждений можно рассматривать SDN-сеть с коммутаторами $s_1, \dots, s_k$ для $k \geq 2$, в которой

- коммутаторы s_1 и s_2 соединены с двумя хостами;
- для любого $i \in \{3, \dots, k\}$ коммутатор s_i соединенён с одним хостом.

Запишем условие существования совершенного множества путей в виде формулы логики первого порядка. Если такая формула выполнима, то существует максимальное совершенное множество путей, иначе такого множества не существует. Введем следующие обозначения:

- к коммутатору s_1 присоединены хосты h_1 и h_2;
- к коммутатору s_2 присоединены хосты h_3 и h_4;
- для любого $i \in \{3, \dots, k\}$ к коммутатору s_i присоединён коммутатор h_{i+2}.

Количество хостов в такой SDN-сети равно $k + 2$. Напомним, что любой вершинно-простой полный путь $path = hss_{i_1} \dots s_{i_m}s'h'$, где $m \leq k$. Обозначим ребро hs как *начальный отрезок* пути $path$, и ребро $s'h'$ - как *конечный отрезок* пути $path$. Полный путь между хостами h_i и h_j обозначим $path_{i,j}$. Определим следующие предикаты:

- $diffSwitch(h, h')$ принимает значение «истина», если хосты h и h' присоединены к различным коммутаторам;
- $diffBeg(path, path')$ принимает значение «истина», если полные пути $path$ и $path'$ имеют различные начальные отрезки;
- $diffEnd(path, path')$ принимает значение «истина», если полные пути $path$ и $path'$ имеют различные конечные отрезки.

Обозначим через $len(path)$ длину полного пути $path = hss_{i_1} \dots s_{i_m}s'h'$. Для $x \in \{1, len(path)\}$ элемент, стоящий на позиции x, обозначим $path[x]$. Тогда $h = path[1]$, и $h' = path[len(path)]$. Запишем условие отсутствия пересечения ($Ind(path, path')$) по ребру двух полных путей $path$ и $path'$:

$$Ind(path, path') = \forall x \forall y\big((x \in \{2, len(path) - 1\}) \& (y \in \{2, len(path') - 1\})\big) \rightarrow \big((path[x]! = path[y]) \cup (path[x + 1]! = path[y + 1])\big).$$

Истинность формулы ниже устанавливает наличие максимального совершенного множества путей. Для любых $i, j, k, l \in \{1, \dots, k + 2\}$ справедливо: если h_i и h_j присоединены к различным коммутаторам, и h_k и h_l присоединены к различным коммутаторам, то должна быть истинна следующая формула

$$\big(diffBeg(path_{ij}, path_{kl}) \& diffEnd(path_{ij}, path_{kl})\big) \rightarrow Ind(path_{ij}, path_{kl}). \quad (1)$$

Интуитивно, формула означает, что если любые два полных пути не сливаются и не разделяются, то они не имеют общих рёбер. Далее эта формула проверяется при помощи солвера Z3.

5.2 Случай с 4-мя коммутаторами и 6-ю хостами

Рассмотрим применение формулы (1) для нахождения наибольшего совершенного множества путей для SDN-сети, изображённой на рис. 1.

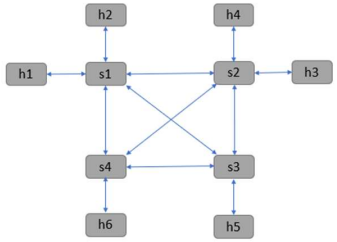

Рис.1. SDN-сеть с 4-мя коммутаторами и 6-ю хостами
Fig. 1. SDN network with 4 switches and 6 hosts

Формула (1) не верна для топологии, изображённой на рис. 1. Соответствующий код на языке python, реализующий данную топологию и использующий солвер Z3, доступен по ссылке [13].

В [7] было показано, что наличие совершенного множества путей является достаточным условием для того, чтобы каждое нормальное множество пар хостов имело строгую реализацию без циклов и дублирования. Необходимость этого условия принималась как гипотеза, которая должна быть доказана или опровергнута. Эксперименты показали, что эта гипотеза не верна: для SDN-сети, изображённой на рис. 1, максимального совершенного множества не существует, но каждое нормальное множество пар хостов имеет строгую реализацию без циклов и дублирования. Отчёт выполнения солвера Z3 доступен по ссылке [13].

Тот факт, что не существует наибольшего совершенного множества для SDN-сети, изображённой на рис. 1, даёт основание для выдвижения следующей гипотезы: для любого $k > 0$, если подграф, порожденный коммутаторами $s_1, \dots, s_k$, является полным графом, и существует более одного коммутатора, к которому подсоединено более одного хоста, то не существует наибольшего совершенного множества путей. Однако это не так, и в следующем разделе мы приводим соответствующий контрпример.

5.3. Случай с 5-мя коммутаторами и 7-ю хостами

Рассмотрим SDN-сеть с 5-ю коммутаторами и 7-ю хостами, изображённую на рис. 2.

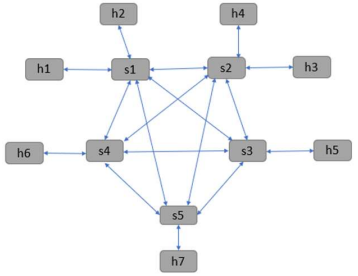

Рис.2. SDN-сеть с 5-ю коммутаторами и 7-ю хостами
Fig. 2. SDN network with 5 switches and 7 hosts

Формула (1) верна для топологии, изображённой на рис. 2. Соответствующий код на языке python, реализующий данную топологию и использующий солвер Z3, доступен по ссылке [13]. Наибольшее совершенное множество путей изображено на рис. 3.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7
h_1	$h_1s_1h_1$	$h_1s_1h_2$	$h_1s_1s_2h_3$	$h_1s_1s_2h_4$	$h_1s_1s_3h_4$	$h_1s_1s_2s_5s_3s_4h_5$	$h_1s_1s_2s_5h_7$
h_2	$h_2s_1h_1$	$h_2s_1h_2$	$h_2s_1s_5s_4s_2h_3$	$h_2s_1s_5s_2h_4$	$h_2s_1s_3h_5$	$h_2s_1s_5s_3\ s_4h_6$	$h_2s_1s_5h_7$
h_3	$h_3s_2s_1h_1$	$h_3s_2s_3s_1h_2$	$h_3s_2h_3$	$h_3s_2h_4$	$h_3s_2s_3h_5$	$h_3s_2s_3s_4h_6$	$h_3s_2s_3s_5h_7$
h_4	$h_4s_2s_1h_1$	$h_4s_2s_4s_3s_1h_2$	$h_4s_2h_3$	$h_4s_2h_4$	$h_4s_2s_4s_3h_5$	$h_4s_2s_4h_6$	$h_4s_2s_4s_3s_5h_7$
h_5	$h_5s_3s_2s_1h_1$	$h_5s_3s_1h_2$	$h_5s_3s_2h_3$	$h_5s_3s_2h_4$	$h_5s_3h_5$	$h_5s_3s_4h_6$	$h_5s_3s_5h_7$
h_6	$h_6s_4s_1h_1$	$h_6s_4s_1h_2$	$h_6s_4s_2h_3$	$h_6s_4s_5s_2h_4$	$h_6s_4s_1s_3h_5$	$h_6s_4h_6$	$h_6s_4s_5h_7$
h_7	$h_7s_5s_1h_1$	$h_7s_5s_1h_2$	$h_7s_5s_1s_4s_2h_3$	$h_7s_5s_2h_4$	$h_7s_5s_1s_3h_5$	$h_7s_5s_3s_4h_6$	$h_7s_5h_7$

Рис.3. Наибольшее совершенное множество путей для SDN-сети с 5-ю коммутаторами и 7-ю хостами
Fig. 3. Largest perfect set of paths for an SDN network with 5 switches and 7 hosts

6. Заключение

Таким образом, в результате проведенных авторами исследований установлены достаточные условия существования наибольшего совершенного множества путей для плоскости данных программно-конфигурируемых сетей (SDN), предложены алгоритмы построения таких множеств и получены оценки их сложности. Компьютерные эксперименты показали, что для SDN-сети наибольшее совершенное множество может не существовать, однако каждое множество пар различных хостов имеет строгую реализацию без циклов и дублирования.

Проведённые компьютерные эксперименты показывают, что условия существования наибольшего совершенного и замкнутого множества для произвольной SDN-сети должны быть изучены отдельно. Эти вопросы авторы планируют исследовать в будущей работе.

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, vol. 51, no. 7, 2013, pp. 36-43.

[2] López J., Kushik N., Yevtushenko N. and Zeghlache D. Analyzing and Validating Virtual Network Requests. In Proc. of the 12th International Conference on Software Technologie, 2017, pp 441-446.

[3] Yevtushenko N, Kossatchev A, Lopez J, Kushik N and Zeghlache D. Test Derivation for the Software Defined Networking Platforms: Novel Fault Models and Test Completeness. In Proc. of the IEEE East-West Design and Test Symposium, 2018, pp 1-6.

[4] Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 69-88. DOI: 10.15514/ISPRAS-2018-30(6)-4 / Burdonov I.B., Yevtushenko N.V. and Kossatchev.A.S. Testing switch rules in software defined networks. Trudy ISP RAN/Proc. ISP RAS, vol 30, issue 6, 2018, pp 69-88 (in Russian).

[5] Burdonov I., Kossatchev A., Yevtushenko N., López J., Kushik N., and Zeghlache D. Verifying SDN Data Path Requests. arXiv:1906.03101, 2019.

[6] Boufkhad Y., De La Paz R., Linguaglossa L., Mathieu F., Perino D., and Viennot L, Forwarding tables verification through representative header sets. arXiv:1601.07002, 2016.

[7] Burdonov I., Yevtushenko N., and Kossatchev A. Implementing a virtual network on the SDN data plane. Accepted at the 2020 IEEE East-West Design and Test Symposium.

[8] Кулямин В.В., Петухов А.А. Обзор методов построения покрывающих наборов. Программирование, том 37, № 3, стр. 3-41 / V.V. Kuliamin, A.A. Petukhov. A survey of methods for constructing covering arrays. Programming and Computer Softwareб vol. 37, № 3, 2011, article no. 121. https://doi.org/10.1134/S0361768811030029.

[9] Kleitman Daniel J. and Spencer Joe. Families of k-independent sets. Discrete mathematics, vol. 6, 1973, pp. 255-262.

[10] Choi Soohak, Kim Hyun Kwang, Oh Dong Yeol. Structures and lower bounds for binary covering arrays. Discrete mathematics, vol. 312, 2012, pp. 2958-2968.

[11] Верещагин Н К, Шень А. Лекции по математической логике и теории алгоритмов. Часть 2. Языки и исчисления. МЦНМО, 2012, 240 стр. / Vereshchagin N.K., Shen A. Lectures on mathematical logic and theory of algorithms. Part 2. Languages and Calculus. Moscow center for continuous mathematical education, 2012, 240 p. (in Russian).

[12] Yurichev D. SAT/SMT by Example. URL; https://yurichev.com/SAT_SMT.html, accessed 20,07.2020.

[13] Vinarskii E. perfect_set. URL: https://github.com/vinevg1996/perfect_set, accessed 20,07.2020.

Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, ведущий научный сотрудник. 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 – Doctor of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

Евгений Максимович ВИНАРСКИЙ – студент магистратуры, кафедра математической кибернетики факультета ВМК. Научные интересы: конечные автоматы, верификация.

Evgeny Maksimovich VINARSKY – Master's student, Department of Mathematical Cybernetics, Faculty of CMC. Research interests: finite state machines, verification.

Нина Владимировна ЕВТУШЕНКО, доктор технических наук, профессор, г.н.с. 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, распределенные системы, протоколы и тестирование программного обеспечения.

Nina Vladimirovna YEVTUSHENKO, Doctor of Technical Sciences, 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. 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 – Candidate of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.