DOI: 10.15514/ISPRAS–2021–33(1)–9



# Безопасная реализация виртуальной сети на плоскости данных SDN

[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**. 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. Возможности передачи пакетов ограничиваются весами (приоритетами) хостов: пакет может быть передан только от хоста к хосту с не меньшим приоритетом. Соответственно множество путей допустимое, если любое подмножество связываемых им пар хостов является допустимым. В работе показывается, что в отличие от случая, когда любая пара различных хостов является допустимой, в графе с приоритетами не для любого множества пар допустимых хостов существует допустимая реализация, т.е. реализация в виде допустимого множества путей. Кроме того, показывается, что в ряде случаев, когда такая реализация существует, она не всегда возможна без путей с циклами, т.е. путей, допускающих бесконечное движение пакетов по циклу, и без дублирующих путей, когда хост получает один и тот же пакет несколько раз. С использованием понятия совершенного множества путей сформулировано и доказано требование к графу с приоритетами, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием.

**Ключевые слова:** программно-конфигурируемые сети; виртуализация сети; безопасность; приоритеты хостов; (приоритетно-)допустимая реализация множества пар хостов

**Для цитирования**: Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Безопасная реализация виртуальной сети на плоскости данных SDN. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 123-136. DOI: 10.15514/ISPRAS–2021–33(1)–9.





# Secure Implementing a Virtual Network on the SDN Data Plane

[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] *Ivannikov 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 paper continues the investigations on 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. The opportunities to transmit a packet are limited by the host weights (priorities): a packet can be only transmitted from a host to a host if the sender has at most the same priority as the recipient, and thus, a set of paths is permissible if its every subset connects permissible host pairs. In the paper, it is proven that differently from the case when every host pair is permissible, in the graph with priorities a permissible path implementation does not exist for every set of permissible hosts. Moreover, it is shown that is some cases when such an implementation exists, the implementation is not possible without paths with cycles where packets can move infinite and without duplicate paths when a host can get the same packet several times. Using the notion of a perfect set of paths a criterion is established when every permissible set of hosts can be safely implemented by a set of paths without cycles but possibly with duplicate paths.

**Ключевые слова:** software defined networks (SDN); network virtualization; security; host priority; permissible implementation of the host set

**For citation**: Burdonov I.B., Yevtushenko N.V., Kossatchev A.S. Secure Implementing a Virtual Network on the SDN Data Plane. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 123-136 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–9.

## *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.

В работе [7] обсуждаются вопросы возможности реализации заданного множества пар (хост, хост) через подходящие пути хост-хост в графе физических связей, которые, в свою очередь, определяют те или иные настройки коммутаторов. Как известно, при решении этой задачи возникают три эффекта. 1) Как показано в [4-5] при реализации (через подходящие настройки коммутаторов) некоторого множества реберно-простых путей, связывающих заданные пары (хост, хост), на панели данных могут появляться непредусмотренные пути хост-хост, т.е. пути, которых нет в этом множестве путей и которые могут связывать пары хостов, отсутствующие в заданном множестве пар хостов. Для учёта этого эффекта при реализации нужно рассматривать только замкнутые по дугам множества реберно-простых путей, которые не меняются при их реализации через настройки коммутаторов. 2) При реализации

пар хостов могут появиться циклы, по которым пакеты будут передаваться бесконечно и, соответственно, бесконечно размножаться. 3) Кроме того, могут появиться дублирующие пути, из-за чего хост-адресат получает один и тот же пакет не один, а несколько раз. В [7] показано, что строгая реализация заданного множества пар хостов, при которой пути связывают все заданные пары хостов и только такие пары хостов, не всегда возможна без появления циклов и/или без дублирования. В то же время нестрогая реализация, когда требуется связать все заданные пары хостов, но разрешено связывать и непредусмотренные пары хостов, всегда возможна без появления циклов и без дублирования.

В данной статье рассматривается задача реализации множества пар хостов с учётом политики безопасности (security), согласно которой некоторые хосты не могут передавать пакеты определенным хостам. Для описания множества хостов, от которых заданный хост, может принимать пакеты, каждому хосту приписано натуральное число (*приоритет*), и хост может передавать пакеты только хостам с не меньшими приоритетами. Соответственно, упорядоченные пары хостов разделяются на (приоритетно-) *допустимые* и *недопустимые*. Все пары из заданного множества пар хостов должны быть допустимыми, но при реализации этого множества разрешается связывать путями и другие допустимые пары хостов. В то же время недопустимые пары хостов ни при каких условиях не должны связываться путями. Такое ослабление требования о непредусмотренных парах хостов даёт больше свободы по сравнению со строгой реализацией, но ограничивает свободу нестрогой реализации, а именно для заданных пар хостов допускается нестрогая реализация, но не любая, а только та, которая удовлетворяет требованиям безопасности.

Таким образом, реализующее множество путей тоже должно быть (приоритетно-) допустимым, т.е. не должно нарушать правило приоритетов, и не иметь циклов, по которым пакеты будут циркулировать бесконечно и бесконечно размножаться. Кроме того, в ряде случаев для сокращения нагрузки на сеть желательно, чтобы множество не содержало дублирующих путей, т.е. разных путей, соединяющих одну и ту же пару хостов. В работе показано, что в отличие от подобной задачи (произвольной нестрогой реализации) для графа без приоритетов, допустимая реализация не всегда возможна, и аналогично строгой реализации не всегда возможна без дублирования и не всегда возможна без циклов.

Сформулировано и доказано требование к графу, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием. С этой целью понятие почти совершенного множества путей из [7] расширяется на граф с приоритетами, и требование формулируется как наличие в графе множества путей, которое соединяет все допустимые пары хостов и в котором пути после слияния (с проходом по одной или нескольким общим дугам) не разделяются. Если требуется отсутствие дублирования, то в достаточном условии после разделения путей запрещается их слияние (с проходом по общей дуге).

## 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. 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 $a_1 \ldots a_n$, 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 (реберно-простым).





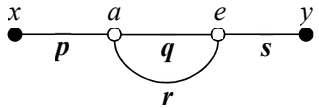
Вершины графа будем обозначать строчными буквами *a, b, c, … x, y, z*, пути – жирными строчными буквами ***p, q, r,***…, а множества путей – прописными буквами – *P, Q, R, …* На рисунках коммутаторам соответствуют белые кружки, а хостам — чёрные кружки.

Мы будем предполагать, что каждый хост x подсоединен ровно к одному коммутатору [3]. Поэтому хост – это терминальная вершина графа, т.е. вершина степени 1. Если коммутатор a имеет степень 1 и соединен с вершиной *b*, то любой полный путь, проходящий через *a*, имеет вид *…bab…*; 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 $\sigma abc$, где $a$ и $c$ соседи $b$, а $\sigma$ вектор значений параметров заголовка пакета, которые используются в правилах. Такое правило означает, что коммутатор $b$, получив пакет с вектором $\sigma$ от соседа $a$, пересылает его соседу $c$. В настоящей работе предполагается, что коммутатор не меняет $\sigma$. Тем самым, для вектора $\sigma$ 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 $a_1$ соединен с коммутатором $a_2$ и хост $a_n$ соединен с коммутатором $a_{n-1}$. Если в коммутаторе есть два правила $\sigma abc$ 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 ***pabqabs***, где $a$ и $b$ коммутаторы. Чтобы понять, каким образом путь с циклами связан с понятиями слияния и разделения по дуге обозначим путь с циклами как ***paqer***(***aqer***)****aqes***, где отрезок ***p*** начинается в хосте $x$, отрезки ***p*** и ***r*** не заканчиваются в одной вершине, после этих отрезков в пути следует коммутатор $a$, отрезок ***aqer*** проходится несколько раз, после коммутатора $e$ отрезки ***r*** и ***s*** не начинаются в одной вершине, и отрезок ***s*** заканчивается в хосте $y$ (см. рис. 1). Двигаясь вдоль пути из хоста $x$ в хост $y$, 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 ***aqer*** $k$ раз, то $(k + 1)$ раз встречается как разделение после слияния, так и слияние после разделения). Пакеты будут не только бесконечно ходить по циклу ***aqer***, но и бесконечно клонироваться в вершине $e$, так что хост $y$ будет получать бесконечное число клонов пакета.




коммутатор $b$ находится на отрезке $(a, e]$, т.е. на $b \in Im(\boldsymbol{q})$ или $b = e$

*Рис. 1. Порождение цикла*
*Fig. 1. Cycle Generation*

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 ***xpabqy*** и ***x`p`abq`y`*** с разными начальными и конечными хостами $x \neq x`$ и $y \neq y`$, то порождаются и новые пути ***xpabq`y`*** и ***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. Если множество $P$ полных реберно-простых путей конечно и замкнуто по дугам, то в нем 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 $D \subseteq H(P)$. Множество $D$ *реализуется без циклов*, если $P$ конечно, и *строго реализуется*, если $D = H(P)$. Множество $D$ *реализуется без дублирования*, если $P$ содержит ровно один $xy$-путь для каждой пары $xy \in D$.

Как отмечается в работе [7], если адрес хоста-отправителя (хоста-получателя) входит в вектор параметров σ, то проблемы со строгой реализацией без циклов и дублирования любого множества пар разных хостов не существует. Действительно, если адрес хоста-отправителя входит в вектор параметров σ, то правила коммутатора 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 множества $D$ пар разных 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. Реализация множества пар разных хостов 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 $x$.

Одна из причин, по которой адрес хоста-отправителя и адрес хоста-получателя не включают в вектор параметров σ и не используют в правилах коммутаторов, заключается в том, что при таком использовании число правил коммутатора зависит от числа хостов в сети (растёт вместе с ним). Поэтому в данной работе мы рассматриваем реализацию множества разных пар хостов для случая, 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.

Политика безопасности (security) может запрещать передавать пакеты от хоста к некоторым другим хостам, а такая передача может случиться, если появляются «непредусмотренные» пути, когда заданное множество пар хостов реализуется не замкнутым по дугам множеством путей. Для описания множества хостов, от которых данный хост может принимать пакеты, каждому хосту $h$ приписан приоритет — натуральное число, обозначаемое $A(h)$. Имеется в виду, что чем больше приоритет, тем больше прав хоста по доступу, т.е. к получению пакетов. Будем считать, что пакет $m$ имеет приоритет $A(m)$, равный приоритету хоста-отправителя $h$: $A(m) = A(h)$, и хост $h`$ может принимать пакет $m$, если $A(m) \le A(h`)$. Граф $G$, хостам которого

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





приписаны приоритеты, будем называть графом с приоритетами и обозначать $G^*$. На рисунках приоритет хоста обозначается числом через запятую после идентификатора хоста.

Полный путь $p$ от хоста $h$ к хосту $h`$ назовём *допустимым*, если $A(h) \le A(h`)$, и *недопустимым* в противном случае. Множество $P$ полных путей *допустимое*, если каждый путь из его замыкания по дугам $P{\downarrow}{\uparrow}$ является допустимым. Замкнутое по дугам множество полных путей допустимое тогда и только тогда, когда оно состоит из допустимых путей. Нормальное множество пар хостов, не нарушающее правило приоритетов, т.е. $\forall\ (x, y) \in D\ \ A(x) \le A(y)$, будем называть *допустимым*. Будем говорить, что допустимое множество $D$ *допустимо реализуется*, если оно может быть реализовано замкнутым по дугам множеством допустимых полных путей $P$.

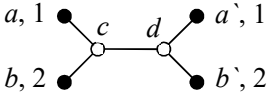
## 3. Допустимая реализация versus циклы и дублирование

В этом разделе мы исследуем связь допустимой реализации множества пар хостов с наличием или отсутствием циклов и дублирования. В статье [7] нами показано, что для любого нормального множества пар хостов существует реализация без циклов и дублирования (утверждение 1), которая возможно не является строгой, однако, как показывает следующее утверждение, это не всегда выполняется, если требовать, чтобы реализация была допустимой.

Утверждение 1. Допустимая реализация не всегда возможна: существует такой граф с приоритетами $G^*$, на котором некоторое допустимое множество пар хостов $D$ может быть реализовано, но не может быть допустимо реализовано.

Доказательство. На рис. 2 показан пример такого графа $G^*$ и множества $D$. Легко убедиться, что $D$ допустимое множество пар хостов. Рассмотрим множество $P = \{\ acda`, bcdb`\ \}$. Легко убедиться, что это множество полных путей и $D \subseteq H(P)$. Следовательно, замыкание по дугам $P{\downarrow}{\uparrow}$ является реализацией $D$. Однако это замыкание содержит путь $bcda`$, который недопустимый, поскольку $A(b) = 2 > 1 = A(a`)$.

Пусть теперь $P$ произвольное замкнутое по дугам множество допустимых полных путей, реализующее множество $D$. Любые два $aa`$-путь и $bb`$-путь имеют общую дугу $cd$, поэтому в множестве $P$ должны оказаться $ba`$-путь, однако этот путь недопустимый. Следовательно, реализация множества $P$ не является допустимой.




$D = \{\ (a, a`), (b, b`)\ \}$

*Рис. 2. Допустимое множество D не имеет допустимой реализации*
*Fig. 2. A permissible set D has no permissible implementation*

□

Пример на рис. 2 показывает также, что строгая реализация (не обязательно допустимая) не всегда возможна: наличие $aa`$-пути и $bb`$-пути в замыкании по дугам порождает $ab`$-путь и $ba`$-путь. Как отмечено выше, если не требовать допустимости (и строгости) реализации, то она всегда возможна.

В работе [7] исследуются также возможности строгой реализации нормального множества пар хостов без циклов и дублирования, и показано, что строгая реализация не всегда возможна без дублирования (утверждение 3) и не всегда возможна без циклов (утверждение 4). В данной работе мы показываем, что при реализации допустимого множества пар хостов свойства из утверждений 3 и 4 [7] сохраняются и для нестрогой, но допустимой, реализации этого множества.

Для полного пути $\boldsymbol{p}$ через $\boldsymbol{p^\circ}$ [7] обозначим путь, который получается из $\boldsymbol{p}$ применением, пока возможно, следующей операции удаления циклов: путь $\boldsymbol{p = qaras}$ превращается в путь $\boldsymbol{qas}$. Заметим, что результат операции "°", вообще говоря, неоднозначный. Для однозначности

будем считать, что удаление цикла применяется только тогда, когда каждая вершина префикса $\boldsymbol{q}$ имеет только одно вхождение в $\boldsymbol{p}$, а $\boldsymbol{r}$ не содержит вхождений вершины $a$. Иными словами, среди всех вершин, имеющих несколько вхождений в $\boldsymbol{p}$, выбирается та вершина $a$, вхождение которой встретилось первым, и удаляется цикл от этого первого вхождения $a$ до ее второго вхождения. Например, для $\boldsymbol{p} = xacbcaby$, $\boldsymbol{p}^\circ = xaby$ (не $xacby$). Для множества полных путей $P$ обозначим через $P^\circ$ множество путей, получаемых удалением циклов из всех путей $P$, т.е. $P^\circ = \{ \boldsymbol{p}^\circ \mid \boldsymbol{p} \in P \}$.

Утверждение 2. Пусть $P$ множество полных путей. Тогда множество $P^\circ$ состоит из вершинно-простых полных путей и связывает те же самые пары хостов, что множество $P$: $H(P^\circ) = H(P)$. Если $P$ замкнуто по дугам, то $P^\circ$ тоже замкнуто по дугам. Если $P$ не содержит дублирующих путей, то $P^\circ$ тоже не содержит дублирующих путей. Если $P$ конечно, то $P^\circ$ тоже конечно. Если $P$ допустимое, то $P^\circ$ тоже допустимое.

Доказательство. Очевидно, что удаление всех циклов из полного пути делает путь вершинно-простым и оставляет его полным. Поэтому $P^\circ$ состоит из вершинно-простых полных путей. Очевидно, что операция удаления одного цикла из одного пути не меняет $H(P)$. Следовательно, цепочка таких операций также не меняет $H(P)$. Следовательно, $P^\circ$ связывает те же самые пары хостов, что множество $P$: $H(P^\circ) = H(P)$.

Докажем, что если $P$ замкнуто по дугам, то $P^\circ$ тоже замкнуто по дугам. Пусть $P^\circ$ содержит пути $\boldsymbol{p}ab\boldsymbol{q}$ и $\boldsymbol{p}_1ab\boldsymbol{q}_1$ с общей дугой $ab$. Эти пути получены удалением циклов из путей $\boldsymbol{r}$ и $\boldsymbol{r}_1$, соответственно, имеющихся в $P$. При удалении циклов непустая последовательность вершин между вхождением $a$ и вхождением $b$ может быть полностью удалена только вместе с удалением вхождения $a$ и/или $b$. Поэтому дуга $ab$ может остаться в результате удаления циклов только в том случае, если она была в исходном пути и некоторое ее вхождение не лежало на удаляемых циклах. Тогда это вхождение дуги $ab$ разбивает путь на три отрезка: отрезок до $a$, данное вхождение дуги $ab$ и отрезок после $b$, причем циклы не содержат данное вхождение дуги $ab$, т.е. любой цикл целиком лежит либо на отрезке до $a$, либо на отрезке после $b$. Следовательно, пути $\boldsymbol{r}$ и $\boldsymbol{r}_1$ можно представить в виде $\boldsymbol{p}`ab\boldsymbol{q}`$ и $\boldsymbol{p}`_1ab\boldsymbol{q}`_1$, соответственно, где $\boldsymbol{p} = \boldsymbol{p}`^\circ$, $\boldsymbol{p}_1 = \boldsymbol{p}`_1{}^\circ$, $\boldsymbol{q} = \boldsymbol{q}`^\circ$, $\boldsymbol{q}_1 = \boldsymbol{q}`_1{}^\circ$. Поскольку $P$ замкнуто по дугам, в нем имеются пути $\boldsymbol{p}`ab\boldsymbol{q}`_1$ и $\boldsymbol{p}`_1ab\boldsymbol{q}`$. А тогда в $P^\circ$ имеются пути $\boldsymbol{p}ab\boldsymbol{q}_1 = \boldsymbol{p}`^\circ ab\boldsymbol{q}_1`^\circ = (\boldsymbol{p}`ab\boldsymbol{q}_1`)^\circ$ и $\boldsymbol{p}_1abq = \boldsymbol{p}`^1{}^\circ ab\boldsymbol{q}`^\circ = (\boldsymbol{p}`_1ab\boldsymbol{q}`)^\circ$. Следовательно, множество $P^\circ$ замкнуто по дугам.

Результатом операции "$^\circ$", примененной к одному пути, является один путь с теми же начальным и конечным хостом. Отсюда непосредственно следует: если $P$ не содержит дублирующих путей, то $P^\circ$ тоже не содержит дублирующих путей; если $P$ конечно, то $P^\circ$ тоже конечно; если $P$ допустимое, то $P^\circ$ тоже допустимое.

□

Данное выше определение множества $P^\circ$ то же самое, что в [7], а доказанное выше утверждение 2 дополняет утверждение 2 из [7].

В следующих утверждениях мы показываем, что в отличие от произвольной нестрогой реализации нормального множества пар хостов, допустимая реализация допустимого множества пар в некоторых случаях невозможна как без циклов, так и без дублирования.

Утверждение 3. Допустимая реализация не всегда возможна без дублирования: существует такой граф с приоритетами $G^*$, на котором некоторое допустимое множество $D$ пар хостов может быть допустимо реализовано, но только с дублированием.

Доказательство. На рис. 3 показан пример графа $G^*$ и множества $D$. Легко убедиться, что $D$ – допустимое множество пар хостов. Рассмотрим множество $P = \{ x_0a_0a_1b_1b_4a_4b_0y_0, x_0a_0b_3a_3a_2b_2b_0y_0, x_0a_0a_1b_1y_1, x_0a_0b_3a_3a_2b_2y_2, x_1a_1b_1b_4a_4b_0y_0, x_1a_1b_1y_1, x_2a_2b_2b_0y_0, x_2a_2b_2y_2, x_3a_2b_3y_3, x_4a_4b_4y_4 \}$. Легко убедиться, что множество $P$ состоит из допустимых полных путей, замкнуто по дугам и $D \subseteq H(P)$ (и $D = H(P)$). Следовательно, множество $P$ допустимо (и строго)





реализует множество $D$. Однако множество $P$ содержит дублирующие пути $x_0a_0a_1b_1b_4a_4b_0y_0$ и $x_0a_0b_3a_3a_2b_2b_0y_0$.

Пусть теперь $P`$ некоторая допустимая реализация множества $D$, т.е. $P`$ есть замкнутое по дугам множество, такое, что все пути в $P`$ допустимые, и $H(P`) \supseteq D$. Согласно утверждению 2 мы можем считать, что $P`$ состоит из вершинно-простых путей. Допустим, что $P`$ не содержит дублирующих путей. В графе $G$ коммутаторы образуют цикл $a_0b_3a_3a_2b_2b_0a_4b_4b_1a_1a_0$, и любой вершинно-простой путь может проходить только часть этого цикла по часовой или против часовой стрелки.

Покажем, что $x_4y_4$-путь из $P`$ должен проходить цикл коммутаторов по часовой стрелке, т.е. это должен быть путь $x_4a_4b_4y_4$. Допустим обратное: $x_4y_4$-путь из $P`$ проходит цикл коммутаторов против часовой стрелки, т.е. это путь $x_4a_4b_0b_2a_2a_3b_3a_0a_1b_1b_4y_4$. Если хотя бы один $x_iy_j$-путь из $P`$, где $(x_i, y_j) \in D \setminus \{(x_4, y_4)\}$, проходит цикл коммутаторов тоже против часовой стрелки, то в замыкании по дугам появляется $x_4y_j$-путь, где $j \neq 4$. А поскольку $P`$ замкнуто по дугам, этот путь есть и в $P`$, но такой путь недопустимый, поскольку хост $x_4$ имеет приоритет 4, который больше чем приоритет любого хоста $y_j$, где $j \neq 4$. Следовательно, все $x_iy_j$-пути из $P`$, где $(x_i, y_j) \in D \setminus \{(x_4, y_4)\}$, проходят цикл коммутаторов по часовой стрелке. Среди этих путей есть пути $x_1a_1a_0b_3a_3a_2b_2b_0y_0$ и $x_2a_2b_2y_2$, которые в замыкании по дугам порождают путь $x_1a_1a_0b_3a_3a_2b_2y_2$, который в силу замкнутости по дугам множества $P`$, должен быть в $P`$, но этот путь недопустимый, так как $A(x_1) = 2 > 1 = A(y_2)$. Мы пришли к противоречию, следовательно, $x_4y_4$-путь из $P`$ проходит цикл коммутаторов по часовой стрелке, т.е. это путь $x_4a_4b_4y_4$.

Покажем, что $x_3y_3$-путь из $P`$ должен проходить цикл коммутаторов против часовой стрелки, т.е. это должен быть путь $x_3a_3b_3y_3$. Действительно, в противном случае в $P`$ есть пути $x_3a_3a_2b_2b_0a_4b_4b_1a_1a_0b_3y_3$ и $x_4a_4b_4y_4$, которые в замыкании по дугам порождают путь $x_4a_4b_4b_1a_1a_0b_3y_3$, который в силу замкнутости по дугам множества $P`$, должен быть в $P`$, но этот путь недопустимый, так как $A(x_4) = 4 > 3 = A(y_3)$. Мы пришли к противоречию, следовательно, $x_3y_3$-путь из $P`$ проходит цикл коммутаторов против часовой стрелки, т.е. это путь $x_3a_3b_3y_3$.

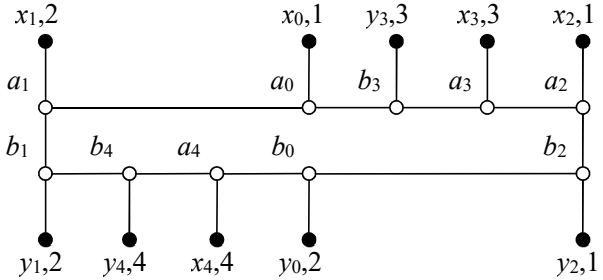
Покажем, что в $P`$ нет $x_2y_1$-пути. Допустим противное, и тогда этот путь проходит цикл коммутаторов либо по часовой стрелке, либо против часовой стрелки. В первом случае это путь $x_2a_2b_2b_0a_4b_4b_1y_1$, который вместе с путем $x_4a_4b_4y_4$ порождает в замыкании по дугам $x_4y_1$-путь $x_4a_4b_4b_1y_1$, который в силу замкнутости по дугам множества $P`$ должен быть в $P`$, но этот путь недопустимый, так как $A(x_4) = 4 > 2 = A(y_1)$. Во втором случае это путь $x_2a_2a_3b_3a_0a_1b_1y_1$, который вместе с путем $x_3a_3b_3y_3$ порождает в замыкании по дугам $x_3y_1$-путь $x_3a_3b_3a_0a_1b_1y_1$, который в силу замкнутости по дугам множества $P`$ должен быть в $P`$, но этот путь недопустимый, так как $A(x_3) = 3 > 2 = A(y_1)$. Мы пришли к противоречию, следовательно, наше допущение не верно, и в $P`$ нет $x_2y_1$-пути.

Также в $P`$ нет $x_1y_2$-пути, поскольку этот путь недопустимый: $A(x_1) = 2 > 1 = A(y_2)$.

Рассмотрим пути из $P`$, которые начинаются в хостах $x_0$, $x_1$, $x_2$ и заканчиваются в хостах $y_0$, $y_1$, $y_2$. Поскольку в $P`$ нет дублирующих путей, и мы показали, что в $P`$ нет $x_2y_1$-пути и $x_1y_2$-пути, то из 9 возможных путей в $P`$ может быть только 7 путей. И эти 7 путей должны быть в $P`$, поскольку в множестве $D$ есть соответствующие пары хостов.

Для того чтобы попасть из хоста $x_i$ в хост $y_j$, где $i = 0, 1, 2$ и $j = 0, 1, 2$, путь должен пройти либо по дуге $a_1b_1$, либо по дуге $a_2b_2$. Пусть для $t = 1, 2$ имеется $m_t$ путей, которые проходят дугу $a_tb_t$, и эти пути начинаются в $n_t$ хостах и заканчиваются в $k_t$ хостах. Тогда $n_1 + n_2 = 3$, $k_1 + k_2 = 3$, $m_1 + m_2 = 7$. Из замкнутости по дугам множества $P`$ следует $n_1k_1 = m_1$, $n_2k_2 = m_2$. Отсюда $n_1k_1 + (3 - n_1)(3 - k_1) = 7$, что влечет $2n_1k_1 = 3(n_1 + k_1) - 2$. Поскольку $k_1 \leq 3$ и $n_1 \leq 3$, имеем: $3k_1 = 2$ для $n_1 = 0$, $k_1 = -1$ для $n_1 = 1$, $k_1 = 4$ для $n_1 = 2$, $3k_1 = 7$ для $n_1 = 3$. Каждое из этих уравнений не имеет решения в целых неотрицательных числах или противоречит условию $k_1 \leq 3$.

Мы пришли к противоречию, следовательно, наше допущение не верно, и в $P$` есть дублирующие пути.




$D = \{ (x_0, y_0), (x_0, y_1), (x_0, y_2), (x_1, y_0), (x_1, y_1), (x_2, y_0), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$

*Рис. 3. Множество D допустимо реализуется только с дублированием*
*Fig. 3. The set D is permissible only with duplication*

Отметим, что в соответствующем утверждении 3 из [7] доказано аналогичное утверждение для строгой реализации, при условии, что передача пакетов возможна для любой пары различных хостов: строгая реализация не всегда возможна без дублирования. Пример на рис. 3 является усложнённым вариантом примера на рис. 1 из [7]. В обоих случаях во множестве $D$ содержится 7 из 9 пар хостов вида $(x_i, y_j)$, где $i = 0, 1, 2$ и $j = 0, 1, 2$, и рассматриваются только вершинно-простые пути. Но в [7] число 7 путей, связывающих эти хосты, непосредственно следует из строгости реализации, а в этой статье для нестрогой, но допустимой, реализации нам потребовалось запретить 2 из 9 возможных путей. Один $x_1y_2$-путь мы запретили соответствующей расстановкой приоритетов, а другой $x_2y_1$-путь мы запретили с помощью введения четырех дополнительных хостов $x_3$, $y_3$, $x_4$, $y_4$ с подходящими приоритетами, четырех коммутаторов $a_3$, $b_3$, $a_4$, $b_4$ и двух пар добавленных хостов $(x_3, y_3)$ и $(x_4, y_4)$.

Утверждение 4. Допустимая реализация не всегда возможна без циклов: существует такой граф с приоритетами $G^*$, на котором некоторое допустимое множество $D$ пар хостов допустимо реализуемо, но только бесконечными замкнутыми множествами путей.

Доказательство. Рассмотрим пример на рис. 4. Множество $D$ допустимое и допустимо (и строго) реализуется замыканием по дугам $P{\downarrow}{\uparrow}$ множества полных путей $P$. Это замыкание допустимое, поскольку его пути соединяют только хосты с равными приоритетами. Но в $P$ есть пути $x_1a_1b_1c_2c_1a_2b_2y_2$ и $x_2a_2b_2d_2d_1a_1b_1y_1$, которые в $P{\downarrow}{\uparrow}$ порождают не реберно-простой путь $x_1a_1b_1c_2c_1a_2b_2d_2d_1a_1b_1y_1$ (проходит дважды по дуге $a_1b_1$), т.е. $P{\downarrow}{\uparrow}$ бесконечно. Пусть замкнутое по дугам множество допустимых путей $P$` реализует множество $D$. Допустим, оно конечно. Тогда по утверждению 2 мы можем выбрать множество $P$`, состоящее из вершинно-простых путей. Коммутаторы образуют цикл $a_1b_1c_2c_1a_2b_2d_2d_1a_1$, вершинно-простой путь может проходить только часть этого цикла по часовой или против часовой стрелки.

1. Пусть $u_1u_2$-путь $\boldsymbol{p}_1$ идет по часовой стрелке, тогда $\boldsymbol{p}_1 = u_1d_1a_1b_1c_2c_1a_2b_2d_2u_2$.

1.1. Если $x_2y_2$-путь $\boldsymbol{p}_2$ идет по часовой стрелке, тогда $\boldsymbol{p}_2 = x_2a_2b_2y_2$; $u_1u_2$-путь $\boldsymbol{p}_1$ и $x_2y_2$-путь $\boldsymbol{p}_2$ имеют общую дугу $a_2b_2$, что в замыкании по дугам порождает $u_1y_2$-путь, но этот путь недопустимый: $A(u_1) = 2 > 1 = A(y_2)$. Значит $\boldsymbol{p}_2$ идет против часовой стрелки, тогда $\boldsymbol{p}_2 = x_2a_2c_1c_2b_1a_1d_1d_2b_2y_2$.

1.2. Если $x_1y_1$-путь $\boldsymbol{p}_3$ идет по часовой стрелке, тогда $\boldsymbol{p}_3 = x_1a_1b_1y_1$; $u_1u_2$-путь $\boldsymbol{p}_1$ и $x_1y_1$-путь $\boldsymbol{p}_3$ имеют общую дугу $a_1b_1$, что в замыкании по дугам порождает $u_1y_1$-путь, но этот путь





недопустимый: $A(u_1) = 2 > 1 = A(y_1)$. Значит $\boldsymbol{p}_3$ идет против часовой стрелки, тогда $\boldsymbol{p}_3 = x_1a_1d_1d_2b_2a_2c_1c_2b_1y_1$.

1.3. Из 1.1 и 1.2 следует, что $x_2y_2$-путь $\boldsymbol{p}_2$ и $x_1y_1$-путь $\boldsymbol{p}_3$ имеют общую дугу $d_2b_2$, что в замыкании по дугам порождает путь $x_2a_2c_1c_2b_1a_1d_1d_2b_2a_2c_1c_2b_1y_1$, который не является реберно-простым, так как дважды проходит по дуге $a_2c_1$, т.е. порождается цикл и, следовательно, бесконечное множество путей, что противоречит конечности множества $P$`.

Следовательно, $u_1u_2$-путь $\boldsymbol{p}_1$ идет против часовой стрелки, т.е. $\boldsymbol{p}_1 = u_1d_1d_2u_2$.

2. В силу симметрии аналогично доказывается (рассматривая $v_1v_2$-путь, $x_1y_1$-путь и $x_2y_2$-путь), что $v_1v_2$-путь $\boldsymbol{p}_4$ идет против часовой стрелки, т.е. $\boldsymbol{p}_4 = v_1c_1c_2v_2$.

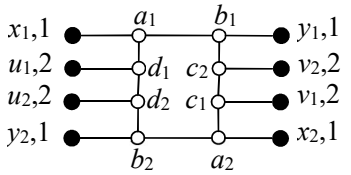
3. Если $x_1y_2$-путь $\boldsymbol{p}_5$ идет против часовой стрелки, тогда $\boldsymbol{p}_5 = x_1a_1d_1d_2b_2y_2$; $u_1u_2$-путь $\boldsymbol{p}_1$ и $x_1y_2$-путь $\boldsymbol{p}_5$ имеют общую дугу $d_1d_2$, что в замыкании по дугам порождает $u_1y_2$-путь, но этот путь недопустимый: $A(u_1) = 2 > 1 = A(y_2)$.

Следовательно, $\boldsymbol{p}_5$ идет по часовой стрелке, т.е. $\boldsymbol{p}_5 = x_1a_1b_1c_2c_1a_2b_2y_2$.

4. В силу симметрии аналогично доказывается (рассматривая $x_2y_1$-путь и $v_1v_2$-путь), что $x_2y_1$-путь $\boldsymbol{p}_6$ идет по часовой стрелке, тогда $\boldsymbol{p}_6 = x_2a_2b_2d_2d_1a_1b_1y_1$.

Но тогда пути $\boldsymbol{p}_5$ и $\boldsymbol{p}_6$ имеют общую дугу $a_2b_2$ и в замыкании по дугам порождают путь $x_1a_1b_1c_2c_1a_2b_2d_2d_1a_1b_1y_1$, который не является реберно-простым, так как дважды проходит по дуге $a_1b_1$, т.е. порождается цикл и, следовательно, бесконечное множество путей, что противоречит конечности множества $P$`.

Мы пришли к противоречию и, следовательно, наше допущение не верно, и $P$` бесконечно.




$D = \{ x_1y_1, x_1y_2, x_2y_2, x_2y_1, u_1u_2, v_1v_2 \}$
$P = \{ x_1a_1b_1y_1, x_1a_1b_1c_2c_1a_2b_2y_2,$
$x_2a_2b_2y_2, x_2a_2b_2d_2d_1a_1b_1y_1,$
$u_1d_1d_2u_2, v_1c_1c_2v_2 \}$

*Рис. 4. Множество D допустимо реализуется только бесконечным множеством путей*
*Fig. 4. The set D may be permissible implemented only by an infinite set of paths*

Отметим, что в соответствующем утверждении 4 из [7] доказано аналогичное утверждение для строгой реализации, при условии, что все хосты имеют одинаковый приоритет: строгая реализация не всегда возможна без циклов. Пример на рис. 4 аналогичен примеру на рис. 2 из [7]. Заметим, что в публикации [7] допущены ошибки в рис. 4 и доказательстве утверждения 4. Правильный рисунок – это рис. 4 в данной статье. Доказательство аналогично доказательству утверждения 4 в данной статье, но только игнорируются приоритеты, и вместо фразы «этот путь недопустимый» следует читать «этот путь соединяет пары хостов, отсутствующие в заданном множестве $D$».

В обоих случаях (в данной статье и в исправленной статье [7]) пути из $u_1$ и $v_1$ не должны заканчиваться в $y_1$ и $y_2$. Но в [7] это следует из строгости реализации, а в данной статье для нестрогой, но допустимой, реализации это следует из правила приоритетов: пакет не должен пересылаться из хоста с большим приоритетом в хост с меньшим приоритетом.

Аналогичный пример демонстрирует «бóльшую свободу» допустимой нестрогой реализации. Для этого достаточно назначить хостам $y_1$ и $y_2$ приоритет 3. Тогда становится возможной допустимая реализация без циклов и дублирования, например, на основе множества кратчайших путей, соединяющих пары хостов из множества $D$: $\{ x_1a_1b_1y_1, x_1a_1d_1d_2b_2y_2, x_2a_2c_1c_2b_1y_1, x_2a_2b_2y_2, u_1d_1d_2u_2, v_1c_1c_2v_2 \}$. Замыкание по дугам этого множества конечно (не порождает циклов), не содержит дублирующих путей, но содержит два дополнительных пути $u_1d_1d_2b_2y_2$ и $v_1c_1c_2b_1y_1$, связывающих пары хостов $(u_1, y_2)$ и $(v_1, y_1)$,

которые нельзя связывать при строгой реализации, но можно связывать при допустимой реализации.

## 4. Достаточное условие допустимой реализации любого множества пар хостов без циклов и дублирования

В этом разделе мы исследуем достаточные условия на граф с приоритетами, позволяющие допустимо реализовывать без циклов и дублирования любые допустимые множества пар хостов.

В [7] введены понятия *разделения после слияния* и *слияния после разделения*. Если для двух путей есть слияние на дуге *ab* и есть разделение после дуги *cd*, то будем говорить, что разделение происходит после слияния, если хотя бы в одном из этих путей сначала проходится дуга *ab*, а потом дуга *cd*. Соответственно, слияние происходит после разделения, если хотя бы в одном из этих путей сначала проходится дуга *cd*, а потом дуга *ab*.

Также в [7] доказаны следующие утверждения:

Утверждение 5. Если множество полных путей конечно, то отсутствие разделения после слияния достаточно, но не необходимо для замкнутости по дугам и отсутствия циклов.

Утверждение 6. Для замкнутого множества полных путей условие отсутствия слияния после разделения необходимо и достаточно для отсутствия дублирования.

Также в [7] даны определения (почти) хорошего графа, (почти) совершенного множества путей и (почти) совершенного графа. Для допустимой, но возможно нестрогой, реализации мы предлагаем следующие аналогичные определения.

Граф, в котором любое допустимое множество пар хостов можно допустимо реализовать без циклов, назовём *почти допустимо-хорошим*. Граф, в котором любое допустимое множество пар хостов можно допустимо реализовать без циклов и без дублирования, назовём *допустимо-хорошим*.

Конечное замкнутое множество $P$ допустимых путей, связывающее все пары хостов, не нарушающие правило приоритетов, т.е. все пары хостов $(x, y)$, где $x \neq y$ и $A(x) \leq A(y)$, назовем *почти допустимо-совершенным*, если в нем нет разделения после слияния путей, и *допустимо-совершенным*, если в нем, кроме того, нет слияния после разделения путей. Граф будем называть *почти допустимо-совершенным* или *допустимо-совершенным*, если в нем есть, соответственно, почти допустимо-совершенное или допустимо-совершенное множество путей.

Достаточное условие допустимой реализации любого допустимого множества пар хостов без циклов и дублирования теперь можно сформулировать в виде следующего утверждения. Оно является аналогом утверждения 7 из [7] для строгой реализации.

Утверждение 7. Почти допустимо-совершенный граф является почти допустимо-хорошим, а допустимо-совершенный граф является допустимо-хорошим. При этом почти допустимо-совершенное множество путей для каждого допустимого множества пар хостов содержит его допустимую реализацию без циклов как подмножество, а допустимо-совершенное множество путей для каждого допустимого множества пар хостов содержит его допустимую реализацию без циклов и без дублирования как подмножество.

Доказательство. Поскольку почти допустимо-совершенное множество путей конечно и в нем нет разделения после слияния, любое его подмножество также конечно и в нем нет разделения после слияния, поэтому, по утверждению 5, это подмножество замкнуто по дугам и не порождает циклов. По определению допустимо-совершенное множество является почти допустимо-совершенным, поэтому любое его подмножество также конечно, замкнуто по дугам и не порождает циклов. Поскольку в допустимо-совершенном множестве путей нет слияния после разделения, в любом его подмножестве также нет слияния после разделения, то, по утверждению 6, это подмножество не порождает дублирование. Для любого





допустимого множества $D$ пар хостов в (почти) допустимо-совершенном множестве $P$ путей можно выбрать подмножество $P(D)$ такое, что $H(P(D)) = D$. Множество $P(D)$ путей допустимо реализует множество $D$ пар хостов без циклов и, если множество $P$ допустимо-совершенное, без дублирования.

□

В [7] была высказана гипотеза о том, что существование наибольшего совершенного множества путей является не только достаточным, но и необходимым условием строгой реализации любого нормального множества пар хостов без циклов и дублирования. Однако в [8] компьютерные эксперименты показали, что это не так. Поэтому существование наибольшего допустимо-совершенного множества путей не необходимо для допустимой реализации любого допустимого множества пар хостов без циклов и дублирования, по крайней мере, для этого случая равноприоритетных хостов.

## 5. Заключение

В статье рассматривается реализация множества пар разных хостов с помощью путей в графе физических связей и соответствующей настройки коммутаторов, при условии, что определены пары хостов, между которыми можно передавать пакеты с заданным приоритетом. Соответственно рассматриваются допустимые множества пар разных хостов, т.е. удовлетворяющих правилу приоритетов: пакеты, посланные из хоста с данным приоритетом, могут получать только хосты с тем же или бо̀льшим приоритетом. Реализующее множество путей тоже должно быть допустимым, т.е. не должно нарушать правило приоритетов, и не иметь циклов, по которым пакеты будут циркулировать бесконечно и бесконечно размножаться. Кроме того, в ряде случаев для сокращения нагрузки на сеть желательно, чтобы множество не содержало дублирующих путей, т.е. нескольких путей, соединяющих одну и ту же пару хостов. В работе показано, что в отличие от подобной задачи для нестрогой реализации на графе без приоритетов, допустимая реализация не всегда возможна, и тем более, не всегда возможна без дублирования и не всегда возможна без циклов.

Сформулировано и доказано требование к графу, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием. Это требование формулируется как наличие в графе наибольшего почти допустимо-совершенного множества путей, которое соединяет все допустимые пары хостов и в котором пути не разделяются после слияния. Если требуется отсутствие дублирования, то в достаточном условии запрещается слияние путей после их разделения.

Дальнейшие исследования могут развиваться в двух направлениях. Одно направление – это исследование свойств графа, которые позволяют или не позволяют иметь в нём наибольшее (почти) допустимо-совершенное множество путей. Другое направление – это обобщение идеи приоритетов, когда некоторые веса приписывается не только хостам (в данной статье приоритеты), но и другим коммуникационным элементам: коммутаторам, дугам графа и парам смежных дуг, т.е. правилам коммутатора, с различными целями и по различным правилам.

## Список литературы / References

[1]. S. Sezer, S. Scott-Hayward, P.K. Chouhan et al. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, vol. 51, no. 7, 2013, pp. 36-43.

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

[3]. N. Yevtushenko, A. Kossatchev, J. Lopez et al. 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 1-5.
[4]. И.Б. Бурдонов, Н.В. Евтушенко, А.С. Косачев. Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 69-88 / I.B. Burdonov, N.V. Yevtushenko, A.S. Kossatchev. Testing switch rules in software defined networks. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 69-88 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-4.
[5]. I. Burdonov, A. Kossatchev, N. Yevtushenko et al. Verifying SDN Data Path Requests. arXiv:1906.03101, 2019.
[6]. Y. Boufkhad, R. De La Paz, L. Linguaglossa et al. Forwarding tables verification through representative header sets. arXiv:1601.07002, 2016.
[7]. I. Burdonov, N. Yevtushenko, and A. Kossatchev. Implementing a virtual network on the SDN data plane. In Proc. of the IEEE East-West Design and Test Symposium, 2020, pp 1-5.
[8]. И.Б. Бурдонов, Е.М. Винарский, Н.В. Евтушенко, А.С. Косачев. Совершенные множества путей в полном графе коммутаторов SDN-сети. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 243–258 / I.B. Burdonov, E.M. Vinarskii, N.V. Yevtushenko, A.S. Kossatchev. Perfect sets of paths in the full graph of SDN network switches. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 243–258 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–18.

## 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.

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.