DOI: 10.15514/ISPRAS-2022-34(3)-11

# Реализация распределённых и параллельных вычислений в сети SDN

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

*Институт системного программирования им. В.П. Иванникова РАН,*
*109004, Россия, г. Москва, ул. А. Солженицына, д. 25*

**Аннотация.** В статье рассматривается выполнение на плоскости данных SDN, моделируемой конечным связным неориентированным графом физических связей, программы задания, которая понимается в духе парадигмы объектно-ориентированного программирования как состоящая из объектов и сообщений, которыми объекты могут обмениваться. Объекты реализуется в хостах, причём в одном хосте может быть реализовано несколько разных объектов, а один и тот же объект может быть реализован в нескольких хостах. Сообщения между объектами, реализованными в разных хостах, помещаются в пакеты, маршрутизацию которых исполняют коммутаторы на основе идентификаторов, присвоенных пакетам и помещаемых в заголовки пакетов как набор значений некоторых параметров пакетов. В работе решаются две задачи: 1) минимизация числа идентификаторов, 2) настройка коммутаторов для реализации путей, которые должны проходить пакеты. Эти задачи решаются в двух случаях: A) пакет, предназначенный для некоторого объекта, должен попасть ровно в один хост, в котором реализован этот объект, B) пакет может попадать в несколько хостов, но в одном и только одном из них должен быть реализован нужный объект. Показано, что задача 1 в случае A эквивалентна задаче о покрытии множества, а минимальное число идентификаторов в наихудшем случае равно $min\{ n, m \}$, где n число объектов, а m число хостов, реализующих объекты. В случае B задача является специальной модификацией задачи о покрытии множества, высказывается гипотеза о том, что минимальное число идентификаторов в наихудшем случае равно $min\{ \lfloor lb(n+1) \rfloor, m \}$. Пока получена верхняя оценка $O( min \{ ln (min \{ n, m \}) \cdot ln ( n, m ) \} )$. Для решения задачи 2 в случаях A и B предложены алгоритмы настройки коммутаторов сложности, соответственно, $O( m )$ и $O( k m )$, где $m$ число рёбер графа физических связей, а $k$ результат решения задачи 1 в случае B как число требуемых идентификаторов пакетов.

**Ключевые слова:** распределенные и параллельные вычисления, программно-конфигурируемые сети, маршрутизация пакетов, задача о покрытии множества

**Для цитирования:** Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Реализация распределённых и параллельных вычислений в сети SDN. Труды ИСП РАН, том 34, вып. 3, 2022 г., стр. 159-172. DOI: 10.15514/ISPRAS–2022–34(3)–11

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





# Implementation of distributed and parallel computing in the SDN network

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

*Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
*25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*

**Abstract.** The paper discusses the execution of a program of tasks on the SDN data plane, modeled by a finite connected undirected graph of physical connections; the execution is understood in the sense of the object-oriented programming paradigm as consisting of objects and messages that objects can exchange. Objects are implemented in hosts. Several different objects can be implemented in one host and the same object can be implemented in several hosts. Messages between objects implemented in different hosts are transmitted in packets which are routed by switches based on identifiers assigned to packets that is on a set of values of some packet parameters in the packet header. Two problems are tackled in the work: 1) minimizing the number of identifiers, 2) setting up switches to implement the paths that packets should take place. These tasks are solved in two cases: A) a packet intended for some object must get into exactly one host in which this object is implemented, B) a packet can get into several hosts, but the desired object must be implemented in one and only one of them. It is shown that problem 1 in case A is equivalent to the set covering problem, and the minimum number of identifiers in the worst case is $min\{ n, m \}$ where $n$ is the number of objects, and $m$ is the number of hosts implementing objects. In case B, the problem is a special modification of the set covering problem, the hypothesis is proposed that the minimum number of identifiers in the worst case is $min\{ \lfloor lb(n+1) \rfloor, m \}$. So far, an upper bound is $O( min \{ ln (min \{ n, m \}) \cdot ln ( n, m ) \} )$. To solve problem 2 in cases A and B, algorithms for switches' setting are proposed which have the complexity$O( m )$ and $O( k m )$, , respectively, where $m$ is the number of the edges of the graph of physical connections and $k$ is the number of the required packet identifiers.

**Keywords:** distributed and parallel computations, software-defined networks (SDN), packet routing, set cover problem

**For citation:** Burdonov I.B., Yevtushenko N.V., Kossatchev A.S. Implementation of distributed and parallel computing in the SDN network. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 3, 2022, pp. 159-172 (in Russian). DOI: 10.15514/ISPRAS–2022–34(3)–11

**Acknowledgements.** The work was supported by the Russian Foundation for Basic Research, project 20-07-00338 A.

## *1. Введение*

Рассматривается плоскость данных SDN, моделируемая конечным связным неориентированным графом $G$, терминальные вершины (вершины степени 1) которого – *хосты*, а внутренние вершины (вершины степени больше 1) – *коммутаторы*. Этот граф называется графом физических связей. *Пакеты* генерируются хостами и двигаются по путям, определяемым правилами коммутаторов. Коммутаторы не меняют пакеты и только пересылают принятые ими пакеты. Пакет должен пройти путь от хоста до хоста, в котором все промежуточные вершины коммутаторы. Такой путь называется *полным*. Коммутаторы различают пакеты по их *идентификаторам*. Идентификатор моделирует набор значений параметров пакета в заголовке пакета. Коммутатор $s$, приняв пакет с идентификаторов $d$ от соседа $a$, посылает пакет соседу $b$ в зависимости от $d$ и $a$, что определяется *правилом пакета* вида $( d, a, s, b )$. Пакет может быть послан нескольким соседям (клонирование пакета), если в таблице коммутатора $s$ есть несколько правил, отличающихся только принимающим узлом $b$.

В данной работе рассматривается выполнение на плоскости данных SDN *задания*, которое задаётся 1) *начальным хостом*, инициирующим выполнение задания, 2) *описанием*

*программы*, 3) *набором параметров* выполнения задания. *Программа* понимается в духе парадигмы объектно-ориентированного программирования как состоящая из *объектов* и *сообщений*, которыми объекты могут обмениваться. Объект реализуется в хосте, причём в одном хосте может быть реализовано несколько разных объектов, а один и тот же объект может быть реализован в нескольких хостах (в этом случае говорят о нескольких *экземплярах* одного объекта). Если сообщение передаётся от объекта $a$ к объекту $b$, и эти объекты реализованы в одном хосте, то передача сообщения – это просто вызов объекта $b$ из объекта $a$ внутри этого хоста. Если же объекты $a$ и $b$ реализованы в разных хостах $h_a$ и $h_b$, соответственно, то сообщение передаётся в пакете, направляемом по плоскости данных SDN из хоста $h_a$ в хост $h_b$. Параллелизм обеспечивается передачей нескольких сообщений от одного объекта $a$ нескольким объектам $b_1, ..., b_k$, которые могут быть реализованы в нескольких хостах. Выполнение объекта $b$ может зависеть от результатов выполнения нескольких объектов $a_1, ..., a_n$. Это означает, что выполнение объекта $b$ начинается после получения им сообщений от объектов $a_1, ..., a_n$, содержащих аргументы вызова объекта $a$.

Выполнение задания начинается в начальном хосте, который инициирует выполнение задания рассылкой сообщений тем объектам, с которых должно начинаться выполнение задания согласно описанию программы, с теми параметрами, которые определяются набором параметров задания.

Если объекты $a$ и $b$ реализованы в разных хостах $h_a$ и $h_b$, соответственно, сообщение от $a$ к $b$ передаётся в пакете, который содержит 1) идентификатор пакета, помещаемый хостом $h_a$ в заголовок пакета как значения некоторых его параметров и предназначенный для маршрутизации пакетов через коммутаторы, 2) идентификатор объекта $b$, которому направляется сообщение, 3) само сообщение для объекта $b$, 4) описание программы или части программы, которая ещё не выполнена. Хост $h_b$, получив пакет с сообщением для объекта $b$, вызывает этот объект и, при необходимости, формирует пакеты для передачи сообщений объектам, вызываемым из объекта $b$ и реализованным в других хостах.

Мы предполагаем, что идентификатор пакета формируется тем хостом, который этот пакет отправляет, и может отличаться от идентификаторов принятых им пакетов.

Мы будем считать, что распределение объектов по хостам может быть любым, но оно задано, а начальным хостом может быть любой хост. При этом мы предполагаем, что каждый объект, который может вызываться в программе задания, реализован хотя бы в одном хосте.

Относительно пакета с сообщением для объекта $b$ возникают вопросы о допустимости следующих ситуаций.

1) Пакет не попадает ни в один из хостов, в которых реализован объект $b$.
2) Пакет в результате клонирования в коммутаторах попадает в несколько хостов, в которых реализован объект $b$.
3) Пакет в результате клонирования в коммутаторах проходит несколько дублирующих путей, заканчивающихся в одном хосте, в котором реализован объект $b$.
4) Пакет попадает в хост, в котором не реализован объект $b$.

Мы считаем, что настройка коммутаторов и правила генерации идентификаторов пакетов должны исключать ситуации 1 (объект $b$ не вызывается) и 2 (вызывается несколько экземпляров объекта $b$). В ситуации 3 хост, повторно получивший пакет, прошедший дублирующий путь, должен этот пакет игнорировать. Допустимость этой ситуации зависит от того, способен ли хост «помнить», какие пакеты он получал, чтобы иметь возможность игнорировать повторные пакеты. В данной работе мы рассматриваем случай, когда хосты не обладают такой способностью и, следовательно, ситуация 3 также должна быть запрещена. Что касается ситуации 4, то возможны разные ответы.

Сначала мы рассмотрим случай, когда ситуация 4 запрещена, т.е. пакет с сообщением для объекта $b$ должен пройти без клонирования один и только один полный путь до хоста, в котором этот объект реализован. Это случай, когда клонирование пакетов запрещено.





Затем мы ослабим требования, разрешив пакету с сообщением для объекта $b$ приходить в хост $h$, в котором этот объект не реализован. В этой ситуации предполагается, что хост $h$ просто игнорирует такой пакет. Таким образом, пакет с сообщением для объекта $b$ должен пройти, быть может, несколько полных путей за счёт его клонирования в коммутаторах, но среди этих путей должен быть один и только один путь, заканчивающийся в хосте, в котором реализован объект $b$. Это случай, когда клонирование пакетов разрешено.

В обоих случаях нас будут интересовать следующие вопросы.

1) Как минимизировать число идентификаторов пакетов, требуемых для выполнения п.1?
2) Возможна ли такая настройка коммутаторов (т.е. задание соответствующих правил в таблицах коммутаторов), чтобы любое задание, инициированное в любом начальном хосте, можно было выполнить?

Число требуемых идентификаторов пакетов – это важная характеристика сети, поскольку от неё зависит размер таблиц коммутаторов.

В данной работе нас не будут интересовать вопросы надёжности, безопасности и оптимизации нагрузки на хосты и коммутаторы, т.е. мы не будем их учитывать при рассмотрении возможности реализации требуемой настройки коммутаторов и минимизации числа идентификаторов пакетов.

Мы будем решать две задачи: 1) Минимизация числа идентификаторов пакетов, соответствующих объектам, реализованным в хостах. 2) Настройка коммутаторов для выбранного соответствия идентификаторов пакетов и объектов. Эти задачи имеют разные решения для двух случаев передачи пакетов: A) Без клонирования, когда пакет должен быть доставлен только одному хосту, в котором реализован требуемый объект. B) С клонированием, когда пакет может быть доставлен нескольким хостам, но в одном и только одном из них должен быть реализован требуемый объект.

## 2. Задача 1 (минимизация числа идентификаторов пакетов) в случае A (без клонирования)

Мы рассматриваем задачу минимизации числа идентификаторов пакетов с дополнительным требованием: идентификатор пакета, предназначенного для того или иного объекта, не зависит от хоста-отправителя. Решение задачи зависит только от множества объектов, множества хостов и распределения объектов по хостам, но не зависит от графа физических связей. С другой стороны, при решении задачи 2 (настройка коммутаторов) уже потребуется граф физических связей, а хост будет пониматься как терминальная вершина этого графа.

Множество всех объектов, реализованных во всех хостах, будем обозначать $X$. Хост будем моделировать множеством объектов, которые реализованы в этом хосте. Множество хостов есть семейство $H$ подмножеств множества $X$. Соответствие идентификаторов пакетов и объектов, реализованных в хостах, задаётся функцией $f$, отображающей каждый объект $x \in X$ в некоторый хост $f(x)$, реализующий этот объект, т.е. $x \in f(x)$. Требуется найти функцию $f: X \to H$ с минимальной мощностью образа $|f(X)|$. Идентификаторы пакетов взаимно-однозначно соответствуют хостам из образа $f(X)$. Если пакет должен быть направлен объекту $x$, то идентификатор этого пакета соответствует хосту $f(x)$, в котором реализован объект $x$.

Теорема 1. Задача 1 для случая A эквивалентна задаче о покрытии множества: для конечного множества $X$ и семейства его подмножеств $H$ требуется найти подсемейство $U \subseteq H$ наименьшей мощности, объединением которого является $X$, т.е. $\cup U = X$.

Доказательство. Утверждение непосредственно следует из того, что для каждой функции $f: X \to H$ её образ есть подсемейство семейства $H$, $f(X) \subseteq H$, и для любого подсемейства $U \subseteq H$ такого, что $\cup U = X$, можно определить функцию $f: X \to H$, выбрав для каждого объекта $x \in X$ один такой хост $h \in U$, что $x \in h$.

□

В данной работе не предлагается каких-то новых решений классической задачи о покрытии множества, нам было важно только показать, что наша задача 1 в случае A эквивалентна ей. Задача о покрытии множества одна из старейших и наиболее изученных *NP*-трудных задач [1]. Существуют различные полиномиальные алгоритмы (в частности, жадный), дающие приближённые решения с той или иной точностью, а также точные полиномиальные алгоритмы для специальных классов семейств *H*. Кроме того, рассматривались различные обобщения этой задачи, например, [1]. На эту тему есть много работ в мировой литературе. Обзор некоторых алгоритмов и методов можно найти в [3][2],[3].

Задача о покрытии множеств имеет наглядную интерпретацию в терминах матриц. Пусть $X = \{ x_1, ..., x_n \}$ и $H = \{ h_1, ..., h_m \}$, где $n = |X|$ число объектов, $m = |H|$ число хостов. Тогда эту пару $X$ и $H$ можно задать матрицей $M$ размером $n \times m$, в которой строки соответствуют объектам, столбцы – хостам, а для $i = 1, ..., n$ и $j = 1, ..., m$ ячейка матрицы $M(i,j) = 1$, если $i$-й объект реализован в $j$-м хосте, т.е. $x_i \in h_j$, и $M(i,j) = 0$ в противном случае. При этом каждая строка матрицы должна содержать хотя бы одну «1» (каждый объект должен быть реализован хотя бы в одном хосте). Такую матрицу будем называть *правильной*. Требуется найти минимальное (по числу столбцов) покрывающее подмножество столбцов, т.е. такое, чтобы в каждой строке хотя бы один из этих столбцов содержал «1». Эти столбцы образуют матрицу $K$ с $n$ строками, которую будем называть *A-производной* матрицей (для случая A). Функция $f$ отображает объект $x_i$ в хост, соответствующий столбцу $A$-производной матрицы $K$, содержащему «1» в $i$-й строке.

Для случая A (без клонирования) через $k_A(M)$ обозначим число столбцов в минимальной (по числу столбцов) матрице $A$-производной от заданной матрицы $M$. Через $k_B(n,m)$ обозначим максимум $k_A(H)$ по всем возможным матрицам размера $n \times m$. $k_A(n,m) = max\{ k_A(H) : |H| = m \ \& \ |X| = n \}$. Докажем следующее простое утверждение.

Теорема 2. Для каждых $n \ge 1$ и $m \ge 1$ имеет место $k_A(n,m) = min\{ n, m \}$.

Доказательство. Поскольку в каждой строке матрицы $M$ имеется «1» в некотором столбце, выберем для каждой строки один такой столбец. Выбранные столбцы, очевидно, образует $A$-производную матрицу, в которой число столбцов не превышает $min\{ n, m \}$, т.е. $k_A(n,m) \le min\{ n, m \}$. Эта оценка достигается в матрице $M$, в которой верхняя левая подматрица размером $min\{ n, m \}$ содержит «1» на главной диагонали и «0» в остальных ячейках; если $n > m$, то нижние $n - m$ строк заполнены «1», а если $m > n$, то правые $m - n$ столбцов заполнены «0». Любая $A$-производная матрица должна содержать первые $min\{ n, m \}$ столбцов, а минимальная $A$-производная матрица – только их.

## 3. Задача 1 (минимизация числа идентификаторов пакетов) в случае B (с клонированием)

В случае B (с клонированием) пакет, предназначенный некоторому объекту, могут получить несколько хостов, но в одном и только в одном из них должен быть реализован этот объект. Это означает, что функция имеет сигнатуру $f: X \to 2^H$ и должна удовлетворять требованию $\forall x \in X \ \exists! \ h \in f(x) \ x \in h$. Требуется найти такую функцию с минимальной мощностью образа $|f(X)|$. Идентификаторы пакетов взаимно-однозначно соответствуют множествам хостов из образа $f(X)$. Если пакет должен быть направлен объекту $x$, то идентификатор этого пакета соответствует множеству хостов $f(x)$, в одном и только одном из которых реализован объект $x$.

Переформулируем эту задачу в терминах матриц. *Суммой* одного или нескольких столбцов двоичной матрицы $M$ будем называть столбец (матрицу размера $n \times 1$), в каждой $i$-й строке которого находится сумма чисел, находящихся в $i$-й строке в суммируемых столбцах (или одному числу, если суммируется один столбец) матрицы $M$. Матрицу $K$ назовём *B-производной* от правильной матрицы $M$ (для случая B), если каждый столбец матрицы $K$ есть





сумма одного или нескольких столбцов матрицы $M$, а каждая строка содержит хотя бы одну «1». Требуется найти $B$-производную матрицу $K$ с минимальным числом столбцов. Функция $f$ отображает объект $x_i$ в множество хостов, соответствующих столбцу $B$-производной матрицы $K$, содержащему «1» в $i$-й строке.

Эта задача эквивалентна специальной модификации задачи о покрытии множества, когда в терминах матриц столбцы рассматриваемой матрицы – это все возможные суммы столбцов исходной матрицы, в которых все числа большие 1 заменены на 0.

Для случая B (с клонированием) через $k_B(M)$ обозначим число столбцов в минимальной (по числу столбцов) матрице $B$-производной от заданной матрицы $M$. Через $k_B(n,m)$ обозначим максимум $k_B(H)$ по всем возможным матрицам размера $n \times m$. $k_B(n,m) = max\{ k_B(H) : |H| = m \ \& \ |X| = n \}$.

Теорема 3. $k_B(n,m) \le m$.

Доказательство. Любая правильная матрица $M$ размером $n \times m$, очевидно, $B$-производна от самой себя и содержит $m$ столбцов. Поэтому $k_B(M) \le m$ для любой правильной матрицы M. Тем самым, $k_B(n,m) \le m$. □

Обозначим через $M_0(m)$ матрицу размером $(2^m - 1) \times m$, строки которой являются двоичными $m$-разрядными представлениями чисел от 1 до $2^m - 1$.

Лемма 1. Для каждого $m \ge 1$ матрица $M_0(m)$ правильная и $k_B(M_0(m)) = m$.

Доказательство. В матрице $M_0(m)$ есть одна строка, состоящая из одних «1», поэтому в любой $B$-производной матрице $K$ должен быть столбец, совпадающий с одним из столбцов матрицы $M_0(m)$. Выберем этот столбец матрицы $M_0(m)$. Далее для невыбранных столбцов матрицы $M_0(m)$ найдётся одна строка, в которой в этих столбцах одни «1», а в уже выбранном столбце «0». Поэтому в матрице $K$ должен быть столбец, являющийся суммой одного из невыбранных столбцов матрицы $M_0(m)$ и, быть может, каких-то уже выбранных столбцов матрицы $M_0(m)$. Выберем этот столбец матрицы $M_0(m)$. И так далее. На каждом шаге для невыбранных столбцов матрицы $M_0(m)$ имеется одна строка, в которой в этих столбцах одни «1», а в уже выбранных столбцах одни «0». Поэтому в матрице $K$ должен быть столбец, являющийся суммой одного из невыбранных столбцов матрицы $M_0(m)$ и, быть может, каких-то уже выбранных столбцов матрицы $M_0(m)$. Выберем этот столбец матрицы $M_0(m)$. В результате окажется, что в матрице $K$ число столбцов равно $m$. □

Теорема 4. Для каждого $n \ge 1$ и $m \ge 1$ существует правильная матрица $M$ размером $n \times m$, для которой $k_B(M) = min\{ \lfloor lb(n+1) \rfloor, m \}$.

Доказательство. Если $n = 2^m - 1$, то по лемме 1 Лемма 1 искомой матрицей является матрица $M_0(m)$ и $k_B(M_0(m)) = m = lb(n+1) = \lfloor lb(n+1) \rfloor = min\{ \lfloor lb(n+1) \rfloor, m \}$. Если $n > 2^m - 1$, то искомая матрица $M$ получается из матрицы $M_0(m)$ добавлением недостающих строк, каждая из которых содержит «1». Тогда $k_B(M) = m = min\{ \lfloor lb(n+1) \rfloor, m \}$. Если $n < 2^m - 1$, то искомая матрица $M$ получается из матрицы $M_0(\lfloor lb(n+1) \rfloor)$ добавлением недостающих столбцов, заполненных «0», и недостающих строк, каждая из которых содержит «1». Тогда $k_B(M) = \lfloor lb(n+1) \rfloor = min\{ \lfloor lb(n+1) \rfloor, m \}$. □

На основании теорем 3 и 4 можно предложить гипотезу о том, что $k_B(n,m) = min\{ \lfloor lb(n+1) \rfloor, m \}$. В настоящее время эта гипотеза не доказана и не опровергнута. Наиболее интересные результаты получены в [4]. Там задача 1 (минимизация числа идентификаторов) в случае B (с клонированием) названа задачей точного покрытия набором подмножеств, показана её связь с задачей о максимальной точной выполнимости монотонной КНФ, доказана APX-трудность[1] обоих задач и для нашей задачи найдено решение, гарантирующее размер точного покрытия не более, чем $O(min\{ ln(min\{ n, m \}) \cdot ln(n, m) \})$.

[1] Класс APX – класс оптимизационных задач, для которых существуют полиномиальные приближенные алгоритмы с мультипликативной ошибкой, не превышающей некоторой абсолютной константы.

## 4. Задача 2 (настройка коммутаторов) в случае A (без клонирования)

Пусть выбрано некоторое решение задачи 1 в случае A, т.е. выбрана функция $f: X \to H$ (с минимальной или близкой к минимальной мощностью образа $|f(X)|$), отображающая объект $x$ в хост $f(x)$, и установлено взаимно-однозначное отображение $d$ образа $f(X)$ во множество идентификаторов пакетов. Пакет, направляемый объекту $x$, будет иметь идентификатор $d(f(x))$.

Напомним, что при решении задачи 1 мы моделировали хост множеством реализуемых им объектов. Теперь задача 2 заключается в том, чтобы так настроить коммутаторы на плоскости данных SDN, чтобы пакет, предназначенный объекту $x$ и имеющий идентификатор $d(f(x))$, будучи отправлен с некоторого (любого) хоста, проходил один и только один полный путь до хоста $f(x)$, в котором реализован объект $x$. Но здесь мы должны вспомнить, что на плоскости данных SDN может быть несколько хостов, реализующих одно и то же множество объектов $f^{-1}(f(x))$.Иными словами, пакет с идентификатором $d(f(x))$ должен проходить полный путь до одного и только одного хоста из множества $U$ хостов, реализующих одно и то же множество объектов $f^{-1}(f(x))$. Пакеты, посылаемые из разных начальных хостов, могут проходить полные пути, заканчивающиеся в разных хостах из множества $U$.

Мы будем решать задачу 2, отвлекаясь от функции $f$. Для случая A формулировка задачи такая: Пусть дан граф физических связей $G$, идентификатор пакетов $d$ и подмножество хостов $U$. Требуется так настроить коммутаторы, чтобы пакет с идентификатором $d$, отправленный с одного (любого) хоста, проходил один и только один полный путь до некоторого хоста из множества $U$. Граф $G$ будем задавать множеством $V$ его вершин (хостов и коммутаторов) и функцией $N$, задающей для каждой вершины $v$ множество всех её соседей $N(v)$.

Правило коммутатора имеет вид $(d, a, s, b)$ и означает, что коммутатор $s$, приняв пакет с идентификаторов $d$ от соседа $a$, посылает пакет соседу $b$. Отсюда следует, во-первых, что маршрутизации пакетов с разными идентификаторами не зависят друг от друга, поскольку определяются непересекающимися наборами правил. Во-вторых, любое множество $P(d)$ полных путей для данного идентификатора $d$ пакета однозначно определяет правила настройки коммутаторов для идентификатора $d$, которые, в свою очередь, однозначно определяют множество полных путей $P(d)\downarrow\uparrow$, которые могут проходить пакеты с идентификатором $d$, как замыкание по дугам множества $P(d)$ **Ошибка! Источник ссылки не найден.**, [6]. Это замыкание по дугам определяется так: если есть два полных пути $p{\cdot}e{\cdot}q$ и $p'{\cdot}e{\cdot}q'$, где $p$, $q$, $p'$ и $q'$ пути, а $e$ дуга графа $G$, то в замыкании будут эти полные пути вместе с полными путями $p{\cdot}e{\cdot}q'$ и $p'{\cdot}e{\cdot}q$. Известно, что при замыкании по дугам полных путей может возникать зацикливание пакетов, когда пакет будет бесконечно двигаться по циклу рёбер и бесконечно клонироваться в точке разветвления цикла и постфикса пути, ведущего в конечный хост, а также дублирование, когда два разных пути имеют общий начальный хост и общий конечный хост, из-за чего конечный хост получит один и тот же пакет дважды.

Таким образом, для заданного идентификатора $d$ и соответствующего ему непустого множества хостов $U$ нужно так настроить коммутаторы плоскости данных SDN, т.е. определить такие правила коммутаторов для данного идентификатора $d$, чтобы они порождали замкнутое по дугам множество путей $P(d)$ без зацикливания и дублирования такое, чтобы из каждого хоста в этом множестве был один и только один полный путь до некоторого хоста из множества $U$. Для этого предлагается приведённый ниже алгоритм 1, который для каждого хоста строит кратчайший путь (один из кратчайших путей) до некоторого хоста из множества $U$.

Алгоритм 1 основан на обходе неориентированного графа в ширину и может рассматриваться как модификация алгоритма Дейкстра, только вместо вычисления длины кратчайшего пути выполняется настройка коммутаторов вдоль пути. Идея алгоритма заключается в следующем. Начиная с множества хостов $U$ по графу распространяется «волна» построения путей с помощью пометок вершин. Фронт этой волны $F$ состоит из





помеченных вершин $v$ с одним и тем же расстоянием до ближайшего хоста из множества $U$, тогда как остальные помеченные вершины находятся на меньшем расстоянии от $U$. Для каждой вершины $v$ из фронта волны просматриваются её соседи и для каждого соседа $w$, который ещё не помечен, выставляется пометка и запоминается его сосед $p(w) = v$, через которого ведёт кратчайший путь из вершины $w$ до ближайшего хоста из множества $U$. Если вершина $v$ коммутатор, в нём устанавливается правило $(d, w, v, p(v))$. Если вершина $w$ коммутатор, она помещается в новый фронт волны $F_{next}$. Когда просмотрены все вершины фронта $F$, начинается новый шаг, на котором фронтом волны становится $F_{next}$. Алгоритм завершает свою работу, когда просмотрены все вершины и фронт $F$ стал пустым.

**Алгоритм_1** ( $V, N, R, d, U$ ) /* Настройка коммутаторов для кратчайшего доступа (без зацикливания и дублирования) от каждого хоста до ближайшего хоста из множества хостов $U$ */

***Input*:** множество $V$ вершин графа физических связей, функция $N$, задающая для каждой вершины $v$ множество $N(v)$ её соседей, текущая настройка коммутаторов $R$, задающая для каждого коммутатора $v$ текущее множество правил вида $(d', a, s, b)$, где $d' \neq d$, новый идентификатор пакетов $d$, непустое множество хостов $U$, соответствующее $d$.

***Output*:** новая настройка коммутаторов $R$.

$r(v)$ – пометка вершины $v$ (***true***/***false***),

$p(v)$ – сосед вершины $v$ на пути к ближайшему хосту из $U$,

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

$F_{next}$ – множество $F$ на следующем шаге.

$F = \varnothing$; $F_{next} = \varnothing$;

***for all*** $v \in V$ ***do*** $r(v) =$ ***false***; /* во всех вершинах нет пометок */

***for all*** $v \in U$ ***do*** $r(v) =$ ***true***; $F = F \cup \{ v \}$; /* в хостах из $U$ есть пометки */

***while*** $F \neq \varnothing$ ***do*** /* пока фронт волны не пуст */

***for all*** $v \in F$ ***do*** /* вершины $v$ из фронта волны */

***for all*** $w \in N(v)$ & $\neg r(w)$ ***do*** /* непомеченные соседи вершины $v$ */

$r(w) =$ ***true***; $p(w) = v$; /* теперь сосед $w$ помечен */

***if*** $|N(v)| > 1$ ***then*** /* $v$ коммутатор */

$R(v) = R(v) \cup \{ (d, w, v, p(v)) \}$; /* правило коммутатора $v$ */

***if*** $|N(w)| > 1$ ***then*** /* $w$ коммутатор */

$F_{next} = F_{next} \cup \{ w \}$; /* помещаем $w$ в следующий фронт */

***return*** $R$;

Теорема 5. Алгоритм 1 правильно настраивает коммутаторы для решения задачи 2 в случае A и имеет сложность $O(m)$, где $m$ число ребер графа $G$ физических связей.

Доказательство. Сначала докажем, что на каждом $i$-ом шаге, $i = 0, 1, ...$, все вершины $v$ фронта волны $F$ имеют одно и то же расстояние $i$ до ближайшего хоста из $U$, другие помеченные вершины находятся на меньшем расстоянии, а все их соседи помечены. Действительно, вначале, на нулевом шаге, $F = U$, т.е. для каждой вершины $v$ из $F$ расстояние равно 0, а других помеченных вершин нет. Пусть утверждение верно на $i$-ом шаге и вершины фронта волны $F$ находятся на расстоянии $i$ от ближайшего хоста из $U$. В следующий фронт волны попадают соседи вершин из фронта волны, которые являются коммутаторами и ещё не помечены, т.е. они тоже имеют одно и то же расстояние $i + 1$ до ближайшего хоста из $U$. Все соседи вершин из старого фронта волны оказываются помеченными.

Поскольку граф связный, при окончании алгоритма все вершины окажутся помеченными. На каждом шаге для каждого непомеченного соседа $w$ вершины $v$ из фронта $F$ запоминается $p(w) = v$. Поэтому после завершения алгоритма для каждого хоста $v$, кроме хостов из $U$, который получил пометку на $i$-м шаге последовательность $P(v) = ( p^0(v) = v, p^1(v) = p(v), p^2(v) = p(p(v)), ..., p^i(v) )$ есть последовательность вершин на кратчайшем пути длиной $i$ от хоста $v$ до ближайшего хоста $p^i(v)$ из $U$. Алгоритм генерирует правила по внутренних вершинах этого пути, которые являются коммутаторами: для $j = 1, ..., p^{i-1}(v)$ в коммутаторе $p^j(v)$ устанавливается правило $(d, p^{j-1}(v), p^j(v), p^{j+1}(v))$. Суммарно эти правила для коммутаторов на пути $P(v)$ обеспечивают проход пакета с идентификатором $d$ из хоста $v$ по пути $P(v)$ до хоста $p^i(v)$ из $U$. Совокупность путей $P(v)$ для всех хостов $v$ из $V \setminus U$ образует лес деревьев, ориентированных к своим корням, которыми являются хосты из $U$. Поэтому это множество путей замкнуто по дугам и не содержит циклических и дублирующих путей.

Оценим сложность алгоритма. Каждое ребро графа $G$ просматривается ровно 2 раза (с обоих его концов), что суммарно даёт $m$ просмотров. Поскольку в связном графе число вершин не превышает $m + 1$, сложность алгоритма $O(m)$. □

## 5. Задача 2 (настройка коммутаторов) в случае B (с клонированием)

Пусть выбрано некоторое решение задачи 1 в случае B, т.е. выбрана функция $f: X \to 2^H$, которая удовлетворяет требованию $\forall x \in X \exists! h \in f(x)\, x \in h.f: X \to H$ и имеет минимальную или близкую к минимальной мощность образа $|f(X)|$), отображающая объект $x$ в множество хостов $f(x)$, и установлено взаимно-однозначное отображение $d$ образа $f(X)$ во множество идентификаторов пакетов. Пакет, направляемый объекту $x$, будет иметь идентификатор $d(f(x))$.

Как и в случае A мы будем решать задачу 2, отвлекаясь от функции $f$. Однако в случае A пакет с идентификатором $d(f(x))$ должен был проходить один и только один полный путь до ближайшего хоста из множества $U$ хостов. В случае B мы имеем не множество, а семейство $U$ множеств хостов, причём хосты одного множества из семейства $U$ реализуют одно и то же множество объектов, а каждый объект $x$ реализован в каждом хосте одного и только одного множества из семейства $U$. Пакет за счёт клонирования должен проходить множество полных путей так, что множество их конечных хостов является выборкой из множеств семейства $U$ по одному и только одному хосту из каждого множества. Когда пакет, предназначенный объекту $x$, попадает в хост из такого множества, происходит вызов объекта $x$. Если же этот пакет попадает в хост из другого множества семейства $U$, он игнорируется, поскольку в этих хостах он не реализован.

Для заданного идентификатора $d$ и соответствующего ему непустого семейства $U$ непустых множеств хостов нужно так настроить коммутаторы плоскости данных SDN, т.е. определить такие правила коммутаторов для данного идентификатора $d$, чтобы они порождали замкнутое по дугам множество путей $P(d)$ без зацикливания и дублирования такое, что для каждого хоста $h$ и каждого множества $U_i$ семейства $U$ подмножество путей, начинающихся в хосте $h$, содержало один и только один полный путь, заканчивающийся в некотором хосте из





множества $U_i$. Для этого предлагается приведённый ниже алгоритм 2, который строит кратчайшие пути.

Алгоритм 2 является модификацией алгоритма 1, только для каждой вершины $v$ вместо одной пометки $r(v)$ и одной ссылки на следующую вершину $p(v)$ пути используются соответствующие вектора размерности $|U|$. Вершина попадает в следующий фронт волны каждый раз, когда хотя бы в одном разряде вектора пометок $r(v)$ пометка меняется с ***false*** на ***true***. Из-за этого вершина может оказываться в нескольких (максимально $|U|$) фронтах волны.

**Алгоритм_2**( $V, N, R, d, U$ ) /* Настройка коммутаторов для кратчайшего доступа (без зацикливания и дублирования) от каждого хоста до одного хоста из каждого множества хостов в семействе множеств хостов $U$ */

***Input*:** множество $V$ вершин графа физических связей, функция $N$, задающая для каждой вершины $v$ множество $N(v)$ её соседей, текущая настройка коммутаторов $R$, задающая для каждого коммутатора $v$ текущее множество правил вида $(d', a, s, b)$, где $d' \neq d$, новый идентификатор пакетов $d$, непустое семейство $U$ непустых множеств хостов, соответствующее $d$, $k = |U|$.

***Output*:** новая настройка коммутаторов $R$.

$r(v) = \{ r(v)(1), ..., r(v)(k) \}$ – вектор пометок (***true***/***false***) вершины $v$,

$p(v) = \{ p(v)(1), ..., p(v)(k) \}$ – вектор соседей вершины $v$ на путях к ближайшим хостам из $U$,

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

$F_{next}$ – множество $F$ на следующем шаге.

$F = \varnothing$; $F_{next} = \varnothing$;

***for all*** $v \in V$ ***do***

  ***for all*** $i \in \{ 1, ..., k \}$ ***do***

    $r(v)(i) =$ ***false***; /* во всех вершинах нет пометок */

***for all*** $i \in \{ 1, ..., k \}$ ***do*** /* разряды $i$*/

  ***for all*** $v \in U(i)$ ***do*** /* вершины в $U(i)$*/

    $r(v)(i) =$ ***true***; $F = F \cup \{ v \}$; /* в хостах из $U(i)$ пометки в разряде $i$ */

***while*** $F \neq \varnothing$ ***do*** /* пока фронт волны не пуст */

  ***for all*** $v \in F$ ***do*** /* вершины $v$ из фронта волны */

    ***for all*** $w \in N(v)$ ***do*** /* соседи $w$ вершины $v$ */

      ***for all*** $i \in \{ 1, ..., k \}$ ***do*** /* разряды $i$*/

        ***if*** $r(v)(i) \setminus r(w)(i)$ ***then*** /* в разряде $i$ в $v$ пометка, а в $w$ нет */

$p(w)(i)$ = **true**; /* теперь сосед $w$ помечен в разряде $i$ */

***if*** $|N(v)| > 1$ ***then*** /* $v$ коммутатор */

$R(v) = R(v) \cup \{ (d, w, v, p(v)(i)) \}$; /* правило коммутатора $v$ */

***if*** $|N(w)| > 1$ ***then*** /* $w$ коммутатор */

$F_{next} = F_{next} \cup \{ w \}$; /* помещаем $w$ в следующий фронт */

***return*** $R$;

Теорема 6. Алгоритм 2 правильно настраивает коммутаторы для решения задачи 2 в случае B и имеет сложность $O(km)$, где $k$ мощность семейства $U$, а $m$ число ребер графа $G$ физических связей.

Доказательство. Расстояние от вершины $v$ до ближайшего хоста из множества $U(i)$, $i = 1, ..., k$, обозначим через $d(v)(i)$. Будем говорить, что расстояние $d(v)(i)$ известно, если $r(v)(i)$ = ***true***. Обозначим последовательность вершин $P(v, i) = (v_0, v_1, ..., v_{d(v)(i)})$, где $v_0 = v$ и $v_l = p(v_{l-1})$ для $l > 0$.

Сначала докажем, что в начале каждого $j$-го шага $j = 0, 1, ...$, т.е. $j$-й итерации цикла ***while***, начиная с 0, 1) все расстояния $d(v)(i) \le j$ и только они известны, 2) последовательность $P(v, i)$ для известного расстояния $d(v)(i)$ определена и есть последовательность вершин кратчайшего пути длиной $d(v)(i)$ от вершины $v$ до ближайшего хоста из$U(i)$, 3) для всех известных расстояний $d(v)(i) = j$ вершина $v$ входит во фронт волны, если $j = 0$ или $j > 0$ и $v$ коммутатор.

Действительно, в начале шага $j = 0$ 1) для $i = 1, ..., k$ хост $v \in U(i)$ имеет расстояние $d(v)(i) = 0$ (это расстояние от хоста $v$ до самого себя), и это расстояние известно, поскольку пометка $r(v)(i)$ = ***true***, а в остальных случаях, когда $v \notin U(i)$, расстояние $d(v)(i) > 0$, и оно неизвестно, поскольку пометка $r(v)(i)$ = ***false***. 2) Для известного расстояния $d(v)(i) = 0$ последовательность $P(v, i) = (v)$ есть последовательность вершин на кратчайшем пути длиной $d(v)(i) = 0$ от вершины $v$ до ближайшего хоста $v$ из $U(i)$, т.е. до самого себя. 3) $j = 0$ и каждая вершина $v \in U(i)$ входит во фронт волны $F = \cup U$.

Пусть утверждение верно в начале $j$-ого шага и докажем, что оно остаётся верным в начале следующего $(j+1)$-го шага. 1) Пусть вершина $w$ для некоторого $i = 1, ..., k$ имеет расстояние $d(w)(i) = j+1$. По предположению шага индукции все расстояния $d(v)(i) \le j$ известны. Следовательно, найдётся такая вершина $v$, для которой $d(v)(i) = j$ и которая является соседом вершины $w$. По предположению шага индукции вершина $v$ входит во фронт волны. Поскольку на каждом шаге алгоритм просматривает всех соседей вершин из фронта волны, будет просмотрена вершина $w$ и установлена пометка $r(w)(i)$ = ***true***, тем самым расстояние $d(w)(i) = j+1$ станет известным к началу следующего $(j+1)$-го шага. Если же вершина $t$ для некоторого $i = 1, ..., k$ имеет расстояние $d(t)(i) > j+1$, то, очевидно, её соседом не может быть вершина $v$, для которой $d(v)(i) = j$, поэтому на следующем шаге останется $r(t)(i)$ = ***false***, т.е. расстояние $d(t)(i)$ останется неизвестным. 2) В вершине $w$ с расстоянием $d(w)(i) = j+1$ также будет установлено $p(w)(i) = v$. Поскольку по предположению шага индукции последовательность $P(v, i)$ определена и есть последовательность вершин кратчайшего пути длиной $d(v)(i)$ от вершины $v$ до ближайшего хоста из $U(i)$, последовательность $P(w) = (w) \cdot P(v)$ также определена и есть последовательность вершин кратчайшего пути длиной $d(w)(i)$ от вершины $w$ до ближайшего хоста из $U(i)$. 3) $j+1 > 0$ и все вершины $w$ с расстоянием $d(w)(i) = j+1$ для $i = 1, ..., k$, которые являются коммутаторами, и только они войдут в следующий фронт волны.

Поскольку граф связный, при окончании алгоритма все расстояния $d(v)(i)$ станут известными и для каждого хоста $v$ последовательность $P(v, i)$ определена и есть последовательность вершин кратчайшего пути от хоста $v$ до ближайшего хоста из $U(i)$. Поскольку на каждом шаге





алгоритм устанавливает правило $(d, w, v, p(v)(i))$ для каждого коммутатора $v$, каждой вершины $w$ и каждого разряда $i$ при условии $p(w)(i) = v$, совокупность установленных правил обеспечивает прохождение пакетов с идентификатором $d$ из каждого хоста $h$ путей $P(h, i)$, $i = 1, ..., k$, которые являются кратчайшими путями из $h$ до ближайших хостов из множеств $U(i)$. Пути, проходящие дугу $(w, v)$, в вершине $v$ разветвляются (пакеты клонируются) на несколько ветвей по дугам $(v, p(v)(i))$, $i = 1, ..., k$ (число дуг может быть меньше $k$, так как дуги могут совпадать для некоторых $i$), через которые каждый путь $P(v, i)$ ведёт в ближайший хост из $U(i)$. Тем самым, множество полных путей $P(h, i)$, $i = 1, ..., k$, по всем хостам $h$, замкнуто по дугам и не содержит циклов и дублирующих путей.

Оценим сложность алгоритма. Каждое ребро графа $G$ просматривается ровно $2k$ раз: с обоих его концов и для каждого разряда $i = 1, ..., k$, что суммарно даёт $2km$ просмотров. Поскольку в связном графе число вершин не превышает $m + 1$, сложность алгоритма $O(km)$.

## *5. Заключение*

В статье рассматривается выполнение на плоскости данных SDN, моделируемой конечным связным неориентированным графом физических связей, программы задания, которая понимается в духе парадигмы объектно-ориентированного программирования как состоящая из объектов и сообщений, которыми объекты могут обмениваться. Объекты реализуется в хостах, причём в одном хосте может быть реализовано несколько разных объектов, а один и тот же объект может быть реализован в нескольких хостах. Сообщения между объектами, реализованными в разных хостах, помещаются в пакеты, маршрутизацию которых исполняют коммутаторы на основе идентификаторов, присвоенных пакетам и помещаемых в заголовки пакетов как набор значений некоторых параметров пакетов. В работе решаются две задачи: 1) минимизация числа идентификаторов и 2) настройка коммутаторов для реализации путей, которые должны проходить коммутаторы. Эти задачи решаются в двух случаях: A) пакет, предназначенный для некоторого объекта, должен попасть ровно в один хост, в котором реализован этот объект, B) пакет может попадать в несколько хостов, но в одном и только одном из них должен быть реализован нужный объект.

Показано, что задача 1 в случае A эквивалентна задаче о покрытии множества, а минимальное число идентификаторов в наихудшем случае равно $min\{n, m\}$, где $n$ число объектов, а $m$ число хостов, реализующих объекты. В случае B задача является специальной модификацией задачи о покрытии множества (задачей точного покрытия набором подмножеств [4]). Доказано, что минимальное число идентификаторов в наихудшем случае не превышает $m$, и для каждых $n$ и $m$ имеется пример, когда это число равно $min\{\lfloor lb(n+1) \rfloor, m\}$. Предложена гипотеза о том, что $min\{\lfloor lb(n+1) \rfloor, m\}$ является также верхней оценкой. В настоящее время эта гипотеза не доказана и не опровергнута. В [4]. доказана верхняя оценка $O(min\{ ln(min\{n, m\}) \cdot ln(n, m) \})$.

Для решения задачи 2 в случаях A и B предложены алгоритмы настройки коммутаторов сложности, соответственно, $O(m)$ и $O(km)$, где $m$ число рёбер графа физических связей, а $k$ результат решения задачи 1 в случае B как число требуемых идентификаторов пакетов.

В дальнейших исследованиях можно попытаться учесть вопросы надёжности, безопасности и оптимизации нагрузки на хосты и коммутаторы, которые игнорируются в данной работе. В частности, можно рассматривать взвешенный граф физических связей и налагать ограничения на маршрутизацию пакетов.

## Список литературы / References

[1]. Н.Н. Кузюрин. Обобщенные покрытия и их аппроксимации. Труды ИСП РАН, том 6, 2004 г., стр. 85-100 / N.N. Kuzyurin. Generalized covers and their approximations. Trudy ISP RAN/Proc. ISP RAS, vol. 6, 2004, pp. 85-100 (in Russian).

[2]. Н.Н. Кузюрин. О сложности построения асимптотически оптимальных покрытий и упаковок. Доклады Академии наук, том 363, no. 1., 1998 г., стр. 11-13 / N. N. Kuzyurin, On the complexity of asymptotically optimal coverings and packing. Doklady Mathematics, vol. 58, no. 3, 1998, pp, 345-346.

[3]. А.В. Еремеев, Л.А. Заозерская, А.А. Колоколов. Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования, Дискретный анализ и исследование операций, том 7, вып. 2, 2000 г., стр. 22-46 / A.V. Eremeev, L.A. Zaozerskaya, A.A. Kolokolov. The set covering problem: complexity, algorithms, and experimental investigations. Discrete Analysis and Operations Research, vol. 7, issue 2, 2000, pp. 22-46 (in Russian).

[4]. D. Lazarev. On MAX Monotonous XSAT, Exact Cover by Sets of Subsets and Parallel Computations in Software-Defined Network. Moscow Journal of Combinatorics and Number Theory, 2023 (in print).

[5]. 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. DOI: 10.15514/ISPRAS-2018-30(6)-4 / 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).

[6]. I. Burdonov, A. Kossachev et al. Preventive Model-based Verification and Repairing for SDN Requests. In Proc. of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering 2021, pp. 421-428.

## Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, г.н.с. ИСП РАН. Научные интересы: формальные спецификации, генерация тестов, технологиякомпиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработкипрограммного обеспечения.

Igor Borisovich BURDONOV – Doctor of Physical and Mathematical Sciences, a Leading Researcher of ISP RAS. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, softwaredevelopment 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.