ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

Том 25 2(74) 2018

Основан в 1999 году
Выходит 6 раз в год

Главный редактор
В.А. Соколов,
доктор физико-математических наук, профессор, Россия

Редакционная коллегия

С.М. Абрамов, д-р физ.-мат. наук, чл.-корр. РАН, Россия; **Л. Авено**, проф., Франция; **В.С. Афраймович**, проф.-исследователь, Мексика; **О.Л. Бандман**, д-р техн. наук, Россия; **В.Н. Белых**, д-р физ.-мат. наук, проф., Россия; **В.А. Бондаренко**, д-р физ.-мат. наук, проф., Россия; **С.Д. Глызин**, д-р физ.-мат. наук, проф., Россия (зам. гл. ред.); **А. Дехтярь**, проф., США; **М.Г. Дмитриев**, д-р физ.-мат. наук, проф., Россия; **В.Л. Дольников**, д-р физ.-мат. наук, проф., Россия; **В.Г. Дурнев**, д-р физ.-мат. наук, проф., Россия; **В.А. Захаров**, д-р физ.-мат. наук, проф., Россия; **Л.С. Казарин**, д-р физ.-мат. наук, проф., Россия; **Ю.Г. Карпов**, д-р техн. наук, проф., Россия; **С.А. Кащенко**, д-р физ.-мат. наук, проф., Россия; **А.Ю. Колесов**, д-р физ.-мат. наук, проф., Россия; **Н.А. Кудряшов**, д-р физ.-мат. наук, проф., Заслуженный деятель науки РФ, Россия; **О. Кушнаренко**, проф., Франция; **И.А. Ломазова**, д-р физ.-мат. наук, проф., Россия; **Г.Г. Малинецкий**, д-р физ.-мат. наук, проф., Россия; **В.Э. Малышкин**, д-р техн. наук, проф., Россия; **А.В. Михайлов**, д-р физ.-мат. наук, проф., Великобритания; **В.А. Непомнящий**, канд. физ.-мат. наук, Россия; **Н.Х. Розов**, д-р физ.-мат. наук, проф., чл.-корр. РАО, Россия; **Н. Сидорова**, д-р наук, Нидерланды; **Р.Л. Смелянский**, д-р физ.-мат. наук, проф., член-корр. РАН, академик РАЕН, Россия; **Е.А. Тимофеев**, д-р физ.-мат. наук, проф., Россия (зам. гл. ред.); **М.Б. Трахтенброт**, д-р комп. наук, Израиль, **Д.В. Тураев**, проф., Великобритания; **Ф. Шнеблен**, проф., Франция

Ответственный секретарь **Е. В. Кузьмин**, д-р физ.-мат. наук, проф., Россия

Адрес редакции: ЯрГУ, ул. Советская, 14, г. Ярославль, 150003, Россия
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru; телефон (4852) 79-77-73

Научные статьи в журнал принимаются по электронной почте. Статьи должны содержать УДК, аннотации на русском и английском языках и сопровождаться набором текста в редакторе LaTeX. Плата с аспирантов за публикацию рукописей не взимается.

12+

©Ярославский государственный университет им. П.Г. Демидова, 2018

СОДЕРЖАНИЕ

Моделирование и анализ информационных систем. Т. 25, №2. 2018

Модели параллелизма

Сети Петри и временные автоматы

Программно-конфигурируемые сети

Модели технических систем

Кодовые криптосистемы

Свидетельство о регистрации СМИ ПИ № ФС 77 – 66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Учредитель – Федеральное государственное бюджетное образовательное учреждение высшего образования "Ярославский государственный университет им. П. Г. Демидова". Подписной индекс – 31907 в Объединенном каталоге "Пресса России". Редактор, корректор А.А. Аладьева. Редактор перевода Э.И. Соколова. Подписано в печать 18.04.2018. Дата выхода в свет 30.04.2018. Формат 60x84$^1/_8$. Усл. печ. л. 11,1. Уч.-изд. л. 10,0. Объем 95 с. Тираж 46 экз. Свободная цена. Заказ 021/018. Адрес типографии: ул. Советская, 14, оф. 109, г. Ярославль, 150003 Россия. Адрес издателя: Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, г. Ярославль, 150003 Россия.

ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

P.G. Demidov Yaroslavl State University

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

Volume 25 No 2(74) 2018

Founded in 1999
6 issues per year

Editor-in-Chief
V. A. Sokolov,
Doctor of Sciences in Mathematics, Professor, Russia

Editorial Board

S.M. Abramov, Prof., Dr. Sci., Corr. Member of RAS, Russia; **V. Afraimovich**, Prof.-researcher, Mexico; **L. Aveneau**, Prof., France; **O.L. Bandman**, Prof., Dr. Sci., Russia; **V.N. Belykh**, Prof., Dr. Sci., Russia; **V.A. Bondarenko**, Prof., Dr. Sci., Russia; **S.D. Glyzin**, Prof., Dr. Sci., Russia (*Deputy Editor-in-Chief*); **A. Dekhtyar**, Prof., USA; **M.G. Dmitriev**, Prof., Dr. Sci., Russia; **V.L. Dol'nikov**, Prof., Dr. Sci., Russia; **V.G. Durnev**, Prof., Dr. Sci., Russia; **L.S. Kazarin**, Prof., Dr. Sci., Russia; **Yu.G. Karpov**, Prof., Dr. Sci., Russia; **S.A. Kashchenko**, Prof., Dr. Sci., Russia; **A.Yu. Kolesov**, Prof., Dr. Sci., Russia; **O. Kouchnarenko**, Prof., France; **N.A. Kudryashov**, Dr. Sci., Prof., Russia; **I.A. Lomazova**, Prof., Dr. Sci., Russia; **G.G. Malinetsky**, Prof., Dr. Sci., Russia; **V.E. Malyshkin**, Prof., Dr. Sci., Russia; **A.V. Mikhailov**, Prof., Dr. Sci., Great Britain; **V.A. Nepomniaschy**, PhD, Russia; **N.H. Rozov**, Prof., Dr. Sci., Corr. Member of RAE, Russia; **Ph. Schnoebelen**, Senior Researcher, France; **N. Sidorova**, Dr., Assistant Prof., Netherlands; **R.L. Smeliansky**, Prof., Dr. Sci., Corr. Member of RAS, Russia; **E.A. Timofeev**, Prof., Dr. Sci., Russia (*Deputy Editor-in-Chief*); **M. Trakhtenbrot**, Dr., Israel; **D. Turaev**, Prof., Great Britain; **V.A. Zakharov**, Prof., Dr. Sci., Russia

Responsible Secretary **E. V. Kuzmin**, Prof., Dr. Sci., Russia

Editorial Office Address: P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru

© P.G. Demidov Yaroslavl State University, 2018

Contents

Modeling and Analysis of Information Systems. Vol. 25, No 2. 2018

Models of Parallelism

Petri Nets and Timed Automata

Software Defined Networks

Models of Technical Systems

Code Cryptosystems

Моделирование и анализ информационных систем. Т. 25, № 2 (2018), с. 155–164
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 155–164

Модели параллелизма

©Васильчиков В. В., 2017

DOI: 10.18255/1818-1015-2018-2-155-164

УДК 519.688: 519.85

О рекурсивно-параллельном алгоритме решения задачи о рюкзаке

Васильчиков В. В.

получена 27 декабря 2017

Аннотация. Предлагается эффективный параллельный алгоритм решения NP-полной задачи о рюкзаке в ее исходном, так называемом 0-1 варианте. Для нахождения ее точного решения издавна применяются алгоритмы, относящиеся к категории "методов ветвей и границ". Для ускорения получения результата с разной степенью эффективности применяются также различные варианты организации параллельных вычислений. Мы предлагаем здесь алгоритм решения задачи, основанный на парадигме рекурсивно-параллельных вычислений. Он представляется нам хорошо пригодным для задач такого рода, когда трудно сразу разбить вычисления на достаточное количество сравнимых по трудоемкости подзадач, поскольку она проявляется динамически во время вычислений. В качестве основного инструмента для программной реализации алгоритма использовалась разработанная автором библиотека RPM_ParLib, позволяющая создавать эффективные приложения для вычислений на локальной сети в среде .NET Framework. Такие приложения обладают способностью порождать параллельные ветви вычислений непосредственно во время выполнения программы и динамически перераспределять работу между вычислительными модулями. При этом в качестве языка программирования может использоваться любой язык с поддержкой .NET Framework. Для проведения экспериментов было написано несколько программ на языке C# с использованием упомянутой библиотеки. Основной целью этих экспериментов было исследование ускорения, достигаемого за счет рекурсивно-параллельной организации вычислений. Подробное описание алгоритма и эксперимента, а также полученные результаты приводятся в работе.

Ключевые слова: задача о рюкзаке, параллельный алгоритм, рекурсия, .NET

Для цитирования: Васильчиков В. В., "О рекурсивно-параллельном алгоритме решения задачи о рюкзаке", *Моделирование и анализ информационных систем*, **25**:2 (2018), 155–164.

Об авторах:
Васильчиков Владимир Васильевич, orcid.org/0000-0001-7882-8906, канд. техн. наук, зав. кафедрой вычислительных и программных систем,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: vvv193@mail.ru

Благодарности:
Работа выполнена в рамках инициативной НИР ВИП-004 (номер госрегистрации АААА-А16-116070610022-6).

Введение

Задача о рюкзаке является одной из классических задач дискретной оптимизации [1]. Для нее существует несколько различных вариантов постановки [2, 3] и алгоритмов решения [2–4]. В данной статье мы рассматривам только одну ее разновидность, а именно классический 0-1 вариант. В такой постановке задача относится к категории NP-полных задач, для которых не найдено алгоритмов точного решения за полиномиальное от размера задачи время [1]. Однако для решения ряда задач такого рода с успехом применяются алгоритмы, относящиеся с категории "методов ветвей и границ". Идея метода было предложена в [5] применительно к задаче целочисленного линейного программирования. Для задачи о рюкзаке алгоритм такого рода и целый ряд его модификаций был разработан и исследован Д. Пизингером [6, 7]. Для оценивания перспективности вычислений на очередной ветви, а это один из ключевых моментов такого рода алгоритмов, он использовал оценки, полученные Г. Данцигом в [8].

По своей природе метод ветвей и границ все-таки является перебором вариантов, хотя упомянутые выше оценки перспективности продолжения вычислений на очередной ветви позволяют существенно увеличить скорость получения результата. Однако совершенно естественным выглядит желание добиться ускорения за счет построения параллельного алгоритма решения задачи. Разные авторы посвящали свои работы построению параллельных алгоритмов решения задачи о рюкзаке, основанных на самых разных парадигмах и инструментах параллельного программирования, в качестве одного из примеров упомянем работу [9]. При этом основной проблемой при построении такого рода алгоритмов является разбиение целой задачи на подзадачи сравнимой трудоемкости и обеспечение достаточно равномерной загрузки вычислительных мощностей.

Метод ветвей и границ практически всегда допускает описание в виде рекурсии, поэтому автору показалось совершенно естественным использовать для распараллеливания алгоритма концепцию рекурсивно-параллельного (РП) программирования и соответствующие библиотеки поддержки. В [10] изложены основные принципы организации рекурсивно-параллельных вычислений и описаны основные алгоритмы и механизмы поддержки этого стиля программирования. Библиотеки [11,12] позволяют относительно легко создавать, отлаживать и эксплуатировать РП-приложения в среде .NET Framework. В [13] подробно описаны функциональные возможности упомянутых библиотек, а также процесс разработки и исследования РП-алгоритма решения NP-полной задачи о клике. В качестве другого примера использования данного стиля программирования и упомянутых библиотек приведем работу [14], в которой разработан и исследован рекурсивно-параллельный алгоритм решения задачи коммивояжера, также относящейся к категории NP-полных.

1. Задача о рюкзаке

Напомним формулировку задачи. При этом мы будем в основном придерживаться обозначений, использованных в [6], поскольку именно один из алгоритмов, изложенных в данной работе, положен в основу нашего параллельного алгоритма.

Существует несколько постановок задачи о рюкзаке. Все эти постановки предполагают, что имеется некоторое количество предметов, для каждого из которых задана ценность p_i и размер c_i. Рюкзак характеризуется предельной вместимостью c. Все перечисленные величины p_i, c_i, c являются целыми положительными числами. Нас будет интересовать постановка задачи в так называемом 0-1 варианте. Она задается следующим образом.

Пусть у нас имеется n предметов с заданной ценностью и размером. Требуется максимизировать величину ценности выбранных предметов $\sum_{i=1}^{n} p_i x_i$ при ограничении на вместимость рюкзака $\sum_{i=1}^{n} c_i x_i \leq c$, где . Здесь x_i – бинарная величина, равная 1, если мы положили предмет в рюкзак, и 0 в противном случае. Другие варианты формулировки задачи о рюкзаке мы здесь не рассматриваем.

2. Последовательный алгоритм решения задачи

Вкратце опишем основные шаги алгоритма, приведенного в [6], поскольку именно он является базовым для построения нашего параллельного алгоритма.

1. Упорядочиваем предметы в порядке неувеличения эффективности $e_i = p_i/w_i$, так что $e_i \geq e_j$ при $i < j$.

2. Набираем предметы в рюкзак "жадным" образом, пока позволяет его вместимость. Пусть первый элемент, не поместившийся в рюкзак, имеет номер b, назовем его предельным. Если обозначить через W_k суммарный размер первых k элементов из нашего упорядоченного множества, имеем соотношение $W_{b-1} \leq c < W_b$. Таким образом мы построили базовый набор из $b-1$ предметов стоимости P_{b-1}. Ему соответствует набор значений x_i, такой что $x_i = 1$ для $1 \leq i \leq b-1$ и $x_i = 0$ для $b \leq i \leq n$.

3. Дальнейшая работа алгоритма заключается в попытках улучшить базовое решение. Изменения фиксируются в виде так называемого списка исключений, содержащего номера элементов из массива $\{x_i\}$, значения которых следует изменить на противоположные.

Действия из последнего пункта выполняются рекурсивно в соответствии с методологией "ветвей и границ". При этом множество потенциальных исключений расширяется в обе стороны от значения b. Оно представляет собой целые значения из промежутка с нижней границей s и верхней границей t. Основными параметрами рекурсивной процедуры являются ценность построенного набора, его размер, s и t. При первом вызове они задаются значениями P_{b-1}, W_{b-1}, $b-1$ и b соответственно.

На очередном шаге рекурсии, если размер предмета с индексом t позволяет его добавить в построенный набор, рекурсивно вычисляется ценность решения для этого варианта, t включается в список исключений и увеличивается на единицу. Если предмет с индексом t не помещается в построенный набор, пытаемся удалить из набора предмет с индексом s, применяя для этого соответствующий рекурсивный вызов. Более понятно этот процесс иллюстрируется приведенной ниже блок-схемой заключительной фазы рекурсивно-параллельного алгоритма, поскольку она фактически повторяет последовательный алгоритм.

Отметим, что важнейшим элементом алгоритмов и категории "методов ветвей и границ" является оценка перспективности вычислений на отдельно взятой ветви. Как в базовом алгоритме [6], так и у нас используется оценка, предложенная Г. Данцигом в [8]. В соответствии с ней, верхней оценкой ценности итогового набора предметов для нашей задачи является

$$u = \lfloor P_{b-1} + \frac{(c - W_{b-1})p_b}{w_b} \rfloor,$$

где $\lfloor a \rfloor$ – наибольшее целое, не превышающее a.

Соответственно, на ветви, где осуществляется проверка перспективности добавления предмета с индексом t, то есть в ситуации $W \leq c$, верхняя оценка не превысит z, если

$$u = \lfloor P + \frac{(c - W)p_t}{w_t} \rfloor \leq z,$$

а при оценке перспективности ветви с удалением предмета s $(W > c)$ следует проверять условие

$$u = \lfloor P + \frac{(c - W)p_s}{w_s} \rfloor \leq z.$$

В программном коде удобнее использовать эквивалентные проверки: для $W \leq c$

$$det(P - z - 1, W - c, p_t, w_t) < 0,$$

а для $W > c$

$$det(P - z - 1, W - c, p_s, w_s) < 0.$$

3. Рекурсивно-параллельный алгоритм

Для задач такого рода, когда алгоритм естественным образом описывается рекурсивно с разбиением на две подзадачи, построение РП-алгоритма, казалось бы, не представляет сложности. Вместе с тем опыт разработки и исследования таких программ [13,14] показывает, что такой прямолинейный подход не всегда является наилучшим, поскольку не позволяет в полной мере использовать возможности механизма динамической балансировки загрузки, заложенные в библиотеки поддержки данного стиля программирования. Поэтому при решении задач, описанных в [13,14], мы использовали специальные приемы оформления подзадач для достижения большего ускорения за счет параллельного выполнения. Какой прием окажется лучше, каждый раз определяется экспериментально.

В случае задачи о рюкзаке достаточно хорошие результаты показал, однако, именно такой прямолинейный подход, когда в качестве подзадачи на каждом шаге рекурсии оформлялась ветка, удовлетворяющая условию $W \leq c$ или $W > c$ соответственно. Здесь, правда, на первый план выступают другие аспекты оформления подзадачи, связанные в основном с необходимостью эффективной организации структур данных, передаваемых подзадаче в качестве исходных, в виде так называемого блока параметров, а также возвращаемых через блок параметров результатов. К слову сказать, в [6] вопросам принадлежности данных к тому или иному

классу внимания уделено не было, а неправильное понимание этого момента приводит к неверной работе всего алгоритма. Поэтому ниже перед описанием собственно алгоритма мы опишем структуру блока параметров рекурсивно-параллельной активации. Другой особенностью данной задачи и алгоритма ее решения является то, что результирующее множество строится на обратном ходе рекурсии, а значит, неизвестно в момент принятия решения о включении или исключении из него очередного предмета. И наконец, проверку перспективности очередной ветви мы производим еще до ее порождения, чтобы не создавать фактически пустых потенциально-мигрирующих активаций РП-процедур.

Основными значениями, включенными в блок параметров, в нашем варианте являются следующие:

- Текущий уровень вложенности рекурсии, он увеличивается только при удвоении их числа. В качестве ограничения рекурсивного деления мы используем некоторое предельное значение уровня вложенности. В нашем случае это позволяет достаточно хорошо управлять процессом рекурсивного деления.

- Стартовые значения W, P, s и t, а также признак того, что на данной ветви результат был улучшен ($improved$), и соответствующее улучшенное значение ценности ($localBest$).

- Список исключений, накопленных данной активацией и ее потомками. Создается на обратном ходе рекурсии в тех случаях, когда он требуется для возврата.

На рисунке 1 приводится блок-схема, представляющая логику исполнения параллельного вызова основной вычислительной процедуры нашего алгоритма. Она позволяет понять, как и в каких случаях осуществляется разбиение задачи на две подзадачи, как осуществляется синхронизация выполнения и формирование результата.

По достижении заданного предела дробления задачи дальнейшие вычисления осуществляет процедура $RecursiveBranch(W, P, s, t)$, которая реализует последовательное решение. Блок-схема ее работы представлена на рисунке 2.

4. Результаты тестирования

Исходные данные для тестирования генерировались случайным образом. При этом размер и ценность предметов варьировались не слишком значительно, чтобы дополнительно усложнить задачу. Было сгенерировано несколько десятков наборов исходных данных объемом в 60 и 80 предметов. Размер и ценность их брались случайно из диапазона от 1850 до 2150. Размер рюкзака позволял взять примерно половину предметов. Но даже для таких примерно однотипных наборов исходных данных время работы последовательного алгоритма варьировалось как минимум в десятки тысяч раз. Некоторые решались быстрее секунды, для других не хватало двух часов, в этих случаях работу программы приходилось завершать, не дождавшись получения окончательного решения. Поэтому при исследовании эффективности параллельного алгоритма мы отобрали только те варианты исходных данных,

на которых последовательный алгоритм выдавал решение за время в пределах от 3 минут до 1 часа.

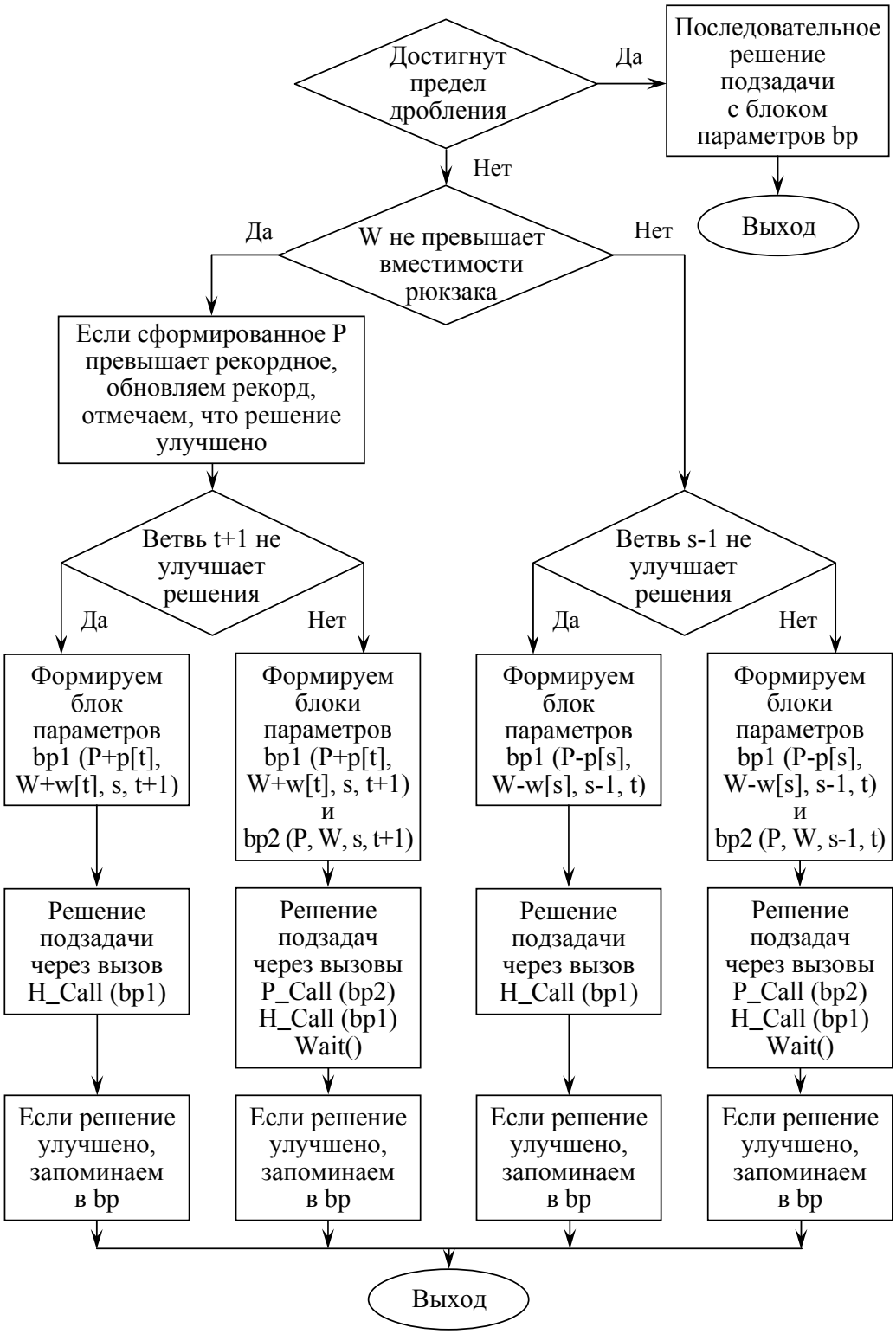

Рис. 1. Блок-схема параллельного выполнения основной процедуры
Fig. 1. Block diagram of parallel execution of the main procedure

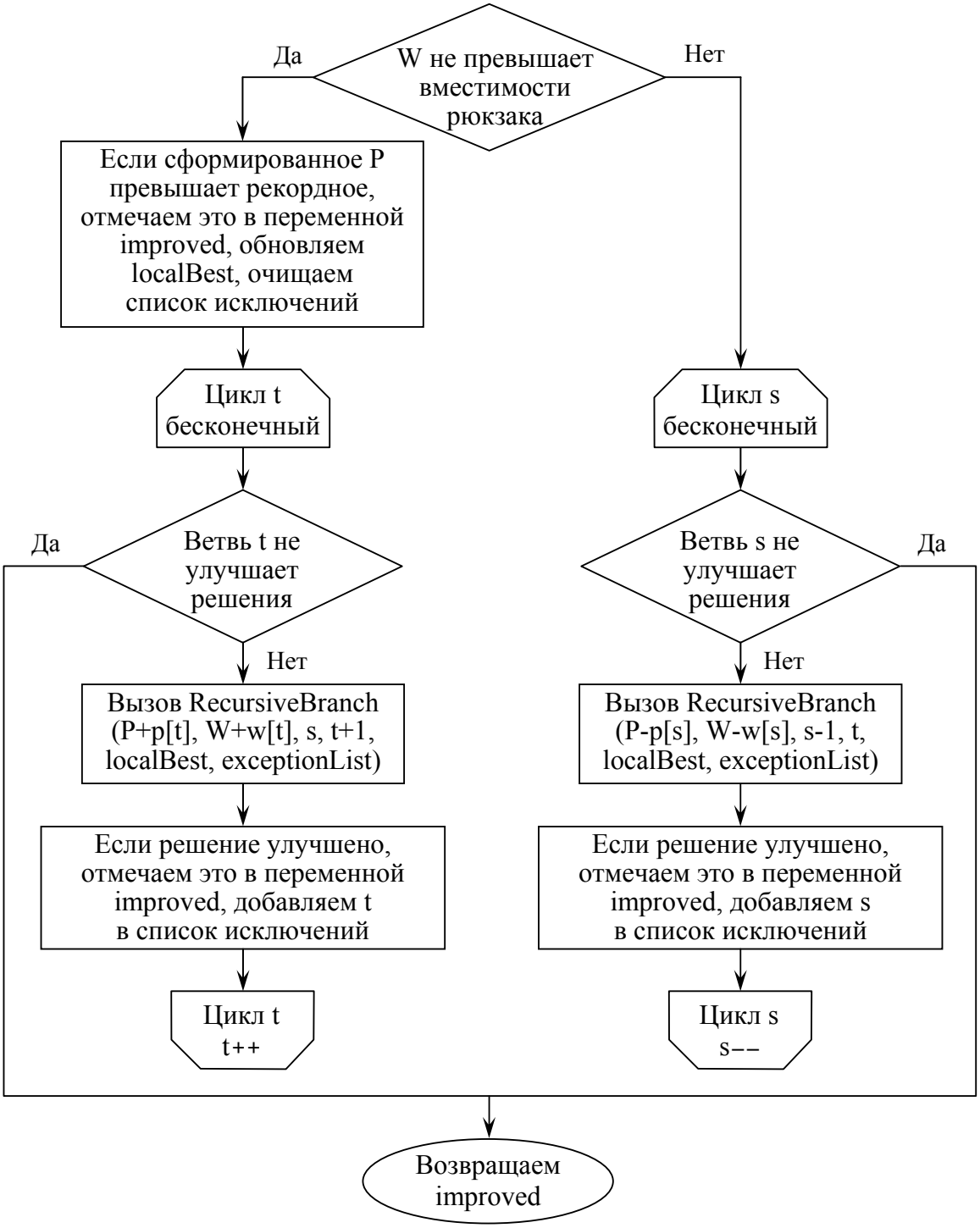

Рис. 2. Блок-схема последовательного выполнения основной процедуры
Fig. 2. Block diagram of serial execution of the main procedure

В процессе тестирования использовались компьютеры на базе четырехъядерного процессора Intel Core i3 с тактовой частотой 3.07 GHz и 4 GB оперативной памяти, работающие под управлением 64-разрядной ОС Windows 7. Пропускная способность сети равнялась 100 Mb/s.

В таблице 1 приведены некоторые результаты вычислений, позволяющие оценить трудоемкость решения задачи, время решения и ускорение, полученное для нескольких случайных наборов исходных данных из 80 предметов. Трудоемкость задачи можно оценить по указанному количеству порожденных параллельных ветвей (суммарно для всех процессорных модулей – ПМ).

Таблица 1. Трудоемкость и ускорение РП-алгоритма на задаче с 80 предметами

Table 1. The complexity and acceleration of the RP-algorithm on a problem with 80 objects

№ п/п	**Кол-во ПМ**	**1**	**2**	**4**	**6**	**8**	**12**	**16**
1	Ветвей (млн)	*10 551.23*	*154.76*	*304.54*	*241.29*	*298.51*	*220.68*	*222.12*
	Время (с)	547.62	8.33	7.37	5.72	5.78	5.79	5.25
	Ускорение	***1.00***	***65.77***	***74.35***	***95.69***	***94.81***	***94.56***	***104.39***
2	Ветвей (млн)	*15 114.60*	*25 538.38*	*51 228.36*	*39 633.71*	*33 172.25*	*16 164.25*	*19 662.56*
	Время (с)	762.86	663.94	654.11	338.66	209.18	70.86	67.27
	Ускорение	***1.00***	***1.15***	***1.17***	***2.25***	***3.65***	***10.77***	***11.34***
3	Ветвей (млн)	*45 510.13*	*33 072.12*	*30 943.51*	*30 197.71*	*15 252.44*	*21 327.09*	*11 504.56*
	Время (с)	2 263.17	859.64	393.76	260.25	98.73	93.61	41.95
	Ускорение	***1.00***	***2.63***	***5.75***	***8.70***	***22.92***	***24.18***	***53.95***
4	Ветвей (млн)	*5 332.43*	*2 359.90*	*2 020.09*	*2 894.13*	*3 846.42*	*5 689.95*	*7 175.60*
	Время (с)	272.35	65.69	31.37	27.42	28.26	27.01	27.40
	Ускорение	***1.00***	***4.15***	***8.68***	***9.93***	***9.64***	***10.08***	***9.94***
5	Ветвей (млн)	*21 477.44*	*3 453.82*	*5 502.24*	*6 892.83*	*8 233.19*	*3 530.34*	*1 559.01*
	Время (с)	1 068.25	94.26	73.25	62.82	56.19	18.59	9.49
	Ускорение	***1.00***	***11.33***	***14.58***	***17.00***	***19.01***	***57.48***	***112.54***
6	Ветвей (млн)	*3 956.54*	*1 843.44*	*3 768.49*	*1 394.06*	*1 518.84*	*2 242.82*	*1 046.64*
	Время (с)	209.42	52.65	51.37	14.37	12.32	12.71	6.87
	Ускорение	***1.00***	***3.98***	***4.08***	***14.58***	***17.00***	***16.48***	***30.47***
7	Ветвей (млн)	*47 836.56*	*33 829.43*	*12 369.67*	*17 925.07*	*9 858.63*	*11 678.20*	*13 985.47*
	Время (с)	2 466.22	878.55	161.71	155.08	64.90	52.94	48.84
	Ускорение	***1.00***	***2.81***	***15.25***	***15.90***	***38.00***	***46.58***	***50.49***
8	Ветвей (млн)	*4 650.47*	*1 041.19*	*1 178.17*	*1 317.48*	*1 432.33*	*1 730.72*	*2 169.74*
	Время (с)	229.81	27.93	16.01	14.47	12.10	10.63	13.60
	Ускорение	***1.00***	***8.23***	***14.36***	***15.88***	***19.00***	***21.62***	***16.90***
9	Ветвей (млн)	*52 898.13*	*92 754.95*	*36 778.80*	*47 898.28*	*21 868.91*	*29 560.00*	*29 560.00*
	Время (с)	2 612.83	2 365.98	489.68	435.12	150.86	139.26	77.43
	Ускорение	***1.00***	***1.10***	***5.34***	***6.00***	***17.32***	***18.76***	***33.74***
10	Ветвей (млн)	*45 674.54*	*9 553.20*	*29 971.79*	*13 136.65*	*24 774.09*	*5 885.24*	*10 219.88*
	Время (с)	2 345.08	252.49	383.23	119.37	160.02	28.44	38.31
	Ускорение	***1.00***	***9.29***	***6.12***	***19.64***	***14.66***	***82.46***	***61.21***

Анализируя полученные результаты, можно сделать следующие выводы. Конечно, точность оценки решения на исследуемой ветви вычислений имеет очень важное значение для скорости получения результата, однако это далеко не единственный фактор, определяющий время вычислений как при последовательном, так и при параллельном решении задачи.

Как показывают результаты экспериментов, едва ли не более важным фактором, определяющим скорость решения задачи, является то, как быстро будет найдено лучшее или достаточно близкое к нему решение. После нахождения такового количество ветвей вычислений, отвергаемых на ранних стадиях ввиду их бесперспективности, резко возрастает. По этой причине время решения конкретной задачи является непредсказуемым, и становится весьма некорректным делать какие-то выводы о зависимости этого времени, скажем, от размера задачи, если только он не изменяется очень сильно.

Мы наблюдали эту особенность и при исследовании других задач [13,14], но при исследовании параллельной версии решения задачи о рюкзаке она проявилась очень заметно. Так, именно ею объясняется тот факт, что при параллельной работе алгоритма очень часто ускорение превышает (и весьма значительно) количество вычислительных модулей. Причиной, очевидно, является то, что одновременно исследуя несколько ветвей, мы почти всегда раньше находим достаточно хорошее решение, что позволяет отказаться от анализа большего количества неперспективных подзадач. Этот факт, на наш взгляд, является веским аргументом в пользу применения РП-программирования для решения задач такого рода.

Список литературы / References

[1] Гэри М., Джонсон Д., *Вычислительные машины и труднорешаемые задачи*, Мир, М., 1982; In English: Garey M.R., Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co, San Francisco, 1979.

[2] Kellerer H., Pferschy U., Pisinger D., *Knapsack Problems*, Springer, Berlin, 2004.

[3] Martello S., Toth P., *Knapsack Problems Algorithms and Computer Implementation*, Wiley, New York, 1990.

[4] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К., *Алгоритмы: построение и анализ*, Вильямс, М., 2006; In English: Cormen T., Leiserson Ch., Rivest R., Stein C., *Introduction to Algorithms*, The MIT Press, 2001.

[5] Land A. H., Doig A. G., "An Autmatic Method of Solving Discrete Programming Problems", *Econometrica*, **28** (1960), 497–520.

[6] Pisinger D., "An expanding-core algorithm for the exact 0-l knapsack problem", *Eur. J. Oper. Res.*, **87**:1 (1995), 175–187.

[7] Pisinger D., "A fast algorithm for strongly correlated knapsack problems", *Discrete Applied Mathematics*, **89**:1–3 (1998), 197–212.

[8] Dantzig G. B., "Discrete Variable Extremum Problems", *Operation Ressearch*, **5** (1957), 266–277.

[9] Посыпкин М.А., Сигал И.Х., "Комбинированный параллельный алгоритм решения задачи о ранце", *Труды четвертой международной конференции "Параллельные вычисления и задачи управления"*, 2008, 177–189; [Posypkin M.A., Sigal I.Kh., "Kombinirovannyj parallelnyj algoritm reshenija zadachi o rance", *Proceedings of the Fourth International Conference "Parallel Computations and Control Problems"*, 2008, 177–189, (in Russian).]

[10] Васильчиков В.В., *Средства параллельного программирования для вычислительных систем с динамической балансировкой загрузки*, Ярославль, 2001; [Vasilchikov V.V., *Sredstva parallelnogo programmirovaniya dlya vychislitelnykh sistem s dinamicheskoy balansirovkoy zagruzki*, Yaroslavl, 2001, (in Russian).]

[11] Васильчиков В.В., *Коммуникационный модуль для организации полносвязного соединения компьютеров в локальной сети с использованием .NET Framework*, Свидетельство о государственной регистрации программы для ЭВМ № 2013619925, 2013; [Vasilchikov V.V., *Kommunikatsionnyy modul dlya organizatsii polnosvyaznogo soedineniya kompyuterov v lokalnoy seti s ispolzovaniem .NET Framework*, Svidetelstvo o gosudarstvennoy registratsii programmy dlya EVM № 2013619925, 2013, (in Russian).]

[12] Васильчиков В.В., *Библиотека поддержки рекурсивно-параллельного программирования для .NET Framework*, Свидетельство о государственной регистрации программы для ЭВМ № 2013619926, 2013; [Vasilchikov V. V., *Biblioteka podderzhki rekursivno-parallelnogo programmirovaniya dlya .NET Framework*, Svidetelstvo o gosudarstvennoy registratsii programmy dlya EVM № 2013619926, 2013, (in Russian).]

[13] Васильчиков В.В., "О поддержке рекурсивно-параллельного программирования в .NET Framework", *Модел. и анализ информ. систем*, **21**:2 (2014), 15–25; [Vasilchikov V. V., "On the Recursive-Parallel Programming for the .NET Framework", *Modeling and Analysis of Information Systems*, **21**:2 (2014), 15–25, (in Russian).]

[14] Васильчиков В.В., "Об оптимизации и распараллеливании алгоритма Литтла для решения задачи коммивояжера", *Модел. и анализ информ. систем*, **23**:4 (2016), 401–411; [Vasilchikov V. V., "On Optimization and Parallelization of the Little Algorithm for Solving the Travelling Salesman Problem", *Modeling and Analysis of Information Systems*, **23**:4 (2016), 401–411, (in Russian).]

Vasilchikov V. V., "On a Recursive-Parallel Algorithm for Solving the Knapsack Problem", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 155–164.

DOI: 10.18255/1818-1015-2018-2-155-164

Abstract. In this paper, we offer an efficient parallel algorithm for solving the NP-complete Knapsack Problem in its basic, so-called 0-1 variant. To find its exact solution, algorithms belonging to the category "branch and bound methods" have long been used. To speed up the solving with varying degrees of efficiency, various options for parallelizing computations are also used. We propose here an algorithm for solving the problem, based on the paradigm of recursive-parallel computations. We consider it suited well for problems of this kind, when it is difficult to immediately break up the computations into a sufficient number of subtasks that are comparable in complexity, since they appear dynamically at run time. We used the RPM_ParLib library, developed by the author, as the main tool to program the algorithm. This library allows us to develop effective applications for parallel computing on a local network in the .NET Framework. Such applications have the ability to generate parallel branches of computation directly during program execution and dynamically redistribute work between computing modules. Any language with support for the .NET Framework can be used as a programming language in conjunction with this library. For our experiments, we developed some C# applications using this library. The main purpose of these experiments was to study the acceleration achieved by recursive-parallel computing. A detailed description of the algorithm and its testing, as well as the results obtained, are also given in the paper.

Keywords: Knapsack Problem, parallel algorithm, recursion, .NET

On the authors:
Vladimir V. Vasilchikov, orcid.org/0000-0001-7882-8906, PhD,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., 14, Yaroslavl, 150003, Russia, e-mail: vvv193@mail.ru

Acknowledgments:
This work was supported by initiative program VIP-004 (state registration number AAAA-A16-116070610022-6).

Моделирование и анализ информационных систем. Т. 25, № 2 (2018), с. 165–173
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 165–173

©Yauhen Klimiankou, 2017

DOI: 10.18255/1818-1015-2018-2-165-173

UDC 004.051; 004.451.35; 004.451.46

Measuring Overhead of Concurrency and Virtual Memory

Yauhen Klimiankou

Received November 7, 2017

Abstract. We present the methodology, as well as results of measurements and evaluation of overhead created by concurrency and virtual memory. A special measurement technique and testbed were used to obtain the most accurate data from the experiments. This technique is focused on the measurements of the overall performance degradation that is introduced by concurrency in the form of lightweight user-level threads on IA-32 processors. We have obtained and compared results of the experiments in an environment with and without enabled virtual memory to understand what loss of performance is caused by virtual memory in itself, and how it affects the overhead associated with concurrency. The results showed that overhead of concurrency outweighs virtual memory overhead and that there is a complex dependency between them. The article is published in the author's wording.

Keywords: virtual memory, concurrency, overhead, performance, measurements

For citation: Yauhen Klimiankou, "Measuring Overhead of Concurrency and Virtual Memory ", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 165–173.

On the authors:
Yauhen Klimiankou, orcid.org/0000-0001-7449-7986, MSc, PhD student
Belarusian State University of Informatics and Radioelectronics,
6 P. Brovki Street, Minsk 220013, Belarus, e-mail: klimenkov@bsuir.by

1. Introduction

In this paper we describe a work motivated by the desire to understand the influence of concurrency and virtual memory to the performance of the system. To understand that, we have measured the overhead created by the multithreading in its lightest-weight form called fibers. Fibers are used to provide concurrency on the application level. Additionally, we wanted to find out how the virtual memory affects performance of both concurrent and serialized workloads.

Accurate and fine-grained performance measurements on the low level of the system is challenging and requires taking into account a number of features of system and CPU. We have used a benefit of having of our own research operating system to create the вЂњClean RoomвЂќ environment, in which we have eliminated interference of the measurements with other activities asynchronously going in the system.

2. Related Work

There are a variety of works done to understand a context switch overhead, yet none of them has considered concurrency in general on the modern CISC processors. It is hard to find the published results of actual measurements of virtual memory overhead. Moreover, interdependence between concurrency and virtual memory overheads was not investigated.

The first research in that field was done by Agarwal et al. [1], where it demonstrated that multiprogramming activity significantly degrades cache performance. After that, Mogul advanced and widened original research [8]. In both cases results were achieved through simulation but not actual experiments and concurrency were considered in the heavyweight form – multiprocessing. Additionally, CPUs have significantly advanced during the last two decades. There is a concern that both works do not reflect the behavior of the modern CPUs.

McVoy et al. measured the cost of context switch between multiple processes using lmbench [7]. While this work provides results of experiments done in real-world environments, it considers multiprocessing rather than concurrency in general and accounts not only CPU but also OS overhead.

Li et al. has conducted a research similar to our work [5]. He has quantified context switch cost, but like in previous cases, he has focused on Linux multiprocessing, and had not cut off overhead created by contention on memory bus.

Finally, the David et al. [2] has considered concurrency in general instead of multiprocessing. But his measurements were performed on RISC processor (ARM), while we were interested in understanding of concurrency overhead created on advanced and full-featured CISC processors (IA-32).

We are not aware about any good paper with evaluation of virtual memory overhead. It is interesting, but the only paper that sheds some light to this issue is an overview of Singularity OS [4]. But even it considers entire overhead created by virtual memory and does not dig into details.

3. Measurement Methodology

Our measurement methodology was designed to provide the cleanest and accurate results with a fine-grained resolution. We have eliminated interference with interrupt handling and operating system kernel scheduler on macro-level. Furthermore, we have taken into account potential disturbance of the branch prediction, influence of the measurements on the instrumented code execution time and potential issues related to caching on micro-level.

3.1. System Structure

Experimental setup is depicted on Figure 1. We have used an advantage of having our own experimetal operating system to create testbed. In particular, the OS loader was modified by injections of three code modifications.

One of the injections is used to disable multiprocessor boot. By this, influence of the contention on memory bus to the results of measurements was eliminated. Despite the

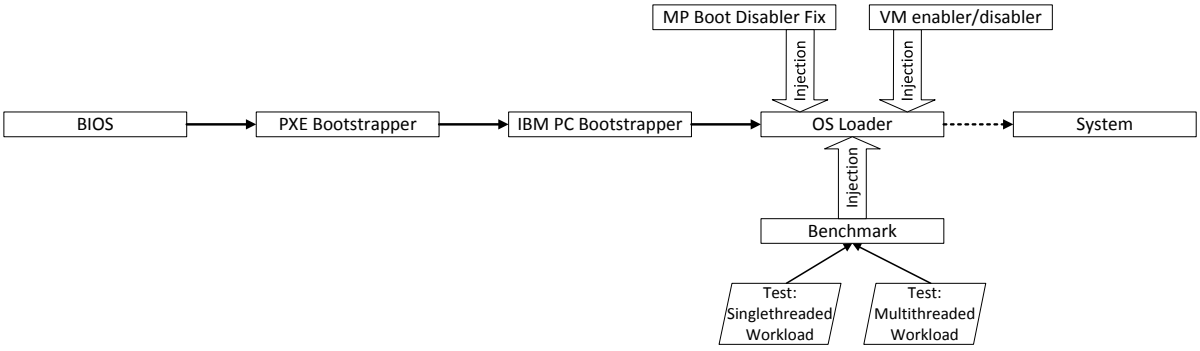

Fig 1. Experiment setup

fact that computer system equipped by two CPUs was used during performance tests, only one CPU was bootstrapped and used. The second one was leaved in the uninitialized state. In addition, the Hyper-Threading technology was disabled through the BIOS for the time of all experiments.

The second code injection has added the switch which controls enabling of the virtual memory. Availability of this switch has allowed to consider influence of virtual memory to the system performance.

Finally, the last code injection contains microbenchmark with two test cases. The first test case reproduces batched singlethreaded processing of workload. In the second scenario, the same workload is processed concurrently by two user-level threads. Benchmark finishes its work by halting the future loading of the operating system, which is unneeded and unsafe.

Benchmark applications use the same preallocated buffer in both test scenarios.

3.2. Benchmark Implementation

Benchmark starts work by disabling interrupts. By this, we eliminate potential interference of measurements with interrupt handling, which can affect accuracy of the measurement results.

Before the start of measurements, user-level multithreading environment is initialized. In our experiments only two threads are in use. Thus thread switch pointer should be set up and two stacks should be allocated and prepared. Each user-mode thread uses its own "top of the stack" pointer, which is also initialized during preparation to measurements, as well as initial stack frame.

Measurements is taken for a number of different workload sizes starting from 0 and ending by 24MB. Due to this benchmark performs a number of iterations of measurements, each time doubling the workload (0, 1b, 2b, 4b, ..., 24MB). On each iterations two experiment scenarios are executed: singlethreaded and multithreaded and, therefore, two measurements are taken. Before each experiment, cache warmup is performed by run of few experiments which execution time is not accounted for in measurement results. Finally, two experiment scenarios are played.

During experiment we emulated work of the application with variable working sets. Two buffers of the same fixed size were used to emulate working sets. The goal of the application was zeroing of buffers. Each scenario was replayed 100000 times.

In the singlethreaded scenario application sequentially refills first buffer by zeroes. In such a way it emulates the first program working with data in its working set. After the first buffer will be refilled 100000 times, application repeats the same procedure for the second buffer, emulating the second sequentially executing program.

In the multithreaded scenario, both emulated applications execute concurrently. After each buffer refill context switch to another application is performed. This process continues until both buffers will be refilled 100000 times.

3.3. User-Level Threads Implementation

The primary goal of our experiments is to understand the impact of concurrency to the performance, but not the impact of some specific implementation of concurrency. Due to this the lightest form of multithreading was chosen.

Implemented multithreading environment supports only two user-level threads and is based on cooperative form of multitasking. As a result, it does not require presence of scheduler. Additionally, run queue has degraded to the switch point, because we always have a system in a state where one thread is running and one thread is in ready state. Hence, context switch can be considered as a simple swapping of the roles/states between these two threads. Due to this environment maintains minimal global state consisting only of one variable, which stores stack pointer of the thread that is in the ready state. State of the user-level threads includes only state of eight general purpose registers of IA-32 processor. During context switch, outgoing thread stores its state on the top of its own stack. Then it swaps content of the global state variable and stack pointer register of processor. Finally, incoming thread loads its state from the top of its own stack.

The function for the user-level thread switch is implemented as follows:

```
; ECX - address of global state
SwitchThread:
    pushad
    mov    EAX , [ECX]
    mov    [ECX] , ESP
    mov    ESP , EAX
    popad
    ret
```

3.4. Measurement Methodology and Points

The primary tool which was used for execution time measurements is a IA-32 time stamp counter. Time stamp counter provides a highest resolution with granularity equal to a tick of system bus. It was safe to use time stamp counter in our experiments, because we had full control over CPU and were assured that thread migration to another CPU is prevented.

To reduce the effects of the measurements on the results, the number of the measurements taken was minimized. Furthermore, all measurements were done using the same framework, where only test scenarios were replaced. As a result the difference in measured time for two experiments will show only difference in execution time of test scenarios. Framework and test scenarios are depicted on Figure 2.

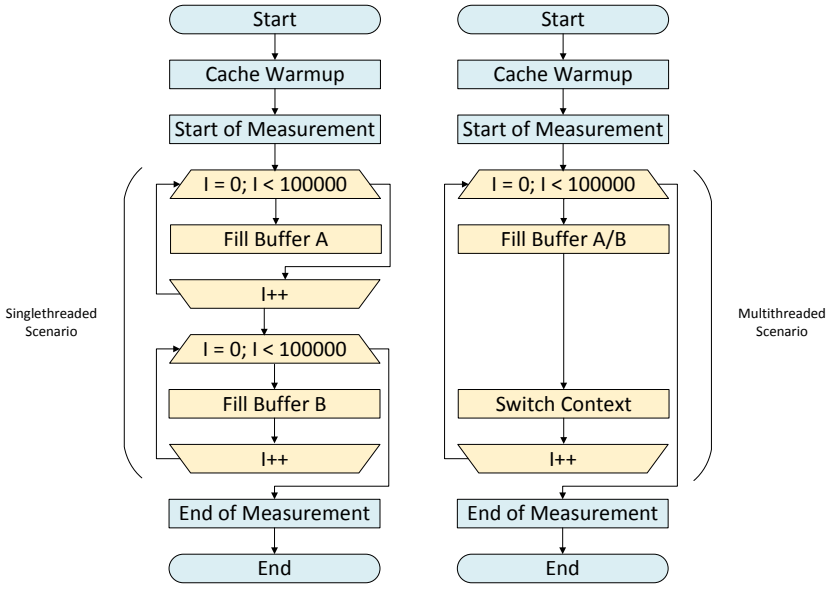

Fig 2. Experiment Scenarios

In our approach we have taken into account all effects of advanced processor features such as superscalarity, branch prediction and out-of-order execution, because we wanted to know the full impact of multithreading and virtual memory on the performance. Due to this the overall time of the entire series of 100000 experiments was measured. Note that in multithreaded experiment scenario two iteration counters were actually used, one per each thread stack. Thus, the entire experiment actually performs 200000 buffer refills, 100000 per each buffer.

3.5. Experimental Platform

All measurements were done on the Dell PowerEdge 2650 server [3] with two Intel Xeon CPUs [6] (Details are provided in Table 1).

Table 1. Experiment platform

CPU	Intel® Xeon® Processor
Microarchitecture	NetBurst
Codename	Prestonia
Frequency	2.4 GHz
TLB	64 + 64 entries
L1 code cache	12 K-Ojop 8-way set associative
L1 data cache	8 KByte 4-way set associative 64 byte line size
L2 cache	512 KByte 8-way set associative 64 byte line size

4. Results

Figures 3 and 4 visualize the measured overhead imposed by concurrency in the form of light-weight user-level multithreading as discussed in section 3.3. Both figures contain two graphs that show overhead measured in the environment with enabled and disabled virtual memory. Figure 3 represents an absolute overhead measured during a series of experiments and Figure 4 shows the same results, but in a form of relative overhead.

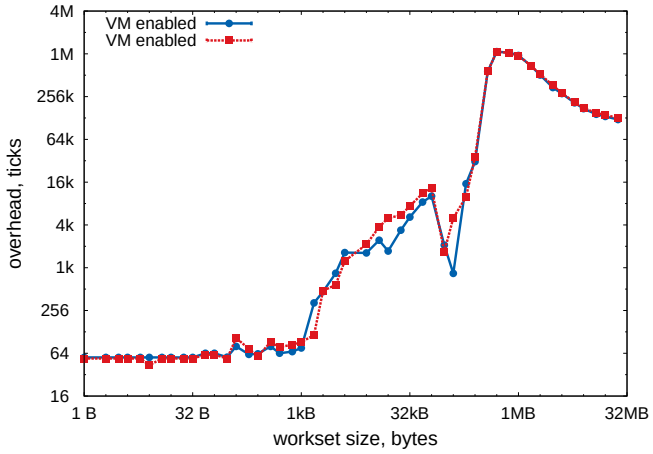

Fig 3. Absolute overhead of multithreading

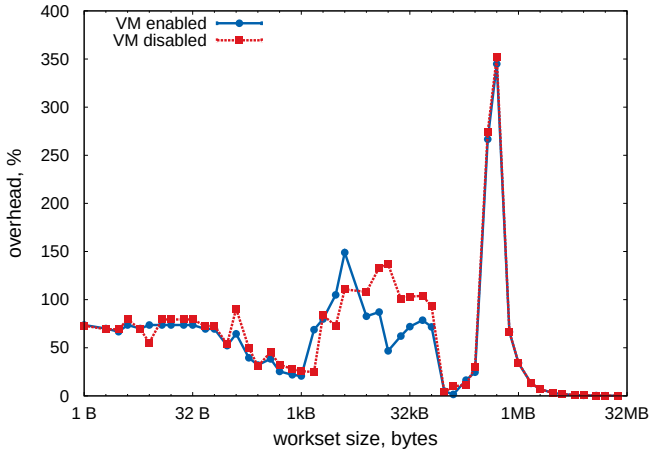

Fig 4. Relative overhead of multithreading

As reader can see, concurrency always creates performance penalty which is in the worst case raises up to almost 1.125 millions of wasted CPU ticks per time quantum used by thread or 352% of overhead. This worst case represents the scenario with working set of size 0.5MB, which is a size of L2 cache in our experimental setup. In the case of absence of concurrency entire working set is loaded into the cache once, where it is processed multiple times. But in the case of concurrency entire cache is considered as completely polluted and thus is constantly reloaded after each context switch. In general, overhead is significant and reaches acceptable levels only for working set sizes of near 128KB and bigger than 2MB.

Figure 5 shows how enabled virtual memory affects the overhead imposed by concurrency. In most cases it reduces performance penalty created by concurrency. In some cases even significantly (up to -90% for the case of working set of 16KB). In other cases, enabled virtual memory can boost concurrency overhead (up to +44% for the case of working set of 1.5KB). It would be interesting to note that influence of enabled virtual memory on concurrency overhead for the scenarios with working sets bigger than 0.5MB is negligible.

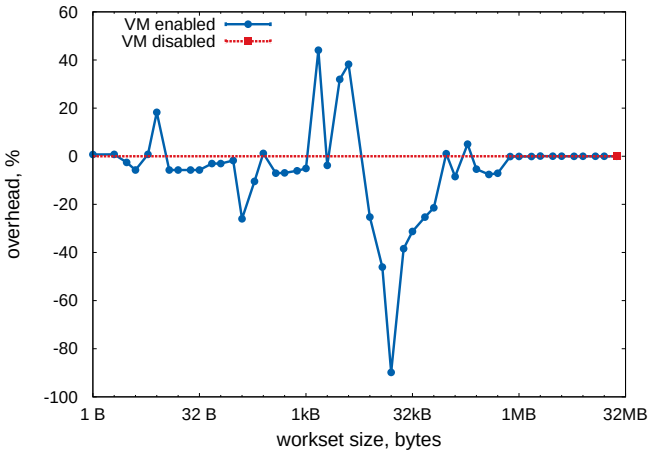

Fig 5. Impact of virtual memory onto the concurrency overhead

Another aspect discussed here is how the enabled virtual memory affects performance of the application and concurrency overhead. Figures 6 and 7 shows the results of measurements of overhead for enabled virtual memory. As in the previous case, Figure 6 represents performance penalty in absolute values, when Figure 7 represents the same results but as relative overhead.

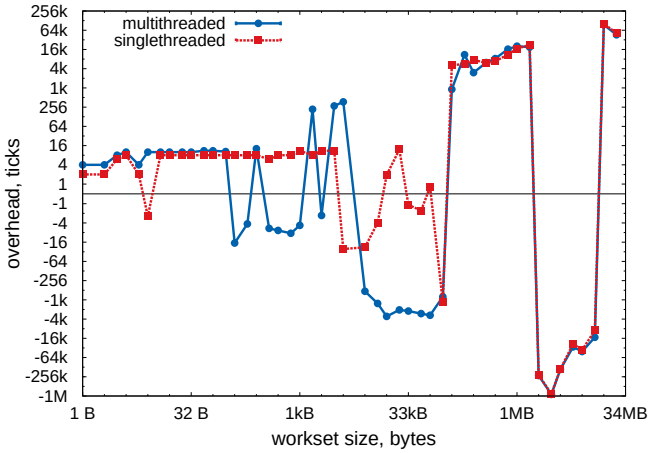

Fig 6. Absolute overhead of virtual memory

It is interesting to note that in contrast to the concurrency, impact of the virtual memory to the application performance is not trivial. In most cases enabled virtual memory degrades performance of the application, or its impact is negligible. But actually,

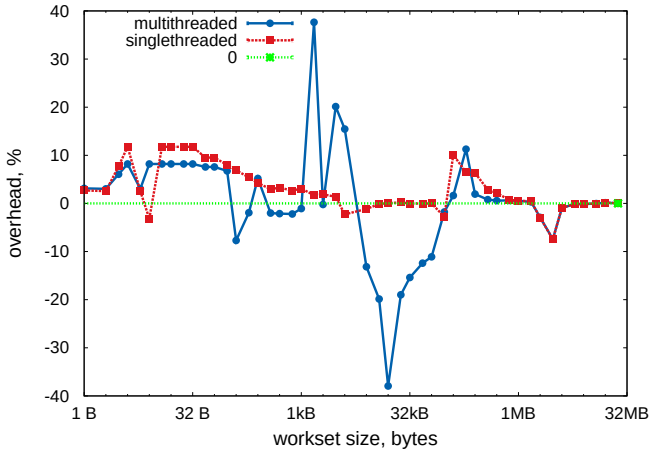

Fig 7. Relative overhead of virtual memory

we can see the cases when virtual memory boosts application performance. What is even more interesting, that boost is significant for the concurrent workload (up to 39% for 16KB working set) when it negligible for the serialized workload (up to 7.5% for 3MB working set). Impact of concurrency onto virtual memory overhead is demonstrated on Figure 8.

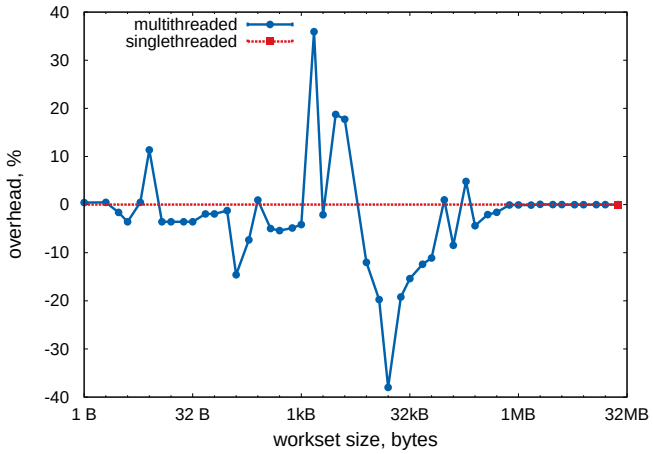

Fig 8. Impact of concurrency on virtual memory overhead

5. Conclusions and Outlook

We presented the results of the measurements of the overhead incurred by concurrency and virtual memory. During measurements special focus was pointed to achievement of the most accurate and fine-grained results. Due to this the “Clean room” methodology was used.

We have demonstrated that concurrency creates significant overhead which can grow up to 350% in a terminal case. Reason for this is an advanced multilevel caching used

by CPUs to hide the cost of access to main memory and by cache pollution introduced by concurrency.

In contrast, impact of virtual memory on the system performance is not trivial. Furthermore, it largely depends on the type of workload. In particular, we have found an interesting phenomena when virtual memory even significantly boosts performance of concurrent workload (up to 39%).

References

[1] A. Agarwal, J. Hennessy, M. Horowitz, "Cache Performance of Operating System and Multiprogramming Workloads", *ACM Trans. Comput. Syst.*, **6**:4 (1988), 393–431.

[2] F. M. David, J. C. Carlyle, R. H. Campbell, "Context Switch Overheads for Linux on ARM Platforms", *Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS'07*, ACM, New York, NY, USA, 2007.

[3] *INFOBrief: Dell PowerEdge 2650*, Dell Inc., 2004.

[4] G. Hunt, J. Larus, "Singularity: Rethinking the Software Stack", *ACM SIGOPS Operating Systems Review*, **41**:2 (2007), 37–49.

[5] C. Li, C. Ding, K. Shen, "Quantifying the Cost of Context Switch", *Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS'07*, ACM, New York, NY, USA, 2007.

[6] *IA-32 Intel® Architecture Software Developer's Manual. Volume 3: System Programming Guide*, Intel Corporation, 2002 (245472-007).

[7] L. McVoy, C. Staelin, "Lmbench: Portable Tools for Performance Analysis", *Proceedings of the USENIX Annual Technical Conference*, San Diego, California, USA, January 22–26, 1996, 279–294.

[8] J. C. Mogul, A. Borg, "The Effect of Context Switches on Cache Performance", *SIGPLAN Not.*, **26**:4 (1991), 75–84.

Клименков Е. И., "Измерение накладных расходов на параллелизм и виртуальную память", *Моделирование и анализ информационных систем*, **25**:2 (2018), 165–173.

DOI: 10.18255/1818-1015-2018-2-165-173

Аннотация. В данной статье представляется методология и результаты измерений и оценки накладных расходов, связанных с параллелизмом и виртуальной памятью. Для получения наиболее точных экспериментальных данных использовалась специальная методика измерений. Данная методика сфокусирована на измерениях совокупных потерь производительности, создаваемых параллелизмом, выраженным в форме легковесных потоков пользовательского режима на процессорах с архитектурой IA-32. Были получены и проанализированы данные, произведенные в средах с виртуальной памятью и без нее. Таким образом стало известно, какая потеря производительности вызывается виртуальной памятью, а также то, как она влияет на накладные расходы, связанные с параллелизмом. Эксперименты показали, что накладные расходы на параллелизм гораздо существеннее накладных расходов на виртуальную память. И тем не менее, между ними существует сложная взаимозависимость. Статья публикуется в авторской редакции.

Ключевые слова: виртуальная память, параллелизм, накладные расходы, измерения

Об авторах:
Клименков Евгений Иванович, orcid.org/0000-0001-7449-7986, магистр техн. наук, аспирант,
Белорусский государственный университет информатики и радиоэлектроники,
ул. П. Бровки, 6, г. Минск, 220013 Республика Беларусь, e-mail: klimenkov@bsuir.by

Моделирование и анализ информационных систем. Т. 25, №2 (2018), с. 174–192
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 174–192

Сети Петри
и временные автоматы

©Глонина А. Б., Балашов В. В., 2017

DOI: 10.18255/1818-1015-2018-2-174-192

УДК 519.7

О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов

Глонина А. Б., Балашов В. В.[1]

получена 2 ноября 2017

Аннотация. Рассматривается задача проверки допустимости конфигураций модульных вычислительных систем реального времени (МВС РВ). Конфигурация считается допустимой, если все работы успевают выполниться на МВС РВ в рамках своих директивных интервалов. Предложена обобщенная модель функционирования МВС РВ и метод построения на её основе модели для конкретной конфигурации. Модель представляет собой сеть временных автоматов с остановкой таймеров. По вычислению сети автоматов предлагается строить временную диаграмму (ВД) функционирования МВС РВ, необходимую для проверки допустимости. В работе обосновывается корректность предложенного подхода. Из спецификаций на МВС РВ был выделен ряд требований, применимых к моделям МВС РВ и их компонентов на выбранном уровне абстракции. Модели считаются корректными, если удовлетворяют этим требованиям. Доказано, что если все модели компонентов системы удовлетворяют соответствующим требованиям, то модель МВС РВ, построенная согласно предложенному подходу, удовлетворяет требованиям к модели системы в целом (то есть корректна), а также является детерминированной. Под детерминированностью понимается однозначность построения ВД сети автоматов, соответствующей заданной конфигурации. Это позволяет использовать для проверки допустимости конфигурации любое вычисление соответствующей сети автоматов, что крайне важно для эффективности предложенного подхода, так как число возможных вычислений сети автоматов растет экспоненциально с числом работ в системе. Выполнение требований корректности к моделям компонентов системы может быть проверено автоматически с использованием верификатора и подхода автоматов-наблюдателей. Все разработанные нами модели компонентов системы удовлетворяют соответствующим требованиям, что было доказано с помощью верификатора UPPAAL. Если пользовательские модели компонентов системы удовлетворяют требованиям корректности, то они могут быть включены в модель МВС РВ, которая при этом останется корректной и детерминированной.

Ключевые слова: моделирование, верификация, интегрированная модульная авионика, планирование вычислений

Для цитирования: Глонина А. Б., Балашов В. В., "О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов", *Моделирование и анализ информационных систем*, **25**:2 (2018), 174–192.

Об авторах: Глонина Алевтина Борисовна, orcid.org/0000-0001-8716-4128, программист, Московский государственный университет им. М.В. Ломоносова, Ленинские горы, д. 1, г. Москва, 119991 Россия, e-mail: alevtina@lvk.cs.msu.su; Балашов Василий Викторович, orcid.org/0000-0001-5211-805X, ст. науч. сотр., канд. физ.-мат. наук, Московский государственный ун-т им. М.В. Ломоносова, Ленинские горы, д. 1, г. Москва, 119991 Россия, e-mail: hbd@cs.msu.su

Благодарности:
[1] Работа выполнена при финансовой поддержке РФФИ (грант № 17-07-01566).

Введение

В настоящее время одним из перспективных типов архитектур встроенных вычислительных систем реального времени является модульная архитектура. В данной статье в качестве примера модульных вычислительных систем реального времени (МВС РВ) рассматриваются системы интегрированной модульной авионики (ИМА) [1].

МВС РВ, в том числе системы ИМА, состоят из стандартизированных вычислительных модулей, связанных между собой коммутируемой сетью с виртуальными каналами. Каждый модуль содержит набор многоядерных процессоров. В МВС РВ могут входить процессоры различных типов, различающихся производительностью.

Рабочая нагрузка представлена набором разделов. Разделом называется группа логически связанных задач, взаимодействующих посредством общей памяти. Как правило, один раздел соответствует одному приложению. Все задачи являются периодическими: за период должен выполниться один экземпляр задачи, называемый работой. Между задачами с одинаковым периодом могут существовать зависимости по данным: очередная работа задачи-получателя не может начать выполнение до тех пор, пока не получит данные от всех соответствующих работ задач-отправителей. Для каждой задачи задано время выполнения на процессоре каждого типа.

Каждый раздел привязан к вычислительному ядру в составе процессора. Несколько разделов могут быть привязаны к одному и тому же ядру. На интервале планирования для вычислительного ядра задается статическое расписание окон, то есть отрезков времени, в течение каждого из которых могут выполняться работы определенного раздела. Каждый раздел имеет собственный планировщик, управляющий постановкой на выполнение работ внутри окон раздела. Допускается использование разных алгоритмов планирования в разных разделах. У каждой работы существует директивный интервал, определяющийся периодом задачи, а также дополнительными параметрами. Работа должна завершить выполнение не позднее определенной для нее правой границы директивного интервала. Если граница директивного интервала достигается до завершения выполнения работы, то работа считается опоздавшей и не может далее выполняться на вычислительном ядре. Таким образом, время, которое опоздавшая работа находилась на вычислительном ядре, меньше заданного времени, необходимого для ее выполнения.

Конфигурация МВС РВ определяет набор ее модулей, характеристики рабочей нагрузки, привязку разделов к ядрам и расписания окон для каждого ядра. Конфигурация МВС РВ называется допустимой, если для нее в процессе функционирования МВС РВ ни одна из работ не становится опоздавшей. При проектировании МВС РВ анализируется ряд потенциальных конфигураций, для которых необходима проверка допустимости.

Одним из методов проверки допустимости является построение временной диаграммы (ВД) функционирования системы. ВД содержит события постановки на выполнение, вытеснение и завершение работ. Каждое событие характеризуется типом, идентификатором работы-источника и временной меткой. ВД может быть получена в результате прогона имитационной модели МВС РВ. В работе [2] была предложена обобщенная модель функционирования МВС РВ, а также алгоритм построения на

ее основе модели МВС РВ заданной конфигурации. В основе данной обобщенной модели лежит математический аппарат сетей временных автоматов с остановкой таймеров, позволяющий формально проверить выполнение ряда требований к моделям.

В данной статье предложен подход к обоснованию корректности и детерминированности класса моделей, синтезируемых на основе обобщенной формальной модели. С помощью данного подхода доказана корректность и детерминированность моделей, построенных авторами.

1. Обобщенная модель функционирования МВС РВ

1.1. Сети временных автоматов с остановкой таймеров

В работе [2] был сформулирован ряд требований к математическому аппарату для моделирования МВС РВ, и в соответствии с требованиями выбран аппарат сетей временных автоматов с остановкой таймеров [3].

Автомат с остановкой таймеров представляет собой конечный автомат с целочисленными переменными и таймерами. Графически такой автомат может быть представлен в виде размеченного ориентированного графа, вершины которого называются локациями, а дуги — переходами. Таймер — это специальная переменная, принимающая вещественные значения. Таймер считается активным, если его условие активности (булево выражение над множеством переменных) в текущей локации автомата истинно. В противном случае таймер считается остановленным.

Каждой локации сопоставлен инвариант — булево выражение над таймерами и переменными. Автомат может оставаться в данной локации до тех пор, пока инвариант этой локации имеет истинное значение.

Каждый переход характеризуется условием перехода, действиями над переменными и таймерами и, возможно, синхронизацией с другими автоматами. Переход активен, если автомат находится в локации, предшествующей этому переходу, значения переменных и таймеров удовлетворяют условию перехода, и инвариант локации-назначения перехода имеет истинное значение. Переход может (но не обязан) быть совершен, если он активен и может быть выполнена синхронизация. При совершении перехода меняется текущая локация автомата, происходит синхронизация и выполняются действия по изменению переменных и таймеров.

Состоянием автомата называется совокупность его текущей локации, значений переменных и таймеров. Состояние автомата может измениться не только в результате выполнения перехода, но и в результате увеличения значений всех таймеров на одно и то же значение. На рисунке 1 приведен пример автомата, моделирующего планировщик раздела.

Сеть автоматов представляет собой набор автоматов, функционирующих совместно и взаимодействующих посредством общих переменных и каналов. Набор переменных и каналов, посредством которых данный автомат взаимодействует с другими автоматами сети, называется *интерфейсом автомата*.

Канал — средство синхронизации автоматов. Существует два парных действия синхронизации: отправка и прием сигнала по каналу. Если в некоторый момент

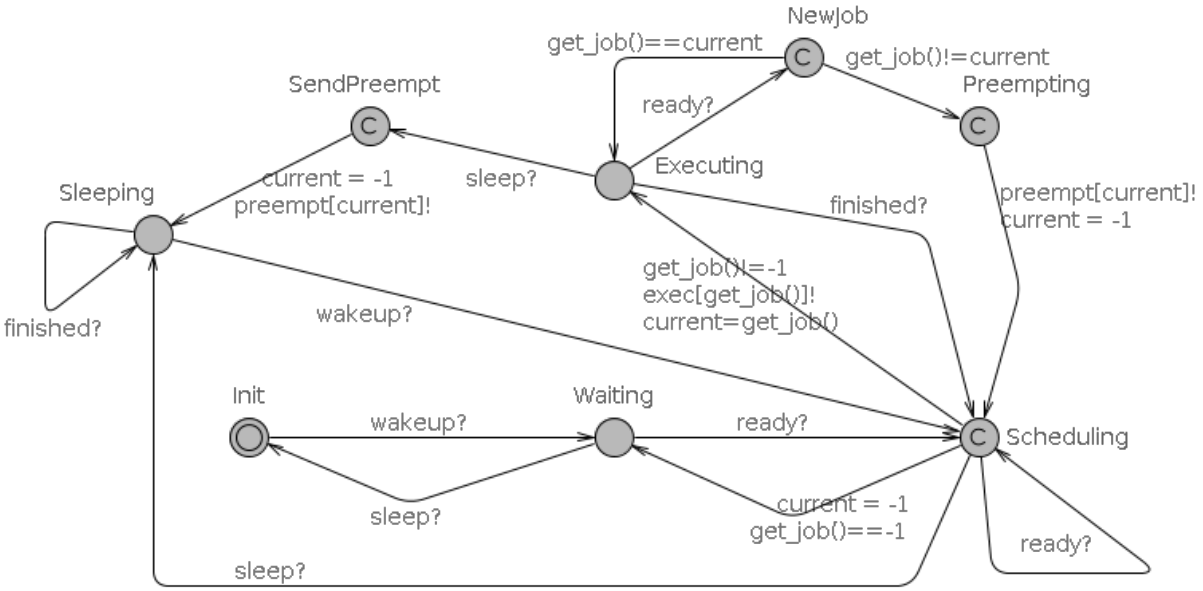

Рис. 1. Пример автомата, моделирующего планировщик раздела
Fig. 1. An automaton example. Partition scheduler model

времени в двух автоматах сети активны переходы с парными действиями синхронизации по одному и тому же каналу, то эти переходы выполняются одновременно. Каналу может быть назначен приоритет. Если в некоторый момент времени в сети автоматов возможны синхронизации по двум различным каналам, то выполняется синхронизация по каналу с высшим приоритетом.

Конечная или бесконечная последовательность состояний, соответствующая некоторому сценарию функционирования сети автоматов, называется *вычислением*.

Для описания предложенной модели первым из соавторов был введен ряд дополнительных определений [2].

Модельное время сети автоматов — значение некоторого служебного таймера, условие активности которого всегда истинно и который никогда не обнуляется.

Событие синхронизации автоматов — это тройка $\langle$ канал, набор автоматов-участников синхронизации, модельное время $\rangle$. *Временная диаграмма сети автоматов* — набор событий синхронизации, соответствующий некоторому вычислению данной сети.

Параметризованный автомат — это автомат, в арифметических и логических выражениях которого наряду с числами и переменными используются параметры с незаданными значениями.

Базовый тип автоматов определяется только интерфейсом. Параметризованный автомат *реализует* базовый тип автоматов, если интерфейс параметризованного автомата содержит все каналы и переменные интерфейса базового типа, а также, возможно, другие каналы и переменные.

Введенные уровни абстракции автоматов соответствуют в рамках предлагаемой схемы моделирования уровням абстракции компонентов МВС РВ. Базовому типу автоматов соответствует базовый тип компонентов. Базовый тип компонента МВС РВ определяет сервис, предоставляемый компонентом системы. Примером базово-

го типа компонента является планировщик раздела. Параметризованному автомату соответствует конкретный тип компонента. Конкретный тип компонента МВС РВ определяет детали реализации базового типа. Примером конкретного типа компонента является планировщик раздела, работающий по алгоритму планирования с фиксированными приоритетами и вытеснением. Экземпляру автомата соответствует экземпляр компонента системы.

Набор базовых типов автоматов назовем *обобщенной сетью автоматов*. Набор параметризованных автоматов — *параметризованной сетью автоматов*.

1.2. Структура обобщенной модели функционирования МВС РВ

В работе [2] А. Глониной предложена обобщенная модель функционирования МВС РВ, представляющая собой обобщенную сеть временных автоматов с остановкой таймеров, состоящую из следующих базовых типов автоматов:

- CS, моделирующий планировщик ядра;
- TS, моделирующий планировщик раздела;
- T, моделирующий функциональную задачу;
- L, моделирующий виртуальный канал (ВК).

Структура предложенной обобщенной сети автоматов приведена на рисунке 2.

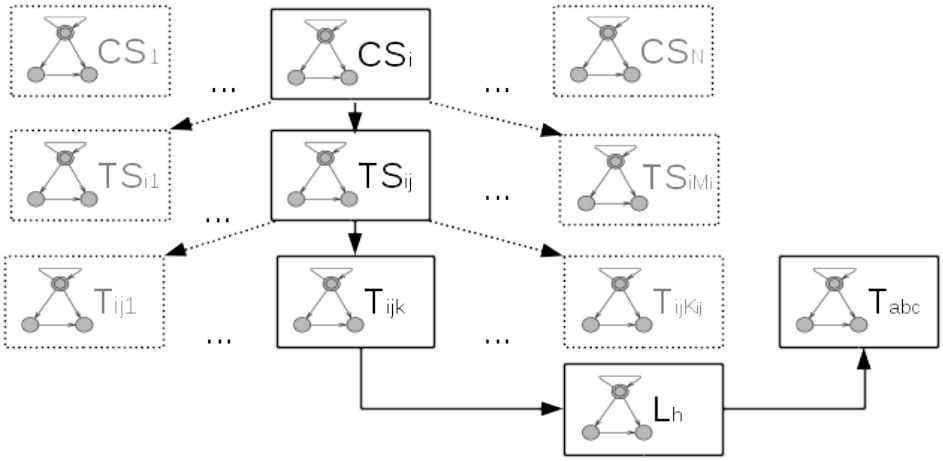

Рис. 2. Структура предложенной обобщенной сети автоматов
Fig. 2. The proposed general stopwatch automata network structure

Полное описание перечисленных базовых типов и сети приведено в [2]. Далее будут рассматриваться временные диаграммы сетей автоматов, представляющие собой наборы событий синхронизации, поэтому укажем каналы синхронизации, входящие в интерфейсы перечисленных базовых типов автоматов (далее под типом будем понимать тип автоматов):

- *exec*, *preempt* — каналы для взаимодействия базовых типов T и TS, синхронизация по которым моделирует соответственно постановку работы задачи на выполнение и ее вытеснение; каждый автомат, реализующий базовый тип T, использует собственные каналы *exec* и *preempt*; автомат, реализующий базовый тип TS, использует по n таких каналов, где n — количество задач раздела, планировщик которого моделируется этим автоматом;

- *ready*, $finished$ — каналы для взаимодействия базовых типов T и TS, синхронизация по которым моделирует соответственно готовность к запуску и завершение работы некоторой задачи раздела; автомат, реализующий базовый тип TS, использует по одному каналу *ready* и $finished$; все автоматы, реализующие базовый тип T и моделирующие задачи одного раздела, используют один и тот же канал *ready* и один и тот же канал $finished$;

- *wakeup*, *sleep* — каналы для взаимодействия базовых типов TS и CS, синхронизация по которым моделирует соответственно открытие и закрытие окна некоторого раздела; каждый автомат, реализующий базовый тип TS, использует собственные каналы *wakeup* и $finished$; автомат, реализующий базовый тип CS, использует по n таких каналов, где n — количество разделов вычислительного ядра, планировщик которого моделируется этим автоматом;

- *send*, *receive* — каналы для взаимодействия базовых типов T и L, синхронизация по которым моделирует соответственно отправку и прием сообщения; каждому автомату, реализующему базовый тип T, соответствует по одному каналу *send* и *receive*; при этом несколько автоматов, реализующих базовый тип L, могут использовать один и тот же канал *send* или *receive* (так моделируется отправка и получение работой задачи сообщений по нескольким виртуальным каналам).

Первым из соавторов была разработана параметризованная сеть автоматов, содержащая реализации перечисленных базовых типов. Эти реализации представляют собой модели планировщика ядра, функциональной задачи и виртуального канала, а также модели трех часто используемых в МВС РВ планировщиков разделов: с фиксированными приоритетами и вытеснением [4], с фиксированным приоритетами без вытеснения [5], планировщик, работающий по алгоритму EDF [6]. Одна из моделей планировщиков раздела приведена на рисунке 1. Также в [2] приведен предложенный А. Глониной алгоритм построения модели МВС РВ с конкретной конфигурацией на основе параметризованной модели по описанию конфигурации. Показано, что для заданных параметризованной модели и конфигурации модель МВС РВ строится однозначно.

Кроме того, в [2] описан разработанный первым автором алгоритм однозначного построения ВД МВС РВ по ВД ее модели: из ВД модели выбираются события синхронизации по каналам *exec*, *preempt* и $finished$, и каждому из выбранных событий однозначно ставится в соответствие событие ВД МВС РВ. Тип события в ВД МВС РВ (постановка на выполнение, вытеснение или завершение работы) определяется каналом синхронизации, идентификатор работы-источника — автоматом-участником синхронизации, временная метка — модельным временем.

Полученная ВД МВС РВ используется для проверки критерия допустимости конфигурации: зная для некоторой работы временные метки её постановок на выполнение, вытеснение и завершение, можно определить суммарное время её выполнения на вычислительном ядре. Если для каждой работы это время равно заданному в конфигурации времени выполнения работы на ядре процессора соответствующего типа, то конфигурация допустима. Иначе, если для некоторой работы суммарное время выполнения на ядре меньше указанного в конфигурации, то работа была снята с ядра по причине достижения границы директивного интервала, и, следовательно, конфигурация недопустима.

2. Корректность проверки допустимости конфигураций МВС РВ

Из раздела 1 следует, что для проверки допустимости некоторой конфигурации достаточно получить соответствующую ей ВД МВС РВ на интервале планирования. Следовательно, для обоснования корректности проверки допустимости достаточно обосновать корректность получаемых согласно предложенному методу ВД МВС РВ.

При проектировании МВС РВ отсутствует возможность экспериментального сравнения ВД, получаемых при прогоне моделей, с ВД функционирования целевых систем. Поэтому корректность всего класса ВД, получаемых согласно предложенному методу, необходимо доказать формально.

Введем понятия корректности модели МВС РВ и корректности ВД модели МВС РВ. Спецификации и стандарты на МВС РВ содержат ряд требований корректности к функционированию системы. ВД моделей содержат события, соответствующие событиям, происходящим в МВС РВ. Из спецификаций МВС РВ с архитектурой ИМА [4, 7] нами были выделены требования, применимые к ВД моделей, то есть представимые в виде ограничений на порядок происходящих в моделях событий и на длительность интервалов между ними. Кроме того, были выделены применимые к ВД моделей требования детерминированности, основанные на спецификациях МВС РВ и, как правило, использующиеся на этапе проектирования (например, в работах [8–11]). Примеры требований приведены ниже, в разделах 2.1.2. и 2.2. ВД модели МВС РВ является корректной, если для нее выполнены все выделенные требования. Модель МВС РВ будем считать корректной, если все возможные ее ВД корректны. Далее под требованием к модели будем понимать требование к ее ВД. Таким образом, модель МВС РВ корректна, если она удовлетворяет всем выделенным требованиям, то есть для всех ее ВД все выделенные требования выполняются.

Напомним, что модель МВС РВ с заданной конфигурацией строится автоматически по описанию этой конфигурации, на основе параметризованной сети автоматов, реализующей обобщенную модель МВС РВ. Обоснование корректности всех моделей, синтезируемых согласно предложенному методу, выполняется следующим образом. Для каждого базового типа автомата, формирующего обобщенную модель МВС РВ (то есть входящего в состав этой модели), на основе спецификаций МВС РВ [4,7,12] выделяется набор требований, которым должны удовлетворять все параметризованные автоматы, реализующие данный базовый тип. Далее доказывается,

что если для всех параметризованных автоматов, формирующих параметризованную модель МВС РВ, выполняются требования, сформулированные для соответствующих базовых типов, то параметризованная модель МВС РВ удовлетворяет всем выделенным требованиям корректности к модели в целом, а также является детерминированной. Под детерминированностью параметризованной модели понимается то, что ее экземпляру, построенному для заданной конфигурации, однозначно соответствует ВД. Детерминированность модели позволяет использовать любое вычисление сети автоматов для построения ВД.

Таким образом, после выполнения указанных доказательств для обоснования корректности ВД некоторой параметризованной сети автоматов, реализующей предложенную обобщенную сеть, достаточно проверить выполнение требований к формирующим её параметризованным автоматам. Проверка выполнения требований к параметризованным автоматам может быть проведена автоматически с помощью верификатора UPPAAL [13]. Для всех построенных автоматов такая проверка была выполнена. Чтобы при добавлении в параметризованную модель нового автомата все ее ВД оставались корректными, достаточно убедиться (с использованием верификатора), что все требования к соответствующему базовому типу автомата для данного автомата выполняются.

2.1. Проверка выполнения требований к моделям компонентов МВС РВ

Для каждого базового типа компонента, которому соответствует базовый тип автоматов, был выделен ряд требований. Для того чтобы любая модель МВС РВ, построенная согласно предложенному методу, была корректной, требуется, чтобы все входящие в нее параметризованные автоматы удовлетворяли требованиям, выделенным для соответствующего базового типа. Кроме того, для каждого конкретного типа компонента могут быть сформулированы дополнительные требования, которым должны удовлетворять параметризованные автоматы, моделирующие этот тип компонента.

2.1.1. Подход на основе автоматов-наблюдателей к верификации параметризованных автоматов

Выделенные требования к параметризованным автоматам представляют собой ограничения на последовательности событий синхронизации и на интервалы между ними. Поэтому для проверки их выполнения был выбран подход на основе автоматов-наблюдателей (observer automata) [14]. Опишем кратко этот подход.

Для проверки некоторого требования к параметризованному автомату строится сеть автоматов, состоящая из данного автомата и автомата-наблюдателя. Автомат-наблюдатель представляет собой автомат, имеющий интерфейс для взаимодействия с исходным автоматом. Во всех локациях автомата-наблюдателя активны переходы с действиями синхронизации, парными ко всем возможным действиям синхронизации исходного автомата. Автомат-наблюдатель имеет некоторую «плохую» локацию, такую что любая не удовлетворяющая требованию последовательность синхронизаций гарантированно приводит автомат-наблюдатель в «плохую» локацию.

Таким образом, проверка выполнения требования сводится к проверке недостижимости этой «плохой» локации. Отметим, что в общем случае задача проверки достижимости в сетях автоматов с остановкой таймеров алгоритмически неразрешима [3], однако в [15] и [16] показано, что для частного случая, соответствующего рассматриваемым в данной работе моделям (детерминированность планировщиков, использование заранее известных оценок времени выполнения работ и времени передачи сообщений), эта проблема разрешима. Кроме того, при использовании верификатора UPPAAL гарантируется [17], что процесс верификации завершается всегда (возможно, с неопределенным результатом). Таким образом, завершение верификации с результатом «плохая локация недостижима» говорит о выполнении требования, завершение верификации с результатом «плохая локация достижима» — о невыполнении требования, завершение верификации с неопределенным результатом — о том, что модель не может быть использована, так как нет подтверждения выполнения для нее требования корректности. В случае завершения верификации с неопределенным результатом, параметризованный автомат необходимо перестроить так, чтобы результат верификации был определен. Согласно [15] и [16], для рассматриваемого класса МВС РВ такое перестроение возможно.

Параметризованный автомат должен удовлетворять требованиям корректности при всех возможных значениях своих параметров. Поэтому в автомате-наблюдателе происходит недетерминированная инициализация параметров исходного автомата, и, таким образом, при верификации рассматриваются все возможные значения параметров из множества допустимых значений. Если хотя бы для одного набора значений параметров «плохая» локация достижима, то исходный автомат считается некорректным.

Кроме того, на порядок событий в исходном автомате могут влиять значения переменных, изменяемых другими автоматами (множество переменных для взаимодействия с другими автоматами определяется интерфейсом соответствующего базового типа автомата). Поэтому в автомате-наблюдателе происходит недетерминированное изменение значений таких переменных. Если хотя бы для одной последовательности значений переменных «плохая» локация достижима, то исходный автомат считается некорректным.

2.1.2. Проверка требований корректности к моделям компонентов МВС РВ

Рассмотрим пример построения автомата-наблюдателя для проверки выполнения приведенного ниже требования к параметризованным автоматам, реализующим базовый тип TS («планировщик раздела»). Один из таких параметризованных автоматов, моделирующий планировщик с фиксированными приоритетами и вытеснением, изображен на рисунке 1.

1. Планировщик раздела может поставить работу на выполнение только в рамках окна этого раздела.

Переформулируем это требование в терминах синхронизаций с участием автомата, моделирующего планировщик раздела. Окнами раздела являются интервалы между синхронизациями по каналам *wakeup* и *sleep*. Постановке на выполнение работы i-й задачи соответствует синхронизация по каналу $exec[i]$. Таким образом,

синхронизации по каналам $exec[i]$ должны происходить только между синхронизациями по каналам $wakeup$ и $sleep$. Последовательности синхронизаций, удовлетворяющие данному требованию, представляют собой циклически повторяющиеся последовательности вида:

1) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $wakeup$ и $exec[i]$ (события вне окна раздела);

2) синхронизация по каналу $wakeup$ (открытие окна);

3) некоторое (возможно, нулевое) количество синхронизаций по любым каналам (в том числе, возможно, $exec[i]$), кроме $sleep$ (события внутри окна);

4) синхронизация по каналу $sleep$ (закрытие окна).

Один цикл соответствует одному окну раздела.

Любая последовательность синхронизаций с участием модели планировщика раздела, для которой нарушается требование 1, имеет вид:

1) некоторое (возможно, нулевое) количество последовательностей вида, описанного выше (корректная отработка нескольких окон);

2) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $wakeup$ и $exec[i]$ (события вне очередного окна);

3) синхронизация по каналу $exec[i]$ (постановка некоторой работы на выполнение вне очередного окна);

Автомат-наблюдатель для проверки сформулированного требования приведен на рисунке 3.

При переходах между локациями $Init1$–$Init5$ происходит недетерминированная инициализация параметров автомата: выбирается количество задач (согласно стандарту ARINC 653 [4], раздел не может содержать более 64 задач), для каждой задачи выбирается приоритет (уникальность приоритета обеспечивается функцией $check\_prio$) и правая граница директивного интервала (левая граница не используется в моделях планировщиков). Если для верифицируемого параметризованного автомата известно, что в его системе переходов какие-то из параметров не используются, то соответствующий этап инициализации целесообразно пропустить для ускорения последующей верификации. Например, разработанная первым из соавторов модель планировщика с фиксированными приоритетами не использует границы директивных интервалов задач (за принудительное завершение опоздавших работ отвечают модели задач), а модель планировщика, работающего по стратегии EDF, не использует приоритеты.

Помимо локаций $Init1$–$Init5$, в автомате-наблюдателе присутствуют локации $Inactive$ и $Active$, соответствующие неактивному (вне окон раздела) и активному (внутри окон) состояниям планировщика, а также «плохая» локация $ERROR$. Из локаций $Inactive$ и $Active$ имеются переходы по всем определенным для базового типа TS действиям синхронизации. Также интерфейс базового типа TS содержит набор переменных $is\_ready[i]$. Поэтому из локаций $Inactive$ и $Active$ имеются переходы с недетерминированным изменением этих переменных. Переход из локации

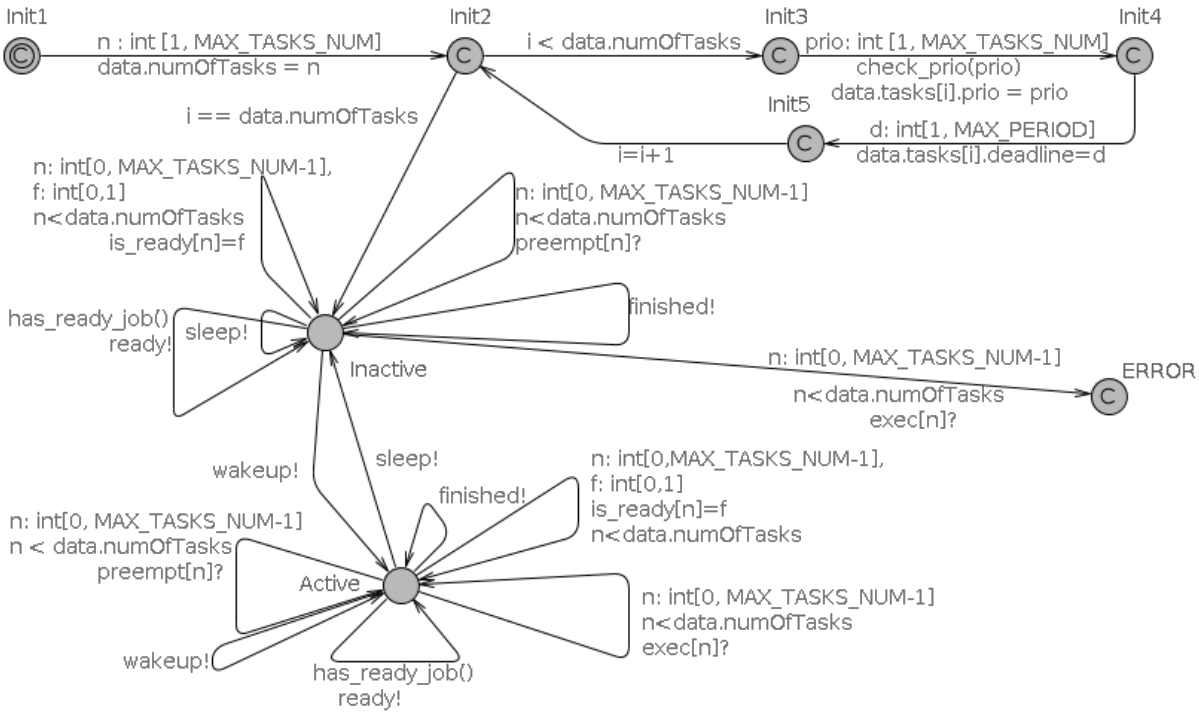

Рис. 3. Пример автомата-наблюдателя
Fig. 3. An observer automaton example

Inactive в локацию *Active* осуществляется при выполнении синхронизации по каналу *wakeup*, из локации *Active* в локацию *Inactive* — при выполнении синхронизации по каналу *sleep*. Если текущей локацией является *Active* и происходит синхронизация по каналу $exec[i]$, то текущей локацией остается *Active*. Если текущей локацией является *Inactive* и происходит синхронизация по каналу $exec[i]$, то это соответствует описанной выше некорректной последовательности синхронизаций и, следовательно, выполняется переход в локацию $ERROR$.

Для каждой из разработанных моделей планировщиков раздела была построена сеть автоматов, состоящая из данной модели (то есть параметризованного автомата) и описанного автомата-наблюдателя. Для всех моделей планировщика раздела для проверки сформулированного требования используется один и тот же автомат-наблюдатель, так как этому требованию должны удовлетворять все реализации базового типа TS. С помощью верификатора UPPAAL было доказано, что локация $ERROR$ во всех построенных сетях автоматов недостижима.

Ниже приведен ряд требований к реализациям базовых типов обобщенной модели и использующихся в дальнейших рассуждениях. Для всех разработанных параметризованных автоматов выполнение соответствующих требований было доказано согласно описанному выше методу.

Требования к реализациям базового типа TS («планировщик раздела»):

2. Для фиксированного состояния очереди готовых работ выбор работы для постановки на выполнение осуществляется однозначно.

3. Если в некоторый момент времени данный раздел активен и появляется одна новая готовая работа, то планировщик обрабатывает это событие мгновенно.

4. Если в некоторый момент времени данный раздел активен и появляется новая готовая работа, то выбор, должна ли она быть поставлена на выполнение, осуществляется однозначно.

5. События открытия и закрытия окон планировщик обрабатывает мгновенно.

6. Планировщик может поставить на выполнение только готовую работу.

7. Работа может быть поставлена на выполнение только вследствие следующих событий: открытие окна раздела, появление новой готовой работы, завершение некоторой работы. При этом постановка на выполнение происходит мгновенно.

8. Если открывается окно раздела и очередь готовых работ раздела не пуста, то на выполнение мгновенно ставится некоторая новая работа.

9. Если завершилась некоторая работа раздела, очередь готовых работ раздела не пуста и в данный момент не происходит закрытие окна раздела, то на выполнение мгновенно ставится некоторая новая работа.

10. Любая работа может быть вытеснена только в двух случаях: либо при закрытии окна данного раздела, либо при появлении новой готовой работы.

11. Если в некоторый момент времени на вычислителе выполняется работа раздела и появляется новая готовая работа, которая должна быть поставлена на выполнение, то текущая работа мгновенно вытесняется.

12. Если в некоторый момент времени на вычислителе выполняется работа раздела и происходит закрытие текущего окна этого раздела, то данная работа мгновенно вытесняется либо завершается.

Требования к реализациям базового типа T («функциональная задача»):

1. Любая работа может быть поставлена на выполнение, только если она является готовой и планировщик раздела об этом оповещен.

2. Очередная работа задачи становится готовой, как только становятся выполненными два условия: модельное время достигает левой границы директивного интервала; получены сообщения от всех соответствующих работ-отправителей (если задача зависит по данным от других задач).

3. Каждая работа задачи должна завершиться, либо когда время ее выполнения на вычислительном ядре достигнет WCET, либо при достижении правой границы своего директивного интервала. Работа не может завершиться, если ни одно из этих двух условий не выполнено.

4. Если задача имеет выходные данные, то каждая ее работа должна послать сообщения через все выходные виртуальные каналы строго в момент завершения работы при условии завершения не позднее правой границы своего директивного интервала.

5. Очередная работа задачи может быть поставлена на выполнение только в рамках своего директивного интервала.

Требования к реализациям базового типа CS («планировщик ядра»):

1. Для любого окна верно, что планировщик ядра оповещает планировщик раздела, соответствующего данному окну, об открытии и закрытии окна строго в моменты, определенные в расписании.

Требования к реализациям базового типа L («виртуальный канал»):

1. Задержка на передачу сообщения через виртуальный канал строго равна значению, указанному в конфигурации системы.

2. Для любого сообщения верно, что получатель оповещается о его получении ровно один раз, в момент окончания передачи.

Помимо приведенных выше требований, также было доказано выполнение порядка десяти требований к реализациям базовых типов автоматов.

Кроме требований, которым должны удовлетворять все реализации соответствующих базовых типов автоматов, существуют требования, специфичные для отдельных параметризованных автоматов (то есть моделей конкретных типов компонентов МВС РВ). В качестве примера одного из таких требований к параметризованным автоматам приведем следующее требование к модели планировщика с фиксированными приоритетами и вытеснением:

На выполнение всегда ставится работа с максимальным приоритетом.

Данное требование специфично для этой реализации планировщика раздела.

На основе требований к конкретным типам компонентов МВС РВ авторами был сформулирован ряд требований, специфичных для реализованных параметризованных автоматов. Их выполнение было доказано с использованием подхода автоматов-наблюдателей. При включении в модель МВС РВ пользовательских параметризованных автоматов к ним также могут быть сформулированы дополнительные требования, проверка выполнения которых осуществляется аналогично.

Отметим, что требования мгновенности реакции на некоторые события можно заменить на требования реакции на соответствующие события с фиксированной задержкой. Например, требование 3 к модели планировщика можно заменить на следующее: «Если в некоторый момент времени данный раздел активен и появляется одна новая готовая работа, то планировщик обрабатывает это событие за N тактов, где значение N задано в конфигурации». Однако в большинстве работ, например в [8–11], данной задержкой пренебрегают, так как работы одного раздела выполняются в общем адресном пространстве и величина задержки мала. Кроме того, модель предполагает, что между окнами разделов могут быть интервалы, не принадлежащие никаким разделам, поэтому такие интервалы могут использоваться для моделирования задержек, необходимых для переключения контекста разделов и инициализации окон.

2.2. Корректность модели в целом

Помимо требований корректности к моделям отдельных компонентов МВС РВ существуют и требования корректности к модели МВС РВ в целом. Проверить выполнение этих требований автоматизированно с помощью верификатора невозможно, так как в общем случае неизвестно количество автоматов в параметризованной сети автоматов, и задача верификации алгоритмически неразрешима [18]. Поэтому выполнение таких требований было доказано нами путем строгих логических рассуждений на основе информации о структуре модели, а также на основе выполнения требований к моделям компонентов МВС РВ.

Приведем пример доказательства выполнения следующего требования:

1. Для любых двух задач верно, что если одна задача зависит по данным от другой, то время начала выполнения каждой работы задачи-получателя данных не меньше времени завершения соответствующей работы задачи-отправителя данных плюс время передачи данных.

Доказательство выполнения данного требования основано на выполнении требований к реализациям базовых типов L и T. Из выполнения требования 2 к модели задачи следует, что время начала выполнения очередной работы не меньше времени получения сообщения от соответствующей работы-отправителя. Из выполнения требования 1 к модели виртуального канала следует, что время получения сообщения равно времени его отправки плюс значение задержки, указанное в конфигурации. Из выполнения требования 4 к модели задачи следует, что отправка сообщения происходит строго в момент завершения работы отправителя. Таким образом, из выполнения перечисленных требований к моделям компонентов следует выполнение указанного требования к модели в целом.

Приведем еще несколько требований к модели в целом:

2. Для любого ядра системы верно, что в каждый момент времени на нем выполняется не более одной работы.

Доказательство выполнения данного требования основано на выполнении требований к реализациям базовых типов CS и TS.

Доказательство выполнения следующих двух требований основано на структуре модели в целом, а именно на информации о приоритетах каналов синхронизации.

3. В момент закрытия окна работа не может быть поставлена на выполнение.

4. Если момент закрытия окна раздела совпадает с моментом завершения выполняющейся на вычислительном ядре работы, то сначала происходит завершение работы, а затем закрытие окна, то есть завершение работы не переносится на начало следующего окна данного раздела.

Нами было доказано выполнение еще семи требований к модели в целом, помимо перечисленных выше.

2.3. Детерминированность модели

Назовем модель МВС РВ с заданной конфигураций (то есть экземпляр сети автоматов) детерминированной, если этому экземпляру сети автоматов однозначно соответствует ее ВД. Докажем детерминированность модели от противного.

Предположим, что модель не является детерминированной, то есть для экземпляра сети автоматов, соответствующей некоторой конфигурации, могут быть получены две разные ВД. Упорядочим события этих двух ВД по времени. Так как события в системе могут происходить одновременно, то одному значению модельного времени соответствует некоторое множество событий. Пусть t_i — наименьшее значение модельного временем, которому в двух ВД соответствуют различные множества событий. Это означает, что как минимум одно событие синхронизации присутствует в множестве событий для этого значения времени одной ВД (будем считать эту ВД первой) и отсутствует в соответствующем множестве другой (будем считать эту ВД второй).

Рассмотрим все возможные варианты каналов данной синхронизации (для краткости далее под «событием X» будем понимать событие синхронизации по каналу X). Покажем, что для каждого канала предположение не верно, то есть если событие синхронизации по каналу присутствует в первой ВД, то оно присутствует и во второй ВД.

1. $wakeup_j$ или $sleep_j$. Из выполнения требования 5 к моделям планировщиков

раздела и требования 1 к моделям планировщиков ядра следует, что полученные ВД соответствуют конфигурациям, различающимся расписаниями окон для ядра j-го раздела. Это противоречит тому, что ВД являются результатом выполнения одной и той же модели.

2. $finished_j$. Возможны два варианта:

   - согласно первой ВД некоторая работа выполнялась на вычислительном ядре ровно WCET единиц времени; до момента t_i ВД совпадают, поэтому из отсутствия во второй ВД события $finished_j$ следует, что работа выполнялась на вычислительном ядре более WCET единиц времени, что противоречит требованию 3 к модели задачи;

   - согласно первой ВД для некоторой работы в момент t_i наступила правая граница директивного интервала; значение этой границы зависит только от номера работы и характеристик соответствующей задачи, поэтому и согласно второй ВД момент t_i является правой границей директивного интервала той же работы; однако, согласно второй ВД, в момент t_i эта работа не завершается, что противоречит требованию 3 к модели задачи.

3. $send_j$. Из выполнения требования 4 к модели задачи следует, что в первой ВД в момент t_i присутствует событие $finished_k$ и t_i не превышает правой границы директивного интервала завершившейся работы. Из рассуждений предыдущего пункта следует, что событие $finished_k$ присутствует и во второй ВД с такой же временной меткой. Следовательно, согласно второй ВД, нарушено требование 4 к модели задачи, поэтому данный вариант не возможен.

4. $receive_j$. Из наличия данного события в первой ВД и выполнения требований 1 и 2 к модели j-го ВК следует, что в первой ВД присутствует событие $send_j$ с временной меткой $t_i - d_j$, где d_j — задержка на передачу сообщения через j-й ВК. Возможны два варианта:

   - $d_j > 0$; следовательно, событие $send_j$ присутствует и во второй ВД, так как до момента t_j ВД совпадают; отсутствие события $receive_j$ в момент t_i в таком случае говорит о невыполнении требований 1 либо 2 к модели виртуального ВК, поэтому данный вариант невозможен;

   - $d_j = 0$; если событие $send_j$ присутствует и во второй ВД в момент t_i, то это свидетельствует о невыполнении требований 1 либо 2 к модели ВК (аналогично варианту $d_j > 0$), что невозможно; следовательно, событие $send_j$ не присутствует во второй ВД, но это противоречит рассуждениям предыдущего пункта и поэтому невозможно.

5. $ready_j$. Событие соответствует тому, что некоторая работа стала готовой к выполнению согласно первой ВД. Из выполнения требования 2 к модели задачи следует, что t_i не меньше левой границы директивного интервала работы и в первой ВД присутствуют события $receive_{k_1}$, ..., $receive_{k_n}$ с временными метками, не большими t_i. Из совпадения ВД до момента t_i и рассуждений

предыдущего пункта следует, что все эти события $receive_{k_1}$, ..., $receive_{k_n}$ присутствуют также и во второй ВД. Таким образом, отсутствие во второй ВД в момент t_i события $ready_j$ свидетельствует о нарушении требования 2 к модели задачи и поэтому невозможно.

6. $exec_j$. Событие соответствует тому, что некоторая работа поставлена на выполнение. Из выполнения требования 3 к модели в целом следует, что t_i не является моментом закрытия окна раздела. Из выполнения требования 6 к модели планировщика раздела и требования 5 к модели задачи следует, что данная работа готова и t_i принадлежит ее директивному интервалу. Из совпадения ВД до момента t_i и из рассуждений предыдущего пункта следует, что согласно обеим ВД на момент t_i есть как минимум одна готовая работа. Из выполнения требования 7 к модели планировщика раздела следует, что имеет место один из трех вариантов:

   - в первой ВД в момент t_i присутствует событие $wakeup_k$. Из рассуждений пункта 1 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 2 и 8 к модели планировщика следует, что в момент t_i во второй ВД присутствует событие $exec_j$, что противоречит введенному предположению;

   - в первой ВД в момент t_i присутствует событие $finished_k$. Из рассуждений пункта 2 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 2 и 9 к модели планировщика следует, что в момент t_i во второй ВД присутствует событие $exec_j$, что противоречит введенному предположению;

   - в первой ВД в момент t_i присутствует событие $ready_k$. Из рассуждений пункта 5 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 2 и 4 к модели планировщика следует, что в момент t_i во второй ВД присутствует событие $exec_j$, что противоречит введенному предположению;

7. $preempt_j$. Событие соответствует вытеснению с вычислительного ядра некоторой работы. Согласно требованиям 10, 11 и 12 к модели планировщика раздела имеет место один из двух вариантов:

   - В первой ВД в момент t_i присутствует событие $sleep_k$. Согласно рассуждениям пункта 1, во второй ВД также присутствует событие $sleep_k$ с той же временной меткой. Из выполнения требования 3 к модели в целом следует, что событие $exec_j$, предшествующее $preempt_j$, имеет временную метку, строго меньшую t_i. Поэтому событие $exec_j$ присутствует и во второй ВД. Следовательно, согласно требованию 12 к модели планировщика раздела, в момент t_i во второй ВД имеется событие $preempt_j$, что приводит к противоречию.

   - В первой ВД в момент t_i присутствует событие $ready_k$. Согласно рассуждениям пункта 5, это же событие присутствует и во второй ВД. Из

выполнения требований 4 и 7 к модели планировщика раздела следует, что во второй ВД в момент t_i присутствует событие $exec_h$. Это же событие присутствует и в первой ВД (см. предыдущий пункт). Из совпадения ВД до момента t_i следует, что согласно обеим ВД на данном вычислительном ядре выполняется одна и та же работа. Поэтому из выполнения требования 11 к модели планировщика следует, что во второй ВД, как и в первой, должно присутствовать событие $preempt_j$, что противоречит предположению.

Таким образом, доказано от противного, что предположение о существовании двух различных ВД, соответствующих одному экземпляру рассматриваемой сети автоматов, ложно. Следовательно, представленная в виде этой сети модель является детерминированной.

Заключение

В работе предложен подход к обоснованию корректности моделей МВС РВ, синтезируемых на основе обобщенной модели, введенной первым из соавторов в [2]. Выбранный для моделирования математический аппарат сетей временных автоматов с остановкой таймеров позволяет проверить с помощью верификатора выполнение ряда требований корректности к параметризованным автоматам, моделирующим отдельные компоненты системы. Для разработанных параметризованных автоматов такая проверка была выполнена. Посредством строгих логических рассуждений, основанных на выполнении требований к моделям компонентов МВС РВ, а также на информации о структуре модели системы в целом, доказано выполнение ряда требований корректности к моделям МВС РВ, построенных согласно предложенному методу, а также их детерминированность. С помощью данного подхода доказана корректность всех моделей, синтезируемых с использованием разработанных параметризованных автоматов, моделирующих компоненты МВС РВ. Также показано, что для сохранения корректности и детерминированности параметризованной модели МВС РВ при включении в неё пользовательской модели компонента системы достаточно проверить выполнение требований корректности к добавляемой модели компонента. Эта проверка производится автоматически с использованием верификатора.

Предложенные методы и средства реализованы программно, интегрированы с САПР планирования вычислений в МВС РВ [8] и успешно апробированы на данных, близких к реальным. Результаты экспериментов приведены в [2].

Отметим, что предложенный подход к построению моделей и доказательству их корректности может быть использован для моделирования вычислительных систем других типов (например, с федеративной архитектурой), что является одним из возможных направлений дальнейших исследований.

Список литературы / References

[1] Watkins C. B., Walter R., "Transitioning from Federated Avionics Architectures to Integrated Modular Avionics", *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*, 2007, 2.A.1-1–2.A.1-10.

[2] Glonina A. B., Bahmurov A. G., "Stopwatch Automata-Based Model for Efficient Schedulability Analysis of Modular Computer Systems", *Parallel Computing Technologies*, LNCS, **10421**, Springer, 2017, 289–300.

[3] Cassez F., Larsen K, "The Impressive Power of Stopwatches", *CONCUR 2000 — Concurrency Theory*, LNCS, **1877**, Springer, 2000, 138–152.

[4] *Avionics Application Software Standard Interface. Arinc Specification 653*, Aeronautical Radio, 1997.

[5] Marouf M., Sorel Y., "Scheduling Non-Preemptive Hard Real-Time Tasks with Strict Periods", *Proceedings of the 16th IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011, 1–8.

[6] Mallachiev K. M., Pakulin N. V., Khoroshilov A. V., "Design and Architecture of Real-Time Operating System", *Proceedings of ISP RAS*, **28**:2 (2016), 181–192.

[7] *RTCA: DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations*, Radio Technical Commission for Aeronautics, 2005.

[8] Balashov V. V., Balakhanov V. A., Kostenko V. A., "Scheduling of Computational Tasks in Switched Network-Based IMA Systems", *Proceedings of International Conference on Engineering and Applied Sciences Optimization*, 2014, 1001–1014.

[9] Третьяков А., "Автоматизация построения расписаний для периодических систем реального времени", *Труды Института системного программирования РАН*, **22** (2012), 375–400; [Tretyakov A., "Automation of Scheduling for Periodic Real-Time Systems", *Proceedings of ISP RAS*, **22** (2012), 375–400, (in Russian).]

[10] Wang D., Han J., Ma D., Zhao X., "Studying on ARINC653 Partition Run-time Scheduling and Simulation", *Proceedings of World Academy of Science, Engineering and Technology*, 2012, 1583–1587.

[11] Lee Y. H., Kim D., Younis M., Zhou J., "Scheduling Tool and Algorithm for Integrated Modular Avionics Systems", *Proceedings of the 19th IEEE Digital Avionics Systems Conference*, 2000, 1C2/1–1C2/8.

[12] *Aircraft Data Network Part 7. Avionics Full Duplex Switched Ethernet (AFDX) Network*, Aeronautical Radio, 2005.

[13] "UPPAAL Home", http://www.uppaal.org/.

[14] Andre E., "Observer Patterns for Real-Time Systems", *Proceedings of the 18th International Conference on Engineering of Complex Computer Systems*, 2013, 125–134.

[15] Abdeddaim Y., Maler O., "Preemptive Job-Shop Scheduling using Stopwatch Automata", *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, **2280**, Springer, 2002, 113–126.

[16] Krcal P., Yi W., "Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata", *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, **2988**, Springer, 2004, 236–250.

[17] David A., Illum J., Larsen K., Skou A., "Model-based Framework for Schedulability Analysis Using Uppaal 4.1", *Model-Based Design for Embedded Systems*, 2009, 93–119.

[18] Apt K. R., Kozen D. C., "Limits for Automatic Verification of Finite-State Concurrent Systems", *Information Processing Letters*, **22**:6 (1986), 307–309.

Glonina A. B., Balashov V. V., "On the Correctness of Real-Time Modular Computer Systems Modeling with Stopwatch Automata Networks", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 174–192.

DOI: 10.18255/1818-1015-2018-2-174-192

Abstract. In this paper, we consider a schedulability analysis problem for real-time modular computer systems (RT MCS). A system configuration is called schedulable if all the jobs finish within their deadlines. The authors propose a stopwatch automata-based general model of RT MCS operation. A model instance for a given RT MCS configuration is a network of stopwatch automata (NSA) and it can be built automatically using the general model. A system operation trace, which is necessary for checking the schedulability criterion, can be obtained from the corresponding NSA trace. The paper substantiates the correctness of the proposed approach. A set of correctness requirements to models of system components and to the whole system model were derived from RT MCS specifications. The authors proved that if all models of system components satisfy the corresponding requirements, the whole system model built according to the proposed approach satisfies its correctness requirements and is deterministic (i.e. for a given configuration a trace generated by the corresponding model run is uniquely determined). The model determinism implies that any model run can be used for schedulability analysis. This fact is crucial for the approach efficiency, as the number of possible model runs grows exponentially with the number of jobs in a system. Correctness requirements to models of system components models can be checked automatically by a verifier using observer automata approach. The authors proved by using UPPAAL verifier that all the developed models of system components satisfy the corresponding requirements. User-defined models of system components can be also used for system modeling if they satisfy the requirements.

Keywords: modeling, model checking, integrated modular avionics, scheduling

On the authors:
Alevtina B. Glonina, orcid.org/0000-0001-8716-4128, programmer,
Lomonosov Moscow State University,
1 Leninskie Gory, Moscow 119991, Russia, e-mail: alevtina@lvk.cs.msu.su

Vasily V. Balashov, orcid.org/0000-0001-5211-805X, Ph.D. in Mathematics, senior research fellow
Lomonosov Moscow State University,
1 Leninskie Gory, Moscow 119991, Russia, e-mail: hbd@cs.msu.su

Acknowledgments:
This work was supported by RFBR (grant No 17-07-01566).

Моделирование и анализ информационных систем. Т. 25, № 2 (2018), с. 193–206
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 193–206

©Шмелёва Т. Р., 2017

DOI: 10.18255/1818-1015-2018-2-193-206

УДК 004.94, 004.724.4

Сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла к индуцированным тупикам

Шмелёва Т. Р.

получена 3 сентября 2017

Аннотация. Рассматриваются классификация и области применения методов коммутации, их достоинства и недостатки. Построена модель вычислительной решетки в форме раскрашенной сети Петри с узлом, реализующим сквозную коммутацию пакетов. Модель состоит из узлов коммутации пакетов, генераторов трафика и пушек, которые формируют злонамеренный трафик, замаскированный под обычный пользовательский трафик. Исследованы характеристики модели решетки в условиях рабочей нагрузки с различной интенсивностью. Оценено влияние злонамеренного трафика типа «дуэль трафика» на параметры качества обслуживания решетки. Проведен сравнительный анализ устойчивости вычислительных решеток с узлами, реализующими технологию передачи пакетов с обязательной буферизацией, и сквозной коммутации. Показано, что производительности решеток примерно одинаковы в условиях рабочей нагрузки; а в условиях пиковой нагрузки решетка с узлом, реализующим технологию передачи пакетов с принудительной буферизацией, более устойчива. Решетка с узлами, реализующими технологию SAF, приходит к полному тупику через дополнительную нагрузку менее чем 10 процентов. После детального исследования показано, что конфигурация «дуэль трафика» не оказывает влияния на решетку с узлами cut-through при увеличении рабочей нагрузки до пиковой, при которой решетка приходит к полному тупику. Периодичность запуска пушек, генерирующих злонамеренный трафик, определена случайной функцией с пуассоновским распределением. Для построения моделей и измерений характеристик используется моделирующая система CPN Tools. Производительность решетки и среднее время доставки пакета оценивается при различных вариантах нагрузки на решетку.

Ключевые слова: безопасность вычисления на решетках, сквозная коммутация, защита против атак трафика, оценка производительности, раскрашенная сеть Петри, тупик

Для цитирования: Шмелёва Т. Р., "Сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла к индуцированным тупикам", *Моделирование и анализ информационных систем*, **25**:2 (2018), 193–206.

Об авторах:
Шмелёва Татьяна Рудольфовна, orcid.org/0000-0002-4799-3842, канд. техн. наук, доцент,
Одесская национальная академия связи им. А.С. Попова,
ул. Кузнечная, 1, г. Одесса, 65029 Украина, e-mail: tishtri@rambler.ru

Благодарности: Автор благодарит программный комитет и участников PSSV-2017 за обсуждение результатов, представленных в статье.

Введение

Сети вычислительных ресурсов [1] или вычислительные решетки (Grid Computing) решают проблемы, связанные с интенсивными вычислениями, обработкой супербольших массивов данных, что требует использования разнородных и сверхскоростных ресурсов. Не последнее место в этой системе занимают устройства передачи данных [2]: коммутаторы и маршрутизаторы. Для повышения производительности решетки и стабильности функционирования становится актуальным выбор метода коммутации пакетов [2, 3] в устройствах передачи данных.

Ранее были изучены параметры качества обслуживания и эффективность телекоммуникационных сетей [4–6] и вычислительных решеток с узлами, реализующими метод обязательной буферизации [7]. Модели были построены в форме раскрашенных сетей Петри [8]. Проведены исследования влияния злонамеренного трафика на поведение и характеристики вычислительной решетки [9], с коммуникационными устройствами типа store-and-forward. Оценка производительности решетки и среднего времени доставки пакета проведена в [10], для решеток с коммутационным устройством, реализующим сквозную коммутацию пакетов.

Целью настоящей статьи является сравнительный анализ устойчивости вычислительных решеток [7, 10] с различной архитектурой узла к индуцированным тупикам; дальнейшее развитие методов анализа прямоугольных коммуникационных решеток для узлов, реализующих сквозную коммутацию пакетов. Методы предназначены для применения при проектировании вычислительных решеток, в разработке новых телекоммуникационных устройств и в интеллектуальных системах защиты.

1. Классификация и области применения методов коммутации

В современных телекоммуникационных системах доминируют два основных метода [2, 3] пакетной коммутации: первый с обязательной буферизацией пакета, или store-and-forward (SAF); второй без буферизации, или cut-through, другое популярное название метода сквозной коммутации – «на лету» (on the fly). Применяются в сетях и гибридные коммутаторы, которые могут автоматически менять режим работы с cut-through на SAF и наоборот. Переключение между режимами основано на определении производительности узла и целостности пакета.

Технология коммутации с обязательной буферизацией является традиционной для большинства современных сетей [2, 5, 6]. Она обеспечивает передачу пакета отправителю только после приема пакета и проверки контрольной суммы (FCS). Пакет удаляется, если он короче 64 байтов или длиннее 1518 байтов, или контрольная сумма ошибочна. Для метода SAF время доставки пакета увеличивается пропорционально размеру пакета: чем больше размер пакета, тем больше времени требуется на прием и проверку. Однако коммутаторы, реализующие метод обязательной буферизации, обладают существенными преимуществами: устройство может быть оснащено портами, поддерживающими разные технологии и скорости передачи, поэтому задержка, вносимая коммутацией store-and-forward при передаче кадров, оказывает-

ся незначительной; и второе, проверка целостности пакета позволяет не нагружать сеть поврежденными пакетами.

Технология сквозной коммутации буферизует только заголовок сообщения [2]. Коммутаторы cut-through не выполняют селекцию пакетов [3], благодаря этому они самые быстрые в своем классе. Недостаток этого метода коммутации заключается в том, что он передает любые пакеты, в том числе содержащие неправильную контрольную сумму, однако современная сетевая инфраструктура позволяет свести вероятность возникновения ошибочных пакетов к минимуму. В некоторых устройствах со сквозной коммутацией используется метод ICS (interim cut-through switching), который фильтрует пакеты длиной менее 64 байтов. Коммутаторы без обязательной буферизации [3] в основном используются в центрах обработки данных, где необходимо обеспечить непрерывную передачу большого объема трафика с минимальными задержками.

2. Модель вычислительной решетки со сквозной коммутацией пакетов

В телекоммуникационных сетях одной из основных компонент является активное оборудование, такое как коммутаторы или маршрутизаторы. Модели коммуникационных прямоугольных решеток [1] с основным элементом, представленным моделью коммуникационного устройства, реализующий технологию SAF, изучены в [7, 9]. Рассмотрим модель узла со сквозной коммутацией пакетов [10], или, иначе, с прямой передачей пакета из порта в порт без принудительной буферизации. Используемые типы, функции, переменные и константы описаны в [9]. Для построения модели решетки используются две основные модели: модель узла со сквозной коммутацией пакетов как коммуникационное устройство и модель генератора трафика как терминальное устройство. Для исследования влияния злонамеренного трафика на функционирование решетки построены модели пушек пакетов. Все модели построены в моделирующей системе CPN Tools [8].

2.1. Модель узла со сквозной коммутацией пакетов

Модель узла основана на стандартных процедурах передачи пакетов [3] в современных сетях и решетках. На рисунке 1 представлена базовая модель узла сетевого устройства со сквозной коммутацией пакетов [10], которая будет использована для построения модели прямоугольной решетки. Модель узла содержит четыре порта, обеспечивающие полнодуплексный режим работы для одновременной передачи и приема пакетов. Каждый порт состоит из четырех позиций, которые моделируют входной и выходной каналы: буфер выходного канала порта описывается позицией *po* и позицией-ограничителем *pol*; буфер входного канала порта представлен позицией *pi* и ее ограничителем – позицией *pil*. Позиции портов описаны как тип *pkt*, ограничитель порта равен единице, которая соответствует пропускной способности канала порта – одному пакету. Для спецификации всех портов к имени порта добавляется индекс порта, равный от одного до четырех. Для построения модели решетки узлы портов узлов располагаются по сторонам квадрата в следующем по-

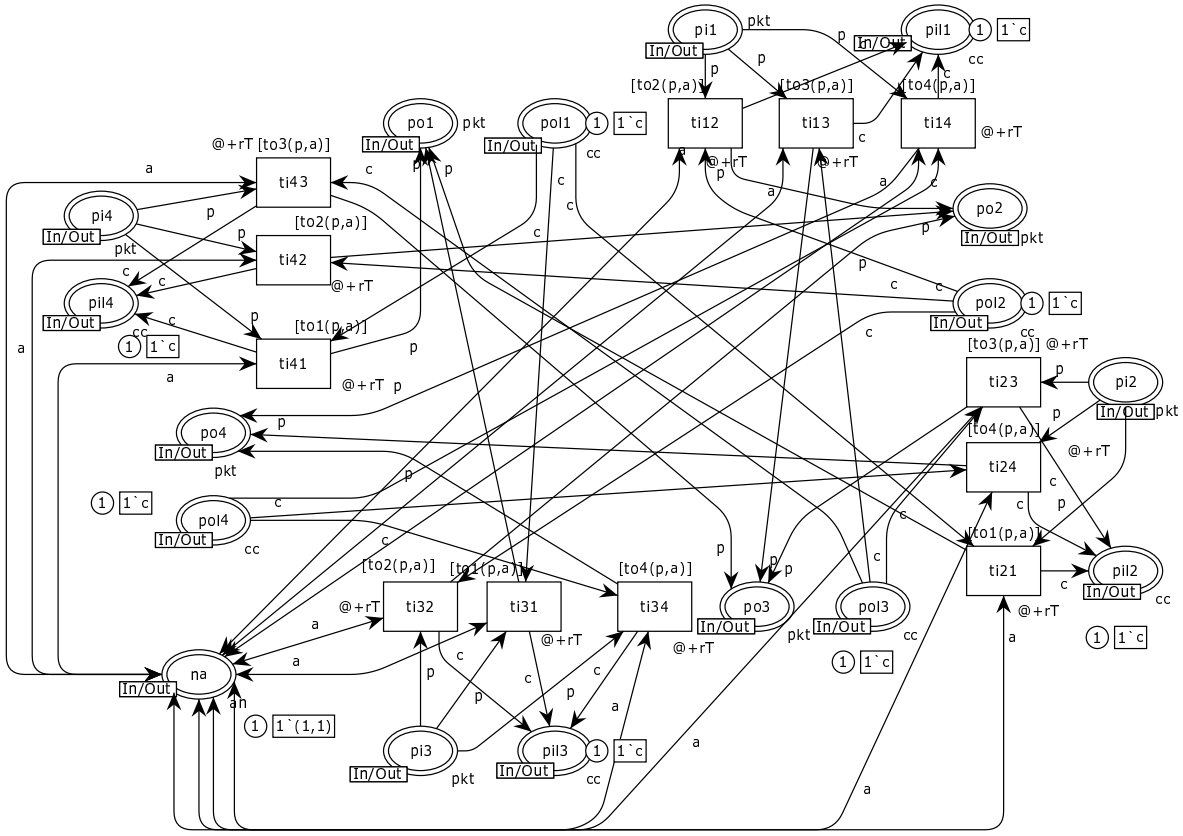

Рис. 1. Модель коммуникационного узла

Fig. 1. Model of a communication node

рядке: верхний порт – это первый порт, описывается позициями $po1, pol1, pi1, pil1$; правый порт – это второй порт с позициями $po2, pol2, pi2, pil2$; нижний порт является третьим портом с позициями $pi3, pil3, po3, pol3$; и левый порт – это четвертый порт с позициями $pi4, pil4, po4, pol4$. Позиции портов и их ограничителей являются контактными позициями для композиции решетки любого размера.

Для адресации узлов решетки используются индексы (i, j), где первый индекс описывает номер строки, а второй – номер столбца. Каждый узел имеет уникальный адрес, который хранится в контактной позиции na. Выходной канал порта моделируется только двумя позициями: po, pol, а входной канал порта моделируется двумя позициями pi, pil и тремя переходами. Для описания перенаправления пакета с входного порта на выходной порт к имени перехода ti добавляются два индекса для каждого возможного направления передачи (верхний, нижний, левый или правый). Например, переход $ti43$ передает пакет из входного порта $pi4$ в выходной порт $po3$.

Временные характеристики модели узла, представленной в [9], заданы двумя параметрами sT и rT, которые представляют собой временную задержку отправки и получения пакета соответственно. В рассматриваемой модели для описания сквозной коммутации пакетов каждый переход наделен временным атрибутом rT, который описывает время получения и перенаправления пакета из входного порта в выходной порт. Напомним, что в соответствии с алгоритмом сквозной коммутации [3] пакет перенаправляется из входного порта в выходной порт, если выходной

порт свободен. Для определения выходного порта назначения [7] в модели узла используются специальные предикаты, они представлены как атрибуты переходов. Например, предикат $to4(p, a)$ определяет передачу пакета в четвертый выходной порт, где переменная p содержит информацию о пакете (адрес назначения, адрес отправителя) и a адрес текущего узла.

В начальной маркировке все позиции ограничителей портов $pil*$ и $pol*$ содержат маркировку 1'c; все входные $pi*$ и выходные $po*$ позиции пустые, т.е. не содержат фишек. Модели узла решетки присваивается имя в соответствии с номером строки и столбца решетки, например, устройство $n6-4$ является четвертым элементом в шестой строке прямоугольной решетки.

2.2. Модель генератора трафика

Для формирования нагрузки и исследования параметров качества обслуживания решетки построена модель генератора трафика (терминального устройства), которая подробно описана в [9]. Терминальным устройствам присваиваются имена в соответствии с первой буквой имени границы, к которой они присоединены, к букве добавляется пара индексов – номера строки и столбца. Например, в имени «правого» терминального устройства первая буква r, терминальное устройство $r5-9$ пятое в девятом столбце сетки. Модель состоит из трех основных частей: генерация и отправка пакета [4]; получение пакета; обработка сообщения [6] и вычисление параметров качества обслуживания решетки [5, 10].

Первая часть описывает процесс генерации трафика, интенсивность и функцию распределения трафика, правила отправки пакетов. Каждый пакет состоит из адреса получателя, адреса отправителя, информационного поля и временного штампа, равного времени отправки сообщения. Адрес получателя выбирается из всего множества адресов с помощью случайной функции. Счетчик отправленных пакетов в начальной маркировке равен нулю и при формировании сообщения увеличивается на единицу. Обязательным условием генерации пакета является доступность выходного канала порта. Получение пакета моделируется двумя позициями входного канала порта принимающего узла, далее все пакеты используются для нахождения параметров QoS решетки в процессе вычислительного эксперимента. Третья часть модели содержит счетчик полученных пакетов, накапливает сумму времен доставки пакетов, рассчитывает среднее время доставки пакета, производительность решетки.

2.3. Модель пушки пакетов

Для исследования поведения решетки и оценки параметров качества обслуживания в условиях влияния злонамеренного трафика построены и присоединены к границам решетки модели пушек пакетов [7]. Модель пакетной пушки, которая является упрощенной моделью генератора трафика, представлена на рисунке 2. Формирование пакетов, передача их в сеть и увеличение счетчика отправленных пакетов выполняется переходом $GunGenP$. Периодичность запуска определена случайной функцией $gDelay()$ и переменной tic, которая хранится в позиции $GunClock$, количество сгенерированных пушкой пакетов вычисляется в позиции $N - sndg$. Адрес

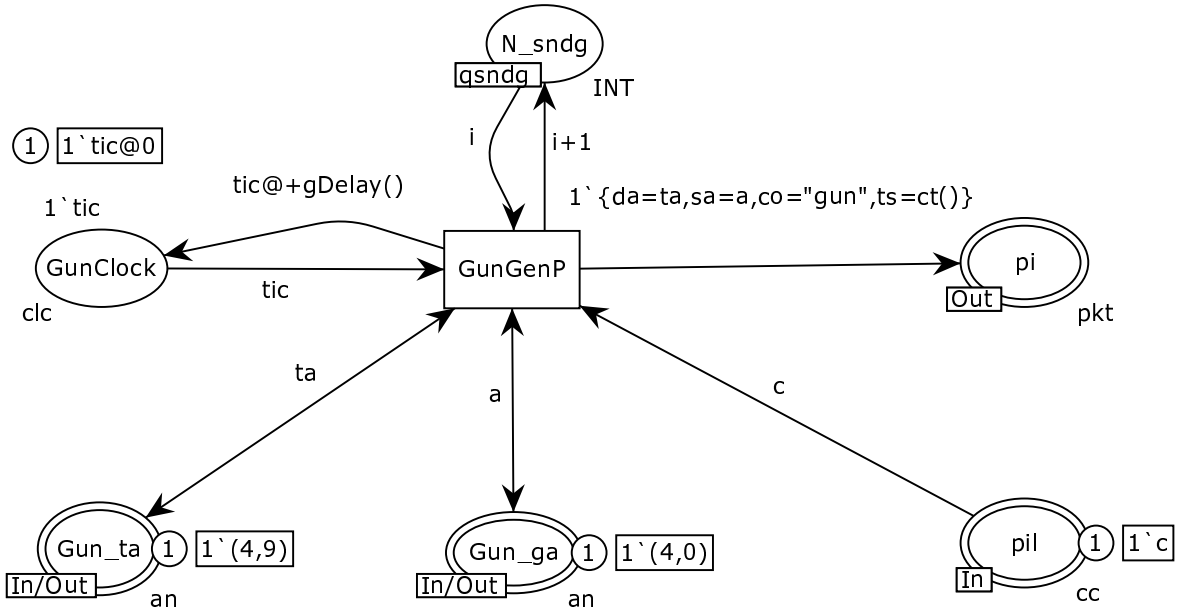

Рис. 2. Модель пушки пакетов

Fig. 2. Model of a packet gun

отправителя задан в позиции $Gun-ga$; адрес получателя назначается в зависимости от типа злонамеренного трафика в позиции $Gun-ta$; позиции pi и pil моделируют канал передачи пакета. Получение и обработку пакетов пушек выполняют терминальные устройства, для идентификации пакета пушки информационное поле содержит слово gun. Основные характеристики пушек, которые влияют на поведение модели решетки, такие как: интенсивность работы, количество пушек и мишеней, их расположение – изучены в [9].

2.4. Модель вычислительной решетки с узлом, реализующим сквозную коммутацию пакетов

Модель вычислительной решетки [1] представляет собой объединение моделей узлов, реализующих сквозную коммутацию пакетов, и моделей генераторов трафика [7]. В соответствии с иерархической структурой моделей, построенных в CPN Tools [8], модели всех устройств представлены как подмодели в виде переходов на главной странице модели. К переходам добавлены позиции, моделирующие порты устройств, и позиция, которая содержит уникальный адрес устройства. Модель вычислительной решетки размера 8x8 с узлом, реализующим сквозную коммутацию пакетов, и с текущей маркировкой, полученной в процессе моделирования, представлена на рисунке 3. К модели решетки добавлены две пушки, образующие конфигурацию «дуэль трафика», это пара пушек с взаимными мишенями, которые присоединены к терминальным устройствам с индексами (4,0) слева и (4,9) справа.

Композиция модели решетки производится согласно правилам, представленным в [7, 9]. Модель решетки является главной страницей модели, имеет вид прямоугольной матрицы, состоящей из подмоделей коммуникационных узлов, реализу-

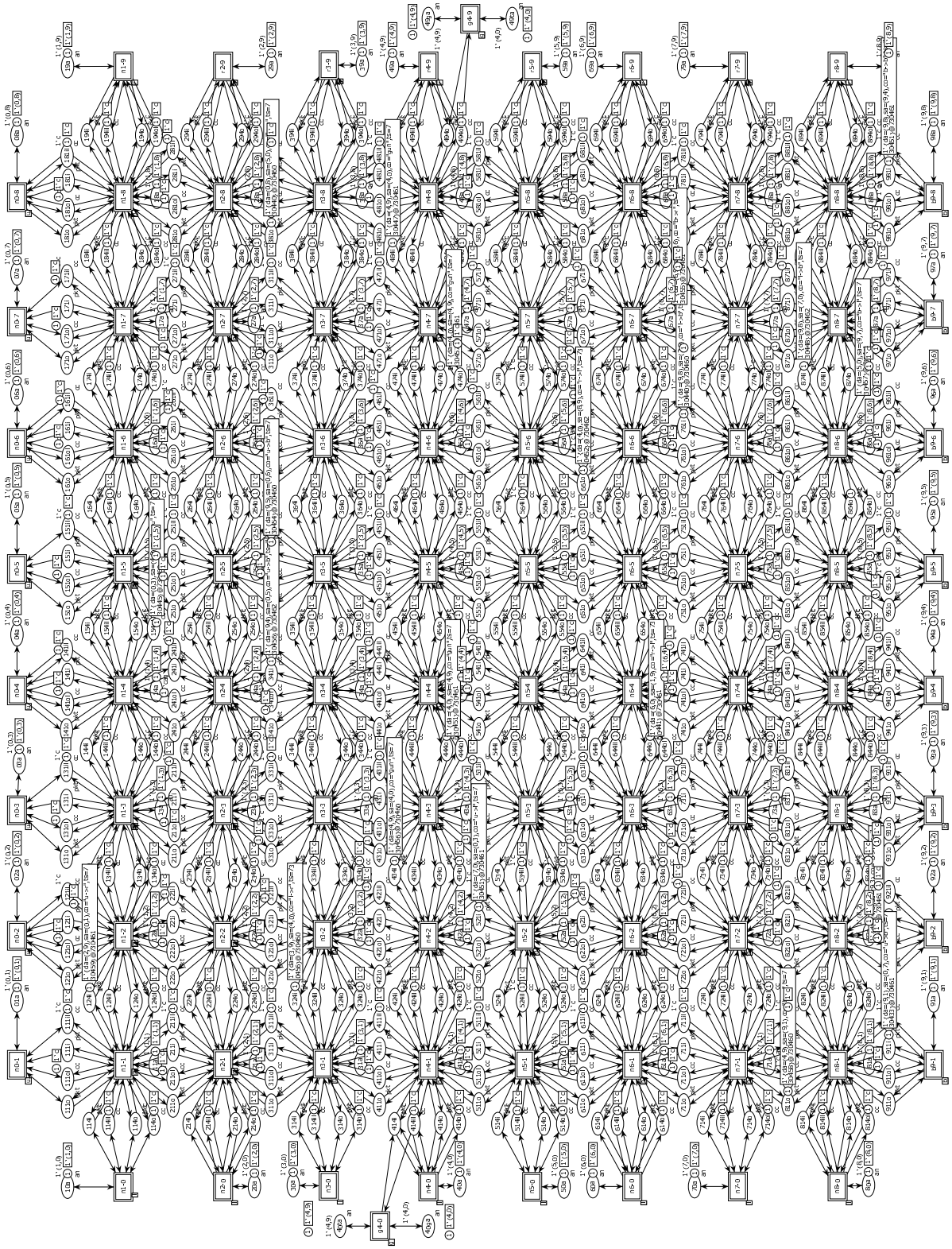

Рис. 3. Модель вычислительной решетки 8x8

Fig. 3. Model of an 8x8 computing grid

ющих метод сквозной коммутации. Модели генераторов трафика и модели пушек присоединены к границам модели решетки. Коммуникационные и терминальные устройства представлены переходами сети Петри, позиции с индексами (* – i, j, p: i – номер строки, j – номер столбца, p – номер порта) и суффиксами $*o, *ol, *i, *il$ являются контактными позициями, которые описывают выходные и входные порты узлов сетки и терминальных устройств.

Первый и четвертый порты каждого узла представлены индексом этого узла; второй и третий порты объединяются с первым и четвертым портами следующих узлов и имеют индексы следующих узлов. Например, позиция $361i$ является первым входным портом узла $n3-6$ и третьим выходным портом узла $n2-6$. Контактные позиции последней строки и правого столбца решетки описывают первый и четвертый порты узлов, которых нет в модели, эти позиции используются для присоединения нижних и правых терминальных устройств, генерирующих пользовательский трафик. Например, позиция $494o$ является четвертым выходным портом узла $n4-9$, но узла с таким индексом (4,9) нет в решетке, эта позиция объединяется с входным портом терминального устройства $r4-9$. Далее все исследования будут проводиться на модели вычислительной решетки размера 8x8.

3. Сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла

3.1. Масштабирование времени

После построения и отладки модели имитационное моделирование предусматривает специальную организацию экспериментов с моделью. Наличие стационарного режима функционирования модели позволяет провести оценку средних характеристик и других статистических моментов. Моделирующая система CPN Tools [8] используется как обычная система имитационного моделирования, которая позволяет моделировать на больших интервалах времени поведение сети любой сложности [4, 9]. Больший интерес представляют модели, которые используют случайные функции [4–6], в этом случае исследуются такие особенности сети, как среднее время доставки пакета, производительность, количество отправленных и принятых пакетов т.д.

В CPN Tools нет инструментов для управления временем, но предоставляется возможность выбрать достаточно большое количество шагов, чтобы обеспечить длительность процесса моделирования, соответствующую реальному времени. Также не вычисляется первичная статистическая информация: максимальное, минимальное и среднее количество фишек в позициях, частоты срабатывания переходов и т.д. Но система предоставляет язык для описания процессов накопления и вычисления характеристик, который используется в измерительных фрагментах сети [4–6].

Эксперименты с моделью предполагают: масштабирование времени, существование стационарного режима поведения модели, оценку характеристик. Масштабирование времени представляет большой интерес, чтобы сделать модель реалистичной. Время в CPN Tools измеряется в единицах модельного времени (MTU), которые не имеют размерности, и представляется натуральным числом. Поэтому для исследуе-

мой сети сначала определяются времена в реальных единицах (мс, нс) из описания технологии, аппаратных средств и программного обеспечения. Затем выбирается MTU как наименьший интервал времени. Затем все времена модели пересчитываются в MTU, после получения результатов моделирования время пересчитывается обратно в реальные единицы времени.

Для определения временных характеристик модели вычислительной решетки и расчета параметров качества обслуживания рассмотрим масштабирование времени для технологии 10 Гбит/с. Временные характеристики узла решетки заданы двумя параметрами sT и rT, которые представляют собой временную задержку отправки и получения пакета соответственно. Скорость передачи полезной информации в рассматриваемой технологии составляет 8 Гбит/с, или, иначе 1 Гбайт/с. Примем длину пакета равной максимальной длине кадра Ethernet, 1518 байтов, и длину заголовка кадра 18 байтов. Тогда весь пакет передается за $1,518 \cdot 10^{-6}$ секунды, пакет без адресной информации за $1,5 \cdot 10^{-6}$, заголовок пакета за $0,018 \cdot 10^{-6}$. Значение $0,018 \cdot 10^{-6}$, как минимальное, примем за 1 MTU. Пересчитаем остальные значения: $1,5 \cdot 10^{-6}$ равно 83 MTU, $1,518 \cdot 10^{-6}$ равно 84 MTU.

Тогда для метода коммутации с обязательной буферизацией временная задержка получения пакета rT равна 84 MTU, временная задержка отправки пакета sT равна 1 MTU, общее время задержки 85 MTU. Для метода сквозной коммутации временные задержки получения пакета и отправки пакета являются объединенной величиной (rT), в соответствии с характеристикой технологии [2,3].

Для решетки 8x8 идеальное время доставки пакета при методе сквозной коммутации (передача через 8 устройств-хопов) равно 84 MTU. Идеальное время предполагает, что заголовок пакета находится в 8 узле, а хвост пакета еще передается в первом узле. Тогда временная задержка на одном узле примерно равна 11 MTU. Для того чтобы упростить оценку средних и сделать модель более реалистичной, примем следующие значения временных характеристик: для метода коммутации с обязательной буферизацией общая временная задержка 100 MTU, для метода сквозной коммутации 20 MTU. Разделим значения на 10, для узлов SAF временные характеристики на переходах модели узла назначим равными $sT = rT = 5$, для узлов cut-through $rT = 2$.

3.2. Исследование характеристик моделей решеток в условиях рабочей нагрузки

Модель вычислительной решетки отлажена в условиях рабочей нагрузки, которую формируют модели генераторов трафика [10]. Выполнена оценка параметров качества обслуживания и поведения решетки в условиях рабочей и пиковой нагрузки. Исследования производительности решетки и среднего времени доставки пакета выполнены для распределения Пуассона с различной интенсивностью.

В таблице 1 представлены результаты исследования характеристик решетки в условиях рабочей нагрузки для узлов без обязательной буферизации. Аббревиатура MTU (Model Time Unit) обозначает единицу измерения модельного времени; символ $*$ показывает, что решетка приходит к полному тупику, то есть в модели отсутствуют активные переходы. Производительность решетки измеряется в количестве

Таблица 1. Характеристики модели решетки 8x8 при рабочей нагрузке

Table 1. Characteristics of an 8x8 grid model under a workload

Интенсивн. рабочей нагрузки Workload intensity	Шаг Step	Время Time	Кол-во отправлн. пакетов N sent packets	Кол-во принятых пакетов N received packets	Производит. решетки Grid performance	Ср.время доставки пакета Average packetDT
90.0	10000000	2615179	930023	901051	0.34	15
50.0	10000000	1453296	930061	900918	0.61	15
40.0	10000000	1163360	930316	901580	0.77	15
30.0	10000000	872079	930017	900635	1.03	15
20.0	10000000	581430	930397	901152	1.54	15
15.0*	9080431	396103	844938	818415	2.06	15
10.0*	8339863	242597	775968	751454	3.09	16
9.0*	2182011	57315	203181	196658	3.44	16
8.0*	2164592	50464	201381	194996	3.86	16
7.0*	7488036	152909	696902	674855	4.41	16
6.0*	1366960	26111	127387	123307	5.11	17
5.0*	381924	5950	35588	34339	5.81	19
4.0*	49317	842	4729	4397	5.62	24

пакетов в единицу модельного времени ($packets/MTU$), среднее время доставки пакета в MTU.

Полученные результаты (таблица 1) показывают, что решетка приходит в тупик даже при рабочей нагрузке, в большинстве случаев тупик означает, что порты назначения узлов заняты. Максимальная производительность решетки, или пиковая нагрузка, достигается при интенсивности рабочей нагрузки wl=15.0 и заданной интенсивности обслуживания rT=2, и равна gp=2 пакета за единицу модельного времени.

Проведен сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла. В таблице 2 представлены результаты исследования решеток посредством рабочей нагрузки для узлов, реализующих технологию передачи пакетов с принудительной буферизацией SAF [9], и сквозной коммутации. Размер буфера в модели узла, реализующего SAF, равен 100 пакетам.

Производительности решеток примерно одинаковы в условиях рабочей нагрузки. В условиях пиковой нагрузки решетка с узлом, реализующим технологию передачи пакетов с принудительной буферизацией, более устойчива, в отличие от сквозной коммутации, у которой производительность падает в 1,5 раза и более. Среднее время доставки пакета значительно ухудшается, примерно в 2 раза, у решетки с узлом SAF; при этом у решетки с узлом cut-through в 1,5 раза. Решетка с узлами cut-through приходит к полному тупику при интенсивности пиковой нагрузки wl=10.0 (характеристики решетки в таблице 2 отмечены символом $*$), решетка с узлами SAF – при интенсивности пиковой нагрузки, равной wl=8.0.

Таблица 2. Характеристики моделей решетки 8x8 с различной архитектурой узла при рабочей нагрузке

Table 2. Characteristics of 8x8 grid models with different node architecture under a workload

Интенсивность рабоч.нагрузки Workload intensity	Производ-ть решетки Grid perform. SAF	Средн.время доставки пак. Average packet delivery time	Производ-ть решетки Grid perform. $Cut-through$	Средн.время доставки пак. Average packet delivery time
90.0	0.34	78	0.34	15
50.0	0.62	78	0.61	15
40.0	0.78	79	0.77	15
30.0	1.03	79	1.03	15
20.0	1.55	81	1.54	15
10.0	3.1	97	3.09*	16*
9.0	3.44	130	3.44*	16*
8.0*	2.51	236	3.86	16
7.0*	2.16	301	4.41	16
6.0*	2.22	288	5.11	17
5.0*	3.41	231	5.81	19
4.0*	3.25	189	5.62	24

3.3. Оценка влияния злонамеренного трафика на параметры качества обслуживания решеток

Исследуем производительность решетки и среднее время доставки пакета в условиях злонамеренного трафика, с различной интенсивностью и пуассоновской функцией распределения. В таблице 3 представлены результаты исследования характеристик решетки для узлов без обязательной буферизации, интенсивность рабочей нагрузки wl=30.0. Значение интенсивности рабочей нагрузки выбрано для дальнейшего сравнения с результатами, полученными для SAF [7,9]. К параметрам таблицы, представленным в предыдущем подразделе, добавлено значение $gsnd$ – количество пакетов (выстрелов), отправленных пушками; количество операций (шагов), выполняемых моделью, осталось равным $Step$=10000000.

Результаты вычислительного эксперимента, представленные в таблице 3, показывают, что среднее время доставки пакета в решетке при интенсивности злонамеренного трафика в диапазоне 4.0–9.0 практически не изменяется, равно 15–16 MTU и равно среднему времени доставки в условиях рабочей нагрузки (таблица 1). Производительность решетки увеличивается, при максимальной интенсивности пушки gl=4.0 производительность решетки в 1.5 раза больше, чем при рабочей нагрузке, и равна gp=1.54 пакета за единицу модельного времени. Решетка не приходит в тупик, т.е. конфигурация «дуэль» не оказывает влияние на решетку при интенсивности рабочей нагрузки wl=30.0.

Таблица 3. Характеристики модели решетки 8x8 при разной интенсивности злонамеренного трафика

Table 3. Characteristics of an 8x8 grid model under ill-intentioned traffic with different intensity

Интенсивн. трафика пушки Traffic gun intensity	Время Time	Кол-во отправл-х пакетов Number of packets sent	Кол-во принятых пакетов Number of packets received	Кол-во пакетов выстрел-х из пушек N fired packets	Производи-тельность решетки Grid performance	Среднее время доставки пакета Average packetDT
4.0	607791	648331	931033	302864	1.54	16
5.0	646515	689875	926856	258690	1.43	16
5.5	662607	706690	925542	241035	1.39	16
6.0	675425	720558	923412	225116	1.36	16
8.0	715934	763637	918985	179083	1.28	15
9.0	730551	778991	916985	162344	1.28	15

Проведем сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла в условиях злонамеренного трафика, типа «дуэль трафика». Пушки присоединены к узлам решетки (рис. 3) с индексами (4,0) и (4,9). В таблице 4 представлены результаты исследования решеток с узлами, реализующими технологию передачи пакетов с обязательной буферизацией [9], и сквозной коммутации.

Таблица 4. Характеристики моделей решетки 8x8 с различной архитектурой узла в условиях злонамеренного трафика

Table 4. Characteristics of 8x8 grid models with different node architecture under an ill-intentioned traffic

Интенсивность трафика пушки Traffic gun intensity	Производит-ть решетки Grid performance SAF	Средн.время доставки пак. Average packet delivery time	Производит-ть решетки Grid performance $Cut-through$	Средн.время доставки пак. Average packet delivery time
4.0	0.34*	76	1.54	16
5.0	0.43*	73	1.43	16
5.5	0.67*	78	1.39	16
6.0	1.37	67	1.36	16
8.0	1.28	64	1.28	15
9.0	1.26	66	1.28	15

При интенсивности рабочей нагрузки wl=30.0 дуэль трафика приводит решетку с узлами, реализующими технологию SAF, к полному тупику через дополнительную нагрузку менее чем 10 процентов. Решетка с узлами, реализующими cut-through, работает в штатном режиме. Дальнейшие исследования показали, что конфигурация «дуэль трафика» не оказывает влияние на решетку с узлами cut-through при увеличении рабочей нагрузки до пиковой wl=15.0, при которой решетка приходит к полному тупику.

Построена модель вычислительной решетки с узлами, реализующими технологию сквозной коммутации пакетов, в форме раскрашенной сети Петри. Исследована безопасность решеток и возможность взаимоблокировок узлов решетки в условиях рабочей нагрузки. Для оценки параметров качества обслуживания в условиях злонамеренного трафика к модели решетки добавлены модели пушки пакетов, имитирующие «вредный» трафик. Проведен сравнительный анализ устойчивости вычислительных решеток с различной архитектурой узла в условиях рабочей нагрузки и злонамеренного трафика.

Таким образом, результаты имитационного моделирования подтвердили ранее полученные результаты, что современная архитектура коммутационных устройств не гарантирует безопасность решетки. Должны быть разработаны специальные протоколы для обнаружения и предотвращения взаимоблокировок, которые предполагают взаимодействие нескольких узлов.

Направлением дальнейших исследований является оценка эффективности и характеристик QoS решетки при произвольных функциях распределения генераторов трафика, типов тупиков, конфигураций устройств, формирующих скрытые атаки, построение рентерабельной модели для исследования структур решеток любого размера.

Список литературы / References

[1] Preve N. P., *Grid Computing: Towards a Global Interconnected Infrastructure*, Springer, 2011, 312 pp.

[2] Зайцев Д. А., Шмелёва Т. Р., Гуляев К. Д., *Отчет о научно-исследовательской работе «Разработка новых систем адресации глобальных сетей»*, номер госрегистрации 0108U008900, ОНАС, Одесса, 2009, 124 pp., (на украинском языке); [Zaitsev D. A., Shmeleva T. R., Guliaiev K. D., *Report on scientific-research work "Development of New World-wide Networks Addressing Systems"*, state registration number 0108U008900, ONAT, Odessa, 2009, 124 pp., (in Ukrainian).]

[3] Liberzon D., *Switching in Systems and Control*, Birkhauser, Boston, 2003, 230 pp.

[4] Sakun A. L., Zaitsev D. A., "An Evaluation of MPLS Efficacy using Colored Petri Net Models", *Proceedings of International Middle Eastern Multiconference on Simulation and Modelling (MESM'2008)*, Amman (Jordan), August 26–28, 2008, 31–36

[5] Shmeleva T. R., Zaitsev D. A., "Switched Ethernet Response Time Evaluation via Colored Petri Net Model", *Proccedings of International Middle Eastern Multiconference on Simulation and Modelling*, August 28–30, 2006, Alexandria (Egypt), 68–77

[6] Zaitsev D. A., Shmeleva T. R., "Parametric Petri Net Model for Ethernet Performance and Qos Evaluation", *Proceedings of 16th Workshop on Algorithms and Tools for Petri Nets*, September 25–26, 2009, University of Karlsruhe, Germany, 15–28

[7] Zaitsev D. A., Shmeleva T. R., Retschitzegger W. and Proll B., "Blocking Communication Grid via Ill-Intentioned Traffic", *14th Middle Eastern Simulation and Modelling Multiconference*, February 3–5, 2014, Muscat, Oman, 63–71

[8] Jensen K., Kristensen L.M., *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, 2009, 384 pp.

[9] Retschitzegger W., Proll B., Zaitsev D. A., Shmeleva T. R., "Security of grid structures under disguised traffic attacks", *Cluster Computing*, **19**:3 (2016), 1183–1200

[10] Shmeleva T. R., "Security of Grid Structures with Cut-through Switching Nodes", *System Informatics*, 2017, № 10, 23–32

Shmeleva T. R., "Comparative Analysis of Stability to Induced Deadlocks for Computing Grids with Various Node Architectures", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 193–206.

DOI: 10.18255/1818-1015-2018-2-193-206

Abstract. In this paper, we consider the classification and applications of switching methods, their advantages and disadvantages. A model of a computing grid was constructed in the form of a colored Petri net with a node which implements cut-through packet switching. The model consists of packet switching nodes, traffic generators and guns that form malicious traffic disguised as usual user traffic. The characteristics of the grid model were investigated under a working load with different intensities. The influence of malicious traffic such as ≪traffic duel≫ was estimated on the quality of service parameters of the grid. A comparative analysis of the computing grids stability was carried out with nodes which implement the store-and-forward and cut-through switching technologies. It is shown that the grids performance is approximately the same under work load conditions, and under peak load conditions the grid with the node implementing the store-and-forward technology is more stable. The grid with nodes implementing SAF technology comes to a complete deadlock through an additional load which is less than 10 percent. After a detailed study, it is shown that the ≪traffic duel≫ configuration does not affect the grid with cut-through nodes if the workload is increases to the peak load, at which the grid comes to a complete deadlock. The execution intensity of guns which generate a malicious traffic is determined by a random function with the Poisson distribution. The modeling system CPN Tools is used for constructing models and measuring parameters. Grid performance and average package delivery time are estimated in the grid on various load options.

Keywords: computing grid security, cut-through switching, traffic attack defence, performance evaluation, colored Petri net, deadlock

On the authors:
Tatiana R. Shmeleva, orcid.org/0000-0002-4799-3842, PhD,
A.S. Popov Odessa National Academy of Telecommunications,
1 Kuznechnaya str., Odessa 65029, Ukraine, e-mail: tishtri@rambler.ru

Acknowledgments: The author would like to thank Program Committee and the participants of PSSV-2017 workshop for discussion of some results presented in this paper.

Моделирование и анализ информационных систем. Т. 25, № 2 (2018), с. 207–216
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 207–216

Программно-конфигурируемые сети

©Morzhov S. V., Alekseev I. V., Nikitinskiy M. A., 2017

DOI: 10.18255/1818-1015-2018-2-207-216

UDC 004.415.25

Organization of Multi-controller Interaction in Software Defined Networks

Morzhov S. V., Alekseev I. V.[1], **Nikitinskiy M. A.**[1]

Received December 25, 2017

Abstract. *Software Defined Networking* (SDN) is a promising paradigm for network management. It is a centralized network intelligence on a dedicated server, which runs network operating system, and is called SDN controller. It was assumed that such an architecture should have an improved network performance and monitoring. However, the centralized control architecture of the SDNs brings novel challenges to reliability, scalability, fault tolerance and interoperability. These problems are especially acute for large data center networks and can be solved by combining SDN controllers into clusters, called multi-controllers. Multi-controller architecture became very important for SDN-enabled networks nowadays. This paper gives a comprehensive overview of SDN multi-controller architectures. The authors review several most popular distributed controllers in order to indicate their strengths and weaknesses. They also investigate and classify approaches used. This paper explains in details the difference among various types of multi-controller architectures, the distribution method and the communication system. Furthermore, it provides already implemented architectures and some examples of architectures under consideration by describing their design, communication process, and performance results. In this paper, the authors show their own classification of multi-controllers and claim that, despite the existence of undeniable advantages, all reviewed controllers have serious drawbacks, which must be eliminated. These drawbacks hamper the development of multi-controllers and their widespread adoption in corporate networks. In the end, the authors conclude that now it is impossible to find a solution capable to solve all the tasks assigned to it adequately and fully. The article is published in the authors' wording.

Keywords: SDN, software defined network, distributed controller, multi-controller, CoVisor, DISCO, ELASTICON, FlowBrick, FlowVisor, HyperFlow, Kandoo, ONIX, ONOS, ORION

For citation: Morzhov S. V., Alekseev I. V., Nikitinskiy M. A., "Organization of Multi-controller Interaction in Software Defined Networks", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 207–216.

On the authors:
Sergey V. Morzhov, orcid.org/0000-0001-6652-3574, graduate student,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia, e-mail: smorzhov@gmail.com

Igor V. Alekseev, orcid.org/0000-0001-8321-2399, Director of the Internet Center, Ph.D.,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl, 150003, Russia, e-mail: aiv@yars.free.net

Mikhail A. Nikitinskiy, orcid.org/0000-0001-8830-8613, system analyst, programmer,
A-Real Group, Energiya-Info Inc., 144 Souznaya str., Yaroslavl, 150008, Russia, e-mail: man@a-real.ru

Acknowledgments:
[1]The reported study was funded by RFBR according to the research project № 16-07-01103a.

Introduction

Specialists of the universities of Berkeley and Stanford in 2006 proposed the basic ideas of the SDN. This approach involves the construction of computer networks in such a manner that the control plane is separated from the data plane. The control plane is usually a dedicated server that runs network *operating system* (OS) called controller. Not only academic circles were inspired by the SDN architecture, some crucial commercial companies also became interested in it and formed the *ONF* (Open Networking Foundation) community. This community is developing specifications and standards in the field of SDN. According to ONF, the main ideas of the SDN approach are as follows:

- The control plane is separated from the data plane.
- Unified, vendor independent protocol called OpenFlow is used between planes.
- Control plane is logically centralized, implemented as a network OS, called controller, and runs some network applications.

The controller manages the network infrastructure and data streams in it, while network applications are designed to extend the controller's functionality and implemented on other equipment to facilitate the functioning of the controller. In fact, the controller - the core of the SDN - serves as a place where the logic defined by the network applications is converted into a set of switch configuration rules. Thus, the controller can become both a point of failure of the network and its performance bottleneck. If the controller fails, the operation will fail, leading to a complete or partial loss of network management. In this case, the switching equipment will continue to forward packets according to the current routing rules. However, if one or more switching equipment fails, the network will not be able to continue functioning. Therefore, a very important aspect when implementing the SDN approach for corporate, operators, providers or data centers networks is to ensure trouble-free operation of the controller.

To solve the above problem, the distributed interaction of controllers can serve, which will allow, if one controller fails, to retain control over the switching equipment. The presence of several controllers in the network implies the existence of a mechanism for exchanging control information between these controllers. The multi-controller architecture is a set of controllers working together to provide high performance and scalability. The organization of a multi-controller SDN can be logically or physically centralized, having one rank or hierarchical structure. Works in these areas are carried out within the framework of such projects as CoVisor, DISCO, ELASTICON, FlowBrick, FlowVisor, HyperFlow, Kandoo, ONIX, ONOS and ORION.

1. Distributed controller

1.1. CoVisor

CoVisor [1] is a hypervisor that allows you to use multiple controllers located in the same SDN to manage it. Unlike existing hypervisors, which are focused on splitting the network into subnets, CoVisor allows several controllers to simultaneously work with traffic. In

fact, CoVisor allows you to choose the best functionality from different controllers. For example, you can use the firewall application from the Floodlight controller, the load balancer from Ryu and so on, regardless of the programming language and libraries used. The resulting set of functions can be applied to traffic in whole or in part. CoVisor abstracts from a specific network topology and creates a virtual one in its place, which allows the administrator to regulate controller access to packages that it can monitor, modify and redirect. The main technical feature of CoVisor is a set of efficient algorithms that handle the policies of controllers and transform the virtual network into a set of specific OpenFlow rules, as well as ensuring their updating. The hypervisor prototype is based on the OpenVirteX hypervisor and is written in Java. CoVisor testing results show that it handles packets more quickly than OpenVirteX by several orders of magnitude.

1.2. DISCO

DISCO [2] is a SDN multi-controller. Each DISCO controller monitors its own local SDN and provides interconnection of controllers via unique communication channels. The control and transmission of information in these channels is carried out via the Messenger module using the AMQP protocol. The communication channels use specialized agents (Monitoring agent, Reachability agent, Connectivity agent, Reservation agent and others). Specialized agents and Messenger module are implemented in the inter-domain part of the controller, which is responsible for monitoring, synchronization, data exchange between DISCO controllers. The intra-domain part of the DISCO is based on the Floodlight controller. This part is responsible for the direct operation of the controller over the dedicated SDN or network applications. In their work [2], the authors evaluate experimentally the following: breaking the connection between different local networks and rebuilding the interconnection of controllers, processing traffic by controller m, which has priority from controller n, idle time when the controller is migrated to another virtual machine.

1.3. ELASTICON

ELASTICON [3] is a cluster of standalone controllers that share the workload to provide a global network view for the management layer. This global view of the network uses the Distributed Data Store module, through which a logically centralized controller is created. Each module has a TCP channel that connects it to neighboring controllers. This ensures the exchange of messages between controllers, the switching of responsibility zones or coordinating actions while switching equipment is migration process.

The following example can illustrate the typical use case of ELASTICON. The switching equipment is connected to several controllers. One of them is the master and the others are slaves. Each controller has a core controller module that is responsible for connecting the switching equipment and for interacting this equipment with the controller core applications. It should also be noted that this module is responsible for choosing which controller will be connected to the new switching equipment and for the migration of this equipment between the controllers. The controller core applications are responsible for the executable logic of the switching equipment. The information stored in them is transmitted via the Distributed Data Store module to a similar application

in another controller when the switching equipment is migrated. Balancing of switching equipment between controllers is performed periodically and does not depend on traffic load.

1.4. FlowBricks

This controller is based on the Floodlight kernel. FlowBricks [4] is built in such a way that it acts as an intermediary between various controllers and SDN. This approach was chosen because different parts of different controllers operate more or less efficiently. Thus, FlowBricks can send both sequentially and in parallel a packet-in to the controllers. The processing of responses from controllers can be configured and FlowBricks may decides how to modify the packet-out. The prototype of the controller showed that the processing time of the OpenFlow package has not increased significantly.

1.5. FlowVisor

FlowVisor [5] is an approach to organization of the SDN architecture that involves creating an additional layer for processing OpenFlow packets between the SDN controller and OpenFlow equipment. FlowVisor intercepts, modifies and redirects all packets exchanged between controllers and equipment, so that for each individual controller the network (or part of it) looks controlled only by it. In this way, the network is divided into layers that work independently. On the one hand, the advantage of this approach is the relative independence of a particular SDN controller, since the manipulations are done directly with OpenFlow packages, without affecting the controller's operation. As an analogue, you can consider the use of VLAN in a classical network. On the other hand, the administrator is obliged to configure, control and resolve conflicts between controllers. The system was based on the NOX controller.

1.6. HyperFlow

HyperFlow [6] is a network application for the NOX controller, which allows you to turn this controller into a distributed one. This is achieved through WheelFS [7], which provides passive data synchronization between network applications of controllers. This synchronization build on the publishing/subscribing events method. It ensures that the controller does not duplicate the transmitted information, and minimizes the amount of traffic transmitted by the controllers.

1.7. Kandoo

The Kandoo [8] project involves building a hierarchy of controllers in such a way that the lower level of the hierarchy is the controllers that manage their domains, and the top level is the root controller that controls the lower-level controllers. Low-level controllers process packet-in requests for which they have solutions, otherwise they send it to the top-level controller, which has a complete view of the entire SDN. For the Kandoo project to optimize the operation of the SDN two main applications have been created: detect and reroute. The first application works on the lower level of the controllers. It determines

large flows (large is considered to be a stream of more than 1 MB). They overflow the channels and reduce the efficiency of the SDN to transfer other flows. When one of the controllers determines a large flow, it passes this information to the upper level to rebuild the routing of all threads in the controlled SDN.

1.8. ONIX

ONIX [9] is a distributed controller that can be run on a cluster. At the same time on each node of the cluster - a dedicated server - several Onix controllers can be started. The SDN controlled by ONIX is divided into four logical components: a physical infrastructure that includes all network devices; the connection infrastructure, which is the connection between the physical network and ONIX; the management logic that is presented by the ONIX API for network applications and, finally, the ONIX controller itself, interacting with the physical infrastructure.

The ONIX API allows network applications to read and write the status of any network element in the SDN. Each network element can contain one or more data objects to which network applications should have access. In the NIB information database, each state of the network is stored as a graph. ONIX controllers support NIB synchronization and each network applications can read and write status in NIB.

For scalability, the ONIX controller offers three methods: the dynamic load sharing between multiple copies of the controller, the creation of a hierarchical structure with the representation of aggregation of controllers at the lower level and the ability to transfer the state of applications.

1.9. ONOS

The first prototype of the ONOS SDN controller [10] was based on the Floodlight. It has the switch manager, I/O loop, link discovery, module management, and REST APIs. The main characteristics of this prototype were global network view, scalability and fault tolerance. To ensure higher performance, ONOS was build using the following frameworks. Firstly, it is the Titan – a distributed graph database, which is storing the network topology. Secondly, it is Cassandra – a distributed No-SQL database, which is implemented as a key-value storage and were used for storing information about switches, ports and so on. Thirdly, is Blueprints API – the interface, on which network applications receive information about the current network topology.

If the global network view has changed, OpenFlow managers detect it and implement changes in the real network. Each controller, in this architecture, is responsible for its own domain. In this case, each switch is connected to several controllers. If a controller stops responding, then using the ZooKeeper algorithm, there are elections among the active controllers, who will take responsibility for managing the SDN slice that is left without a controller.

The main drawback of the proposed approach is low performance due to the high overhead required to update the network topology stored in Titian and Cassandra. With the fall of one controller, the reorganization of the SDN lasted up to 30 seconds, which is unacceptable.

While creating the second prototype, the focus was on improving the performance of the first one, keeping the Global network view. In this case, there are two main areas where the performance gain can be got. The first one concerns making remote operations as fast as possible, while the second approach focuses on reducing the number of remote operations. It was decided to change the Titan, Cassandra and Blueprint API stack to faster in-memory DBs. The creators chose RAMCloud DB, because it provides an interface compatible with the Blueprint API. In addition, the authors suggested using topology cache. The essence of this technology lies in the fact that topology changes are not added immediately in the DB, but store in the cache. If after some time there are no any new changes, then the cache data is inserted to the DB. In addition, to improve the inter-controller communication through notifications, the authors applied inter-instance publish-subscribe event notification and communication system based on Hazelcast. With the help of it, several notification queues have been created. Network applications can subscribe to them, and, thus, they will quickly receive the information they need.

1.10. ORION

ORION [11] is a hybrid hierarchical control plane, which is a mix of flat and hierarchical architectures. It tries to combine the benefits of both designs and put them all together in a hybrid structure. A network controlled by ORION has three layers: the physical layer, which contains all the physical devices, such as OpenFlow switches; the bottom layer of the control plane that includes the controllers, which handle collecting physical device and link information, as well as dealing with intra-area requests and updates; and, finally, the upper layer, which contains the domain controllers.

ORION interdomain controllers' communication relies on the Horizontal Communication Module that synchronizes the information among domain controllers to build a global network view. The interarea and domain controllers' communication relies on the Vertical Communication Module, which is a set of TCP connections that permit area controllers to send the abstracted topology of the infrastructure layer and request information from the domain controller when a host in some domain wants to reach a particular host in another domain.

ORION has made a theoretical and an experimental evaluation to test the performance of its control plane. On one hand, the theoretical evaluation shows that the computing time of ORION has a linear growth, which is much lower than the traditional Dijkstra routing algorithm. On the other hand, the experimental evaluation tried to verify the feasibility and the effectiveness of ORION shows that the delay time increases gradually.

2. Distribution systems analysis

After analyzing the presented multi-controllers and the principles of organizing a distributed, scalable and fault-tolerant control system for SDN, we classify the controllers into subgroups and define the methodology for creating the distributed SDN.

Fig. 1 shows the classification of multi-controllers. Physically centralized multi-controllers are monocontrollers with multi-threaded organization of event processing. For examples, NOX and Beacon controllers, as well as the CoVisor, FlowBrick or FlowVisor

hypervisors are typical representatives of this group. Physically distributed multi-controllers are controllers running as separate instances. Logically distributed multi-controller is a cluster of SDN controllers that are responsible for different domains and perform various functions. Depending on the functionality, they can have hierarchical (Kandoo) or flat (ORION) architecture. Logically centralized multi-controllers performs a single task in one domain, which is distributed between them depending on the total load. They can have both a dynamic structure (the number of instances increases and decreases as necessary) – DISCO, ONIX, ONOS, and the static structure (the number of instances is invariable) – Elasticon, HyperFlow. Representatives of this class have flat architecture.

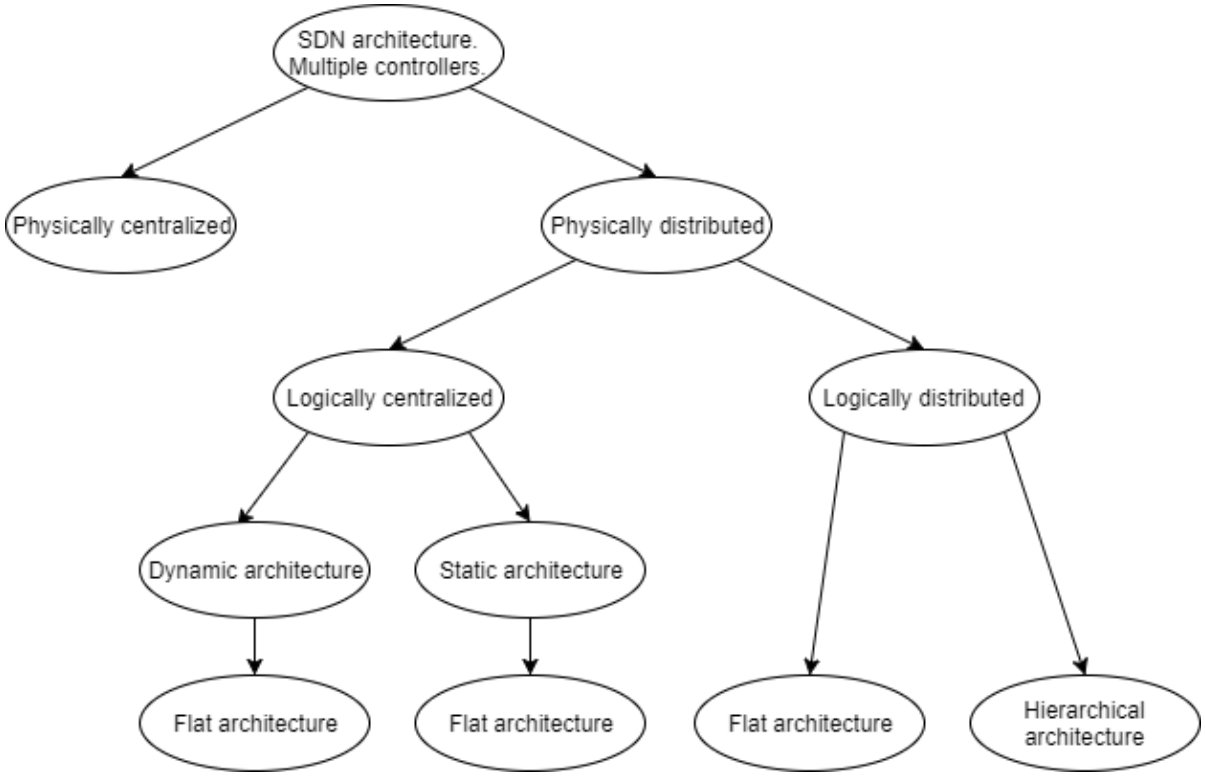

Fig 1. The classification of multi-controllers

According of the above-mentioned review of controllers, the distributed system can be developed with the help of:

- hypervisors – physically centralized controllers;
- special applications functioning as a mediator between the controller and network applications (HyperFlow);
- special controller core modules linked to corresponding module of another controller via TCP channel. Most multi-controllers function this way.

The last approach is the most scalable and fault-tolerant. Developers implement it using publish-subscribe event notification mechanism. Synchronization is achieved by

updating the information on the monitored SDN and the actions performed on it between different controllers.

Despite the fact that the main ideas of the SDN approach are the vendor dependence reduction and centralized network management, the developers of distributed controllers again impose a restriction that involves using only their multi-controller. On the one hand, there are hypervisors that can solve this problem, but they do not solve the bottleneck problem. On the other hand, modern controllers are just a tool for collecting information and installing logic into network applications, while controllers and network applications are controlled by a higher-level entity, for example, the ETSI MANO standard solutions. Thus, to make this system to functioning properly, both highly qualified specialists and equipment of the data center level are required.

A possible solution for creating distributed interaction between different controllers and various network applications, without using a higher level entity capable for managing a distributed SDN that is not a data center, is the creation of a unified kernel module which should follow several requirements. Firstly, the interaction between controllers and network applications must be performed using the RPC method, since many controllers already use it. Secondly, the interconnection between the modules should be performed via an encrypted TCP channel. Thirdly, the module must have different APIs for interaction with various controllers and network applications. In fact, this module should become a distributed hypervisor at the core level of the controller with the ability to interact with network applications.

3. Conclusion

To create a unified kernel module, the authors developed the PreFirewall [12] network application and a random rule generator for the firewall of the Floodlight controller core module. The generator allows you to simulate the interaction of an arbitrary network application with the SDN controller. In addition, the external module of the Floodlight controller core Topology Tracker [13] was developed for storing and exchanging information on the current network topology. It also allows subscribing to events of a similar module with a distributed controller. The authors designed an external module of the Floodlight SDN controller called DEventBus [14]. It is able to send events that occurred in module Topology Tracker to network applications signed for them. The DEventBus module is a prototype for the organization of interaction between multi-controller and network applications.

In the future, the authors are planning to unify this module for organizing multi-controller interaction. For this purpose, we are going to implement various API algorithms for checking the availability of resources, opening and supporting communication channels.

References

[1] X. Jin, J. Gossels, J. Rexford, D. Walker, “CoVisor: A Compositional Hypervisor for Software-Defined Networks”, Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’15) (May 4–6, 2015, Oakland, CA, USA), 87–101.

[2] K. Phemius, M. Bouet, and J. Leguay, “DISCO: distributed multi-domain SDN

controllers", Proceedings of the IEEE Network Operations and Management Symposium (NOMS'14) (Krakow, Poland, May 2014), 1–4.

[3] A. Dixit et al., "Towards an elastic distributed SDN controller", Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13) (ACM, Hong Kong, 2013), 7–12.

[4] A. Dixit, K. Kogan, P. Eugster, "HyperFlow: a distributed control plane for OpenFlow", Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols (October 21 – 24, 2014), 287–292.

[5] L. Liao, A. Shami, V.C.M. Leung, "Distributed flowvisor: a distributed flowvisor platform for quality of service aware cloud network virtualisation", *IET Networks*, **4**:5 (2015), 270–277.

[6] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow", Proceedings of the Internet Network Management Conference on Research on Enterprise Networking (INM/WREN'10) (Berkeley, Calif, USA, 2010), 1–6.

[7] J. Stribling et al., "Flexible, wide-area storage for distributed systems with WheeIFS", Proceedings of the 6th USENIX symposium on Networked systems design and implementation (Boston, Massachusetts, April 22 – 24, 2009), 43–58.

[8] S.H. Yeganeh, "Kandoo: a framework for efficient and scalable offloading of control applications", Proceedings of the 1st ACM Workshop on Hot Topics in Software Defined Networks (HotSDN'12) (Helsinki, Finland, August 2012), 19–24.

[9] M.T. Koponen et al., "Onix: a distributed control platform for largescale production networks", Proceedings of USENIX Operating Systems Design and Implementation (OSDI'10) (October 4-6, Vancouver, Canada, 2010), 351–364.

[10] U. Krishnaswamy et al., "ONOS: an open source distributed SDN OS", Proceedings of the third workshop on Hot topics in software defined networking (HotSDN'14) (Chicago, Illinois, USA, August 22, 2014), 1–6.

[11] Y. Fu et al., "Orion: a hybrid hierarchical control plane of software-defined networking for large-scale networks", Proceedings of the 22nd IEEE International Conference on Network Protocols (ICNP'14) (Raleigh, NC, USA, October 2014), 569–576.

[12] S. Morzhov, I. Alekseev, M. Nikitinskiy, "Firewall application for Floodlight SDN controller", International Siberian Conference on Control and Communications (SIBCON) (12–14 May, 2016, Moscow, Russia), 1–5.

[13] A.A. Noskov, M.A. Nikitinskiy, I.V. Alekseev, "Development of an active external network topology module for Floodlight software-defined network controller", *Automatic Control and Computer Sciences*, **50**:7 (2016), 546–551.

[14] I. Alekseev, M. Nikitinskiy, "EventBus Module for Distributed OpenFlow Controllers", Proceedings of the 17th Conference of Open Innovations Association FRUCT (20–24 April, 2015, Yaroslavl, Russia), 3–8.

Моржов С. В., Алексеев И. В., Никитинский М. А., "Организация мультиконтроллерного взаимодействия в программно-конфигурируемых сетях", *Моделирование и анализ информационных систем*, **25**:2 (2018), 207–216.

DOI: 10.18255/1818-1015-2018-2-207-216

Аннотация. Программно-конфигурируемая сеть (ПКС) – это перспективная парадигма управления сетью, в которой для повышения производительности уровень управления сетью отделен от уровня передачи данных и реализуется программно на выделенном сервере. Несмотря на очевидные преимущества подхода централизованной архитектуры управления ПКС, она создает новые проблемы, связанные с надежностью, масштабируемостью, отказоустойчивостью и интероперабельностью сети. Эти проблемы встают особенно остро для больших сетей дата-центров

и решаются путем объединения нескольких контроллеров ПКС в кластер, называемый мульти-контроллером. В данной статье представлен обзор некоторых наиболее популярных мульти-контроллеров ПКС, выделены их сильные и слабые стороны, а также приведена классификация используемых ими подходов к организации распределенного взаимодействия. Подробно рассматриваются различия между несколькими типами архитектур мульти-контроллеров, среди которых есть как находящиеся на этапе разработки, так и успешно функционирующие в данное время в дата-центрах. Авторы на примере разработанной ими классификации мульти-контроллеров показывают, что, несмотря на наличие неоспоримых преимуществ, все рассмотренные контроллеры имеют недостатки, которые необходимо устранить. Устранение данных недостатков поможет развитию мульти-контроллеров и сделает возможным их широкое использование в корпоративных сетях. В заключение авторы приходят к выводу, что на данный момент нельзя найти решение, способное в полной мере решить все поставленные задачи. Статья публикуется в авторской редакции.

Ключевые слова: ПКС, программно-конфигурируемая сеть, распределенный контроллер, мульти-контроллер, CoVisor, DISCO, ELASTICON, FlowBrick, FlowVisor, HyperFlow, Kandoo, ONIX, ONOS, ORION

Об авторах:
Моржов Сергей Владимирович, orcid.org/0000-0001-6652-3574, магистрант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: smorzhov@gmail.com

Алексеев Игорь Вадимович, orcid.org/0000-0001-8321-2399, канд. физ.-мат. наук, директор Интернет центра,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: aiv@yars.free.net

Никитинский Михаил Александрович, orcid.org/0000-0001-8830-8613, программист-аналитик,
ООО «Энергия-Инфо», ул. Союзная, 144, г. Ярославль, 150008 Россия, e-mail: man@a-real.ru

Благодарности:
Работа выполнена при финансовой поддержке РФФИ, проект № 16-07-01103а.

Моделирование и анализ информационных систем. Т. 25, № 2 (2018), с. 217–231
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 217–231

Модели
технических систем

©Терехов С. М., Немтинов В. А.,Корнилов К. С., 2017

DOI: 10.18255/1818-1015-2018-2-217-231

УДК 004.942

Математическая модель подключения оптимального числа потенциальных потребителей тепла к тепловой сети

Терехов С. М., Немтинов В. А., Корнилов К. С.

получена 11 декабря 2017

Аннотация. В современном мире эффективное использование энергоносителей является крайне важным аспектом человеческой деятельности. В частности, системы теплоснабжения имеют значительное экономическое, экологическое и социальное значение как для потребителей тепла, так и для теплоснабжающих организаций. От эффективности функционирования систем теплоснабжения зависит экономическое состояние всех участников процесса теплоснабжения. От надежности функционирования систем зависят жизненно важные процессы, такие как работа больниц и промышленных предприятий. При такой тесной сетевой коммуникации критически важно безотказное и эффективное функционирование систем энергоснабжения. В данной статье рассмотрены пути повышения эффективности работы систем теплоснабжения. Представлена математическая модель для планирования работы систем теплоснабжения путем подключения оптимального множества новых потребителей тепла. Для отдельно взятого потенциального потребителя, каждый раз, когда возникает альтернативный вариант подключения этого потребителя к существующей тепловой сети, возможно выбрать единственное оптимальное решение. Это становится возможно за счет наложения ограничений и процедуры отбора вариантов из подмножества бинарных переменных, соответствующих альтернативам. Представлена процедура поиска оптимального числа потребителей для подключения к существующей тепловой сети, являющаяся обоснованием для увеличения числа существующих потребителей. Проведено тестирование и представлены результаты работы математической модели на примере тестовых тепловых сетей, сконфигурированных на основе ручного ввода основных условий и параметров работы. Определены направления дальнейших исследований по повышению эффективности систем теплоснабжения и интеграции представленной математической модели с современными программными комплексами.

Ключевые слова: системный подход, система теплоснабжения, оптимизация

Для цитирования: Терехов С. М., Немтинов В. А.,Корнилов К. С., "Математическая модель подключения оптимального числа потенциальных потребителей тепла к тепловой сети", *Моделирование и анализ информационных систем*, **25**:2 (2018), 217–231.

Об авторах:
Терехов Сергей Михайлович, orcid.org/0000-0002-4103-1167, аспирант,
Тамбовский государственный технический университет,
Советская ул., 106, г. Тамбов, 393000 Россия, e-mail: Stpodiumz3@gmail.com

Немтинов Владимир Алексеевич, orcid.org/0000-0003-2917-3610, д-р техн. наук, профессор,
Тамбовский государственный технический университет,
Советская ул., 106, г. Тамбов, 393000 Россия, e-mail: nemtinov@mail.gaps.tstu.ru

Корнилов Кирилл Сергеевич, orcid.org/0000-0003-1729-8360, младший научный сотрудник,
Тамбовский государственный технический университет,
Советская ул., 106, г. Тамбов, 393000 Россия, e-mail: i_sirotinka@rambler.ru

Введение

Существующие методики проектирования и расчета [1–2] систем теплоснабжения не позволяют с высокой степенью точности планировать работу тепловых сетей с минимальными издержками из-за большого количества неучтенных факторов для каждой отдельно взятой системы. Потребность в оптимизации работы таких систем обусловлена высокой стоимостью генерации тепла и прогнозируемым [3] ростом цен на энергоносители.

В связи с потребностью в оптимизации существующих тепловых сетей в данной работе рассматриваются вопросы выбора оптимального множества новых потребителей для подключения к существующей системе теплоснабжения при максимизации реализуемой тепловой энергии и минимизации затрат на реконструкцию тепловых сетей. С помощью представленной математической модели подключения оптимального числа потребителей, а также процедуры поиска оптимального числа потребителей, производится расчет состояния устойчивого функционирования системы с учетом технических требований к существующей системе.

1. Постановка задачи исследования

Задача исследования – выбор оптимального множества новых потребителей $opt(V_P)$ для подключения к существующей системе теплоснабжения при максимизации реализуемой тепловой энергии $max(PI_v)$ и минимизации затрат на реконструкцию сетей $min(C_r)$.

Задача состоит в определении подмножества потенциальных потребителей, которых необходимо подключить к существующей тепловой сети, для максимизации объема реализуемой энергии при определенных теплогидравлических возможностях тепловой сети, в соответствии со всеми требованиями [4] к качеству услуг, экологии, безопасности, предъявляемым к тепловым сетям.

Обоснованием для увеличения числа существующих потребителей, а значит, наращивания теплопроизводительности тепловой сети является сопоставление фактических и максимальных показателей функционирования тепловой сети. Индекс фактического u^s_{fact} и индекс потенциального u^s_{potn} использования ресурсов существующей системы теплоснабжения являются условными обобщенными показателями степени загруженности тепловой сети.

Если по результатам анализа $u^s_{fact} \geq u^s_{potn}$ – тепловая сеть перегружена или работает на пределе своих технических возможностей, при $u^s_{fact} < u^s_{potn}$ существует возможность увеличить теплопроизводительность тепловой сети, а значит, подключить потенциальных потребителей тепла:

$$u^s_{potn}(R(x)) = (Q^s_{max}, opt(V_P))|\xi^s_{sum}, RI_i, C_r; \tag{1}$$

$$u^s_{fact}(R(x)) = (Q^s_{fact}, ct)|\xi^s_{sum}, RI_i, C_r. \tag{2}$$

Если $u^s_{potn}(R(x)) > u^s_{fact}(R(x))$, тогда постановка подзадачи определения опти-

мального множества потенциальных потребителей в общем виде будет выглядеть так:

$$opt(V_p) = (Q^s_{max}, opt(V_p), R(x))|\xi^s_{sum}, RI_i, C_r; \quad (3)$$

где Q^s_{max} , $opt(V_p)$ – критерии оптимальности поставленной задачи, $R(x)$ – режим устойчивого функционирования системы, $Rl^n_i, n := (0, 1)$, $i \in I$ – исправность нерезервируемого i-элемента системы, C_r – экономические ограничения в общем виде, ξ^s_{sum} – совокупность интервальных факторов, влияющих на функционирование системы теплоснабжения.

Режим ($R(x)$) функционирования системы теплоснабжения представляет собой определенный набор параметров работы $(F1x, \ldots, Fkx)$, изменение которых зависит от изменения интервальных факторов ξ^s_n.

Структура внешних и внутренних интервальных факторов, влияющих на процесс теплоснабжения: $\xi^s_{sum} = (\xi^s_{11}, \xi^s_{12}, \xi^s_2, \xi^s_3, \xi^s_4, \xi^s_5)$, где ξ^s_{11} – вектор тепловой нагрузки на отопление в момент времени t; ξ^s_{12} – вектор тепловой нагрузки на горячее водоснабжение в момент времени t; ξ^s_2 – вектор расхода сетевой воды в системе в момент времени t; ξ^s_3 – вектор температуры внешней среды в момент времени t; ξ^s_4 – вектор неисправностей n-элемента системы S в момент времени t; ξ^s_5– вектор неисправностей в i-узле.

2. Математическая модель системы

Представим математическую модель подключения оптимального числа потребителей, основанную на графовом представлении системы теплоснабжения, в которой узлы и дуги графа соответствуют некоторым элементами или группам элементов сети теплоснабжения. Рисунок 1 иллюстрирует пример графового представления схемы сети теплоснабжения. Каждый узел (обычно это тепловая камера или тепловой пункт) тепловой сети представлен на графе в виде пары узлов: один с положительным индексом ассоциируется с питающей линией, второй парный узел имеет отрицательный индекс и ассоциируется с обратным трубопроводом. Графовая модель системы позволяет идентифицировать источник тепла, потребителей тепла и промежуточные узлы тепловой сети.

Ориентация дуг графа соответствует направлениям движения теплоносителя в питающей и возвратной линии тепловой сети. Существующие связи отмечены сплошными дугами, потенциальные связи с потенциальными потребителями отмечены на графе пунктирными линиями.

Компоновка элементов существующей тепловой сети связана с начальной конфигурацией из одного или нескольких источников тепла и подключенным к этим источникам тепла множеством потребителей тепла.

Дано: ориентированный граф $G = (V, A)$, где V– множество узлов, A — множество ребер графа. Множество узлов включает в себя узлы v с положительным индексом $v > 0$, принадлежащие питающей линии теплосети, узлы с отрицательным индексом – v, принадлежащие обратной линии теплосети. Множество V раскладывается на ряд соответствующих подмножеств:

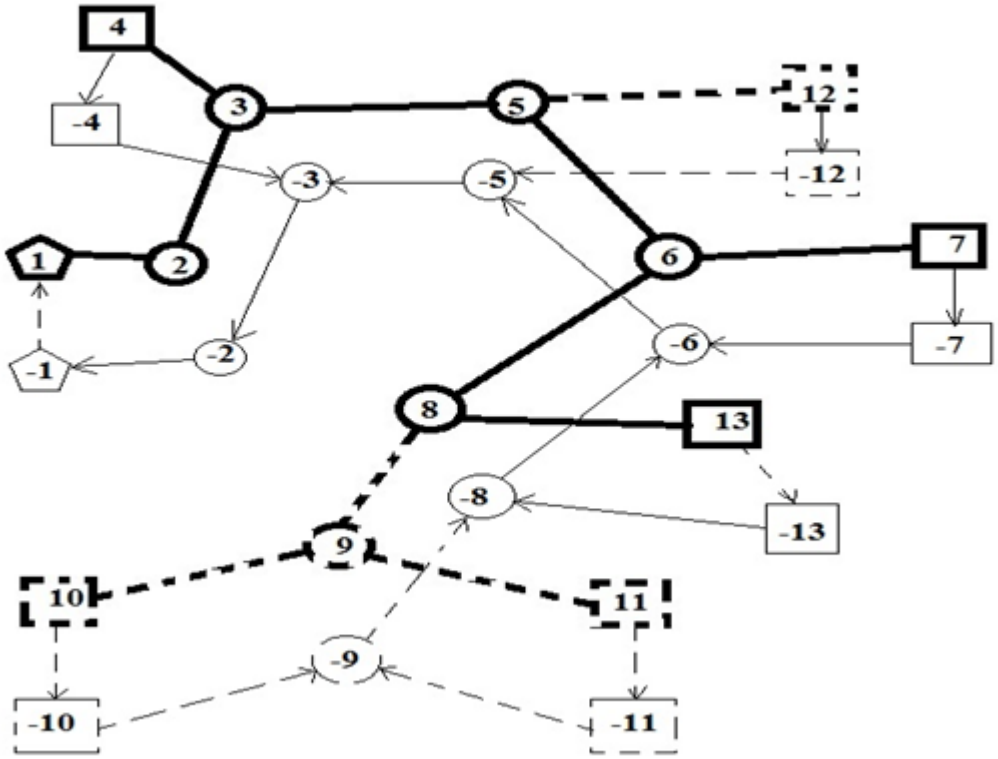

Рис. 1. Графовое представление типовой тепловой сети
Fig. 1. Graph representation of a typical heat network

V_I – множество источников тепла;
V_S – множество узлов существующих потребителей тепла;
V_P – множество узлов потенциальных потребителей тепла;
V_T – множество узлов тепловых камер, $V_T = V_{TE} \cup V_{TP}$,
где V_{TE} – подмножество существующих узлов-тепловых камер;
V_{TP} – подмножество потенциальных узлов-тепловых камер;
V_E – множество всех существующих узлов теплосети, включая потребителей тепла, $V_E = V_S \cup V_{TE}$.
Множество дуг A в свою очередь раскладываем на пять подмножеств:
$A = A_R \cup A_S \cup A_F \cup A_P \cup A_I$.
Множество $A_F = A_{FE} \cup A_{FP}$ включает все линии подающего трубопровода, как существующие – подмножество A_{FE}, так и потенциальные – подмножество A_{FP}.
Множество $A_R = A_{RE} \cup A_{RP}$ включает все линии обратного трубопровода, как существующие A_{RE}, так и потенциальные A_{RP}.
Аналогично A_S – множество существующих концевых линий теплообмена, A_P – множество потенциальных концевых линий теплообмена, A_I – множество линий трубопровода на источниках тепла.

Спрос на тепло каждого отдельного потребителя $v \in (V_S \cup V_P)$ рассчитываем

в виде требуемого теплового потока μ_e теплоносителя, соответствующего дуге расчетного потребителя $e = (v, -v) \in (A_S \cup A_P)$.

Общая дуга линии подачи тепла обозначается как $e = (i, j) \in A_F$, а соответствующая дуга линии обратного трубопровода обозначается как $r(e) = (-j, -i) \in A_R$. Дуги, соединяющие потребителей $(v, -v), v \in (V_S \cup V_P)$. Дуги в пунктах теплоснабжения представлены как $(-v, v), v \in V_I$. Для каждой дуги, относящейся к линии трубопровода $e = (i, j) \in (A_F \cup A_R)$, определяем обычную ориентацию от узла i до узла j, где $i < j$.

Экономические параметры математической модели: прибыль будущих периодов от подключения потенциальных потребителей к сети теплоснабжения, затраты на монтаж и наладку тепловых сетей для подключения потенциальных потребителей. Для каждой дуги, связывающей теплообменник с потенциальными потребителями $(e = (v, -v) \in A_P)$, параметр R_e означает чистую прибыль от подключения этой дуги к тепловой сети. Параметр чистой прибыли означает разность между доходом будущих периодов (выбирается расчетный временной горизонт T, обычно 10 лет и более) от реализованной тепловой энергии подключенным потенциальным потребителям и затратами на техническое подключение расчетных потребителей. Более того, для каждой потенциальной линии трубопровода (дуги $e = (i, j) \in A_{FP}$) затраты C_e составляют общую стоимость монтажа и эксплуатации этой линии, включая: материалы, топливо, монтаж, пуско-наладочные работы, обслуживание, оплату труда. Для упрощения математической модели будем ассоциировать стоимость прокладки питающей линии тепловой сети со стоимостью прокладки смежной обратной линии тепловой сети.

Физические характеристики тепловой сети описываются несколькими параметрами. Давление на каждом из функционирующих узлов теплосети должно быть выше минимально установленного значения P^{min}, тогда как на каждом из узлов источника тепла $v \in V_I$ давление теплоносителя не должно превышать максимально предельное значение давления для подающей линии P_v^{max}. Каждая дуга, ассоциированная с трубопроводом $e = (i, j) \in (A_F \cup A_R)$, ограничена предельно максимальной тепловой нагрузкой μ_e^{max}, зависящей от диаметра трубы и заданной максимальной скорости теплоносителя. Для каждой дуги, ассоциированной с потребителем, $e = (v, -v) \in (A_E \cup A_P)$, уровень снижения давления должен быть выше предельно минимального значения ΔP^{min}.

Задача исследования заключается в нахождении подмножества потенциальных потребителей тепла, которые могут быть присоединены к существующей системе теплоснабжения в соответствии с принципами максимизации чистой прибыли и в рамках технических возможностей системы теплоснабжения обеспечивать тепло потребителей в соответствии с предъявляемыми требованиями к объему предоставляемой тепловой энергии. Как было сказано, системы теплоснабжения проектируются с учетом возможности функционирования системы в пиковых нагрузках. Будем использовать бинарные переменные x_e, $e = (v, -v) \in A_P$, обозначающие состояние у отдельного потребителя или множества потребителей, x_e принимает значение 1, если потребитель $v \in V_P$ подключен к сети, и 0 в ином случае. Бинарные переменные y_e, $e = (i, j) \in A_{FP}$ обозначают состояние функционирования отдельной питающей линии тепловой сети, y_e принимает значение 1, если выбранная линия трубопрово-

да используется оптимально, в ином случае – 0. Для отражения гидравлического режима функционирования системы будем использовать переменные узлового давления P_v, $v \in V$, переменные потерь давления в трубах ΔP_e, переменные потока теплоносителя в трубах $m_e^{'}$, $e \in A$. В результате получаем математическую модель функционирования системы согласно указанным параметрам работы:

$$f(R_x^s) = \sum_{e \in V_P V_e} (Q_e^{V_P V_E}, P_v, m_e^{'}, x_e, m_e^{'}, y_e, \Delta P_e x_e), \tag{4}$$

$$f(R_e, C_e) = max \sum_{e \in A_P} R_e x_e - min \sum_{e \in A_{FP} C_e y_e}, \tag{5}$$

$$f(R_e) = max \sum_{e \in A_P} (R_e x_e + R_{ep}), \tag{6}$$

$$f(C_e) = min \sum_{e \in A_{FP}} (akC_e y_e + bC_e y_e + cC_e y_e + dC_e y_e), \tag{7}$$

$$C_e^{V_P V_E} \leq \Delta C_e^{max} \forall v \in V, \tag{8}$$

$$m_e^{'} = \mu_e \forall e \in A_S, \tag{9}$$

$$m_e^{'} = \mu_e x_e \forall e \in A_P, \tag{10}$$

$$m_e^{'} \leq \mu_e^{max} x_e \forall e \in A_{FE}, \tag{11}$$

$$m_e^{'} \leq \mu_e^{max} y_e \forall e \in A_{FP}, \tag{12}$$

$$m_{r(e)}^{'} = m_e^{'} \forall e \in A_F, \tag{13}$$

$$\sum_{e=(i,v) \in E} m_e^{'} - \sum_{e=(v,j) \in E} m_e^{'} = 0 \forall v \in V, \tag{14}$$

$$P_v \leq P_v^{max} \forall v \in V_I, \tag{15}$$

$$P_v \geq P^{min} \forall v \in V, \tag{16}$$

$$\Delta P_e \leq \Delta P^{max} \forall e \in A_I, \tag{17}$$

$$\Delta P_e \geq \Delta P^{min} \forall e \in A_S, \tag{18}$$

$$\Delta P_e \geq \Delta P^{min} x_e \forall e \in A_P, \tag{19}$$

$$\Delta P_e = f_L(m_e^{'}) \forall e \in A_F \cup A_R, \tag{20}$$

$$m_e^{'} \geq 0 \forall e \in A_F \cup A_R, \tag{21}$$

$$P_v \geq 0 \forall v \in V, \tag{22}$$

$$\Delta P_e \geq 0 \forall e \in A, \tag{23}$$

$$Q_e^{V_P V_E} \leq 0,85 \cdot Q_e^{max} \forall v \in V, \tag{24}$$

$$Q_e^{V_P} \geq 0 \forall e \in A, \tag{25}$$

$$Q_e^{V_e} \geq 0 \forall e \in A, \tag{26}$$

$$Q_e^{V_P} \leq Q_e^{max} \forall e \in A, \tag{27}$$

$$Q_e^{V_e} \leq Q_e^{max} \forall e \in A, \tag{28}$$

$$x_e \in (0,1) \forall e \in A_P, \tag{29}$$

$$y_e \in (0,1) \forall e \in A_{FP}. \tag{30}$$

Целевая функция (5) максимизации чистой прибыли при оптимальной работе системы определяется как разность между чистой прибылью от выручки будущих периодов и стоимостью технического присоединения потенциальных потребителей к тепловой сети.

Ограничения данной математической модели подключения оптимального числа потребителей могут быть сгруппированы в четыре категории. В первой категории ограничения (9) и (10) определяют расход $m_e^{'}$ в р-узлах, относящихся к потребителям: существующим потребителям, узловые расходы которых определены расчетным тепловым спросом этих потребителей. Вместе с тем $x_e = 1$ для потенциальных потребителей только в том случае, когда эти потенциальные потребители уже подключены к существующей тепловой сети. Неравенства (11), (12) определяют верхнюю границу теплового потока сети. Там, где верхняя граница теплового потока равна нулю, – потенциальная линия тепловой сети не используется, $y_e = 0$. Тепловой поток обратного трубопровода $r(e)$ принимает значение (13) эквивалентно значению питающей линии на рассматриваемом участке.

Вторая категория ограничений относится к узлам тепловой сети и отражает связь источников тепла с потребителями по трубопроводу и другим элементам, относящимся к рассматриваемым участкам тепловой сети. Уравнение (14) иллюстрирует баланс входящего и исходящего потоков каждого рассматриваемого узла

тепловой камеры. Нижняя граница давления в узлах тепловых камер установлена в (16), в то время как верхние границы давления на i-узлах источников тепла установлены неравенством (15).

Третья категория ограничений относится к потерям давления на участках трубопровода тепловой сети. Максимально установленное значение потерь давления теплоносителя на источнике теплоснабжения содержится в (17), наряду с минимально допустимыми потерями давления концевых существующих (18) и потенциальных потребителей (19). Линеаризованная зависимость между расходом и давлением в системе отражена в равенстве (20). Важно отметить, что существуют различные нормы потерь давления для различных типов и профилей трубопровода, соответственно разные профили труб в дальнейшем развитии математической модели должны быть описаны, например, средствами выбора контрольных точек $b_0 ... b_n$ и аппроксимации функции потери давления в системе теплоснабжения по большему количеству контрольных точек для более точных результатов.

Четвертая группа ограничений (21), (22), (23) включает в себя множество нижних оценок для непрерывных переменных расхода теплоносителя и давления в трубопроводе. Присвоение бинарных переменных указывается в (29), (30).

Представленная выше математическая модель подключения оптимального числа потребителей имеет общий вид и не позволяет решать прикладные задачи проверки возможности подключения потенциальных потребителей к тепловой сети. Это связано с неполнотой математической модели системы. Так, например, направление потока теплоносителя вдоль подающей и обратной линии трубопровода определено неявно. Для определения направления потока в распределительных и магистральных сетях используются соответствующие бинарные переменные. Кроме того, переменные давления и переменные расхода для существующих узлов тепловой сети должны быть дублированы для подающей и обратной линий соответственно, чтобы учесть направление потока и потери давления в рассматриваемом направлении.

Некоторые практические требования, предъявляемые к системе теплоснабжения, могут быть включены в представленную математическую модель подключения оптимального числа потребителей введением ограничений. Введены ограничения, обусловленные потребностью в ограничении максимальной величины давления теплоносителя на отдельных участках тепловой сети аналогично с (15), (17) для подмножества узлов и дуг графовой модели тепловой сети. Для учета стоимости передачи теплоносителя, учета минимальных значений расхода и давления теплоносителя добавляем в целевую функцию представленной математической модели предельные ограничения и стоимостные величины, подходящие для отдельно взятого примера тепловой сети.

Экономические параметры также должны быть учтены путем добавления ограничений на совокупные затраты на реализацию проекта подключения потенциального потребителя. Объем совокупных затрат может сильно различаться для отдельных потенциальных потребителей (например, частного сектора или промышленного предприятия) и зависит от согласования объема финансирования с потенциальным потребителем.

Стоимость подключения отдельного узла формируется следующим образом (31):

$$C^V_{e(n)} = k \cdot (C^a_{e(n)} + C^b_{e(n)}) + C^c_{e(n)} + C^d_{e(n)}, \tag{31}$$

где $C^a_{e(n)}$ – усредненная стоимость одного метра прямой и обратной линий трубопровода, включая запорную арматуру и т.д.; $C^b_{e(n)}$ – средняя стоимость прокладки 1 погонного метра теплотрассы в отдельно взятой местности; k^* – длина линии теплотрассы до потребителя (м); $C^c_{e(n)}$ – совокупные затраты на обвод инфраструктурных и коммуникационных элементов, восстановление дорожного покрытия и т.д.; $C^d_{e(n)}$ – совокупные затраты на модернизацию источников тепла.

Для решения данной задачи использован алгоритм перебора для сужения множества допустимых решений для каждого потенциального потребителя, с целью создания оптимального сценария подключения. Для решения подобных задач широко используется метод ветвей и границ [6], поэтому далее будет использован именно этот метод. Отдельно решается подзадача трассировки от источников тепла к потребителям [7].

Диапазоны значений решения по верхней границе (32) и эвристического решения (33) вычисляются следующим образом:

$$f(U) = (U - Z)/Z \cdot 100, \tag{32}$$

$$f(H) = (H - Z)/Z \cdot 100, \tag{33}$$

где U – экстремальное значение по верхней границе диапазона допустимых решений, H – экстремальное значение эвристического решения, Z – значение оптимального решения задачи.

3. Результаты проверки математической модели

Для анализа представленной математической модели подключения оптимального числа потребителей на различных тепловых сетях сгенерируем набор из 100 тепловых сетей различных характеристик и конфигураций. Будем использовать 5 классов тепловых сетей, характеризующихся различной величиной, в частности количеством ассоциированных с существующими потребителями узлов V_E. Рассмотрим тепловые сети с показателем $|V_E| \in (100, 200, 300, 400, 500)$. Для каждого класса тепловой сети рассмотрим 4 различных варианта количественного состава потенциальных потребителей $|V_P|$, определенных пропорционально $|V_E|$ и описанных в виде: $|V_P| = |V_E|/2$; $|V_P| = |V_E|$; $|V_P| = |V_E| + |V_E|/2$; $|V_P| = 2 * |V_E|$. Для каждой пары $|V_P|$ и $|V_E|$ рассмотрим 5 различных состояний функционирования, которым соответствует некоторый набор параметров функционирования.

Для $|V_E|$ присваиваем коэффициент, соответствующий окружности с диаметром D (км) тепловой сети, обеспечивающей потребности группы существующих потребителей, относящихся к $|V_E|$. Будем считать, что D = 5, если $|V_E| \leq 200$, иначе D = 10. Каждому узлу в пределах окружности с диаметром D присваивается случайная координата. Дугам, соединяющим узлы и обозначающим линии трубопровода, присваиваются веса, равные евклидовому расстоянию между узлами, замыкающими

эти дуги. Узел с наименьшим значением координаты по оси абсцисс принимается за источник тепла. Так можно получить сгенерированное дерево, обозначающее тепловую сеть с одним источником тепла, а узлы – ветви этого дерева будут обозначать существующих потребителей тепла.

Далее генерируются p-узлы, обозначающие группы потенциальных потребителей, располагающиеся рядом с дугами существующей тепловой сети. p-узлы соединяются с существующей тепловой сетью через потенциальные p-дуги в точке ближайшей к существующей тепловой сети. Если ближайшей к p-узлу является p-дуга, то p-узел подключается к потенциальной p-дуге, образуя форму ветви дерева. Каждый существующий и потенциальный потребитель ассоциирован со случайно сгенерированным объемом теплопотребления PI_e, равным по среднему значению 75 кВт за месяц. Отметим, что такое значение составляет примерно 60 % от типичного теплопотребления. Требуемый расход теплоносителя вдоль дуг существующих потребителей вычисляется при помощи отношений:

$$\mu_e = PI_e/(\Delta T \cdot c_p), \tag{34}$$

где $\Delta T = 27$ К, $c_p = 4,18$.

Расход теплоносителя на p-дугах, ассоциированных с потенциальными потребителями, эквивалентен требуемому расходу теплоносителя в концевых p-узлах, ассоциированных с потенциальными потребителями. Диаметр труб (см), требуемый для подключения существующих и потенциальных потребителей, округляется до следующих существующих стандартных значений диаметра трубопровода (25, 32, 40, 50, 65, 80, 100, 125, 150, 200, 250, 300, 350, 400, 500, 600). Когда диаметры и длины трубопровода известны, можно посчитать их стоимость и определить коэффициенты K_1, K_2, используемые в уравнении расчета потерь давления вдоль рассматриваемого трубопровода.

Результаты моделирования представлены в таблицах [1–2], описаны сценарии с различными значениями V_P и V_E, где
- пара $|V_P|/|V_E|$ – отношение числа p-узлов, ассоциированных с потенциальными потребителями, к числу всех существующих узлов тепловой сети;
- $|V_s|$ – среднее число существующих s-узлов, ассоциированных с существующими потребителями тепла;
- con. V_p – число (шт.) подключенных потребителей тепла;
- $V_{s(pow)}$ – среднее значение требуемой теплопроизводительности для обеспечения существующих потребителей (кВт);
- $V_{p(pow)}$ – среднее значение требуемой теплопроизводительности для обеспечения потенциальных потребителей (кВт);
- $con.V_{p(\%)}$ – среднее значение подключенных потенциальных потребителей тепла;
- $con.V_{p(pow)}$ – среднее значение общей теплопроизводительности (кВт) для обеспечения подключенных потенциальных потребителей тепла;
- $f(U)$ и $f(H)$ – диапазоны значений решений по верхней границе и эвристических решений соответственно;
- $B\&C$ – среднее значение – количество $B\&C$ узлов.

Предложены различные сценарии, рассчитанные для разных объемов вырабаты-

ваемого тепла и максимального числа потенциальных узлов, которые могут быть подключены к существующей системе теплоснабжения.

Компьютерное тестирование проведено при помощи программного продукта IBM Ilog Cplex для решения моделей частично-целочисленного линейного программирования методом ветвей и границ [6].

Таблица 1. Результаты выполнения первого сценария
Table 1. The results of the first scenario

$\|V_P\|/\|V_E\|$	$\|V_s\|$	$V_{s(pow)}$, кВт	$V_{p(pow)}$, кВт	con. V_p	con. $V_{p(\%)}$	con. $V_{p(pow)}$, кВт	con. $V_{p(pow)}$ (%)	$f(U)$ (%)	$f(H)$ (%)	B & C узлы
50/100	22,00	1561	3436	35	69,20	3041	88,52	0,24	0,00	0,0
100/100	21,00	1621	7553	66	66,00	6243	82,65	0,73	-0,31	19,8
150/100	23,60	1750	11671	104	69,20	10077	86,34	0,10	-0,06	4,2
200/100	22,20	1578	15918	136	68,10	13487	84,72	0,16	-0,25	17,6
100/200	44,40	3381	7513	71	71,40	6740	89,71	0,19	0,00	0,0
200/200	43,00	3689	14926	144	71,90	13380	89,65	0,16	-0,04	6,2
300/200	43,20	3281	22941	204	68,07	19816	86,38	0,17	-0,03	28,2
400/200	45,80	3149	24995	236	58,90	20108	80,45	0,14	-0,43	69,0
150/300	65,20	4828	11011	113	75,07	10107	91,79	0,02	0,00	0,0
300/300	65,20	4994	22622	215	71,73	20140	89,03	0,22	-0,21	23,2
450/300	66,00	5099	33736	331	73,47	30508	90,43	0,07	-0,10	28,6
600/300	64,00	3707	35707	356	59,40	28817	80,70	0,18	-0,19	81,6
200/400	98,20	6985	14854	149	74,40	13504	90,91	0,09	-0,07	3,6
400/400	86,40	6188	29583	293	73,30	26848	90,76	0,18	-0,06	13,6
600/400	91,40	6855	45611	444	74,03	41122	90,16	0,09	-0,21	58,6
800/400	87,00	5323	47398	512	63,95	40040	84,48	0,10	-0,18	375,8
250/500	109,60	7885	18841	187	74,88	17265	91,64	0,08	-0,01	0,4
500/500	107,00	8284	38324	376	75,24	35023	91,39	0,08	-0,07	39,2
750/500	108,80	8562	56134	551	73,44	50649	90,23	0,04	-0,08	35,0
1000/500	109,60	6582	60132	629	62,92	50658	84,24	0,06	-0,10	327,6

Первый сценарий представлен исходя из логики максимальной теплопроизводительности системы (Таблица 1) для обслуживания всех потенциальных потребителей без ограничений на максимальное число подключаемых участков трубопровода. Также из результатов первого комплекса тестов наблюдаются малые интервалы допустимых решений по эвристической шкале и шкале верхней границы значений.

Для повышения соответствия представленной математической модели реальным тепловым сетям, приводится второй комплекс тестов, который включает в себя несколько дополнительных ограничений:
– лимит теплопроизводительности оборудования снижен до 85% от максимального;
– сумма затрат C_e на подключение, реконструкцию, модернизацию, обслуживание сетей, в частности линии $e = (i, j) \in A_{FP}$, для подключения потенциальных потребителей составляет $\Delta C_e^{max} \leq 1,075 \sum_{e \in A_{FP}} C_e C_f$.

Таблица 2. Результаты выполнения второго сценария
Table 2. The results of the second scenario

$\|V_P\|/\|V_E\|$	$\|V_s\|$	$V_{s(pow)}$, кВт	$V_{p(pow)}$, кВт	con. V_p	con. $V_{p(\%)}$	con. $V_{p(pow)}$, кВт	con. $V_{p(pow)}$ (%)	$f(U)$ (%)	$f(H)$ (%)	B & C узлы
250/500	109,60	7885	18841	56	22,40	8528	45,26	0,04	-0,15	810,0
500/500	107,00	8284	38324	47	9,40	8965	23,39	0,04	-0,47	1501,0
750/500	108,80	8562	56134	46	6,08	9271	16,52	0,02	-0,37	1664,0
1000/500	109,60	6585	60132	38	3,80	7090	11,79	0,05	-0,39	2453,0

Отметим, что величина V_E может быть равной 0, то есть тепловая сеть еще не имеет подключенных потребителей тепла и проектируется с нуля. Для каждого рассматриваемого сценария определено такое оптимальное множество подключаемых потребителей, чтобы любое изменение при подключении потребителей неизбежно вызывало увеличение минимизируемого критерия себестоимости или уменьшение максимизируемого критерия выручки от реализации тепла.

На рис. 2 проиллюстрирован пример сравнения результатов расчета распределенной системы теплоснабжения города с населением около 32 тысяч человек, рассчитанной в результате вывода из эксплуатации централизованной системы теплоснабжения. Рассматриваемая система состоит из 5 газовых котельных общей мощностью 89,8 мВт. При расчете использовались одинаковые исходные данные о потребителях, котельных и расположении. Потребителям первой категории присваивается статус V_e для отдельных подсистем. Остальным потребителям присваивается V_p.

Таблица 3. Сравнение результатов инженерного расчета (расчет) и результатов расчета с помощью представленной математической модели (ММ)
Table 3. Comparison of engineering calculation results and calculation results with the help of the presented mathematical model

№	Мощность, мВт	V_e	V_p	Con V_p, расчет	Con V_p, ММ	Con $V_{p(pow)}$, расчет, мВт	Con $V_{p(pow)}$, ММ, мВт	Corr, %
К №1	19,8	0	250	202	196	17,2	16,82	-2,2
К №2	26	11	350	305	331	22,78	24,1	+5,5
К №3	5	11	12	12	12	2,66	2,66	0
К №4	8	0	100	77	79	6,59	6,79	+3
К №5	5	0	50	44	46	3,76	4,16	+0,2
К №6	26	0	300	262	237	22,32	18,12	-18,8

Расчет выполнен без учета функции стоимости подключения потенциального потребителя из-за отсутствия способов быстрой оценки стоимости подключения. Con $V_{p(pow)}$ – максимальная нагрузка на систему теплоснабжения (отопление и горячее водоснабжение), Corr,% – расхождение расчетов подключаемой максимальной нагрузки, полученной с помощью ММ, с инженерными расчетами.

Рис. 2. Расположение расчетных котельных в черте города

Fig. 2. Location of settlement boiler houses in the city

Использование представленной математической модели в текущем виде позволяет проводить последовательный расчет подключения потенциальных потребителей в заданном порядке. Видно, что результаты решений по количеству подключенных потенциальных потребителей коррелируют с результатами расчетных значений в диапазоне от -18,8 до +5,5 процентов, не превышают лимиты тепловых нагрузок. Для небольшой системы, как котельная 3, видно полное совпадение результатов.

4. Заключение

В результате комплекса вычислений получены следующие результаты. Для каждого из рассматриваемых вариантов систем существует такое оптимальное множество потребителей для подключения к тепловой сети, чтобы соответствовать заданным данной математической моделью критериям для каждого потенциального потребителя. Количество потенциальных потребителей, подключенных к тепловой сети, снижено от максимально планированных значений за счет снижения максимальной нагрузки $Q^{max} \leq 85\%$ на источник теплоснабжения. Большое количество B & C узлов – точек экстремума положительно влияет на точность эвристического решения. Из результатов комплекса тестов наблюдаются малые интервалы допустимых решений по эвристической шкале и шкале верхней границы значений.

Сравнение полученного решения с существующими решениями показало высокую степень достоверности результатов. Для оптимизации подключения потребителей в подобных распределенных системах необходимы данные о стоимости подключения. Алгоритм трассировки как инструмент оценки стоимости подключения и прокладываемого пути предложен в [7].

Перспективным путем применения представленной математической модели выбора оптимального множества новых потребителей для подключения к существующей тепловой сети является интеграция с существующими ГИС-решениями с открытым исходным кодом. Создание новых инструментов, подобных программному комплексу Zulu [8] для систем теплоснабжения, позволит значительно повысить эффективность при проектировании и принятии решений, в том числе при нетипичных постановках задач проектирования систем теплоснабжения.

Список литературы / References

[1] Беляйкина И. В. и др., *Водяные тепловые сети*, Справочное пособие по проектированию, Энергоатомиздат, М., 1988, 375 с.; [Belyaykina I.V. et al., *Vodyanie teplovie seti*, Spravochnoe posobie po proektirovaniu, Energoatomizdat, M., 1988, 375 pp., (in Russian).]

[2] *РД 10 ВЭП – 2006 «Методические основы разработки схем теплоснабжения поселений и промышленных узлов РФ»*, в развитие СНиП 41-02-2003 «Тепловые сети», Объединение ВНИПИЭнергопром, М., 2006, 61 с.; [*RD 10 VJeP – 2006 «Metodicheskie osnovy razrabotki shem teplosnabzhenija poselenij i promyshlennyh uzlov RF»*, v razvitie SNiP 41-02-2003 «Teplovye seti», Obedinenie VNIPIEnergoprom, M., 2006, 61 pp., (in Russian).]

[3] *Прогноз розничных цен на электроэнергию в субъектах Российской Федерации на период до 2020 года*, Национальный исследовательский университет «Высшая школа экономики», Институт проблем ценообразования и регулирования естественных

монополий, М., 2015, 31 с.; [*Prognoz roznichnyh cen na elektroenergiyu v subjektah Rossiyskoy Federacii na period do 2020 goda*, Vysshaya shkola ekonomiki, Institut problem cenoobrazovaniya i regulirovaniya estestvennyh monopoliy, M., 2015, 31 pp., (in Russian).]

[4] *Строительные нормы и правила Российской Федерации, СНиП 41-02-2003 «Тепловые сети»*, Государственный комитет РФ по строительству и жилищно-коммунальному комплексу, М., 2003, 36 с.; [*Stroitelnye normy i pravila Rossiyskoy Federacii, SNiP 41-02-2003 «Teplovye seti»*, Gosudarstvennyy komitet RF po stroitelstvu i zhilishhno-kommunalnomu kompleksu, M., 2003, 36 pp., (in Russian).]

[5] Nemtinov V. A. et al., "Use of Geographic Information Systems for Assessment of Groundwater Quality in Industrial Hubs", *14th Geoconference on Informatics, Geoinformatics and Remote Sensing*, Bulgaria, 17–26 June, 2014, **1**, 911–916.

[6] Mitchell J. E., "Branch-and-Cut Algorithms for Combinatorial Optimization Problems", *Handbook of Applied Optimization*, Oxford University Press, Oxford, 2002, 65–77.

[7] Терехов С. М. и др., "Особенности решения задач оптимизации с помощью алгоритма поиска кратчайшего пути", *Виртуальное моделирование, прототипирование и промышленный дизайн*, Международ. науч. конф., Тамбов, 2017; [Terehov S.M. et al., "Osobennosti reshenija zadach optimizacii s pomoshhju algoritma poiska kratchajshego puti", *Virtualnoe modelirovanie, prototipirovanie i promyshlennyy dizayn*, Mezhdunarod. nauch. konf., Tambov, 2017, (in Russian).]

[8] *ZuluThermo*, ГеоИнфоГрад, Научно-внедренческий центр МФТИ, http://www.geoinfograd.ru/zulu.htm#thermo.

Terekhov S. M., Nemtinov V. A., Kornilov K. S., "Model of the Connecting Optimal Number of Heat Consumers", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 217–231.

DOI: 10.18255/1818-1015-2018-2-217-231

Abstract. In the modern world, the efficient use of energy is an extremely important aspect of human activity. In particular, heat supply systems have significant economic, environmental and social importance for both heat consumers and heat supply organizations. The economic status of all participants in the heat supply process depends on the efficiency of the functioning of the heat supply systems. The reliability of the functioning of systems depends on vital processes such as the work of hospitals and industrial enterprises. With such a close network communication, reliable and efficient operation of power supply systems is critical. In this article, ways to improve the efficiency of heat supply systems are considered. A mathematical model for improved planning of heat supply systems by connecting the optimal set of new heat consumers is presented. For each single customer, when there is an alternative option for connecting this consumer to the existing heat network, it is possible to choose the only optimal solution. This becomes possible due to the restrictions and the procedure for selecting variants from a subset of binary variables corresponding to alternatives. The procedure for finding the optimal number of consumers for connection to the existing heat network is presented, which is the rationale for increasing the number of existing consumers of the heat network. The testing was carried out and the results of the mathematical model by an example of test heat networks are presented. Directions of further study of increasing the efficiency of heat supply systems and integrating the presented mathematical model with modern software complexes are determined.

Keywords: system approach, heat supply system, optimisation

On the authors:
Sergey M. Terekhov, orcid.org/0000-0002-4103-1167, graduate student,
Tambov State Technical University, 106 Sovetskaya st. Tambov, 393000, Russia, e-mail: Stpodiumz3@gmail.com

Vladimir A. Nemtinov, orcid.org/0000-0003-2917-3610, Doctor of Technical Sciences, Professor,
Tambov State Technical University, 106 Sovetskaya st. 106, Tambov, 393000, Russia, e-mail: nemtinov@mail.gaps.tstu.ru

Kirill S. Kornilov, orcid.org/0000-0003-1729-8360, graduate student,
Tambov State Technical University, 106 Sovetskaya st. 106, Tambov, 393000, Russia, e-mail: i_sirotinka@rambler.ru

Моделирование и анализ информационных систем. Т. 25, №2 (2018), с. 232–245
Modeling and Analysis of Information Systems. Vol. 25, No 2 (2018), pp. 232–245

Кодовые криптосистемы

©Веденёв К. В., Деундяк В. М., 2017

DOI: 10.18255/1818-1015-2018-2-232-245

УДК 517.9

Коды в диэдральной групповой алгебре

Веденёв К. В., Деундяк В. М.

получена 2 декабря 2017

Аннотация. В 1978 году Р. Мак-Элисом построена первая асимметричная кодовая криптосистема, основанная на применении помехоустойчивых кодов Гоппы, при этом эффективные атаки на секретный ключ этой криптосистемы до сих пор не найдены. К настоящему врмени известно достаточно много кодовых криптосистем, но их криптографическая стойкость уступает стойкости классической криптосистемы Мак-Элиса. В связи с развитием квантовых вычислений кодовые криптосистемы рассматриваются как альтернатива теоретико-числовым, поэтому актуальной представляется задача поиска перспективных классов кодов для построения новых стойких кодовых криптосистем. Для этого можно использовать некоммутативные коды, т.е. идеалы в групповых алгебрах $\mathbb{F}_q G$ над конечными некоммутативными группами G. Ранее изучалась стойкость криптосистем на кодах, индуцированных кодами на подгруппах. Важной для исследования некоммутативных кодов является теорема Веддерберна, доказывающая существование изоморфизма групповой алгебры на прямую сумму матричных алгебр, но конкретный вид слагаемых и конструкция изоморфизма этой теоремой не определены, и поэтому для каждой группы остается задача построения представления Веддерберна. Ф. Е. Б. Мартинесом получено полное представление Веддерберна для групповой алгебры $\mathbb{F}_q D_{2n}$ над диэдральной группой D_{2n} в случае, когда мощность поля и порядок группы взаимно просты. С использованием этих результатов в настоящей работе исследуются коды в групповой алгебре $\mathbb{F}_q D_{2n}$. Решена задача о структуре всех кодов и описана структура кодов, которые индуцированы кодами над циклическими подгруппами группы D_{2n}, что представляет интерес для криптографических приложений.

Ключевые слова: некоммутативные группы, групповые алгебры, некоммутативные коды, кодовые криптосистемы

Для цитирования: Веденёв К. В., Деундяк В. М., "Коды в диэдральной групповой алгебре", *Моделирование и анализ информационных систем*, **25**:2 (2018), 232–245.

Об авторах:
Веденёв Кирилл Владимирович, orcid.org/0000-0002-7893-655X, студент,
Южный Федеральный Университет,
ул. Большая Садовая, 105/42, г. Ростов-на-Дону, 344006 Россия, e-mail: vedenev@sfedu.ru

Деундяк Владимир Михайлович, orcid.org/0000-0001-8258-2419, канд. физ.-мат. наук, доцент,
Южный Федеральный Университет,
ул. Большая Садовая, 105/42, г. Ростов-на-Дону, 344006 Россия,
ФГНУ НИИ "Спецвузавтоматика",
пер. Газетный, 51, г. Ростов-на-Дону, 344002, Россия, e-mail: vl.deundyak@gmail.com.

Введение

В работе [1] Р. Мак-Элис предложил асимметричную криптосистему, основанную на применении помехоустойчивых кодов Гоппы, причем к настоящему времени эф-

фективные структурные атаки на эту криптосистему не известны. После пионерской работы Р. Мак-Элиса появилось много других кодовых криптосистем на различных кодах, в частности, на ранговых кодах, кодах Рида–Соломона, кодах Рида–Маллера, алгебро-геометрических кодах. Небольшой обзор работ по анализу стойкости кодовых криптосистем содержится в [2]. Представляется актуальной задача поиска перспективных классов кодов для построения новых стойких кодовых криптосистем. Одним из вариантов решения этой задачи является использование некоммутативных алгебраических структур и, в частности, некоммутативных кодов, являющихся идеалами в групповых алгебрах $\mathbb{F}_q G$ над конечными некоммутативными группами G. Например, в работах [2], [3], [4] рассмотрены криптосистемы на индуцированных некоммутативных групповых кодах и их тензорных произведениях, а также проведен анализ их стойкости как к структурным атакам, так и к атакам на шифрограмму.

Целью настоящей работы является исследование кодов в диэдральной групповой алгебре в случае, когда мощность группы взаимно проста, с мощностью поля Галуа. В статье решена задача о структуре кодов, описаны все подгруппы диэдральной группы, и на этой основе получена структура диэдральных кодов, индуцированных циклическими кодами.

Для полупростых групповых алгебр есть теорема Веддерберна (см. [5]), доказывающая существование изоморфизма групповой алгебры на прямую сумму некоторых матричных алгебр. Эта теорема является очень важной для исследования некоммутативных кодов. Однако конкретный вид матричных слагаемых и конструкция изоморфизма этой теоремой не определены, т.е. для каждой группы остается задача построения представления Веддерберна. Некоторые результаты о структуре диэдральной групповой алгебры содержатся в работе [6], а в работе [7] получено полное представление Веддерберна для диэдральной групповой алгебры в случае, когда мощность поля и порядок группы взаимно просты.

Структура работы. Раздел 1 содержит определения диэдральной группы и групповой алгеброй, а также необходимые вспомогательные результаты. Основные результаты статьи о виде кодов, в том числе и индуцированных, а также нижняя оценка кодовых расстояний приведены в разделе 2. В разделе 3 рассмотрен пример построения диэдральных кодов.

1. Предварительные сведения о диэдральной групповой алгебре

Диэдральной группой D_{2n}, где $n \geq 2$, называется группа симметрий правильного плоского n-угольника с центром в точке O, состоящая из поворотов вокруг точки O на углы, кратные $\frac{2\pi}{n}$, и отражений относительно прямых, проходящих через O и одну из вершин или середину одной из сторон. Группа D_{2n} порождается поворотом a на угол $\frac{2\pi}{n}$ и произвольным отражением b, при этом выполняются следующие соотношения:

$$a^n = e,\ b^2 = e,\ bab = a^{-1} \tag{1}$$

(см. [8] с. 171). Таким образом, D_{2n} допускает копредставление $\langle a, b :\ a^n,\ b^2,\ (ba)^2\rangle$, при этом формально можно считать, что n – произвольное натуральное число. Из

(1) вытекает, что для произвольного i

$$a^i b = ba^{-i}, \tag{2}$$

поэтому

$$D_{2n} = \{e, a, a^2, ..., a^{n-1}, b, ab, a^2b, ..., a^{n-1}b\}. \tag{3}$$

Следующая теорема о структуре подгрупп диэдральной группы, возможно, известна специалистам, однако, в доступных источниках её найти не удалось. Через $d(n)$ обозначим количество делителей натурального числа n.

Теорема 1. *В группе D_{2n} ($n \geq 2$) имеются следующие собственные подгруппы:*

а) n различных подгрупп вида $\{e, a^k b\}$, где $0 \leq k < n$, каждая из которых изоморфна $\mathbb{Z}_2$;

б) $d(n)$ подгрупп вида $\langle a^k \rangle$, где k – любой натуральный делитель n, каждая из которых изоморфна $\mathbb{Z}_{\frac{n}{k}}$;

в) для каждого собственного делителя k числа n существует $k-1$ различных подгрупп вида $\langle a^l b, a^k \rangle$, где $0 \leq l < k$, каждая из которых изоморфна $D_{2\frac{n}{k}}$.

Других собственных подгрупп нет. Подгруппы вида б) нормальны, подгруппы вида а) нормальны только при $n = 2$, а подгруппы вида в) нормальны только при $k = 2$.

Пусть G – конечная группа и $\mathbb{F}_q$ – поле Галуа мощности q. Групповой алгеброй $\mathbb{F}_q G$ называется множество формальных линейных комбинаций вида

$$\alpha = \sum_{g \in G} a_g g, \quad a_g \in \mathbb{F}_q$$

с покомпонентно определёнными операциям сложения и умножения на скаляр и операцией умножения по следующему правилу:

$$(\sum_{g \in G} a_g g)(\sum_{g \in G} b_g g) = \sum_{g \in G} (\sum_{h \in G} a_h b_{h^{-1}g}) g \; (= \sum_{g,h \in G} a_g b_h gh)$$

(см. [5], с.139). Определено естественное вложение группы G в групповую алгебру $\mathbb{F}_q G$, переводящее элемент группы $g \in G$ в элемент групповой алгебры $1g \in \mathbb{F}_q G$. Аналогично определено вложение поля $\mathbb{F}_q$ в $\mathbb{F}_q G$, переводящее $\lambda \in \mathbb{F}_q$ в $\lambda e \in \mathbb{F}_q G$, где $e \in G$ – нейтральный элемент группы. В $\mathbb{F}_q G$ имеется естественная инволюция, индуцированная инверсией в группе G:

$$(\sum_{g \in G} a_g g)^* = \sum_{g \in G} a_g g^{-1},$$

которая устанавливает взаимно однозначное соответствие между левыми и правыми идеалами, поэтому в этой статье мы будем рассматривать только левые идеалы. Всякий левый идеал $I \subset \mathbb{F}_q G$ называется групповым G-кодом над полем $\mathbb{F}_q$ (см. [9]). Если идеал I – двусторонний, то будем называть его центральным кодом.

Рассмотрим групповую алгебру $\mathbb{F}_q D_{2n}$. В силу (2), (3) любой элемент $u \in \mathbb{F}_q D_{2n}$ может быть представлен следующим образом:

$$u = P(a) + bQ(a) = P(a) + Q(a^{-1})b, \tag{4}$$

где P и Q – многочлены степени, не превосходящей n.

В работе [7] доказана теорема о виде разложения Веддерберна для $\mathbb{F}_q D_{2n}$, однако прежде чем сформулировать её в удобном для дальнейшего виде, приведём необходимые вспомогательные сведения. Для каждого многочлена $g \in \mathbb{F}_q[x]$ такого, что $g(0) \neq 0$, возвратным многочленом называется многочлен $g^*(x) := x^{\deg(g)} g(\frac{1}{x})$. Говорят, что многочлен g самовозвратный, если g и g^* имеют одни и те же корни в своём поле разложения, т. е. эти многочлены отличаются на постоянный ненулевой множитель. Ниже будем полагать, что наибольший общий делитель $\gcd(2n, q)$ чисел $2n$ и q равен единице.

Известно, что многочлен $x^n - 1 \in \mathbb{F}_q[x]$ разлагается на неприводимые над $\mathbb{F}_q$ множители; следуя [7], запишем это разложение следующим образом:

$$x^n - 1 = (f_1 f_2 \dots f_r)(f_{r+1} f^*_{r+1} f_{r+2} f^*_{r+2} \dots f_{r+s} f^*_{r+s}), \tag{5}$$

где $f_1 = x - 1$, при $1 < j \leq r$ выполнено равенство $f^*_j = f_j$ и $f_2 = x + 1$ в случае чётного n. Здесь r — количество самовозвратных множителей в этом разложении, а $2s$ — несамовозвратных.

Всякий неприводимый над полем $\mathbb{F}_q$ многочлен h степени m имеет корень в расширении этого поля F_{q^m}, обозначим его через α, при этом элементы

$$\alpha^0 = 1, \alpha^1, \alpha^2, \dots, \alpha^{m-1}$$

составляют базис расширения F_{q^m} как векторного пространства над $\mathbb{F}_q$ (см. [8], с. 409). Поэтому

$$\mathbb{F}_q[\alpha] := \{\sum_{j=0}^{k} c_j \alpha^j \in F_{q^m} \mid k \in \mathbb{N} \cup \{0\},\ c_j \in \mathbb{F}_q\}$$

совпадает с полем F_{q^m}. Если $\deg(h) = 1$, то $\alpha \in \mathbb{F}_q$ и, следовательно, $F[\alpha] = \mathbb{F}_q$. Аналогично, элементы

$$1, \beta, \beta^2, \dots \beta^{n-1},$$

где $\beta = \alpha^{-1}$ — корень многочлена h^*, тоже образуют базис в F_{q^n}, поэтому

$$F_{q^m} = F[\alpha] = F[\alpha^{-1}].$$

Обозначим далее α_j — корень многочлена f_j из (5),

$$\delta(n) := \begin{cases} 1, & n \text{ — нечётное}, \\ 2, & n \text{ — чётное}. \end{cases}$$

Рассмотрим гомоморфизмы τ_j алгебры $\mathbb{F}_q D_{2n}$, определяемые своими значениями на порождающих элементах a и b:

a) $\tau_1 : \mathbb{F}_q D_{2n} \to \mathbb{F}_q \oplus \mathbb{F}_q, \quad \tau_1(a) = (1, 1),\ \tau_1(b) = (1, -1)$;

b) $\tau_2 : \mathbb{F}_q D_{2n} \to \mathbb{F}_q \oplus \mathbb{F}_q, \quad \tau_2(a) = (-1, -1),\ \tau_2(b) = (1, -1), \quad \delta(n) = 2$;

c) $\tau_j : \mathbb{F}_q D_{2n} \to \mathrm{M}_2(\mathbb{F}_q[\alpha_j]),\ \tau_j(a) = \begin{pmatrix} \alpha_j & 0 \\ 0 & \alpha_j^{-1} \end{pmatrix}, \tau_j(b) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad j \geq \delta(n) + 1.$

Для $j = \delta(n)+1, \ldots, r$ рассмотрим автоморфизмы σ_j алгебр (2×2)-матриц $\mathrm{M}_2(\mathbb{F}_q[\alpha_j])$, определяемые формулой

$$\sigma_j(X) = Z_j^{-1} X Z_j, \quad Z_j := \begin{pmatrix} 1 & -\alpha_j \\ 1 & -\alpha_j^{-1} \end{pmatrix}. \tag{6}$$

Отметим, что $\mathbb{F}_q[\alpha_j + \alpha_j^{-1}] \subset \mathbb{F}_q[\alpha_j]$ и при $\delta(n) + 1 \le j \le r$

$$\sigma_j(\mathrm{im}(\tau_j)) = \mathrm{M}_2(\mathbb{F}_q[\alpha_j + \alpha_j^{-1}]) \simeq \mathrm{M}_2(F_{q^{\deg(f_j)/2}}).$$

Замечание 1. В [7], с. 209, отмечено, что если $u := P(a) + Q(a)b \in \mathbb{F}_q D_{2n}$, $v := P(a) + bQ(a) \in \mathbb{F}_q D_{2n}$, то

$$\tau_j(u) = \begin{pmatrix} P(\alpha_j) & Q(\alpha_j) \\ Q(\alpha_j^{-1}) & P(\alpha_j^{-1}) \end{pmatrix}, \quad j > \delta(n),$$

$$\tau_j(v) = \begin{pmatrix} P(\alpha_j) & Q(\alpha_j^{-1}) \\ Q(\alpha_j) & P(\alpha_j^{-1}) \end{pmatrix}, \quad j > \delta(n),$$

$$\tau_1(u) = \tau_1(v) = (P(1) + Q(1), P(1) - Q(1)),$$

$$\tau_2(u) = \tau_2(v) = (P(-1) + Q(-1), P(-1) - Q(-1)), \quad \delta(n) = 2.$$

Теорема 2. *[7] Пусть* $\gcd(q, 2n) = 1$, *тогда имеет место изоморфизм:*

$$p : \mathbb{F}_q D_{2n} \to \bigoplus_{j=1}^{r+s} A_j, \tag{7}$$

где

$$A_j = \begin{cases} \mathbb{F}_q \oplus \mathbb{F}_q, & j \le \delta(n) \\ \mathrm{M}_2(\mathbb{F}_q[\alpha_j + \alpha_j^{-1}]), & \delta(n) + 1 \le j \le r \\ \mathrm{M}_2(\mathbb{F}_q[\alpha_j]), & r + 1 \le j \le r + s \end{cases},$$

$$p := \bigoplus_{j=1}^{r+s} p_j, \quad p_j := \begin{cases} \sigma_j \circ \tau_j, & \delta(n) + 1 \le j \le r \\ \tau_j, & 1 \le j \le \delta(n),\ r + 1 \le j \le r + s \end{cases},$$

$$\mathbb{F}_q[\alpha_j + \alpha_j^{-1}] \simeq F_{q^{\deg(f_j)/2}}, \quad \delta(n) + 1 \le j \le r,$$

$$\mathbb{F}_q[\alpha_j] \simeq F_{q^{\deg(f_j)}}, \quad r + 1 \le j \le r + s.$$

Замечание 2. В случае, когда в разложении (5) все множители линейные, единственными самовозвратным многочленами первой степени являются $x + 1$ и $x - 1$, тогда $r = \delta(n)$ и поэтому

$$A_j = \begin{cases} \mathbb{F}_q \oplus \mathbb{F}_q, & j \le \delta(n) \\ \mathrm{M}_2(\mathbb{F}_q[\alpha_j]), & \delta(n) + 1 \le j \le \delta(n) + s \end{cases}.$$

2. Структура кодов в алгебре $\mathbb{F}_q D_{2n}$

По теореме 2 групповая алгебра $\mathbb{F}_q D_{2n}$ при $\gcd(2n, q) = 1$ изоморфна алгебре

$$\Delta := \bigoplus_{j=1}^{r+s} A_j$$

(см. (7)), поэтому изучение кодов в $\mathbb{F}_q D_{2n}$ сводится к изучению левых идеалов в Δ. Пусть F — произвольное поле Галуа, будем далее называть ненулевой вектор $(x, y) \in F^2$ нормированным, если $x = 1$ или $x = 0,\ y = 1$.

Для алгебры A_1 из разложения (7) и нормированного вектора (x, y) положим

$$I_1(x, y) := \{\lambda x, \mu y \mid \lambda, \mu \in \mathbb{F}_q\}. \tag{8}$$

Отметим, что если $x = 1$ и $y \neq 0$, то $I_1(x, y) = A_1$. Для других нормированных векторов имеем

$$I_1(1, 0) = \mathbb{F}_q \oplus 0,\ I_1(0, 1) = 0 \oplus \mathbb{F}_q.$$

Легко видеть, что других собственных идеалов в алгебре A_1 нет. Аналогично при чётном n определяются идеалы $I_2(x, y)$ в A_2.

Далее будем использовать следующий результат Н. Джекобсона: пусть V – конечномерное векторное пространство над полем F, тогда всякий левый идеал в алгебре линейных эндоморфизмов $\mathcal{L}(V)$ имеет вид

$$I(K) = \{\theta \in \mathcal{L}(V) \mid K \subset \ker(\theta)\},$$

где K — некоторое подпространство V (см. [11], с. 93).

Рассмотрим разложение (7) и обозначим

$$R_j := \begin{cases} \mathbb{F}_q, & 1 \leq j \leq \delta(n), \\ \mathbb{F}_q[\alpha_j + \alpha_j^{-1}], & \delta(n) + 1 \leq j \leq r, \\ \mathbb{F}_q[\alpha_j], & r + 1 \leq j \leq r + s. \end{cases} \tag{9}$$

При $\delta(n) + 1 \leq j \leq r + s$ отождествим алгебру A_j с алгеброй $\mathcal{L}(R_j^2)$. Всякое собственное подпространство K_j пространства R_j^2 имеет размерность 1, т.е. является линейной оболочкой нормированного вектора (x, y). Через $I_j(x, y)$ обозначим идеал $I(K_j)$ в соответствующей алгебре A_j.

Лемма 1. *Если $\delta(n) + 1 \leq j \leq r + s$, то*

$$I_j(x, y) = \{\begin{pmatrix} ky & -kx \\ ty & -tx \end{pmatrix} = k \begin{pmatrix} y & -x \\ 0 & 0 \end{pmatrix} + t \begin{pmatrix} 0 & 0 \\ y & -x \end{pmatrix} \mid k, t \in R_j\}.$$

Доказательство. Действительно, если

$$L = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in I_j(x, y),$$

то

$$L \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} Ax + By \\ Cx + Dy \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Тогда найдутся такие $k, t \in R_j$, что

$$A = ky, B = -kx, C = ty, D = -tx.$$

С другой стороны, очевидно, что для любых $k, t \in R_j$ при $A = -ky, B = kx, C = -ty, D = tx$ получаем $L \in I_j(x, y)$. □

Теорема 3. *Пусть* $\gcd(q, 2n) = 1$. *Рассмотрим разложение (7) групповой алгебры* F_pD_{2n}. *Для любого кода* $I \subset \mathbb{F}_qD_{2n}$ *найдутся такие непересекающиеся множества* $J_1, J_2 \subset \{1, \ldots, r+s\}$ *и набор нормированных векторов* $\{(x_j, y_j)\}_{j\in J_2}$, *где* $x_j, y_j \in R_j$, *что*

$$p(I) = \bigoplus_{j=1}^{r+s} B_j, \tag{10}$$

$$B_j := \begin{cases} A_j, & j \in J_1 \\ I_j(x_j, y_j) & j \in J_2 \\ 0, & j \notin J_1 \cup J_2 \end{cases}. \tag{11}$$

С другой стороны, для любых $J_1, J_2 \subset \{1, \ldots, r+s\}$ *таких, что* $J_1 \cap J_2 = \varnothing$, *и для любого набора векторов* $\{(x_j, y_j)\}_{j\in J_2}$ *над соответствующими полями* R_j *(см. (9)) множество*

$$p^{-1}(\bigoplus_{j=1}^{r+s} B_j),$$

где B_j *определены равенством (11), является кодом в* $\mathbb{F}_qD_{2n}$.

Доказательство. В силу теоремы 2 при $\gcd(q, 2n) = 1$ алгебры $\mathbb{F}_qD_{2n}$ и Δ изоморфны, поэтому изоморфизм p устанавливает взаимно однозначное соответствие между кодами в $\mathbb{F}_qD_{2n}$ и левыми идеалами в Δ. Рассмотрим идеалы в алгебре Δ. Известно, что всякий левый идеал в прямой сумме алгебр есть прямая сумма левых идеалов в слагаемых. Лемма 1 устанавливает вид собственных левых идеалов в матричных слагаемых, а собственные идеалы в $\mathbb{F}_q \oplus \mathbb{F}_q$ указаны ранее (см. (8)). Отсюда вытекают оба утверждения теоремы. □

Замечание 3. Для построения кодов в $\mathbb{F}_qD_{2n}$ необходимой является конструкция обратного к p изоморфизма p^{-1}. Чтобы получить её, рассмотрим несколько шагов. Ниже будем пользоваться представлением (4) элементов групповой алгебры $\mathbb{F}_qD_{2n}$ и конструкциями, использовавшимися для построения изоморфизма p, в частности (5), (6), (7). Введём также обозначения: id_X — тождественное отображение на множестве X и $\xi := (1+1)^{-1} \in \mathbb{F}_q$.

1. Определим мономорфизм $\epsilon_1 : \mathbb{F}_q \oplus \mathbb{F}_q \to \mathbb{F}_qD_{2n}$ по формуле

$$\epsilon_1(w) := \xi[(w_1 + w_2) + (w_1 - w_2)b]M_1(a)M_1',$$

$$M_1(x) := \frac{x^n - 1}{x - 1} \in \mathbb{F}_q[x],\ M_1' := (M_1(1))^{-1}.$$

2. При чётном n аналогично определим мономорфизм $\epsilon_2 : \mathbb{F}_q \oplus \mathbb{F}_q \to \mathbb{F}_qD_{2n}$ по формуле

$$\epsilon_2(w) := \xi[(w_1 + w_2) + (w_1 - w_2)b]M_2(a)M_2',$$

$$M_2(x) := \frac{x^n - 1}{x + 1},\ M_1' := (M_2(-1))^{-1}.$$

3. Для $r + 1 \le j \le r + s$ определим мономорфизмы $\epsilon_j : \mathrm{M}_2(\mathbb{F}_q[\alpha_j]) \to \mathbb{F}_q D_{2n}$ по формуле

$$\epsilon_j(\begin{pmatrix} A(\alpha_j) & C(\alpha_j^{-1}) \\ B(\alpha_j) & D(\alpha_j^{-1}) \end{pmatrix}) = [A(a) + bB(a)]M_j(a)M_j'(a) + [D(a) + bC(a)]N_j(a)N_j'(a),$$

$$M_j(x) := \frac{x^n - 1}{f_j(x)} \in \mathbb{F}_q[x], \quad M_j(\alpha_j)M_j'(\alpha_j) = 1,$$

$$N_j(x) := \frac{x^n - 1}{f_j^*(x)} \in \mathbb{F}_q[x], \quad N_j(\alpha_j^{-1})N_j'(\alpha_j^{-1}) = 1.$$

Из замечания 1 видно, что $\tau_j\epsilon_j = \mathrm{id}_{A_j}$ и для любого $i \ne j$ выполняется $\tau_i\epsilon_j = 0$.

4. Для $\delta(n) + 1 \le j \le r$ формулой $\gamma_j(X) = Z_j X Z_j^{-1}$ определим отображение

$$\gamma_j : M_2(\mathbb{F}_q[\alpha_j + \alpha_j^{-1}]) \to M_2(\mathbb{F}_q[\alpha_j]),$$

а формулой

$$\psi_j(\begin{pmatrix} P(\alpha_j) & Q(\alpha_j) \\ Q(\alpha_j^{-1}) & P(\alpha_j^{-1}) \end{pmatrix}) = M_j(a)M_j'(a)P(a) + M_j(a)M_j'(a)Q(a)b,$$

где

$$M_j(x) := \frac{x^n - 1}{f_j(x)} \in \mathbb{F}_q[x],\ M_j(\alpha_j)M_j'(\alpha_j) = 1,$$

отображение $\psi_j : \mathrm{im}(\gamma_j) \to \mathbb{F}_q D_{2n}$. Видно, что $\sigma_j\tau_i\psi_j\gamma_j = \mathrm{id}_{\mathrm{A_j}}$ и для любого $i \ne j$ выполняется $\tau_i \circ \psi_j = 0$.

Пусть

$$q := \sum_{j=1}^{\delta(n)} \epsilon_j + \sum_{j=\delta(n)+1}^{r} \psi_j\gamma_j + \sum_{j=r+1}^{r+s} \epsilon_j. \tag{12}$$

Тогда

$$pq = p(\sum_{j=1}^{\delta(n)} \epsilon_j + \sum_{j=\delta(n)+1}^{r} \psi_j\gamma_j + \sum_{j=r+1}^{r+s} \epsilon_j) = \bigoplus_{j=1}^{\delta(n)} \tau_j\epsilon_j \oplus \bigoplus_{j=\delta(n)+1}^{r} \sigma_j\tau_j\psi_j\gamma_j \oplus \bigoplus_{j=r+1}^{r+s} \tau_j\epsilon_j =$$

$$= \bigoplus_{j=1}^{r+s} \mathrm{id}_{A_j} = \mathrm{id}_\Delta.$$

Следовательно, $p^{-1} = q$. □

В [10] рассмотрен способ переноса код из групповой алгебры над подгруппой в групповую алгебру над всей группой. Пусть G – конечная группа, H – её подгруппа, $\mathcal{T}$ – правая трансверсаль G по H, а I – код в $\mathbb{F}_q H$. Индуцированным кодом называется код

$$J = (\mathbb{F}_q G)I,$$

при этом если $B(I) = \{n_1, n_2, ...n_k\}$ – $\mathbb{F}_q$-базис I, то $\mathbb{F}_q$-базисом J будет

$$B(J) := \mathcal{T}B(I).$$

Если $[n, k, d]$ — параметры кода I над подгруппой, то $[n|\mathcal{T}|, k|\mathcal{T}|, d]$ — параметры индуцированного кода J (см. [10]).

Рассмотрим циклическую подгруппу $\langle a\rangle$ диэдральной группы D_{2n}; все идеалы в алгебре $\mathbb{F}_q\langle a\rangle$ — циклические коды (см. [12]). Ниже будем рассматривать коды в алгебре $\mathbb{F}_q D_n$, индуцированные циклическими кодами, что может иметь применение в задаче усиления кодовых криптосистем (см. [2], [3]).

Известно, что всякий код C в $\mathbb{F}_q\langle a\rangle$ порождается многочленом $g(x)$, делителем $x^n - 1$ (см. [12]), при этом его $\mathbb{F}_q$-базисом является набор

$$\{g(a), ag(a), \ldots, a^{n-\deg(g)-1}g(a)\}.$$

Рассмотрим индуцированный код $T = (\mathbb{F}_q D_{2n})C$. Правая трансверсаль $\mathcal{T}$ группы D_{2n} по подруппе $\langle a\rangle$ имеет вид

$$\mathcal{T} = \{e, b\},$$

поэтому базисом индуцированного кода T является набор

$$\{g(a), ag(a), \ldots, a^{n-\deg(g)-1}g(a), bg(a), bag(a), \ldots, ba^{n-\deg(g)-1}g(a)\}. \tag{13}$$

Теорема 4. *Пусть* $\gcd(q, 2n) = 1$. *Рассмотрим разложение (7) групповой алгебры* $\mathbb{F}_q D_{2n}$. *Пусть* C_g *— циклический код в* $\mathbb{F}_q\langle a\rangle$, *порождённый многочленом* $g(x)$. *Тогда для индуцированного этим циклическим кодом кода* $T_g = (\mathbb{F}_q D_{2n})C_g$ *разложение Веддерберна имеет следующий вид:*

$$p(T_g) = (\bigoplus_{j=1}^{r+s} B_j)$$

где

$$B_j = \begin{cases} A_j, & j \in J_1 \\ I_j(0,1) & j \in J_2 \\ I_j(1,0) & j \in J_3 \\ 0, & j \notin J_1, J_2, J_3 \end{cases}, \tag{14}$$

$$J_1 := \{j \in 1, \ldots, r+s \ : \ (f_j \not| g) \ \wedge \ (f_j^* \not| g)\},$$
$$J_2 := \{j \in \delta(n)+1, \ldots, r+s \ : \ (f_j \not| g) \wedge (f_j^* \mid g)\},$$
$$J_3 := \{j \in \delta(n)+1, \ldots, r+s \ : \ (f_j \mid g) \wedge (f_j^* \not| g)\}.$$

Доказательство. В силу (13) любой элемент из индуцированного кода T_g представим в виде

$$u = P(a)g(a) + bQ(a)g(a).$$

В силу замечания 1 при $j \geq \delta(n) + 1$ получаем

$$\tau_j(u) = \begin{pmatrix} P(\alpha_j)g(\alpha_j) & Q(\alpha_j^{-1})g(\alpha_j^{-1}) \\ Q(\alpha_j)g(\alpha_j) & P(\alpha_j^{-1})g(\alpha_j^{-1}) \end{pmatrix}.$$

Если f_j делит многочлен g, то нетрудно видеть, что левый столбец при любых P и Q обращается в 0; справедливо и обратное: если при любых P и Q левый столбец обращается в 0, то f_j делит g. Действительно, положим $P(x) = Q(x) = 1$, тогда $g(\alpha_j) = 0$, т.е. α_j – корень многочлена g и $f_j | g$. Аналогичные рассуждения справедливы для правых столбцов и возвратных многочленов f_j^*. Таким образом, при $j \leq \delta(n) + 1$:

$$\begin{array}{rcl} (f_j \not| g) \wedge (f_j^* \not| g) & \Rightarrow & \tau_j(T_g) = A_j, \\ (f_j | g) \wedge (f_j^* \not| g) & \Rightarrow & \tau_j(T_g) = I_j(0,1), \\ (f_j \not| g) \wedge (f_j^* | g) & \Rightarrow & \tau_j(T_g) = I_j(1,0), \\ (f_j | g) \wedge (f_j^* | g) & \Rightarrow & \tau_j(T_g) = 0. \end{array}$$

Далее, в силу замечания 1

$$\tau_1(u) = (P(1)g(1) - Q(1)g(1), P(1)g(1) + Q(1)g(1)).$$

Аналогично этот вектор обращается в 0 при любых P и Q тогда и только тогда, когда $g(1) = 0$. Аналогично при чётном n:

$$\tau_2(u) = (P(-1)g(-1) - Q(-1)g(-1), P(-1)g(-1) + Q(-1)g(-1))$$

обращается в 0 при любых P и Q тогда и только тогда, когда $g(-1) = 0$. А т.к. изоморфизм p имеет вид (7), то теорема доказана. □

Введём линейное отображение $\mathrm{pr}_a : \mathbb{F}_q D_{2n} \to \mathbb{F}_q\langle a \rangle$, действующее по правилу

$$\mathrm{pr}_a(P(a) + bQ(a)) := P(a).$$

Лемма 2. *Если I — левый идеал в алгебре $\mathbb{F}_q D_{2n}$, то $\mathrm{pr}_a(I)$ — идеал в $\mathbb{F}_q\langle a \rangle$, т.е. циклический код.*

Доказательство. Действительно, пусть $P(a) \in \mathrm{pr}_a(I)$, тогда существует такое $u = P(a) + bQ(a) \in I$. Тогда для любого $S(a) \in \mathbb{F}_q\langle a \rangle$ имеем

$$(S(a) + 0b)u = S(a)P(a) + bQ(a)S(a^{-1}) \in I,$$

т.е. $S(a)P(a) = \mathrm{pr}_a((S(a) + 0b)u) \in \mathrm{pr}_a(I)$. □

Теорема 5. *Для всякого кода $I \subset \mathbb{F}_q D_{2n}$ найдутся циклические коды $C_1, C_2 \subset \mathbb{F}_q\langle a \rangle$ такие, что $(\mathbb{F}_q D_{2n})C_1 \subset I \subset (\mathbb{F}_q D_{2n})C_2$.*

Доказательство. Положим $C_2 := \mathrm{pr}_a(I)$. Покажем, что $I \subset (\mathbb{F}_q D_{2n})C_2$, пусть $u = P(a) + bQ(a) \in I$, тогда

$$\mathrm{pr}_a(u) = P(a) \in \mathrm{pr}_a(I)$$
$$\mathrm{pr}_a(bu) = Q(a) \in \mathrm{pr}_a(I),$$

откуда $P(a), Q(a), bP(a), bQ(a) \in (\mathbb{F}_q D_{2n})C_2$, следовательно, $u \in (\mathbb{F}_q D_{2n})C_2$.

Положим $C_1 := I \cap \mathbb{F}_q\langle a \rangle$, очевидно, что это действительно циклический код и что $(\mathbb{F}_q D_{2n})C_1 \subset I$.

□

Через $\mathrm{dist}(I)$ обозначим минимальное кодовое расстояние кода I.

Следствие 1. *Пусть I — код в $\mathbb{F}_q D_{2n}$, тогда* $\mathrm{dist}(I) \geq \mathrm{dist}(\mathrm{pr}_a(I))$.

Доказательство следует из того, что у индуцированных кодов наименьшее кодовое расстояние такое же, как и у кодов, которыми они индуцированы.

3. Пример

Рассмотрим пример диэдрального кода, который не входит в класс кодов, индуцированных циклическими.

Пусть $\mathbb{F}_q$ – конечное поле, n – некоторый делитель числа $q-1$, $\omega \in \mathbb{F}_q$ – элемент порядка n. Выберем целый параметр $d \le \frac{n}{2} - 1$ и рассмотрим код Рида–Соломона (см. [12]) C_g с порождающим многочленом

$$g(x) = \begin{cases} (x-\omega^{\frac{n}{2}})[(x-\omega^{\frac{n}{2}-1})(x-\omega^{\frac{n}{2}+1})]\dots[(x-\omega^{\frac{n}{2}-d})(x-\omega^{\frac{n}{2}+d})], & n \text{ — четное}, \\ [(x-\omega^{\lfloor\frac{n}{2}\rfloor})(x-\omega^{\lceil\frac{n}{2}\rceil})]\dots[(x-\omega^{\lfloor\frac{n}{2}\rfloor-d})(x-\omega^{\lceil\frac{n}{2}\rceil+d})], & n \text{ — нечетное}. \end{cases}$$

Отметим, что если многочлен f — делитель многочлена g, то его возвратный многочлен f^* тоже делит g. Длина кода C_g равна n, минимальное кодовое расстояние – $(2d+1)$ при нечётном n и $(2d+2)$ при чётном. Размерность кода C_g равна $(n-2d)$ в случае чётного n и $(n-2d-1)$ в случае нечётного. Рассмотрим индуцированный код $T_g = (\mathbb{F}_q D_{2n})C_g$. По теореме 4

$$p(T_g) = \bigoplus_{j=1}^{r+s} B_j,$$

где

$$B_j = \begin{cases} A_j, & j \in S_1 \\ 0, & j \in S_2 \end{cases}, \tag{15}$$

$$S_1 \subset \{1,\dots,r+s\}, \quad S_2 = \{1,\dots,r+s\} \setminus S_1.$$

Теперь построим новый код T', вложенный в T_g. Для этого выберем произвольное непустое множество $S_3 \subset S_1 \setminus \{1, \delta(n)\}$ и набор таких нормированных векторов $\{(x_j, y_j)\}_{j\in S_3}$, что $x_j \neq 0$ и $y_j \neq 0$. Определим левый идеал $D = \bigoplus_{j=1}^{r+s} B'_j$ в алгебре Δ, где

$$B'_j = \begin{cases} A_j, & j \in S_1 \setminus S_3 \\ I_j(x_j, y_j), & j \in S_3 \\ 0, & j \in S_2. \end{cases}. \tag{16}$$

Воспользуемся построенным в замечании 3 изоморфизмом p^{-1} и определим код $T' = p^{-1}(D)$. Заметим, что этот код не является кодом, индуцированным циклическим. Действительно, если бы T' был таким кодом, то по теореме 4 левый идеал D имел бы вид (14), но по построению это не так.

Теперь рассмотрим коды C_g, T_g и T' для конкретных числовых параметров. Пусть $q = 11$, $n = 10$, $\omega = 2$ — примитивный элемент поля $\mathbb{F}_{11}$. Выпишем для многочлена $x^{10} - 1$ разложение (5) на неприводимые множители:

$$x^{10} - 1 = f_1 f_2 [f_3 f_3^* f_4 f_4^* f_5 f_5^* f_6 f_6^*],$$

где

$$f_1(x) = x - 1,\ f_2(x) = x + 1,$$

$$f_3(x) = x - 2,\ f_4(x) = x - 3,\ f_5(x) = x - 7,\ f_6(x) = x - 9,$$

$$f_3^*(x) = x - 6,\ f_4^*(x) = x - 4,\ f_5^*(x) = x - 8,\ f_6^*(x) = x - 5.$$

В нашем случае $r = 2$ — количество самовозвратных множителей в этом разложении, а $2s = 8$ — количество несамовозвратных. Положим $d = 2$. Тогда

$$g(x) = (x - \omega^5)[(x - \omega^4)(x - \omega^6)][(x - \omega^3)(x - \omega^7)],$$

$$g(x) = (x - 10)[(x - 5)(x - 9)][(x - 8)(x - 7)],$$

$$g(x) = f_2(x)f_6(x)f_6^*(x)f_5(x)f_5^*(x).$$

Код C_g является $[10, 5, 6]$–кодом, индуцированный код $T_g = (\mathbb{F}_{11}D_{20})C_g$ является $[20, 10, 6]$–кодом. По теореме 4

$$p(T_g) = B_1 \oplus B_2 \oplus B_3 \oplus B_4 \oplus B_5 \oplus B_6$$

(см. (14), (16)), где

$$B_1 = \mathbb{F}_{11} \oplus \mathbb{F}_{11},\ B_2 = 0 \oplus 0,$$

$$B_3 = \mathrm{M}_2(\mathbb{F}_{11}[2]),\ B_4 = \mathrm{M}_2(\mathbb{F}_{11}[3]),\ B_5 = 0,\ B_6 = 0.$$

Теперь для наших параметров вычислим код T'. Пусть

$$D = B_1' \oplus B_2' \oplus B_3' \oplus B_4' \oplus B_5' \oplus B_6',$$

$$B_1' = \mathbb{F}_{11} \oplus \mathbb{F}_{11},\ B_2' = 0 \oplus 0,$$

$$B_3' = I_3(1, -1),\ B_4' = \mathrm{M}_2(\mathbb{F}_{11}[3]),\ B_5' = 0,\ B_6' = 0,$$

где

$$M_3(x) = \frac{x^{10} - 1}{x - 2}, \quad N_3(x) = \frac{x^{10} - 1}{x - 4}.$$

Лемма 1 устанавливает вид базиса в $I_3(1, -1)$, вид базисов в остальных слагаемых левого идеала D известен, теперь с помощью замечания 3 построим $\mathbb{F}_q$-базис кода $T' = p^{-1}(D)$:

$$B_1 = e + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9,$$

$$B_3 = 6e + 3a + 7a^2 + 9a^3 + 10a^4 + 5a^5 + 8a^6 + 4a^7 + 2a^8 + 1a^9,$$

$$B_5 = 2e + 4a + 8a^2 + 5a^3 + 10a^4 + 9a^5 + 7a^6 + 3a^7 + 6a^8 + 1a^9,$$

$$B_7 = 3e + 9a + 5a^2 + 4a^3 + 1a^4 + 3a^5 + 9a^6 + 5a^7 + 4a^8 + 1a^9 +$$

$$+b(4e + 5a + 9a^2 + 3a^3 + 1a^4 + 4a^5 + 5a^6 + 9a^7 + 3a^8 + 1a^9),$$

$$B_2 = bB_1,\ B_4 = bB_3,\ B_6 = bB_5,\ B_8 = bB_7.$$

В результате прямой вычислительной проверки установлено, что кодовое расстояние данного кода равно 8, таким образом получен диэдральный $[20, 8, 8]$-код.

Список литературы / References

[1] McEliece R.J., "A Public-Key Cryptosystem Based on Algebraic Coding Theory", *DSN Progress Report*, **42–44** (1978), 114–116.

[2] Деундяк В.М., Косолапов Ю.В., "Криптосистема на индуцированных групповых кодах", *Модел. и анализ информ. систем*, **23**:2 (2016), 137–152; [Deundyak V. M., Kosolapov Y. V., "Cryptosystem Based on Induced Group Codes", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 137–152, (in Russian).]

[3] Деундяк В.М., Косолапов Ю.В., Лелюк Е.А., "Декодирование тензорного произведения MLD-кодов и приложения к кодовым криптосистемам", *Модел. и анализ информ. систем*, **24**:2 (2017), 239–252; [Deundyak V. M., Kosolapov Y. V., Lelyuk E. A., "Decoding the Tensor Product of MLD Codes and Applications for Code Cryptosystems", *Modeling and Analysis of Information Systems*, **24**:2 (2017), 239–252, (in Russian).]

[4] Деундяк В.М., Косолапов Ю.В., "Использование тензорного произведения кодов Рида–Маллера в асимметричной криптосистеме типа Мак-Элиса и анализ ее стойкости к атакам на шифрограмму", *Вычислительные технологии*, **22**:4 (2017), 43–60; [Deundyak V. M., Kosolapov Y. V., "The use of the tensor product of Reed–Muller codes in asymmetric McEliece type cryptosystem and analysis of its resistance to attacks on the cryptogram", *Computational Technologies*, **22**:4 (2017), 43–60, (in Russian).]

[5] Milies C. P., Sehgal S. K., *An inroduction to Group Rings*, Kluwer Academic Publishers, Boston, 2002.

[6] Сидельников В. М., Казарин Л. С., "О групповой алгебре группы диэдра и сложности умножения матриц второго порядка", *Тр. по дискр. матем.*, **11**:1 (2008), 109–118; [Sidel'nikov V. M., Kazarin L. S., "On a group algebra of a dihedral group and complexity of multiplication of second order matrices", *Tr. Diskr. Mat.*, **11**:1 (2008), 109–118, (in Russian).]

[7] Martinez F. E. B., "Structure of finite dihedral group algebra", *Finite Fields and Their Applications*, **35** (2015), 204–214.

[8] Винберг Э. Б., *Курс алгебры*, МЦНМО, М., 2013; [E. B. Vinberg, *Course in Algebra*, Moscow, 2013, (in Russian).]

[9] Циммерман К.-Х., *Методы теории модулярных представлений в алгебраической теории кодирования*, МЦНМО, М., 2011; [Tsimmerman K.-Kh, *Metody teorii modulyarnykh predstavleniy v algebraicheskoy teorii kodirovaniya*, Moscow, 2011, (in Russian).]

[10] Деундяк В. М., Косолапов Ю. В., "Алгоритмы для мажоритарного декодирования групповых кодов", *Модел. и анализ информ. систем*, **22**:4 (2015), 464–482; [Deundyak V. M., Kosolapov Y. V., "Algorithms for Majority Decoding of Group Codes", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 464–482, (in Russian).]

[11] Jacobson N., *Structure of rings*, American Mathematical Soc., 1956.

[12] Сидельников В. М., *Теория кодирования*, Физматлит, М., 2011; [Sidelnikov V. M., *Teoriya kodirovaniya*, Fizmatlit, Moscow, 2011, (in Russian).]

Vedenev K. V., Deundyak V. M., "Codes in Dihedral Group Algebra", *Modeling and Analysis of Information Systems*, **25**:2 (2018), 232–245.

DOI: 10.18255/1818-1015-2018-2-232-245

Abstract. Robert McEliece developed an asymmetric encryption algorithm based on the use of binary Goppa codes in 1978 and no effective key attacks has been described yet. Variants of this cryptosystem are known due to the use of different codes types, but most of them were proven to be less secure. Code cryptosystems are considered an alternate to number-theoretical ones in connection with the development of quantum computing. So, the new classes of error-correcting codes are required

for building new resistant code cryptosystems. Non-commutative codes, which simply are ideals of finite non-commutative group algebras, are an option. The Artin–Wedderburn theorem implies that a group algebra is isomorphic to a finite direct sum of matrix algebras, when the order of the group and the field characteristics are relatively prime. This theorem is important to study the structure of a non-commutative code, but it gives no information about summands and the isomorphism. In case of a dihedral group these summands and the isomorphism were found by F. E. Brochero Martinez. The purpose of the paper is to study codes in dihedral group algebras as and when the order of a group and a field characteristics are relatively prime. Using the result of F. E. Brochero Martinez, we consider a structure of all dihedral codes in this case and the codes induced by cyclic subgroup codes.

Keywords: non-commutative groups, group algebra, non-commutative codes, code cryptosystems

On the authors:
Kirill V. Vedenev, orcid.org/0000-0002-7893-655X, BSc student,
Southern Federal University
105/42 Bolshaya Sadovaya Str., Rostov-on-Don 344006, Russia, e-mail: vedenevk@gmail.com

Vladimir M. Deundyak, orcid.org/0000-0001-8258-2419, PhD,
Southern Federal University
105/42 Bolshaya Sadovaya Str., Rostov-on-Don 344006, Russia,
FGNU NII "Specvuzavtomatika"
51 Gazetniy lane, Rostov-on-Don 344002, Russia, e-mail: vl.deundyak@gmail.com