УДК 004.9
doi: 10.21685/2072-3059-2023-1-4

Реализация масштабируемых интеллектуальных систем и приложений на базе мобильных агентов

В. И. Волчихин[1], Н. С. Карамышева[2], Д. С. Свищев[3], С. А. Зинкин[4]

[1,2,3,4]Пензенский государственный университет, Пенза, Россия

[1]president@pnzgu.ru, [2,3]vt@pnzgu.ru, [4]zsa49@yandex.ru

Аннотация. *Актуальность и цели.* Описана технология реализации распределенного алгоритма в агентно-ориентированной метакомпьютерной или облачно-сетевой грид-среде, основанная на поддержке динамического взаимодействия, перемещения, добавления и удаления агентов. Целью работы является выработка рекомендаций по использованию логических методов и моделей искусственного интеллекта при создании масштабируемых интеллектуальных приложений. Решение данной задачи актуально и связано с выделением структурированной информации из больших объемов данных и последующей ее обработкой для размещения в базах данных и знаний. *Материалы и методы.* Использованы концептуальные и другие модели искусственного интеллекта, обеспечивающие интероперабельность интеллектуальных систем на уровнях внешнего и внутреннего проектирования, т.е. взаимную согласованность решаемых задач с методами проектирования самой интеллектуальной системы. *Результаты.* Технология позволяет масштабировать вычислительные ресурсы за счет добавления новых агентов в группу, экземпляры которой параллельно и независимо друг от друга выполняют заданные операции. Реализован динамический поиск агентов для их взаимодействия «на лету». *Выводы*. Предложена и апробирована на практике методика реализации распределенного метакомпьютерного приложения в среде компьютерной сети, основанная на комплексном использовании предметно-ориентированной мультиагентной системы и концепции агентно-ориентированного программирования в соответствии с заданными концептуальными графами. В отличие от других методик, предложено код программы создавать непосредственно по концептуальному графу. Используется открытая, динамично развивающаяся агентская платформа, обеспечивающая скоординированную работу агентов в соответствии с отношениями концептуального графа. Предложенная методика позволяет оперативно создавать распределенные масштабируемые приложения для обработки больших данных.

Ключевые слова: масштабируемые вычисления, мобильные агенты, распределенные вычисления, логическая модель, концептуальная модель, интероперабельность системной и функциональной архитектуры

Для цитирования: Волчихин В. И., Карамышева Н. С., Свищев Д. С., Зинкин С. А. Реализация масштабируемых интеллектуальных систем и приложений на базе мобильных агентов // Известия высших учебных заведений. Поволжский регион. Технические науки. 2023. № 1. С. 44–63. doi: 10.21685/2072-3059-2023-1-4

Implementation of scalable intelligent systems and applications based on mobile agents

V.I. Volchikhin[1], N.S. Karamysheva[2], D.S. Svishchev[3], S.A. Zinkin[4]

[1,2,3,4]Penza State University, Penza, Russia
[1]president@pnzgu.ru, [2,3]vt@pnzgu.ru, [4]zsa49@yandex.ru

Abstract. *Background*. A method for implementing a distributed metacomputer application in a computer network environment is proposed and tested in practice, based on the integrated use of a domain-specific multi-agent system and the concept of agent-oriented programming in accordance with given conceptual graphs. The purpose of the study is to develop recommendations on the use of logical methods and models of artificial intelligence when creating scalable intelligent applications. The solution of this problem is relevant and is associated with the extraction of structured information from large volumes of data and its subsequent processing for placement in databases and knowledge. *Materials and methods.* Conceptual and other models of artificial intelligence were used to ensure the interoperability of intelligent systems at the levels of external and internal design, i.e. mutual consistency of the tasks to be solved with the design methods of the intelligent system itself. *Results.* The technology allows scaling computing resources by adding new agents to a group, instances of which perform the specified operations in parallel and independently of each other. Implemented a dynamic search for agents for their interaction "hurriedly". *Conclusions*. A method for implementing a distributed metacomputer application in a computer network environment is proposed and tested in practice, it is based on the integrated use of a domain-specific multi-agent system and the concept of agent-oriented programming in accordance with given conceptual graphs. Unlike other methods, it is proposed to create the program code directly according to the conceptual graph. An open, dynamically developing agent platform is used, which ensures the coordinated work of agents in accordance with the relations of the conceptual graph. The proposed technique allows you to quickly create distributed scalable applications for processing big data.

Keywords: scalable computing, mobile agents, distributed computing, logical model, conceptual model, system and functional architecture interoperability

For citation: Volchikhin V.I., Karamysheva N.S., Svishchev D.S., Zinkin S.A. Implementation of scalable intelligent systems and applications based on mobile agents. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2023;(1):44–63. (In Russ.). doi: 10.21685/2072-3059-2023-1-4

Введение

Отличительной особенностью задач, выполняемых в рамках научных работ или возникающих процессе проведения масштабных исследований, является необходимость использования распределенных вычислительных систем (РВС), обеспечивающих высокую производительность.

Реализация распределенных вычислительных сред, являющихся результатом интеграции сетевых, облачных и высокопроизводительных систем, обеспечивает условия, позволяющие реализовывать сложные распределенные алгоритмы. Реализация подобных сред способствует увеличению мощности совокупного ресурса вычислительной среды.

Динамически развивающимся и набирающим популярность подходом к управлению вычислениями в РВС является разработка уникальных приложений, основанных на использовании мультиагентных систем (МАС). Исследования, проводимые в области МАС, показали, что при их разработке могут быть использованы концептуальные модели искусственного интеллекта [1, 2]. На текущий момент на основе МАС разработано большое число программных продуктов, связанных с агентными платформами [3, 4]. Приложе-

ния МАС разнообразны, но решение распределенных задач в сетевой среде продолжает оставаться проблемой, требующей эффективного решения.

Наиболее популярным представителем агентно-ориентированных платформ является фреймворк Java Agent Development Framework (JADE), который разрабатывается и поддерживается компанией Telecom Italia Lab. Дальнейшее описание фреймворка JADE будет соответствовать источникам [5, 6].

Платформа JADE представляет собой программную среду, используемую для разработки как МАС, так и корпоративных приложений. Платформа поддерживает стандарты интеллектуальных систем и предоставляет возможность динамического взаимодействия с сетевыми узлами, называемыми «агентами» в терминологии МАС JADE. Среда предоставляет программные средства для регистрации, активации и деактивации агентов в мультиагентной системе, что позволяет реализовать их включение и исключение из вычислительных процессов.

В качестве показательного примера обработки большого потока данных дальнейшее рассмотрение проведено на основании типичного графа обработки данных на рис. 1 ([7], с изменениями).

1. Инфраструктура для развертывания агентно-базированных интеллектуальных систем

Работам по использованию сетевой среды при масштабных вычислениях было уделено внимание в статьях [8–13], но, в отличие от них, в настоящей работе большее внимание уделено созданию новой информационной технологии для проектирования крупных масштабируемых приложений на основе концептуальных и поведенческих моделей.

В качестве инфраструктуры для развертывания приложения, представленного в рамках данной работы, используется глобальная сеть, обеспечивающая жизненный цикл платформы JADE, в рамках которого регистрируются и создаются необходимые для функционирования среды контейнеры системы. Рисунок 2 демонстрирует виртуальное размещение узлов, содержащих JADE-агенты, а также каналы связи, обеспечивающие их взаимодействие. Структура, изображенная на рис. 2, представляет собой обычную TCP/IP сеть, которая не привязана к какой-либо конкретной сети; рассмотренная в настоящей работе реализация распределенного интеллектуального приложения может быть развернута в любом фрагменте TCP/IP сети, в том числе при посредничестве провайдера облачных сервисов.

Платформа JADE для создания мультиагентных систем, помимо стандартных программных библиотек, включает в свой состав среду исполнения. Эта среда обеспечивает систему базовыми службами и является основой для ее функционирования, поэтому для нее определен высший приоритет при развертывании программного компонента.

Для данной платформы определена следующая терминология, необходимая для реализации процессов: *контейнером* называется экземпляр JADE-приложения, развернутый для исполнения; *набор контейнеров системы* является платформой, которая предоставляет обобщенный интерфейс, представленный в виде однородного слоя и обеспечивающий сокрытие реализации низкоуровневого функционала по взаимодействию контейнеров внутри сети [5, 6].

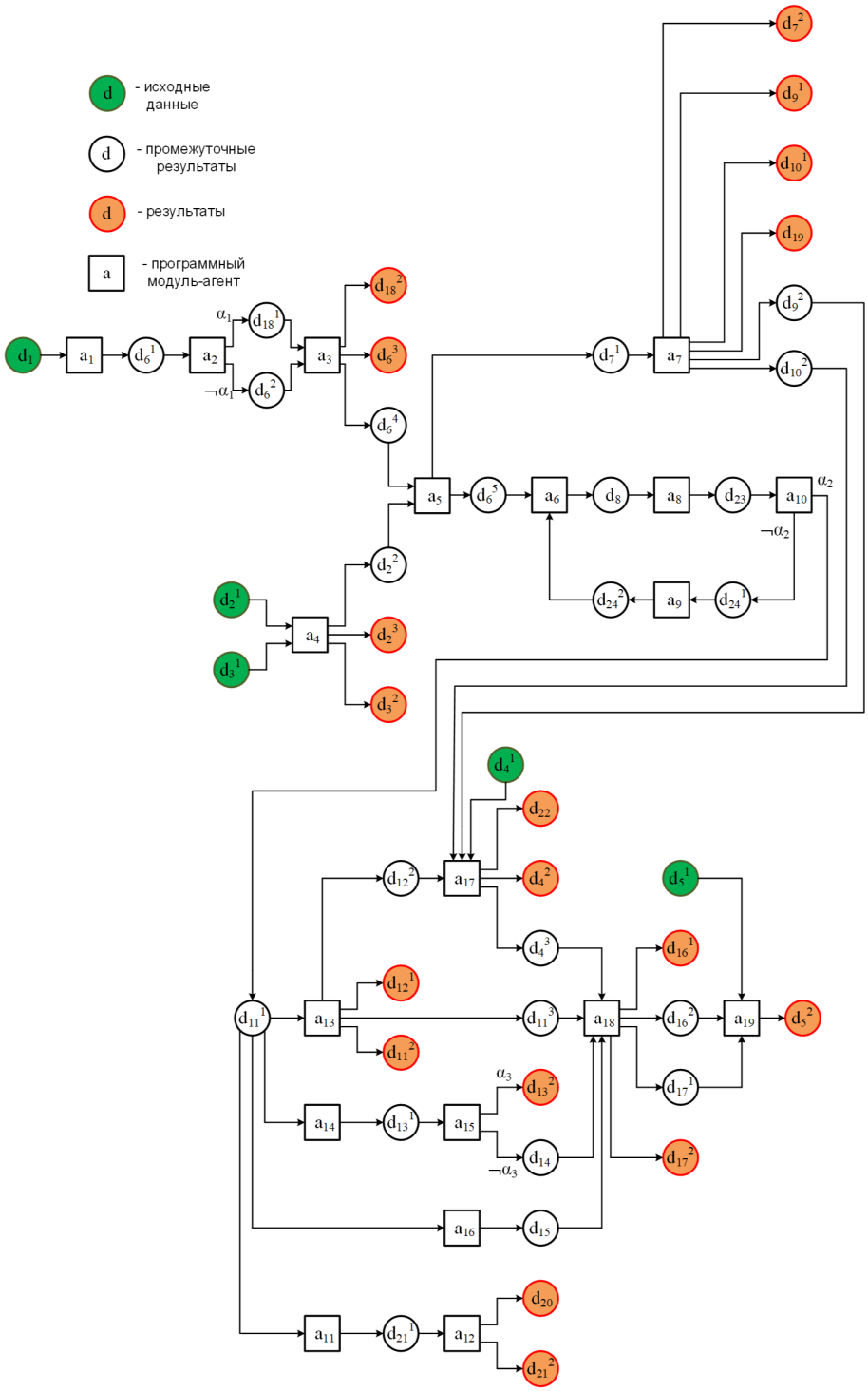

Рис. 1. Потоки обрабатываемых данных в агентно-ориентированной среде

Фреймворк MAC используется в качестве программного обеспечения промежуточного уровня (*middlware software*), которое предназначено для разработки мультиагентных систем, базирующихся на основе транспортной

архитектуры «точка-точка». Фреймворк включает в свой состав следующие компоненты: среду, обеспечивающую жизненный цикл агентов и реализующую регистрацию и деактивацию агентов в среде; набор библиотечных классов и функций; совокупность графических утилит, предназначенных для администрирования, наблюдения и сбора метрик о жизненном цикле агентов [4–6].

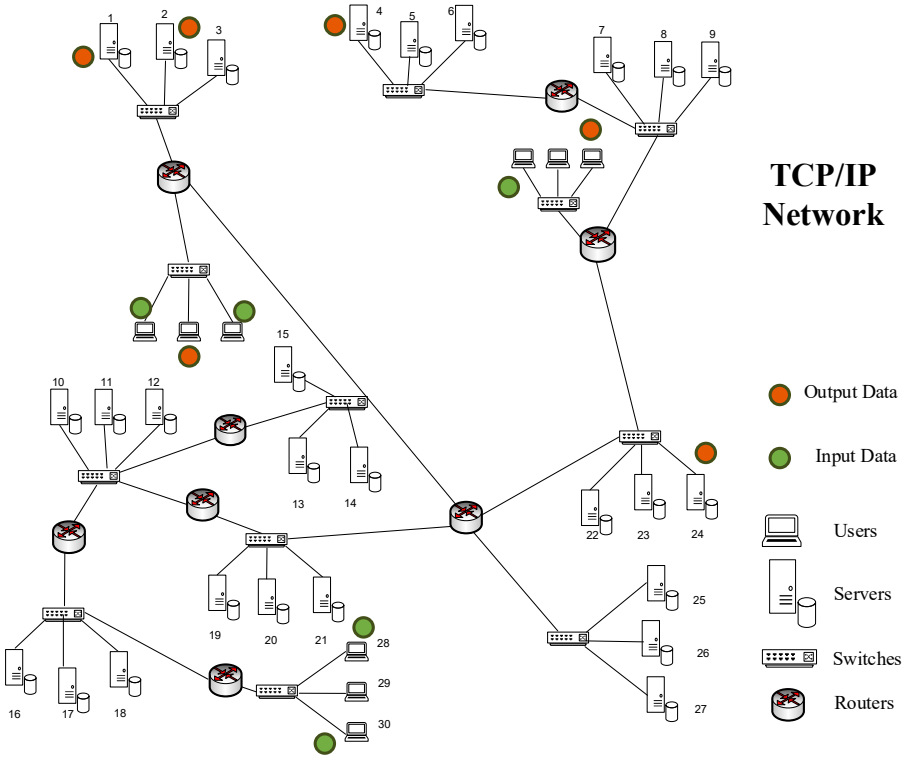

Рис. 2. Инфраструктура для развертывания агентно-базированных интеллектуальных систем – TCP/IP сеть

Программное обеспечение метакомпьютера представляет собой кросс-платформенное приложение, разработанное на языке программирования Java и обеспечивающее обработку данных в рамках распределенного алгоритма, который основан на парадигме передачи сообщений и реализуется в сети ЭВМ стационарными агентами. На рисунке 3 представлена диаграмма развертывания, отображающая процессы взаимодействия и размещения агентов по узлам, определенным в рамках алгоритма.

2. Технические требования системы

Для развертывания, запуска и обеспечения корректной работы агентной платформы JADE, а также распределенного приложения, построенного на ее основе, определены следующие минимальные требования [5, 6]:

– в качестве минимально необходимой реализации JVM (виртуальной машины Java) для исполнения мультиагентного приложения используется JRE (*Java Runtime Environment*) – окружение, которое включает в себя саму JVM и набор стандартных библиотек и классов;

– другим основным требованием является наличие среды для разработки программ на языке Java – JDK (*Java Development Kit*), которая представляет собой совокупность JRE и основательный набор инструментов разработчика: компилятор Java, библиотеки стандартных классов и примеров, а также документацию и вспомогательные классы-утилиты.

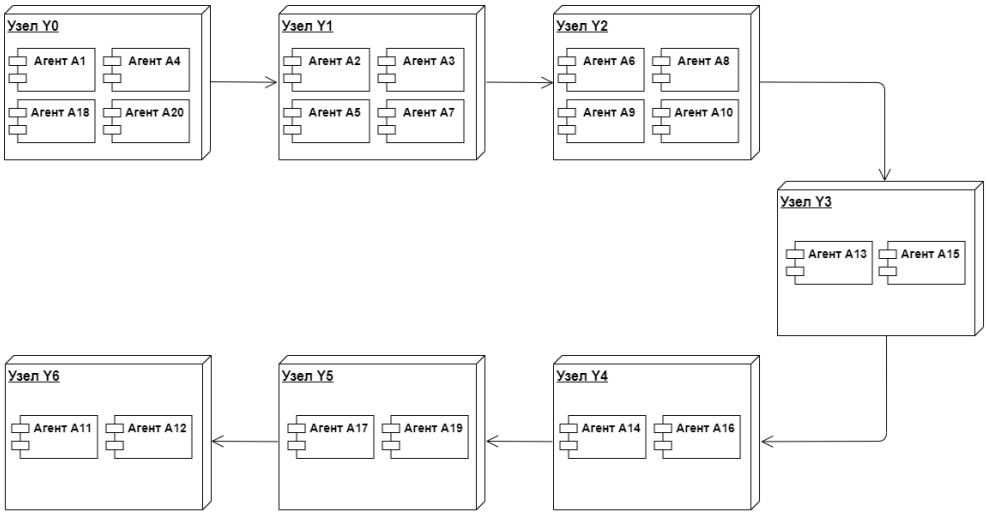

Рис. 3. UML-диаграмма развертывания агентов в метакомпьютерной среде

При разработке агентно-базированного приложения использованы: язык программирования Java, среда разработки IntelliJ IDEA, фреймворк JADE и фреймворк Apache Maven, предназначенный для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM.

3. Конфигурация приложения

В качестве конфигурационных данных для старта приложения используются следующие: определение Main-класса, который запускает процесс работы приложения; в данном случае таким классом является класс jade.Boot, предоставляемый платформой JADE. Определение типа и наименования агентов, а также порядка их создания и настройки за счет передаваемых параметров, осуществляется посредством определения аргументов программы, а именно используются: аргумент для старта проекта с административным GUI для работы с системой (флаг: -gui); аргументы для создания агентов, используемые сразу же после инициализации главного контейнера, что позволяет уйти от «ручного» создания агентов и производить данные манипуляции на программном уровне, указав наименование и класс агента.

4. Концептуальная модель приложения

На рис. 4 представлен концептуальный граф, описывающий декларативные и процедурные знания о распределенном приложении, выполняемом в сетевой среде.

При помощи сообщений в распределенном алгоритме передается управление от одного оператора другому; сообщения могут содержать ре-

зультаты вычислений, запросы к базам данных и результаты выполнения данных запросов.

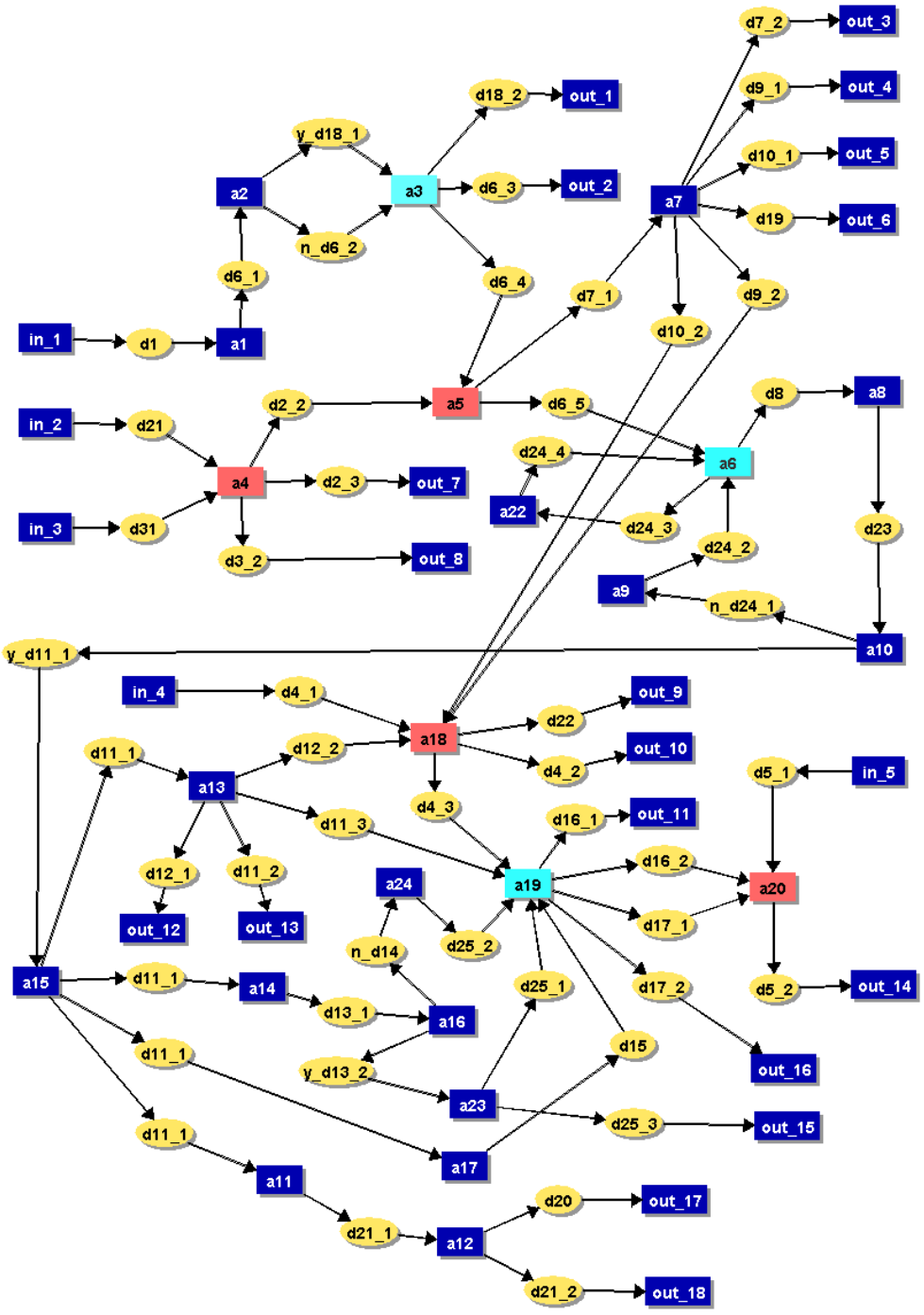

Рис. 4. Концептуальный граф распределенного приложения

На концептуальном графе представлены агенты, обозначенные концептами. Агенты делятся на три вида: входные агенты, которые в текущей реали-

зации проекта заменены одним агентом, являющимся входной точкой программы; промежуточные агенты, которые осуществляют обработку данных в соответствии с логикой, заданной концептуальным графом; терминальные агенты, которые в рамках данной работы рассматриваются как агенты, принимающие итоговые сообщения. Для агентов, имеющих более одного входа и/или выхода, входная и выходная логика доопределены логико-алгебраическими выражениями.

При построении концептуальных графов использовано открытое приложение CharGer [14, 15].

Представленная ниже система правил вывода для распределенного метакомпьютерного приложения определяет последовательность обработки данных и составлена непосредственно по концептуальному графу.

Система правил вывода, определяющих причинно-следственные связи при обработке данных распределенным метакомпьютерным приложением

1. $d_1(in_1, a_1) \supset d_{6,1}(a_1, a_2)$;
2. $d_{6,1}(a_1, a_2) \,\&\, z_{6,1} \supset yd_{18,1}(a_2, a_3)$;
3. $d_{6,1}(a_1, a_2) \,\&\, \neg z_{6,1} \supset nd_{6,2}(a_2, a_3)$;
4. $yd_{18,1}(a_2, a_3) \supset d_{18,2}(a_3, out_1) \,\&\, d_{6,3}(a_3, out_2) \,\&\, d_{6,4}(a_3, a_5)$;
5. $nd_{6,2}(a_2, a_3) \supset d_{18,2}(a_3, out_1) \,\&\, d_{6,3}(a_3, out_2) \,\&\, d_{6,4}(a_3, a_5)$;
6. $d_{6,4}(a_3, a_5) \,\&\, d_{2,2}(a_4, a_5) \supset d_{7,1}(a_5, a_7) \,\&\, d_{6,5}(a_5, a_6)$;
7. $d_{21}(in_2, a_4) \,\&\, d_{31}(in_3, a_4) \supset d_{2,2}(a_4, a_5) \,\&\, d_{2,3}(a_4, out_7) \,\&\, d_{3,2}(a_4, out_8)$;
8. $d_{7,1}(a_5, a_7) \supset d_{7,2}(a_7, out_3) \,\&\, d_{9,1}(a_7, out_4) \,\&\, d_{10,1}(a_7, out_5) \,\&\, d_{19}(a_7, out_6) \,\&$
$\&\, d_{9,2}(a_7, a_{18}) \,\&\, d_{10,2}(a_7, a_{18})$;
9. $d_{9,2}(a_7, a_{18}) \,\&\, d_{10,2}(a_7, a_{18}) \,\&\, d_{4,1}(in_4, a_{18}) \,\&\, d_{12,2}(a_{13}, a_{18}) \supset$
$\supset d_{22}(a_{18}, out_9) \,\&\, d_{4,2}(a_{18}, out_{10}) \,\&\, d_{4,3}(a_{18}, a_{19})$;
10. $(d_{6,5}(a_5, a_6) \vee d_{24,4}(a_{22}, a_6)) \,\&\, d_{24,2}(a_9, a_6) \supset d_8(a_6, a_8) \,\&\, d_{24,3}(a_6, a_{22})$;
11. $d_8(a_6, a_8) \supset d_{23}(a_8, a_{10})$;
12. $d_{23}(a_8, a_{10}) \,\&\, z_{23} \supset yd_{11,1}(a_{10}, a_{15})$;
13. $d_{23}(a_8, a_{10}) \,\&\, \neg z_{23} \supset nd_{24,1}(a_{10}, a_9)$;
14. $nd_{24,1}(a_{10}, a_9) \supset d_{24,2}(a_9, a_6)$;
15. $d_{24,3}(a_6, a_{22}) \supset d_{24,4}(a_{22}, a_6)$;
16. $yd_{11,1}(a_{10}, a_{15}) \supset d_{11,1}(a_{15}, a_{13}) \,\&\, d_{11,1}(a_{15}, a_{14}) \,\&\, d_{11,1}(a_{15}, a_{17}) \,\&$
$\&\, d_{11,1}(a_{15}, a_{11})$;
17. $d_{11,1}(a_{15}, a_{13}) \supset d_{12,2}(a_{13}, a_{18}) \,\&\, d_{11,3}(a_{13}, a_{19}) \,\&\, d_{11,2}(a_{13}, out_{13}) \,\&$
$\&\, d_{12,1}(a_{13}, out_{12})$;
18. $d_{11,3}(a_{13}, a_{19}) \,\&\, d_{4,3}(a_{18}, a_{19}) \,\&\, (d_{25,1}(a_{23}, a_{19}) \vee d_{25,2}(a_{24}, a_{19})) \,\&$
$\&\, d_{15}(a_{17}, a_{19}) \supset d_{16,1}(a_{19}, out_{11}) \,\&\, d_{16,2}(a_{19}, a_{20}) \,\&\, d_{17,1}(a_{19}, a_{20}) \,\&$
$\&\, d_{17,2}(a_{19}, out_{16})$;
19. $d_{13,1}(a_{14}, a_{16}) \,\&\, z_{13,1} \supset yd_{13,2}(a_{16}, a_{23})$;
20. $d_{13,1}(a_{14}, a_{16}) \,\&\, \neg z_{13,1} \supset nd_{14}(a_{16}, a_{24})$;
21. $nd_{14}(a_{16}, a_{24}) \supset d_{25,2}(a_{24}, a_{19})$;
22. $yd_{13,2}(a_{16}, a_{23}) \supset d_{25,1}(a_{23}, a_{19}) \,\&\, d_{25,3}(a_{23}, out_{15})$;
23. $d_{16,2}(a_{19}, a_{20}) \,\&\, d_{17,1}(a_{19}, a_{20}) \,\&\, d_{5,1}(in_5, a_{20}) \supset d_{5,2}(a_{20}, out_{14})$;
24. $d_{11,1}(a_{15}, a_{17}) \supset d_{15}(a_{17}, a_{19})$;

25. $d_{11,1}(a_{15}, a_{11}) \supset d_{21,1}(a_{11}, a_{12})$;

26. $d_{21,1}(a_{11}, a_{12}) \supset d_{20}(a_{12}, out_{17})$ & $d_{21,2}(a_{12}, out_{18})$.

На основании правил вывода далее формируется система логико-алгебраических операционных выражений (ЛАОВ), используемая в качестве исполнимых спецификаций при программировании интеллектуального приложения; правила формирования ЛАОВ описаны в работе [9], здесь ограничимся только комментарием. Операция выбора по условию, или ветвления, представляется формулой $[(A_i|Z_i)/Y_k](A_j/Y_m \oplus A_h/Y_n)$: «в зависимости от результата Z_i, полученного агентом A_i , размещенном на узле Y_k, управление при Z_i = **true** передается агенту A_j, размещенному на узле Y_m, а при Z_i = **false** управление передается агенту A_h, размещенному на узле Y_n».

Система логико-алгебраических операционных выражений для реализации распределенного приложения

1. $d_1(in_1, a_1) \leftarrow$ **true**;
2. $[d_1(in_1, a_1)](d_{6,1}(a_1, a_2) \leftarrow$ **true**, $d_1(in_1, a_1) \leftarrow$ **false** $\oplus E$);
3. $[d_{6,1}(a_1, a_2)$ & $z_{6,1}](yd_{18,1}(a_2, a_3) \leftarrow$ **true**, $d_{6,1}(a_1, a_2) \leftarrow$ **false** $\oplus$ $nd_{6,2}(a_2, a_3) \leftarrow$ **true**, $d_{6,1}(a_1, a_2) \leftarrow$ **false**);
4. $[yd_{18,1}(a_2, a_3) \vee nd_{6,2}(a_2, a_3)](d_{18,2}(a_3, out_1) \leftarrow$ **true**, $d_{6,3}(a_3, out_2) \leftarrow$ **true**, $d_{6,4}(a_3, a_5) \leftarrow$ **true**, $yd_{18,1}(a_2, a_3) \leftarrow$ **false**, $nd_{6,2}(a_2, a_3) \leftarrow$ **false** $\oplus E$);
5. $[d_{6,4}(a_3, a_5)$ & $d_{2,2}(a_4, a_5)](d_{7,1}(a_5, a_7) \leftarrow$ **true**, $d_{6,5}(a_5, a_6) \leftarrow$ **true**, $d_{6,4}(a_3, a_5) \leftarrow$ **false**, $d_{2,2}(a_4, a_5) \leftarrow$ **false** $\oplus E$);
6. $d_{21}(in_2, a_4) \leftarrow$ **true**;
7. $d_{31}(in_3, a_4) \leftarrow$ **true**;
8. $[d_{21}(in_2, a_4)$ & $d_{31}(in_3, a_4)](d_{2,2}(a_4, a_5) \leftarrow$ **true**, $d_{2,3}(a_4, out_7) \leftarrow$ **true**, $d_{3,2}(a_4, out_8) \leftarrow$ **true**, $d_{21}(in_2, a_4) \leftarrow$ **false**, $d_{31}(in_3, a_4) \leftarrow$ **false** $\oplus E$);
9. $[d_{7,1}(a_5, a_7)](d_{7,2}(a_7, out_3) \leftarrow$ **true**, $d_{9,1}(a_7, out_4) \leftarrow$ **true**, $d_{10,1}(a_7, out_5) \leftarrow$ **true**, $d_{19}(a_7, out_6) \leftarrow$ **true**, $d_{9,2}(a_7, a_{18}) \leftarrow$ **true**, $d_{10,2}(a_7, a_{18}) \leftarrow$ **true**, $d_{7,1}(a_5, a_7) \leftarrow$ **false** $\oplus E$);
10. $[d_{9,2}(a_7, a_{18})$ & $d_{10,2}(a_7, a_{18})$ & $d_{4,1}(in_4, a_{18})$ & $d_{12,2}(a_{13}, a_{18})]$ $(d_{22}(a_{18}, out_9) \leftarrow$ **true**, $d_{4,2}(a_{18}, out_{10}) \leftarrow$ **true**, $d_{4,3}(a_{18}, a_{19}) \leftarrow$ **true**, $d_{9,2}(a_7, a_{18}) \leftarrow$ **false**, $d_{10,2}(a_7, a_{18}) \leftarrow$ **false**, $d_{4,1}(in_4, a_{18}) \leftarrow$ **false**, $d_{12,2}(a_{13}, a_{18}) \leftarrow$ **false** $\oplus E$);
11. $[(d_{6,5}(a_5, a_6) \vee d_{24,4}(a_{22}, a_6))$ & $d_{24,2}(a_9, a_6)](d_8(a_6, a_8) \leftarrow$ **true**, $d_{24,3}(a_6, a_{22}) \leftarrow$ **true**, $d_{6,5}(a_5, a_6) \leftarrow$ **false**, $d_{24,4}(a_{22}, a_6) \leftarrow$ **false**, $d_{24,2}(a_9, a_6) \leftarrow$ **false** $\oplus E$);
12. $[d_8(a_6, a_8)](d_{23}(a_8, a_{10}) \leftarrow$ **true**, $d_8(a_6, a_8) \leftarrow$ **false** $\oplus E$);
13. $[d_{23}(a_8, a_{10})$ & $z_{23}](yd_{11,1}(a_{10}, a_{15}) \leftarrow$ **true**, $d_{23}(a_8, a_{10}) \leftarrow$ **false** $\oplus$ $nd_{24,1}(a_{10}, a_9) \leftarrow$ **true**, $d_{23}(a_8, a_{10}) \leftarrow$ **false**);
14. $[nd_{24,1}(a_{10}, a_9)]$ $(d_{24,2}(a_9, a_6) \leftarrow$ **true**, $nd_{24,1}(a_{10}, a_9) \leftarrow$ **false** $\oplus E$);
15. $[d_{24,3}(a_6, a_{22})](d_{24,4}(a_{22}, a_6) \leftarrow$ **true**, $d_{24,3}(a_6, a_{22}) \leftarrow$ **false** $\oplus E$);
16. $[yd_{11,1}(a_{10}, a_{15})]($ $d_{11,1}(a_{15}, a_{13}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{14}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{17}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{11}) \leftarrow$ **true**, $yd_{11,1}(a_{10}, a_{15}) \leftarrow$ **false** $\oplus E$);
17. $[d_{11,1}(a_{15}, a_{13})](d_{12,2}(a_{13}, a_{18}) \leftarrow$ **true**, $d_{11,3}(a_{13}, a_{19}) \leftarrow$ **true**, $d_{11,2}(a_{13}, out_{13}) \leftarrow$ **true**, $d_{12,1}(a_{13}, out_{12}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{13}) \leftarrow$ **false** $\oplus E$);

18. $[d_{11,3}(a_{13}, a_{19})$ & $d_{4,3}(a_{18}, a_{19})$ & $(d_{25,1}(a_{23}, a_{19}) \vee d_{25,2}(a_{24}, a_{19}))$ & $d_{15}(a_{17}, a_{19})](d_{16,1}(a_{19}, out_{11}) \leftarrow$ **true**, $d_{16,2}(a_{19}, a_{20}) \leftarrow$ **true**, $d_{17,1}(a_{19}, a_{20}) \leftarrow$ **true**, $d_{17,2}(a_{19}, out_{16}) \leftarrow$ **true**, $d_{11,3}(a_{13}, a_{19}) \leftarrow$ **false**, $d_{4,3}(a_{18}, a_{19}) \leftarrow$ **false**, $d_{15}(a_{17}, a_{19}) \leftarrow$ **false**, $d_{25,1}(a_{23}, a_{19}) \leftarrow$ **false**, $d_{25,2}(a_{24}, a_{19}) \leftarrow$ **false** $\oplus E$);

19. $[d_{13,1}(a_{14}, a_{16})$ & $z_{13,1}](yd_{13,2}(a_{16}, a_{23}) \leftarrow$ **true**, $d_{13,1}(a_{14}, a_{16}) \leftarrow$ **false** $\oplus$ $nd_{14}(a_{16}, a_{24}) \leftarrow$ **true**, $d_{13,1}(a_{14}, a_{16}) \leftarrow$ **false**);

20. $[nd_{14}(a_{16}, a_{24})](d_{25,2}(a_{24}, a_{19}) \leftarrow$ **true**, $nd_{14}(a_{16}, a_{24}) \leftarrow$ **false** $\oplus E$);

21. $[yd_{13,2}(a_{16}, a_{23})](d_{25,1}(a_{23}, a_{19}) \leftarrow$ **true**, $d_{25,3}(a_{23}, out_{15}) \leftarrow$ **true**, $yd_{13,2}(a_{16}, a_{23}) \leftarrow$ **false** $\oplus E$);

22. $d_{5,1}(in_5, a_{20}) \leftarrow$ **true**;

23. $[d_{16,2}(a_{19}, a_{20})$ & $d_{17,1}(a_{19}, a_{20})$ & $d_{5,1}(in_5, a_{20})](d_{5,2}(a_{20}, out_{14}) \leftarrow$ **true**, $d_{16,2}(a_{19}, a_{20}) \leftarrow$ **false**, $d_{17,1}(a_{19}, a_{20}) \leftarrow$ **false**, $d_{5,1}(in_5, a_{20}) \leftarrow$ **false** $\oplus E$);

24. $[d_{11,1}(a_{15}, a_{17})](d_{15}(a_{17}, a_{19}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{17}) \leftarrow$ **false** $\oplus E$);

25. $[d_{11,1}(a_{15}, a_{11})](d_{21,1}(a_{11}, a_{12}) \leftarrow$ **true**, $d_{11,1}(a_{15}, a_{11}) \leftarrow$ **false** $\oplus E$);

26. $[d_{21,1}(a_{11}, a_{12})](d_{20}(a_{12}, out_{17}) \leftarrow$ **true**, $d_{21,2}(a_{12}, out_{18}) \leftarrow$ **true**, $d_{21,1}(a_{11}, a_{12}) \leftarrow$ **false** $\oplus E$).

Концептуальный граф развертывания агентов на узлах сети формируется самим пользователем или провайдером облачных сервисов. Вариант такого графа представлен на рис. 5.

В табл. 1 представлен список фактов, соответствующих размещению агентов на узлах компьютерной сети, полученный автоматически на основании концептуального графа на рис. 5.

Таблица 1

place(a1,n3). place(a10,n14). place(a11,n26). place(a12,n27). place(a13,n11). place(a14,n21). place(a15,n19). place(a16,n23). place(a17,n27). place(a18,n12). place(a19,n24). place(a2,n1).	place(a20,n17). place(a22,n8). place(a23,n27). place(a24,n22). place(a3,n1). place(a4,n6). place(a5,n7). place(a6,n8). place(a7,n5). place(a8,n5). place(a9,n13). place(in_1,n2).	place(in_2,n2). place(in_3,n2). place(in_4,n9). place(in_5,n15). place(out_1,n4). place(out_10,n16). place(out_11,n16). place(out_12,n20). place(out_13,n20). place(out_14,n18). place(out_15,n25). place(out_16,n25).	place(out_17,n28). place(out_18,n28). place(out_2,n4). place(out_3,n4). place(out_4,n4). place(out_5,n4). place(out_6,n4). place(out_7,n8). place(out_8,n10). place(out_9,n16).

На рис. 6 представлен концептуальный граф системной архитектуры распределенной интеллектуальной системы, полученный в результате применения правил вывода следующего вида [12, 13]:

$$(\forall a \in A, b \in A, y \in Y, z \in Z)\left[send(a,b)\ \&\ place(a,y)\ \&\ place(b,z) \supset link(y,z)\right];$$

$$(\forall a \in A, b \in A, y \in Y, z \in Z)\left[send(a,b)\ \&\ place(a,y)\ \&\ place(b,z) \supset link(z,y)\right].$$

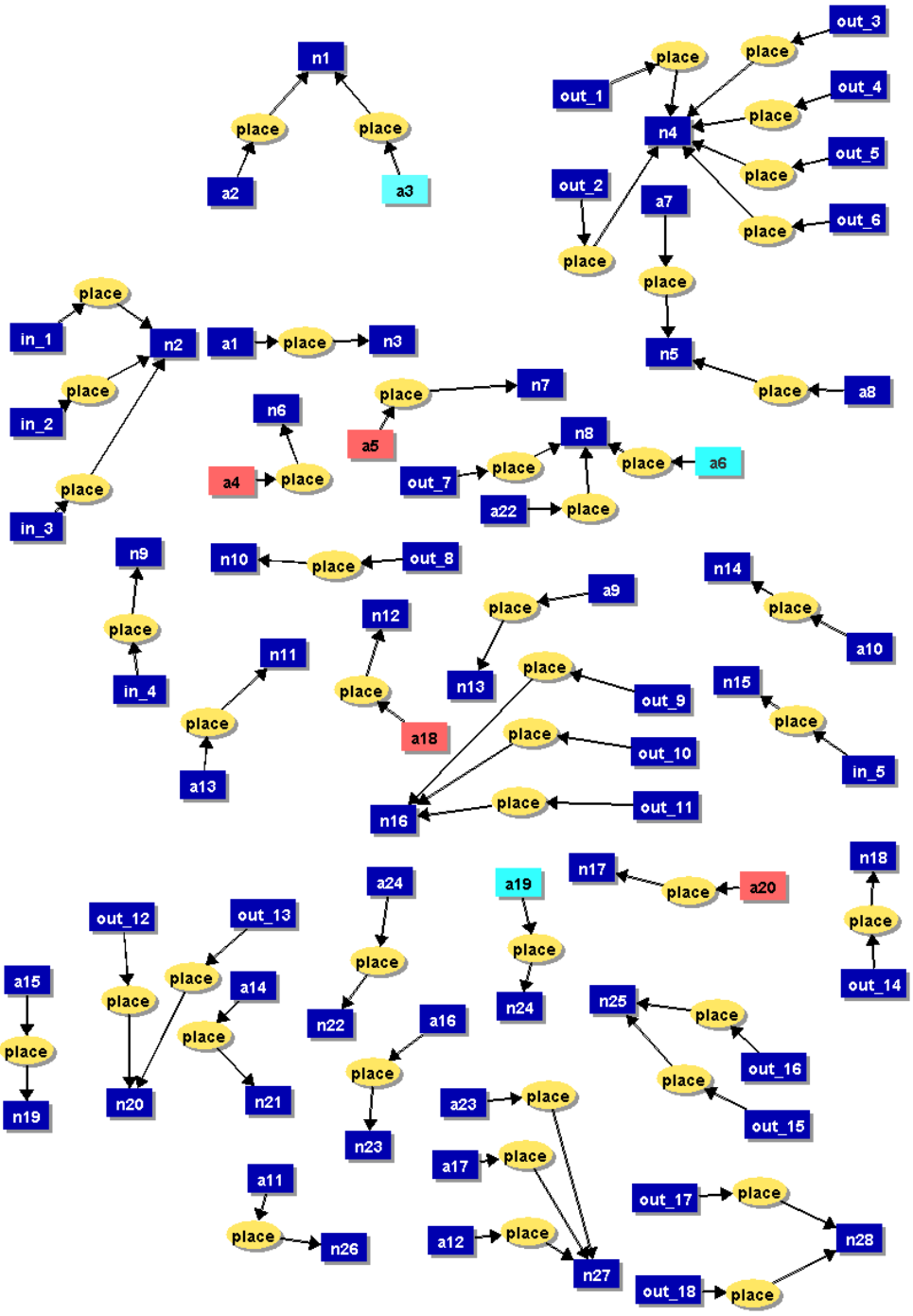

Рис. 5. Концептуальный граф начального развертывания мобильных агентов при реализации распределенного приложения

Здесь первое правило использовано для формирования связей для передачи основных информационных сообщений, а второе – для связей, по которым передаются квитирующие сообщения (эти связи на рис. 6 не показаны во избежание загромождения графа).

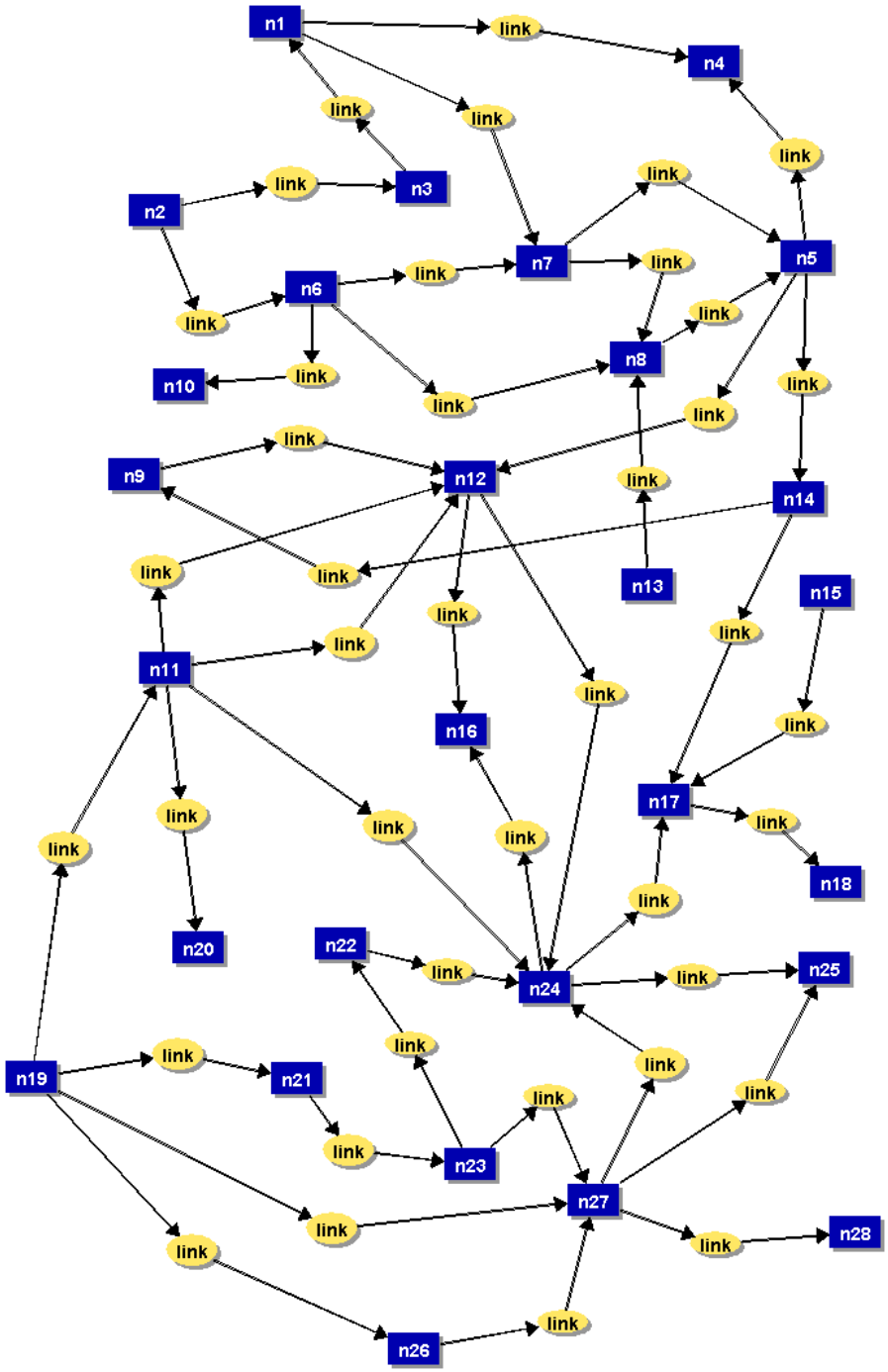

Рис. 6. Концептуальный граф системной архитектуры
распределенной интеллектуальной системы

Для верификации проектируемой системы была использована исполнимая модель на основе сетей Петри, построенная на основе концептуального графа функциональной архитектуры распределенной интеллектуальной системы. Данная сеть Петри приведена на рис. 7.

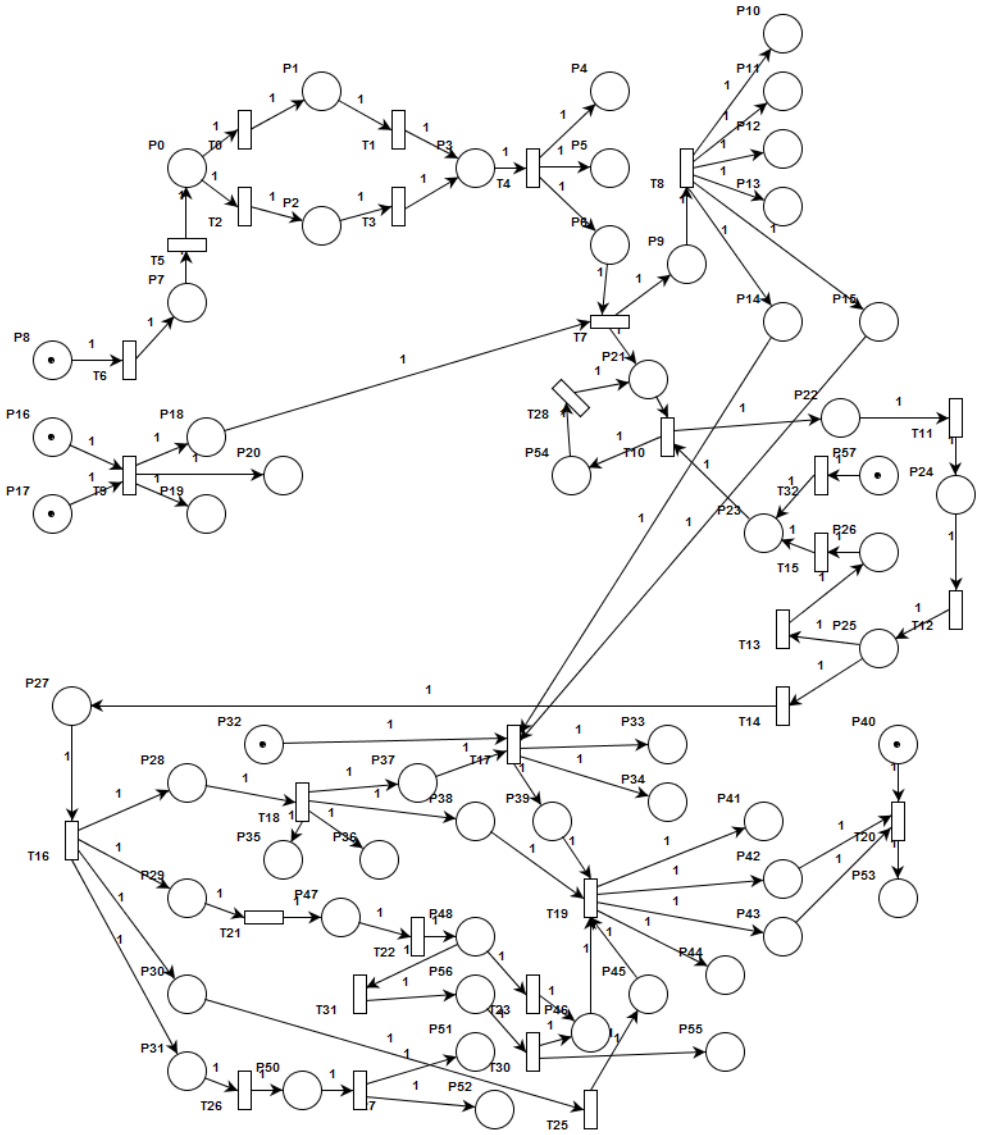

Рис. 7. Сеть Петри, моделирующая перемещение сообщений
в сети интеллектуальных агентов

Полное исследование данной сети было проведено при помощи системы моделирования сетей Петри PIPE [15], которое показало, что сеть в целом обладает свойствами живости и ограниченности (безопасности). При моделировании сеть Петри была закольцована для обеспечения многократного прогона модели.

5. Реализация кросс-платформенного приложения на агентно-базированной платформе JADE

Программа представляет собой программный продукт, реализованный на языке программирования Java с использованием фреймворков JADE (агентная платформа) и Maven (фреймворк для автоматизации и управления сборкой проектов).

Проект разработан в соответствии с концепциями агентно-ориентированного программирования, обеспечивающего описание агентов системы на основе средств, предоставляемых платформой JADE, и объектно-ориентированного программирования, благодаря которым программный продукт представлен в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы составляют иерархию наследования.

Работа распределенного приложения начинается с инициализации агентов и последующей их регистрации в главном контейнере платформы. Последовательность создания экземпляров агентов определяется программным кодом или конфигурационным файлом. Процесс регистрации агента сопровождается установкой его наименования и типа, которые в дальнейшем будут использованы в качестве индексов при поиске агентов в сервисе желтых страниц.

Важной особенностью в реализации распределенного алгоритма является поддержка динамического взаимодействия агентов, их добавления и удаления из системы во время ее исполнения. Данная необходимость может быть объяснена на простом примере возможности масштабирования вычислительных ресурсов за счет добавления нового агента в группу, в которой экземпляры параллельно и независимо друг от друга выполняют заданные операции.

В рамках данной работы реализован динамический поиск агентов для их взаимодействия «на лету». Необходимым требованием каждого агента является регистрация в сервисе желтых страниц каждого созданного экземпляра агента.

Регистрация агентов в сервисе желтых страниц описывается процессом установки следующих параметров для создания агентной публикации: установка значения идентификатора AID (*Agent Identifier*), которое генерируется автоматически при регистрации агента в контейнере; установка типа агента, который отражает функционально выполняемые им функции; установка наименования, которое отражает физическое расположение агента в выполняемом алгоритме.

Далее представлен пример публикации агентов в сервисе желтых страниц, учитывающий специфику распределенного приложения:

```
/**
 * Публикация агента в сервисе желтых страниц
 */
protected void servicePublication(@Nonnull String type) {
    log.info("Try to publish agent {} in yellow pages service", getName());
    DFAgentDescription dfAgentDescription = new DFAgentDescription();
    ServiceDescription serviceDescription = new ServiceDescription();
    dfAgentDescription.setName(getAID());
    serviceDescription.setType(type);
    serviceDescription.setName(getPublishingName());
    dfAgentDescription.addServices(serviceDescription);
    try {
        DFService.register(this, dfAgentDescription);
        log.info("Agent {} successfully published in yellow pages", getName());
    } catch (FIPAException e) {
        log.error("Agent {} publication in yellow pages unavailable", getName());
        takeDown();
    }
}
```

Процесс регистрации агентов является неотъемлемой частью корректного функционирования приложения в динамически изменяемой среде. Такой же важной частью изменяемых в процессе данных является динамический поиск агентов в сервисе желтых страниц.

В качестве критериев поиска опубликованного агента используются следующие параметры: тип зарегистрированного агента, который является обязательным для поиска параметром и охватывает поиск целой группы агентов с указанной в параметре функциональностью; имя опубликованного объекта, которое является опциональным параметром, и в случае его заполнения осуществляется поиск уникального агента, функциональность и наименования которого соответствуют указанным в запросе.

Ниже представлен пример публикации агентов в сервисе желтых страниц, учитывающий специфику распределенного приложения:

```
/**
 * Поиск агентов в сервисе желтых страниц
 * @param agentType тип агента
 * @param agentName наименование агента
 * @return множество идентификаторов найденных объектов
 */
protected Set<AID> findPublishedAgents(@Nonnull String agentType, @Nullable String agentName) {
    log.info("Try to find agents in yellow pages service by name: {}", agentName);
    DFAgentDescription template = new DFAgentDescription();
    ServiceDescription serviceDescription = new ServiceDescription();
    if (agentName != null) {
        serviceDescription.setName(agentName);
    }
    serviceDescription.setType(agentType);
    template.addServices(serviceDescription);
    try {
        return Arrays.stream(DFService.search(this, template))
                .map(DFAgentDescription::getName)
                .collect(Collectors.toSet());
    } catch (FIPAException ex) {
        log.error("An error occurred while searching for a {}: {}", agentName, ex.getMessage());
    }
    return Collections.emptySet();
}
```

6. Взаимодействие агентов в распределенной интеллектуальной системе

В соответствии с распределенным алгоритмом, представленным на рис. 4, запуск взаимодействия агентов по передаче сообщений осуществляется на основе базового интерфейса библиотеки JADE, в ходе чего формируется информационное сообщение входящему агенту в соответствии с определенной статусной моделью.

Агент A_0 регистрирует поведение на прослушивание входящих сообщений и реализацию проксирования полученного сообщения в формате, установленном протоколом взаимодействия, агентам-адресатам, а именно входящим агентам, которыми являются агенты A_1, A_4, A_{18}, A_{20}. Важным усло-

вием является наличие данных агентов в сервисе желтых страниц, а также их успешный поиск по заданным индексам. Поиск сообщений осуществляется с помощью встроенного в платформу механизма обмена сообщениями между агентами. Поиск данных осуществляется по заданным статусам агентов.

Агент A_2 выполняет последовательную передачу полученных данных агенту A_3 на основе выполнения некоторой функции, возвращающей логическое значение, на основе которого выполняется выбор направления передачи сообщения. Агент A_2 обладает стандартным поведением по прослушиванию очереди сообщений, проверки наличия агента-адресата и выполнения логической функции.

Агент A_3 реализует параллельную передачу сообщения сразу нескольким агентам. Поэтому важной особенностью агента является регистрация поведения повторного определения агентов-адресатов, что позволяет динамически отслеживать количество и типы агентов. В данном случае осуществляется передача двух сообщений агенту A_{21} и одного сообщения агенту A_5.

Агент A_4 реализует конъюнктивную функцию на своем входе, ожидая поставки сообщения от всех агентов звена. Только в том случае, когда получены сообщения от всех агентов, осуществляется операция объединения данных и их параллельная отправка агентам A_5, A_{21}.

Агент A_5, аналогично агенту A_4, реализует конъюнктивную функцию на входе, ожидая поставки сообщения от всех агентов звена. Данная функция реализуется за счет специально описанного поведения на основе учета внутренних сообщений от разных агентов, что позволяет параллельно запускать сразу несколько итераций данного алгоритма, в процессе которых сообщения будут накладываться друг на друга. Данный агент, подобно другим, обладает стандартным поведением по поиску и выборке на основе индексов агентов-адресатов A_6, A_7 сообщений, на которые происходит параллельная отправка информации.

Агент A_7 ожидает поступления сообщения от агента A_5 и при его получении осуществляет его обработку и последующую параллельную передачу новых сообщений агентам A_{21}, A_{18}.

Агент A_6 обладает поведением, задающим циклическую обработку получаемых и передаваемых данных. В зависимости от условия, реализуемого агентом A_{10}, агент A_6 прослушивает сообщения по статусной модели от агентов A_9 и A_5, а также осуществляет отправку данных агенту A_8.

Агент A_8 обладает стандартным поведением по прослушиванию очереди сообщений и проверке наличия агента-адресата, которому передается сообщение. В свою очередь агент A_{10}, аналогично агенту A_2, помимо базовых поведений, реализует проверку условия, которое определяет дальнейшего адресата сообщения: либо агента A_9, либо агента A_{15}.

Агент A_{15} осуществляет распараллеливание полученного сообщения сразу на несколько агентов (A_{11}, A_{13}, A_{14}, A_{17}) одного узла, поиск которых осуществляется динамически через сервис желтых страниц, что позволяет управлять динамическим конфигурированием взаимодействия агентов.

После получении сообщения с помощью зарегистрированного поведения агент A_{13} выполняет параллельную передачу данных найденным агентам смежных узлов, а именно агенту A_{19} и терминальному агенту A_{21}.

Агент A_{14} осуществляет последовательную передачу сообщения агенту A_{16} и обладает стандартным поведением, присущим каждому агенту. В свою

очередь агент A_{16}, аналогично агентам A_2 и A_{10}, осуществляет выполнение логической функции, результат которой определяет дальнейшего адресата сообщения: либо агента A_{19}, либо терминального агента A_{21}.

Агенты A_{17} и A_{11} обладают схожим поведением, обеспечивая последовательную передачу полученных данных агентам A_{19} и A_{12} соответственно. Агент A_{12} прослушивает входящие сообщения и при наличии данных в очереди формирует исходящие сообщения, которые передаются терминальному агенту A_{21} в количестве двух партий.

Агенты A_{18}, A_{19} и A_{20} реализуют конъюнктивную функцию на входе, ожидая поставки сообщения от всех агентов звена. Только в том случае, если получены сообщения от всех агентов, осуществляется операция объединения данных и их параллельная отправка сразу нескольким агентам. Тестирование распределенного приложения показало согласованное взаимодействие агентов и правильность получаемых результатов.

Заключение

В работе описана методика реализации распределенного метакомпьютерного приложения в среде компьютерной сети, основанная на комплексном использовании предметно-ориентированной мультиагентной системы и концепции агентно-ориентированного программирования в соответствии с заданными концептуальными графами, описывающими обработку сообщений и данных программными агентами. В отличие от предложенных ранее методик в работах [8–13], код программы создается непосредственно по концептуальному графу. Используется открытая, динамично развивающаяся агентская платформа, обеспечивающая скоординированную работу агентов в соответствии с отношениями концептуального графа. Эта платформа представляет собой эффективную среду для разработки и управления агентскими приложениями в различных предметных областях, в том числе в области метакомпьютинга. Предложенная методика позволяет оперативно создавать распределенные масштабируемые приложения для обработки больших данных.

Список литературы

1. Kravari K., Bassiliades N. A Survey of Agent Platforms // Journal of Artificial Societies and Social Simulation. 2015. Vol. 18 (1), № 11. P. 1–18.
2. Cynthia N., Gregory M. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms // Journal of Artificial Societies and Social Simulation. 2009. Vol. 12, № 2. URL: http://jasss.soc.surrey.ac.uk/12/2/2.html (дата обращения: 02.09.2022).
3. Alluhaybi1 B., Alrahhal M. S., Alzhrani A., Thayananthan V. A Survey: Agent-based Software Technology Under the Eyes of Cyber Security, Security Controls, Attacks and Challenges // International Journal of Advanced Computer Science and Applications (IJACSA). King Abdulaziz University (KAU), Jeddah, Saudi Arabia. 2019. Vol. 10, № 8. P. 211–230.
4. Silva L., Meneguzzi F., Logan B. BDI Agent Architectures: A Survey // Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. Survey track. 2020. P. 4914–4921. doi: 10.24963/ijcai.2020/684
5. Bellifemine F. L., Caire G., Greenwood D. Developing multi-agent systems with JADE. Wiley, 2007. 300 p.
6. Java Agent Development Environment (JADE). URL: http://jade.tilab.com/ (дата обращения: 01.09.2022).

7. Мамиконов А. Г., Кульба В. В. Синтез оптимальных модульных систем обработки данных. М. : Наука, 1986. 276 с.
8. Волчихин В. И., Карамышева Н. С., Горынина А. В., Зинкин С. А. Разработка сетевых агентно-базированных приложений на основе метакомпьютерной технологии // Известия высших учебных заведений. Поволжский регион. Технические науки. 2021. № 4. С. 3–25. doi: 10.21685/2072-3059-2021-4-1
9. Волчихин В. И., Карамышева Н. С., Зинкин С. А., Гурин Е. И. Алгоритмика, логика и моделирование агентно-базированных метакомпьютерных систем с повышенным уровнем параллельности // Известия высших учебных заведений. Поволжский регион. Технические науки. 2022. № 2. С. 5–25. doi: 10.21685/2072-3059-2022-2-1
10. Волчихин В. И., Зинкин С. А., Карамышева Н. С. Организация функционирования облачно-сетевых распределенных вычислительных систем с архитектурой «агенты как сервисы» // Известия высших учебных заведений. Поволжский регион. Технические науки. 2019. № 4. С. 27–50. doi: 10.21685/2072-3059-2019-4-3
11. Карамышева Н. С., Свищев Д. С., Попов К. В., Зинкин С. А. Реализация агентно-базированных метакомпьютерных систем и приложений // Известия Юго-Западного государственного университета. 2022. № 26 (1). С. 148–171. doi: 10.21869/2223-1560-2022-26-1-148-171
12. Zinkin S. A., Volchihin V. I., Karamysheva N. S., Jaafar M. S. Dynamic Topology Transformation of Cloud-Network Computer Systems: Conceptual Level // 2020 International Conference on Engineering Management of Communication and Technology, EMCTECH 2020. Proceedings, Austria, Vienna, 2020. P. 1–10. doi: 10.1109/EMCTECH49634.2020.9261554
13. CharGer Manual v3.5b1 2005-11-30, P. 1–58. URL: http://charger.sourceforge.net/ (дата обращения: 31.10.2022).
14. Delugah H. CharGer – A Conceptual Graph Editor written by Harry Delugah. URL: http://www.cs.uah.edu/~delugach/CharGer/ (дата обращения: 31.10.2022).
15. A suitable environment packed with various drawing tools and analysis modules to create thorough Petri nets and run simulations to test your projects PIPE2 4.3.0 for Windows. Review by Mircea Dragomir on February 2, 2015. URL: https://www.softpedia.com/get/Others/Home-Education/PIPE2.shtml (дата обращения: 31.10.2022).

References

1. Kravari K., Bassiliades N. A Survey of Agent Platforms. *Journal of Artificial Societies and Social Simulation*. 2015;18(11):1–18.
2. Cynthia N., Gregory M. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. *Journal of Artificial Societies and Social Simulation*. 2009;12(2). Available at: http://jasss.soc.surrey.ac.uk/12/2/2.html (accessed 02.09.2022).
3. Alluhaybi1 B., Alrahhal M.S., Alzhrani A., Thayananthan V. A Survey: Agent-based Software Technology Under the Eyes of Cyber Security, Security Controls, Attacks and Challenges. *International Journal of Advanced Computer Science and Applications (IJACSA)*. King Abdulaziz University (KAU), Jeddah, Saudi Arabia. 2019;10(8):211–230.
4. Silva L., Meneguzzi F., Logan B. BDI Agent Architectures: A Survey. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. Survey track*. 2020:4914–4921. doi: 10.24963/ijcai.2020/684
5. Bellifemine F.L., Caire G., Greenwood D. *Developing multi-agent systems with JADE*. Wiley, 2007:300.
6. *Java Agent Development Environment (JADE)*. Available at: http://jade.tilab.com/ (accessed 01.09.2022).

7. Mamikonov A.G., Kul'ba V.V. *Sintez optimal'nykh modul'nykh sistem obrabotki dannykh = Synthesis of optimal modular data processing systems*. Moscow: Nauka, 1986:276. (In Russ.)
8. Volchikhin V.I., Karamysheva N.S., Gorynina A.V., Zinkin S.A. Development of network agent-based applications based on metacomputer technology. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2021;(4):3–25. (In Russ.). doi: 10.21685/2072-3059-2021-4-1
9. Volchikhin V.I., Karamysheva N.S., Zinkin S.A., Gurin E.I. Algorithms, logic and modeling of agent-based metacomputer systems with a high level of parallelism. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2022;(2):5–25. (In Russ.). doi: 10.21685/2072-3059-2022-2-1
10. Volchikhin V.I., Zinkin S.A., Karamysheva N.S. Functioning organization of the cloud-network distributed computer systems with the architecture “agents as the services”. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2019;(4):27–50. (In Russ.). doi: 10.21685/2072-3059-2019-4-3
11. Karamysheva N.S., Svishchev D.S., Popov K.V., Zinkin S.A. Implementation of agent-based metacomputer systems and applications. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2022;(26):148–171. (In Russ.). doi: 10.21869/2223-1560-2022-26-1-148-171
12. Zinkin S.A., Volchihin V.I., Karamysheva N.S., Jaafar M.S. Dynamic Topology Transformation of Cloud-Network Computer Systems: Conceptual Level. *2020 International Conference on Engineering Management of Communication and Technology, EMCTECH 2020*. Proceedings, Austria, Vienna, 2020:1–10. doi: 10.1109/EMCTECH49634.2020.9261554
13. *CharGer Manual v3.5b1 2005-11-30, P. 1–58*. Available at: http://charger.sourceforge.net/ (accessed 31.10.2022).
14. Delugah H. *CharGer – A Conceptual Graph Editor written by Harry Delugah*. Available at: http://www.cs.uah.edu/~delugach/CharGer/ (accessed 31.10.2022).
15. *A suitable environment packed with various drawing tools and analysis modules to create thorough Petri nets and run simulations to test your projects PIPE2 4.3.0 for Windows*. Review by Mircea Dragomir on February 2, 2015. Available at: https://www.softpedia.com/get/Others/Home-Education/PIPE2.shtml (accessed 31.10.2022).

Информация об авторах / Information about the authors

Владимир Иванович Волчихин
доктор технических наук, профессор, президент Пензенского государственного университета (Россия, г. Пенза, ул. Красная, 40)

E-mail: president@pnzgu.ru

Vladimir I. Volchikhin
Doctor of engineering sciences, professor, president of Penza State University (40 Krasnaya street, Penza, Russia)

Надежда Сергеевна Карамышева
кандидат технических наук, доцент кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

Nadezhda S. Karamysheva
Candidate of engineering sciences, associate professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Данил Сергеевич Свищев
магистрант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)\

E-mail: vt@pnzgu.ru

Danil S. Svishchev
Master's degree student, Penza State University (40 Krasnaya street, Penza, Russia)

Сергей Александрович Зинкин
доктор технических наук, доцент, профессор кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: zsa49@yandex.ru

Sergey A. Zinkin
Doctor of engineering sciences, associate professor, professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию / Received 14.11.2022

Поступила после рецензирования и доработки / Revised 28.12.2022

Принята к публикации / Accepted 29.01.2023