УДК 004.9
DOI 10.21685/2072-3059-2019-4-3

В. И. Волчихин, С. А. Зинкин, Н. С. Карамышева

ОРГАНИЗАЦИЯ ФУНКЦИОНИРОВАНИЯ ОБЛАЧНО-СЕТЕВЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С АРХИТЕКТУРОЙ «АГЕНТЫ КАК СЕРВИСЫ»

Аннотация.

Актуальность и цели. На концептуальном уровне проектирования распределенной вычислительной системы могут быть предложены различные схемы взаимодействий клиентов с серверами, при реализации которых клиенты поразному обращаются к одному или к группе серверов. Актуальной является проблема организации подобных взаимодействий в облачно-сетевых распределенных вычислительных системах (ОС РВС) и переход от концептуального представления функциональной архитектуры к сетевым приложениям. Объектом исследования являются ОС РВС. Предметом исследования является методика синтеза функциональной архитектуры подобных систем. Целью исследования является решение актуальной задачи по расширению функциональных возможностей ОС РВС, объединяющих в себе свойства собственно облачных и грид-систем со свойствами мультиагентных систем. Особенностью предлагаемой новой функциональной архитектуры является то, что в аренду пользователю предоставляются мобильные агенты, выполняющие заданные функции.

Материалы и методы. Исследования выполнены на основе сетевой компьютерной интерпретации исполнимых логико-алгебраических моделей и аппарата логических сетей Петри.

Результаты. Предложена концепция построения ОС РВС, основанная на формализации перехода от облачной архитектуры «функция как сервис» (FaaS – Function-as-a-Service) к новой архитектуре «агент как сервис» (AaaS – Agent-as-a-Service). Предложена формализация функциональной архитектуры ОС РВС системами логико-алгебраических операционных выражений (ЛАОВ), относящихся к классу исполнимых моделей и пригодных для непосредственной программной реализации в сетевой компьютерной среде; дополнительным свойством ЛАОВ является возможность реконфигурации (модификации режима функционирования) результирующего сетевого программного обеспечения. Предложена методика отображения системы логико-алгебраических операционных выражений на архитектуру компьютерной сети, учитывающая регулярный характер модели.

Выводы. Методика синтеза облачных систем с новой архитектурой AaaS, основанная на формализованных логико-алгебраических спецификациях, позволяет ускорить создание программного обеспечения и расширить функциональные возможности рассматриваемых систем.

Ключевые слова: облачно-сетевые системы, агенты, сервисы, мультиагентные системы, модифицированные сети Петри, логико-алгебраические спецификации.

V. I. Volchikhin, S. A. Zinkin, N. S. Karamysheva

FUNCTIONING ORGANIZATION OF THE CLOUD-NETWORK DISTRIBUTED COMPUTER SYSTEMS WITH THE ARCHITECTURE "AGENTS AS THE SERVICES"

Abstract.

Background. At the conceptual level of designing a distributed computing system, there can be proposed various schemes of client interactions with servers, in the implementation of which clients differently access one or a group of servers. An urgent problem is the organization of such interactions in cloud-based distributed computing systems (CB DCS) and the transition from a conceptual representation of the functional architecture to network applications. The object of the study is the CB DCS. The subject of the research is the synthesis technique of the functional architecture of such systems. The aim of the study is to solve the urgent task of expanding the functionality of the CB DCS, combining the properties of the actual cloud and grid systems with the properties of multi-agent systems. A feature of the proposed new functional architecture is that mobile agents performing specified functions are provided for rent to the user.

Materials and methods. The studies were performed on the basis of a networked computer interpretation of executable logical-algebraic models and apparatus of logical Petri nets.

Results. The concept of building the CB DCS based on the formalization of the transition from the cloud-based architecture of FaaS – Function-as-a-Service to the new architecture AaaS – Agent-as-a-Service is proposed. The formalization of the functional architecture of the CB DCS based on the systems of logical-algebraic operational expressions (LAOE), belonging to the class of executable models and suitable for direct software implementation in a networked computer environment, is proposed; an additional property of LAOE is the possibility of reconfiguration (modification of the operating mode) of the resulting network software. A technique for mapping a system of logical-algebraic operational expressions onto a computer network architecture that takes into account the regular nature of the model is proposed.

Conclusions. The synthesis technique of cloud systems with the new AaaS architecture, based on formalized logical-algebraic specifications, allows you to speed up the creation of software and expands the functionality of the systems in question.

Keywords: cloud-based network systems, agents, services, multi-agent systems, modified Petri nets, logical-algebraic specifications.

Введение

В практике проектирования распределенных вычислительных систем (РВС) и сетевых приложений редко используются так называемые исполнимые модели, основанные на формализованных спецификациях системной и функциональной архитектуры распределенных систем. При проектировании РВС разработчики опираются в основном на свою интуицию и опыт предыдущих разработок. Целью настоящей работы является разработка методики синтеза приложений для распределенных виртуальных облачно-сетевых РВС (ОС РВС) с системной и функциональной архитектурой, соответствующей запросам пользователей. В основу методики положено использование исполнимых моделей, основанное агентно-ориентированном подходе, логико-

алгебраических формализованных спецификациях, получаемых на базе сетевой компьютерной интерпретации логических сетей Петри. Предложенная методика позволяет отобразить системы логико-алгебраических операционных выражений на архитектуру компьютерной сети.

1. Применение агентно-ориентированного подхода AaaS (Agent-as-a-Service) в облачно-сетевых средах типа SaaS (Software-as-a-Service) и FaaS (Function-as-a-Service)

Программное обеспечение как сервис (SaaS – Software-as-a-Service) предоставляет возможность пользователям подключаться к облачным приложениям и использовать их через Интернет [1, 2]. Облачный сервис SaaS позволяет организациям быстро запускать приложения с минимальными предварительными расходами. Агентно-ориентированный подход к построению облачно-сетевых вычислительных сред (AaaS – Agent-as-a-Service) может позволить пользователю не только арендовать, но и самому запускать в сеть приложения, существующие в форме взаимодействующих мобильных агентов.

Подход, развиваемый в настоящей работе, пригоден для реализации на базе многих существующих платформах мобильных агентов, управляющих функционированием агентов [3, 4]. Известен ряд инструментальных средств и языков для агентно-ориентированного программирования. Для реализации архитектуры ОС РВС типа AaaS большой интерес представляют агентные платформы, базирующиеся на кроссплатформенном языке Java [5–7]: Agent Factory, AgentBuilder, AgentScape, Aglets, AGLOBE, Cougaar, CybelePro, EMERALD, JACK, JADE, Jadex, Jason, JIAC, Jas, MASON, SeSAm, Swarm и др. В настоящей работе учитываются основные свойства открытых платформ мобильных агентов JADE и Aglets, хорошо зарекомендовавшие себя при создании сетевых, метакомпьютерных и других видов Internet-приложений [8–16].

Агентно-ориентированный подход в данной работе предлагается совместить с так называемым бессерверным подходом, обеспечивающим гибкость системы и малую связность компонентов, что позволяет расширять ее функциональные возможности за счет использования основных свойств и особенностей мобильных агентов.

Бессерверный подход (serverless), практически эквивалентный FaaS-подходу (Function-as-a-Service – функция как сервис) [17–21] основан на известной облачной модели, в которой платформа берет на себя функции распределения машинных ресурсов между клиентами. При традиционном подходе запрос «тонкого» клиента вызывает некоторые службы, размещенные на сервере. При новом подходе в подобных системах программными компонентами реализуются функции, которые исполняют запросы к платформе и получают доступ к ее ресурсам. Данная архитектура обеспечивает широкие возможности для удовлетворения требований клиентов, упрощает процессы адресации, анализа емкости хранилища и выделения вычислительных мощностей. Платформа облачной среды занимается настройкой сервера и распределением машинных ресурсов между собой и другими пользователями.

В работе [19] предлагается архитектура облачной системы NFaaS (Named-Function-as-a-Service – именованные функции как сервис), в которой в качестве сервисов используются именованные функции, которые могут быть загружены и запущены любым узлом в сети. Функции могут переме-

щаться между узлами в соответствии с требованиями пользователя. Компоненты хранилища, используемые в данной облачной системе, отвечают не только за хранение функций, но и за принятие решений о том, какие функции запускать. Архитектура NFaaS построена на основе протокола маршрутизации и использует ряд стратегий пересылки для развертывания и динамической миграции функций внутри сети. Однако при реализации данной платформы не используются агентно-ориентированные технологии, применение которых поможет упростить управление процессами поиска и выполнения функций.

Облачная модель FaaS функционирует следующим образом [21]:

1. Клиент делает запрос к серверной вычислительной платформе для выполнения определенной функции.

2. Серверная вычислительная платформа проверяет, работает ли данная функция на каком-либо из ее серверов. Если функция еще не инициализирована, то платформа загружает ее из хранилища данных.

3. Платформа разворачивает функцию на одном из своих серверов, которые предварительно сконфигурированы со средой исполнения.

4. Выполняется функция и фиксируется результат.

5. Результат возвращается клиенту.

Формально архитектура облачно-сетевых сервисов типа AaaS определяется отношением, являющимся областью истинности бинарного предиката

$$U_{AaaS}: F \times A \rightarrow \{\textbf{true, false}\},$$

где F – множество функций-сервисов; A – множество мобильных агентов, реализующих эти сервисы.

Изменяемая архитектура AaaS фиксируется путем выполнения множества правил модификации предиката U_{AaaS} следующего вида:

$$U_{AaaS}(F_i, a_j) \leftarrow b,$$

где b – булев терм; $F_i \in F$, $a_j \in A$. При b = **true** функция F_i связывается с агентом a_j, а при b = **false** эта связь не реализуется или разрывается. Если принято, что каждая функция реализуется одним агентом, то отношение связности должно иметь функциональный характер. Один и тот же агент может реализовывать несколько функций. Рисунок 1 иллюстрирует закрепление функций за агентами в мультиагентной системе, реализующей архитектуру ОС РВС типа AaaS. Здесь A_0 – агент-менеджер, в функции которого входит создание клонов остальных «облачных» агентов и наделение их требуемой функциональностью. Минимальные требования к агентам – реализация требуемых функций и способность выполнять операции, проиллюстрированные рис. 2.

Способы конкретной реализации процессов в мультиагентных системах описаны в работах [8, 9, 16] и характерны для многих мультиагентных платформ. В данной работе предполагается, что в мультиагентной системе выполняются операции создания копий (клонов) агентов, обмена сообщениями между агентами, перемещения агентов на новый узел сети, обмена данными и их копиями.

Размещение агентов и организацию связей между агентами и узлами в вычислительной сети иллюстрирует рис. 3.

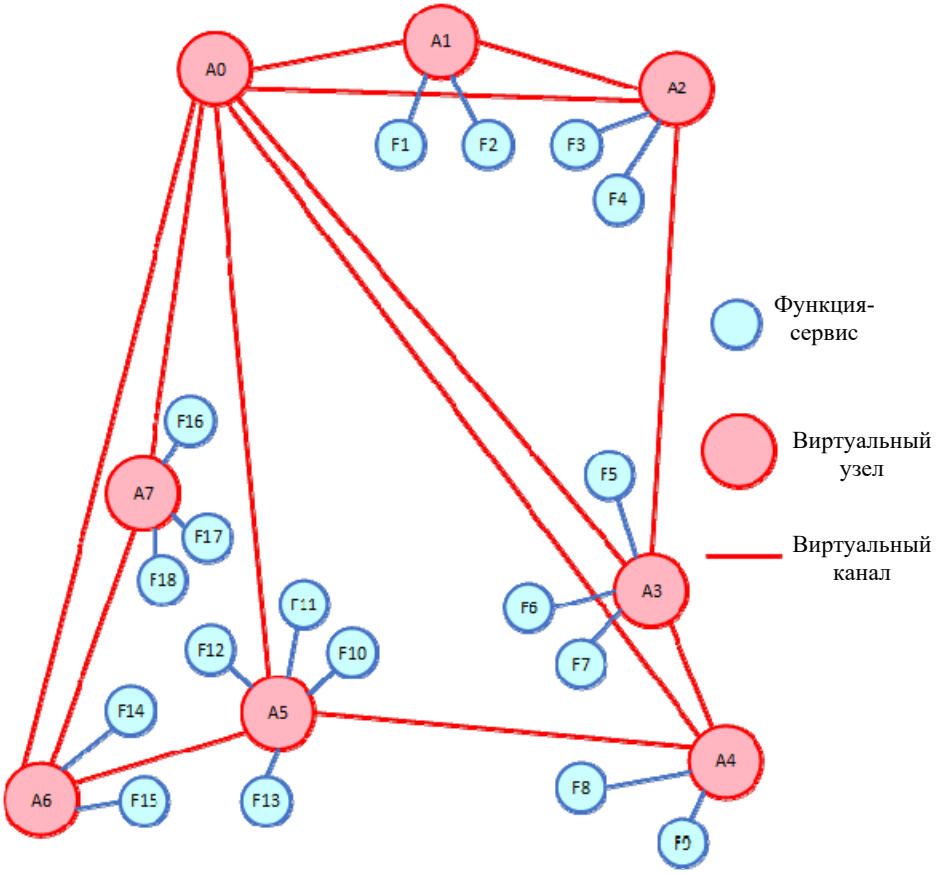

Рис. 1. Многофункциональная мультиагентная система, реализующая архитектуру AaaS

Виртуальные связи в реальной компьютерной TCP/IP сети на сетевом уровне реализуются при помощи коммутаторов и маршрутизаторов. Агенты и сообщения в данной системе могут перемещаться по произвольным маршрутам. Например, агент A_0, размещенный в узле 2, может передавать свои клоны и сообщения по кольцу физических узлов <3, 5, 7, 22, 26, 20, 17, 12> или по хордам <3, 5>, <3, 7>, <3, 22>, <3, 26>, <3, 20>, <3, 17>, <3, 12>.

2. Абстрактный и структурный синтез функциональной архитектуры распределенных вычислительных систем на основе логико-алгебраической интерпретации реконфигурируемых сетей Петри

По существу, для организации ОС РВС с архитектурой AaaS «агент как сервис» достаточно использовать архитектуры «клиент–сервер» или «клиент–агент–сервер». Мультиагентные платформы возможно также реализовать с применением пиринговых технологий P2P (peer-to-peer) [22], используемых также в файлообменных системах. Архитектура P2P похожа на архитектуры «клиент–сервер» и «клиент–агент–сервер» с тем различием, что размещенный на узле вычислительной сети программный модуль или агент может поочередно выполнять функции как клиента, так и сервера. В этой связи далее рассматривается сетевая архитектура ОС РВС, включающей в свой состав

клиентскую (возможно, беспроводную и мобильную) сеть и TCP/IP сеть – облачный фрагмент Internet Cloud сети Internet. На концептуальном уровне проектирования могут быть предложены различные схемы взаимодействий клиентов с серверами, при реализации которых клиенты по-разному обращаются к одному или к группе серверов при реализации обменов функциями согласно рис. 2 и 3. На рис. 4 представлена упрощенная концептуальная модель взаимодействий нескольких агентов-клиентов C_i (i = 1, …, m) с несколькими агентами-серверами R_j (j = 1, …, n) в многосерверной системе «клиент–агент–сервер». Здесь принято, что данная архитектура доступа агентов-клиентов к агентам-серверам, далее называемым просто клиентами и серверами, реализуется через составную TCP/IP сеть.

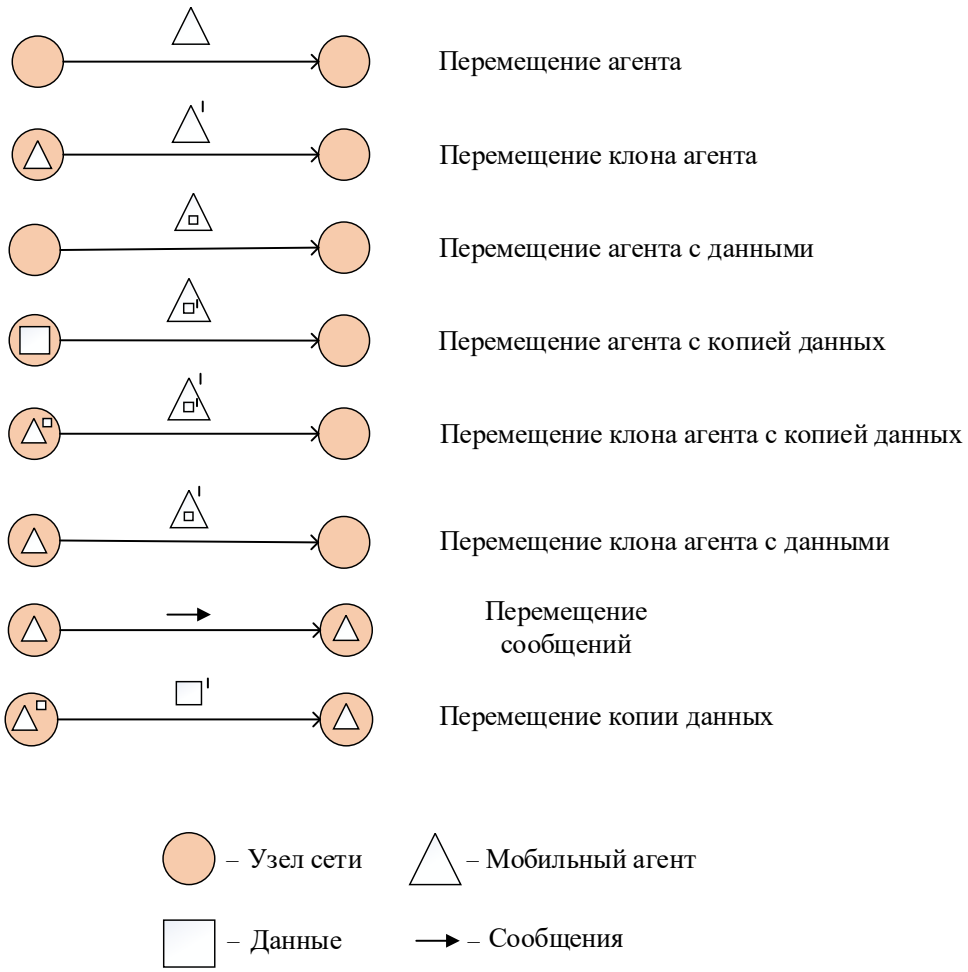

Рис. 2. Разновидности способов перемещения агентов и данных в агентно-ориентированных сетях

Рассматриваемая далее методика абстрактного и структурного синтеза функциональной архитектуры ОС РВС на основе логико-алгебраической интерпретации реконфигурируемых сетей Петри для ряда других сетевых систем была ранее предложена в работах [23–27] с тем существенным отличи-

ем, что согласно новой методике формализованы сетевые обмены сообщениями и агентами. Поэтому определенные далее формализмы имеют более широкую область применения, чем описанные в отмеченных работах.

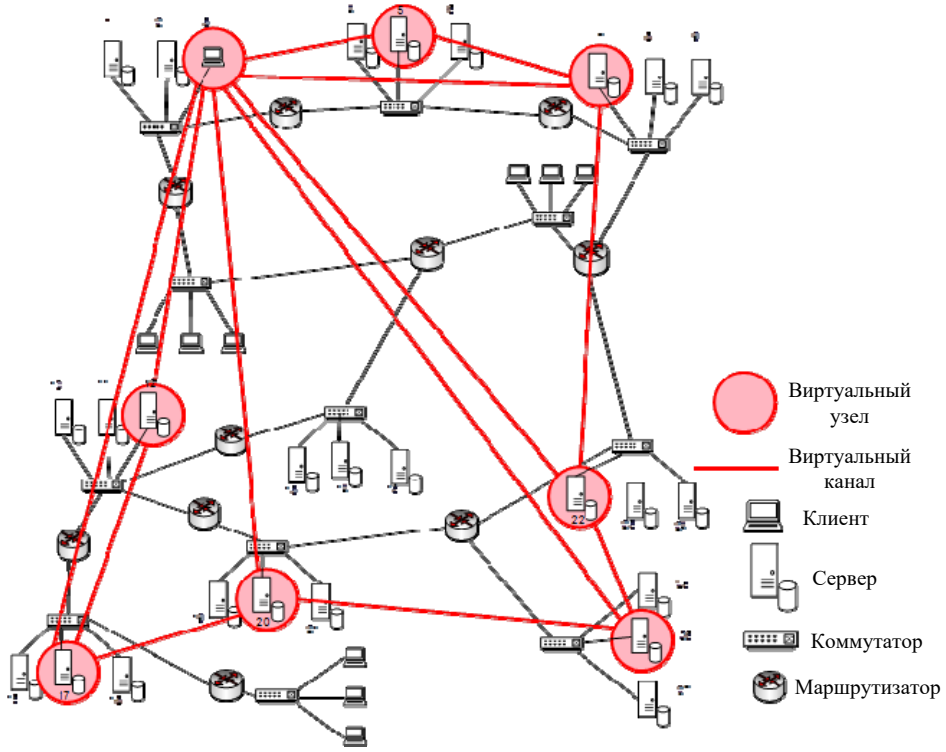

Рис. 3. Организация ОС РВС путем вложения мультиагентной системы в архитектуру глобальной вычислительной сети

В настоящей работе используются логические, или бинарные, сети Петри, эквивалентные по графическому представлению безопасным сетям Петри [28–31]. Такие сети использовались в работе [32] для описания интерфейса передачи сообщений в сетях, а в работе [33] – для описания работы мобильных агентов в метакомпьютерной среде для управления распределенной базой данных.

Возможна организация различных алгоритмов взаимодействий клиентов с серверами, при реализации которых учитываются различные критерии поиска информации, блокировки и разблокировки серверов. Особенностью большинства работ по проектированию РВС является использование в основном содержательных описаний моделей функционирования и рутинного программирования на основе практического опыта разработчиков.

Реконфигурируемые поведенческие исполнимые модели при их программной реализации позволяют оперативно изменять режим работы распределенных вычислительных систем. Для данных систем это имеет первостепенное значение, так как клиенту предоставляется возможность изменения режима функционирования «заказанной» архитектуры в соответствии с изменяющимися условиями решения задач. В целях иллюстрации возможностей использования реконфигурируемых исполнимых моделей предлагаются для

рассмотрения две схемы (*схема 1* и *схема 2*) клиент-серверных взаимодействий при выполнении запросов клиентов. Согласно технологиям FaaS и AaaS функции и агенты, как и сообщения, также могут перемещаться между узлами в соответствии с требованиями пользователя.

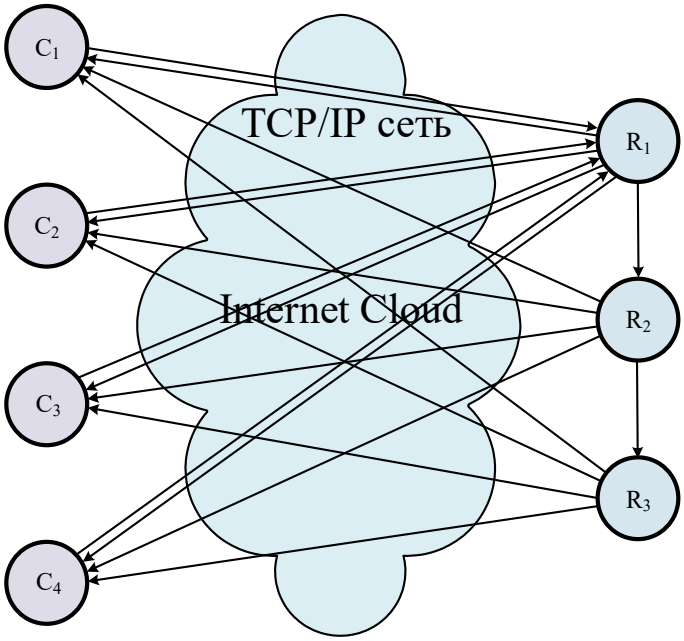

Рис. 4. Концептуальная модель взаимодействий
в многосерверной системе «агент–клиент–сервер»

При работе по *схеме 1* клиент C пытается отыскать свободный сервер из серверов R_1, R_2 и R_3. Для этого он посылает сообщение, последовательно обходящее серверы. Если будет найден свободный сервер, то этот сервер посылает ответ клиенту. Данное сообщение будет циркулировать в сети Internet Cloud, переходя от одного сервера к другому до тех пор, пока не будет найден свободный сервер. Дальнейшее описание целесообразно проиллюстрировать моделью, построенной на основе модификации безопасной сети Петри с дополнительными ингибиторными и информационными дугами и внедренными в ЛАОВ формализованными сетевыми примитивами, соответствующими передачам сообщений в компьютерной сети.

Поведенческая модель $M_1(C, R_1, R_2, R_3)$ взаимодействий по *схеме 1* в системе «клиент–серверы» для одного клиента и трех серверов представлена на рис. 5. Здесь модель клиентского приложения C представлена фрагментом бинарной сети Петри, содержащим позиции a_1, a_2, a_3, a_{10} и переходы at_1, at_2, at_3. Переход at_1 в модели выполняет функцию передачи сообщения, которое самостоятельно отыскивает свободный сервер. Серверные приложения на рис. 5 представлены следующими позициями и переходами: сервер R_1 представлен в модели позициями a_4, a_7, r_1 и переходами at_4, at_7, at_8, причем позиция r_1 характеризует состояние ресурса R_1: незанятость ресурса при наличии метки в позиции r_1 ($M(r_1)$ = **true**) и его занятость, т.е. недоступность клиенту, при отсутствии метки в позиции r_1 ($M(r_1)$ = **false**). Здесь M – унарный преди-

кат текущей маркировки позиции r_1. Аналогично, сервер R_2 представлен в модели позициями a_5, a_8, r_2 и переходами at_5, at_9, at_{10}, а сервер R_3 представлен в модели позициями a_6, a_9, r_3 и переходами at_6, at_{11}, at_{12}.

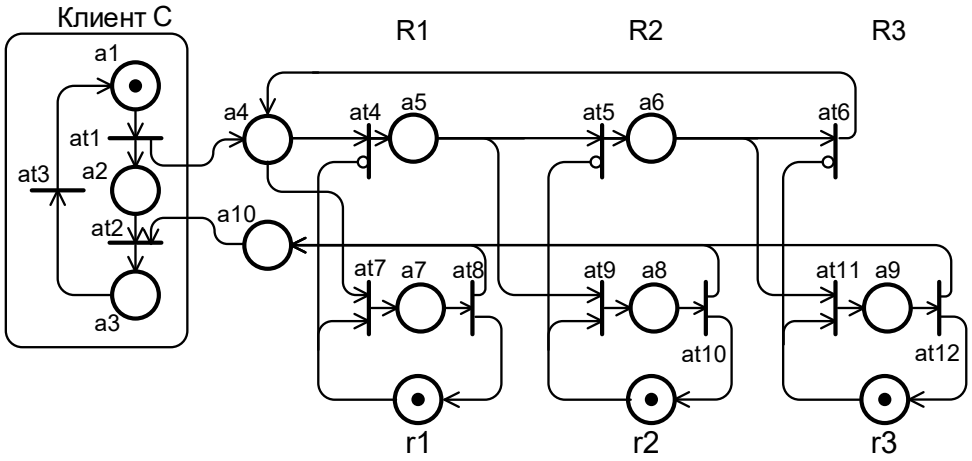

Рис. 5. Поведенческая модель $M1(C, R1, R2, R3)$
взаимодействий в системе «клиент–серверы»

Позиции и переходы в бинарной сети Петри связаны простыми и ингибиторными дугами. Простая дуга разрешает срабатывание перехода, когда в его входной позиции есть метка, а ингибиторная дуга (стрелка с кружком на конце), наоборот, при наличии метки во входной позиции запрещает срабатывание перехода.

Система Σ_1 логико-алгебраических операционных выражений (ЛАОВ) для сети, представленной на рис. 5, имеет следующий вид:

$At_1 = [P(a_1)](\{P(a_1) \leftarrow \mathbf{false}, P(a_2) \leftarrow \mathbf{true}, P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_2 = [P(a_2)\ \&\ P(a_{10})](\{P(a_2) \leftarrow \mathbf{false}, P(a_{10}) \leftarrow \mathbf{false}, P(a_3) \leftarrow \mathbf{true}\} \vee E)$;
$At_3 = [P(a_3)](\{P(a_3) \leftarrow \mathbf{false}, P(a_1) \leftarrow \mathbf{true}\} \vee E)$;
$At_4 = [P(a_4)\ \&\ \neg P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(a_5) \leftarrow \mathbf{true}\} \vee E)$;
$At_5 = [P(a_5)\ \&\ \neg P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(a_6) \leftarrow \mathbf{true}\} \vee E)$;
$At_6 = [P(a_6)\ \&\ \neg P(r_3)](\{P(a_6) \leftarrow \mathbf{false}, P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_7 = [P(a_4)\ \&\ P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{false}, P(a_7) \leftarrow \mathbf{true}\} \vee E)$;
$At_8 = [P(a_7)](\{P(a_7) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_9 = [P(a_5)\ \&\ P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{false}, P(a_8) \leftarrow \mathbf{true}\} \vee E)$;
$At_{10} = [P(a_8)](\{P(a_8) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_{11} = [P(a_6)\ \&\ P(r_3)](\{P(a_6) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{false}, P(a_9) \leftarrow \mathbf{true}\} \vee E)$;
$At_{12} = [P(a_9)](\{P(a_9) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$.

Здесь и далее используется унарный предикат, или функция разметки позиций, P: $A \cup R \rightarrow \{\mathbf{true}, \mathbf{false}\}$, где A – множество «рабочих» позиций, а R – множество ресурсных позиций, $A \cap R = \varnothing$. Система Σ_1 ЛАОВ содержит продукционные правила срабатывания переходов бинарной сети Петри, представленные для удобства программирования исполнимой модели в целях со-

здания реального сетевого приложения в виде альфа-дизъюнкций Глушкова: нотация $[\alpha](B \vee D)$ означает, что в зависимости от истинности или ложности замкнутого (без свободных предметных переменных) логического выражения α выполняются действия B или D. В фигурные скобки в выражениях для системы Σ_1 заключены операции модификации предиката P. Символ E обозначает пустое действие при ложности условия в квадратных скобках.

При работе по *схеме 2* клиент так же, как и при работе по *схеме 1*, пытается отыскать свободный сервер из серверов R_1, R_2 и R_3. Отличие заключается в том, что в случае, когда все серверы оказались заняты, сообщение возвращается к клиенту. Модифицированная поведенческая модель $M_2(C, R_1, R_2, R_3)$, описывающая работу системы по *схеме 2*, представлена на рис. 6. Продукционные правила срабатывания At_1*, At_2* и At_6* для переходов t_1, t_2 и t_6 изменятся (здесь и в приведенной ниже системе ЛАОВ Σ_2 измененные правила отмечены звездочками).

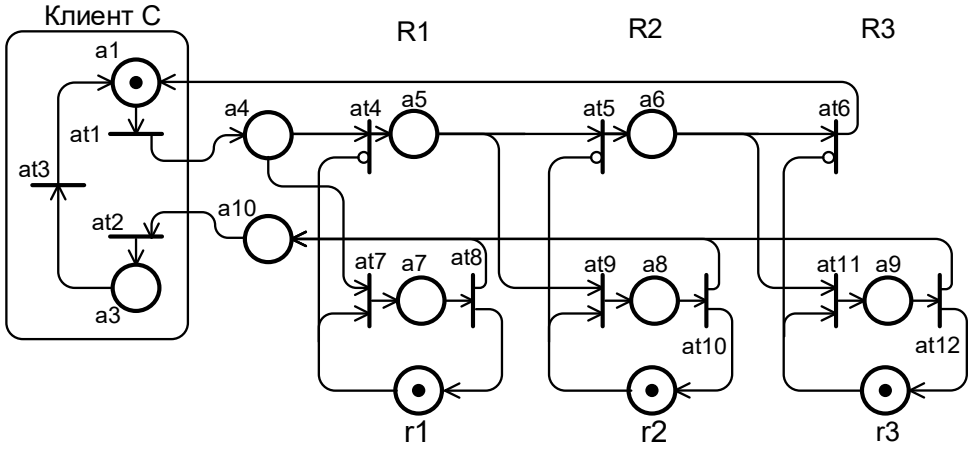

Рис. 6. Модифицированная поведенческая модель $M_2(C, R_1, R_2, R_3)$ взаимодействий в системе «клиент–серверы»

Система Σ_2 логико-алгебраических операционных выражений для сети $M_2(C, R_1, R_2, R_3)$, представленной на рис. 6, примет следующий вид:

$At_1^* = [P(a_1)](\{P(a_1) \leftarrow \mathbf{false}, P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_2^* = [P(a_{10})](\{P(a_{10}) \leftarrow \mathbf{false}, P(a_3) \leftarrow \mathbf{true}\} \vee E)$;
$At_3 = [P(a_3)](\{P(a_3) \leftarrow \mathbf{false}, P(a_1) \leftarrow \mathbf{true}\} \vee E)$;
$At_4 = [P(a_4) \,\&\, \neg P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(a_5) \leftarrow \mathbf{true}\} \vee E)$;
$At_5 = [P(a_5) \,\&\, \neg P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(a_6) \leftarrow \mathbf{true}\} \vee E)$;
$At_6^* = [P(a_6) \,\&\, \neg P(r_3)](\{P(a_6) \leftarrow \mathbf{false}, P(a_1) \leftarrow \mathbf{true}\} \vee E)$;
$At_7 = [P(a_4) \,\&\, P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{false}, P(a_7) \leftarrow \mathbf{true}\} \vee E)$;
$At_8 = [P(a_7)](\{P(a_7) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_9 = [P(a_5) \,\&\, P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{false}, P(a_8) \leftarrow \mathbf{true}\} \vee E)$;
$At_{10} = [P(a_8)](\{P(a_8) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_{11} = [P(a_6) \,\&\, P(r_3)](\{P(a_6) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{false}, P(a_9) \leftarrow \mathbf{true}\} \vee E)$;
$At_{12} = [P(a_9)](\{P(a_9) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$.

Объединяя две системы Σ_1 и Σ_2, получим следующую систему выражений для новой системы ЛАОВ $\Sigma_{2\text{-}3}$:

$At_1 = [\neg S(z)\ \&\ P(a_1)](\{P(a_1) \leftarrow \mathbf{false}, P(a_2) \leftarrow \mathbf{true}, P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_1^* = [S(z)\ \&\ P(a_1)](\{P(a_1) \leftarrow \mathbf{false}, P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_2 = [\neg S(z)\ \&\ P(a_2)\ \&\ P(a_{10})](\{P(a_2) \leftarrow \mathbf{false}, P(a_{10}) \leftarrow \mathbf{false},$
$P(a_3) \leftarrow \mathbf{true}\} \vee E)$;
$At_2^* = [\neg S(z)\ \&\ P(a_{10})](\{\ P(a_{10}) \leftarrow \mathbf{false}, P(a_3) \leftarrow \mathbf{true}\} \vee E)$;
$At_3 = [P(a_3)](\{P(a_3) \leftarrow \mathbf{false}, P(a_1) \leftarrow \mathbf{true}\} \vee E)$;
$At_4 = [P(a_4)\ \&\ \neg P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(a_5) \leftarrow \mathbf{true}\} \vee E)$;
$At_5 = [P(a_5)\ \&\ \neg P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(a_6) \leftarrow \mathbf{true}\} \vee E)$;
$At_6 = [\neg S(z)\ \&\ P(a_6)\ \&\ \neg P(r_3)](\{P(a_6) \leftarrow \mathbf{false},$
$P(a_4) \leftarrow \mathbf{true}\} \vee E)$;
$At_6^* = [S(z)\ \&\ P(a_6)\ \&\ \neg P(r_3)](\{P(a_6) \leftarrow \mathbf{false},$
$P(a_1) \leftarrow \mathbf{true}\} \vee E)$;
$At_7 = [P(a_4)\ \&\ P(r_1)](\{P(a_4) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{false}, P(a_7) \leftarrow \mathbf{true}\} \vee E)$;
$At_8 = [P(a_7)](\{P(a_7) \leftarrow \mathbf{false}, P(r_1) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_9 = [P(a_5)\ \&\ P(r_2)](\{P(a_5) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{false}, P(a_8) \leftarrow \mathbf{true}\} \vee E)$;
$At_{10} = [P(a_8)](\{P(a_8) \leftarrow \mathbf{false}, P(r_2) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$;
$At_{11} = [P(a_6)\ \&\ P(r_3)](\{P(a_6) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{false}, P(a_9) \leftarrow \mathbf{true}\} \vee E)$;
$At_{12} = [P(a_9)](\{P(a_9) \leftarrow \mathbf{false}, P(r_3) \leftarrow \mathbf{true}, P(a_{10}) \leftarrow \mathbf{true}\} \vee E)$.

Для настройки системы на работу по *схеме 1* или по *схеме 2* введен предикат S и предметная константа z. При ложности высказывания $S(z)$ система настраивается на работу по *схеме 1*, а при его истинности – по *схеме 2*.

3. Логико-алгебраические спецификации с формализованными расширениями для представления передач сообщений и агентов в облачно-сетевых вычислительных системах

Приведенные ранее модели $M_1(C, R_1, R_2, R_3)$ и $M_2(C, R_1, R_2, R_3)$ являются неполными в том плане, что в них не учтены сетевые обмены, протекающие в среде вычислительной сети Internet Cloud. Поэтому целесообразно рассмотреть полные модели, в которых учитываются сетевые взаимодействия между клиентами и серверами. На рис. 7 представлена полная сетевая модель $M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$, описывающая доступ четырех клиентов C_1, C_2, C_3 и C_4 к трем серверам R_1, R_2 и R_3 по *схеме 1*, а на рис. 8 представлена полная сетевая модель $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$, описывающая доступ такого же числа клиентов к серверам по *схеме 2*.

Сетевые модели, представленные на данных рисунках, имеют регулярную структуру и хорошо масштабируются, что упрощает программную интерпретацию моделей и последующую реализацию сетевых приложений.

Сетевые спецификации представлены логико-алгебраическими операционными выражениями, построенными на основе правил срабатывания переходов в бинарных сетях Петри. Вводятся унарные предикаты, одноименные клиентам и серверам: C_1, C_2, C_3, C_4, R_1, R_2, R_3. Эти предикаты представляют собой функции разметки позиций. В качестве значений предметных переменных используются предметные константы, имена которых совпадают с име-

нами позиций, используемых в моделях $M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ и $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$.

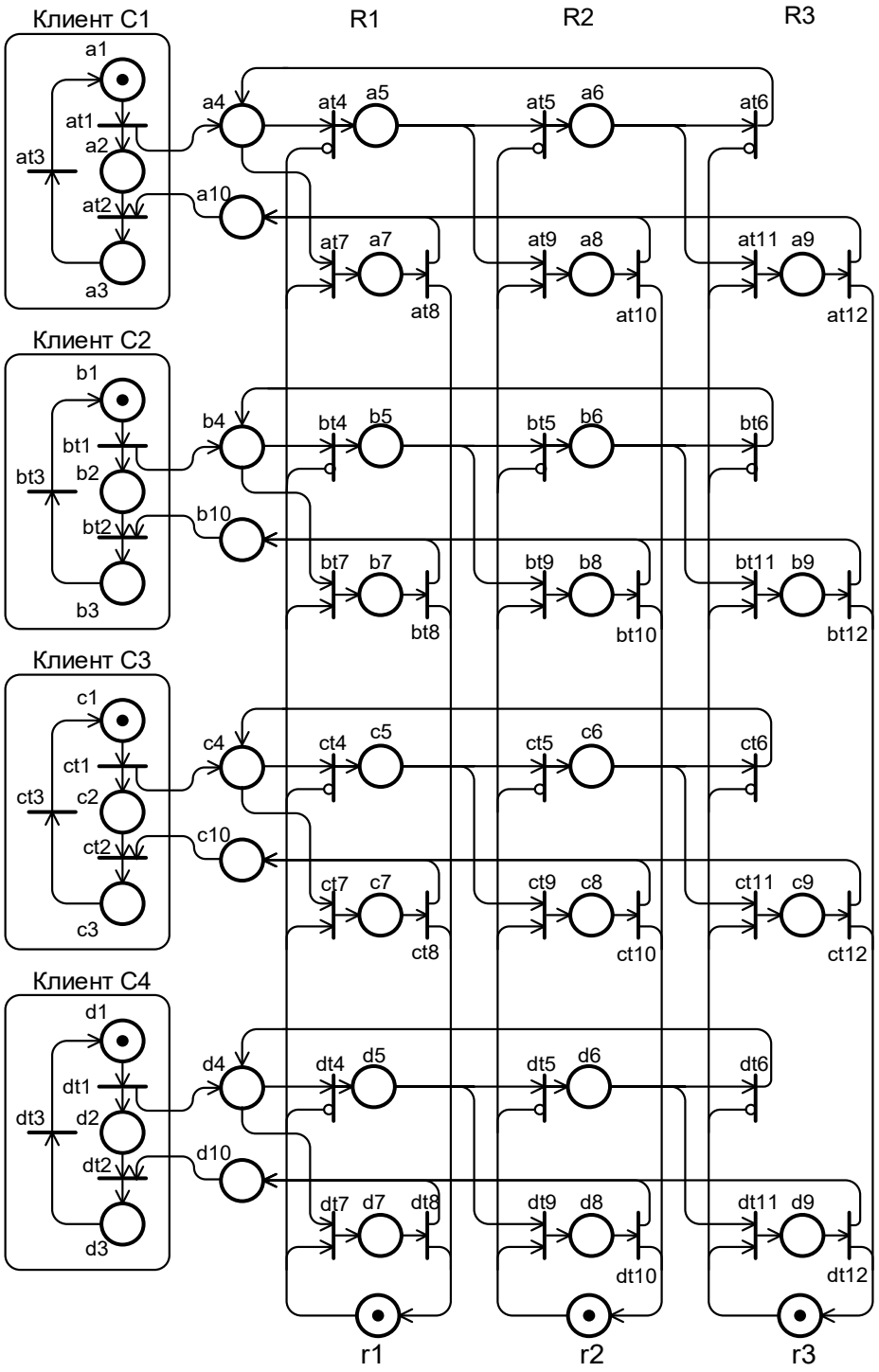

Рис. 7. Сетевая модель $M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$

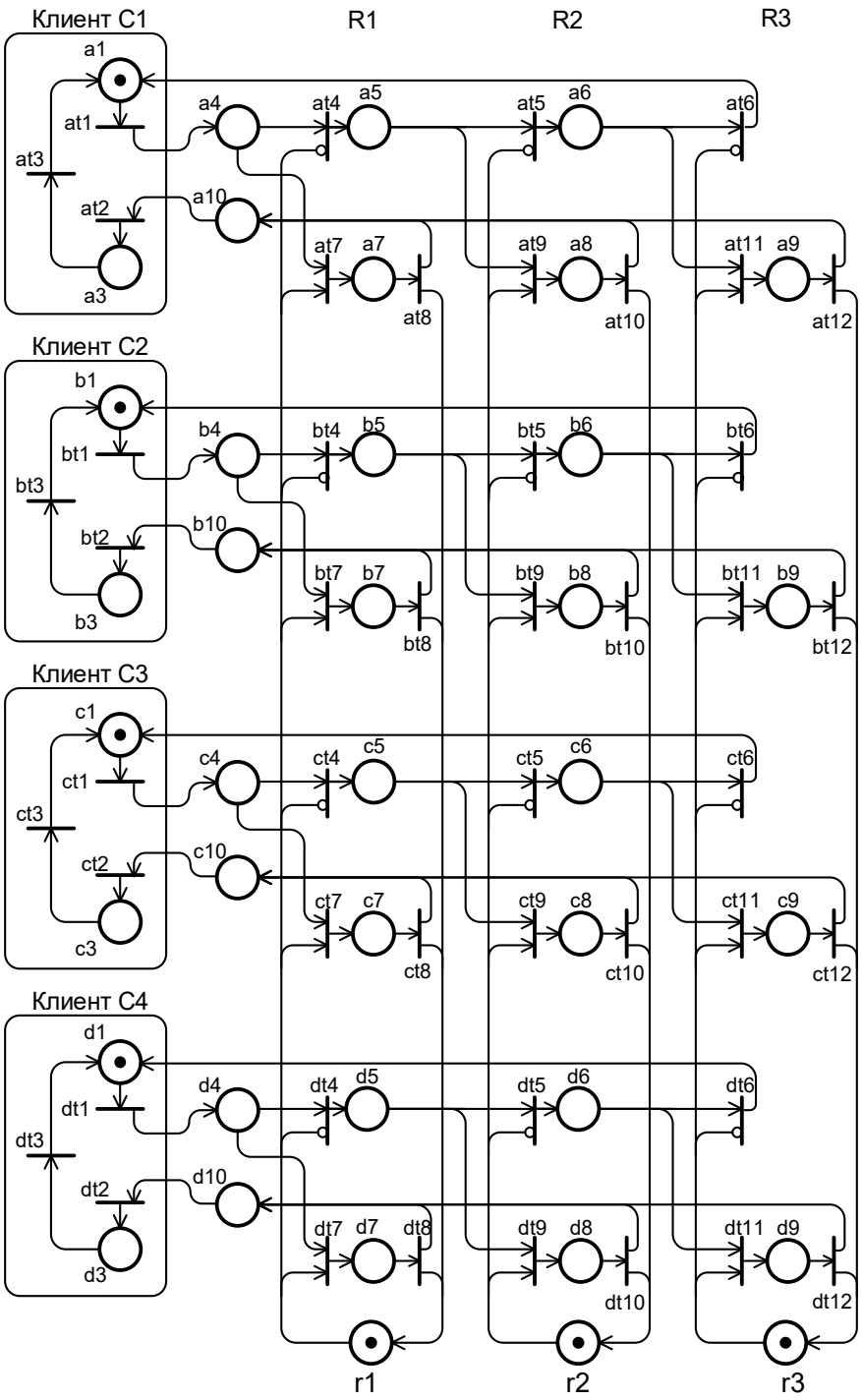

Рис. 8. Сетевая модель $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$

Совмещенные сетевые формализованные спецификации Σ_{4x3} для моделей, обеспечивающих использование приложений в составе четырех агентов-клиентов и ресурсов трех агентов-серверов, соответствующие как модели

$M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ при $S(z) =$ **false**, так и модели $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ при $S(z) =$ **true**, представлены в следующем виде:

– сетевые спецификации для модулей, реализуемых приложением-клиентом C_1:

$At_1/C_1 = [\neg S(z)\ \&\ Start\ \&\ C_1(a_1)](\{Send(C_1, R_1, M_{C1,R1}) \leftarrow$ **true** $\#\ R_1(a_4) \leftarrow$ **true**,
$C_1(a_1) \leftarrow$ **false**, $C_1(a_2) \leftarrow$ **true**$\} \vee E)$;
$At_1{*}/C_1 = [S(z)\ \&\ Start\ \&\ C_1(a_1)](\{Send(C_1, R_1, M_{C1,R1}) \leftarrow$ **true** $\#\ R_1(a_4) \leftarrow$ **true**,
$C_1(a_1) \leftarrow$ **false**$\} \vee E)$;
$At_2/C_1 = [\neg S(z)\ \&\ (Send(R_1, C_1, M_{R1,C1}) \vee Send(R_2, C_1, M_{R2,C1}) \vee Send(R_3, C_1, M_{R3,C1}))\ \&\ C_1(a_2)\ \&\ C_1(a_{10})](\{Send(R_1, C_1, M_{R1,C1}) \leftarrow$ **false**, $Send(R_2, C_1, M_{R2,C1}) \leftarrow$ **false**, $Send(R_3, C_1, M_{R3,C1}) \leftarrow$ **false**, $C_1(a_2) \leftarrow$ **false**, $C_1(a_{10}) \leftarrow$ **false**, $C_1(a_3) \leftarrow$ **true**$\} \vee E)$;
$At_2{*}/C_1 = [S(z)\ \&\ (Send(R_1, C_1, M_{R1,C1}) \vee Send(R_2, C_1, M_{R2,C1}) \vee Send(R_3, C_1, M_{R3,C1}))\ \&\ C_1(a_{10})](\{Send(R_1, C_1, M_{R1,C1}) \leftarrow$ **false**, $Send(R_2, C_1, M_{R2,C1}) \leftarrow$ **false**, $Send(R_3, C_1, M_{R3,C1}) \leftarrow$ **false**, $C_1(a_{10}) \leftarrow$ **false**, $C_1(a_3) \leftarrow$ **true**$\} \vee E)$;
$At_3/C_1 = [C_1(a_3)](\{C_1(a_3) \leftarrow$ **false**, $C_1(a_1) \leftarrow$ **true**$\} \vee Ret_{\tau,n}(At_1/C_1))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_1 для приложения-клиента C_1:

$At_4/R_1 = [Send(C_1, R_1, M_{C1,R1})\ \&\ R_1(a_4)\ \&\ \neg R_1(r_1)]$
$(\{Send(C_1, R_1, M_{C1,R1}) \leftarrow$ **false**, $R_1(a_4) \leftarrow$ **false**, $Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **true** $\#$
$R_2(a_5) \leftarrow$ **true**$\} \vee E)$;
$At_7/R_1 = [Send(C_1, R_1, M_{C1,R1})\ \&\ R_1(a_4)\ \&\ R_1(r_1)]$
$(\{Send(C_1, R_1, M_{C1,R1}) \leftarrow$ **false**, $R_1(a_4) \leftarrow$ **false**, $R_1(a_7) \leftarrow$ **true**,
$R_1(r_1) \leftarrow$ **false**$\} \vee E)$;
$At_8/R_1 = [R_1(a_7)](\{R_1(a_7) \leftarrow$ **false**, $R_1(r_1) \leftarrow$ **true**,
$Send(R_1, C_1, M_{R1,C1}) \leftarrow$ **true** $\#\ C_1(a_{10}) \leftarrow$ **true**$\} \vee E)$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_2 для приложения-клиента C_1:

$At_5/R_2 = [Send(R_1, R_2, M_{R1,R2})\ \&\ R_2(a_5)\ \&\ \neg R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **false**, $R_2(a_5) \leftarrow$ **false**, $Send(R_2, R_3, M_{R2,R3}) \leftarrow$ **true #**
$R_3(a_6) \leftarrow$ **true**$\} \vee E)$;
$At_9/R_2 = [Send(R_1, R_2, M_{R1,R2})\ \&\ R_2(a_5)\ \&\ R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **false**, $R_2(a_5) \leftarrow$ **false**, $R_2(a_8) \leftarrow$ **true**,
$R_2(r_2) \leftarrow$ **false**$\} \vee E)$;
$At_{10}/R_2 = [R_2(a_8)](\{R_2(a_8) \leftarrow$ **false**, $R_2(r_2) \leftarrow$ **true**,
$Send(R_2, C_1, M_{R2,C1}) \leftarrow$ **true** $\#\ C_1(a_{10}) \leftarrow$ **true**$\} \vee E)$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_3 для приложения-клиента C_1:

$At_6/R_3 = [\neg S(z)\ \&\ Send(R_2, R_3, M_{R2,R3})\ \&\ R_3(a_6)\ \&\ \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow$ **false**, $R_3(a_6) \leftarrow$ **false**, $Send(R_3, R_1, M_{R3,R1}) \leftarrow$ **true** $\#$

$R_1(a_4) \leftarrow$ **true**$\} \vee E$);
At_6*/$R_3 = [S(z)$ & $Send(R_2, R_3, M_{R2,R3})$ & $R_3(a_6)$ & $\neg R_3(r_3)]$
($\{Send(R_2, R_3, M_{R2,R3}) \leftarrow$ **false**, $R_3(a_6) \leftarrow$ **false**, $Send(R_3, C_1, M_{R3,C1}) \leftarrow$ **true** #
$C_1(a_1) \leftarrow$ **true**$\} \vee E$);
$At_{11}/R_3 = [Send(R_2, R_3, M_{R2,R3})$ & $R_3(a_6)$ & $R_3(r_3)]$
($\{Send(R_2, R_3, M_{R2,R3}) \leftarrow$ **false**, $R_3(a_6) \leftarrow$ **false**, $R_3(a_9) \leftarrow$ **true**,
$R_3(r_3) \leftarrow$ **false**$\} \vee E$);
$At_{12}/R_3 = [R_3(a_9)](\{R_3(a_9) \leftarrow$ **false**, $R_3(r_3) \leftarrow$ **true**,
$Send(R_3, C_1, M_{R3,C1}) \leftarrow$ **true** # $C_1(a_{10}) \leftarrow$ **true**$\} \vee E$);

– сетевые спецификации для модулей, реализуемых приложением-клиентом C_2:

$Bt_1/C_2 = [\neg S(z)$ & $Start$ & $C_2(b_1)](\{Send(C_2, R_1, M_{C2,R1}) \leftarrow$ **true** # $R_1(b_4) \leftarrow$ **true**,
$C_2(b_1) \leftarrow$ **false**, $C_2(b_2) \leftarrow$ **true**$\} \vee E$);
Bt_1*/$C_2 = [S(z)$ & $Start$ & $C_2(b_1)](\{Send(C_2, R_1, M_{C2,R1}) \leftarrow$ **true** # $R_1(b_4) \leftarrow$ **true**,
$C_2(b_1) \leftarrow$ **false**$\} \vee E$);
$Bt_2/C_2 = [\neg S(z)$ & $(Send(R_1, C_2, M_{R1,C2}) \vee Send(R_2, C_2, M_{R2,C2}) \vee Send(R_3, C_2, M_{R3,C2}))$ & $C_2(b_2)$& $C_2(b_{10})](\{Send(R_1, C_2, M_{R1,C2}) \leftarrow$ **false**, $Send(R_2, C_2, M_{R2,C2}) \leftarrow$ **false**,
$Send(R_3, C_2, M_{R3,C2}) \leftarrow$ **false**, $C_2(b_2) \leftarrow$ **false**, $C_2(b_{10}) \leftarrow$ **false**, $C_2(b_3) \leftarrow$ **true**$\} \vee E$);
Bt_2*/$C_2 = [\neg S(z)$ & $(Send(R_1, C_2, M_{R1,C2}) \vee Send(R_2, C_2, M_{R2,C2}) \vee Send(R_3, C_2, M_{R3,C2}))$ & $C_2(b_{10})](\{Send(R_1, C_2, M_{R1,C2}) \leftarrow$ **false**, $Send(R_2, C_2, M_{R2,C2}) \leftarrow$ **false**, $Send(R_3, C_2, M_{R3,C2}) \leftarrow$ **false**, $C_2(b_{10}) \leftarrow$ **false**, $C_2(b_3) \leftarrow$ **true**$\} \vee E$);
$Bt_3/C_2 = [C_2(b_3)](\{C_2(b_3) \leftarrow$ **false**, $C_2(b_1) \leftarrow$ **true**$\} \vee Ret_{\tau,n}(Bt_1/C_2))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_1 для приложения-клиента C_2:

$Bt_4/R_1 = [Send(C_2, R_1, M_{C2,R1})$ & $R_1(b_4)$ & $\neg R_1(r_1)]$
($\{Send(C_2, R_1, M_{C2,R1}) \leftarrow$ **false**, $R_1(b_4) \leftarrow$ **false**, $Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **true** #
$R_2(b_5) \leftarrow$ **true**$\} \vee E$);
$Bt_7/R_1 = [Send(C_2, R_1, M_{C2,R1})$ & $R_1(b_4)$ & $R_1(r_1)]$
($\{Send(C_2, R_1, M_{C2,R1}) \leftarrow$ **false**, $R_1(b_4) \leftarrow$ **false**, $R_1(b_7) \leftarrow$ **true**,
$R_1(r_1) \leftarrow$ **false**$\} \vee E$);
$Bt_8/R_1 = [R_1(b_7)](\{R_1(b_7) \leftarrow$ **false**, $R_1(r_1) \leftarrow$ **true**,
$Send(R_1, C_2, M_{R1,C2}) \leftarrow$ **true** # $C_2(b_{10}) \leftarrow$ **true**$\} \vee E$);

– сетевые спецификации для модулей, реализуемых приложением-сервером R_2 для приложения-клиента C_2:

$Bt_5/R_2 = [Send(R_1, R_2, M_{R1,R2})$ & $R_2(b_5)$ & $\neg R_2(r_2)]$
($\{Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **false**, $R_2(b_5) \leftarrow$ **false**, $Send(R_2, R_3, M_{R2,R3}) \leftarrow$ **true** #
$R_3(b_6) \leftarrow$ **true**$\} \vee E$);
$Bt_9/R_2 = [Send(R_1, R_2, M_{R1,R2})$ & $R_2(b_5)$ & $R_2(r_2)]$
($\{Send(R_1, R_2, M_{R1,R2}) \leftarrow$ **false**, $R_2(b_5) \leftarrow$ **false**, $R_2(b_8) \leftarrow$ **true**,

$R_2(r_2) \leftarrow \mathbf{false}\} \vee E$);
$Bt_{10}/R_2 = [R_2(b_8)](\{R_2(b_8) \leftarrow \mathbf{false}, R_2(r_2) \leftarrow \mathbf{true},$
$Send(R_2, C_2, M_{R2,C2}) \leftarrow \mathbf{true} \# C_2(b_{10}) \leftarrow \mathbf{true}\} \vee E$);

– сетевые спецификации для модулей, реализуемых приложением-сервером R_3 для приложения-клиента C_2:

$Bt_6/R_3 = [\neg S(z)\ \&\ Send(R_2, R_3, M_{R2,R3})\ \&\ R_3(b_6)\ \&\ \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(b_6) \leftarrow \mathbf{false}, Send(R_3, R_1, M_{R3,R1}) \leftarrow \mathbf{true} \#$
$R_1(b_4) \leftarrow \mathbf{true}\} \vee E$);
$Bt_6{}^*/R_3 = [S(z)\ \&\ Send(R_2, R_3, M_{R2,R3})\ \&\ R_3(b_6)\ \&\ \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(b_6) \leftarrow \mathbf{false}, Send(R_3, C_2, M_{R3,C2}) \leftarrow \mathbf{true} \#$
$C_2(b_1) \leftarrow \mathbf{true}\} \vee E$);
$Bt_{11}/R_3 = [Send(R_2, R_3, M_{R2,R3})\ \&\ R_3(b_6)\ \&\ R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(b_6) \leftarrow \mathbf{false}, R_3(b_9) \leftarrow \mathbf{true},$
$R_3(r_3) \leftarrow \mathbf{false}\} \vee E$);
$Bt_{12}/R_3 = [R_3(b_9)](\{R_3(b_9) \leftarrow \mathbf{false}, R_3(r_3) \leftarrow \mathbf{true},$
$Send(R_3, C_2, M_{R3,C2}) \leftarrow \mathbf{true} \# C_2(b_{10}) \leftarrow \mathbf{true}\} \vee E$);

– сетевые спецификации для модулей, реализуемых приложением-клиентом C_3:

$Ct_1/C_3 = [\neg S(z)\ \&\ Start\ \&\ C_3(c_1)](\{Send(C_3, R_1, M_{C3,R1}) \leftarrow \mathbf{true} \# R_1(c_4) \leftarrow$
$\mathbf{true}, C_3(c_1) \leftarrow \mathbf{false}, C_3(c_2) \leftarrow \mathbf{true}\} \vee E$);
$Ct_1{}^*/C_3 = [S(z)\ \&\ Start\ \&\ C_3(c_1)](\{Send(C_3, R_1, M_{C3,R1}) \leftarrow \mathbf{true} \# R_1(c_4) \leftarrow$
$\mathbf{true},$
$C_3(c_1) \leftarrow \mathbf{false}\} \vee E$);
$Ct_2/C_3 = [\neg S(z)\ \&\ (Send(R_1, C_3, M_{R1,C3}) \vee Send(R_2, C_3, M_{R2,C3}) \vee Send(R_3, C_3,$
$M_{R3,C3}))\ \&\ C_3(c_{10})](\{Send(R_1, C_3, M_{R1,C3}) \leftarrow \mathbf{false}, Send(R_2, C_3, M_{R2,C3}) \leftarrow \mathbf{false},$
$Send(R_3, C_3, M_{R3,C3}) \leftarrow \mathbf{false}, C_3(c_{10}) \leftarrow \mathbf{false}, C_3(c_2) \leftarrow \mathbf{false}, C_3(c_3) \leftarrow$
$\mathbf{true}\} \vee E$);
$Ct_2{}^*/C_3 = [S(z)\ \&\ (Send(R_1, C_3, M_{R1,C3}) \vee Send(R_2, C_3, M_{R2,C3}) \vee Send(R_3, C_3,$
$M_{R3,C3}))\ \&\ C_3(c_{10})](\{Send(R_1, C_3, M_{R1,C3}) \leftarrow \mathbf{false}, Send(R_2, C_3, M_{R2,C3}) \leftarrow \mathbf{false},$
$Send(R_3, C_3, M_{R3,C3}) \leftarrow \mathbf{false}, C_3(c_{10}) \leftarrow \mathbf{false}, C_3(c_3) \leftarrow \mathbf{true}\} \vee E$);
$Ct_3/C_3 = [C_3(c_3)](\{C_3(c_3) \leftarrow \mathbf{false}, C_3(c_1) \leftarrow \mathbf{true}\} \vee Ret_{\tau,n}(Ct_1/C_3))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_1 для приложения-клиента C_3:

$Ct_4/R_1 = [Send(C_3, R_1, M_{C3,R1})\ \&\ R_1(c_4)\ \&\ \neg R_1(r_1)]$
$(\{Send(C_3, R_1, M_{C3,R1}) \leftarrow \mathbf{false}, R_1(c_4) \leftarrow \mathbf{false}, Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{true} \#$
$R_2(c_5) \leftarrow \mathbf{true}\} \vee E$);
$Ct_7/R_1 = [Send(C_3, R_1, M_{C3,R1})\ \&\ R_1(c_4)\ \&\ R_1(r_1)]$
$(\{Send(C_3, R_1, M_{C3,R1}) \leftarrow \mathbf{false}, R_1(c_4) \leftarrow \mathbf{false}, R_1(c_7) \leftarrow \mathbf{true},$
$R_1(r_1) \leftarrow \mathbf{false}\} \vee E$);
$Ct_8/R_1 = [R_1(c_7)](\{R_1(c_7) \leftarrow \mathbf{false}, R_1(r_1) \leftarrow \mathbf{true},$
$Send(R_1, C_3, M_{R1,C3}) \leftarrow \mathbf{true} \# C_3(c_{10}) \leftarrow \mathbf{true}\} \vee E$);

– сетевые спецификации для модулей, реализуемых приложением-сервером R_2 для приложения-клиента C_3:

$Ct_5/R_2 = [Send(R_1, R_2, M_{R1,R2}) \,\&\, R_2(c_5) \,\&\, \neg R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{false}, R_2(c_5) \leftarrow \mathbf{false}, Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{true} \,\#$
$R_3(c_6) \leftarrow \mathbf{true}\} \vee E)$;
$Ct_9/R_2 = [Send(R_1, R_2, M_{R1,R2}) \,\&\, R_2(c_5) \,\&\, R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{false}, R_2(c_5) \leftarrow \mathbf{false}, R_2(c_8) \leftarrow \mathbf{true},$
$R_2(r_2) \leftarrow \mathbf{false}\} \vee E)$;
$Ct_{10}/R_2 = [R_2(c_8)](\{R_2(c_8) \leftarrow \mathbf{false}, R_2(r_2) \leftarrow \mathbf{true},$
$Send(R_2, C_3, M_{R2,C3}) \leftarrow \mathbf{true} \,\# C_3(c_{10}) \leftarrow \mathbf{true}\} \vee E)$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_3 для приложения-клиента C_3:

$Ct_6/R_3 = [\neg S(z) \,\&\, Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(c_6) \,\&\, \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(c_6) \leftarrow \mathbf{false}, Send(R_3, R_1, M_{R3,R1}) \leftarrow \mathbf{true} \,\#$
$R_1(c_4) \leftarrow \mathbf{true}\} \vee E)$;
$Ct_6{}^*/R_3 = [S(z) \,\&\, Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(c_6) \,\&\, \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(c_6) \leftarrow \mathbf{false}, Send(R_3, C_3, M_{R3,C3}) \leftarrow \mathbf{true} \,\#$
$C_3(c_1) \leftarrow \mathbf{true}\} \vee E)$;
$Ct_{11}/R_3 = [Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(c_6) \,\&\, R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(c_6) \leftarrow \mathbf{false}, R_3(c_9) \leftarrow \mathbf{true},$
$R_3(r_3) \leftarrow \mathbf{false}\} \vee E)$;
$Ct_{12}/R_3 = [R_3(c_9)](\{R_3(c_9) \leftarrow \mathbf{false}, R_3(r_3) \leftarrow \mathbf{true},$
$Send(R_3, C_3, M_{R3,C3}) \leftarrow \mathbf{true} \,\# C_3(c_{10}) \leftarrow \mathbf{true}\} \vee E)$;

– сетевые спецификации для модулей, реализуемых приложением-клиентом C_4:

$Dt_1/C_4 = [\neg S(z) \,\&\, Start \,\&\, C_4(d_1)](\{Send(C_4, R_1, M_{C4,R1}) \leftarrow \mathbf{true} \,\# R_1(d_4) \leftarrow$
$\mathbf{true}, C_4(d_2) \leftarrow \mathbf{true}, C_4(d_1) \leftarrow \mathbf{false}\} \vee E)$;
$Dt_1{}^*/C_4 = [S(z) \,\&\, Start \,\&\, C_4(d_1)](\{Send(C_4, R_1, M_{C4,R1}) \leftarrow \mathbf{true} \,\# R_1(d_4) \leftarrow$
$\mathbf{true}, C_4(d_1) \leftarrow \mathbf{false}\} \vee E)$;
$Dt_2/C_4 = [(Send(R_1, C_4, M_{R1,C4}) \vee Send(R_2, C_4, M_{R2,C4}) \vee Send(R_3, C_4, M_{R3,C4}))$
$\&\, C_4(d_{10})](\{Send(R_1, C_4, M_{R1,C4}) \leftarrow \mathbf{false}, Send(R_2, C_4, M_{R2,C4}) \leftarrow \mathbf{false},$
$Send(R_3, C_4, M_{R3,C4}) \leftarrow \mathbf{false}, C_4(d_{10}) \leftarrow \mathbf{false}, C_4(d_3) \leftarrow \mathbf{true}\} \vee E)$;
$Dt_3/C_4 = [C_4(d_3)](\{C_4(d_3) \leftarrow \mathbf{false}, C_4(d_1) \leftarrow \mathbf{true}\} \vee Ret_{\tau,n}(Dt_1/C_4))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_1 для приложения-клиента C_4:

$Dt_4/R_1 = [Send(C_4, R_1, M_{C4,R1}) \,\&\, R_1(d_4) \,\&\, \neg R_1(r_1)]$
$(\{Send(C_4, R_1, M_{C4,R1}) \leftarrow \mathbf{false}, R_1(d_4) \leftarrow \mathbf{false}, Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{true} \,\#$
$R_2(d_5) \leftarrow \mathbf{true}\} \vee E)$;
$Dt_7/R_1 = [Send(C_4, R_1, M_{C4,R1}) \,\&\, R_1(d_4) \,\&\, R_1(r_1)]$
$(\{Send(C_4, R_1, M_{C4,R1}) \leftarrow \mathbf{false}, R_1(d_4) \leftarrow \mathbf{false}, R_1(d_7) \leftarrow \mathbf{true},$
$R_1(r_1) \leftarrow \mathbf{false}\} \vee E)$;
$Dt_8/R_1 = [R_1(d_7)](\{R_1(d_7) \leftarrow \mathbf{false}, R_1(r_1) \leftarrow \mathbf{true},$
$Send(R_1, C_4, M_{R1,C4}) \leftarrow \mathbf{true} \,\# C_4(d_{10}) \leftarrow \mathbf{true}\} \vee Ret_{\tau,n}(At_4/R_1))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_2 для приложения-клиента C_4:

$Dt_5/R_2 = [Send(R_1, R_2, M_{R1,R2}) \,\&\, R_2(d_5) \,\&\, \neg R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{false}, R_2(d_5) \leftarrow \mathbf{false}, Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{true} \,\#$
$R_3(d_6) \leftarrow \mathbf{true}\} \vee E)$;
$Dt_9/R_2 = [Send(R_1, R_2, M_{R1,R2}) \,\&\, R_2(d_5) \,\&\, R_2(r_2)]$
$(\{Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{false}, R_2(d_5) \leftarrow \mathbf{false}, R_2(d_8) \leftarrow \mathbf{true},$
$R_2(r_2) \leftarrow \mathbf{false}\} \vee E)$;
$Dt_{10}/R_2 = [R_2(d_8)](\{R_2(d_8) \leftarrow \mathbf{false}, R_2(r_2) \leftarrow \mathbf{true},$
$Send(R_2, C_4, M_{R2,C4}) \leftarrow \mathbf{true} \,\# C_4(d_{10}) \leftarrow \mathbf{true}\} \vee Ret_{\tau,n}(At_5/R_2))$;

– сетевые спецификации для модулей, реализуемых приложением-сервером R_3 для приложения-клиента C_4:

$Dt_6/R_3 = [\neg S(z) \,\&\, Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(d_6) \,\&\, \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(d_6) \leftarrow \mathbf{false}, Send(R_3, R_1, M_{R3,R1}) \leftarrow \mathbf{true},$
$R_1(d_4) \leftarrow \mathbf{true}\} \vee E)$;
$Dt_6{*}/R_3 = [S(z) \,\&\, Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(d_6) \,\&\, \neg R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(d_6) \leftarrow \mathbf{false}, Send(R_3, C_4, M_{R3,C4}) \leftarrow \mathbf{true} \,\#$
$C_4(d_1) \leftarrow \mathbf{true}\} \vee E)$;
$Dt_{11}/R_3 = [Send(R_2, R_3, M_{R2,R3}) \,\&\, R_3(d_6) \,\&\, R_3(r_3)]$
$(\{Send(R_2, R_3, M_{R2,R3}) \leftarrow \mathbf{false}, R_3(d_6) \leftarrow \mathbf{false}, R_3(d_9) \leftarrow \mathbf{true},$
$R_3(r_3) \leftarrow \mathbf{false}\} \vee E)$;
$Dt_{12}/R_3 = [R_3(d_9)](\{R_3(d_9) \leftarrow \mathbf{false}, R_3(r_3) \leftarrow \mathbf{true},$
$Send(R_3, C_4, M_{R3,C4}) \leftarrow \mathbf{true} \,\# C_4(d_{10}) \leftarrow \mathbf{true}\} \vee Ret_{\tau,n}(At_6/R_3))$.

4. Методика отображения логико-алгебраических операционных выражений на архитектуру ОС РВС типа FaaS и AaaS

Приведенные выше выражения описывают в логико-алгебраической форме срабатывания переходов в бинарных сетях Петри, представленных на рис. 7 и 8. Построенные модели относятся к классу исполнимых, поскольку на их основе возможно построить прототипное программное обеспечение для компьютерной сети, реализующей функции ОС РВС типа FaaS или AaaS. Для облегчения этого построения в выражениях используются правила обновления тернарных предикатов вида $Send(C_1, R_1, M_{C1,R1}) \leftarrow \mathbf{true}$ и $Send(C_1, R_1, M_{C1,R1}) \leftarrow \mathbf{false}$. Первое из этих правил соответствует в сетевом приложении команде отправки сообщения $M_{C1,R1}$ из узла C_1, а второе – состоявшемуся приему этого сообщения узлом R_1.

Методика отображения логико-алгебраических операционных выражений на архитектуру ОС РВС типа FaaS или AaaS состоит в формировании логико-алгебраических операционных выражений по модульному принципу, распределении модулей по серверам. В связи с регулярностью моделей $M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ и $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ это распределение выполняется естественным образом.

Клиентские (C_1, C_2, C_3, C_4) и серверные (R_1, R_2, R_3) агентно-ориентированные приложения реализуются в виде сетевых программных модулей на основе логико-алгебраических операционных выражений. Например, клиентское приложение C_1 включает модули – программные реализации выражений At_1/C_1, $At_1{*}/C_1$, At_2/C_1, $At_2{*}/C_1$ и $At_3/C1$, а серверное приложение R_3 – модули At_6/R_3, $At_6{*}/R_3$, At_{11}/R_3, At_{12}/R_3, Bt_6/R_3, $Bt_6{*}/R_3$, Bt_{11}/R_3, Bt_{12}/R_3, Ct_6/R_3,

Ct_6*/R_3, Ct_{11}/R_3, Ct_{12}/R_3, Dt_6/R_3, Dt_6*/R_3, Dt_{11}/R_3, Dt_{12}/R_3. Звездочками здесь отмечены альтернативно исполняемые выражения системы Σ_{4x3} для реконфигурируемой модели, построенной на основе совмещения моделей $M_3(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$ и $M_4(C_1, C_2, C_3, C_4, R_1, R_2, R_3)$.

В отличие от предыдущих упрощенных моделей $M_1(C, R_1, R_2, R_3)$ и $M_2(C, R_1, R_2, R_3)$, при построении последней логико-алгебраической модели Σ_{4x3} на формальном уровне учтены особенности объекта моделирования при работе в сети Internet Cloud. При описании модели и сетевого приложения будут, как это часто принято для сокращения описания, смешиваться понятия об объектах модели и объектах предметной области. Проанализируем некоторые основные моменты построения формализованных спецификаций на примере следующего выражения At_1/C_1 для перехода at_1:

$$At_1/C_1 = [\neg S(z) \,\&\, Start \,\&\, C_1(a_1)](\{Send(C_1, R_1, M_{C1,R1}) \leftarrow \mathbf{true} \,\#\, R_1(a_4) \leftarrow \mathbf{true}, C_1(a_1) \leftarrow \mathbf{false}, C_1(a_2) \leftarrow \mathbf{true}\} \vee E).$$

Имя At_1/C_1 означает, что процедура At_1/C_1 перехода at_1 в виде программного модуля реализуется на компьютере клиента C_1. Модуль начинает выполнение с проверки истинности составного высказывания $\neg S(z)\&Start\&C_1(a_1)$, где $S(z)$ – предикат настройки модуля на заданный режим работы по *схеме 1* или *схеме 2*, *Start* – стартовое условие для модуля, $C_1(a_1)$ – условие готовности клиента к выдаче сообщения. Если указанное условие истинно, то далее выполняется следующая последовательность действий: $Send(C_1, R_1, M_{C1,R1})\leftarrow$**true** $\#R_1(a_4)\leftarrow$**true**.

Обновлению тернарного предиката $Send(C_1, R_1, M_{C1,R1})\leftarrow$**true** соответствует отправка сообщения $M_{C1,R1}$ от приложения клиента C_1 приложению сервера R_1. Обновлению унарного предиката разметки $R_1(a_4)\leftarrow$**true** соответствует факт приема сообщения $M_{C1,R1}$ приложением сервера R_1. Символ "#" означает, что два действия причинно связаны и выполняются последовательно. Выполнение правила $C_1(a_2)\leftarrow$**true** соответствует переходу приложения клиента в состояние ожидания ответного сообщения от одного из серверов. Логико-алгебраическое выражение

$$At_4/R_1 = [Send(C_1, R_1, M_{C1,R1}) \,\&\, R_1(a_4) \,\&\, \neg R_1(r_1)]$$
$$(\{Send(C_1, R_1, M_{C1,R1}) \leftarrow \mathbf{false}, R_1(a_4) \leftarrow \mathbf{false}, Send(R_1, R_2, M_{R1,R2}) \leftarrow \mathbf{true} \,\#\, R_2(a_5) \leftarrow \mathbf{true}\} \vee E)$$

реализуется программным модулем приложения сервера R_1. Этот модуль принимает сообщение $M_{C1,R1}$. О наличии этого сообщения в модели сервера R_1 свидетельствует истинность составного высказывания $Send(C_1, R_1, M_{C1,R1})\&R_1(a_4)$. Сервер R_1 недоступен при ложности высказывания $R_1(r_1)$. Поэтому в модели сервер R_1 аннулирует сообщение путем обновления предикатов $Send(C_1, R_1, M_{C1,R1})\leftarrow$**false** и $R_1(a_4)\leftarrow$**false** и далее посылает сообщение $M_{R1,R2}$ следующему серверу R_2. В модели серверного приложения этот факт отмечается последовательным выполнением следующих действий: $Send(R_1, R_2, M_{R1,R2})\leftarrow$**true**$\# R_2(a_5)\leftarrow$**true**.

Остальные выражения составлены по такому же принципу. В некоторых выражениях применяются операторы вида $Ret_{\tau,n}$, используемые для передачи управления конкретному модулю. Например, после выполнения выражения

$$At_3/C_1 = [C_1(a_3)](\{C_1(a_3) \leftarrow \textbf{false}, C_1(a_1) \leftarrow \textbf{true}\} \vee Ret_{\tau,n}(At_1/C_1))$$

и при $C_1(a_3) = \textbf{false}$ выполняется оператор $Ret_{\tau,n}(At_1/C_1)$, передающий управление модулю At_1/C_1) после отработки тайм-аута τ, но не более n раз.

Заключение

Предложена организация функционирования облачно-сетевых РВС, основанная на формализации перехода от известной облачной архитектуры «функция как сервис» (FaaS – Function-as-a-Service) к новой архитектуре «агент как сервис» (AaaS – Agent-as-a-Service). Предложена формализация функциональной архитектуры ОС РВС системами логико-алгебраических операционных выражений, относящихся к классу исполнимых моделей и пригодных для непосредственной программной реализации в сетевой компьютерной среде; дополнительным свойством ЛАОВ является возможность реконфигурации (модификации режима функционирования) результирующего сетевого программного обеспечения. Предложена методика отображения системы логико-алгебраических операционных выражений на архитектуру компьютерной сети, учитывающая регулярный характер модели.

Методика синтеза облачных систем с новой архитектурой AaaS, основанная на формализованных логико-алгебраических спецификациях, позволяет ускорить создание программного обеспечения для сервис-ориентированных распределенных вычислительных систем и расширить их функциональные возможности.

В качестве направления дальнейших исследований предлагается развивать методы абстрактного и структурного синтеза облачно-сетевых сервис-ориентированных распределенных вычислительных систем и поддерживающих их агентских платформ на основе формализованных спецификаций и автоматической генерации сетевого программного обеспечения промежуточного уровня (*middleware level*).

Библиографический список

1. Apache CloudStack: Open Source Infrastructure as a Service Cloud Computing Platform / R. Kumar, K. Jain, H. Maharwal, N. Jain, A. Dadhich // International Journal of advancement in Engineering technology, Management and Applied Science (IJAETMAS). – 2014. – Vol. 1, № 2. – P. 111–116.
2. **Kumar, R.** Comparison between Cloud Computing, Grid Computing, Cluster Computing and Virtualization / R. Kumar // International Journal of Modern Computer Science and Applications (IJMCSA). – 2015. – Vol. 3, № 1. – P. 42–47.
3. **Tanenbaum, A. S.** Distributed Systems: principles and paradigms / A. S. Tanenbaum, Maarten van Steen. – 2nd Edition. – Pearson Education, Inc., 2007. – 669 p.
4. FIPA Specifications. – URL: http://www.fipa.org/specifications/index.html, свободный (дата обращения: 12.11.2019).
5. A Survey of Programming Languages and Platforms for Multi-Agent Systems / R. H. Bordini et al. // Informatica. – 2006. – Vol. 30. – P. 33–44.
6. **Kravari, K.** A Survey of Agent Platforms / K. Kravari, N. Bassiliades // Journal of Artificial Societies and Social Simulation. – 2015. – Vol. 18 (1), № 11. – P. 1–18.
7. **Cynthia, N.** Tools of the Trade: A Survey of Various Agent Based Modeling Platforms / N. Cynthia, M. Gregory // Journal of Artificial Societies and Social Simulation. – 2009. – Vol. 12, № 2 2. – URL: http://jasss.soc.surrey.ac.uk/12/2/2.html.

8. **Bellifemine, F. L.** Developing multi-agent systems with JADE / F. L. Bellifemine, G. Caire, D. Greenwood. – Wiley, 2007. – 300 p.
9. Java Agent Development Environment (JADE). – URL: http://jade.tilab.com/, свободный (дата обращения: 12.11.2019).
10. Development of Mobile Agents with Aglets (A Java Based Tool) / M. Yadav, P. Sethi, D. Juneja, N. Chauhan // Int. Journal of Innovations & Advancement in Computer Science. – 2015. – Vol. 4, Special Issue. – P. 245–251.
11. **Evripidou, P.** Metacomputing with Mobile Agents / P. Evripidou, G. Samaras // Int. Journal of Parallel Programming. – 2006. – Vol. 34, № 5. – P. 429–458.
12. **Barelos, D.** Mobile agents procedures: metacomputing in Java / D. Barelos, E. Pitoura, G. Samaras // Proc. of the ICDCS Workshop on Distributed Middleware (in conjunction with the 19th IEEE International Conference on Distributed Computing Systems (ICDCS99)). – Austin, TX USA, 1999. – P. 90–95.
13. **Samaras, G.** Extendible, Mobile-Agent Based Services for the Materialization and Maintenance of Personalized and Shareable Web Views ViSMA / G. Samaras, K. Karenos, P. K. Chrysanthis, E. Pitoura // In Proc. 11th DEXA Int. Workshop on Mobility in Databases and Distributed Systems, 2003. – P. 974–979.
14. **Kausar, Md. Abu** Web Crawler Based on Mobile Agent and Java Aglets / Md. Abu Kausar, V. S. Dhaka, Sanjeev Kumar Singh // International Journal of Information Technology and Computer Science (IJITCS). MECS Publisher. – 2013. – Vol. 5, № 10. – P. 85–91.
15. **Dada, E. G.** Performance Evaluation of AGLETS and JADE Mobile Agent Using Encryption and Decryption Time / E. G. Dada, S. B. Joseph, M. K. Mishra // Radioelectronics&Informatics. – 2010. – № 4. – P. 16–20.
16. **Lange, D.** Programming and deploying Java mobile agents with aglets / D. Lange, M. Oshima. – Addison : Wesley Professional, 1998. – 256 p.
17. **Стельмах, С.** Шесть ключевых преимуществ бессерверной архитектуры / С. Стельмах. – URL: https://www.itweek.ru/its/article/detail.php?ID=208802:, свободный (дата обращения: 12.11.2019).
18. FaaS (Function-as-a-Service) / Материал из Национальной библиотеки им. Н. Э. Баумана. – URL: https://ru.bmstu.wiki/FaaS_(Function-as-a-Service), свободный (дата обращения: 12.11.2019).
19. **Krol, M.** NFaaS: Named Function as a Service / M. Krol, I. Psaras // In Proceedings of ICN '17 (Berlin, Germany, September 26–28, 2017). – Berlin, 2017. – 11 p.
20. **Ellis, A.** Introducing Functions as a Service – OpenFaaS / A. Ellis. Опубл. 08.08.2017. – URL: https://blog.alexellis.io/introducing-functions-as-a-service/, свободный (дата обращения: 12.11.2019).
21. **Haines, S.** Serverless computing with AWS Lambda, 2018 / S. Haines. – URL: https://www.javaworld.com/article/3210726/serverless-computing-with-aws-lambda.html, свободный (дата обращения: 12.11.2019).
22. **Gorodetsky, V.** P2P Agent Platform: Implementation and Testing / V. Gorodetsky, O. Karsaev, V. Samoylov, S. Serebryakov // The AAMAS Sixth International Workshop on Agents and Peer-to-Peer. Computing (AP2PC 2007) (Honolulu, May 14–18, 2007). – Honolulu, 2007. – P. 41–54.
23. **Волчихин, В. И.** Абстрактное и структурное моделирование сетей хранения и обработки данных / В. И. Волчихин, С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2011. – № 4 (20). – С. 3–18.
24. **Волчихин, В. И.** Логико-алгебраические модели и методы в проектировании функциональной архитектуры распределенных систем хранения и обработки дан-

ных / В. И. Волчихин, С. А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2012. – № 2 (22). – С. 3–16.

25. **Зинкин, С. А.** Элементы новой объектно-ориентированной технологии для моделирования и реализации систем и сетей хранения и обработки данных / С. А. Зинкин // Информационные технологии. – 2008. – № 10. – С. 20–27.
26. **Карамышева, Н. С.** Методы и модели логического управления дискретными процессами в распределенных вычислительных системах на основе концепции согласования / Н. С. Карамышева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2011. – № 1 (17). – С. 35–47.
27. **Карамышева, Н. С.** Обобщенная логико-алгебраическая модель и организация мультиагентных метакомпьютерных систем / Н. С. Карамышева // Перспективы науки. – 2011. – № 4. – С. 98–108.
28. **Peterson, J. L.** Petri Net Theory and the Modeling of Systems / J. L. Peterson. – New Jersey : Prentice-Hall, 1981. – 288 p.
29. **Iordache, M. V.** Supervisory Control of Concurrent Systems. A Petri Net Structural Approach / M. V. Iordache, P. J. Antsaklis. – Boston : Birkhauser, 2006. – 281 p.
30. Petri Nets Fundamental Models, Verification and Applications / Edited by Michel Diaz. – John Wiley & Sons, 2009. – 585 p.
31. Методы параллельного микропрограммирования / П. А. Анишев, С. М. Ачасова, О. Л. Бандман и др. – Новосибирск : Наука, 1981. – 182 с.
32. Implementation of the interface for sending messages in agent-oriented cloud/grid systems based on formalized specifications / D. Pashchenko, S. Zinkin, A. Dubravin, N. Karamisheva // In IEEE Proc. of the Int. Siberian Conf. on Control and Communications (SIBCON 2016) (Moscow, Russia, 12–14 May 2016). – Moscow, 2016. – P. 173–178.
33. **Zinkin, S. A.** Organization of Autonomous Agent-Robots Interactions for Managing a Very Large Distributed Database System in a Metacomputer Environment / S. A. Zinkin, V. B. Mehanov, N. S. Karamisheva V. I. Volchihin // In Proceedings of the 2019 IEEE International Conference on Real-time Computing and Robotics (Irkutsk, Russia August 4–9, 2019). – Irkutsk, 2019. – P. 846–851.

References

1. Kumar R., Jain K., Maharwal H., Jain N., Dadhich A. *International Journal of advancement in Engineering technology, Management and Applied Science (IJAET-MAS)*. 2014, vol. 1, no. 2, pp. 111–116.
2. Kumar R. *International Journal of Modern Computer Science and Applications (IJMCSA)*. 2015, vol. 3, no. 1, pp. 42–47.
3. Tanenbaum A. S., Van Steen M. *Distributed Systems: principles and paradigms*. 2nd Edition. Pearson Education, Inc., 2007, 669 p.
4. *FIPA Specifications*. Available at: http://www.fipa.org/specifications/index.html, svobodnyy (accessed Nov. 12, 2019).
5. Bordini R. H. et al. *Informatica*. 2006, vol. 30, pp. 33–44.
6. Kravari K., Bassiliades N. *Journal of Artificial Societies and Social Simulation*. 2015, vol. 18 (1), no. 11, pp. 1–18.
7. Cynthia N., Gregory M. *Journal of Artificial Societies and Social Simulation*. 2009, vol. 12, no. 2 2. Available at: http://jasss.soc.surrey.ac.uk/12/2/2.html.
8. Bellifemine F. L., Caire G., Greenwood D. *Developing multi-agent systems with JADE*. Wiley, 2007, 300 p.
9. *Java Agent Development Environment (JADE)*. Available at: http://jade.tilab.com/, svobod-nyy (accessed Nov. 12, 2019).

10. Yadav M., Sethi P., Juneja D., Chauhan N. *Int. Journal of Innovations & Advancement in Computer Science*. 2015, vol. 4, Special Issue, pp. 245–251.
11. Evripidou P., Samaras G. *Int. Journal of Parallel Programming*. 2006, vol. 34, no. 5, pp. 429–458.
12. Barelos D., Pitoura E., Samaras G. *Proc. of the ICDCS Workshop on Distributed Middleware (in con-junction with the 19th IEEE International Conference on Distributed Computing Systems (ICDCS99))*. Austin, TX USA, 1999, pp. 90–95.
13. Samaras G., Karenos K., Chrysanthis P. K., Pitoura E. *In Proc. 11th DEXA Int. Workshop on Mobility in Databases and Distributed Systems*. 2003, pp. 974–979.
14. Md. Abu Kausar, Dhaka V. S., Sanjeev Kumar Singh *International Journal of Information Technology and Computer Science (IJITCS). MECS Publisher*. 2013, vol. 5, no. 10, pp. 85–91.
15. Dada E. G., Joseph S. B., Mishra M. K. *Radioelectronics&Informatics*. 2010, no. 4, pp. 16–20.
16. Lange D., Oshima M. *Programming and deploying Java mobile agents with aglets*. Addison : Wesley Professional., 1998, 256 p.
17. Stel'makh S. *Shest' klyuchevykh preimushchestv besservernoy arkhitektury* [Six key advantages of serverless architecture]. Available at: https://www.itweek.ru/its/article/detail.php?ID=208802:, svobodnyy (accessed Nov. 12, 2019). [In Russian]
18. *FaaS (Function-as-a-Service)*. Material iz Natsional'noy biblioteki im. N. E. Baumana. Available at: https://ru.bmstu.wiki/FaaS_(Function-as-a-Service), svobodnyy (accessed Nov. 12, 2019).
19. Krol M., Psaras I. *In Proceedings of ICN '17 (Berlin, Germany, September 26–28, 2017)*. Berlin, 2017, 11 p.
20. Ellis A. *Introducing Functions as a Service – OpenFaaS*. Publ. 08.08.2017. Available at: https://blog.alexellis.io/introducing-functions-as-a-service/, svo-bodnyy (accessed Nov. 12, 2019).
21. Haines S. *Serverless computing with AWS Lambda, 2018*. Available at: https://www.javaworld.com/article/3210726/serverless-computing-with-aws-lambda.html, svobodnyy (accessed Nov. 12, 2019).
22. Gorodetsky V., Karsaev O., Samoylov V., Serebryakov S. *The AAMAS Sixth International Work-shop on Agents and Peer-to-Peer. Computing (AP2PC 2007) (Honolulu, May 14–18, 2007)* [ ]. Honolulu, 2007, pp. 41–54.
23. Volchikhin V. I., Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2011, no. 4 (20), pp. 3–18. [In Russian]
24. Volchikhin V. I., Zinkin S. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2012, no. 2 (22), pp. 3–16. [In Russian]
25. Zinkin S. A. *Informatsionnye tekhnologii* [Information technologies]. 2008, no. 10, pp. 20–27. [In Russian]
26. Karamysheva N. S. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2011, no. 1 (17), pp. 35–47. [In Russian]
27. Karamysheva N. S. *Perspektivy nauki* [Prospects of science]. 2011, no. 4, pp. 98–108. [In Russian]
28. Peterson J. L. *Petri Net Theory and the Modeling of Systems*. New Jersey: Prentice-Hall, 1981, 288 p.
29. Iordache M. V., Antsaklis P. J. *Supervisory Control of Concurrent Systems. A Petri Net Structural Approach*. Boston: Birkhauser, 2006, 281 p.
30. *Petri Nets Fundamental Models, Verification and Applications*. Edited by Michel Diaz. John Wiley & Sons, 2009, 585 p.

31. Anishev P. A., Achasova S. M., Bandman O. L. et al. *Metody parallel'nogo mikroprogrammirovaniya* [Methods of parallel microprogramming]. Novosibirsk: Nauka, 1981, 182 p. [In Russian]
32. Pashchenko D., Zinkin S., Dubravin A., Karamisheva N. *In IEEE Proc. of the Int. Siberian Conf. on Control and Communications (SIBCON 2016). (Moscow, Russia, 12–14 May 2016)*. Moscow, 2016, pp. 173–178.
33. Zinkin S. A., Mehanov V. B., Karamisheva N. S., Volchihin V. I. *In Proceedings of the 2019 IEEE International Conference on Real-time Computing and Robotics (Irkutsk, Russia August 4–9, 2019)*. Irkutsk, 2019, pp. 846–851.

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

E-mail: cnit@pnzgu.ru

Volchihin Vladimir Ivanovich
Doctor of engineering sciences, professor, president of Penza State University (40 Krasnaya street, Penza, Russia)

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

E-mail: vt@pnzgu.ru

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

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

E-mail: vt@pnzgu.ru

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

Образец цитирования:

Волчихин, В. И. Организация функционирования облачно-сетевых распределенных вычислительных систем с архитектурой «агенты как сервисы» / В. И. Волчихин, С. А. Зинкин, Н. С. Карамышева // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2019. – № 4 (52). – С. 27–50. – DOI 10.21685/2072-3059-2019-4-3.