ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

COMPUTER SCIENCE, COMPUTER ENGINEERING AND CONTROL

УДК 004.9
doi: 10.21685/2072-3059-2024-2-1

Реализация управляемой приложением функциональной архитектуры пиринговой распределенной вычислительной системы, определяемой концептуальными и логическими моделями искусственного интеллекта. II. Декларативно-императивный и логико-алгебраический подходы

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

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

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

Аннотация. *Актуальность и цели*. В настоящее время ведутся интенсивные поиски решений в области создания инфраструктуры для сетевых приложений, например для децентрализованного Интернета на основе нескольких основных технологий: блокчейна, машинного обучения, Семантической паутины и Интернета вещей. Рассматривается возможность организации инфраструктуры децентрализованного Интернета (*Distributed Web – DWeb*) в виде одноранговой P2P-сети (*peer-to-peer*), узлами которой являются устройства пользователей. Целью настоящей работы является исследование одного из подходов к практической реализации управляемой приложением функциональной архитектуры сетевой распределенной вычислительной системы (РВС) ADFA-DCS (*Application-Driven Functional Architecture of Distributed Computing System*), определяемой концептуальными и логическими моделями искусственного интеллекта. *Материалы и методы.* Предложена новая функциональная архитектура РВС, определяемая приложением, реализация которой отличается от известных тем, что работа приложения предварительно определяется исполнимой моделью – концептуальным графом сети Петри. Затем формируется структура, задающая логику работы приложения, и на ее основе генерируется распределенная управляющая программа. Этот принцип позволяет реализовывать произвольные распределенные алгоритмы без существенных затрат на перепрограммирование приложения. *Результаты.* Предложенная функциональная архитектура реализуется на платформе глобальной или локальной вычислительной сети, на которой определена логическая системная архитектура РВС. Для реализации системной архитектуры рассматриваемых РВС выбраны технологии, близкие к пиринговой технологии P2P. Авторы настоящей работы полагают, что концепция РВС с предложенной архитектурой хорошо согласуется с платформой территориально-распределенных корпоративных сетей SD-WAN

(*Software Defined Wide Area Network*), при построении которых реализован принципиально новый подход к управлению сетью. *Выводы.* Разработана технология создания функциональной архитектуры РВС, управляемой приложением, которая согласована с технологиями пиринговых и программно-определяемых сетевых технологий. Процесс программирования распределенного приложения сопровождается использованием функций передачи сообщений и данных, поиском данных в таблицах, копии которых находятся на всех узлах P2P-сети и реализацией операторов, выполняющих функции, для которых и была предназначена создаваемая сетевая РВС.

Ключевые слова: распределенные вычислительные системы, P2P-вычисления, функциональная архитектура, технологии локальных и глобальных сетей, гибридные P2P-сети, декларативно-императивная модель, концептуальная логическая сеть Петри, формализация декларативно-императивной и логико-алгебраический модели распределенных вычислений

Для цитирования: Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Реализация управляемой приложением функциональной архитектуры пиринговой распределенной вычислительной системы, определяемой концептуальными и логическими моделями искусственного интеллекта. II. Декларативно-императивный и логико-алгебраический подходы // Известия высших учебных заведений. Поволжский регион. Технические науки. 2024. № 2. С. 5–31. doi: 10.21685/2072-3059-2024-2-1

The implementation of an application-driven functional architecture of a peer-to-peer distributed computing system defined by conceptual and logical models of artificial intelligence. II. The declarative-imperative and logical-algebraic approaches

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

[1,2,3,4]Penza State University, Penza, Russia

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

Abstract. *Background*. Currently, there is an intensive search for solutions in the field of creating infrastructure for network applications, for example, for the decentralized Internet based on several main technologies: blockchain, machine learning, Semantic Web and Internet of Things. The possibility of organizing the infrastructure of the decentralized Internet, or Distributed Web – DWeb, in the form of a peer-to-peer (P2P) network, the nodes of which are user devices, is being considered. The *aim* of this work is to study one of the approaches to the practical implementation of the application-driven functional architecture of the ADFA-DCS (Application-Driven Functional Architecture of Distributed Computing System), defined by conceptual and logical artificial intelligence models. *Materials and methods.* A new functional architecture of the DCS is proposed, which is determined by the application, the implementation of which differs from the known ones in that the operation of the application is preliminarily determined by the executable model – the conceptual Petri net graph. Then a structure is formed that specifies the logic of the application, and on its basis a distributed control program is generated. This principle makes it possible to implement arbitrary distributed algorithms without significant costs for reprogramming the application. The authors of this work believe that the concept of a distributed network with the proposed architecture is in good agreement with the platform of geographically distributed corporate networks SD-WAN (Software Defined Wide Area Network), the construction of which implements a fundamentally new approach to network management. *Results.* The proposed functional architecture is implemented on a global (WAN) or local area network platform (LAN), on which the logical system architecture of a distributed computing

system is defined. To implement the system architecture of the DCS under consideration, technologies close to peer-to-peer technology were selected. The process of programming a distributed application is accompanied by the use of message and data transfer functions, searching for data in tables, copies of which are located on all nodes of the P2P network, and the implementation of operators performing functions for which the created network DCS was intended.

Keywords: distributed computing systems, P2P computing, functional architecture, local and global network technologies, hybrid P2P networks, declarative-imperative model, conceptual logical Petri net, formalization of declarative-imperative and logical-algebraic models of distributed computing

For citation: Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. The implementation of an application-driven functional architecture of a peer-to-peer distributed computing system defined by conceptual and logical models of artificial intelligence. II. The declarative-imperative and logical-algebraic approaches. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2024;(2):5–31. (In Russ.). doi: 10.21685/2072-3059-2024-2-1

Введение

Распределенные вычислительные системы (РВС) появились как результат развития новых технологий, направленных на реализацию удаленного доступа пользователей к удаленным вычислительным ресурсам. Получили развитие архитектуры типа «клиент-сервер» и «мастер-исполнитель», а собственно сложные в алгоритмическом плане и в плане использования вычислительных ресурсов вычисления реализуются в удаленных кластерных и мощных массивно-параллельных системах, в рамках архитектуры грид-систем (англ. *grid* – сетка, решетка) [1–3]. Многими специалистами получены решения фундаментальных проблем, связанных с архитектурой грид-вычислений в открытых средах. На основе концепции грид-вычислений предложен, но до конца не реализован, ряд новых самонастраивающихся, адаптивных и развивающихся архитектур.

Пиринговые, или *peer-to-peer* (P2P), вычисления, стали, в числе других моделей, распространенным вариантом реализации крупномасштабных распределенных вычислений [4–6]. Распределенные вычислительные системы на базе P2P-сетей способны поддерживать декомпозиции задач и взаимодействие между задачами. В отличие от архитектуры типа «клиент-сервер» и родственной ей архитектуры «мастер-исполнитель», такая организация вычислений позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов.

Проведенный анализ известных публикаций в области создания РВС показывает, что описания известных решений касаются в основном системной архитектуры и внутреннего управления функционированием инфраструктуры на промежуточном (*middleware*) уровне. Алгоритмические проблемы, возникающие на прикладном уровне, равно как и реализация распределенных приложений в распределенной вычислительной среде, в литературе описана недостаточно. Обычно пользователю предоставляются готовые сервисы, реализованные в P2P-среде. Однако эта среда может использоваться и для создания пользовательских сетевых приложений, которые фактически задают функциональную архитектуру РВС.

Представляет большой интерес известная технология SD-WAN (англ. *Software-Driven Wide Area Network*) [7, 8] обладающая рядом преимуществ, которые, как предполагают многие специалисты, в перспективе будут востребованы в большинстве современных сетей. Появляющиеся приложения и рабочие сценарии предъявляют строгие требования к передаче данных на большие расстояния, заставляя сетевых операторов проектировать глобальные сети с новой точки зрения. Программно-определяемая глобальная сеть, т.е. SD-WAN, считается многообещающей архитектурой глобальных сетей следующего поколения. Отметим, что реализация основных стратегий маршрутизации важна и в так называемых неструктурированных P2P-сетях для последующего эффективного поиска и обмена данными. Остается актуальным решение следующих задач: выбор наилучшей стратегии планирования и управления ресурсами; обеспечение совместимости и взаимодействия пиринговых узлов. Частично эти задачи можно решать путем использования технологии программно-определяемых распределенных сетей SD-WAN, которые специально предназначены для управления сетью и передачи данных между центром и периферией. Функция центра, не являющегося сервером, не противоречит идее пиринговых вычислений, тем более, что во многих P2P-сетях центральный узел обеспечивает координацию работы равноправных узлов и управление безопасностью сети. Сети SD-WAN обеспечивают интеллектуальное управление трафиком, который передается от центра к периферии и обратно; для этих сетей характерна также единая точка управления и мониторинга всей инфраструктуры.

Возможны различные технологии разработки приложений подобного вида, из которых наиболее близкими являются технологии на основе какого-либо агентно-ориентированного фреймворка, например JADE (англ. *Java Agent DEvelopment Framework*) [9, 10], и технологии волонтерских вычислений VC (англ. *Volunteer Computing*), положенные в основу создания программного комплекса BOINC (англ. *Berkeley Open Infrastructure for Network Computing*) [11, 12]. Волонтерские вычисления основаны на использовании пользовательских компьютеров и позволяют выполнять большой объем вычислительных работ. Платформа BOINC (широко используемая система промежуточного программного обеспечения с открытым исходным кодом) позволяет избавиться от многих недостатков, присущих волонтерским вычислениям и связанных с их ненадежностью, неоднородностью и переменным составом ресурсов.

В проектах, основанных на волонтерских вычислениях, активно участвуют около 700 тыс. устройств. Эти устройства имеют около 4 млн процессорных ядер и 560 тыс. графических процессоров, в совокупности обеспечивая среднюю пропускную способность 93 петафлопс. Устройства представляют собой в основном современные компьютеры высокого класса: их средняя производительность составляет 16,5 гигафлопс процессора, а емкость оперативной памяти – 11,4 гигабайт; большинство из них имеют графические процессоры, способные выполнять вычисления общего назначения с использованием OpenCL или CUDA [12].

Другой крупный проект – Worldwide LHC Computing Grid, WLCG (англ. *Worldwide LHC Large Hadron Collider Computing Grid*) [13], реализуется на основе глобального сотрудничества около 170 вычислительных цен-

тров, объединяющих национальные и международные грид-инфраструктуры, расположенные в более чем 40 странах. WLCG объединяет глобальные вычислительные ресурсы для хранения, распространения и анализа примерно 200 петабайт данных, ожидаемых каждый год от работы Большого адронного коллайдера (БАК) в ЦЕРНе – Европейском центре ядерных исследований (CERN – The European Center for Nuclear Research), крупнейшем в мире исследовательском центре физики элементарных частиц.

Несмотря на огромные вычислительные мощности компьютеров в приведенных в качестве примеров вычислительных платформах, их архитектура практически не приспособлена для реализации концепции, выраженной слоганом фирмы SUN Microsystems (ныне в составе фирмы Oracle) “The network is the computer” (“Сеть – это компьютер”). Некоторые информационно-коммуникационные инфраструктуры, поддерживающие эту концепцию, первоначально были рассмотрены в работе [14].

Описываемый в настоящей работе проект сохраняет преемственность по отношению к проектам РВС, реализации которых выполнены на платформе JADE для мультиагентных систем [15–18], отличаясь от данных работ платформой и методом формирования функциональной архитектуры.

1. Декларативно-императивный и логико-алгебраический подходы к синтезу функциональной архитектуры РВС на платформе P2P-сети

Концепция управляемой приложением функциональной архитектуры распределенной сетевой вычислительной системы

В настоящей работе поставлена задача реализации управляемой приложением функциональной архитектуры распределенной сетевой вычислительной системы ADFA-DCS (англ. *Application-Driven Functional Architecture of Distributed Computing System*), позволяющей реализовать схему взаимодействия компьютеров при выполнении распределенных вычислений согласно концептуальному графу распределенного приложения. Приложение должно решать такие задачи, как реализация любой схемы взаимодействия удаленных узлов между собой и возможность выполнения определенной задачи на конкретном узле. Реализация таким образом определенной архитектуры затрагивает прикладной (*application*), промежуточный (*middleware*) и транспортный (*transport*) уровни P2P-сети. Начальное описание технологии ADFA-DCS было дано в работе [19].

Сходство технологий ADFA-DCS и SD-WAN основано на том, что они обе реализуются в компьютерных сетях и, естественно, могут быть интегрированы. Однако функциональная архитектура РВС, создаваемая на основе технологии ADFA, предназначена не для решения проблем, связанных с маршрутизацией, балансировкой нагрузки и поддержки политик безопасности для всей сети, т.е. не для организации эффективного управления сетью, а для поддержки технологии прикладного (*application*) распределенного программирования в сетевой среде, обеспечивающего хранение и обработку данных на пиринговых узлах и задействующего при выполнении одного и того же приложения множество узлов на основе механизма передачи сообщений (англ. *Message Passing Mechanism*).

Поддержка технологии ADFA-DCS может осуществляться как на прикладном уровне при возможном доступе к некоторым протоколам транспорт-

ного уровня, так и на уровне промежуточного программного обеспечения (*middleware*).

Существенным отличием предлагаемого подхода от известных является использование технологии непосредственной реализации распределенного алгоритма в форме промежуточного сетевого программного обеспечения MOM (англ. *Message-Oriented Middleware*), т.е. обеспечения слоя, ориентированного на обработку сообщений и занимающего подуровень AD MOM – *Application-Driven Message-Oriented Middleware*, возможно, непосредственно над подуровнем SD-WAN *Middleware.* За отправную точку принята реализация функциональной архитектуры РВС, задаваемой приложением (англ. *Application-Driven Functional Architecture*). Системная архитектура РВС реализуется на платформе произвольной глобальной или локальной компьютерной сети, например, на базе TCP/IP сети.

Определение процедурной составляющей модели представления знаний о функционировании РВС

Концепция использования декларативного подхода к созданию функциональной архитектуры, т.е. системного программного обеспечения (ПО) РВС на уровне *middleware*, была предложена в работе [19]. Недостатком подобного подхода является то, что проектирование основано на использовании фактов, полученных на основе анализа концептуального графа (КГ) распределенного алгоритма. Правила реализуются на основе анализа фактов непосредственно при составлении программы. Процедурная составляющая модели представления знаний, такой как КГ, выражена неявно, и полученные факты трудно использовать при программировании распределенного приложения, поскольку правила вывода придется составлять вручную, используя предикаты следования. В настоящей работе предложено использовать подход, основанный на применении интегрированной модели представления знаний – концептуальной логической сети Петри (КЛ СП), в которой явно представлены как декларативная, так и процедурная части [20, 21]. На рис. 1 представлена КЛ СП, построенная на основе КГ для распределенного алгоритма.

Здесь процедурная составляющая модели представления знаний задана явно при помощи переходов сети Петри, «встроенных» непосредственно в концептуальный граф.

Операторы-позиции A_0, A_1, …, A_{14} представлены концептами. Аналогично концептами представлены пиринговые узлы Y_0, Y_1, …, Y_8.

В графе G_1 ромбами представлены переходы T_0, T_1, …, T_{10}, имена которых начитаются с буквы T; имя $T^{\&\&}$ означает, что на входе и выходе перехода реализуется конъюнктивная логика, T** – что на входе и выходе реализуется дизъюнктивная логика. Предполагается, что вычисление значения 1, 2 или 3 параметра k (что соответствует истинности одного из условий α_1, α_2 или α_3) осуществляется при выполнении оператора *A9*. Имя *D* используется для обозначения отношения развертывания операторов распределенного алгоритма по узлам P2P-сети. Унарный предикат R задает начальную разметку КЛ СП, а именно оператора-позиции A_0.

Интеллектуальный редактор концептуальных графов позволяет автоматически получить не только факты, но и правила срабатывания переходов КЛ СП. Факты и правила позволяют сформировать содержимое конфигурационного файла сетевого приложения (табл. 1).

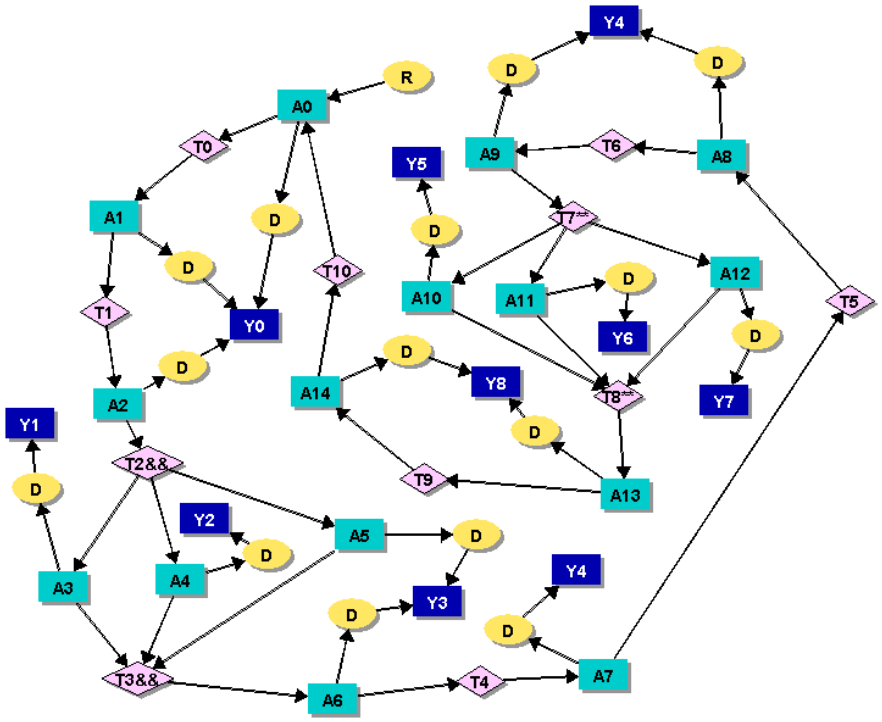

Рис. 1. Концептуальная логическая сеть Петри для распределенного приложения, реализуемого в P2P-сети (граф G_1)

Таблица 1

Факты, правила и содержимое конфигурационного файла

Facts Факты	Rules Правила	Содержимое конфигурационного файла сетевого приложения
an A0 is the R		
a Y0 is the D of an A0;	**an A0 results from T10 of an A14;**	**A0; Y0; A14; A1**
a Y0 is the D of an A1;	**an A1 results from T0 of an A0;**	**A1; Y0; A0; A2**
a Y0 is the D of an A2;	**an A2 results from T1 of an A1;**	**A2; Y0; A1; A3 / A4 / A5**
a Y1 is the D of an A3;	**an A3 and an A4 and an A5 results from T2&& of an A2;**	**A3; Y1; A2; A6**
a Y2 is the D of an A4;		**A4; Y2; A2; A6**
a Y3 is the D of an A5;		**A5; Y3; A2; A6**
a Y3 is the D of an A6;	**an A6 results from T3&& of an A3 and an A4 and an A5;**	**A6; Y3; A3 / A4 / A5; A7**
a Y4 is the D of an A7;	**an A7 results from T4 of an A6;**	**A7; Y4; A6; A8**
a Y4 is the D of an A8;	**an A8 results from T5 of an A7;**	**A8; Y4; A7; A9**
a Y4 is the D of an A9;	**an A9 results from T6 of an A8;**	**A9; Y4; A8; A10, A11, A12**
a Y5 is the D of an A10;	**an A10 or an A11 or an A12 results from T7** of an A9;**	**A10; Y5; A9; A13**
a Y6 is the D of an A11;		**A11; Y6; A9; A13**
a Y7 is the D of an A12;		**A12;Y7;A9;A13**
a Y8 is the D of an A13;	**an A13 results from T8** of an A10 or an A10 or an A12;**	**A13; Y8; A10, A11, A12; A14**
a Y8 is the D of an A14;	**an A14 results from T9 of an A13;**	**A14; Y8; A13; A0**

Для хранения текстового представления концептуального графа в приложении выбран текстовый файл. Приложение способно реализовывать произвольные схемы взаимодействия узлов сети; для изменения схемы взаимодействия достаточно изменить конфигурационный файл [19]. Данные формируются и записываются в файл по следующей схеме:

“[Оператор] ; [Узел] ; [Родительские операторы] ; [Дочерние операторы]”.

В случае отсутствия родительских или дочерних операторов допускается ничего не указывать на месте соответствующих полей, тогда формат записи данных имеет следующий вид: **“*A*0; *Y*0; ;”**.

Пропуск полей оператора и узла не допускается.

2. Формализация декларативно-императивной и логико-алгебраической модели распределенных вычислений

Формализованные сетевые спецификации Σ_1 *для пиринговой сети с равноправными участниками*

На основании КЛ СП G_1 (см. рис. 1 и табл. 1) получена следующая система Σ_1 продукционных правил, составленная с использованием логики предикатов первого порядка (здесь и далее символы операторов A заменены на символы позиций P с сохранением индексов).

Система $\boldsymbol{\Sigma_1}$:

$$T_0: Start \,\&\, R(P_0) \,\&\, \neg\, R(P_1) \to \neg\, R(P_0) \,\&\, R(P_1);$$

$$T_1: R(P_1) \,\&\, \neg\, R(P_2) \to \neg\, R(P_1) \,\&\, R(P_2);$$

$$T_2^{\&\&}: R(P_2) \,\&\, \neg\, R(P_3) \,\&\, \neg\, R(P_4) \,\&\, \neg\, R(P_5) \to$$

$$\to \neg\, R(P_2) \,\&\, R(P_3) \,\&\, R(P_4) \,\&\, R(P_5);$$

$$T_3^{\&\&}: R(P_3) \,\&\, R(P_4) \,\&\, R(P_5) \,\&\, \neg\, R(P_6) \to$$

$$\to \neg\, R(P_3) \,\&\, \neg\, R(P_4) \,\&\, \neg\, R(P_5) \,\&\, R(P_6).$$

$$T_4: R(P_6) \,\&\, \neg\, R(P_7) \to \neg\, R(P_6) \,\&\, R(P_7);$$

$$T_5: R(P_7) \,\&\, \neg\, R(P_8) \to \neg\, R(P_7) \,\&\, R(P_8);$$

$$T_6: R(P_8) \,\&\, \neg\, R(P_9) \to \neg\, R(P_8) \,\&\, R(P_9);$$

$$T_7^{**}: R(P_9) \,\&\, \neg\, R(P_{10}) \,\&\, \neg\, R(P_{11}) \,\&\, \neg\, R(P_{12}) \to$$

$$\to \neg\, R(P_9) \,\&\, (\alpha_{10} \,\&\, R(P_{10}) \vee \alpha_{11} \,\&\, R(P_{11}) \vee \alpha_{12} \,\&\, R(P_{12}));$$

$$T_8^{**}: (\alpha_{10} \,\&\, R(P_{10}) \,\&\, \neg\, R(P_{13}) \to R(P_{13}) \,\&\, \neg\, R(\mathrm{P}_{10})) \vee$$

$$\vee\, (\alpha_{11} \,\&\, R(P_{11}) \,\&\, \neg\, R(P_{13}) \to R(P_{13}) \,\&\, \neg\, R(P_{11})) \vee$$

$$\vee\, (\alpha_{12} \,\&\, R(P_{12}) \,\&\, \neg\, R(P_{13}) \to R(P_{13}) \,\&\, \neg\, R(P_{12}));$$

$$T_9: R(P_{13}) \,\&\, \neg\, R(P_{14}) \to \neg\, R(P_{13}) \,\&\, R(P_{14});$$

$$T_{10}: R(P_{14}) \,\&\, \neg\, R(P_0) \to \neg\, R(P_{14}) \,\&\, R(P_0).$$

Символ импликации «→» разделяет антецедент-посылку (слева) и консеквент-следствие (справа) в продукционных правилах. Фактически данная система продукционных правил доопределяет функциональную архитектуру РВС, реализация которой рассмотрена в работе [19]. Условно систему Σ_1 продукционных правил можно отнести к исполнимым моделям в том смысле, что она может быть основой при программировании сетевого приложения для РВС. Однако такая система остается сложной при программировании, так как требуется учесть многие неформализованные особенности сетевого программирования. Поэтому при рассмотрении следующего примера уделено особое внимание уделяется представлению сетевых команд в рамках формализованных спецификаций.

Формализованные сетевые спецификации для пиринговой сети с равноправными участниками и выделенным узлом-диспетчером

Вариант логической системной архитектуры РВС на базе P2P-сети с выделенным узлом-диспетчером *Comm* представлен на рис. 2. Выделенный узел-диспетчер не является сервером, он участвует во всех обменах сообщениями и данными между узлами-участниками P2P-сети, что в целом, как было сказано ранее, не противоречит идее пиринговых вычислений.

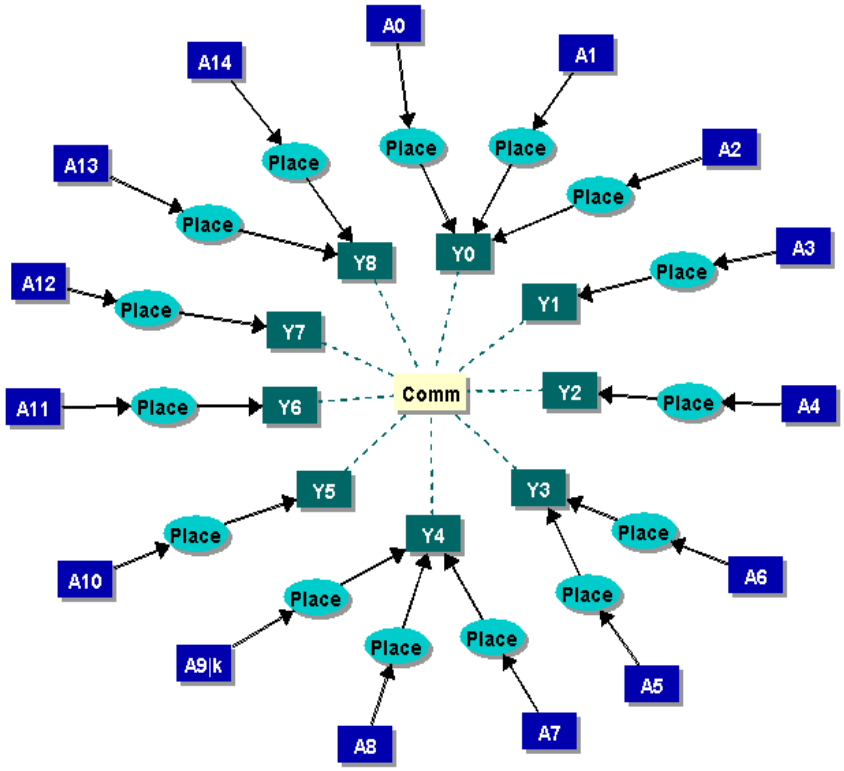

Рис. 2. Логическая системная архитектура централизованной пиринговой сети с равноправными участниками и выделенным узлом-диспетчером

Концептуальная логическая сеть Петри для распределенного приложения, реализуемого в P2P-сети с центральным коммутатором – диспетчером узлов *Comm* (граф G_2), представлена на рис. 3.

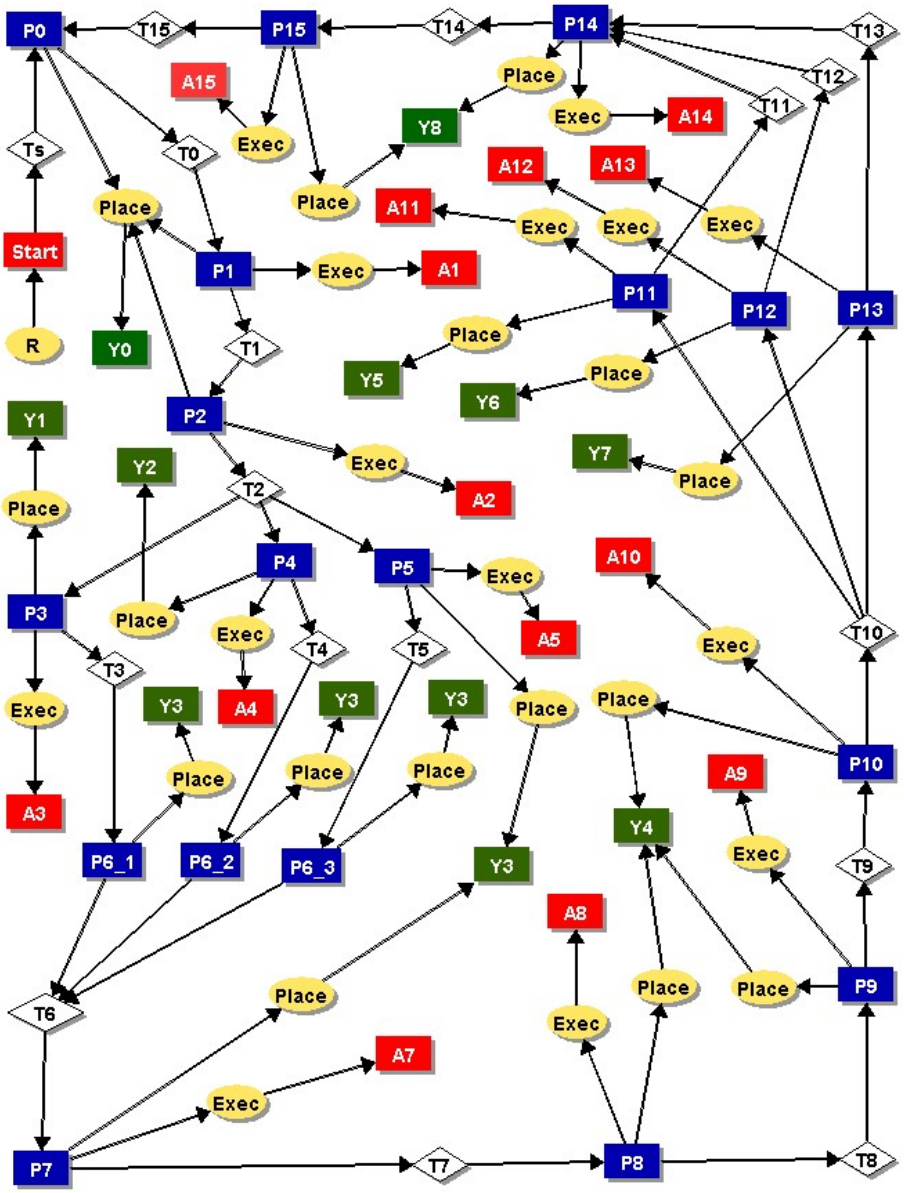

Рис. 3. Концептуальная логическая сеть Петри для распределенного приложения, реализуемого в P2P-сети с центральным коммутатором – диспетчером узлов *Comm* (граф G_2)

Работа сети начинается со срабатывания актора-перехода *Ts*. Унарный предикат *R* задает начальную разметку позиции *Start*. Далее сообщения перемещаются по сети в соответствием с распределенным алгоритмом. Узлы Y_0, Y_1, …, Y_8 в P2P-сети равнодоступны друг другу через диспетчер узлов *Comm*. На узлах размещены модули-задачи A_0, A_1, …, A_{14}, которые при описании концептуального графа называются операторами-концептами. Бинарный предикат *Exec* связывает позиции P_0, P_1, …, P_{15} с операторами-концептами. Бинарный предикат *Place* связывает позиции с узлами сети, на которых они расположены.

Акторы-переходы T_0, T_1, …, T_{15} реализуются программно. Размещение модулей-задач и программ акторов-переходов T_0, T_1, …, T_{15} учитывается при реализации распределенного приложения. Число узлов и программных модулей соответствует распределенному приложению, которое реализовано реально и рассмотрено в качестве примера. При дальнейшем описании КЛ СП, как это нередко делается, используется отождествление объектов РВС и распределенного приложения с самой моделью.

Акторы-переходы и позиции в графе G_2 задают императивную, или процедурную, составляющую модели представления знаний о работе РВС. Правила срабатывания акторов-переходов реализуются программно в соответствии с логико-алгебраическими операционными выражениями (ЛАОВ). Эти выражения построены на основании системы продукционных правил и являются декларативно-императивной моделью представления знаний о функционировании РВС. В графе G_2 на рис. 3 не учтено наличие в РВС узла-диспетчера *Comm*, чтобы не загромождать рисунок. Наличие узла-диспетчера будет учтено непосредственно в системе ЛАОВ Σ_2.

Система $\boldsymbol{\Sigma_2}$:

$$Ts/P_0\colon [R^*(Start)]\,(R^*(P_0) \leftarrow \mathbf{true} \oplus E);$$

$$T_0/P_0\colon [R(P_0)](Exec(P_0, A_0) \leftarrow D_{A0},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_0, P_{Com}) \leftarrow D_{A0},$$
$$Ack^*(P_{Com}, P_0) \leftarrow \mathbf{true}, Msg(P_{Com}, P_1) \leftarrow D_{A0}, R^*(P_1) \leftarrow \mathbf{true},$$
$$Ack^*(P_1, P_{Com}) \leftarrow \mathbf{true}), Q^{**}(P_{Com}) \leftarrow \mathbf{false},$$
$$R(P_0) \leftarrow \neg\, R(P_0) \ \oplus E);$$

$$T_1/P_1\colon [R(P_1)](Exec(P_1, A_1) \leftarrow D_{A1},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_1, P_{Com}) \leftarrow D_{A1},$$
$$Ack^*(P_{Com}, P_1) \leftarrow \mathbf{true}, Msg(P_{Com}, P_2) \leftarrow D_{A1}, R^*(P_2) \leftarrow \mathbf{true},$$
$$Ack^*(P_2, P_{Com}) \leftarrow \mathbf{true}), Q^{**}(P_{Com}) \leftarrow \mathbf{false}, R(P_1) \leftarrow \neg\, R(P_1) \ \oplus E);$$

$$T_2/P_2\colon [R(P_2)](Exec(P_2, A_2) \leftarrow D_{A2},$$
$$([Q^{**}(P_{Com})]\{E\}(Msg(P_2, P_{Com}) \leftarrow D_{A2},$$
$$Ack^*(P_{Com}, P_2) \leftarrow \mathbf{true}, Msg(P_{Com}, P_3) \leftarrow D_{A2}, R^*(P_3) \leftarrow \mathbf{true},$$
$$Ack^*(P_3, P_{Com}) \leftarrow \mathbf{true}, Q^{**}(P_{Com}) \leftarrow \mathbf{false})),$$
$$([Q^{**}(P_{Com})]\{E\}(Msg(P_2, P_{Com},) \leftarrow D_{A2}, Ack^*(P_{Com}, P_2) \leftarrow \mathbf{true},$$
$$Msg(P_{Com}, P_4) \leftarrow D_{A2}, R^*(P_4) \leftarrow \mathbf{true},$$
$$Ack^*(P_4, P_{Com}) \leftarrow \mathbf{true}, Q^{**}(P_{Com}) \leftarrow \mathbf{false})),$$

$$([Q^{**}(P_{Com})]\{E\}(Msg(P_2, P_{Com}) \leftarrow D_{A2},$$
$$Ack^{*}(P_{Com}, P_2) \leftarrow \textbf{true}, Msg(P_{Com}, P_5) \leftarrow D_{A2}, R^{*}(P_5) \leftarrow \textbf{true},$$
$$Ack^{*}(P_5, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false})),$$
$$R(P_2) \leftarrow \neg R(P_2) \oplus E);$$

$$T_3/P_3\colon [R(P_3)](Exec(P_3, A_3) \leftarrow D_{A3},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_3, P_{Com}) \leftarrow D_{A3},$$
$$Ack^{*}(P_{Com}, P_3) \leftarrow \textbf{true}, Msg(P_{Com}, P_{6,1}) \leftarrow D_{A3}, R^{*}(P_{6,1}) \leftarrow \textbf{true},$$
$$Ack^{*}(P_{6,1}, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_3) \leftarrow \neg R(P_3) \oplus E);$$
$$T_4/P_4\colon [R(P_4)](Exec(P_4, A_4,) \leftarrow D_{A4},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_4, P_{Com}) \leftarrow D_{A4}, Ack^{*}(P_{Com}, P_4) \leftarrow \textbf{true},$$
$$Msg(P_{Com}, P_{6,2}) \leftarrow D_{A4}, R^{*}(P_{6,2}) \leftarrow \textbf{true},$$
$$Ack^{*}(P_{6,2}, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_4) \leftarrow \neg R(P_4) \oplus E);$$

$$T_5/P_5\colon [R(P_5)](Exec(P_5, A_5) \leftarrow D_{A5},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_5, P_{Com}) \leftarrow D_{A5},$$
$$Ack^{*}(P_{Com}, P_5) \leftarrow \textbf{true}, Msg(P_{Com}, P_{6,3}) \leftarrow D_{A5}, R^{*}(P_{6,3}) \leftarrow \textbf{true},$$
$$Ack^{*}(P_{6,3}, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_5) \leftarrow \neg R(P_5) \oplus E);$$

$$T_6/P_6\colon [R^{*}(P_{6,1}) \,\&\, R^{*}(P_{6,2}) \,\&\, R^{*}(P_{6,3})](Exec(P_{6,1}, D_{A3}) \leftarrow D_{6,1},$$
$$Exec(P_{6,2}, D_{A4}) \leftarrow D_{6,2}, Exec(P_{6,3}, D_{A5}) \leftarrow D_{6,3},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{6,1}, P_{Com}) \leftarrow D_{6,1},$$
$$Msg(P_{6,2}, P_{Com}) \leftarrow D_{6,2}, Msg(P_{6,3}, P_{Com}) \leftarrow D_{6,3},$$
$$Ack^{*}(P_{Com}, P_{6,1}) \leftarrow \textbf{true}, Ack^{*}(P_{Com}, P_{6,2}) \leftarrow \textbf{true},$$
$$Ack^{*}(P_{Com}, P_{6,3}) \leftarrow \textbf{true}, Msg(P_{Com}, P_7) \leftarrow D_{6,1},$$
$$Msg(P_{Com}, P_7) \leftarrow D_{6,2}, Msg(P_{Com}, P_7) \leftarrow D_{6,3},$$
$$R^{*}(P_7) \leftarrow \textbf{true}, Ack^{*}(P_7, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}) \oplus E);$$

$$T_7/P_7\colon [R(P_7)](Exec(P_7, A_7,) \leftarrow D_{A7},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_7, P_{Com}) \leftarrow D_{A7},$$
$$Ack^{*}(P_{Com}, P_7) \leftarrow \textbf{true}, Msg(P_{Com}, P_8) \leftarrow D_{A7}, R^{*}(P_8) \leftarrow \textbf{true},$$
$$Ack^{*}(P_8, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_7) \leftarrow \neg R(P_7) \oplus E);$$

$$T_8/P_8\colon [R(P_8)](Exec(P_8, A_8) \leftarrow D_{A8},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_8, P_{Com}) \leftarrow D_{A8},$$
$$Ack^{*}(P_{Com}, P_8) \leftarrow \textbf{true}, Msg(P_{Com}, P_9) \leftarrow D_{A8}, R^{*}(P_9) \leftarrow \textbf{true},$$
$$Ack^{*}(P_9, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_8) \leftarrow \neg R(P_8) \oplus E);$$

$$T_9/P_9\colon [R(P_9)](Exec(P_9, A_9) \leftarrow D_{A9},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_9, P_{Com}) \leftarrow D_{A9},$$
$$Ack^{*}(P_{Com}, P_9) \leftarrow \textbf{true}, Msg(P_{Com}, P_{10}) \leftarrow D_{A9}, R^{*}(P_{10}) \leftarrow \textbf{true},$$
$$Ack^{*}(P_{10}, P_{Com}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_9) \leftarrow \neg R(P_9) \oplus E);$$

$$T_{10}/P_{10}\colon [R(P_{10})](Exec(P_{10}, A_{10}) \leftarrow D_{A10},$$
$$[\alpha_1]([Q^{**}(P_{Com})]\{E\}(Msg(P_{10}, P_{Com}) \leftarrow D_{A10}, Ack^{*}(P_{Com}, P_{10}) \leftarrow \textbf{true},$$
$$Msg(P_{Com}, P_{11}, D_{A10}) \leftarrow D_{A10}, Ack^{*}(P_{11}, P_{Com}) \leftarrow \textbf{true},$$
$$R^{*}(P_{11}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}) \oplus$$
$$\oplus ([\alpha_2]([Q^{**}(P_{Com})]\{E\}(Msg(P_{10}, P_{Com}) \leftarrow D_{A10}, Ack^{*}(P_{Com}, P_{10}) \leftarrow \textbf{true},$$
$$Msg(P_{Com}, P_{12}) \leftarrow D_{A10}, Ack^{*}(P_{12}, P_{Com}) \leftarrow \textbf{true},$$
$$R^{*}(P_{12}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}) \oplus$$
$$\oplus ([\alpha_3]([Q^{**}(P_{Com})]\{E\}(Msg(P_{10}, P_{Com}) \leftarrow D_{A10}, Ack^{*}(P_{Com}, P_{10}) \leftarrow \textbf{true},$$
$$Msg(P_{Com}, P_{13}) \leftarrow D_{A10}, Ack^{*}(P_{13}, P_{Com}) \leftarrow \textbf{true}, R^{*}(P_{13}) \leftarrow \textbf{true},$$
$$Q^{**}(P_{Com}) \leftarrow \textbf{false}) \oplus E))), R(P_{10}) \leftarrow \neg R(P_{10}) \oplus E);$$

$$T_{11}/P_{11}\colon [R(P_{11})](Exec(P_{11}, A_{11}) \leftarrow D_{A11},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{11}, P_{Com}) \leftarrow D_{A11},$$
$$Ack^{*}(P_{Com}, P_{11}) \leftarrow \textbf{true}, Msg(P_{Com}, P_{14}) \leftarrow D_{A11},$$
$$Ack^{*}(P_{14}, P_{Com}) \leftarrow \textbf{true}, R^{*}(P_{14}) \leftarrow \textbf{true},$$
$$Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_{11}) \leftarrow \neg R(P_{11}) \oplus E);$$

$$T_{12}/P_{12}\colon [R(P_{12})](Exec(P_{12}, A_{12}) \leftarrow D_{A12},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{12}, P_{Com}) \leftarrow D_{A12},$$
$$Ack^{*}(P_{Com}, P_{12}) \leftarrow \textbf{true}, Msg(P_{Com}, P_{14}) \leftarrow D_{A12},$$
$$Ack^{*}(P_{14}, P_{Com}) \leftarrow \textbf{true}, R^{*}(P_{14}) \leftarrow \textbf{true},$$
$$Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_{12}) \leftarrow \neg R(P_{12}) \oplus E);$$

$$T_{13}/P_{13}\colon [R(P_{13})](Exec(P_{13}, A_{13}) \leftarrow D_{A13},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{13}, P_{Com}) \leftarrow D_{A13},$$
$$Ack^{*}(P_{Com}, P_{13}) \leftarrow \textbf{true}, Msg(P_{Com}, P_{14}) \leftarrow D_{A13},$$
$$Ack^{*}(P_{14}, P_{Com}) \leftarrow \textbf{true}, R^{*}(P_{14}) \leftarrow \textbf{true},$$
$$Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_{13}) \leftarrow \neg R(P_{13}) \oplus E);$$

$$T_{14}/P_{14}\colon [R(P_{14})](Exec(P_{14}, A_{14}) \leftarrow D_{A14},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{14}, P_{Com}) \leftarrow D_{A14},$$
$$Ack^{*}(P_{Com}, P_{14}) \leftarrow \textbf{true}, Msg(P_{Com}, P_{15}) \leftarrow D_{A14},$$
$$Ack^{*}(P_{15}, P_{Com}) \leftarrow \textbf{true}, R^{*}(P_{15}) \leftarrow \textbf{true},$$
$$Q^{**}(P_{Com}) \leftarrow \textbf{false}), R(P_{15}) \leftarrow \neg R(P_{15}) \oplus E);$$

$$T_{15}/P_{15}\colon [R(P_{15})](Exec(P_{15}, A_{15},) \leftarrow D_{A15},$$
$$[Q^{**}(P_{Com})]\{E\}(Msg(P_{15}, P_{Com}) \leftarrow D_{A15},$$
$$Ack^{*}(P_{Com}, P_{15}) \leftarrow \textbf{true}, Q^{**}(P_{Com}) \leftarrow \textbf{false}),$$
$$R(P_{15}) \leftarrow \neg R(P_{15}) \oplus E).$$

При составлении приведенных выражений использована следующая нотация (из алгебры алгоритмов Глушкова) для оператора, подобного оператору *if* в процедурно-ориентированных языках:

$$[\alpha](A \oplus B).$$

Данное выражение означает, что при истинности условия α выполняется действие A, а при его ложности – действие B. Данный оператор может использоваться в следующей частной форме:

$$[\alpha](A \oplus E),$$

где E – «пустое» действие.

Оператору цикла соответствует следующее выражение:

$$[\alpha]\{A\}.$$

Это означает, что после каждой проверки условия α, если оно оказывалось ложным, циклически выполняется оператор A до тех пор, пока условие α не станет истинным.

Развернутые комментарии к системе Σ_2 логико-алгебраических операционных выражений

Используемые ЛАОВ относятся к классу непосредственно исполнимых моделей и реализуются акторами-переходами в виде программных модулей. Условно будем считать, что эти модули «размещаются» на концептах-

позициях. Работа сетевого приложения начинается с запуска программного модуля, реализующего выражение для перехода *Ts*:

$$Ts/P_0\text{: } [R^*(Start)]\ (R^*(P_0) \leftarrow \mathbf{true} \oplus E),$$

где запись T_s/P_0 означает, что актор-переход T_s реализуется на концепте-позиции P_0. Далее слово «концепт» будет опускаться. Запись $[R^*(Start)]$ означает, что запуск приложения осуществляется при $R^*(Start)$ = **true**; символ звездочки указывает на то, что после успешной проверки истинного значения высказывания $R^*(Start)$ на истинность его значение «по умолчанию» становится ложным. Запись $R^*(P_0) \leftarrow$ **true** означает активизацию позиции P_0, что приводит к передаче управления программному модулю, реализующему следующее выражение T_0/P_0 и размещенному на той же позиции P_0. Затем истинное значение высказывания $R^*(P_0)$ «сбрасывается», т.е. оно становится ложным, на что указывает по умолчанию символ звездочки.

В процессе программной интерпретации следующего выражения T_0/P_0 выполняются следующие действия. Запись $[R^*(P_0)]$ означает, что в модуле реализуется ожидание истинного значения высказывания $R^*(P_0)$, которое поступает из модуля Ts/P_0. Звездочка здесь и далее имеет описанный ранее смысл. При интерпретации выражения $Exec(P_0, A_0) \leftarrow D_{A0}$ в модуле T_0/P_0 на позиции P_0 происходит выполнение оператора A_0 и формируется результат D_{A0}. Стрелка соответствует операции присваивания нового значения бинарной функции *Exec*. Записи $[Q^{**}(P_{Com})]\{E\}$ соответствует процесс ожидания состояния готовности диспетчера *Comm* пиринговой РВС к приему сообщения с результатом D_{A0}, после чего выполняется операция $Msg(P_0, P_{Com}) \leftarrow D_{A0}$ передачи сообщения из позиции P_0 в позицию P_{Com}. Здесь *Msg* – бинарная функция. Две звездочки у символа Q означают, что от позиции P_0 модуля T_0/P_0 циклически направляются опрашивающие сообщения к позиции P_{Com} до тех пор, пока не будет получено сообщение о готовности диспетчера *Comm* к приему информационного сообщения с результатом D_{A0}. Оператору $Ack^*(P_{Com}, P_0) \leftarrow$ **true** соответствует получение позицией P_0 квитирующего сообщения от позиции P_{Com}.

При программной интерпретации выполнения операции $Msg(P_{Com}, P_1) \leftarrow$ $\leftarrow D_{A0}$ результат D_{A0} передается от позиции P_{Com} к следующей позиции P_1, реализуемой модулем T_1/P_1. Затем при помощи операции $R^*(P_1) \leftarrow$ **true** управление от модуля T_0/P_0 через позицию P_{Com} передается модулю T_1/P_1 с последующей посылкой квитирующего сообщения от позиции P_1 к позиции P_{Com}. Последнее действие реализуется путем интерпретации операции $Ack^*(P_1, P_{Com}) \leftarrow$ **true**. Операции $Q^{**}(P_{Com}) \leftarrow$ **false** – освобождение позиции P_{Com} диспетчера, и $R(P_0) \leftarrow \neg R(P_0)$ – для исключения повторного выполнения модуля T_0/P_0.

В заключение комментариев отмечается, что во всех выражениях для системы ЛАОВ включение в качестве аргумента в предикате или функции позиции P_{Com} означает, что необходимо, в зависимости от окружающего контекста, либо послать запрос $Q^{**}(P_{Com})$ о состоянии узла-диспетчера *Comm* и принять от него ответ, либо послать от какого-либо модуля с реализованной позицией P_0 сообщение с данными (например, $Msg(P_0, P_{Com})$ для последующей передачи очередному модулю (например, $Msg(P_{Com}, P_1)$.

Логико-алгебраические операционные выражения предназначены для программирования декларативно-императивной составляющей представления знаний о работе распределенного пирингового приложения для РВС. Полученное приложение соответствует функциональной архитектуре РВС. Непосредственная исполнимость ЛАОВ означает на практике, что только они могут быть последним этапом сетевой технологии, предшествующим кодированию программы для распределенного приложения.

На этом теоретическая часть настоящей работы окончена. Главным отличием предлагаемой методики является непосредственная исполнимость формализованных спецификаций, основанных на ЛАОВ, что ускоряет создание управляемых приложениями функциональных архитектур сетевых РВС. Вопросы проектирования РВС на основе концептуального подхода рассматривались также в работах [15–21], в которых не ставилась задача обеспечения управляемости приложениями функциональных архитектур, но создан базис для настоящей работы.

3. Разработка распределенного пирингового приложения

Разработка программного обеспечения сети пиринговых узлов для выполнения распределенных вычислений разделяется на следующие этапы:

– разработка диспетчера узлов;

– разработка логики работы узла;

– разработка логики работы оператора.

Разработка диспетчера узлов

Диспетчер узлов предназначен для временного хранения передаваемых сообщений, заданий, данных и должен выполнять следующие задачи:

– загружать задачу для пиринговой сети из файла конфигурации;

– принимать подключения от доступных узлов сети;

– рассылать задачи подключенным узлам;

– принимать подключения от операторов на узлах;

– обрабатывать запросы от операторов на получение адресов других операторов;

– отслеживать статус работы операторов.

Схема концептуального графа заносится в текстовый файл и находится на сервере. На сервере реализован класс TaskParser, который осуществляет загрузку и разбор файла конфигурации:

```
class TaskParser
{
public:
    TaskParser( const std::string& configFilepath );
    std::vector<Task> GetTasks();
private:
    bool ParseString(const std::string& strTask, AgentData& agent, std::string& nodeName);
private:
    std::ifstream configFile_;
};
```

Метод «*TaskParser*::*ParseString*» разбирает строку текстового файла и заполняет данными структуру «*AgentData*». Метод «*TaskParser*::*GetTasks*» возвращает список задач в виде вектора структур «*Task*»:

```
struct Task
{
    std::string nodeName;
    int numAgents = 0;
    std::vector<AgentData> agents;
    friend class boost::serialization::access;
    template<typename Archive>
    void serialize( Archive& ar, const unsigned version )
    {
        ar& nodeName& numAgents& agents;
    }
};
```

Для сетевого взаимодействия с узлами в программе диспетчера узлов реализован класс «*ServerSocket*». В данном классе реализованы методы для выполнения основных задач сетевого взаимодействия, таких как прослушивание подключений, отправка и прием сообщений, корректное закрытие соединения.

Класс «*ServerSocket*» включает в себя методы для установления и поддержания соединения с удаленным узлом:

– «*ServerInit*» – метод для первоначальной инициализации диспетчера узлов;

– «*Listen*» – метод, осуществляющий запуск прослушивания новых подключений удаленных узлов;

– «*CloseClientSockets*» – метод, осуществляющий закрытие соединения с удаленными узлами;

– «*BroadcastMessage*» – метод, осуществляющий рассылку всем подключенным удаленным узлам сигнала о завершении процесса распределенных вычислений успешно или с ошибкой;

– «*Send*» – метод для отправки данных;

– «*Recieve*» – метод для приема данных.

Основная логика взаимодействия с удаленным узлом заключается в отправке ему задачи и реализована в функции «*InitNode*».

Взаимодействие диспетчера агентов с конкретным оператором реализовано в функции «*AgentHandler*» и заключается в обработке запросов от оператора. Диспетчер узлов может обработать сигнал о завершении вычислений успешно или с ошибкой, а также запрос на получение сетевого адреса оператора.

Разработка логики работы удаленного узла

Логика узла выполняется в основном потоке приложения. Основной задачей разработки программной логики узла является соединение с диспетчером узлов и получение от него задачи.

В задаче указывается количество операторов, выполняемых на текущем узле. Когда узлу станет известна его задача, программа запустит требуемое количество операторов в отдельных потоках.

Реализация сетевого соединения с диспетчером узлов содержится в классе «*ClientSocket*». При создании нового объекта класса «*ClientSocket*» создается новый объект сокета, который хранится как приватная переменная

этого класса. Таким образом, один экземпляр класса «*ClientSocket*» представляет отдельную сетевую единицу и включает в себя методы для работы с сокетом. Методы класса схожи по названию и предназначению с методами класса «*ServerSocket*», рассмотренного выше, за исключением методов «*Connect*» – метод, осуществляющий подключение к серверу по необходимому сетевому адресу, и «*Disconnect*» – метод для корректного отсоединения от сервера.

Разработка логики работы оператора

Основной задачей оператора является прием управления от родительских операторов, обработка данных и дальнейшая передача управления дочерним операторам.

Весь программный модуль работы оператора реализован в классе «*Agent*». Основными методами класса являются следующие:

– «*AgentThread*» – основной метод класса, описывает логику работы оператора;

– «*WaitParents*» – метод класса, осуществляющий ожидания управления от родительских операторов;

– «*DelegateChilds*» – метод класса, выполняющий передачу управления дочерним операторам.

При создании объекта класса ему передается структура «*AgentData*». В ней содержится информация о положении текущего оператора в схеме распределенной вычислительной сети, информация о соседних операторах и номер порта, который будет прослушивать сокет текущего оператора.

Описание полей структуры приведено в табл. 2.

Таблица 2

Описание полей структуры «AgentData»

Название переменной	Описание
«agentName»	Имя агента
«listenPort»	Номер порта, который прослушивает сокет текущего агента
«isChildFork»	Логическая переменная, сигнализирующая о том, является ли переход к дочерним узлам параллельным
«isParentFork»	Логическая переменная, сигнализирующая о том, является ли переход от родительских операторов параллельным
«parentAgents»	Вектор, содержащий имена родительских операторов
«childAgents»	Вектор, содержащий имена дочерних операторов

Тестирование и отладка пирингового приложения на отдельном узле TCP/IP-сети

Работа каждого оператора при тестировании имитируется и сопровождается выводом данных об узле на экран консоли. Было проведено тестирование на виртуальной сети на одном компьютере без учета времени работы сети, а также тестирование на реальной локальной сети с учетом времени работы сети. По схеме соединения операторов в графе, приведенном на рис. 1, был создан конфигурационный файл приложения, содержание которого дано в третьей колонке табл. 1.

На начальном этапе тестирования был произведен запуск программы диспетчера узлов и программ самих узлов на одном компьютере. На рис. 3 был приведен вывод консоли для программы диспетчера узлов. После запуска диспетчер узлов разбирает конфигурационный файл и формирует структуру задач для каждого оператора. Далее диспетчер начинает ожидать подключения узлов. После того как все узлы подключены, начинается рассылка задач узлам.

Далее диспетчер узлов начинает ожидать подключения агентов (операторов). После того как все агенты подключились, диспетчер узлов переходит в режим ожидания. Он ожидает сигнала от каждого агента об успешном выполнении или появлении ошибки. После получения сигналов от всех агентов диспетчер узлов рассылает всем узлам сигнал об успешном или аварийном завершении. На этом вычисления завершаются.

Вывод консоли удаленного узла также содержит подробные сведения о его работе и приведен на рис. 4.

На рис. 5 приведен вывод консоли удаленного узла на примере узла Y_0. Порядок действий остальных улов является аналогичным. Первоначально происходит попытка установления соединения с диспетчером узлов. При успешном соединении узел переходит в режим ожидания задачи. Как только задача будет получена, узел запустит отдельные потоки для выполнения своих агентов (операторов), а сам перейдет в режим ожидания сигнала завершения от диспетчера узлов.

Жизненный цикл агента состоит из ожидания данных от родительских агентов, выполнения функционала самого оператора и передачи данных дочерним агентам. После выполнения агент сообщает диспетчеру узлов о своем успешном завершении или о возникновении ошибки.

Тестирование работы пирингового приложения в локальной TCP/IP-сети

Для проведения тестирования и определения среднего времени работы приложения в реальной локальной сети была разработана утилита тестирования. Приложение для тестирования вычислительной сети состоит из программы сервера и программы клиента. Основной задачей приложения для тестирования является запуск программы диспетчера узлов и программ удаленных узлов требуемое количество раз. Программа сервера предназначена только для тестирования работы приложения, и в рабочей версии ее присутствие не обязательно.

Для проведения тестов с использованием данной утилиты необходимо запустить программу сервера на одном из компьютеров. После запуска сервера для тестирования необходимо ввести количество удаленных узлов, используемых в работе распределенной сети и желаемое количество запусков. Далее сервер ожидает подключения программ клиентов.

На рис. 6 показан вывод консоли программы сервера для тестирования.

Далее необходимо запустить на каждом узле программу клиента для тестирования. После запуска требуемого числа клиентов и их успешного подключения сервер запустит диспетчер узлов и отправит сигнал клиентам, получив который, клиенты запустят программы удаленных узлов на своих устройствах. Далее выполнится логика работы распределенного приложения,

описанная выше. После завершения работы диспетчера узлов сервер для тестирования снова запустит диспетчер узлов и также разошлет сигналы запуска подключенным клиентам. Такой алгоритм повторится требуемое количество раз. На рис. 7 показан вывод консоли программы клиента для тестирования.

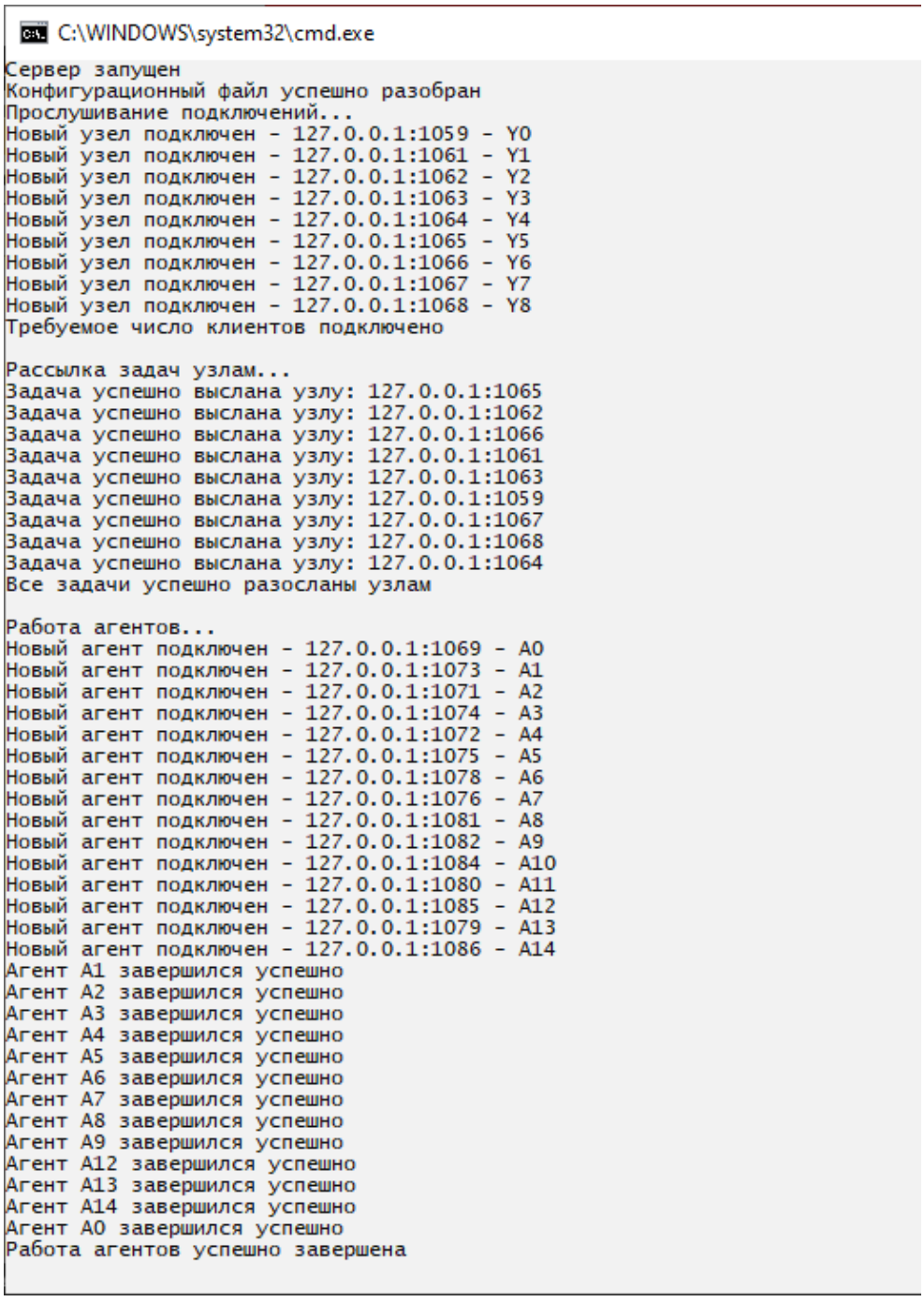

Рис. 4. Вывод консоли программы диспетчера узлов

```
Соединение с диспетчером узлов установлено

Ожидание задачи от диспетчера узлов
Задача получена

Узел: Y0
Операторы узла: A0 A1 A2

[ A0 ] Ожидание сигнала о начале вычислений от диспетчер узлов
[ A1 ] Ожидание данных от родительских узлов: A0
[ A2 ] Ожидание данных от родительских узлов: A1
[ A0 ] Сигнал старта получен
Оператор A0 успешно выполнен
[ A0 ] Отправка данных дочернем узлам
[ A0 ] Данные успешно отправлены дочерним узлам: A1
[ A0 ] Ожидание данных от родительских узлов: A14
[ A1 ] Данные получены: A0
Оператор A1 успешно выполнен
[ A1 ] Отправка данных дочернем узлам
[ A1 ] Данные успешно отправлены дочерним узлам: A2
[ A2 ] Данные получены: A1
Оператор A2 успешно выполнен
[ A2 ] Отправка данных дочернем узлам
[ A2 ] Данные успешно отправлены дочерним узлам: A3 A4 A5
[ A0 ] Данные получены: A14
Вычисления завершены успешно.
```

Рис. 5. Вывод консоли программы удаленного узла

```
Введите количество узлов в сети: 9
Введите желаемое количество запусков: 5
Ожидание подключения тестовых клиентов...
```

Рис. 6. Вывод консоли сервера для тестирования

```
Клиент успешно подключен
Запуск программы узла 1
Запуск программы узла 2
Запуск программы узла 3
Запуск программы узла 4
Запуск программы узла 5
Корректное завершение тестов
```

Рис. 7. Вывод консоли клиента для тестирования

Для успешной работы утилиты тестирования исполняемый файл диспетчера узлов и конфигурационный файл должны находиться в одном каталоге вместе с исполняемым файлом программы сервера для тестирования. В свою очередь исполняемый файл программы удаленного узла должен находиться в одном каталоге вместе исполняемым файлом программы клиента для тестирования.

Описание результатов тестирования в локальной TCP/IP-сети

Для тестирования пиринговой сети в реальных условиях было задействовано 10 компьютеров, объединенных в локальную TCP/IP-сеть. Один из компьютеров являлся сервером и осуществлял функцию управления. На него были помещены исполняемые файлы диспетчера узлов и программы сервера для тестирования, а также конфигурационный файл, содержащий текстовое представление графа. Остальные 9 компьютеров выступали в роли удаленных

узлов, и на них были помещены исполняемые файлы программы удаленных узлов и программы клиента для тестирования.

Ручной единичный запуск приложения показал успешную работу приложения с учетом реальной сетевой нагрузки на приложение. Вывод консоли диспетчера узлов при работе в локальной сети приведен на рис. 8.

```
Сервер запущен
Конфигурационный файл успешно разобран
Прослушивание подключений...
Новый узел подключен - 192.168.10.49:7609 - Y0
Новый узел подключен - 192.168.10.30:1300 - Y1
Новый узел подключен - 192.168.10.50:2191 - Y2
Новый узел подключен - 192.168.10.32:1286 - Y3
Новый узел подключен - 192.168.10.42:2153 - Y4
Новый узел подключен - 192.168.10.43:29740 - Y5
Новый узел подключен - 192.168.10.44:1398 - Y6
Новый узел подключен - 192.168.10.45:1789 - Y7
Новый узел подключен - 192.168.10.46:1434 - Y8
Требуемое число клиентов подключено

Рассылка задач узлам...
Задача успешно выслана узлу: 192.168.10.49:7609
Задача успешно выслана узлу: 192.168.10.30:1300
Задача успешно выслана узлу: 192.168.10.50:2191
Задача успешно выслана узлу: 192.168.10.32:1286
Задача успешно выслана узлу: 192.168.10.42:2153
Задача успешно выслана узлу: 192.168.10.43:29740
Задача успешно выслана узлу: 192.168.10.44:1398
Задача успешно выслана узлу: 192.168.10.45:1789
Задача успешно выслана узлу: 192.168.10.46:1434
Все задачи успешно разосланы узлам

Работа агентов...
Новый агент подключен - 192.168.10.49:7610 - A0
Новый агент подключен - 192.168.10.50:2192 - A1
Новый агент подключен - 192.168.10.49:7611 - A2
Новый агент подключен - 192.168.10.49:7612 - A3
Новый агент подключен - 192.168.10.43:29741 - A4
Новый агент подключен - 192.168.10.42:2154 - A5
Новый агент подключен - 192.168.10.44:1399 - A6
Новый агент подключен - 192.168.10.45:1790 - A7
Новый агент подключен - 192.168.10.42:2155 - A8
Новый агент подключен - 192.168.10.42:2156 - A9
Новый агент подключен - 192.168.10.46:1435 - A10
Новый агент подключен - 192.168.10.46:1436 - A11
Новый агент подключен - 192.168.10.30:1301 - A12
Новый агент подключен - 192.168.10.32:1288 - A13
Новый агент подключен - 192.168.10.32:1287 - A14
Агент A1 завершился успешно
Агент A2 завершился успешно
Агент A4 завершился успешно
Агент A3 завершился успешно
Агент A5 завершился успешно
Агент A6 завершился успешно
Агент A7 завершился успешно
Агент A8 завершился успешно
Агент A9 завершился успешно
Агент A12 завершился успешно
Агент A13 завершился успешно
Агент A14 завершился успешно
Агент A0 завершился успешно
Работа агентов успешно завершена
```

Рис. 8. Вывод консоли диспетчера узлов при работе в локальной сети

Далее было проведено 100 запусков приложения. Было получено время работы сети при каждом запуске и вычислено среднее время работы приложения с учетом реальных задержек сети. На рис. 9 показан вывод консоли программы сервера для тестирования.

```
Время работы сети при 60 запуске - 3.608682 секунд
Время работы сети при 61 запуске - 3.608135 секунд
Время работы сети при 62 запуске - 3.769100 секунд
Время работы сети при 63 запуске - 3.057100 секунд
Время работы сети при 64 запуске - 3.071273 секунд
Время работы сети при 65 запуске - 3.284292 секунд
Время работы сети при 66 запуске - 2.713378 секунд
Время работы сети при 67 запуске - 3.528337 секунд
Время работы сети при 68 запуске - 2.810711 секунд
Время работы сети при 69 запуске - 3.593142 секунд
Время работы сети при 70 запуске - 3.834504 секунд
Время работы сети при 71 запуске - 2.948805 секунд
Время работы сети при 72 запуске - 2.904773 секунд
Время работы сети при 73 запуске - 2.518278 секунд
Время работы сети при 74 запуске - 2.793271 секунд
Время работы сети при 75 запуске - 3.724942 секунд
Время работы сети при 76 запуске - 3.645432 секунд
Время работы сети при 77 запуске - 3.789274 секунд
Время работы сети при 78 запуске - 3.551512 секунд
Время работы сети при 79 запуске - 3.833070 секунд
Время работы сети при 80 запуске - 2.591254 секунд
Время работы сети при 81 запуске - 3.571357 секунд
Время работы сети при 82 запуске - 3.302580 секунд
Время работы сети при 83 запуске - 3.489278 секунд
Время работы сети при 84 запуске - 3.438564 секунд
Время работы сети при 85 запуске - 3.417757 секунд
Время работы сети при 86 запуске - 3.029633 секунд
Время работы сети при 87 запуске - 3.258103 секунд
Время работы сети при 88 запуске - 2.780892 секунд
Время работы сети при 89 запуске - 3.211951 секунд
Время работы сети при 90 запуске - 3.045681 секунд
Время работы сети при 91 запуске - 2.976229 секунд
Время работы сети при 92 запуске - 3.300312 секунд
Время работы сети при 93 запуске - 2.981066 секунд
Время работы сети при 94 запуске - 3.802705 секунд
Время работы сети при 95 запуске - 2.708492 секунд
Время работы сети при 96 запуске - 3.033126 секунд
Время работы сети при 97 запуске - 3.022849 секунд
Время работы сети при 98 запуске - 3.481389 секунд
Время работы сети при 99 запуске - 3.024569 секунд
Среднее время работы сети при 100 запусках - 3.227716 секунд
Корректное завершение тестов
```

Рис. 9. Вывод консоли программы сервера для тестирования

Все разработанные программы, в том числе программы, обеспечивающие работу P2P-сети, выполнены по предлагаемой технологии, созданной на основе декларативно-императивного и логико-алгебраического подходов. В ходе дальнейших экспериментов РВС на базе P2P-сети показала устойчивую работу. Модули-операторы работают согласованно и могут реализовывать произвольные алгоритмы, в том числе алгоритмы предварительно обученных искусственных нейронных сетей, реализуя концепцию «Сеть – это компьютер» в полном объеме.

Заключение

Предложена управляемая приложением функциональная архитектура пиринговой распределенной вычислительной системы, реализующая концепцию «Сеть – это компьютер». Предложены декларативно-императивный (де-

кларативно-процедурный) и логико-алгебраический подходы к формализованной спецификации распределенных вычислений, реализующих основные управляющие конструкции структурного программирования и парадигму управляемого сообщениями распределенного вычислительного процесса.

Предложено описание распределенных приложений новым видом семантических сетей – концептуальными графами сетей Петри, за счет чего реализовано оперативное управление настройкой пиринговой РВС на решение алгоритмически сложных задач; предполагается, что реализация данного подхода в перспективе способна обеспечить семантическую, синтаксическую и междоменную интероперабельность систем с различной организацией.

Реализована экспериментально управляемая приложением функциональная архитектура пиринговой распределенной вычислительной системы, основанная на настройке распределенного приложения «под задачу».

Список литературы

1. Manoj Kumar Mishra, Yashwant Singh Patel, Moumita Ghosh, Mund G. B. A Review and Classification of Grid Computing Systems // International Journal of Computational Intelligence Research. 2017. Vol. 13, № 3. P. 369–402.
2. Радченко Г. И. Распределенные вычислительные системы. Челябинск : Фотохудожник, 2012. 184 с.
3. Cabani A., Ramaswamy S., Itmi M., Haziqah Shukri S., Pecuchet J.-P. Distributed Computing Systems: P2P versus Grid Computing Alternatives // Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. 2007. P. 47–52. doi: 10.1007/978-1-4020-6268-1_9
4. Giridhar Kumar D., SubbaRao Ch. D. V. Peer-To-Peer Computing: Architectures, Applications and Challenges // International Journal of Recent Technology and Engineering (IJRTE). 2019. Vol. 8, № 1S4. P. 630–636.
5. Kahanwal B., Singh T. P. The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle // International Journal of Latest Research in Science and Technology. 2012. Vol. 1, № 2. P. 183–187.
6. Quang Hieu Vu, Mihai Lupu, Beng Chin Ooi. Peer-to-Peer Computing Principles and Applications. Springer Publication, 2010. doi: 10.1007/978-3-642-03514-2
7. SD-WAN Software Defined программно-определяемая WAN-сеть. 2022. URL: https://www.tadviser.ru/index.php/Статья:SD-WAN_(Software_Defined)_Программно-определяемая_WAN-сеть (дата обращения: 01.09.2023).
8. Zhenjie Yang, Yong Cui, Baochun Li, Yadong Liu, Yi Xu. Software-Defined Wide Area Network (SD-WAN): Architecture, Advances and Opportunities // 28th International Conference on Computer Communication and Networks (ICCCN). Valencia, Spain, 2019. P. 1–9. doi: 10.1109/ICCCN.2019.8847124
9. Youssfi M., Bouattane O., Bensalah M. A Parallel Computational Model Based on Mobile Agents for High Performance Computing // Contemporary Engineering Sciences. 2015. Vol. 8, № 15. P. 677–698. doi: 10.12988/ces.2015.55153
10. Bergenti F., Iotti E., Monica S., Poggi A. Agent-oriented model-driven development for JADE with the JADEL programming language // Comput. Lang. Syst. Struct. 2017. Vol. 50. P. 142–158. doi: 10.1007/978-3-319-93581-2_9
11. Publications by BOINC projects. 2018. URL: https://boinc.berkeley.edu/wiki/Publications_by_BOINC_projects (дата обращения: 01.09.2023).
12. Anderson D. P. BOINC: A Platform for Volunteer Computing. University of California, Berkeley. Space Sciences Laboratory Berkeley, 2018. 37 p.
13. Welcome to the Worldwide LHC Computing Grid. URL: https://wlcg.web.cern.ch (дата обращения: 01.09.2023).

14. Зинкин С. А., Джафар М. С. Развитие информационно-коммуникационных инфраструктур распределенных вычислительных систем на основе концепции «Сеть – это компьютер» // Известия Юго-Западного государственного университета. 2018. Т. 22, № 4. С. 75–93.
15. 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
16. 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. Т. 26, № 1. С. 148–171. doi: 10.21869/2223-1560-2022-26-1-148-171
17. 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
18. 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
19. Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Реализация управляемой приложением функциональной архитектуры пиринговой распределенной вычислительной системы, определяемой концептуальными и логическими моделями искусственного интеллекта. I. Декларативный и логический подходы // Известия высших учебных заведений. Поволжский регион. Технические науки. 2024. № 1. С. 19–38. doi: 10.21685/2072-3059-2024-1-2
20. Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Представление и структурирование знаний в семантико-ориентированной вычислительной среде. Часть I. Интеграция концептуальных графов и логических сетей на основе формализации структурированных ситуаций // Известия высших учебных заведений. Поволжский регион. Технические науки. 2023. № 2. С. 24–51. doi: 10.21685/2072-3059-2023-2-3
21. Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Представление и структурирование знаний в семантико-ориентированной вычислительной среде. Часть II. Интерпретации концептуальных событийных сетевых моделей для заданных предметных областей // Известия высших учебных заведений. Поволжский регион. Технические науки. 2023. № 3. С. 41–71. doi: 10.21685/2072-3059-2023-3-4

References

1. Manoj Kumar Mishra, Yashwant Singh Patel, Moumita Ghosh, Mund G.B. A Review and Classification of Grid Computing Systems. *International Journal of Computational Intelligence Research*. 2017;13(3):369–402.
2. Radchenko G.I. *Raspredelennye vychislitel'nye sistemy = Distributed computing systems*. Chelyabinsk: Fotokhu-dozhnik, 2012:184. (In Russ.)
3. Cabani A., Ramaswamy S., Itmi M., Haziqah Shukri S., Pecuchet J.-P. Distributed Computing Systems: P2P versus Grid Computing Alternatives. *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*. 2007:47–52. doi: 10.1007/978-1-4020-6268-1_9
4. Giridhar Kumar D., SubbaRao Ch.D.V. Peer-To-Peer Computing: Architectures, Applications and Challenges. *International Journal of Recent Technology and Engineering (IJRTE)*. 2019;8(1S4):630–636.

5. Kahanwal B., Singh T.P. The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle. *International Journal of Latest Research in Science and Technology*. 2012;1(2):183–187.
6. Quang Hieu Vu, Mihai Lupu, Beng Chin Ooi. *Peer-to-Peer Computing Principles and Applications*. Springer Publication, 2010. doi: 10.1007/978-3-642-03514-2
7. SD-WAN Software *Defined programmno-opredelyaemaya WAN-set'*. 2022. Available at: https://www.tadviser.ru/index.php/Stat'ya:SD-WAN_(Software_Defined)_Programmno-opredelyaemaya_WAN-set' (accessed 01.09.2023).
8. Zhenjie Yang, Yong Cui, Baochun Li, Yadong Liu, Yi Xu. Software-Defined Wide Area Network (SD-WAN): Architecture, Advances and Opportunities. *28th International Conference on Computer Communication and Networks (ICCCN)*. Valencia, Spain, 2019:1–9. doi: 10.1109/ICCCN.2019.8847124
9. Youssfi M., Bouattane O., Bensalah M. A Parallel Computational Model Based on Mobile Agents for High Performance Computing. *Contemporary Engineering Sciences*. 2015;8(15):677–698. doi: 10.12988/ces.2015.55153
10. Bergenti F., Iotti E., Monica S., Poggi A. Agent-oriented model-driven development for JADE with the JADEL programming language. *Comput. Lang. Syst. Struct*. 2017;50:142–158. doi: 10.1007/978-3-319-93581-2_9
11. *Publications by BOINC projects*. 2018. Available at: https://boinc.berkeley.edu/wiki/Publications_by_BOINC_projects (accessed 01.09.2023).
12. Anderson D.P. *BOINC: A Platform for Volunteer Computing*. University of California, Berkeley. Space Sciences Laboratory Berkeley, 2018:37.
13. *Welcome to the Worldwide LHC Computing Grid*. Available at: https://wlcg.web.cern.ch (accessed 01.09.2023).
14. Zinkin S.A., Dzhafar M.S. Development of information and communication infrastructures of distributed computing systems based on the concept of “The network is a computer”. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the South-West State University*. 2018;22(4):75–93. (In Russ.)
15. 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
16. 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 South-West State University*. 2022;26(1):148–171. (In Russ.). doi: 10.21869/2223-1560-2022-26-1-148-171
17. Volchikhin V.I., Karamysheva N.S., Zinkin S.A., Gurin E.I. Algorithms, logic and modeling of agent-based metacomputer systems with an increased 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
18. 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
19. Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. The implementation of an application-driven functional architecture of a peer-to-peer distributed computing system defined by conceptual and logical models of artificial intelligence. I. The declarative and logical approaches. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2024;(1):19–38. (In Russ.). doi: 10.21685/2072-3059-2024-1-2

20. Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. Presentation and structuring of knowledge in a semantically oriented computational environment. Part 1. Integration of conceptual graphs and logical sets on the basis of formalization of structured situations. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2023;(2):24–51. (In Russ.). doi: 10.21685/2072-3059-2023-2-3
21. Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. Presentation and structuring of knowledge in a semantically oriented computational environment. Part 2. Interpretation of conceptual events of set models for given objective areas. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2023;(3):41–71. (In Russ.). doi: 10.21685/2072-3059-2023-3-4

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: cnit@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: karamyshevans@yandex.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)

Максим Александрович Митрохин
доктор технических наук, доцент, заведующий кафедрой вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

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

Сергей Александрович Зинкин
доктор технических наук, профессор, 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, 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 18.09.2023

Поступила после рецензирования и доработки / Revised 21.12.2023

Принята к публикации / Accepted 15.02.2024