ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

COMPUTER SCIENCE, COMPUTER ENGINEERING AND CONTROL

УДК 004.9
doi:10.21685/2072-3059-2021-4-1

Разработка сетевых агентно-базированных приложений на основе метакомпьютерной технологии

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

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

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

Аннотация. *Актуальность и цели.* Несмотря на то, что к настоящему времени были разработаны новые масштабируемые архитектуры для распределенных вычислительных систем, содержащих сотни и тысячи компьютеров, задача создания прозрачного программного обеспечения уровня *middleware* остается актуальной. Ретроспективная цепочка современных технологий включает облачные, грид- и метакомпьютерные технологии, а также различные варианты коммунальных вычислений. Особый интерес представляет создание метакомпьютерных технологий масштаба Интернета, апробированных при завершении ряда известных исследовательских проектов. Большой и трудно решаемой проблемой при реализации распределенных вычислений остается организация эффективного управления огромным количеством ресурсов, которые доступны в сетевой среде, развернутой на большой площади. Объектом исследования настоящей работы являются метакомпьютерные системы, реализованные на основе глобальных сетей, а предметом исследования – организация управления глобальными вычислительными процессами в сетях на основе метакомпьютерной технологии. Целью исследования является такое совершенствование данной технологии, которое позволит рассматривать сеть с заданной коммуникационной инфраструктурой как единый ресурс с возможной организацией произвольных распределенных алгоритмов. *Материалы и методы.* Исследования выполнены на основе построения и программной реализации концептуальной модели распределенных вычислений, реализуемой в виртуальной метакомпьютерной среде, являющейся результатом интеграции сетевых, грид и облачных систем с агентно-базированными системами. *Результаты.* Предложена организация метакомпьютерных агентно-базированных сетевых распределенных вычислений, реализующих основные конструкции распределенного программирования, где сеть рассматривается реально как компьютер с распределенным программным управлением (*message-driven computing*), а не как средство реализации простейших клиент-серверных или *master-slave* приложений. Модули, или агенты, распределенного приложения способны работать как в реактивном режиме, ожидая прием данных и передачу управления, так и в проактивном режиме, запрашивая данные и управление от предшествующих модулей (агентов). *Вы-*

воды. Проведенные на сети эксперименты позволили подтвердить работоспособность метакомпьютерного приложения и его способность к масштабированию и расширению функциональных возможностей вплоть до свойств облачно-сетевых технологий *AaaS* (*Agent as a Service*) и *FaaS* (*Function as a Service*).

Ключевые слова: метакомпьютеры, программное обеспечение промежуточного уровня, распределенное управление, реализация концептуальной модели, агентно-базированная система

Для цитирования: 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

Development of metacomputer network agent-based applications

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

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

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

Abstract. *Background.* Despite the fact that by now new scalable architectures have been developed for distributed computing systems containing hundreds and thousands of computers, the task of creating transparent software of the “middleware” level remains urgent. The technology retrospective chain includes cloud, grid and metacomputer technologies, as well as various utility computing options. The creation of Internet-scale metacomputer technologies, tested at the completion of a number of well-known research projects, is of particular interest. The organization of effective management of the huge amount of resources that are available in a network environment deployed over a large area is a large and difficult problem in the implementation of distributed computing. The object of the study is metacomputer systems implemented on the basis of global networks, and the subject of the study is the organization of control of global computational processes in networks based on metacomputer technology. The purpose of the study is an improvement of this technology, which will allow considering a network with a given communication infrastructure as a single resource with the possible organization of arbitrary distributed algorithms. *Materials and methods.* The studies are carried out on the basis of the construction and software implementation of a conceptual model of distributed computing, implemented in a virtual metacomputer environment, which is the result of the integration of network, grid and cloud systems with agent-based systems. *Results.* The organization of metacomputer-based agent-based network distributed computing, which implements the basic constructions of distributed programming, where the network is really considered as a computer with distributed program control (message-driven computing), but not as a means of implementing the simplest client-server or master-slave applications. Modules, or agents, of a distributed application are capable of operating both in a reactive mode, waiting for data to be received and control transfer, and in a proactive mode, requesting data and control from previous modules (agents). *Conclusions.* Experiments carried out on a real network made it possible to confirm the operability of a metacomputer application and its ability to scale and expand its functional capabilities up to the properties of cloud-based network technologies AaaS (Agent as a Service) and FaaS (Function as a Service).

Keywords: metacomputers, middle-level software, distributed control, conceptual model implementation, agent-based system

For citation: Volchikhin V.I., Karamysheva N.S., Gorynina A.V., Zinkin S.A. Development of metacomputer network agent-based applications. *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

Введение

С развитием сети Интернет появилась возможность объединения многих компьютеров в интегрированные распределенные системы, предоставляющие большое число приложений цифровых технологий для науки, торговли, образования и развлечений [1, 2]. Несмотря на быстрое развитие аппаратной и коммуникационной инфраструктуры Интернета, поиски и разработки новых программных инфраструктур продолжаются. Создаются разнообразные приложения на основе коммуникационных сервисов. На транспортном уровне, например, работа приложений организуется при помощи протокола управления передачей данных TCP (Transmission Control Protocol), однако разработчики приложений должны сами реализовывать службы именования объектов, репликации, миграции и безопасности, для чего во Всемирной Паутине (World Wide Web) поверх протокола TCP реализуется протокол передачи гипертекста, использующий унифицированные указатели ресурсов.

К настоящему времени был предложен ряд новых масштабируемых архитектур для глобальных распределенных вычислительных систем, из которых наиболее известны проект Амстердамского свободного университета (Нидерланды) Globe – Global Object Based Environment [3]; проект Opus [4] университета Дюка (США) – продолжение проекта Калифорнийского университета WebOS (Беркли, США); проект Project Oxygen [4], участниками которого являются агентство DARPA, а также компании Delta Electronics, Hewlett-Packard, Nokia, NTT и Philips Electronics. В процессе работы над этими проектами была создана «прозрачная» сетевая операционная среда, явившаяся прототипом для последующих разработок. Особый интерес представляет первая разработка Globe, основанная на использовании концепции одноранговой пиринговой сети P2P (Peer-to-Peer) [5], в которой узлы сети работают без централизованного сервера-хранилища, а коммуникационная сеть связывает все узлы. В этом проекте, кроме того, было впервые разработано программное обеспечение промежуточного слоя (*middleware*), позволившее создавать сетевые приложения без непосредственного использования транспортного уровня.

Ретроспективная цепочка современных технологий включает облачные, грид и метакомпьютерные технологии, а также различные варианты коммунальных вычислений. Главная особенность этих технологий – сетевая реализация на основе парадигмы передачи сообщений. Предшественниками метакомпьютерных технологий являются широко используемые в настоящее время технологии построения вычислительных систем типа PVM (Parallel Virtual Machine) и MPI (Message Passing Interface), ориентированных в первую очередь на параллельную обработку информации при использовании концепции передачи сообщений [6].

Из метакомпьютерных проектов наиболее известны Distributed.net, GIMPS – Great Internet Mersenne Prime Search, SETI@home, TERRA ONE, Globus, The Metacomputing Project, PACX-MPI, Condor, Legion, метакомпьютеры общецелевого назначения Charlotte, Javelin, WebFlow, Gateway, CX и ранние версии метакомпьютера Globus [7]. Современная версия проекта Globus стала стандартом де-факто для грид-вычислений [8].

Вычислительные сети масштаба Интернета появляются в результате успешного завершения различных исследовательских проектов. В первую очередь это проекты US National Technology Grid (NTG) [9], European Data Grid (EDG) [10] и глобальный гуманитарный проект World Community Grid (WCG) [11].

Результатом развития проекта EDG явилась Европейская грид-инфраструктура (EGI – European Grid Infrastructure), обеспечивающая доступ к ресурсам с использованием методов распределенных вычислений и связывающая вычислительные центры в разных странах Европы для поддержки международных исследований во многих научных дисциплинах. Консорциум участников проекта включает более 70 институтов из 27 европейских стран.

Сеть WCG, например, содержит более двух миллионов подключенных к ней пользовательских, в основном волонтерских, компьютеров и предназначена для реализации распределенных вычислений сотнями тысяч пользователей. Однако большой и трудно решаемой проблемой при реализации распределенных вычислений остается организация эффективного управления огромным количеством ресурсов, которые доступны в сетевой среде, развернутой на большой площади.

Объектом исследования настоящей работы являются метакомпьютерные системы, реализованные на основе TCP/IP сетей (рис. 1), а предметом исследования – организация управления глобальными вычислительными процессами в сетях на основе метакомпьютерной технологии.

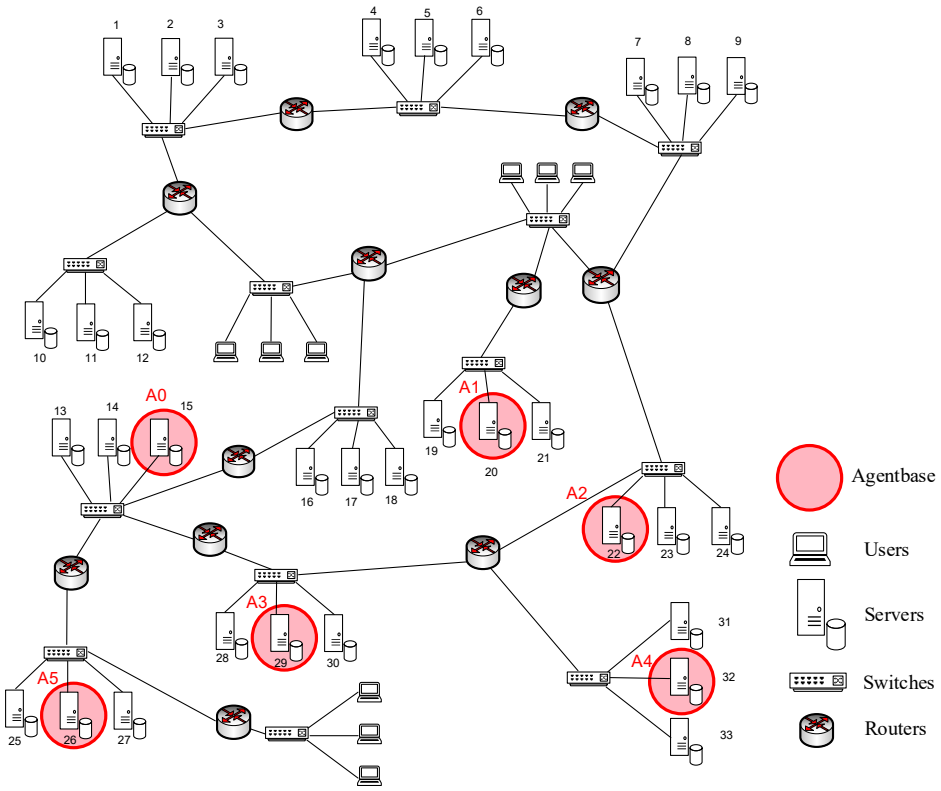

Рис. 1. Развертывание агентов в физической глобальной TCP/IP сети

1. Выбор платформы для реализации метакомпьютерных систем

Анализ литературы, приведенный во введении, показывает, что основной проблемой при создании метакомпьютерных систем является обеспечение масштабируемости глобальной распределенной вычислительной инфраструктуры, в которой они реализуются. Эта проблема может быть решена путем использования одноранговых сетевых архитектур типа P2P и интеграции этих архитектур с системами типа грид. При этом следует учитывать, что архитектура P2P, в отличие от грид, нашла широкое применение при работе с мультимедийной информацией и требуется некоторая доработка для ее интеграции с грид-системами. Одноранговые пиринговые узлы действуют как клиенты и как серверы, что позволяет минимизировать рабочую нагрузку на каждый узел за счет повышения уровня загруженности общих вычислительных ресурсов сетевой среды. Обзоры фундаментальных концепций P2P-компьютинга с анализом сопутствующих сетевых топологий даны в работах [5, 12].

Мультиагентные системы могут быть интегрированы с грид-средами и пиринговыми сетями и осуществлять управление вычислительными ресурсами и ресурсами хранения данных. В общем случае агентный подход может облегчить управление крупномасштабными сетями. Агентно-ориентированные метакомпьютерные приложения способны обеспечить масштабирование как по количеству агентов, так и по количеству ресурсов.

Ряд агентных систем позволяет создавать крупномасштабные сетевые приложения. К таким системам относятся AgentScape, CoABC Grid, JADE с расширением JADEL, Jason, JaCaMo [13, 14] и др. В настоящей работе понятие масштабируемости уровня Интернета относится к глобальной распределенной вычислительной инфраструктуре, сервисам и приложениям.

В системе AgentScape [13] используется агентный подход к управлению ресурсами в грид-средах и реализована агентная инфраструктура, которая может быть интегрирована со слоем промежуточного программного обеспечения грид. Эта инфраструктура обеспечивает поддержку мобильных агентов и масштабируется как по количеству агентов, так и по количеству ресурсов. В системе CoABS Grid [13] реализованы несколько уровней приложений, функциональность которых может адаптироваться к изменяющимся требованиям и масштабироваться.

Агентная платформа JADE (Java Agent Development Framework) [15–17] по мере развития превратилась в крупный проект программного обеспечения с мировым охватом и базой пользователей. Это одна из наиболее апробированных и перспективных технологических платформ для программных агентов. JADE – это платформа с открытым исходным кодом для разработки приложений на основе одноранговых агентов. Помимо абстракции агента, она также предоставляет модель выполнения и композиции задач, одноранговую связь агентов на основе асинхронной передачи сообщений и службу желтых страниц, которая поддерживает механизм обнаружения публикации и подписки. Системы на основе JADE могут быть распределены по машинам с различными операционными системами и использоваться многими языками (например, Jason и JaCaMo) в качестве инфраструктуры. В работах [16–18] авторы представляют язык JADEL (JADE Language), расширение JADE, которое обеспечивает поддержку создания агентов и мультиагентных систем

поверх JADE без необходимости непосредственного использования средств языка Java. Программная среда JADE может подключаться к любому проекту на языке Java.

Реализации агентно-базированных приложений на платформе JADE с применением грид и пиринговых P2P технологий пригодны для использования в качестве основы метакомпьютерной технологии.

2. Агентно-базированная инфраструктура для реализации вычислительных процессов в метакомпьютерах

Уточняя определение метакомпьютера в современном контексте, назовем метакомпьютером множество, числом вплоть до нескольких миллионов взаимосвязанных через сеть коммуникаций компьютеров, которые реализуют распределенные алгоритмы. Метакомпьютерный подход получил свое воплощение в современных облачно-сетевых распределенных вычислительных системах, ставя проблему создания полнофункциональной среды и инфраструктуры для сетевых вычислений.

В процессе разработки прототипа метакомпьютерной системы в данной статье не рассматривается реализация высокопроизводительных параллельных вычислений и необходимость использования высокоскоростной системы коммуникаций. Поставлена задача реализации сетевых распределенных вычислений в произвольной TCP/IP сети с готовой инфраструктурой для хранения и передачи данных и программ (см. рис. 1).

Принято, что распределенная программа реализована в виде агентно-базированного приложения на платформе JADE с применением некоторых концепций пиринговых P2P технологий. При реализации агентно-ориентированной метакомпьютерной технологии здесь используется только платформа JADE, образующая промежуточный уровень (*middleware*) сетевого программного обеспечения. Ни грид-платформа, ни какое-либо пиринговое промежуточное программное обеспечение в настоящей работе не используется. На узлах глобальной TCP/IP сети размещены контейнеры с агентами JADE, обозначенные на рис. 1 кружками. Алгоритм распределенных вычислений представляется концептуальной схемой и реализуется в виде временно организуемой мультиагентной системы. Из концепций, на основе которых реализуются пиринговые P2P вычисления, выбраны концепции связности графа коммуникаций и объединенных клиент-серверных, а точнее, приемо-передающих функций модулей-агентов.

Основные компоненты агентно-ориентированной среды JADE представлены на рис. 2. Агенты работают под управлением среды, предоставляемой платформой JADE, в ней предусмотрены механизмы создания, удаления, миграции, поиска и взаимодействия агентов.

Система управления агентами AMS (Agent Management System) предоставляет средства управления жизненным циклом агента, а также содержит в себе пространство имен агентов. Данная система работает в рамках платформы как агент с именем AMS. Служба каталогов DF (Directory Facilitator) представляет собой службу «желтой страницы», где агенты могут публиковать информацию о предоставляемых ими сервисах. С помощью службы каталогов агент может найти агента, предоставляющего конкретный функцио-

нал и вступить с ним в переговоры. Главный контейнер платформы всегда содержит агенты AMS и DF, запускаемые автоматически одновременно с контейнером.

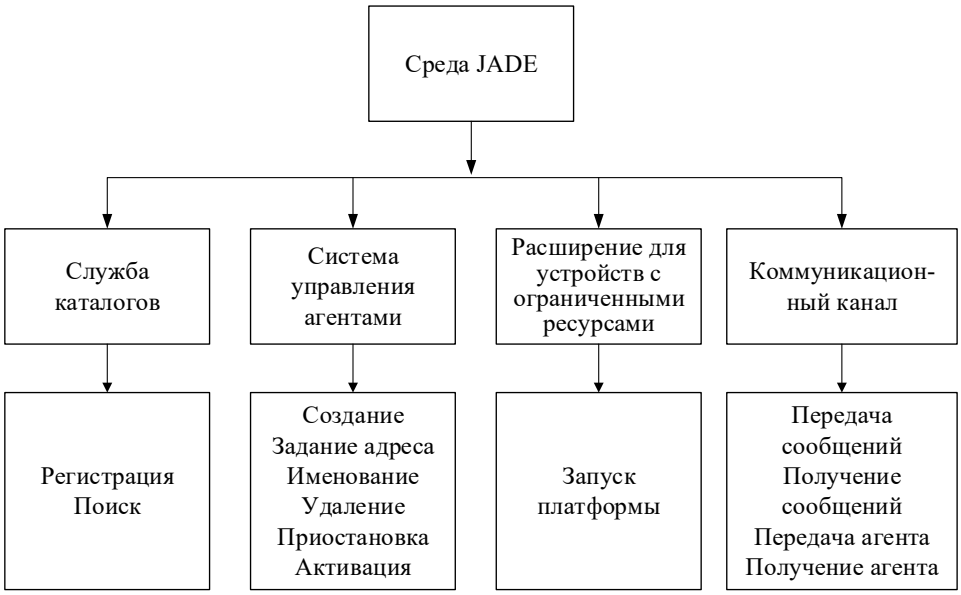

Рис. 2. Основные компоненты агентно-ориентированной среды JADE

Особенностью JADE является возможность развертывания агентской платформы на устройствах с ограниченными ресурсами за счет использования расширения LEAP (Lightweight Extensible Agent Platform).

Агентная платформа в общем случае состоит из агента AMS, одного или нескольких агентов DF и других агентов общего назначения (рис. 3). Внутри одной платформы может существовать несколько DF-агентов, предоставляющих информацию о различных группах сервисов или о сервисах групп агентов. Платформа может состоять как только из главного контейнера, так и из главного контейнера и нескольких неглавных, располагающихся на других хостах сети. Агенты могут реализовывать несколько сервисов, описание которых хранит DF-агент. Система управления агентами, или AMS-агент, хранит описание и контролирует именование агентов.

При взаимодействии агенты используют сервис MTS (Message Transport Service). Внутри платформы сообщения пересылаются в виде Java-объектов, а при обмене между платформами или с другими мультиагентными системами – в виде XML-строки. При отправке сообщения оно попадает в очередь сообщений агента-адресата, о чем агент уведомляется. Для выбора нужного типа сообщения из очереди предусмотрены фильтры.

Агентная платформа JADE полностью соответствует спецификациям организации FIPA (Foundation of Physical Intelligent Agents) [19] для интеллектуальных агентов (рис. 4). В частности, один из стандартов FIPA регламентирует, что в сетевом приложении агенты идентифицируются уникальными именами и представляют набор служб (сервисов), в результате чего облегчается реализация новейших облачных архитектур «функция как сервис»

(FaaS – Function-as-a-Service) и «агент как сервис» (AaaS – Agent-as-a-Service), предложенных в работе [20].

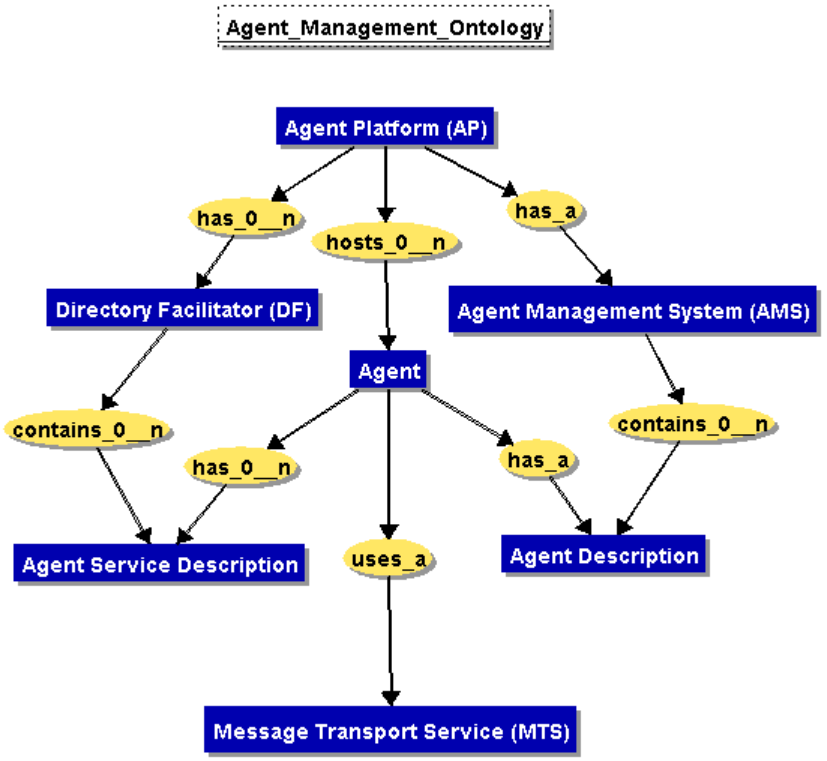

Рис. 3. Концептуальный граф (онтология) агентной платформы JADE

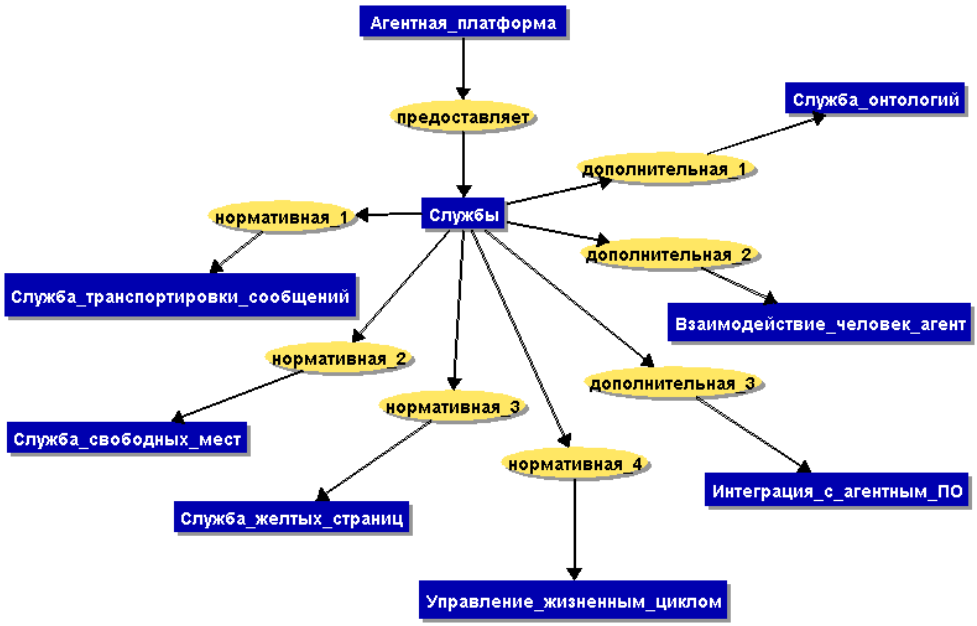

Рис. 4. Службы, предоставляемые агентской платформой
в соответствии со стандартом FIPA

3. Представление распределенного алгоритма метакомпьютерного приложения концептуальным графом

На рис. 5 представлен концептуальный граф распределенного алгоритма для некоторого метакомпьютерного приложения. Операторы распределенного алгоритма реализуются агентами A0, A1, …, A8, размещенными на узлах (хостах) Y1, Y2, …, Y6 компьютерной сети. Агенты записывают в ACL-сообщение информацию о сети и оправляют следующему агенту. Для взаимодействия с пользователем в удобной ему форме вводится агент ServiceAgent, который контролирует целостность сети, систематизирует и хранит статистику по сети. Агенты A1 и A6 осуществляют случайный выбор следующего агента. Агентом A3 каждое нечетное сообщение оправляется агенту A4, каждое четное – агенту A5.

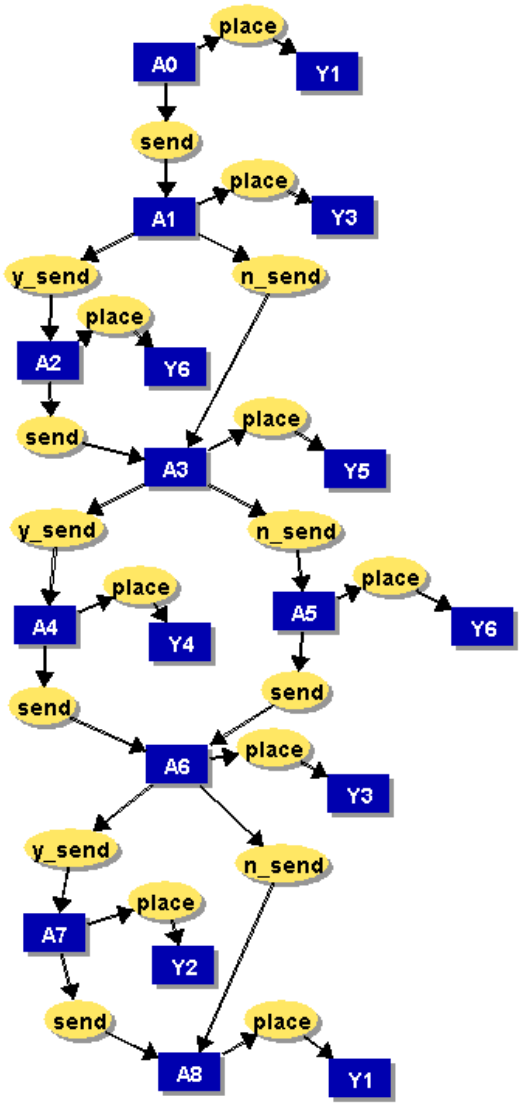

Рис. 5. Концептуальный граф распределенного алгоритма метакомпьютерного приложения

На рис. 5 агенты связаны отношениями *send*, *y_send*, *n_send*. Отношение *send* соответствует передаче управления и, возможно, данных от одного агента другому. Для отношений *y_send* и *n_send* требуется особое пояснение. По умолчанию предполагается, что после выполнения агентом Ai каких-либо вычислений может быть определено значение условия Zi. Этот факт обозначается как Ai | Zi.

При Ai | Zi = true конкретизируется отношение *y_send* и управление передается агенту, связанному с этим отношением. При Ai | Zi = false конкретизируется отношение *n_send* и управление передается по альтернативному пути. Аналитически этот факт записывается следующим образом:

$$[(Ai \mid Zi) / Yn] (Aj / Ym \oplus Ak / Yp).$$

Через косую черту "/" записаны имена узлов сети, на которых агенты реализуют свои функции. Во избежание загромождения концептуального графа лишними обозначениями условия Z на графе принято опускать.

4. Аналитическая форма записи концептуальных схем распределенных алгоритмов

В операторных схемах, построенных для децентрализованной реализации вычислительного процесса на основании парадигмы передачи сообщений и в случае использования стационарных агентов, определены следующие обозначения для выполняемых операций:

Ai/Yk – предварительное размещение агента Ai на узле Yk;

(Ai|Zi)/Yk – вычисление значения условия Zi в процессе выполнения действий агента Ai на узле Yk;

[(Ai|Zi)/Yk] – проверка значения условия Zi, полученного в процессе выполнения действий агента Ai на узле Yk;

[(Ai|Zi)/Yk](Aj/Ym⊕Ah/Yn) – выполнение действий агента Aj на узле Ym при Zi = true или выполнение действий агента Ah на узле Yn при Zi = false.

{(Ai|Zi)/Yk} – циклическая реализация действий агента Ai на узле Yk при Zi = false и выход из цикла при Zi = true, где значение данного условия вычисляется заново при каждом повторении выполнения действий агента Ai;

||Ai/Yk|| – обозначение для внутреннего параллелизма агента Ai, реализующего свои действия на узле Yk;

||Ai/Yk; Aj/Ym; …; Ak/Yn|| – обозначение для «сетевого» параллелизма при работе агентов Ai, Aj, …, Ak, размещенных на узлах компьютерной сети Yk, Ym, …, Yn соответственно, где символ «;» обозначает операцию «возможно, одновременного» выполнения действий агентов. Модальность «возможно» предупреждает пользователя о том, что в компьютерной сети может сложиться ситуация, когда не все агенты Ai, Aj, …, Ak смогут выполняться одновременно, с возможным пересечением времен реализации действий.

Символ «,» соответствует операции Ai/Yk, Aj/Ym последовательного выполнения действий агентов Ai и Aj, размещенных на узлах Yk и Ym соответственно. В случае, если $k \neq m$, управление вычислительным процессом осуществляется путем передачи сообщения (возможно, с результатами промежуточных вычислений) от агента Ai агенту Aj.

Пример соответствующего рис. 5 операторного выражения для описания действий в мультиагентной сетевой вычислительной среде при децентрализованной реализации вычислительного процесса на основании парадигмы передачи сообщений в сети P2P (рис. 6) со стационарными агентами:

$$\mathbf{A_D} = A0/Y1, [(A1|Z1)/Y3](A2/Y6 \oplus (A3|Z3)/Y5), [(A3|Z3)/Y5](A4/Y4 \oplus A5/Y6),$$
$$[(A6|Z6)/Y3](A7/Y2, A8/Y1 \oplus A8/Y1).$$

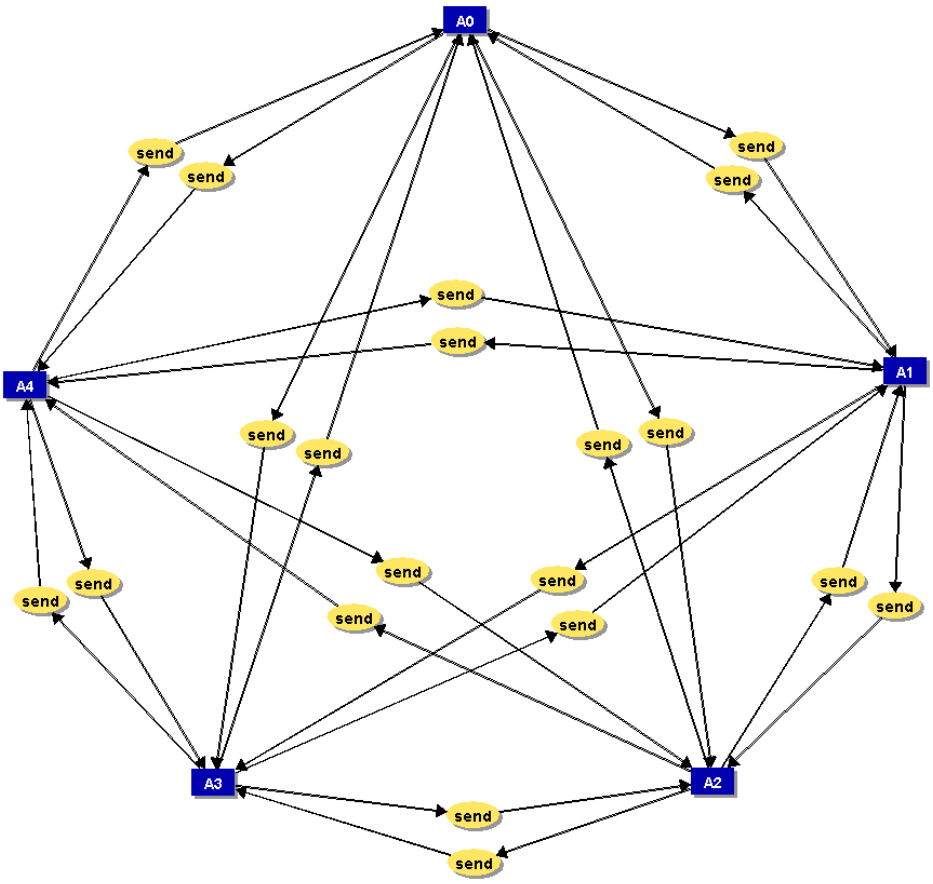

Рис. 6. Логическая архитектура агентно-ориентированной сети типа P2P

Выражение $\mathbf{A_D}$ построено на основе известных в программировании принципах структурного программирования. Однако во многих случаях распределенные программы не поддаются простому структурированию. В подобных случаях следует использовать раздельное описание выполняемых передач сообщений между агентами в виде правил. В качестве примера рассмотрена система правил-продукций $\mathbf{S_D}$, в целом реализующих то же децентрализованное управление вычислениями в соответствии с предыдущим выражением $\mathbf{A_D}$:

$$\mathbf{S_D} = (A0/Y1 \rightarrow A1/Y3;$$
$$(A1|Z1)/Y6 \rightarrow A2/Y6;$$
$$(A1|\neg Z1)/Y6 \rightarrow A3/Y5;$$

A2/Y6 → A3/Y5;
(A3|Z3)/Y5 → A4/Y4;
(A3|¬Z3)/Y5 → A5/Y6;
A4/Y4 → A6/Y3;
A5/Y6 → A6/Y3;
(A6|Z6)/Y3 → A7/Y2;
(A6|¬Z6)/Y3 → A8/Y1;
A7/Y2 → A8/Y1).

Достоинством подобного описания $\mathbf{S_D}$, в отличие от выражения $\mathbf{A_D}$, является представление передачи сообщений в явной форме, что облегчает программирование распределенного приложения.

В рассмотренных выше примерах учитывалась стационарность агентов. В случае использования мобильных агентов станет возможным осуществить начальное развертывание агентов с одного пользовательского компьютера, что имеет большое значение для реализации распределенных приложений в глобальной компьютерной сети.

Следующий пример иллюстрирует реализацию операторной схемы для случая использования мобильных агентов в сети с архитектурой P2P при децентрализованном управлении распределенным вычислительным процессом:

$$\mathbf{B_D} = A0 \rightarrow D0/Y1, [(D1|Z1)/Y3](D2/Y6 \oplus (D3|Z3)/Y5),$$

$$[(D3|Z3)/Y5](D4/Y4 \oplus D5/Y6), [(D6|Z6)/Y3](D7/Y2, D8/Y1 \oplus D8/Y1);$$

где → – операция ввода мобильного агента A0 в сеть, на которой размещены данные D0, …, D8. Здесь при выполнении операторной схемы агент A0 реализует поток управления вычислительным процессом, проходя по узлам с данными. Например, данными могут являться содержимое некоторых баз данных, в которых агент A0 реализует операции поиска.

В общем случае при выполнении операторной схемы **B** могут участвовать несколько мобильных агентов, запускаемых последовательно и реализующих «конвейерный» параллелизм:

$$A0 \rightarrow A1 \rightarrow, \ldots, \rightarrow An \rightarrow \mathbf{B}.$$

В этом случае необходимо учитывать тот факт, что в процессе выполнения операций над данными порядок следования агентов может измениться.

5. Централизованное управление распределенным вычислительным процессом в сети с клиент-серверной архитектурой

Клиент-серверная архитектура приложений относится к числу наиболее распространенных архитектур, реализуемых в компьютерных сетях. Это сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. При выполнении функций агента-клиента и агента-сервера агенты JADE должны реализовывать логику приложений клиентской и серверной частей, а также коммуникационные функции. На основе клиент-серверной архитектуры реализуются, например, все Web-сервисы с доступом к удаленным базам данных. Ввиду того, что при реализации распределенных вычислений на основе мобильных или стационарных агентах возможно ис-

пользовать готовые программные компоненты клиент-серверных приложений, целесообразно рассмотреть организацию распределенных агентно-ориентированных вычислений с централизованным управлением (рис. 7).

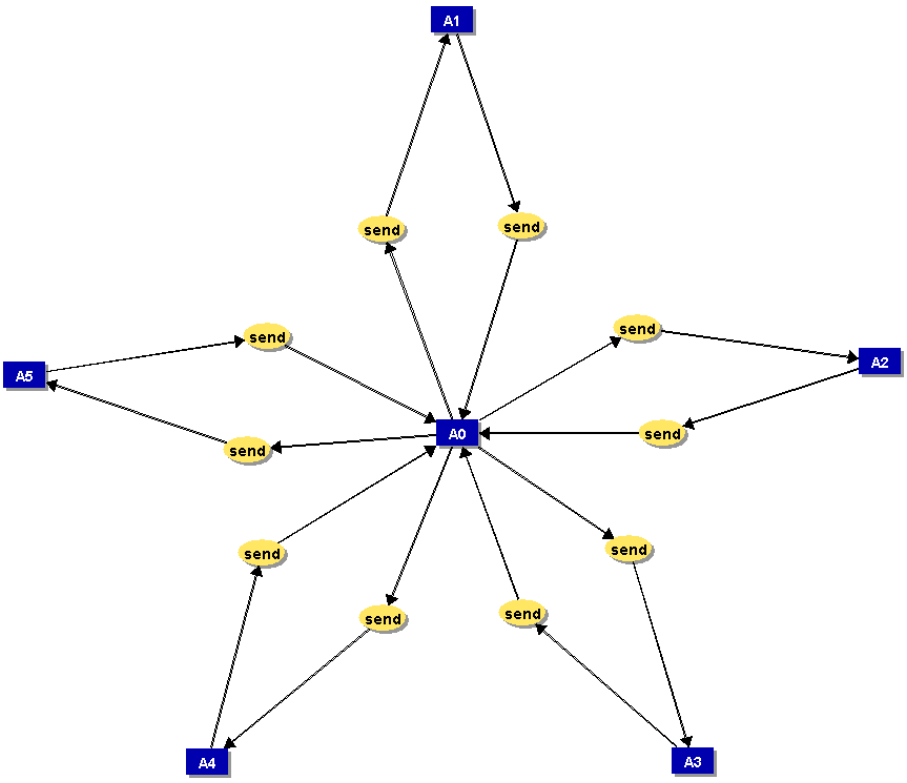

Рис. 7. Логическая архитектура агентно-ориентированной сети типа «один клиент – много серверов»

Пусть агент-клиент A0, осуществляющий управление вычислительным процессом, размещен в центре сети – на узле Y1, а остальные агенты-серверы – на серверных узлах, указанных на концептуальной схеме распределенного алгоритма (рис. 5). В системе принят режим работы «запрос-ответ», при котором все передачи управления вычислительным процессом и промежуточных результатов осуществляются при посредничестве агента-клиента A0. Следует отметить, что подобная архитектура могла бы получить название «мастер-слуги» (англ. вариант названия master-slaves), где «мастером» является агент A0, связанный с пользователем, а «слугами» – все остальные агенты. Распределенные вычисления под управлением одного агента-клиента могут быть описаны путем следующей модификации рассмотренной ранее системы $\mathbf{S_D}$:

$$\mathbf{S_C} = (A0/Y1 \rightarrow A1/Y3;$$
$$(A1|Z1)/Y6 \rightarrow A0/Y1,\ A0/Y1 \rightarrow A2/Y6;$$
$$(A1|\neg Z1)/Y6 \rightarrow A0/Y1,\ A0/Y1 \rightarrow A3/Y5;$$
$$A2/Y6 \rightarrow A0/Y1,\ A0/Y1 \rightarrow A3/Y5;$$
$$(A3|Z3)/Y5 \rightarrow A0/Y1,\ A0/Y1 \rightarrow A4/Y4;$$

(A3|¬Z3)/Y5 → A0/Y1, A0/Y1 → A5/Y6;
A4/Y4 → A0/Y1, A0/Y1 → A6/Y3;
A5/Y6 → A0/Y1, A0/Y1 → A6/Y3;
(A6|Z6)/Y3 → A0/Y1, A0/Y1 → A7/Y2;
(A6|¬Z6)/Y3 → A0/Y1, A0/Y1 → A8/Y1;
A7/Y2 → A0/Y1, A0/Y1 → A8/Y1);
A8/Y1 → A0/Y1).

Стрелками, как и ранее, обозначается передача сообщений (возможно, с данными), управляющих распределенным вычислительным процессом. Символ «,» в данной системе соответствует последовательному выполнения действий.

6. Пример реализации метакомпьютерного приложения на основе фреймворка JADE (по схемам A_D и S_D)

6.1. Два этапа работы метакомпьютерного приложения

В общем случае работу системы следует разбить на два основных этапа – этап регистрации агентов и этап работы линии операторов.

На рис. 8 представлена диаграмма сотрудничества агентов в процессе регистрации. Агенты отправляют сервисному агенту сообщение типа REQUEST со своим локальным именем. После подключения всех агентов сервисный агент уведомляет пользователя о возможности начала работы системы.

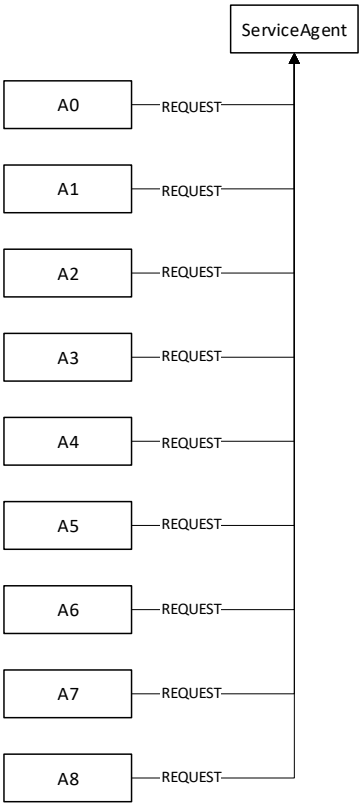

Рис. 8. Диаграмма сотрудничества: процесс регистрации агентов

На рис. 9 представлена диаграмма сотрудничества агентов в процессе старта и остановки линии. Для старта процесса генерации сообщений пользователь должен через графический интерфейс отправить от агента ServiceAgent агенту A0 сообщение с типом действия CONFIRM и контентом start.

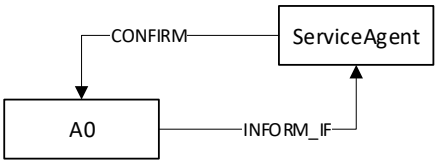

Рис. 9. Диаграмма сотрудничества: процесс старта и остановки линии

Агент A0 в таком случае отвечает сообщением с типом INFORM_IF и контентом «ok». После старта линии A0 начинает формировать сообщения с интервалом в секунду и отправлять агенту A1. Для остановки процесса генерации сообщений пользователь должен через графический интерфейс отправить от агента ServiceAgent агенту A0 сообщение с типом действия CONFIRM и контентом stop. A0 в таком случае отвечает сообщением с типом INFORM_IF и контентом, где записано количество отправленных пакетов.

6.2. Состав и работа приложения

Состав и работа приложения:

– jade.core.Agent – класс стандартной библиотеки классов JADE. Описывает агента, предлагает функционал для работы с очередью агента;

– AService.A0Agent, AService.A1Agent, AService.A2Agent, AService.A3Agent, AService.A4Agent, AService.A5Agent, AService.A6Agent, AService.A7Agent, AService.A8Agent – классы агентов-операторов, наследуются от jade.core.Agent;

– SAgent – класс сервисного агента, является дочерним классом класса jade.core.Agent;

– LineManager, WaitMessage, WaitRegistrationMessage, WaitPCMessage, WaitStatisticsMessage – внутренние классы агентов, поведения, наследуют CyclicBehaviour;

– Registration – класс-поведение, экземпляры класса используют агенты-операторы на этапе регистрации;

– DelayBehaviour – класс-поведение, используется агентами-операторами для симуляции обработки данных на узле;

– jade.core.behaviours.CyclicBehaviour – класс стандартной библиотеки классов JADE, представляет собой повторяющееся поведение;

– jade.core.behaviours.OneShotBehaviour – класс стандартной библиотеки классов JADE, представляет собой поведение, выполняемое только один раз;

– jade.core.behaviours.WakerBehaviour – класс стандартной библиотеки классов JADE, представляет собой поведение, выполняемое с некоторой задержкой;

– jade.core.behaviours. SimpleBehaviour – класс стандартной библиотеки классов JADE, реализует простое поведение.

На рис. 10 представлена диаграмма классов сетевого приложения.

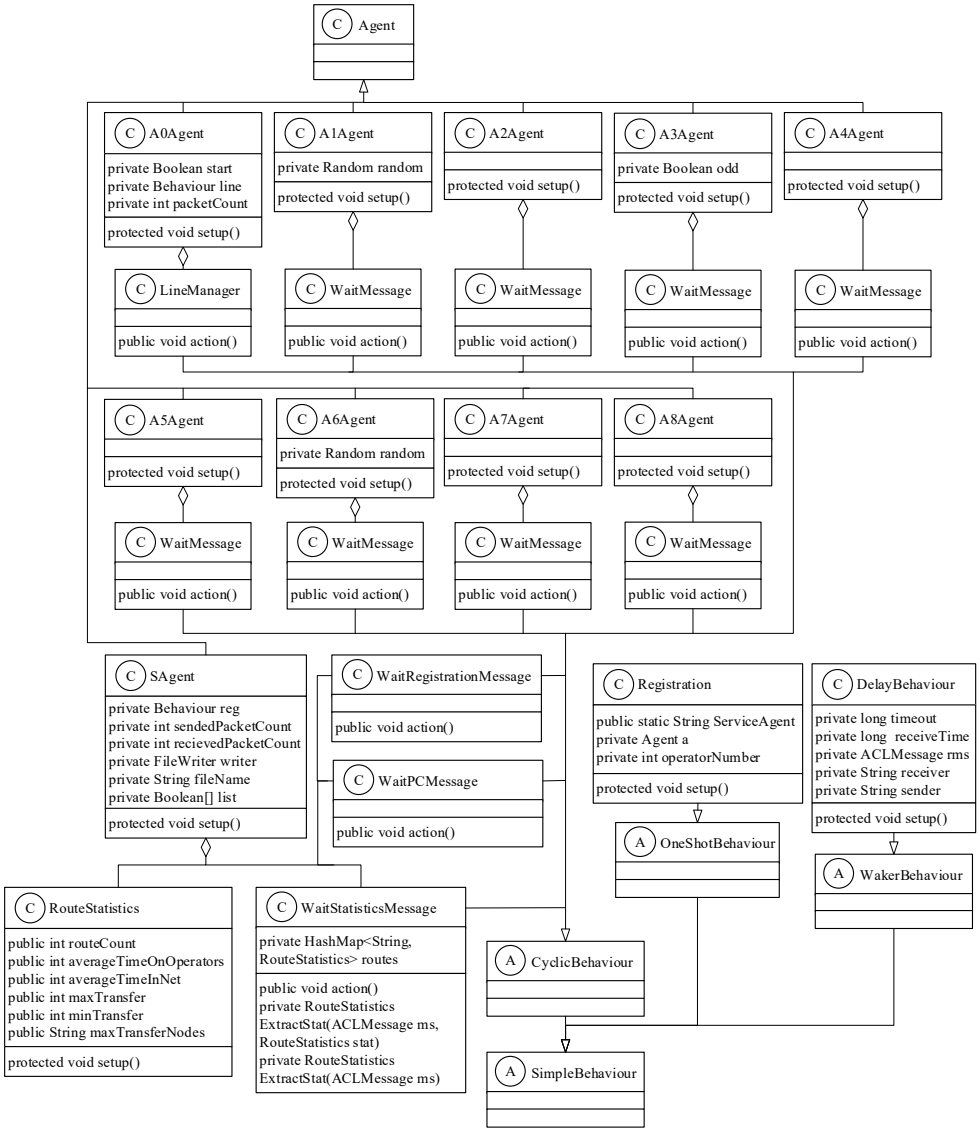

Рис. 10. Диаграмма классов сетевого приложения

Агенты-операторы реализуют поведение Registration, которое является наследником OneShotBehaviour. Затем агенты выполняют циклическое поведение ожидания сообщений. Строго говоря, поведения выполняются параллельно, однако в данном случае событие «Сообщение» не может произойти раньше регистрации агента ввиду ограничений человеческих возможностей.

Агент A0 выполняет циклическое поведение ожидания сообщений типа CONFIRM WaitMessage. Когда линия запускается сообщением от ServiceAgent, A0 получает еще одно поведение, экземпляр класса TickerBehaviour, выполняющееся каждую секунду:

```
line = new TickerBehaviour(myAgent, 1000) {
protected void onTick() {
ACLMessage ms = new ACLMessage();
ms.addReceiver(new AID("A1", AID.ISLOCALNAME));
```

```
        ms.setContent(Integer.toString(packetCount++) + " " + getLocalName() + " "
+ Objects.toString(System.currentTimeMillis(), null));
        ms.setPerformative(ACLMessage.INFORM);
        myAgent.send(ms);
        }
        };
        addBehaviour(line);
```

После остановки линии поведение line удаляется.

SAgent использует сразу несколько параллельно исполняемых циклических поведений, срабатывающих на разные события: WaitRegistrationMessage для ожидания запросов на регистрацию, WaitPCMessage для ожидания контрольных сообщений от A0, WaitStatisticsMessage для ожидания сообщений со статистикой от A8.

Для симуляции работы на узлах сообщения отправляются на следующий узел с некоторой задержкой. Она реализована с использованием поведения DelayBehaviour, наследующегося от WakerBehaviour. При этом поведение получает ACL-сообщение как аргумент и оперирует им. Класс DelayBehaviour описан ниже:

```
public class DelayBehaviour extends WakerBehaviour{
   private long timeout, receiveTime;
   private ACLMessage rms;
   private String receiver, sender;
  public DelayBehaviour(Agent a, long timeout, ACLMessage receivedms,
String receiver) {
      super(a,timeout);
      sender = a.getLocalName();
      this.timeout = timeout;
      this.rms = receivedms;
      this.receiver = receiver;
      receiveTime = System.currentTimeMillis();
   }
   protected void onWake() {
       String [] args = rms.getContent().split(" ", 3);
       String mes = args[0] + " " + args[1] + sender + " " +  args[2]  + " " +
Long.toString(receiveTime)          +          "          "          +          Ob-
jects.toString(System.currentTimeMillis());
      ACLMessage ms = new ACLMessage();
      ms.addReceiver(new AID(receiver, AID.ISLOCALNAME));
      ms.setContent(mes);
      ms.setPerformative(ACLMessage.INFORM);
      myAgent.send(ms);
   }
}
```

При тестировании системы были задействованы 7 узлов сети 192.168.10.0, созданы главный контейнер и 6 неглавных, запущены 9 агентов-операторов с именами A0, …, A8 и сервисный агент ServiceAgent. Принадлежность агентов контейнерам отображена на рис. 11.

На рис. 12 приведен графический интерфейс главного контейнера после запуска всех агентов-операторов и сервисного агента. Сразу после запуска каждый агент-оператор регистрируется у сервисного агента.

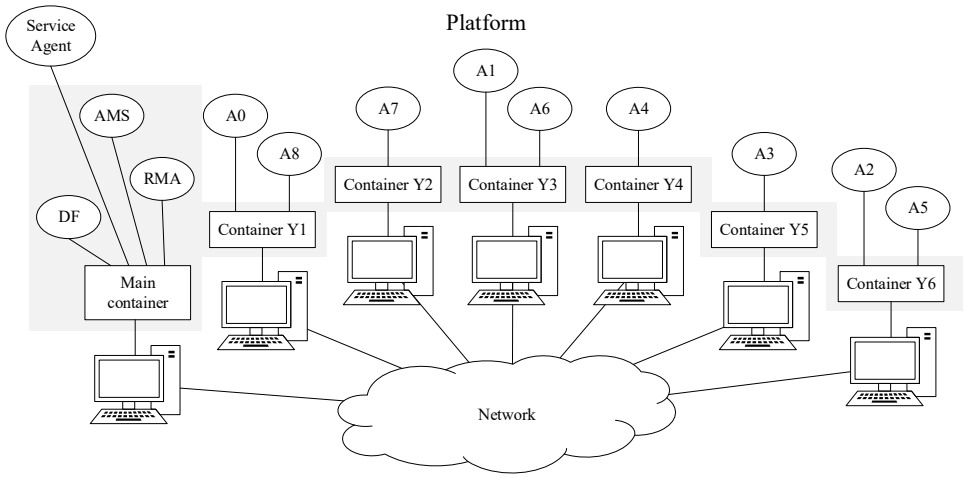

Рис. 11. Расположение агентов в экспериментальной сети

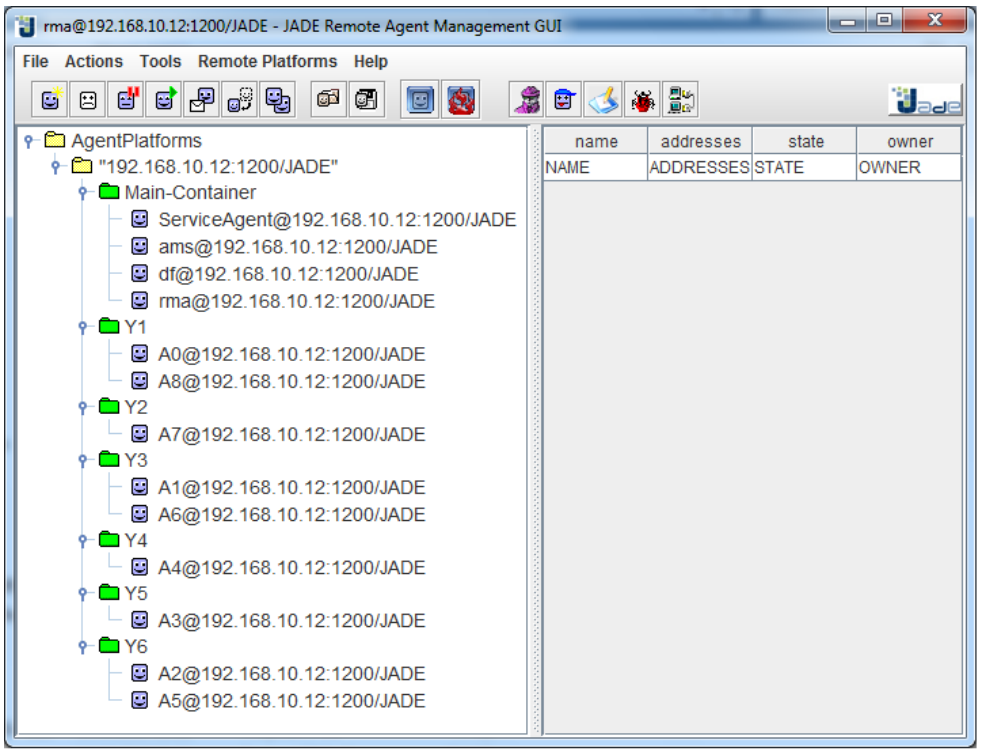

Рис. 12. Вид графического интерфейса главного
контейнера после запуска всех агентов

Сервисный агент уведомляет агенты-операторы о возможности начала работы линии. Определены команды старта линии и остановки: сообщения типа CONFIRM от ServiceAgent агенту A0 с контентом start и stop соответственно.

Целью проведенного эксперимента было определение работоспособности метакомпьютерного приложения, что и получило свое подтверждение.

Метакомпьютерное приложение выполнялась на сети Fast Ethernet в условиях наличия постороннего трафика средней интенсивности. В процессе эксперимента были сгенерированы 46 сообщений, в процессе прохождения которых изменялись значения логических условий и, следовательно, маршруты перемещений сообщений. Общее время задержки на прохождение 46 сообщений от команды start до команды stop равнялось 55395 мс. Задаваемое минимальное время выполнения T_{agent} каждого агента-оператора представлено в табл. 1. Реальное время складывается из времени выполнения собственных операций агента-оператора по обработке сообщения, задержки T_{agent} для симуляции работы и из времени ожидания новых сообщений.

Таблица 1

Агенты-операторы	T_{agent} (мс)
A1	2000
A2	3000
A3	1000
A4	600
A5	600
A6	1400
A7	700
A8	1200

В процессе проведения эксперимента учитывалось также время, затрачиваемое на обмен сообщениями по сети. Собранные статистические данные о среднем времени T_{rout} прохождения каждого маршрута распределенного алгоритма в компьютерной сети представлены в табл. 2.

Таблица 2

Маршруты сообщений	T_{rout} (мс)
A0 A1 A2 A3 A5 A6 A7 A8	9968
A0 A1 A3 A5 A6 A7 A8	6960
A0 A1 A2 A3 A4 A6 A7 A8	9968
A0 A1 A2 A3 A5 A6 A8	9262
A0 A1 A2 A3 A4 A6 A8	9258
A0 A1 A3 A5 A6 A8	6250
A0 A1 A3 A4 A6 A8	6248
A0 A1 A3 A4 A6 A7 A8	6955

Значения времени прохождения маршрутов сообщений при реализации вычислений в среде реальной компьютерной сети показывают работоспособность реализации мультиагентной метакомпьютерной системы.

Заключение

Предложена организация метакомпьютерных агентно-базированных сетевых распределенных вычислений, реализующих основные конструкции распределенного программирования, где сеть рассматривается реально как компьютер с распределенным программным управлением (message-driven computing), а не как средство реализации простейших клиент-серверных или

master-slave приложений. Модули, или агенты, распределенного приложения способны работать как в реактивном режиме, ожидая прием данных и передачу управления, так и в проактивном режиме, запрашивая данные и управление от предшествующих модулей (агентов). При развертывании системы агентов на одной и той же платформе они сохраняют свойство мобильности, что способствует гибкости и надежности работы метакомпьютера.

Существенные отличия в реализации распределенного приложения по предлагаемой методике по сравнению с распространенными реализациями:

1) реализация конвейерного параллелизма, при котором новые запуски приложения инициируются без ожидания конечных результатов предыдущих инициаций;

2) модули распределенного приложения, реализуемые с применением технологии агентов, вносят дополнительную возможность выполнения параллельных вычислений, в том числе многоядерных и на основе графических ускорителей.

Проведенные на сети эксперименты позволили подтвердить работоспособность метакомпьютерного приложения и его способность к масштабированию и расширению функциональных возможностей вплоть до свойств облачно-сетевых технологий AaaS (Agent as a Service) и FaaS (Function as a Service).

В дальнейшем планируется адаптация под конкретные задачи и исследование работы распределенных агентно-базированных метакомпьютерных приложений в глобальной сети с волонтерскими компьютерами.

Список литературы

1. Cohen-almagor R. The Future of the Internet // Academia Letters. 2021. July. P. 1–5. doi:10.20935/AL1962
2. Digital economy report 2021. Cross-border data flows and development: for whom the data flow. Geneva, United Nations, 2021. 238 p.
3. Steen van M., Homburg P., Tanenbaum A. S. Globe: A Wide-Area Distributed System // IEEE Concurrency. 1999. № 1. P. 70–78.
4. Vaughan-Nichols S. J. Developing the distributed-computing OS // Computer. 2002. Vol. 35, № 9. P. 19–21.
5. Laouni D., Rachida M. Convergence of Grid and Peer-to-Peer Computing Monitoring and Resource Management in P2P GRID-based Web Services // International Journal of Scientific & Engineering Research. 2012. Vol. 3, № 9. P. 1–6.
6. Geist G. A., Kohl J. A., Papadopoulos P. M. PVM and MPI: a Comparison of Features // Calculateurs Paralleles. 1996. Vol. 8, № 2. P. 137–150.
7. Что такое метакомпьютинг (краткий обзор технологий организации распределенных вычислений в Интернете). URL: https://parallel.ru/computers/reviews/metacomputing.html, метод доступа свободный (дата обращения: 01.11.2021).
8. Introduction to Grid Computing Computing with Globus / IBM International Technical Support Organization, SG24-6895-01. 2003. 292 p.
9. Stevens R. L., Woodward P. R., DeFanti T. A., Catlett C. E. From the I-WAY to the National Technology Grid // Communications of the ACM. 1997. Vol. 40, № 11. P. 50–60. doi:10.1145/265684.265692
10. Segal B., Robertson L., Gagliardi F., Carminati F. Grid computing: the European Data Grid Project // Conference: Nuclear Science Symposium Conference Record, 2000 IEEE. 2000. Vol. 1. P. 1–7. doi:10.1109/NSSMIC.2000.948988
11. World Community Grid: Request for Proposal (RFP). 2020. P. 1–4. URL: https://www.worldcommunitygrid.org/bg/rfp.pdf, метод доступа свободный (дата обращения: 01.11.2021).

12. Kahanwal B., Tejinder Pal Singh 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.
13. Overeinder B. J., Wijngaards N. J. E., van Steen M. and Brazier F. M. T. Multi-Agent Support for Internet-Scale Grid Management // Proceedings of the AISB'02 Symposium on AI and Grid Computing. 2002. P. 18–22.
14. Cardoso R. C., Ferrando A. A Review of Agent-Based Programming for Multi-Agent Systems // Computers. 2021. Vol. 10, № 16. P. 1–15. doi:10.3390/computers10020016
15. Bellifemine F., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE Wiley Series in Agent Technology; John Wiley & Sons: Hoboken, NJ, USA, 2007. 286 p. doi:10.1002/9780470058411
16. 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
17. Bergenti F., Iotti E., Monica S., Poggi A. A comparison between asynchronous backtracking pseudocode and its JADEL implementation // Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART). 2017. Vol. 2. P. 250–258.
18. Bergenti F., Monica S., Petrosino G. A scripting language for practical agent-oriented programming // In Proceedings of the 8th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control. Boston, MA, USA, 2018. P. 62–71.
19. FIPA Specifications. URL: http://www.fipa.org/specifications/index.html, метод доступа свободный (дата обращения: 01.11.2021).
20. Волчихин В. И., Зинкин С. А., Карамышева Н. С. Организация функционирования облачно-сетевых распределенных вычислительных систем с архитектурой «агенты как сервисы» // Известия высших учебных заведений. Поволжский регион. Технические науки. 2019. № 4. С. 27–50. doi:10.21685/2072-3059-2019-4-3

References

1. Cohen-almagor R. The Future of the Internet. *Academia Letters*. 2021;July:1–5. doi:10.20935/AL1962
2. *Digital economy report 2021. Cross-border data flows and development: for whom the data flow*. Geneva, United Nations, 2021:238.
3. Steen van M., Homburg P., Tanenbaum A.S. Globe: A Wide-Area Distributed System. *IEEE Concurrency*. 1999;(1):70–78.
4. Vaughan-Nichols S.J. Developing the distributed-computing OS. *Computer*. 2002;35(9):19–21.
5. Laouni D., Rachida M. Convergence of Grid and Peer-to-Peer Computing Monitoring and Resource Management in P2P GRID-based Web Services. *International Journal of Scientific & Engineering Research*. 2012;3(9):1–6.
6. Geist G.A., Kohl J.A., Papadopoulos P.M. PVM and MPI: a Comparison of Features. *Calculateurs Paralleles*. 1996;8(2):137–150.
7. *Chto takoe metakomp'yuting (kratkiy obzor tekhnologiy organizatsii raspredelennykh vychisleniy v Internete) = What is metacomputing (a brief overview of the technology of organizing distributed computing on the Internet).* (In Russ.). Available at: https://parallel.ru/computers/reviews/meta-computing. html, metod dostupa svobodnyy (accessed 01.11.2021).
8. *Introduction to Grid Computing Computing with Globus*. IBM International Technical Support Organization, SG24-6895-01. 2003:292.
9. Stevens R.L., Woodward P.R., DeFanti T.A., Catlett C.E. From the I-WAY to the National Technology Grid. *Communications of the ACM*. 1997;40(11):50–60. doi:10.1145/265684.265692

10. Segal B., Robertson L., Gagliardi F., Carminati F. Grid computing: the European Data Grid Project. *Conference: Nuclear Science Symposium Conference Record, 2000 IEEE*. 2000;1:1–7. doi:10.1109/NSSMIC.2000.948988
11. *World Community Grid: Request for Proposal (RFP)*. 2020:1–4. Available at: https://www.worldcommunitygrid.org/bg/rfp.pdf, metod dostupa svobodnyy (accessed 01.11.2021).
12. Kahanwal B., Tejinder Pal Singh The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle. *International Journal of Latest Research in Science and Technology*. 2012;1(2):183–187.
13. Overeinder B.J., Wijngaards N.J.E., van Steen M. and Brazier F.M.T. Multi-Agent Support for Internet-Scale Grid Management. *Proceedings of the AISB'02 Symposium on AI and Grid Computing*. 2002:18–22.
14. Cardoso R.C., Ferrando A. A Review of Agent-Based Programming for Multi-Agent Systems. *Computers*. 2021;10(16):1–15. doi:10.3390/computers10020016
15. Bellifemine F., Caire G., Greenwood D. *Developing Multi-Agent Systems with JADE Wiley Series in Agent Technology*; John Wiley & Sons: Hoboken, NJ, USA, 2007:286. doi:10.1002/9780470058411
16. 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
17. Bergenti F., Iotti E., Monica S., Poggi A. A comparison between asynchronous back-tracking pseudocode and its JADEL implementation. *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART)*. 2017;2:250–258.
18. Bergenti F., Monica S., Petrosino G. A scripting language for practical agent-oriented programming. *In Proceedings of the 8th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control*. Boston, MA, USA, 2018:62–71.
19. *FIPA Specifications*. Available at: http://www.fipa.org/specifications/index.html, metod do-stupa svobodnyy (accessed 01.11.2021).
20. Volchikhin V.I., Zinkin S.A., Karamysheva N.S. Organization of the functioning of cloud-network distributed computing systems with the architecture "agents as services". Izvestiya *vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2019;(4):27–50. (In Russ.). doi:10.21685/2072-3059-2019-4-3

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: vt@pnzgu.ru

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

Анастасия Валерьевна Горынина
студентка, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

Anastasiya V. Gorynina
Student, Penza State University
(40 Krasnaya street, Penza, Russia)

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

E-mail: zsa49@yandex.ru

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

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

Поступила в редакцию / Received 12.11.2021

Поступила после рецензирования и доработки / Revised 26.11.2021

Принята к публикации / Accepted 11.12.2021