Компьютерные инструменты в образовании, 2022
№ 1: 30–45
УДК: 004.75:004.8
http://cte.eltech.ru
doi:10.32603/2071-2340-2022-1-30-45

АРХИТЕКТУРА МНОГОАГЕНТНЫХ СИСТЕМ ДЛЯ ФЕДЕРАТИВНОГО ОБУЧЕНИЯ

Гонсалес П. Ю.[1], аспирант, ✉ yuleisy2688@gmail.com
Холод И. И.[1], доктор технических наук, доцент, iiholod@mail.ru

[1]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), ул. Проф. Попова, 5, 197022, Санкт-Петербург, Россия

Аннотация

Концепция федеративного обучения получила широкое распространение в работе с данными, главным образом благодаря тому, что она позволяет проводить обучение по данным непосредственно на узлах, где они хранятся. В результате передача данных не требуется. После завершения обучения на каждом узле только обученная модель передается на центральный сервер для агрегации. Многоагентные системы ведут себя аналогичным образом, поскольку агенты позволяют обучать модели машинного обучения на локальных устройствах, сохраняя при этом конфиденциальную информацию. Способность агентов взаимодействовать друг с другом позволяет обобщать (агрегировать) такие модели и повторно использовать их.

В этой статье представлена архитектура многоагентных систем для федеративного обучения. Она выделяет элементы, которые составляют платформу агентов и структуру платформы JADE. Описывает жизненный цикл всех агентов, используемых для выполнения полного цикла обучения в среде MAC_FL. Анализируются и описываются конфигурации размещения агентов для каждой из предложенных архитектур многоагентных систем федеративного обучения: централизованной, децентрализованной и иерархической.

Ключевые слова: *агент, архитектура, федеративное обучение, многоагентные системы.*

Цитирование: Гонсалес П. Ю., Холод И. И. Архитектура многоагентных систем для федеративного обучения // Компьютерные инструменты в образовании. 2022. № 1. С. 30–45. doi: 10.32603/2071-2340-2022-1-30-45

1. ВВЕДЕНИЕ

Федеративное обучение (FL) — это развивающаяся парадигма распределенного машинного обучения, позволяющая нескольким устройствам обучать модели локально на клиентах и формировать глобальную модель без обмена данными. Проектирование системы FL требует учитывать различные варианты работы клиентов, сервера ивзаимодействия между ними [1, 2]. Концепция FL очень близка к многоагентным системам (MAC), поскольку агенты могут обучать модели машинного обучения на локальных устройствах (клиентах), сохраняя при этом конфиденциальную информацию. Возможности агентов взаимодействовать друг с другом позволяют обобщать (агрегировать) такие модели и повторно их использовать.

Архитектура системы определяет механизмы, которые позволяют агентам взаимодействовать с внешней средой, что заставляет их реагировать на изменения среды, воздействовать и общаться с ней и т. д.

При выборе архитектуры МАС необходимо иметь в виду два ее аспекта:

— архитектуру системы, включая взаимодействие агентов в процессе функционирования;
— архитектуру отдельного агента.

Проектирование архитектуры МАС для FL становится сложной задачей и проблемой, учитывая разнообразие стратегий, которые может реализовывать FL система [1].

Предложенная формальная модель в [3] послужила основой для разработки архитектуры МАС, используемой для задач FL. На основе выделенных типов агентов и видов сообщений разработаны: платформа агентов, сервисы и сами агенты с определенными жизненными циклами. Это позволяет строить гибкие и масштабируемые системы, реализующие методы федеративного обучения, на основе МАС.

2. АРХИТЕКТУРА ФЕДЕРАТИВНАЯ ОБУЧЕНИЯ

Архитектура FL систем включает в себя два элемента: сервер и клиенты [1]. *Сервер* — это отдельный компонент, который создает глобальную модель, инициализируя ее параметры. Помимо этого, сервер может предварительно обучить глобальную модель, используя либо самостоятельно созданный набор данных, либо небольшой объем данных, собранных с каждого клиента [4]. *Клиенты* — это компонент, который проводит обучение модели с использованием локально доступных наборов данных. Все клиенты получают исходную глобальную модель, расшифровывают и извлекают ее параметры. После этого они проводят обучение локальной модли [4].

В зависимости от их размещения, выделяют три основных типа архитектур FL систем: централизованный, децентрализованный и иерархический [5].

Централизованная архитектура представлена на рисунке 1а. В ней сервер инициирует и координирует процесс обучения, тогда как клиенты выполняют собственно обучение модели. После инициализации глобальной модели сервер передает глобальную модель участвующим клиентам. Глобальная модель может транслироваться на все клиенты в каждом раунде или только на определенные клиентские устройства, либо случайным образом, либо путем выбора, основанного на производительности обучения модели и доступности ресурсов. Точно так же обученные локальные модели собираются либо со всех участвующих клиентов, либо только с выбранных.

Наконец, сервер выполняет агрегирование модели, когда он получает все или определенное количество обновлений, с последующим перераспределением обновленной глобальной модели на клиенты. Весь этот процесс продолжается до тех пор, пока не будет достигнута конвергенция. Как правило, модель обучается за несколько раундов и эпох перед загрузкой обратно на центральный сервер для агрегирования модели [6]. После этого клиенты отправляют результаты обучения (параметры модели) обратно на центральный сервер для глобального агрегирования [7].

Этот тип архитектуры часто рассматривают на примере медицинских систем. В этом случае (рис. 1а) каждый клиентский узел (c) размещается в отдельной клинике, где осуществляется обучение модели (α) на данных о пациентах (d). Центральный сервер (s) может находиться в медицинском региональном центре, где будет проводиться агрегация (agg) модели (m).

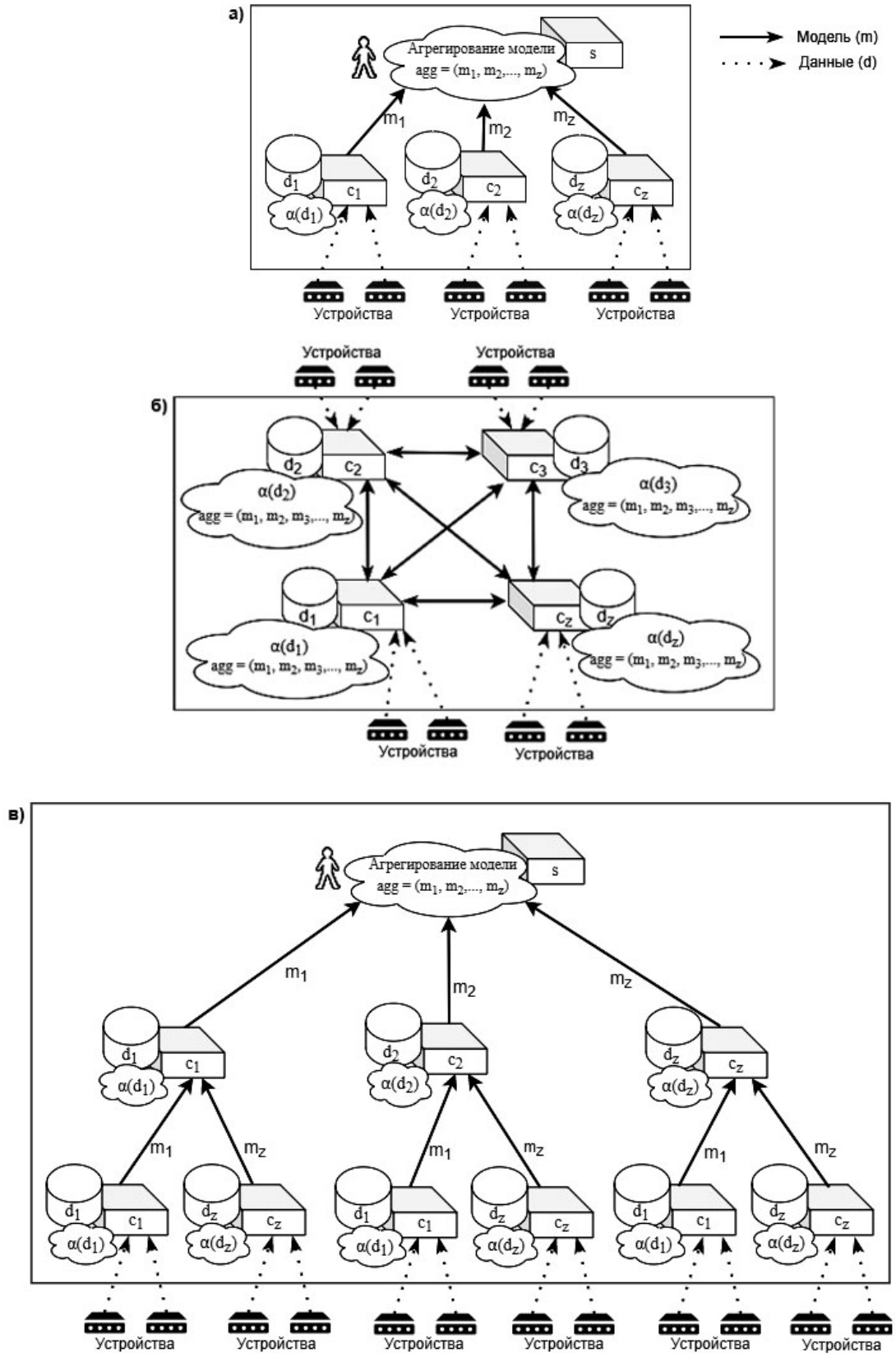

Рис. 1. Архитектуры FL: а) централизованная, б) децентрализованная, в) иерархическая

В децентрализованных FL системах (рис. 1б) клиенты обмениваются данными между собой без механизма управления сервера. Первоначальная модель создается локально каждым клиентским устройством с использованием локальных наборов данных. Модели обновляются с использованием подхода, основанного на консенсусе, который позволяет устройствам отправлять обновления моделей и получать параметры от узлов-соседей. Клиенты подключены через одноранговую сеть. На каждом устройстве есть копии обновленных моделей всех клиентов. После достижения консенсуса все клиенты будут проводить обучение модели с использованием нового градиента.

Этот тип архитектуры применяется на практике, например, в частных клиниках, конкурирующих между собой. В этом примере клиенты (c) размещаются в клинике, где осуществляется обучение модели (α) на данных о пациентах (d) и выполнение агрегации (agg) модели (m). Модели пересылаются между клиниками, которые являются равноправными в рамках федеративного обучения.

Иерархическая архитектура FL систем разделяет ее на суб-федерации с централизованной и/или децентрализованной схемами (рис. 1в).

Такая архитектура может быть использована в случае иерархической подчиненности. Например, клиенты (c) размещаются в клинике, где осуществляется обучение модели (α) на данных о пациентах (d). Эта модель отправляется другим клиентам, которые могут находиться, например, в других клиниках этого региона и/или в региональный медицинский центр. В них также может происходить обучение модели, и отсюда устанавливается связь с центральным узлом (s), который может находиться в федеральном центре, где будет осуществляться агрегация (agg) модели (m).

3. АРХИТЕКТУРА МНОГОАГЕНТНЫХ СИСТЕМ

3.1. Спецификация FIPA

Архитектура MAC описана в спецификациях ассоциации FIPA, которые определяют характеристики платформы управления MAC, служб и агентов. Помимо реализаций, изначально задуманных на основе этого стандарта (таких как FIPA OS и JADE), в настоящее время большинство сред разработки и исполнения агентов, как правило, адаптированы или совместимы с FIPA.

Спецификации FIPA влияют не только на методы взаимодействия между агентами, но и на базовую архитектуру (рис. 2), которую должна реализовывать MAC, соответствующая данному стандарту. Согласно архитектуре FIPA средой существования агентов являются агентные платформы [8].

Агентная платформа (AP — Agent Platform) — это промежуточное программное обеспечение, поддерживающее создание, интерпретацию, запуск, перемещение и уничтожение агентов. Она включает в себя следующие компоненты [8]:

1. Система управления агентом (AMS — Agent Management System) управляет созданием, удалением, деактивацией, возобновлением, аутентификацией и миграцией агентов, находящихся на платформе.
2. Служба каталога (DF — Directory Facilitator) реализует сервис «желтых страниц», который хранит описание агентов.
3. Менеджер безопасности платформы агентов (APSM — Agent Platform Security Manager) отвечает за осуществление политики безопасности на транспортном уровне и проверку выполнения операций управления агентами.

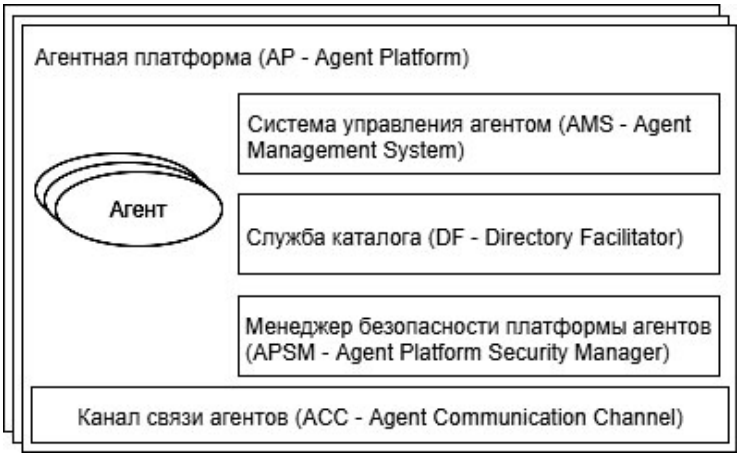

Рис. 2. Агентная платформа

4. Канал связи агентов (ACC — Agent Communication Channel) использует информацию, предоставляемую системой управления агентов для маршрутизации сообщений между агентами.

3.2. Жизненный цикл агентов FIPA

Агенты FIPA физически существуют на AP и используют возможности, предлагаемые AP, для реализации своих функций. В этом контексте агент как физический программный процесс имеет жизненный цикл. На рисунке 3 представлен унифицированный жизненный цикл [9**?**].

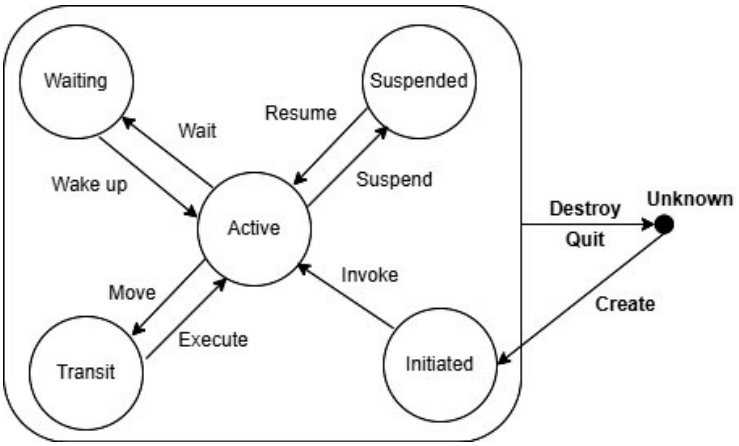

Рис. 3. Модель жизненного цикла агента

Ниже перечислены состояния жизненного цикла агента и действия, которые AMS выполняет от имени AP в каждом из них:

1. Активный (Active): служба передачи сообщений (СПС) оставляет сообщения агенту, как обычно.
2. Инициировано / Ожидание / Приостановлено (Initiated / Waiting / Suspended): СПС либо буферизует сообщения, пока агент не вернется в активное состояние, либо пересылает сообщения в новое место (если для агента установлена переадресация).

3. Транзит (Transit): СПС либо буферизует сообщения, до тех пор пока агент не станет активным, либо пересылает сообщения в новое место. Только мобильные агенты могут иметь состояние «Транзит».
4. Неизвестно (Unknown): СПС либо буферизует сообщения, либо отклоняет их.

Переходы между состояниями агентов можно описать как:

- Создавать (Create): создание или установка нового агента.
- Вызвать (Invoke): вызов нового агента.
- Уничтожить (Destroy): принудительное увольнение агента.
- Условно прекратить (Quit): предложение на удаление агента, которое он может игнорировать.
- Приостановить (Suspend): переводит агента в приостановленное состояние.
- Состояние для продолжения работы (Resume): выводит агента из приостановленного состояния.
- Состояние ожидания (Wait): переводит агента в состояние ожидания.
- Пробуждающее (Wake up): выводит агента из состояния ожидания.

3.3. Агентные платформы

Было разработано множество программных реализаций агентных платформ, каждая из которых имеет свои особенности, достоинства и недостатки. Одной из реализаций стандарта FIPA является проект с открытым кодом (JADE — Java Agents Development Framework). Это настраиваемая стандартная среда для разработки распределенных многоагентных приложений [**?**].

Платформа JADE — это одновременно среда, в которой агенты получают услуги связи, необходимые для взаимодействия агентов. Кроме того, JADE — это платформа, которая позволяет агентам легко иметь свои внутренние свойства, связанные с их способностью переходить на удаленные серверы, управляемые многоагентной платформой. Платформа JADE выделяется среди других многоагентных платформ соответствием спецификации FIPA и полностью разработана на языке программирования Java [10].

В соответствии со стандартом FIPA, платформа системы JADE, включает в себя DF и ACC.

Платформа JADE является распределенной и представляет собой набор контейнеров (рис. 4).

4. АРХИТЕКТУРА MAC ДЛЯ ФЕДЕРАТИВНОГО ОБУЧЕНИЯ

Архитектура — это наиболее общее описание системы, объясняющее ее назначение и принципы работы. MAC определяется архитектурой, которая определяет структуру и топологию его агентов; со своей стороны, архитектура FL определяется структурой и топологией между его компонентами. Структурными элементами архитектуры являются функциональные модули, выполняющие определенную работу под управлением модуля управления системой.

В настоящее время большинство систем интеллектуального анализа данных, использующих многоагентные технологии, реализовано на основе JADE и использует FIPA_ACL [**?**]. В связи с этим, предложенная архитектура MAC для FL (MAC_FL) будет рассмотрена также с использованием фреймворка JADE, однако может быть адаптирована

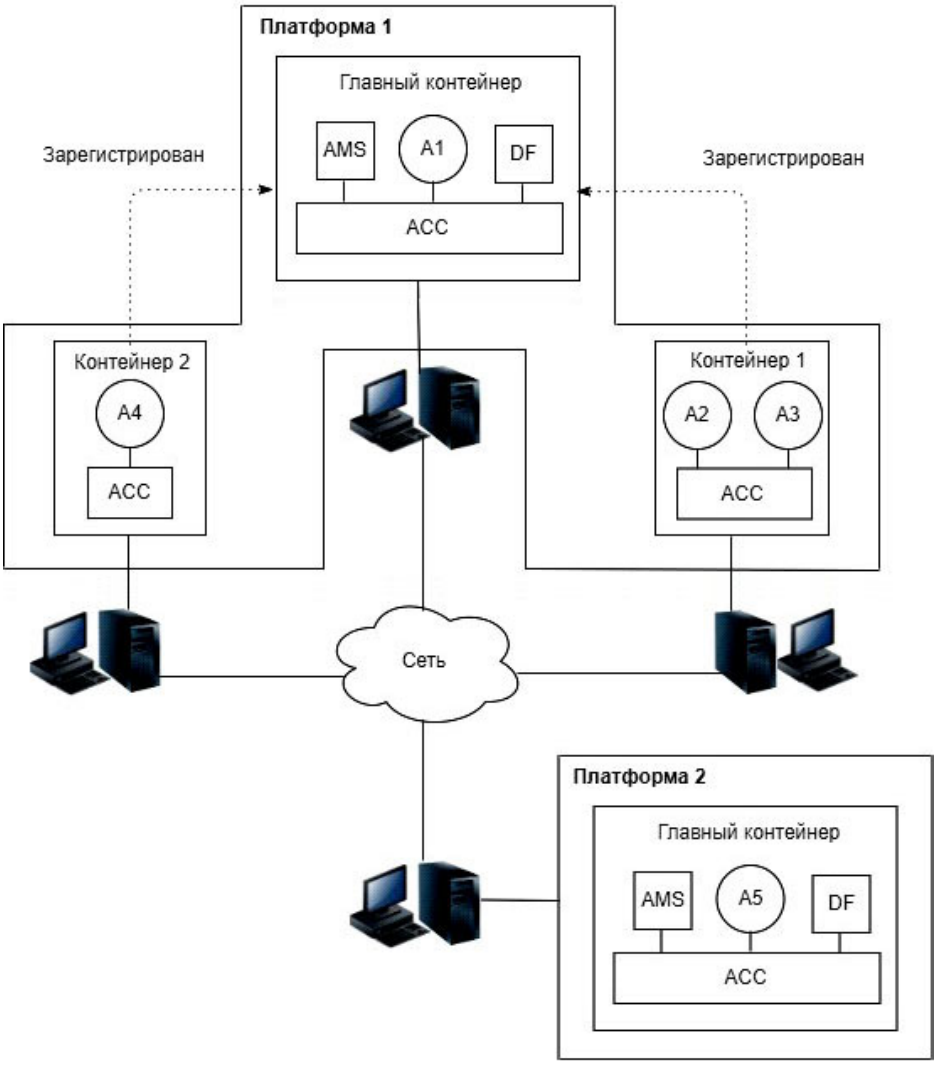

Рис. 4. Платформа JADE

для любого другого фреймворка, соответствующего стандарту FIPA, так как использует службы, предписанные им.

Архитектура MAC_FL будет зависеть от стратегий мета-обучения, описанных в [3]. В дополнение к необходимым агентам AP, находящимся в основном контейнере, она должна включать в себя агентов для FL:

— агент пользователя (**UA — User Agent**) взаимодействует с пользователем и принимает от него задачу на обучение;
— агент посредник (**FA — Facilitator Agent**) собирает информацию о среде выполнения;
— агент задачи (**TA — Task Agent**) координирует работу всех агентов для выполнения задачи пользователя;
— агенты ИАД (**DMA — Data Mining Agent**) выполняют алгоритм обучения на данных;
— агент данных (**DA — Data Agent**) взаимодействует с данными;
— агент агрегации (**AA — Aggregate Agent**) выполняет объединение моделей.

На рисунке 5 показана MAC_FL для централизованной архитектуры FL. В главном контейнере начинается и координируется процесс обучения. В нем размещаются следующие агенты: UA, TA, FA и AA. В других контейнерах выполняется фактическое обучение модели, и после ее готовности она отправляется в главный контейнер для агрегации. В этих контейнерах будут размещаться только агенты DMA и DA.

Службы AP: AMS, DF и ACC будут находиться на главном контейнере, который будет связан со всей средой MAC_FL. При этом:

— AMS управляет агентами;
— DF хранит информацию о контейнерах на клиентах и агентах на них;
— ACC реализует канал взаимодействия между агентами.

ACC будет находиться на всех контейнерах для осуществления связи между ними.

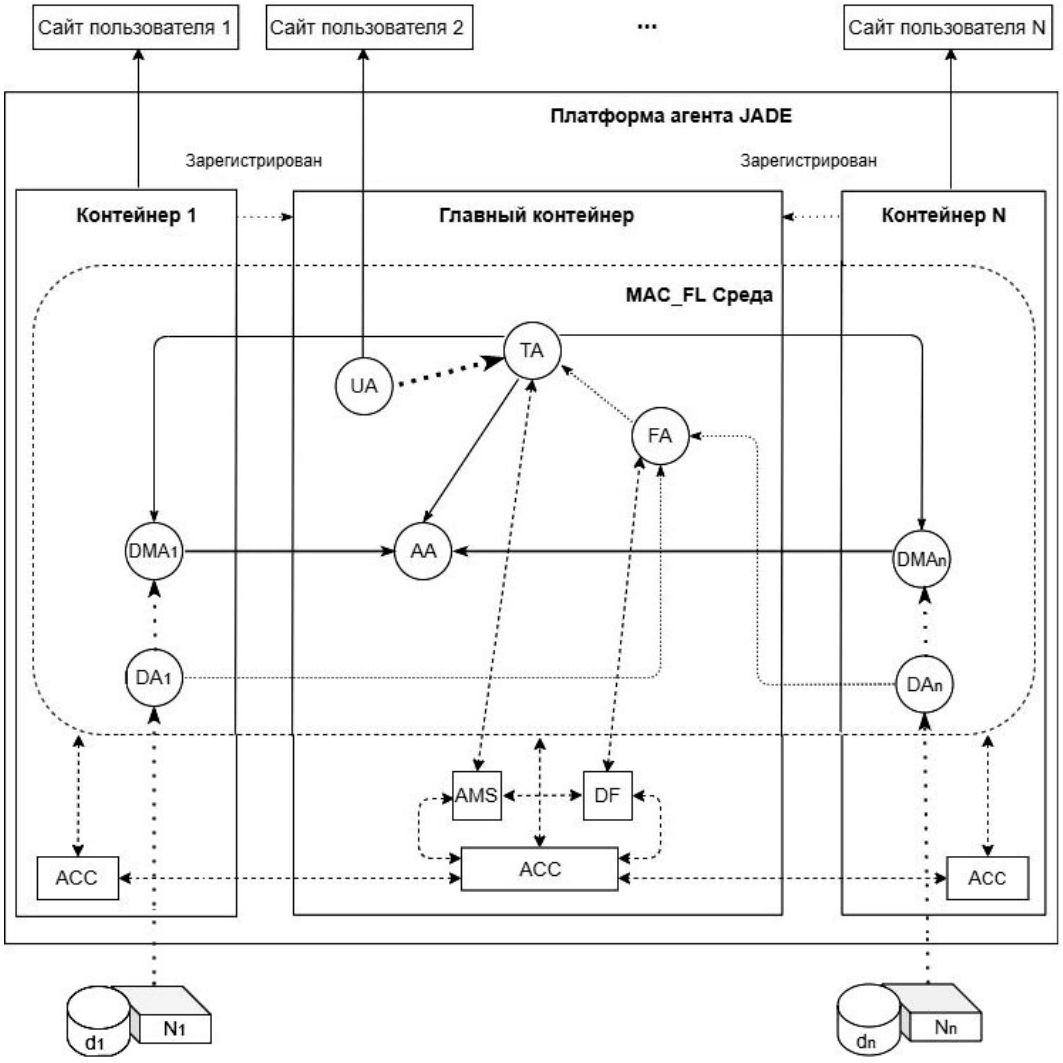

Рис. 5. Централизованная архитектура MAC_FL

На рисунке 6 представлена MAC_FL для децентрализованной архитектуры FL. В главном контейнере начинается и координируется процесс обучения. В нем будут размещены следующие агенты: UA, TA, FA, DMA и AA. В других контейнерах будут размещаться

только агенты DMA, AA и DA. Во всех контейнерах проводится обучение модели. После получения всех моделей от DMA агенты AA объединяют их и возвращают результаты обратно DMA. Обмен между AA и DMA происходит до завершения обучения.

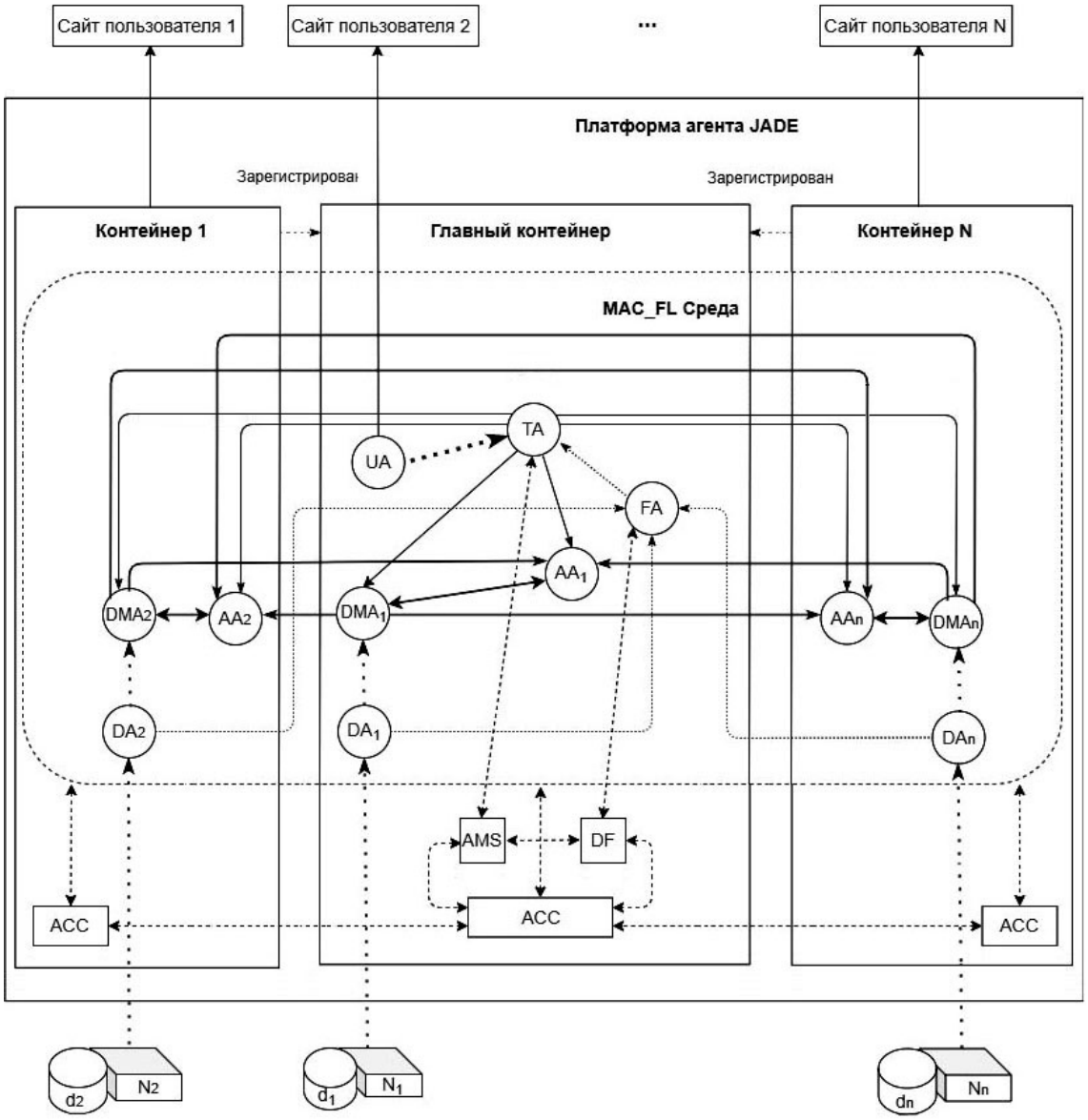

Рис. 6. Децентрализованная архитектура MAC_FL

На рисунке 7 представлена MAC_FL для иерархической архитектуры FL. В главном (корневом) контейнере начинается и координируется процесс обучения. В нем будут размещены следующие агенты: UA, TA, FA и AA. Платформы могут реализовать централизованные и/или децентрализованные архитектуры, описанные выше. В зависимости от архитектуры, каждый контейнер будет содержать разный состав агентов, как показано на рисунках 5 и 6. Из главного контейнера задача будет отправлена в главные контейнеры каждой платформы. После завершения процесса обучения на каждой из этих платформ модель будет отправлена в корневой контейнер для агрегации.

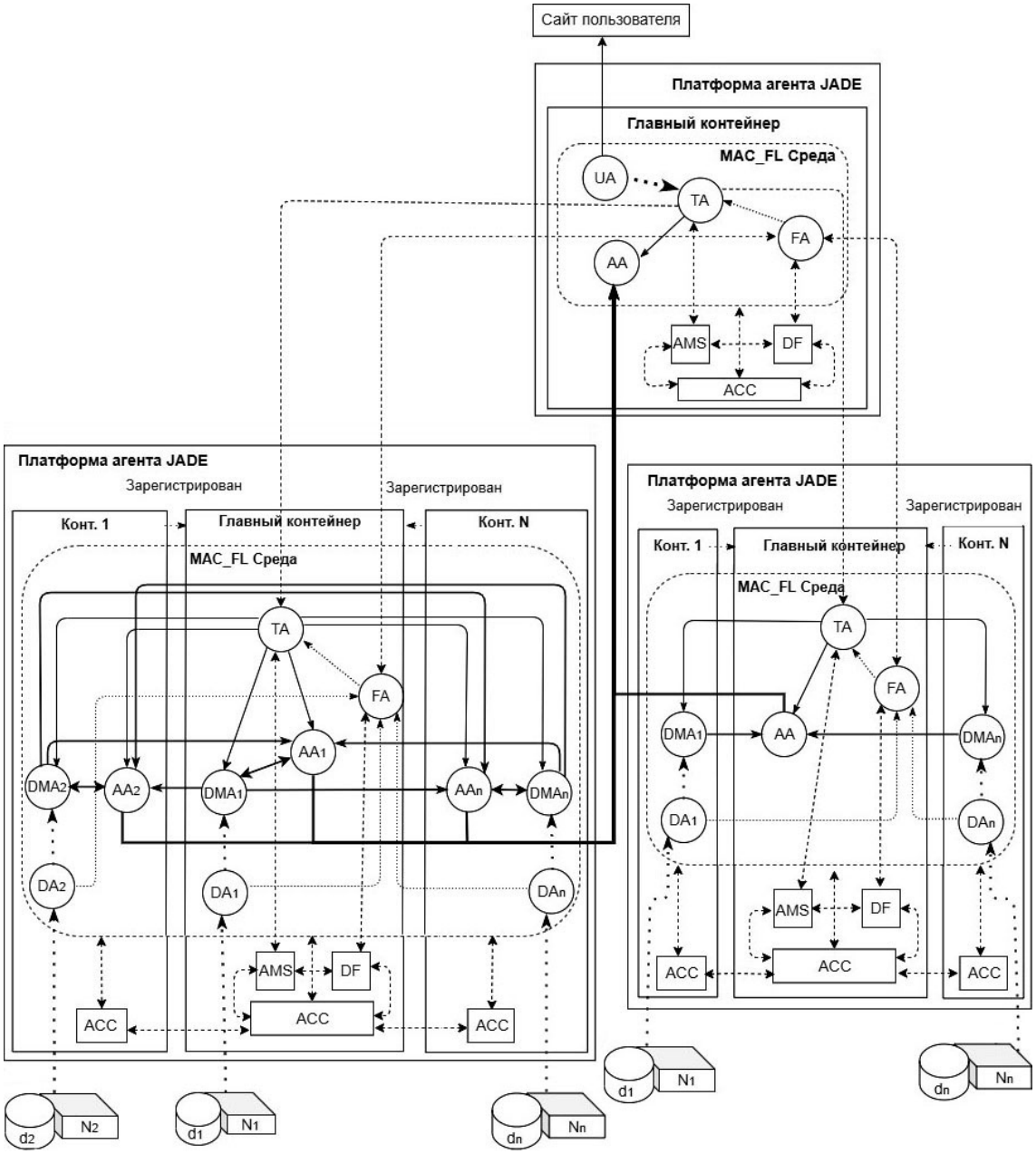

Рис. 7. Иерархическая централизованная архитектура MAC_FL

5. ЖИЗНЕННЫЙ ЦИКЛ АГЕНТОВ В СРЕДЕ MAC_FL

Жизненный цикл агентов в модели MAC_FL дополняется состояниями, зависящими от действий, выполняемых в FL системе (на рисунках выделены серым цветом):

1) состояние Collect, в котором агент собирает данные и информацию со всех агентов;
2) состояние Schedule, в котором агент создает план;
3) состояние Sending, в котором агент отправляет результат назначенной задачи (план, модели, информация, данные);
4) состояние Interpret, в котором агент интерпретирует данные для создания плана;

5) состояние Training, в котором агент обучает модель на локальных данных;
6) состояние Agg, в котором агент агрегирует модели, которые были обучены на клиентах.

AMS создает всех агентов, которые будут оставаться в состоянии ожидания, пока им не будет назначены задачи.

Выполнение MAC_FL начинается, когда UA получает задачу от пользователя, интерпретирует ее и передает ее агенту TA (рис. 8).

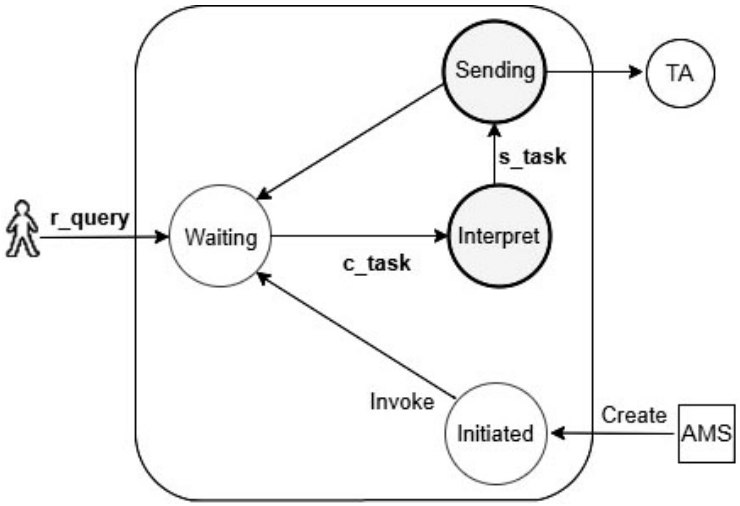

Рис. 8. Жизненный цикл UA в среде MAC_FL

DF обменивается данными и информацией с FA, который регистрирует агентов, созданных в среде. FA запрашивает информацию от DA о данных, доступных на узлах, и передает эту информацию (info) TA (рис. 9).

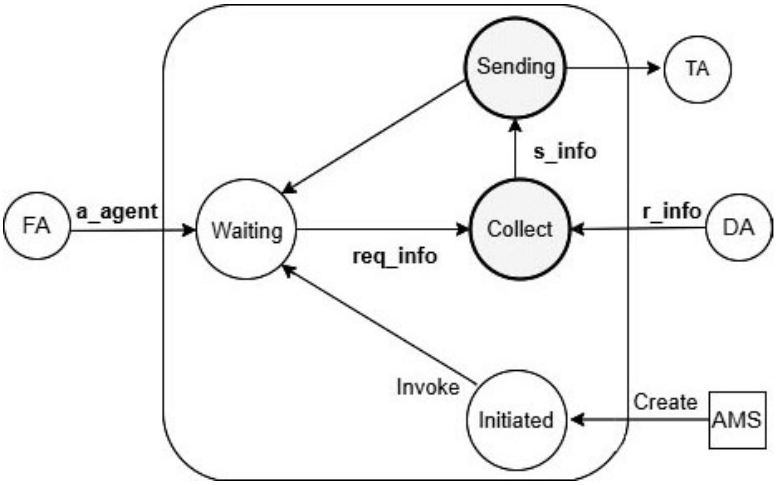

Рис. 9. Жизненный цикл FA в среде MAC_FL

TA получает всю информацию, необходимую для выполнения FL: задачу пользователя и информацию о среде, в которой выполняется анализ. На основе этой информации TA генерирует план выполнения FL. После этого, в зависимости от используемой архитектуры будут выполнены разные действия. В случае централизованной и децентрализован-

ной архитектуры сгенерированный план передается для исполнения AA и DMA. В случае иерархической архитектуры сформированный план отправляется на исполнение из корневого контейнера в TA, расположенных в основных контейнерах платформ, (рис. 10).

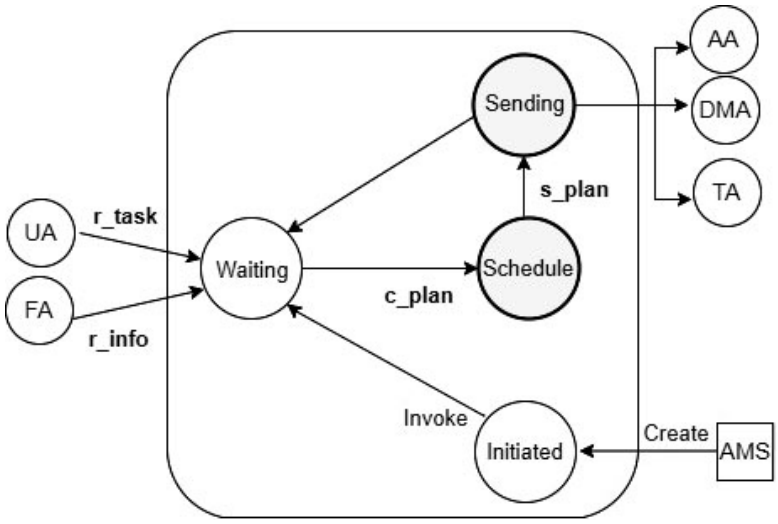

Рис. 10. Жизненный цикл TA в среде MAC_FL

В случае централизованной архитектуры каждый DMA, получив план, обращается к DA за данными (qdata) и получает доступ к данным (data) с его помощью. На этих данных DMA обучает модель (model) и передает ее AA. В случае децентрализованной архитектуры на этих данных DMA обучает модель (model) и передает ее всем AA. В случае централизованной иерархической архитектуры, когда каждая платформа заканчивает обучение, она отправляет модель (model) в AA, расположенный в центральном контейнере, для агрегации (рис. 11).

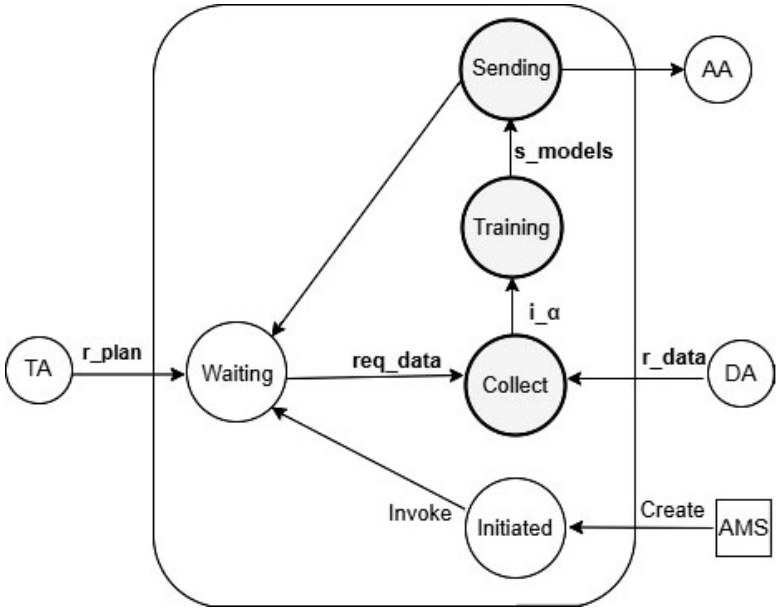

Рис. 11. Жизненный цикл DMA в среде MAC_FL

DA отправляет данные (data) и информацию в FA и DMA, (рис. 12).

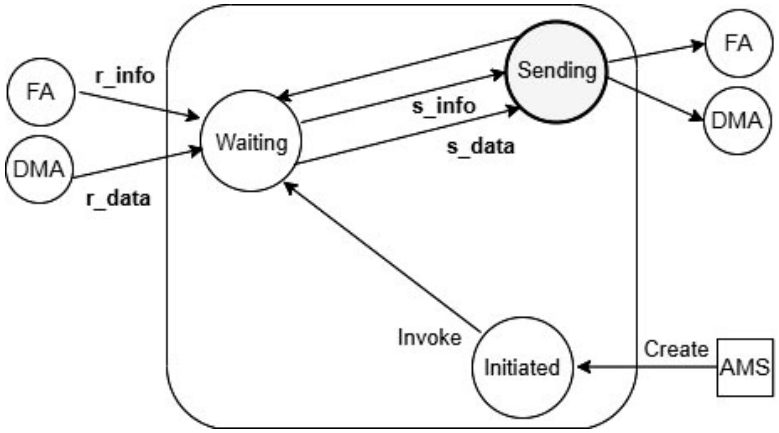

Рис. 12. Жизненный цикл DA в среде MAC_FL

В случае централизованной архитектуры после получения всех моделей от DMA AA объединяет их и возвращает DMA. В случае децентрализованной архитектуры после получения моделей от всех DMA каждый AA объединяет их и возвращает локальному DMA. В случае централизованной иерархической архитектуры после получения моделей всех AA, находящихся в основном контейнере платформ, AA, находящийся в центральном контейнере, добавляет их (рис. 13).

Обмен между AA и DMA осуществляется до завершения обучения.

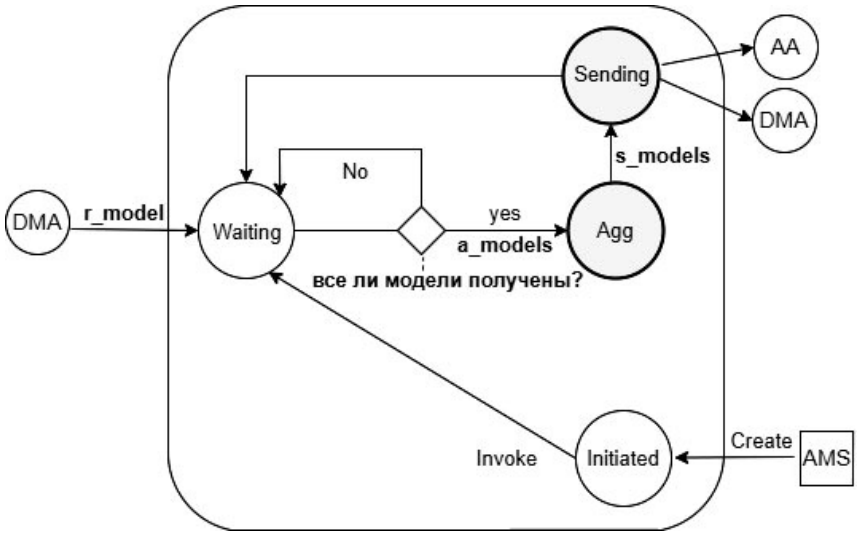

Рис. 13. Жизненный цикл AA в среде MAC_FL

6. ЗАКЛЮЧЕНИЕ

Исследование позволило построить три типа архитектур: централизованную, децентрализованную и иерархическую для среды MAC_FL. Для них описаны процессы, которые агенты выполняют, их действия и взаимосвязи в среде. Описание жизненного цикла каж-

дого из агентов, составляющих среду, позволило понять, как будет осуществляться связь и координация для получения обученной модели FL.

Предлагаемая архитектура может служить основой для разработки систем федеративного обучения на основе МАС. Это, в свою очередь, позволит создавать гибкие и масштабируемые системы.

Дальнейшие исследования будут направлены на повышение эффективности предложенного подхода с точки зрения производительности и оперативности решения задач.

Список литературы

1. *Sin Kit L., Qinghua L., Liming Z., Hye-Young P., Xiwei X., Chen W.* Architectural Patterns for the Design of Federated Learning Systems // arXiv. Jun 2021. P. 1.
2. *Sin Kit L., Qinghua L., Hye-Young P., Liming Z.* FLRA: A Reference Architecture for Federated Learning Systems // arXiv. Jun 2021. P. 1.
3. *Гонсалес Ю., Холод И.* Формальная модель многоагентных систем для федеративного обучения // Программные продукты и системы. Т. 35, вып. 1, 2022. С. 37–44.
4. *Sin Kit L., Qinghua L., Chen W., Hye-Young P., Liming Z.* A Systematic Literature Review on Federated Machine Learning: From A Software Engineering Perspective // arXiv. May 2021. P. 20.
5. *Kairouz P., Mcmahan B. H., Avent B.* Advances and Open Problems in Federated Learning // arXiv. Т. 14, вып. 1, 2021. С. 4–11.
6. *Guha N., Talwalkar A., Smith V.* One-Shot Federated Learning // arXiv. Mar 2019. P. 3.
7. *Tzu-Ming H. H., Hang Q., Matthew B.* Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification // arXiv. Sep 2021. P. 3.
8. *Agüero J. L.* Diseño de organizaciones virtuales ubícuas utilizando desarrollo dirigido por modelos // Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Tesis Doctorado 2014.
9. FIPA. Foundation for Intelligent Physical Agents. [Электронный ресурс], http://www.fipa.org/specs/fipa00023/XC00023H.html (дата обращения: 07.03.2022).
10. *Aguayo F. J.* Técnicas para despliegue de arquitectura distribuida en sistemas expertos basados en reglas empleando el paradigma multiagente // Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León , PhD, 2017.

Поступила в редакцию 07.02.2022, окончательный вариант — 24.03.2022.

Гонсалес Перес Юлейси, аспирант кафедры вычислительной техники факультета компьютерных технологий и информатики СПбГЭТУ «ЛЭТИ», ✉ yuleisy2688@gmail.com

Холод Иван Иванович, доктор технических наук, доцент, декан факультета компьютерных технологий и информатики СПбГЭТУ «ЛЭТИ», iiholod@mail.ru

Computer tools in education, 2022
№ 1: 30–45
http://cte.eltech.ru
doi:10.32603/2071-2340-2022-1-30-45

Multi-agent Architecture for Federated Learning

González Y. [1], Postgraduate, ✉ yuleisy2688@gmail.com
Kholod I. I.[2], PhD, Associate Professor, iiholod@mail.ru

[1]Saint Petersburg Electrotechnical University,
5, building 3, st. Professora Popova, 197022, Saint Petersburg, Russia

Abstract

The concept of federated learning has become widespread in working with data, mainly due to the fact that it allows training on data directly on the nodes where they are stored. As a result, no data transfer is required. After the training is completed on each node, only the trained model is transmitted to the central server for aggregation. Multi-agent systems behave in a similar way, because agents allow you to train machine learning models on local devices, while preserving confidential information. The ability of agents to interact with each other makes it possible to generalize (aggregate) such models and reuse them.

This article presents the architecture of multi-agent systems for federated learning. It highlights the elements that make up the agent platform and the structure of the JADE platform. Describes the lifecycle of all agents used to perform a full training cycle in the MAC_FL environment. The configurations of agent placement for each of the proposed architectures of multi-agent systems of federated learning are analyzed and described: centralized, decentralized and hierarchical.

Keywords: *agent, architecture, federated learning, multi-agent systems.*

Citation: Y. PGonzález and I. I. Kholod, “Multi-agent Architecture for Federated Learning,” *Computer tools in education*, no. 1, pp. 30–45, 2022 (in Russian); doi: 10.32603/2071-2340-2022-1-30-45

References

1. L. Sin Kit, L. Qinghua, Z. Liming, P. Hye-Young, and X. Xiwei, W. Chen, “Architectural Patterns for the Design of Federated Learning Systems,” in *arXiv*, Jun 2021, p. 1.
2. L. Sin Kit, L. Qinghua, P. Hye-Young, and Z. Liming, “FLRA: A Reference Architecture for Federated Learning Systems,” in *arXiv*, Jun 2021, p. 1.
3. Y. González and I. Holod, “A formal model of multiagent systems for federated learning,” *Software & Systems*, vol. 35, no.1, pp. 37–44, 2022 (in Russian); doi: 10.15827/0236-235X.137.037-044
4. L. Sin Kit, L. Qinghua, W. Chen, P. Hye-Young, and Z. Liming, “A Systematic Literature Review on Federated Machine Learning: From A Software Engineering Perspective,” in *arXiv*, pp. 1–20, May 2021.
5. P. Kairouz, B. H. Mcmahan, and B. Avent, “Advances and Open Problems in Federated Learning,” in *arXiv*, vol. 14, no. 1, 2021. pp. 4–11.
6. N. Guha, A. Talwalkar, and V. Smith, “One-Shot Federated Learning,” in *arXiv*, Mar 2019, pp. 1–3.
7. H. H. Tzu-Ming, Q. Hang, and B. Matthew, “Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification,” in *arXiv*, Sep 2021, pp. 1–3.

8. J. L. Agüero, *Design of ubiquitous virtual organizations using model-driven development*, [PhD Thesis], Department of Information Systems and Computing, Polytechnic University of Valencia, Valencia, Spain, 2014.
9. FIPA, “Foundation for Intelligent Physical Agents,” in *fipa.org*, 03 Oct. 2001. [Online]. Available: http://www.fipa.org/specs/fipa00023/XC00023H.html
10. G. Tsochev, R. Trifonov, and R. Yoshinov, “Multi-agent framework for intelligent networks,” in *Proc. of 29th International Conference on Information Technologies (InfoTech-2015), 17–18 Sept 2015, Varna, Bulgaria*, Sofia, 2015, pp. 109–116.
11. N. Cepero and M. Moreno, “Transformación del Q-Learning para el Aprendizaje en Agentes JADE,” *Lámpsakos*, no. 14, pp. 25–32, 2015.
12. F. J. Aguayo, *Techniques for deployment of distributed architecture in rule-based expert systems using the multi-agent paradigm*, [PhD Thesis], Department of Electrical and Systems Engineering and Automation, University of León, France, 2017.
13. Y. González and I. Holod, “Analysis of Multiagent System for Data Analysis,” in *Proc. of XXIII International Conference on Soft Computing and Measurements (SCM), Sept. 2020, St. Petersburg, Russia*, St. Petersburg, Russia, 2020, pp. 218–221; doi: 10.1109/SCM50615.2020.9198765

Received 07-02-2022, the final version — 24-03-2022.

Yuleisy González Pérez, Postgraduate of the Department of Computer Engineering of the Faculty of Computer Technologies and Informatics of Saint Petersburg Electrotechnical University, ✉ **yuleisy2688@gmail.com**

Ivan Kholod, PhD, Associate Professor, Dean of the Faculty of Computer Technologies and Informatics of Saint Petersburg Electrotechnical University, iiholod@mail.ru