Адаптивная мультиагентная операционная система реального времени[1]

К. С. Амелин, к. ф.-м. н, *М. В. Баклановский*,
О. Н. Граничин, д. ф.-м. н., *Ю. В. Иванский, А. Д. Корнивец, Н. В. Мальковский, Д. Г. Найданов, Р. Е. Шеин*[2]
Санкт-Петербургский государственный университет
(konstantinamelin, oleg_granichin)@mail.ru

Сегодняшняя тенденция развития вычислительной техники — переход от приоритетов бесконечного наращивания тактовой частоты и мощности одного процессора к многоядерности, параллелизму, к объединению отдельных процессоров в комплексы. Но и у подхода комбинирования вычислительных узлов в одной системе есть свои ограничения. Обычно такие системы обладают жесткой иерархией, что с одной стороны, ограничивает область их эффективного применения в силу нацеленности конкретной конфигурации вычислительной системы на решение конкретного круга задач; с другой стороны, отказ малого числа узлов на верхних уровнях иерархии может сказаться на работе всей системы. Кроме того, серьезные трудности возникают при планировании загрузки вычислительных узлов из-за их большого количества и сложной системы коммутации. При создании крупных систем для преодоления подобных сложностей все чаще применяют идеологию мультиагентных систем. В таких системах нет жестко заданных связей между элементами и каждый элемент — агент — обладает определенной самостоятельностью и способен образовывать связи с другими агентами в процессе решения задач по мере необходимости.

В статье описываются основные черты проекта авторов, нацеленного на разработку прототипа новой операционной системы реального времени (ОСРВ), предназначенной для организации работы комплексов из большого количества вычислительных узлов и базирующейся на принципах мультиагентных технологий. Такая децентрализованная ОСРВ должна обеспечить эффективность загрузки вычислительных устройств и иметь способности к изменению топологии сети связей между узлами, вызванной как структуризацией сети в процессе решения задачи, так и потерей или добавлением узла.

Ключевые слова: адаптивные системы, мультиагентные системы, операционная система реального времени.

1. Введение

Начало построению моделей и применению искусственных мультиагентных систем (МАС) на практике было положено в 1960-х годах. В качестве основы были взяты достижения таких областей дея-

[1]Работа выполнена в рамках исследований в СПбГУ по теме №6.38.71.2011 “Изучение свойств и возможностей применения рандомизированных алгоритмов кластеризации, оптимизации и оценивания” и при поддержке РФФИ (проект №13-07-00250-а).

[2]©К. С. Амелин, М. В. Баклановский, О. Н. Граничин, Ю. В. Иванский, А. Д. Корнивец, Н. В. Мальковский, Д. Г. Найданов, Р. Е. Шеин, 2013

тельности человека, как системы искусственного интеллекта (Artificial Intelligence), параллельные вычисления (Parallel Computing), распределенное решение задач (Distributed Problem Solving). Многоагентные системы имеют реальную возможность интегрировать в себе самые передовые достижения перечисленных областей, демонстрируя принципиально новые качества. Сейчас МАС — одно из наиболее динамично развивающихся и перспективных направлений в области искусственного интеллекта.

МАС являются популярным методом решения сложных задач, плохо решаемых в рамках классического централизованного подхода [1]. Традиционная схема организации многопроцессорной вычислительной системы предполагает наличие управляющего узла, задачей которого является распределение заданий между остальными узлами системы. Такой подход к организации работы системы имеет ряд недостатков.

• При увеличении числа узлов в системе возрастает сложность организации их взаимодействия.

• Обычно такие системы способны быстро решать только определенный круг задач, для решения которых применяется та или иная декомпозиция, отраженная в топологии вычислительной сети.

• Кроме того, при выходе из строя главного узла, отвечающего за распределение задач, остановится работа всей системы (если такая ситуация не предусмотрена при проектировании системы).

Уже сейчас во многих областях применяются МАС для преодоления подобного рода ограничений в случае большого числа узлов в системе, для обеспечения отказоустойчивости системы, если нужно выполнять разнородные задачи или если алгоритм точного решения слишком сложен, не найден, или его не существует. Суть мультиагентных технологий заключается в принципиально новом методе решения задач. В отличие от классического способа, когда проводится поиск некоторого четко определенного (детерминированного) алгоритма, позволяющего найти наилучшее решение проблемы, в мультиагентных технологиях решение получается автоматически в результате взаимодействия множества самостоятельных целенаправленных программных модулей — агентов.

Таким образом, мультиагентные технологии способны не только организовать работу распределенной системы, но и самостоятельно найти способ решения задачи. Мультиагентные технологии нахо-

дят применение для решения задач управления формациями, роения, управления перегрузкой в сетях связи, управления группами БПЛА, относительного выравнивания групп спутников, в автоматизации управления ресурсами предприятий в реальном времени, для сбора и обработки информации в таких сферах, как проектирование объектов, промышленное производство, финансовое планирование и анализ рисков, распознавание образов, извлечение знаний из данных, понимание текста и решение других сложных проблем [2].

2. Агенты и мультиагентные системы

Часто утверждается, что агенты не просто совершают действия, но они действуют автономно и рационально. Под автономностью обычно понимают, что агент действует без прямого вмешательства человека или другой управляющей сущности. Под рациональностью понимают стремление агента оптимизировать значение некоторой оценочной функции. Мера рациональности неявно указывает на то, что агент имеет цели (желания, англ. desires), которых агент хочет достичь, и представления о внешнем мире (убеждения, англ. beliefs), на которые агент опирается при выборе действия (реализации намерений, англ. intentions — множество избранных, совместимых и достижимых желаний). Еще одним важным свойством агента является то, что он помещен во внешнюю среду, с которой он способен взаимодействовать. Обычно среда не контролируется агентом, он лишь способен влиять на нее.

Разделение намерений и желаний необходимо, так как агент может иметь несовместимые желания или желания могут быть недостижимы. Поскольку агент ограничен в ресурсах и не может достичь всех желаний одновременно, естественно выбирать наиболее значимые цели — намерения. Итак, агент — разумная сущность, помещенная во внешнюю среду, способная взаимодействовать с ней, совершая автономные рациональные действия для достижения целей, т. е. интеллектуальный агент — агент, обладающий следующими свойствами [2]: реактивность (англ. reactivity) — агент ощущает внешнюю среду и реагирует на изменения в ней, совершая действия, направленные на достижение целей; проактивность (англ. pro-activeness) — агент показывает управляемое целями поведение, проявляя инициативу, совершая действия, направленные на дости-

жение целей; социальность (англ. social ability) — агент взаимодействует с другими сущностями внешней среды (другими агентами, людьми и т. п.) для достижения целей. Отдельно стоит отметить такую особенность агентов, как адаптивность, то есть способность автоматически приспосабливаться к неопределенным и изменяющимся условиям в динамической среде, которая часто оказывается очень полезной на практике, когда невозможно задать все условия функционирования системы.

Многочисленные агенты, находясь в одной среде, взаимодействуя друг с другом, образуют системы, характеризующиеся такими свойствами, как стохастичность, гибридность, асинхронность, кластерность.

Благодаря все более активному применению мультиагентных технологий на практике для решения различного рода задач, к ним растет интерес со стороны исследователей.

В настоящее время наибольшей популярностью при создании мультиагентных систем пользуются следующие средства [2]:

• JADE (JavaAgentDevelopmentFramework) — широко используемая программная среда для создания МАС и приложений, поддерживающая FIPA-стандарты для интеллектуальных агентов. Включает в себя среду выполнения агентов (агенты регистрируются и работают под управлением среды), библиотеку классов, которые используются для разработки агентных систем, набор графических утилит для администрирования и наблюдения за жизнедеятельностью активных агентов. Программная среда JADE подключается к любому проекту на языке Java. Агенты JADE могут быть совершенно разными: от простых, только реагирующих, до сложных ментальных.

• JACK Intelligent Agents — Java-платформа для создания МАС. Так же, как и JADE, расширяет Java своими классами. JACK — одна из немногих платформ, где используются модель логики агентов, основанная на убеждениях-желаниях-намерениях, и встроенные формально-логические средства планирования работы агентов.

• MadKIT — модульная и масштабируемая мультиагентная платформа, написанная на Java. Поддерживает агентов на разных языках: Java, Python, Jess, Scheme, BeanSchell. Красиво визуализирует и позволяет управлять этими агентами.

• AgentBuilder — большой коммерческий продукт, выпускаемый также и в Academic Edition. Агенты достаточно интеллектуаль-

ны, общаются на языке KQML (Knowledge Query and Manipulation Language) и обладают ментальной моделью. Платформа является Java-ориентированной.

• Cougaar (CognitiveAgentArchitecture) — также Java-ориентированная платформа для построения распределенных МАС. Включает не только исполняющую систему (run-time engine), но и некоторые средства для визуализации, управления данными и др.

• NetLogo — кроссплатформенное программируемое окружение для программирования МАС.

• VisualBots — бесплатный мультагентный симулятор в Microsoft Excel с синтаксисом VisualBasic.

• MASON — Java-библиотека для моделирования МАС.

• REPAST — набор инструментов для создания систем, основанных на агентах.

• CogniTAO — C++-платформа разработки автономных МАС, ориентированная на реальных роботов и виртуальных существ (CGF).

Определение языка — основная часть создания агентов. Наиболее используемые исследования МАС, например RETSINA, DECAF (Graham and Decker, 2000), Infosleuth (Nodine et al., 1999), Jade (JADE, 2000) среди прочих, используют KQML (Finin et al., 1997) или FIPA ACL (FIPA, 2000) как язык связи агентов.

Примером применения подхода МАС на практике может служить МАС консолидации грузов в междугородних грузоперевозках, разработанная в научно-производственной компании "Генезис знаний" [3]. Также МАС активно используются для развития многих приложений таких, как управление портфолио ценных бумаг, электронная коммерция, техническое обслуживание воздушных судов и военная логистика:

Система WARREN является одним из приложений, которое занимается сбором информации и управлением портфолио ценных бумаг, состоит из трех типов агентов: агент-интерфейс, демонстрирующий портфолио пользователям, агент-задача, который "помогает" пользователю в управлении его портфолио, и информационный агент, который собирает соответствующую информацию по поводу акций, находящихся в портфолио (например, цены и финансовые отчеты компаний). С помощью агента-интерфейса пользователь может покупать, продавать акции, отслеживать стоимость его портфолио. Два агента-задачи помогают пользователю, Контроллер и Критик риска. Первый записывает состояние портфолио и

может взаимодействовать с брокерами для покупки акций. Второй выступает в роли финансового советника и дает сигнал пользователю, когда приобретение или продажа акций модифицируют риски, связанные с портфолио. Информационный агент мониторит Интернет для оценки акций и их текущих рисков для агента-критика рисков.

Система COALA является приложением, использующимся для электронных аукционов. Система управляет покупкой коллекционных книг, связывая большие группы покупателей в коалицию. Тестовая система состоит из объединенного сервера, агента-аукционера, агентов-поставщиков и набора веб-интерфейсов — по одному для каждого конечного пользователя. Система построена на протоколе предварительных переговоров и различных сделок, которые позволяют поставщикам раскрыть их политику лидеру коалиции покупателей. Лидер коалиции использует веб-интерфейс для внесения изменений в аукцион с помощью агентов-поставщиков. Агенты, в свою очередь, принимают решение по поводу ступенчатой функции для составления некоторого "расписания" оптовых скидок и делают свои ставки, согласно планируемому размеру коалиции. После того, как аукцион закрывается, лидер открывает коалицию для новых членов, которые могут присоединиться к группе, если они соответствуют вступительным требованиям. После того как сформирована группа, сервер выполняет транзакцию.

Система Retsina была применена в области логистики для поддержания процесса принятия решения сразу нескольких пользователей. В одной из таких областей агенты помогли трем пользователям спланировать гипотетическую эвакуацию города Кувейт. Согласно сценарию, военачальник, посол и ответственный за транспортировку должны были найти безопасный выход из города. Они находятся в разных местах и каждый использует агента, который называется мессенджер, для связи с другими. Агенты "подслушивают" переговоры людей для определения нужд пользователя, а также ожидают некоторую информацию, которая поможет в процессе принятия решения. Каждый мессенджер использует MAC для определения агентов, которые отслеживают интересующую информацию, такую, например, как положение спутников, сводки погоды, новостные отчеты и т. д. Роль агентов здесь не ограничена сбором информации. Они также могут помочь пользователям в переговорах, освобождая тех от передачи деталей другим пользователям.

3. Новая адаптивная мультиагентная ОСРВ

МАС — это нечто большее, чем просто набор агентов, собранных в одной системе. Для того чтобы работать вместе, агенты должны иметь способ найти друг друга, общий язык, онтологию, которую они делят между собой, — это необходимо для того, чтобы они могли понять сообщения других агентов. Роль инфраструктуры заключается в обеспечении онтологии, языка и сервисов, которые позволят им объединиться и обмениваться информацией. Результатом является МАС, возникающая как скопление агентов вокруг инфраструктуры, которая является "клеем", держащим агентов вместе.

Исторические тенденции развития операционных систем (ОС) показывают, что обычно ОС вбирает в себя те функции, которые не несут полезной нагрузки, однако составляют ту инфраструктуру, в которой разработчикам удобно решать свои задачи. Так, большинство идей, которые планируется реализовать в новой ОС, уже имеют свои аналоги по отдельности, вне единой ОС.

Несмотря на то, что МАС — крайне популярный и эффективный подход, для каждой конкретной задачи архитектура либо разрабатывается "с нуля", либо диктуется средствами разработки системы (например, разработанная в JADE МАС всегда подразумевает наличие в каждой сети агента, знающего адреса всех агентов данной сети и адреса других сетей, ответственного за связь между остальными агентами). Наличие гибкой обобщенной архитектурной платформы позволит стандартизовать и значительно ускорить процесс разработки МАС за счет сокращения времени архитектурной эксплуатации.

В проекте создания новой мультиагентной ОСРВ планируется разработка обобщенной архитектуры гетерогенной адаптивной мультиагентной системы для решения широкого спектра задач. В качестве тестовой модели планируется реализовать прототип МАС, который будет использован для решения задачи управления группой беспилотных летательных аппаратов (далее БПЛА).

Предлагается использовать следующую модель. Основными метафорами новой операционной системы будут "заказы" и "ресурсы". "Внутренний мир" системы состоит из узлов-агентов. Каждый агент обладает набором методов и используемых ресурсов, которые он может использовать для решения задач, свойствами производительности, стоимости единицы времени и загруженности. Задачи

имеют определенные цены, характеризующие свойства, цели, времена постановки, решения и критического ответа-подтверждения о приемке заказа. При поступлении задачи в систему на одном из ресурсов генерируется агент, отвечающий за прием-отказ в решении задачи и выбор способа ее решения. Для анализа возможных способов решения задачи используются знания, формализованные в распределенной онтологии системы, в которой, кроме описания абстрактных понятий системы, заданы известные правила декомпозиции задач по определенным типам ресурсов с соответствующими затратами. При отсутствии в онтологии необходимых правил решения или при загруженности необходимых ресурсов при наличии времени и свободных ресурсов выбор новых правил решения задачи будет проводиться с помощью рандомизированных алгоритмов, обеспечивая адаптацию к возникающим неопределенностям. Если в ходе такого подхода будут найдены новые способы решения, то они будут добавляться в онтологию, которая разделяет задачи по определенным свойствам на типы агентов и затраты на их решения. Таким образом, агенты системы могут самостоятельно решать поставленные задачи, производить декомпозицию задач, порождать во внутренний мир новые задачи, пополнять онтологию после успешного решения ранее неизвестных задач.

Разработан следующий план по реализации проекта:

1. Разработка общей архитектуры системы (“платформо независимой” части: языка и библиотеки, написанной на том же самом языке.)

1.1 Разработка топологии связей между агентами.

1.2 Разработка модели коммуникации между агентами.

1.3 Разработка модели онтологии системы.

2. Теоретическое обоснование эффективности выбранной архитектуры и разработка симулятора для тестирования системы.

3. Создание прототипа системы с заданной архитектурой для решения задачи управления группой БПЛА.

3.1 Определение протоколов и методов связи между БПЛА и БПЛА-Базовая станция.

3.2 Определение платформы для разработки МАС.

3.3 Разработка системы.

3.4 Апробация системы на коллективных задачах поиска заданных образов, исследования заданной территории и сопровождения до найденного объекта в условиях динамического перераспределе-

ния нагрузок и подзадач.

Для решения поставленной на первом этапе задачи предполагается разработка специализированного предметно-ориентированного языка программирования, предназначенного для описания поведения, форматов и стратегий взаимодействия агентов. При этом будут проведены:

• разработка абстрактного синтаксиса, конкретного синтаксиса и семантики языка;

• разработка стандартной библиотеки с реализацией наиболее часто используемых стратегий, форматов и т. п. с использованием рандомизированных алгоритмов;

• разработка и реализация симулятора MAC с интерпретатором языка для тестирования;

• реализация компилятора для некоторой реальной системы и ее апробация;

• разработка плагина поддержки языка для некоторой интегрированной среды разработки.

Создание удобного языка позволит существенно облегчить разработку MAC.

Предметно-ориентированный язык (англ. Domain-Specific Language, DSL) — язык программирования, специально разработанный для решения определенных задач из заданной предметной области. Использование DSL обладает следующими преимуществами:

• программы, написанные с использованием DSL лаконичны, что позволяет ускорить процесс разработки и улучшает читаемость исходного кода;

• DSL позволяет на уровне абстракции, соответствующей домену, проводить оптимизацию и верификацию;

• так как разработка на DSL ведется в терминах модели, то ПО, написанное на DSL, подвержено меньшему количеству ошибок, чем то же ПО, написанное на уровне кода.

В настоящее время есть работы как по практической части, т. е. конкретные реализации языков программирования, так и по теоретической. Большинство практических работ являются т. н. встроенными DSL (embedded DSL), т. е. они реализованы поверх некоторого языка программирования общего назначения. Например, DSL на языке Scala. Такие языки, во-первых, ограничены своим родительским языком (host language), (в частности, уже упомянутый язык Scala позволяет изменить алгоритм вывода типов и разреше-

ния имен символов путем использования макросов, но не позволяет изменить абстрактный или конкретный синтаксис), во-вторых, привязаны к платформам, на которых реализованы их родительские языки. Теоретические работы — это, в основном, описание некоторых свойств и концепций языка, которые должны упростить разработку МАС. Обычно в таких подходах присутствует как минимум один из следующих двух недостатков: во-первых, у них нет практической реализации, т. е. нельзя оценить, насколько предложенные подходы хороши. Во-вторых, они — не языки программирования, а являются языками графического описания, т.е. в них нельзя использовать уже известные приемы, подходы из других языков (например, было бы полезно иметь в DSL для описания МАС некоторые элементы контрактно-ориентированного программирования, как в языке Eiffel, или схему описания коммуникации через сообщения, как в языке Promela).

Классические подходы, представленные во многих средствах разработки, могут оказаться непригодны для создания обощенной системы: в них часто подразумевается синхронность (в той или иной мере) системы, а также необходимость координирующих узлов, аккумулирующих знания от нескольких агентов. В изначально асинхронных задачах поддержание синхронности сопряжено с дополнительными затратами, которых желательно избежать. Как в задаче достижения консенсуса, хороший результат может быть получен в условиях децентрализации за время, сравнимое со временем сбора данных о нагрузке всей сети [4,5]. Кроме того, при возможности ошибки узлов, децентрализованная модель управления, основанная на независимых агентах, оказывается эффективнее классической иерархической системы [6].

Для представления знаний интересна онтологическая модель из [6]. Разделение знаний агента на несколько онтологий позволяет упростить каждую онтологию, что ускоряет процессы принятия решений и разработки онтологии. Также предполагается различие онтологий различных агентов, что приближает архитектуру системы к структуре предметной области (различные объекты, представляемые агентами, обладают различными наборами знаний), а также позволяет избавиться от хранения и передачи нерелевантной информции (данные передаются только при необходимости). Организация эффективного взаимодействия агентов — одна из основных задач при разработке новой системы. Контуры ее эффективного ре-

шения пока не достаточно четко сформулированы, но мы надеемся на прогресс в ходе реализации проекта. Для более глубокого понимания надо лучше осознать психологические основания проблемы искусственного интеллекта [7].

4. Подсистема новостной авторизации

Один из главных вопросов, который возникает при создании МАС, это безопасность, так как агенты могут неправильно себя вести, "обманывая" других агентов, тем самым влияя на интеграцию системы. На первом этапе выполнения проекта мы ставим своей задачей разностороннее применение концепции новостной исполняющей подсистемы для современных ОС и апробацию всей схемы на примере комплекса приложений-агентов. Основным элементом в МАС является агент. Агент — это не что иное, как программное приложение с заданными параметрами и характеристиками. Агент работает с данными, и легко спрогнозировать, что сообщений о работе программ будет настолько много, что даже в небольшом круге доверия распределенной системы их сохранение для последующей обработки может со временем стать задачей сложной и дорогой для реализации. В этой связи, необходимо сделать обработку сообщений "на лету", т. е. обработку небольших порций сообщений, не допуская таким образом накопления больших архивов сообщений. Значительная часть оригинальных сообщений будет уничтожаться (не будет сохраняться) после обработки. Необходимость в такой организации вытекает, например, из таких распространенных криминальных практик, как подмена программ или их частей с целью добавления функциональности криминального характера. Традиционные методы проверки целостности программ (вычисление контрольных сумм и т. п.) не смогут больше использоваться, поскольку программа, делающая карьеру, формирующая свое "резюме", будет с неизбежностью менять себя или какую-то свою часть. В этой ситуации мы можем обязать программу запоминать не только информацию о себе, но и какую-то часть информации из сообщений других приложений, работающих в том же круге доверия. Проверять эти знания мы можем либо постоянно с какой-то не перегружающей систему периодичностью, либо после периодов продолжительного отсутствия программы в круге доверия, либо в случае резкого изменения характеристик ее работы. Выполня-

емая таким образом аутентификация программы не будет давать 100% уверенности в полученном результате, но зато высвободит колоссальный человеческий и вычислительный ресурс, выполняющий сегодня более точную аутентификацию, и, самое главное, позволит расширить область контроля (пусть и не 100-процентного) до таких величин, которые совершенно недоступны традиционным точным методам в силу заведомого недостатка необходимых ресурсов.

5. Балансировка загрузки сети

Важной задачей, встающей при использовании MAC на практике, является распределение ресурсов между агентами. Если связи между агентами постоянны, передача информации из одной точки сети в другую не является сложной задачей. В ситуации постоянного изменения связей между агентами передача информации из одной точки сети в другую может быть сведена к задаче достижения консенсуса или согласования характеристик. Каждый агент в ходе достижения консенсуса стремится уменьшить отклонение своей целевой переменной от соответствующих переменных своих соседей. Если речь идет об отклонении от среднего арифметического состояний соседей, то ставится задача достижения усредненного консенсуса. Для распределенных систем параллельных вычислений актуальна задача разделения пакета заданий между несколькими вычислительными устройствами, которая сводится к задаче балансировки загрузки сетей, т. к. обычно при распределении заданий по узлам их стараются распределять так, чтобы загрузка вычислительных узлов была равномерной. Подобные задачи возникают не только в вычислительных сетях, но также и в производственных сетях, сетях обслуживания, транспортных и логистических сетях, и др. На практике используются как централизованные стратегии динамической балансировки загрузки, так и полностью распределенные (децентрализованные). При централизованной стратегии выделяется специальный ресурс, который собирает информацию о состоянии всей сети и принимает решения о распределении заданий для каждого из узлов. При полностью распределенной стратегии мультиагентные алгоритмы балансировки загрузки выполняются на всех узлах, обменивающихся информацией о состояниях с другими узлами сети. Перемещение происходит только между соседними (связанными) узлами [4, 5].

6. Заключение. МАС управления группой БПЛА

Перспективным направлением применения МАС являются группы беспилотных летательных аппаратов. В современном мире беспилотные летательные аппараты (англ. Unmanned Aerial Vehicles, далее БПЛА) приобретают все большую популярность в качестве легких и недорогих инструментов для исследований территорий, разведки и воздушных съемок. В основной массе современная беспилотная техника не обладает автономностью (обычно дистанционно управляется оператором со специального пульта). Есть успешные разработки одиночных БПЛА, которые под управлением автопилота способны в автономном режиме облететь территорию по заданному маршруту, собирая ту или иную информацию или выполняя другие задания. Анализ большинства задач, решаемых с использованием одиночных аппаратов, показывает, что они могут более эффективно решаться группой, так как у группы эффективно взаимодействующих легких и маленьких самолетов появляются дополнительные полезные свойства [8].

Для крупных беспилотных летательных аппаратов в ближайшее время также станут актуальными проблемы группового взаимодействия, при этом отработку возможных сценариев этого взаимодействия удобнее и дешевле провести на малых и легких самолетах. Мы планируем реализацию прототипа новой ОС на вычислительных блоках, которые представляют собой небольшие микрокомпьютеры, установленные на бортах легких летательных аппаратов. Групповой полет таких летательных аппаратов позволит получать дополнительные возможности в полете, автономное и адаптивное принятие решений для достижения общей цели. Сейчас управление БПЛА обычно организуется по двухуровневой системе управления. На нижнем уровне находятся исполнительные механизмы и автопилот. Основные задачи автопилота — поддерживать заданные параметры движения по заданному маршруту. Маршрут задается автопилоту в виде последовательности координат точек, которых необходимо достичь в порядке их записи. Формирование текстового файла с координатами и сбор данных происходит на верхнем уровне. Мы добавляем средний уровень управления, функции которого выполняет дополнительный микрокомпьютер. Его задачей является частичное выполнение функций верхнего уровня — запись

новой последовательности координат следования в зависимости от новой поступающей информации из внешнего мира (от датчиков, других микрокомпьютеров, базовой станции).

Список литературы

[1] *Виттих В.А., Скобелев П.О.* Метод сопряженных взаимодействий для управления распределением ресурсов в реальном масштабе времени // Автометрия. 2009. №. 2. С. 78–87.

[2] *Амелина Н.О.* Мультиагентные технологии, адаптация, самоорганизация, достижение консенсуса // Стохастическая оптимизация в информатике. Т. 7. С. 149–185.

[3] *Амелина Н.О., Лада А.Н., Майоров И.В., Скобелев П.О., Царев А.В.* Исследование моделей организации грузовых перевозок с применением мультиагентной системы для адаптивного планирования мобильных ресурсов в реальном времени // Проблемы управления. 2011. № 6. С. 31–37.

[4] *Амелин К.С., Амелина Н.О., Граничин О.Н., Корявко А.В.* Применение алгоритма локального голосования для достижения консенсуса в децентрализованной сети интеллектуальных агентов // Нейокомпьютеры: разработка и применение. 2012. №11. С. 39–47.

[5] *Амелина Н.О., Фрадков А.Л.* Приближенный консенсус в стохастической динамической сети с неполной информацией и задержками в измерениях // Автоматика и Телемеханика. 2012. №11. С. 6–30.

[6] *Виттих В.А.* Введение в теорию интерсубъективного управления. — Самара. Самарский научный центр РАН, 2013. — 64 с.

[7] *Сергеев С.Ф.* Психологические основания проблемы искусственного интеллекта // Мехатроника, автоматизация, управление. 2011. №7. С. 2–6.

[8] *Амелин К.С., Антал Е.И., Васильев В.И., Амелина Н.О.* Адаптивное управление автономной группой беспилотных летательных аппаратов // Стохастическая оптимизация в информатике. Т. 5. 2009. С. 157–166.