УДК 004.415:04.423; 004.453 DOI 10.18522/2311-3103-2021-1-125-137

И.О. Шальнев

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К ОПИСАНИЮ ВЗАИМОДЕЙСТВИЯ ГРУППЫ РОБОТОТЕХНИЧЕСКИХ СРЕДСТВ НА ОСНОВЕ РАСПРЕДЕЛЕННОЙ ВИРТУАЛЬНОЙ МАШИНЫ*

Проведен анализ подходов к построению робототехнических систем. Показано, что робототехнические системы можно рассматривать как распределенную систему взаимодействия отдельных компонентов робототехнической системы и взаимодействия робототехнических систем в рамках единого комплекса. В первом случае робототехническая система является совокупностью отдельных модулей в рамках одного робота. Так, например, отдельные моторы, сервоприводы для управления полезной нагрузкой беспилотного летательного аппарата (БЛА) можно рассматривать как отдельные модули всего БЛА в рамках распределенной робототехнической системы. Во втором случае робототехнической системой может считаться совокупность взаимодействующих робототехнических систем. Так, например, отдельно взятый БЛА является робототехнической системой в составе общей распределенной робототехнической системы, определяющей групповое взаимодействие. Необходим подход позволяющий единый способ описания такой иерархии робототехнических систем. В области робототехники существует множество подходов построения таких систем, каждый из которых определяет средства связи и передачи данных. Данная статья описывает существующие подходы, их достоинства и недостатки, а также предлагает иной подход для создания распределенных робототехнических систем. Связь отдельных узлов информационной сети, в существующих подходах, обеспечивается путем передачи данных с последующей их обработкой. В статье описывается подход, основанный на инкапсуляции исполнимого кода в передаваемые сетевые пакеты. Взаимодействие объектов осуществляется посредством передачи управляющей информации, интерпретируемой распределенной виртуальной машиной. Расширение парадигмы объектно-ориентированного программирования (ООП) понятием комплементарного объекта, позволяет создавать распределенную систему, абстрагируясь от особенностей сетевого программирования. Объектно-ориентированный подход, основанный на использовании комплементарных объектов, позволяет разрабатывать распределенную систему как единую программу, концентрируясь на реализации логики. В таком случае мы переходим от концепции распределенной системы как реализации отдельных модулей к концепции единой распределенной программы без «синтаксического разрыва». В статье предлагается подход, позволяющий представлять распределенную робототехническую систему в парадигме (ООП), как совокупность взаимодействующих через коммуникационную среду объектов, обеспечивающий передачу данных через аргументы удаленных вызываемых методов.

Робототехнические системы; распределённые системы; языковые виртуальные машины.

I.O. Shalnev

COMMUNICATION AND DATA TRANSFER IN DISTRIBUTED ROBOTICS SYSTEMS

The robotic systems building approaches is analyzed in this paper. Robotics systems can be considered as a distributed system which interacted between individual components of the robotic system and interacted between robotic systems within a single complex. In first case a robotic system can be a collection of individual modules within a single robot. For example, individual motors, servo drives for camera control, and the camera itself control of the unmanned aerial vehicle (UAV) can be considered as separate modules of the entire UAV within a distributed robotic system. In second case robotic system devoted to a massive of connected robotic system. For

* Работа выполнена в рамках реализации Государственного задания на 2021 г. № 0073-2019-0005.

example, each UAV is a robotic system which is a part of whole robotic system that define UAV group interaction. An approach that allows a unified way to describe such robotic systems hierarchy is needed. In the field of robotics systems, there are many approaches to building it, which defines the means of communication and data transfer. This paper describes the existing approaches, their advantages and disadvantages, and suggests another approach for creating distributed robotic systems. Each nodes network connection in the existing approaches is provided by transferring data with their subsequent processing The article describes an approach based on the encapsulation of executable code in transmitted network packets. The object interaction is carried out through the control data transfer that interpreted by a distributed virtual machine. The object-oriented programming (OOP) paradigm extension by the concept of a complementary object, allows to create a distributed system that abstracted from the network programming difficulty. An object-oriented approach based on complementary object usage allows to develop a distributed system as a single program, concentrating on the implementation of logic. So we passing from the distributed system as an implementation of separate modules concept to the single distributed program concept without a "syntax breaking".The proposed approach allows to represent a distributed robotic system in the OOP paradigm as a set of objects interacting over a communication network.

Robotic systems; distributed systems; virtual machines.

Введение. В настоящее время активно развиваются технологии «интернета вещей» и робототехнические системы. Оба эти направления объединяет необходимость взаимодействия в инфокоммуникационной среде независимых агентов, что обуславливает появление модифицированных протоколов связи, соответствующих поставленным задачам. Так, разработчики робототехнических систем предлагают свои подходы к построению взаимодействующих элементов, определяют свою логику передачи данных и протоколов взаимодействия подсистем. Очевидно, что эти подходы отвечают требованиям, выдвинутым в рамках определённого круга поставленных задач.

В работе делается попытка разработки методологии представления целостной структуры системы как совокупности взаимодействующих атомарных информационных объектов. Такими объектами могут быть и логические программные объекты, и цифровые двойники объектов реального мира.

Особенностью работы является повышение уровня абстракции программного кода, что избавляет программиста от проблем реализации коммуникационных функций и синхронизации физических объектов.

В статье рассматриваются достоинства и недостатки существующих решений и предлагается способ построения программно-аппаратных комплексов на основе объектно-ориентированного взаимодействия робототехнических средств в распределенной среде. Проблемы, изучаемые в текущем исследовании, привели к необходимости заново осмыслить протокол взаимодействия объектов виртуальной среды.

Существующие протоколы взаимодействия. В настоящий момент наиболее стандартизованным протоколом для обмена информацией между наземной станцией и любыми малыми беспилотными аппаратами является MAVLink (Micro Air Vehicle Link) [1] – это легковесный протокол передачи сообщений для обеспечения взаимосвязи беспилотных летательных аппаратов – БЛА (в том числе и для бортовых компонентов БЛА). Протокол MAVLink придерживается стандартных шаблонов проектирования [2]: издатель-подписчик (publish-subscribe) и точка-точка (point-to-point).

Издатель-подписчик – это шаблон, обеспечивающий обмен сообщениями между издателем (publisher) и подписчиком (subscriber). Издатель публикует сообщения в канал, на который подписчики могут подписаться. На рис. 1 показан пример такого взаимодействия. Модуль ввода публикует сообщения в канал предварительной обработки. Модуль обработки подписывается на этот канал, чтобы получать все публикуемые модулем ввода сообщения. Модуль обработки, в даль-

нейшем, публикует результат обработки в канал пост-обработки. И, наконец, модули вывода подписываются на последний канал, для получения публикуемых в канал постобработки сообщений.

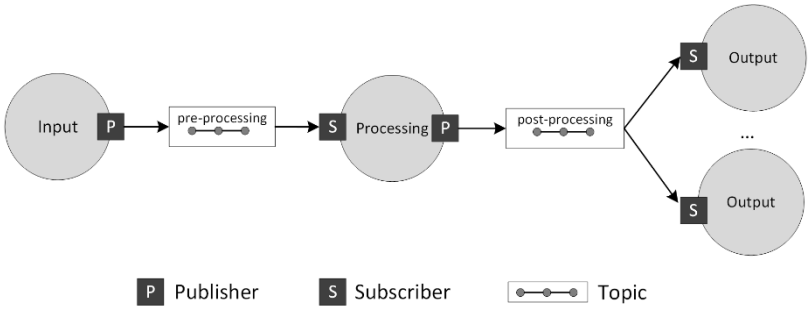

Рис. 1. Пример передачи сообщений в шаблоне «издатель-подписчик»

Основное преимущество шаблона «издатель-подписчик» заключается в том, что он позволяет отправлять сообщения между компонентами системы, которые не знают друг о друге. Такой подход позволяет создавать модули системы изолированно друг от друга, концентрируясь на реализации модуля и не задаваясь вопросом о маршрутизации сообщений. В таком подходе не источник определяет потребителей информации, а потребители подписываются на канал потока данных [2–4].

В основе шаблона «издатель-подписчик» лежат другие шаблоны проектирования, например, шаблон «очередь обработки сообщений», который реализует обмен сообщениями. Обмен сообщениями через очередь часто используется для безопасного межпроцессного и межпоточного взаимодействия, при этом очередь событий должна присутствовать у каждого участника взаимодействия. Во время публикации сообщения оно сначала отправляется в локальную очередь, а затем передается в очередь получателя сообщения [4–5].

В шаблоне «издатель-подписчик» подписчики получают подмножество всего множества публикуемых сообщений. Процесс отбора сообщений называют фильтрацией. Существуют две основные формы фильтрации: основанная на теме (топике) и основанная на содержимом.

В системе, основанной на содержимом, сообщения отправляются в специализированные логические каналы, именуемые топиками (тема канала). Все подписчики такого канала получают все сообщения, публикуемые в топик издателями.

Фильтрация сообщений, основанная на содержимом, предполагает, что сообщения маркируются атрибутами, которые определяют допустимость чтения сообщений теми или иными подписчиками. Ответственность за классификацию сообщений может брать на себя посредник (брокер) или же сам подписчик. В последнем случае осуществляется симуляция широковещательной передачи данных.

Точка-точка – это шаблон взаимодействия, позволяющий передавать сообщения непосредственно между отправителем и получателем, который обеспечивает получение сообщений только одним приёмником (получателем). Если множество получателей пытается обработать одно сообщение, то реализация канала гарантирует, что только один из них сможет это сделать. Таким образом, у получателей нет необходимости во взаимодействии друг с другом (рис. 2). Хотя канал управляет совокупностью подписчиков, только один получатель способен принять уникальное сообщение [6].

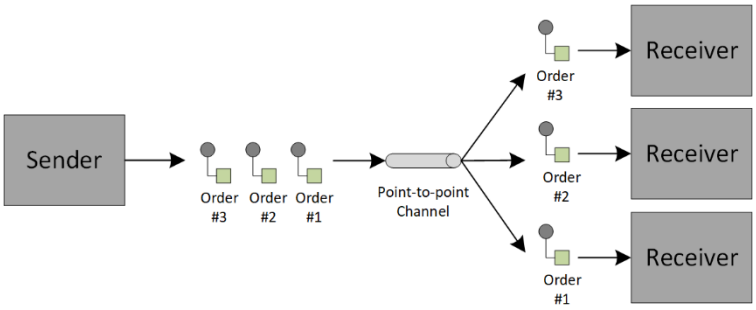

Рис. 2. Пример передачи сообщений в архитектуре точка-точка

В MAVLink потоки данных отправляются (публикуются) в определенные топики без гарантии доставки, однако существуют запросы, требующие гарантированной доставки (например, изменение конфигурации), что требует применения point-to-point канала с ретрансляцией. Структура сообщений задана в формате XML [7], где определяется набор сообщений (диалект), который поддерживается уникальной MAVLink системой. MAVLink использует XML-определение сообщений для генерации фрагментов кода, которые описывают формат обмена бинарными пакетами данных, выраженного на поддерживаемых языках программирования. Данные передаются в бинарном формате в виде последовательности сетевых пакетов, структура которого показана на рис. 3.

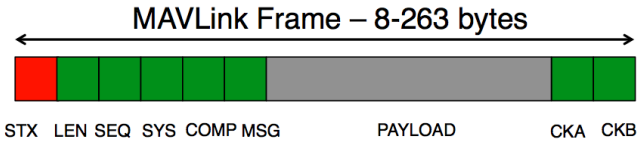

Рис. 3. Формат пакета MAVLink

Первый байт сетевого пакета (STX) – это символ начала сообщения. LEN – длинна полезной нагрузки (сообщения). SEQ – содержит счётчик сетевого пакета (0–255), который выявляет потерю сообщения. SYS (System ID) – идентификатор, отправляющий системы, а COMP (Component ID) – идентификатор отправляющего компонента. MSG (Message ID) – тип сообщения, от него зависит, какие данные будут лежать в полезной нагрузке сетевого пакета. PAYLOAD – полезная нагрузка пакета, сообщение, размером от 0 до 255 байт. Два последних байта пакета – CKA и CKB содержат контрольную сумму сетевого пакета.

В библиотеке MAVLink кодирование и раскодирование сетевых пакетов производится согласно протоколу, но не регламентируется, какими аппаратными и программными средствами данные будут отправлены. Входные данные обрабатываются побайтово. Каждая система или компонент может одновременно обмениваться данными по разным источникам, тогда как для каждого источника назначается специальный идентификатор канала.

В области робототехники существует стандартизованное средство разработки Robot Operation System (ROS) [8]. Это набор инструментов для создания и взаимодействия узлов, а также методология разработки робототехнических систем. В качестве средства связи и передачи данных в ROS доступны следующие варианты: издатель-подписчик, сервис-ориентированный подход и взаимодействие через сервер параметров.

«Издатель-подписчик», описанный выше, используется как основной способ асинхронного взаимодействия узлов. Сервис-ориентированный подход предполагает взаимодействие в виде запрос-ответ и является синхронным. Сервер параметров — это разделяемый между узлами словарь переменных, который доступен через коммуникационную сеть. Обычно используется для хранения редко изменяемых данных, таких как конфигурационные параметры.

Глобальное использование сервис-ориентированного подхода приведет к уменьшению производительности отдельного узла из-за простоев процессора во время ожидания синхронного вызова. Сервис параметров также отличается низкой производительностью и рекомендован к использованию для конфигурации системы, а не для передачи данных в процессе обычной работы системы. Исходя из этого основным средством передачи данных в ROS является шаблон «издатель-подписчик».

Объектно-ориентированная парадигма описания распределенных систем. Описанные выше протоколы передачи данных реализуют взаимодействие модулей, связанных по сети через передачу и обработку полученных сообщений. Некоторые архитектурные решения упрощают разработку таких модулей, но не исключают наличия «синтаксического разрыва» программы, который устраняется протоколом передачи данных. В случае, когда модули являются частями программы, работающей на одном вычислительном устройстве, то данные могут передаваться через память. Стандартными средствами языков программирования для подобного взаимодействия является передача данных в процедуру, функцию или объект. Одним из возможных способов осуществления подобного взаимодействия в рамках сетевого программирования может стать вызов функций или процедур в адресном пространстве иного процесса. Такой класс технологий называют удаленным вызовом процедур (RPC от английского Remote Procedure Call) [9].

В языках объектно-ориентированного программирования (ООП) [10] объектная декомпозиция задачи является описанием предметной области. Если же отдельные классы предполагают исполнение на разных узлах распределенной системы, то в классическом случае это приводит к разрыву объектной декомпозиции. Иной подход предполагает наличие подмножества удалённых (remote) объектов, обладающих возможностью удаленного вызова методов (RMI от английского remote method invocation). При этом не нарушается декомпозиция объектов (рис. 4).

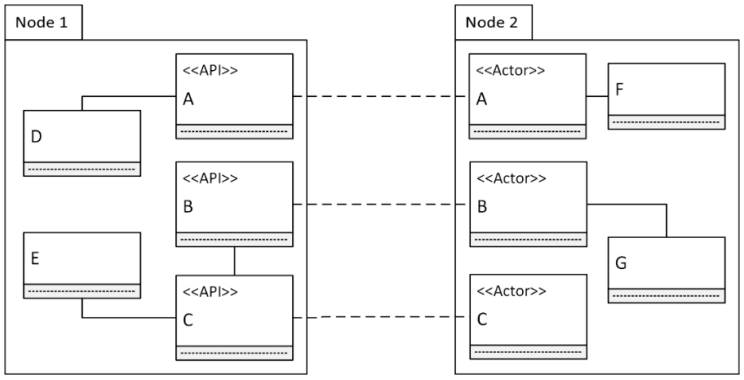

Рис. 4. Пример работы комплементарных объектов

Такие удалённые объекты будем называть «комплементарными», потому что состоят из двух частей, расположенных на разных узлах распределенной сети. Первая часть является программным интерфейсом (API) и отвечает за жизненный

цикл всего комплементарного объекта (его создание и удаление), а также, за вызов удаленных методов. Вторая – функциональная часть (Actor) – реализация объекта, отвечающая за исполнение вызванного на стороне программного интерфейса функционала [11]. Взаимодействие этих объектов обеспечивается путем передачи управляющей информации в виде последовательности команд псевдоассемблерного кода (байт-код). Управляющие данные интерпретируются языковой виртуальной машиной [12], находящейся на каждом узле распределенной системы. В качестве примера реализации может быть использована, в частности, технология «активных данных», основанная на инкапсуляции исполнимого кода в передаваемые сетевые пакеты [13].

Несмотря на некоторые особенности функционирования комплементарных объектов [14] данный подход позволяет использовать их как обычные объекты, скрывая от пользователя факт нахождения объекта в ином адресном пространстве удалённого узла. Это позволяет разрабатывать распределенную систему как единую программу, концентрируясь на реализации логики. В таком случае мы переходим от концепции распределенной системы как реализации отдельных модулей к концепции единой распределенной программы без «синтаксического разрыва».

Распределенная виртуальная машина (РВМ). Программа в объектно-ориентированной методологии представляет собой представление совокупности связанных между собой объектов. Когда мы разрабатываем распределенную систему как единую сетевую программу, то её объекты находятся на разных узлах сети. Обеспечение их взаимосвязи реализуется транспортным слоем сетевого взаимодействия и функциями виртуальной машины.

Уровень виртуальной машины занимается сбором управляющей информации и её интерпретацией. Он состоит из собственно виртуальной машины и слоя интерпретации-ассемблера байт-кода комплементарных объектов. Виртуальная машина хранит комплементарные объекты в таблице объектов и обеспечивает имплементацию команд вставки, удаления и извлечения комплементарных объектов в таблице объектов. Также виртуальная машина интерпретирует набор команд для работы с аргументами, переданными в метод или конструктор объекта. Слой интерпретации-ассемблера – это пара, обеспечивающая взаимодействие комплементарного объекта в сети. Ассемблер собирает байт-код, а интерпретатор исполняет байт-код, собранный на противоположной стороне. Интерпретатор и ассемблер находятся как на стороне программного интерфейса, так и на стороне функционального объекта (Actor).

Рассмотрим алгоритм вызова метода удаленного объекта:

1. Ассемблер программного интерфейса формирует байт-код, помещая в информационный пакет данные о идентификаторе объекта, идентификаторе метода объекта и упаковывает параметры в сообщение (parameter marshaling).

2. Виртуальная машина на стороне API определяет наличие сформированного пакета и отравляет его по нужному каналу связи.

3. Принятый на удалённой стороне информационный пакет обрабатывает виртуальная машина, которая по идентификатору объекта извлекает созданный ранее объект из таблицы и передает необходимую информацию интерпретатору функционального объекта.

4. Интерпретатор функционального объекта по идентификатору метода определяет вызываемый метод и после извлечения аргументов метода (demarshaling) вызывает метод функционального объекта.

Алгоритм отправки и обработки результата вызванного удалённого метода осуществляется по схожей схеме:

1. Функциональный объект формирует байт-код с информацией об идентификаторе объекта и методе, помещая результат в пакет данных.

2. После определения наличия сформированного пакета виртуальная машина отправляет байт-код по каналу связи.

3. После приема данных на стороне API виртуальная машина извлекает объект из таблицы объектов и передает информацию о методе и его результате в интерпретатор объекта.

4. По идентификатору метода при помощи интерпретатора объекта API определяется метод, по которому пришел результат.

5. Вызывается функция обратного вызова с результатом в качестве аргумента.

Здесь важно отметить, что функция обратного вызова определяется в момент вызова удаленного метода. Связь результата с функцией обратного вызова осуществляется при помощи событийно-ориентированной архитектуры. Событие, каковым является получение результата удалённого метода, помещается в очередь обработки событий и будет обработано функцией обратного вызова в порядке очереди. Преимуществом такой обработки результата по сравнению с аналогичными [15, 16] является то, что она асинхронна и не имеет блокирующих функций.

Транспортный слой. В задачу транспортного слоя входит передача байт-кода между узлами. PBM не ограничивает разработчика в выборе протокола передачи данных, хотя накладывает ряд требований к нему. Во-первых, протокол передачи должен гарантировать последовательность приёма отправленных данных. Если байт-код придёт не в той последовательности, то поведение системы будет неопределенно и ошибочно. Во-вторых, он должен обеспечивать целостность данных, то есть данные не должны быть изменены в процессе передачи. В-третьих, протокол передачи данных должен уметь обнаружить разрывы соединения узлов в сети.

Также транспортный слой обеспечивает маршрутизацию передаваемого байт-кода. На рис. 5 показан пример объектной декомпозиции, объекты которой находятся на трёх разных узлах. Первый и второй узлы связаны с третьим, но прямое соединение между первым и вторым отсутствует. Классы на узлах 2 и 3 являются комплементарными и удаленно управляются пользовательским классом 3. Вызов метода «doSmth» влечёт за собой вызов удалённого метода «foo» класса 2, в аргументы которого передаётся API объект класса 1. В таком случае, сформированный на стороне узла 3 байт-код будет передан узлу 2, который инициирует вызов метода «foo». Виртуальная машина узла 2 – API объект класса 1 (аргумент метода «foo») создается виртуальной машиной этого узла. При этом возникает проблема: каким образом в отсутствии прямого соединения между узлами 1 и 2 обеспечить удалённый вызов метода «bar»? Эта проблема может решаться двумя способами.

Первый способ предполагает передачу байт-кода через узел 3, имеющий соединение с узлом 1. В этом случае узел 3 выступает как прокси между узлом 1 и 2. Второе решение заключается в создании прямого соединения между узлами 1 и 2.

И первое, и второе решение реализуется при помощи передачи управляющей информации (байт-кода), которая инициирует создание API объекта первого или второго типа и определяет необходимость создания нового соединения. Иными словами, PBM подготавливает узел 2 в соответствии с настройками, указанными пользователем. В рамках примера, узел 3 формирует байт-код, который подготавливает виртуальную машину узла 2 и создает такой объект API класса, который во время удаленного вызова знает, как отправлять данные. Такой процесс называют связыванием объектов (binding).

Описанное взаимодействие порождает такую ситуацию, при которой с одним функциональным объектом связаны сразу два объекта программного интерфейса. Такой функциональный объект называется разделяемым и не удалится до тех пор,

пока существует хотя бы одна привязка. При удалении API объекта на узле 3 функциональный объект класса 1 не удалится, так как с ним связан API, находящийся на узле 2.

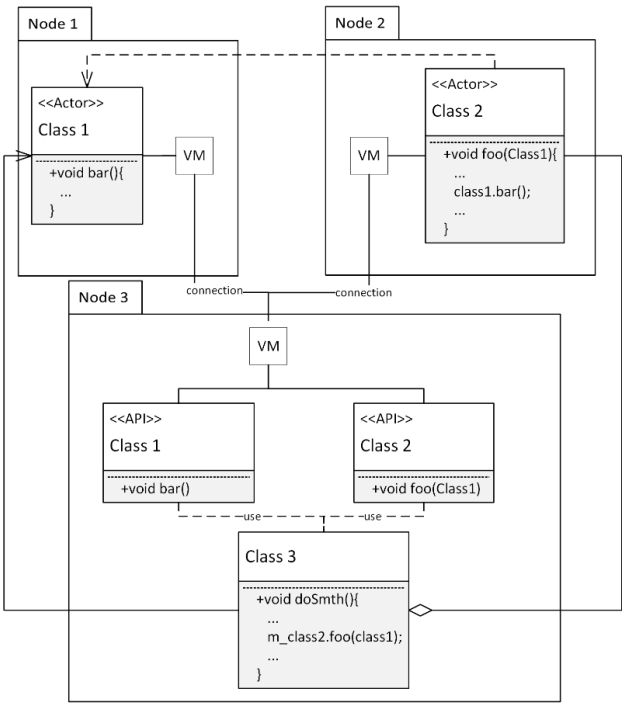

Рис. 5. Объектная декомпозиция, распределённая на трёх разных узлах

Модульность. Для обеспечения принципа модульности можно использовать систему взаимодействующих программных пакетов (по аналогии с ROS [8]). Все модули, которые предполагают работу на отдельном узле должны реализовываться внутри отдельного программного пакета, определяя необходимую логику посредством определения как локальных (обычных), так и комплементарных объектов. Любой другой пакет может использовать сгенерированные системой интерфейсные объекты этого пакета для управления удаленным объектом. Объекты можно представлять как библиотеки, дающие возможность удаленного взаимодействия с ними.

Протокол взаимодействия распределенных объектов обладает гибкостью с точки зрения повторного использования кода. Так, например, код для работы с одними и теми же датчиками, сервоприводами, моторами может быть использованы в различных продуктах. Например, датчик давления может быть использован на производственном оборудовании, в быту, в системах интернета вещей и прочих автоматизированных системах. Специфика использования такого датчика в каждой из сфер порождает проблему однотипного кода, который отличается деталями, но не общими принципами. Эта проблема может быть частично решена путем реализации специфических особенностей на удаленном по отношению к датчику узлу. Код управления для самого датчика инкапсулируется в объект или объекты, удалённое использование которых обеспечивает возможность дистанционного программирования или реконфигурирования (перепрограммирования), если это необходимо для задачи. Таким образом, вместе с подключением самого датчика к

контроллеру устанавливается объект или набор объектов, обеспечивающий дистанционное управление этим датчиком. Такой подход позволяет решать задачи программирования в парадигме ООП используя модели объектов реального мира.

Реализация управляющего протокола поверх MAVLink. Вполне естественно, что технология РВМ может работать поверх стека существующих сетевых протоколов. Например, она может работать на прикладном уровне модели OSI или модели DOD [17, 18]. Предположим, что удаленное объектное взаимодействие можно реализовать поверх протокола MAVLink. Управляющие данные, относящиеся к объектному протоколу, должны помещаться в секцию полезной нагрузки пакетов этого протокола и передаваться через каналы шаблона издатель-подписчик. Каждый узел подписывается на канал Input для приема управляющей информации, а для отправки результата или события возникающего на узле публикует пакеты в канал Output. В случае, когда множество узлов вызывают удаленный метод, то при публикации результата в канал Output возникает неопределенность адресата этого результата. В такой модели каждый подписчик получит результат, но не имеет возможности определить его адресата (рис. 6). В этом случае приходится искусственно вводить избыточную информацию об адресате как при вызове, так и при возврате результата удаленного метода, что не требуется при point-to-point соединении. В другом варианте предполагается создание топиков Input и Output для каждой пары узлов, что является организацией point-to-point соединения и нецелесообразно, так как шаблон издатель-подписчик создавался как альтернатива шаблону точка-точка.

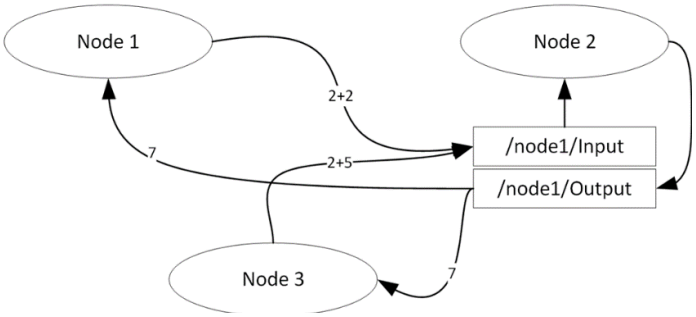

Рис. 6. Проблема определения адресата для взаимодействия комплементарных объектов

Протокол объектного взаимодействия может быть реализован поверх более простых протоколов, имея возможность определять структуру сетевого взаимодействия. К сожалению, какая-то одна архитектура не может являться универсальным решением для всего множества задач. Взаимодействие сервера с множеством клиентов предполагает наличие пула соединений, где виртуальная машина каждого соединения должна быть изолирована от другой для устранения возможных конфликтов или попыток взлома. Такой сценарий обычно используется для реализации однотипных терминалов [19], отличающихся наполнением, но не сутью. Другое взаимодействие может предполагать наличие одной виртуальной машины на несколько соединений. Подобное взаимодействие нарушает изолированность, но позволяет использовать комплементарные объекты в едином пространстве независимо от того, где именно эти объекты были созданы.

Использование РВМ в робототехнике. В мире современных встраиваемых и робототехнических систем можно отметить общую проблему нехватки вычислительной мощности, ограниченной возможностями текущего уровня развития эле-

ментной базы. Даже для реализации логики согласованного взаимодействия подсистем робота для решения задач движения требуются значительные вычислительные ресурсы. Чем сложнее требования к движениям робота, тем сложнее алгоритмы и структура согласования двигательных подсистем. Кроме того, дополнительные вычислительные мощности необходимы для задачи анализа сцен и принятия решений для обеспечения автономного поведения, решения целевой задачи робота. Очевидно, что система с одним (даже очень мощным в текущий момент времени) вычислительным модулем не сможет обеспечить необходимый уровень вычислительной мощности. Таким образом, понимая ограничение текущего уровня технологического развития, необходимо поставить вопрос о решении задач создания робототехнических систем как совокупности распределенных, атомарных взаимодействующих подсистем, использующих отдельные вычислительные ядра. Таким образом, разделив логику отдельных подсистем и обеспечив их структурное и многоуровневое взаимодействие, можно преодолеть существующие ограничения элементной базы.

В такой архитектуре отдельный робот может быть представлен совокупностью отдельных подсистем, взаимодействующих по сети передачи данных. Отдельные подсистемы будут структурироваться в отдельные иерархические контуры, состоящие из атомарных групп вычислительных узлов, контроллеров и датчиков.

Сложность поведения таких систем требует другой логики описания алгоритмов поведения и отдельных подсистем, и отдельных контуров, и всего организма как единое целое.

Решить вопросы согласованного программирования разных подсистем и взаимодействующих контуров можно с использованием технологий и подходов PBM.

Описанные проблемы ограниченности вычислительных возможностей куда более глубоки даже чем показано выше. Следует признать, что на практике даже решить задачи движения, которые должны быть реализованы «на борту» робота часто невозможно с достаточным уровнем сложности, что делает безальтернативной необходимость вынесения центров принятия решений на внешние вычислительные платформы, расположенные в стационарных центрах, не ограниченных энергоресурсами, процессорными объёмами и объёмами памяти.

Заключение. В работе предложен комплекс решений для взаимодействия робототехнических средств в распределенной системе. Впервые предложено использование инфраструктуры распределенной виртуальной машины для организации сетевого взаимодействия между территориально-распределенными физическими устройствами. В отличие от построения коммуникационого пространства с использованием взаимодействий типа «точка-точка» разработчик получает инструмент с новым уровнем абстракции, повышающий гибкость управления устройствами путем передачи управляющего кода.

Рассмотрено применение распределенной виртуальной машины и методология использования комплементарных объектов в качестве платформы взаимодействия группы робототехнических средств. Предложено расширение протокола MAVLink для устранения его основных недостатков.

Управление БЛА, основанное на протоколе MAVLink осуществляется через непрерывную передачу информации. Так в MAVLink для достижения БЛА заданной координаты и поддержания его в этой точке необходимо с определенной частотой отправлять пакет данных с этой координатой, а для определения достижимости БЛА заданной точки постоянно получать текущее местоположение и сравнивать его с заданным значением. В связи с этим в канале связи регулярно происходит передача одного и того же пакета. Такой подход сложно назвать рациональ-

ным с точки зрения экономии ресурса канала связи, хотя он может являться полезным для своевременного обнаружения разрыва соединения, если бы MAVLink не обязывал разработчиков реализовывать в своих системах передачу heartbeat сообщений, которые являются системными и разработаны для этой цели. Намного рациональней отправить координату единожды и асинхронно получить информацию о ее достижении БЛА.

Использование шаблона издатель-подписчик предполагает отправку данных вне зависимости от того нужны ли ему эти данные в текущий момент или нет. Из-за этого возникает проблема перерасхода не только ресурса канала связи, но и ресурса источника питания. Также постоянная публикация телеметрии в канал связи увеличивает загруженность внешнего канала связи или повышает риск обнаружения БЛА.

Групповое взаимодействие БЛА можно рассматривать как множество взаимодействующих самостоятельных программных модулей [20], следуя внутренней логике, но в совокупности подчиняющихся единым алгоритмам, определяющим структуру поведения всей системы как единого целого. В подходах к разработке распределенных систем существует тенденция разбиения большой системы на изолированные подсистемы. Такая тенденция наблюдается и в подходах MAVLink и ROS. Но возрастающее число взаимодействующих элементов, образующих единое логическое и поведенческое целое, неминуемо ведет к росту сложности управления и программирования. В данной ситуации становится сложно рассматривать программирование множества отдельных подсистем изолированно. Встаёт вопрос о единой точке структурной и программной разработки распределенной сетевой системы как совокупности взаимодействующий модулей.

Таким способом программирования может стать подход, рассматривающий многоуровневую иерархию взаимодействующих подсистем как единый, доступный для разработчика набор объектов, независимо существующих на множестве различных автономных устройств в рамках виртуальных языковых машин, взаимодействующих по сети. Такой набор объектов позволит разработчику представлять логику сколь угодно сложной распределенной системы как единое целое, описывая поведение отдельных подсистем как единый программно-аппаратный комплекс, абстрагируя программистов высокого уровня от низкоуровневых процессов и давая возможность оперировать абстракциями требуемого уровня в традиционной объектно-ориентированной парадигме.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Dronecode foundation. – Свободный режим доступа: https://mavlink.io/en/ (дата обращения: 17.01.2021).
2. *Hohpe, G. and Woolf, B.* Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. – Pearson Education, 2012. – P. 106. – ISBN 9780133065107.
3. *Matthew O'Riordan* Everything You Need To Know About Publish/Subscribe. – Свободный режим доступа: https://www.ably.io/topic/pub-sub (дата обращения: 09.02.2021).
4. *Robert Nystrom* Game Programming Patterns. – Genever Benning. – November 2014. – P. 354.
5. *Stevens R.W.* UNIX Network Programming – Interprocess Communication // Englewood Cliffs. Prentice Hall. – 2nd ed. – 1999, August. – 400 p.
6. *Pethuru Raj, Anupama Raman, Harihara Subramanian*. Architectural Patterns // Packt. – 2017, December. – P. 468.
7. Extensible Markup Language (XML). – Свободный режим доступа: https://www.w3.org/XML/ https://www.ably.io/topic/pub-sub (дата обращения: 10.02.2021).
8. ROS Documentation. – Свободный режим доступа: http://wiki.ros.org/.
9. *Таненбаум Э., ван Стеен М.* Распределённые системы (принципы и парадигмы). – СПб.: Питер, 2003. – 880 с.

10. *Буч Г., Максимчук Р.А., Энгл М.У., Янг Б.Дж., Коналлен Д., Хьюстон К.А.* Объектно-ориентированный анализ и проектирование с примерами приложений. – 3-е изд. – М.: Вильямс, 2010. – 720 с.
11. *Шальнев И.О.* Подход к построению распределенной виртуальной машины на основе объектно-ориентированного программирования // Известия Тульского государственного университета. – 2020. – № 9. – С. 40-47.
12. *Wei Chen, Weixia Wu, Zhiying Wang, Qiang Zhao* A Formalization of An Emulation based Co-Designed Virtual Machine // Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. – 2011. – P. 164-168.
13. *Alexandrov V.V., Kuleshov S.V. and Zaytseva A.A.* Active Data in Digital Software Defined Systems Based on SEMS Structures // Logical Analysis of Data and Knowledge with Uncertainties in SEMS – A.E. Gorodetskiy (ed.), Smart Electromechanical Systems, Studies in Systems, Decision and Control. – 2016. – Vol. 49. – P. 61-69.
14. *Шальнев И.О.* Особенности работы с комплементарными объектами в распределенной виртуальной среде // Матер. конференции «Информационные технологии в управлении». – 2020. – С. 224-226.
15. *Henning M., Vinoski S.* Advanced CORBA Programming with C++ // Addison-Wesley Professional; 1st edition. – 1999, February. – P. 560.
16. *Grosso W.* Java RMI // O'Reilly Media, Inc. – 2001, October. – P. 576.
17. *Таненбаум Э.* Компьютерные сети. – СПб.: Питер, 2020. – 992 с.
18. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологии, протоколы. – СПб.: Питер, 2002. – 672 с.
19. *Шальнев И.О., Аксенов А.Ю.* Генерация пользовательского интерфейса на основе технологии распределенной виртуальной среды // Информационно-измерительные и управляющие системы. – 2019. – Т. 17, № 5. – С. 44-50. – Doi: 10.18127/j20700814-201905-06.
20. *Kuleshov S.V., Zaytseva A., Aksenov A.Y.* The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data transmission network // International Journal of Intelligent Unmanned Systems. – Vol. 6, Issue 4. – P. 174-183. – Doi: 10.1108/IJIUS-04-2018-0010.

REFERENCES

1. Dronecode foundation. Available at: https://mavlink.io/en/ (accessed 17 January 2021).
2. *Hohpe, G. and Woolf, B.* Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Pearson Education, 2012, pp. 106. ISBN 9780133065107.
3. *Matthew O'Riordan* Everything You Need To Know About Publish/Subscribe. Available at: https://www.ably.io/topic/pub-sub (accessed 09 February 2021).
4. *Robert Nystrom* Game Programming Patterns. Genever Benning. November 2014, pp. 354.
5. *Stevens R.W.* UNIX Network Programming – Interprocess Communication, *Englewood Cliffs*. Prentice Hall. 2nd ed., 1999, August, 400 p.
6. *Pethuru Raj, Anupama Raman, Harihara Subramanian.* Architectural Patterns, *Packt*, 2017, December, pp. 468.
7. Extensible Markup Language (XML). Available at: https://www.w3.org/XML/ https://www.ably.io/topic/pub-sub (accessed 10 February 2021).
8. ROS Documentation. Available at: http://wiki.ros.org/.
9. *Tanenbaum E., van Steen M*. Raspredelennye sistemy (printsipy i paradigmy) [Distributed systems (principles and paradigms)]. Saint Petersburg: Piter, 2003, 880 p.
10. *Buch G., Maksimchuk R.A., Engl M.U., Yang B.Dzh., Konallen D., Kh'yuston K.A.* Ob"ektno-orientirovannyy analiz i proektirovanie s primerami prilozheniy [Object-oriented analysis and design with examples of applications]. 3rd ed. Moscow: Vil'yams, 2010, 720 p.
11. *Shal'nev I.O.* Podkhod k postroeniyu raspredelennoy virtual'noy mashiny na osnove ob"ektno-orientirovannogo programmirovaniya [An approach to building a distributed virtual machine based on object-oriented programming], *Izvestiya Tul'skogo gosudarstvennogo universiteta* [Proceedings of the Tula State University], 2020, No. 9, pp. 40-47.
12. *Wei Chen, Weixia Wu, Zhiying Wang, Qiang Zhao* A Formalization of An Emulation based Co-Designed Virtual Machine, *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011, pp. 164-168.

13. *Alexandrov V.V., Kuleshov S.V. and Zaytseva A.A.* Active Data in Digital Software Defined Systems Based on SEMS Structures, *Logical Analysis of Data and Knowledge with Uncertainties in SEMS – A.E. Gorodetskiy (ed.), Smart Electromechanical Systems, Studies in Systems, Decision and Control*, 2016, Vol. 49, pp. 61-69.
14. Shal'nev I.O. Osobennosti raboty s komplementarnymi ob"ektami v raspredelennoy virtual'noy srede [Features of working with complementary objects in a distributed virtual environment], *Mater. konferentsii «Informatsionnye tekhnologii v upravlenii»* [Materials of the conference "Information Technologies in Management"], 2020, pp. 224-226.
15. *Henning M., Vinoski S.* Advanced CORBA Programming with C++, *Addison-Wesley Professional; 1st edition*, 1999, February, pp. 560.
16. *Grosso W.* Java RMI, *O'Reilly Media, Inc*., 2001, October, pp. 576.
17. *Tanenbaum E*. Komp'yuternye seti [Computer networks]. Saint Petersburg: Piter, 2020, 992 p.
18. *Olifer V.G., Olifer N.A.* Komp'yuternye seti. Printsipy, tekhnologii, protokoly [Computer networks. Principles, technologies, and protocols]. Saint Petersburg: Piter, 2002, 672 p.
19. *Shal'nev I.O., Aksenov A.Yu.* Generatsiya pol'zovatel'skogo interfeysa na osnove tekhnologii raspredelennoy virtual'noy sredy [Generation of the user interface based on the distributed virtual environment technology], *Informatsionno-izmeritel'nye i upravlyayushchie sistemy* [Information-measuring and control systems], 2019, Vol. 17, No. 5, pp. 44-50. Doi: 10.18127/j20700814-201905-06.
20. *Kuleshov S.V., Zaytseva A., Aksenov A.Y*. The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data transmission network, *International Journal of Intelligent Unmanned Systems*, Vol. 6, Issue 4, pp. 174-183. Doi: 10.1108/IJIUS-04-2018-0010.

Статью рекомендовал к опубликованию д.т.н., профессор И.С. Лебедев.

Шальнев Илья Олегович – Федеральное государственное бюджетное учреждение науки «Санкт-Петербургский Федеральный исследовательский центр Российской академии наук»; e-mail: shalnev.i@iias.spb.su; г. Санкт-Петербург, Россия; тел.: +79111756194; аспирант; м.н.с.

Shalnev Ilia Olegovich – St. Petersburg Federal Research Center of the Russian Academy of Sciences; e-mail: shalnev.i@iias.spb.su; Saint-Petersburg, Russia; phone: +79111756194; postgraduate student; junior researcher.