УДК 551.46.077:629.584

RCE – ПРОГРАММНАЯ ПЛАТФОРМА ДЛЯ СИСТЕМЫ УПРАВЛЕНИЯ АНПА

Л.А. Наумов, А.И. Боровик, Н.В. Баль

Институт проблем морских технологий ДВО РАН[1]

Рассматриваются вопросы создания программного обеспечения системы управления (СУ) автономного необитаемого подводного аппарата (АНПА) на базе программной платформы (ПП). Приводится аналитический обзор используемых в настоящее время ПП управления роботами с учетом особых требований, выдвигаемых СУ АНПА (полностью автономная работа, работа при наличии каналов связи с низкой пропускной способностью, специфические задачи и область применения АНПА). Описываются особенности создаваемой платформы RCE (Robot Components Engine), предназначенной для системы управления АНПА. Приводится пример архитектуры программного обеспечения СУ АНПА, построенного на описываемой платформе.

ВВЕДЕНИЕ

Одним из важнейших компонентов АНПА, как и любого другого робота, является система управления – комплекс программ и драйверов оборудования, обеспечивающий автономную работу аппарата и выполнение им поставленного задания. Программы, входящие в состав системы управления, могут выполняться на различных компьютерах, находящихся как на борту робота, так и вне его – на сопровождающем судне или береговой базе [1].

В настоящее время в мировой практике используются различные модели архитектуры программного обеспечения для систем управления автономными роботами. В частности, в АНПА, разработанных в ИПМТ ДВО РАН, используется трехуровневая модель разумного поведения (Rational Behavior Model) с элементами расслоенного управления на исполняющем уровне [2].

Какая бы модель архитектуры ни была выбрана для системы управления, одним из начальных этапов создания всего программного комплекса является решение задачи транспорта данных между компонентами системы. Данная задача может быть решена путем реализации транспортных функций в каждом из компонентов системы или путем выбора соответствующего промежуточного программного обеспечения (ППО, middleware). Первый способ может использоваться, если количество компонент системы невелико, но для любой сложной системы его применение неоправданно, так как приводит к усложнению компонентов и отсутствию унификации. Использование ППО, напротив, позволяет добиться модульности системы, облегчает и организует процесс написания отдельных компонентов.

Промежуточное программное обеспечение – это среда, обеспечивающая унифицированный доступ ко всем компонентам программного обеспечения (ПО) системы, средства адресации компонентов и транспорт сообщений между ними. ППО исполняет роль прослойки между компонентами и операционной системой.

Существует несколько технологий организации связи компонентов СУ: клиент-серверная, пиринговая (одноранговая) и гибридная.

В рамках *клиент-серверной* технологии связь между компонентами системы осуществляется посредством выделенной программы-сервера, которая всегда должна находиться в сети. Каждый компонент системы должен иметь доступ к серверу. К плюсам данной технологии относятся простота реализации, высокая скорость нахождения компонентами необходимых для работы данных, а также возможность прозрачного накопления данных (сервер может дублировать все пересылаемые через него сообщения в единую базу данных). Минусы – низкая надежность (выход из строя сервера приводит к неработоспособности всей

[1] 690091, Владивосток, ул. Суханова, 5а, тел.: (423) 2432416, e-mail: imtp@marine.febras.ru

сети) и высокая нагрузка на транспортные механизмы ОС (сеть, сокеты), так как каждое сообщение от источника до получателя транспортируется минимум два раза.

Пиринговая (или *одноранговая)* технология основана на организации сети равноправных компонентов. Все компоненты системы связываются друг с другом независимо. К плюсам данной технологии относится высокая надежность работы (любые компоненты могут аварийно завершаться и запускаться вновь, при этом работа остальных программ СУ не будет нарушена) и малое время отклика системы (данные от источников к потребителям передаются напрямую, минуя сервер). К недостаткам технологии можно отнести сложный процесс настройки сети и избыточную нагрузку на транспортные механизмы источников данных (сообщения должны быть отправлены всем потребителям, что может затормозить процесс рассылки).

Гибридная технология основана на организации частично децентрализованных сетей. В гибридной сети присутствует выделенный сервер (или несколько), осуществляющий синхронизацию и координацию работы компонентов. Компоненты-потребители данных находят компоненты-источники посредством обращения к серверу, после чего устанавливают друг с другом прямое соединение и работают по принципу пиринговой сети. Данная технология совмещает скорость и удобство настройки централизованных (клиент-серверных) сетей и надежность децентрализованных. В децентрализованной сети сервер может брать на себя ответственность по доставке сообщений потребителям, которые по каким-либо причинам не могут установить прямое соединение с источниками. Таким образом, клиент-серверную и одноранговую технологии организации сети компонентов можно рассматривать как частные случаи гибридной.

В настоящий момент существует промежуточное программное обеспечение как общего (предназначенного для связи разнородных программ между собой), так и специального назначения. К наиболее известному ППО общего назначения относятся: *IPC* [3], *ICE* [4], *CORBA* [5]. Под ППО специального назначения понимаются программные платформы (framework), специально предназначенные для решения задач робототехники. Программные платформы, как правило, не только реализуют транспортные функции, но и предоставляют унифицированные средства конфигурирования компонентов, эмулирующие комплексы, вспомогательные программы для отладки алгоритмов и так далее. Наиболее известные программные платформы: *CARMEN* [6], *Orca* [7], *Microsoft Robotics Developer Studio* [8], *Player* [9], *MOOS-IvP* [10]. Программные платформы могут использовать стороннее ППО (CARMEN использует IPC, Orca – ICE, MSRS – CCR) или реализовывать собственные транспортные механизмы (Player).

Обзор программных платформ

С учетом специфики АНПА (автономная работа, плохой канал связи с внешними компонентами системы управления, высокая вероятность аппаратных сбоев, низкая производительность используемых в ЛВС аппарата компьютеров) можно выделить следующие базовые требования к программной платформе системы управления:

- модульность (возможность независимого добавления, изменения и удаления драйверов оборудования, программ управления движением и прочих служебных подпрограмм, входящих в СУ);
- стабильность (ошибки, возникающие в отдельных модулях системы, не должны приводить к выходу из строя всей системы или других компонентов);
- низкая ресурсоемкость (возможность программного обеспечения СУ функционировать на промышленных одноплатных компьютерах, используемых в бортовой сети АНПА);
- малое время отклика (данные от сенсоров, непосредственно участвующих в управлении движением, должны доставляться регуляторам движения с минимально возможными задержками);
- кроссплатформенность (возможность компиляции программного кода ПП под операционными системами семейств Linux, QNX и Windows).

Далее будут рассмотрены наиболее известные программные платформы с открытым исходным кодом – ORCA, CARMEN и Player Project.

Проект ***CARMEN*** позиционируется как набор программ управления автономным мобильным роботом. В рамках проекта решаются задачи навигации, составления карт, планирования траектории, обхода препятствий, протоколирования данных и т.д. [6].

Работа изначально финансировалась подразделением министерства обороны США Defense Advanced Research Projects Agency (DARPA) в рамках программы MARS (Mobile

Autonomous Robot Software). На данный момент проект не развивается, но продолжает активно использоваться.

Компоненты CARMEN функционируют в рамках событийно-управляемой модели. Для обеспечения межпроцессорного взаимодействия разработан механизм обмена сообщениями между модулями через TCP соединение с использованием механизма на основе библиотеки с десятилетней историей Inter Process Communication (IPC). IPC последних версий реализует два альтернативных режима работы: клиент-серверный и пиринговый, однако CARMEN использует только клиент-серверный режим.

Основным архитектурным решением, отличающим ПП CARMEN от аналогичных разработок, является наличие выделенного централизованного сервера параметров. Этот компонент обеспечивает хранение изменяемых данных модулей системы и доступ к ним. Модули могут добавлять свою информацию на сервер, получать данные сервера и подписываться на их обновления. Режимы работы, начальные значения параметров модулей системы и прочие данные определяются из конфигурационного файла при старте сервера параметров.

Несомненным плюсом CARMEN является встроенная возможность симуляции, сохранения и воспроизведения данных, а также визуализации работы эмулируемого аппарата (хотя это реализовано только для двухмерного представления пространства и объектов). К достоинствам системы можно также отнести наличие в платформе драйверов для достаточно большого числа разнообразного промышленного оборудования.

К минусам проекта можно отнести использование потенциально ненадежной клиент-серверной технологии, а также отсутствие поддержки операционной системы QNX (хотя библиотека взаимодействия IPC поддерживает данную ОС). В проекте реализованы интерфейсы обмена некоторыми специфическими для робототехники типами данных, однако добавление собственных затруднено и требует изучения макрокоманд IPC. Кроме того, развитие проекта остановилось в 2008 году, новых сборок с того времени не появлялось.

ORCA – это программная платформа с открытым исходным кодом для разработки систем управления роботами, основанная на «компонентном» подходе. Ключевой идеей такого подхода является концепция строго описанных интерфейсов, регулирующих взаимодействие между компонентами. При этом нет никаких дополнительных ограничений на общую архитектуру системы, набор используемых интерфейсов и внутреннюю архитектуру компонентов. Другими словами, ORCA предоставляет набор правил для описания и создания блоков системы, а также некоторые базовые компоненты, которые могут использоваться для создания и описания как сложных, так и достаточно простых систем управления роботами [7].

Платформа ORCA разрабатывается международным сообществом энтузиастов, основу которого составляют сотрудники Австралийского центра робототехники (Australian Centre for Field Robotics) при университете Сиднея. ORCA применяется в нескольких коммерческих и образовательных проектах, среди которых проект IWARD, спонсируемый Евросоюзом, несколько проектов в Технологическом университете Сиднея (University of Technology Sydney) и некоторые другие.

ПП ORCA базируется на технологии создания распределенных компонентов ICE (Internet Communication Engine), дополняя ее расширениями интерфейсов, специфичными для задач программирования роботов. В рамках этой технологии каждый компонент представляет собой отдельное приложение, получающее данные по одним интерфейсам и публикующее данные по другим. За организацию связи между компонентами отвечают два сервиса:

- сервис именования объектов IceGrid Registry, предназначенный для организации доступа компонентов друг к другу по именам;
- сервис диспетчеризации событий IceStorm, предназначенный для разделения подписчиков и генераторов событий.

Каждый компонент должен иметь возможность находить вышеперечисленные сервисы в сети.

Среда ***ICE*** является современным свободно распространяемым аналогом устаревшей технологии DCOM/CORBA и предоставляет широкие возможности кроссплатформенного взаимодействия приложений [4].

Платформа ORCA в полной мере удовлетворяет требованиям модульности, низкой ресурсоемкости и малого времени отклика. Требования к процессору и памяти со стороны сервисов ICE минимальны – во время тестов с десятью компонентами на операционной системе Linux использование оперативной памяти не превышало 5 Мб. Скорость доставки сообщений от одних компонентов к другим при использовании компьютерной сети со скоростью передачи данных 1 Гбит/с достаточна для

организации обмена сообщений с частотой около 1 ГГц. Тесты производительности описаны в [11].

В то же время платформа не предусматривает возможности зеркалирования сервисов ICEGrid Registry и ICEStorm, поэтому в случае возникновения критических ошибок в их работе (или работе компьютера, на котором они выполняются) дальнейшее продолжение работы компонентами станет невозможным. К минусам платформы также следует отнести слабую поддержку операционной системы QNX (для которой можно собрать лишь отдельные компоненты и модули), а также зависимость от стороннего программного продукта ICE. Внедрение и использование данной платформы требует изучения принципов функционирования среды ICE, а также ставит всю разработку в зависимость от этой платформы. Для добавления новых интерфейсов в платформу также требуется дополнительно изучить набор макрокоманд ICE.

Платформа не содержит собственных средств симуляции, но позволяет использовать симулятор Stage из проекта Player.

Player представляет собой сервер программной платформы, который обеспечивает среду для работы компонентов системы управления и транспорт сообщений между ними. В рамках платформы Player все компоненты системы управления четко разделяются на драйверы, реализованные как функциональные модули программы-сервера, и клиентские программы, общающиеся с сервером посредством протоколов TCP или UDP. Драйверы контролируют работу конкретных устройств или других драйверов (в этом случае они называются «метадрайверы») и выполняются в отдельных программных потоках процесса сервера Player. Клиентские программы могут быть написаны на разнообразных языках программирования и располагаться на любых компьютерах, имеющих связь по сети с роботом. Связь драйверов между собой и с клиентскими программами происходит посредством интерфейсов – правил, описывающих состав передаваемых сообщений, характер их передачи и тип ответной реакции. Клиенты и драйверы системы при работе ориентируются не на конкретные реализации других компонентов, а на набор предоставляемых интерфейсов. В рамках одной системы управления может использоваться любое количество серверов и клиентских программ, располагающихся на большом количестве компьютеров, объединенных в сеть. Сервер платформы Player может быть скомпилирован для большого количества операционных систем (все POSIX-совместимые, Windows) [9, 12].

Платформа Player изначально разрабатывалась сотрудниками Университета Южной Калифорнии (University of Southern California). В настоящий момент Player является самой известной в мире ПП и насчитывает более 100 крупных промышленных пользователей в 27 странах, среди которых такие известные компании, как «Intel», «Boeing» и «Stanford AI Laboratory».

Платформа Player в полной мере удовлетворяет требованиям низкой ресурсоемкости, малого времени отклика и кроссплатформенности.

Платформа обеспечивает достаточно хорошую производительность и может стабильно обеспечивать частоту обмена данными в 1 ГГц между клиентами и сервером, соединенными посредством компьютерной сети 1 Гбит/с. Детально тесты производительности платформы описаны в [12]. Требования к процессору и памяти со стороны механизмов сервера минимальны: на операционной системе Linux во время тестов с 10 компонентами использование памяти потоком сервера не превышало 4 Мб. Player компилируется и стабильно работает под большим количеством операционных систем, в том числе Windows, Linux и QNX. От прочих платформ его также выгодно отличает практически полное отсутствие внешних зависимостей и лаконичность исходного кода. Кроме того, система Player содержит самые продвинутые среди всех рассмотренных платформ средства симуляции как в двумерном (проект Stage), так и в трехмерном (проект Gazebo) пространстве.

Требованиям модульности и стабильности система удовлетворяет не в полной мере. Драйверы реализованы как потоки общей программы-сервера, тем самым критическая ошибка в одном из драйверов ведет к краху программы-сервера и остальных драйверов, выполняющихся на нем. Данная ситуация может быть исправлена путем выполнения каждого драйвера в отдельном сервере, однако подобный подход существенно усложняет настройку всей системы. Кроме того, все функции платформы реализованы на языке C и содержат множество потенциально опасных конструкций. Попытка передать неверно сформированное сообщение может привести не только к краху передающей стороны, но даже и к краху получателя.

Добавление новых драйверов к системе Player возможно

без перекомпиляции сервера – благодаря наличию в системе возможности подключать драйверы, реализованные как динамические библиотеки. В то же время изменение старого или добавление нового интерфейса требует перекомпиляции всего сервера. Кроме того, при добавлении нового интерфейса необходимо использовать особый макроязык описания и следить за тем, чтобы название и номер добавляемого интерфейса не перекликались с существующими в системе.

Использование платформы Player, кроме того, требует на этапе проектирования системы управления разделять программы на серверные (драйверы, метадрайверы) и клиентские, что накладывает определенные ограничения на архитектуру и не всегда оправданно.

Описание программной платформы RCE

В связи с отсутствием среди рассмотренных платформ системы, в полной мере удовлетворяющей указанным требованиям, нами было решено разработать новую платформу, лишенную некоторых присущих другим недостатков. Разрабатываемая платформа для системы управления АНПА RCE (Robot Components Engine) основана на гибридной технологии организации сети. Гибридная технология выбрана как наиболее удовлетворяющая предъявленным требованиям. В разрабатываемой платформе вводится понятие роли компонента системы управления. Каждый компонент может выполнять одну из следующих ролей:

- чистый клиент (поддерживает только одно активное соединение с сервером, по которому передает исходящие и получает входящие сообщения);
- пиринговый клиент (запрашивает у сервера список потребителей данных, после чего устанавливает с ними прямое соединение);
- адресующий сервер (хранит сведения о текущей топологии сети и связывает пиринговые клиенты-источники и пиринговые клиенты-потребители данных между собой);
- передающий сервер (выполняет роль адресующего сервера для пиринговых клиентов и передает данные для чистых клиентов).

Любой компонент системы управления может выполнять одну из описанных ролей независимо от его основного назначения. Все транспортные функции выполняются в отдельном программном потоке независимо от потоков, выполняющих «полезную нагрузку» компонента. Настройка роли компонента может осуществляться как в коде компонента, так и посредством конфигурационного файла, тем самым позволяя организовывать сети любой топологии на основе уже написанных компонентов, без необходимости вносить в них какие-либо изменения. В рамках платформы RCE возможно как создание отдельных программ-серверов, выполняющих определенный функционал (запись всех циркулирующих сообщений в базу данных, слежение за работой и перезапуск других компонентов), так и передачу серверных функций любым компонентам системы. Пиринговые клиенты могут определять максимальное количество соединений, которые они будут обслуживать самостоятельно; доставку данных для всех подписчиков выше этого числа выполняет передающий сервер.

Разрабатываемая платформа имеет ряд ключевых особенностей, отличающих ее от аналогичных продуктов.

- Гибкость настройки сети. Благодаря использованию гибридной технологии, платформа RCE позволяет организовать сети практически любой структуры, как клиент-серверные с выделенным сервером, так и полностью гибридные.
- Прозрачная конфигурируемость. Роль каждого компонента может быть задана на этапе запуска посредством конфигурационного файла, тем самым позволяя организовывать сети различной структуры на основе уже написанных компонентов.
- Динамическая подписка на сообщения. Компонент может динамически подписываться на получение и отписываться от получения определенных сообщений.
- Поддержка симуляции. Все компоненты получают текущее время посредством вызова специальных функций ядра платформы. С помощью передачи специальных системных сообщений эмулятор может ускорить или замедлить течение времени для компонентов, работающих в условиях симуляции.
- Простое описание собственных структур передачи данных. В RCE сообщения передаются в виде объектов определенных программистом C++ классов. Объекты могут содержать данные любых типов, в том числе динамических и определенных программистом. Классы для передачи данных описываются посредством стандартных средств языка C++, от программиста не требуется изучения отдельного макроязыка описания (как в Player, CARMEN и Orca).
- Удобство использования платформы. RCE предоставляет программисту небольшое

количество удобных функций взаимодействия с платформой, выполняя все транспортные функции в отдельных потоках.

● Безопасность использования. Благодаря использованию языка C++ возможность критической ошибки при работе с платформой, не выявляемой на этапе компиляции, сведена к минимуму.

В рамках проекта RCE планируется создание полноценной среды симуляции. Среда симуляции может быть выполнена как адресующий сервер, передающий компонентам СУ данные от эмулируемых источников, а также поправочные коэффициенты для симуляции времени.

■ Пример архитектуры СУ

Все программы, участвующие в управлении движением и задании параметров работы оборудования АНПА, размещены по трем уровням – стратегическому, тактическому и исполняющему. Каждая из этих программ реализована как пиринговый клиент RCE. Программы, работающие с конкретным оборудованием или выполняющие базовый функционал, назовем обслуживающими. К обслуживающим программам относятся драйверы оборудования, сервера RCE и некоторые служебные программы.

В базовой комплектации АНПА типа МТ-3000 [13] или «Пилигрим» [14, 15] содержит следующие устройства, каждому из которых соответствует собственный драйвер:

● навигационно-пилотажные датчики (НПД);

● датчик глубины (ДГ);

● доплеровский лаг (ДЛ);

● измеритель параметров внешней среды (ИПС);

● эхолокационная система (ЭЛС);

● блок управления двигателями (БУД);

● супервизоры питания и системных контроллеров (СП);

● телевизионная система (ТС);

● система акустического зрения (САЗ);

● гидроакустическая навигационная система (ГАНС);

● гидроакустическая система связи (ГАСС);

● инерциальная навигационная система (ИНС);

● аккумуляторные батареи (АБ);

● приемник GPS/ГЛОНАСС.

Драйверы реализованы как пиринговые клиенты RCE с ограниченным числом максимальных подключений.

К обслуживающим программам также относится *программа ведения протокола событий*, которая записывает все циркулирующие в системе данные в базу данных или бинарный файл специальной структуры.

В расслоенную структуру исполняющего уровня объединяются следующие служебные программы:

● *программа навигации* определяет координаты АНПА на основе информации от навигационно-пилотажных датчиков, доплеровского лага, датчика глубины, ИНС, приемника GPS/ГЛОНАСС и ГАНС;

● *регулятор движения* реализует движение АНПА в заданном направлении или к заданной цели с обходом локальных препятствий. Для работы используют данные от программы навигации, навигационно-пилотажных датчиков, датчика глубины и эхолокационной системы;

● *контрольно-аварийная система (КАС) нижнего уровня* осуществляет контроль над работой всех драйверов и прочих служебных программ, а также предпринимает некоторые безусловные действия для ликвидации возникающих аварий.

На тактическом уровне находится служебная программа прокладки маршрута по карте, которая осуществляет планирование движения АНПА к указанной цели с учетом участков, запрещенных к движению (берега, острова, запретные зоны). Программа прокладки маршрута по карте разбивает весь маршрут движения на отдельные простые участки, передаваемые в качестве целей регулятору движения. На тактическом уровне находятся также и другие программы, реализующие сложные поведения АНПА (отслеживание и инспекция кабелей и трубопроводов, приведение к стыковочному модулю, автоматическое докование, перепланирование покрытия заданной акватории и т.п.), и *планировщик программ*, позволяющий запускать и останавливать их.

На верхнем (стратегическом) уровне системы управления находятся программы, определяющие цели запуска, последовательность действий и режимов работы АНПА, а также управляющие работой серверов ПП, драйверов и служебных программ, входящих в их состав.

В системе управления присутствуют следующие программы стратегического уровня [2]:

● *программа-миссия* содержит цели запуска АНПА, маршрут движения и последовательность действий и режимов работы оборудования. Основная программа-миссия составляется и тестируется заранее, дополнительные программы могут быть переданы на аппарат, скомпилированы и запущены посредством функ-

ций контрольно-аварийной системы высшего уровня;

● *программа телеуправления* позволяет оператору напрямую влиять на ход выполнения миссии, а также получать отчет о текущем состоянии оборудования аппарата (при наличии связи с АНПА);

● *КАС высшего уровня* – программа, осуществляющая управление работой серверов программной платформы системы управления, их запуск с определенными настройками и останов. КАС также может компилировать, запускать и останавливать программы-миссии.

Все программы, входящие в состав системы управления АНПА, с учетом их размещения по иерархическим уровням системы представлены на рис. 1.

Бортовая сеть АНПА состоит из нескольких компьютеров, объединенных в сеть. Каждый компьютер контролирует работу определенного набора оборудования и участвует в выполнении общей миссии. Все бортовые устройства подключены к общему последовательному каналу связи, к которому также подключены и все компьютеры АНПА. В результате каждый компьютер робота имеет доступ ко всем устройствам [2]. СУ АНПА типа МТ-3000 или «Пилигрим» содержит три компьютера, роли которых распределены следующим образом:

● главный компьютер (компьютер автопилота) обеспечивает достижение целей миссии, работу основных драйверов и служебных программ;

● компьютер телевизионной системы обеспечивает работу драйверов фото- и видеоаппаратуры, а также программ обработки изображений и распознавания образов;

● компьютер системы акустического зрения обеспечивает работу ГБО и профилографа, а также программ обработки акустических изображений.

Разработанная архитектура предусматривает возможность дублирования функций автопилота другими компьютерами. В бортовой сети АНПА предусмотрен аппаратный супервизор КАС, контролирующий работу компьютеров. В случае выхода из строя компьютера автопилота, аппаратная КАС принимает решение о делегировании функций компьютера автопилота одному из компьютеров АНПА. При этом программа *КАС высшего уровня* запускает на выбранном компьютере компоненты, выполняющиеся на компьютере автопилота, а также приостанавливает выполнение некоторых локальных компонентов, если этого требуют соображения производительности. Благодаря использованию гибридной технологии организации сети данная процедура не требует перенастройки или перезапуска остальных компонентов.

На компьютере автопилота выполняются два сервера *RCE* – основной и дублирующий. Адресация и транспорт

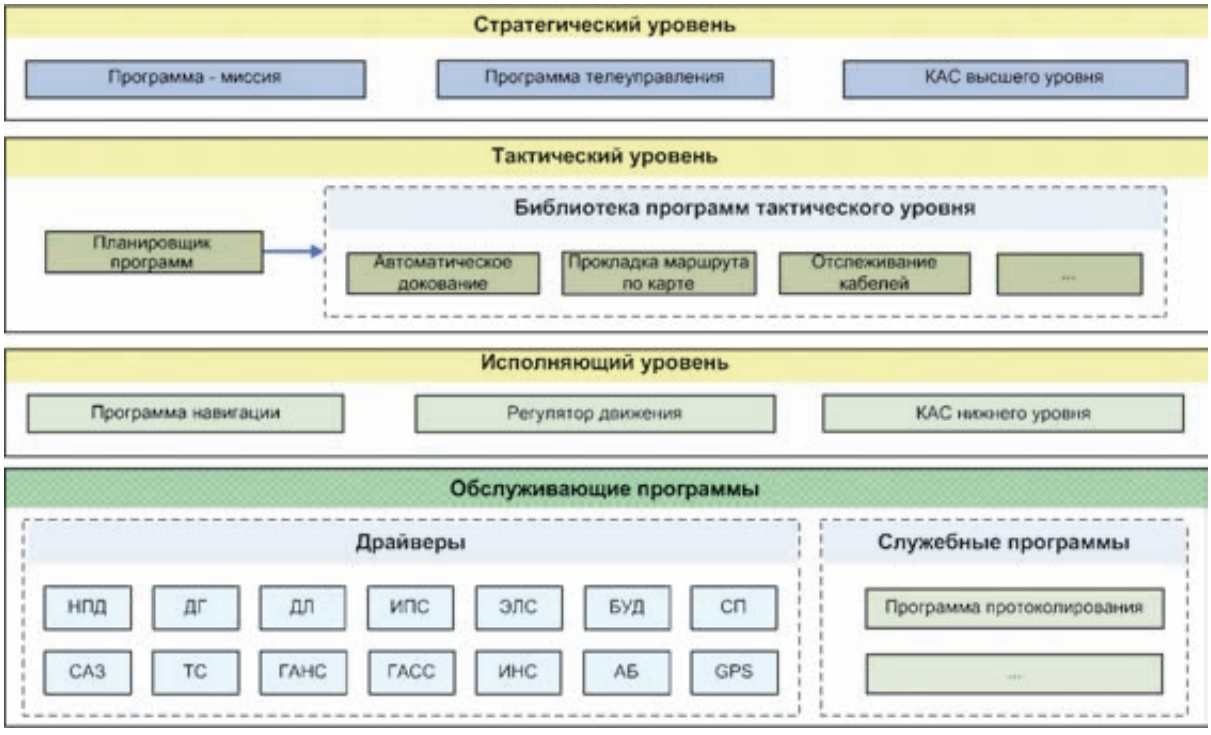

Рис. 1. Архитектура системы управления АНПА

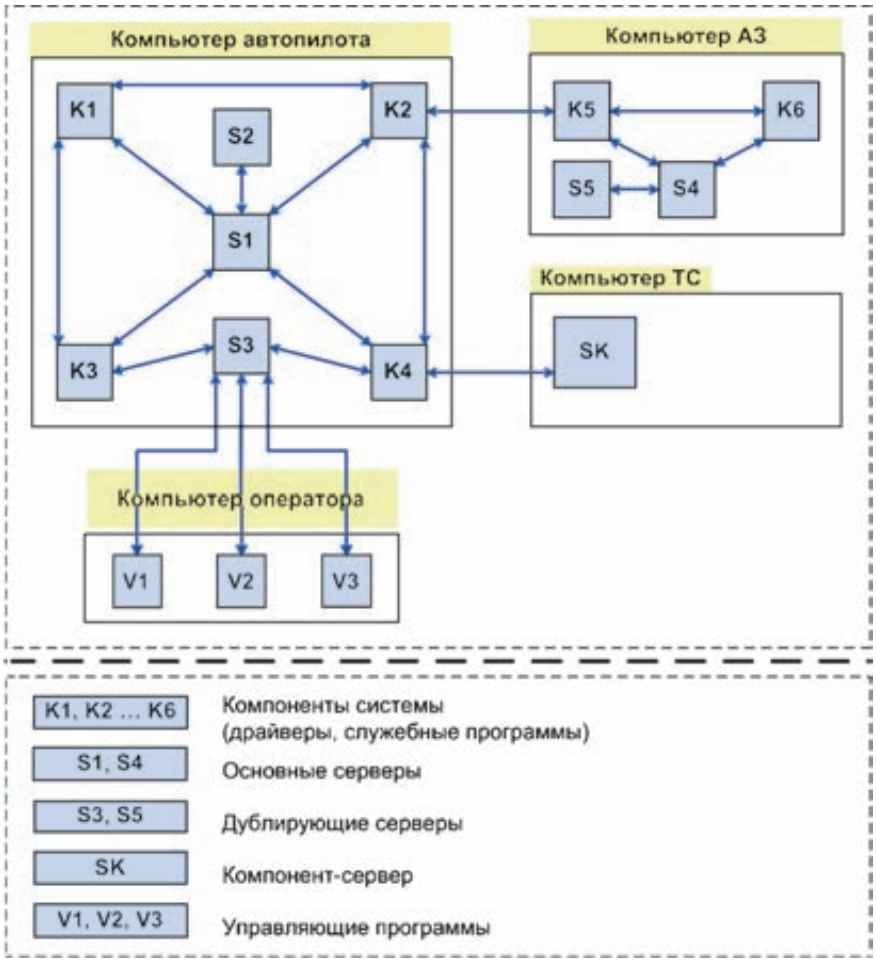

Рис. 2. Пример реализации бортовой сети

(при необходимости) выполняются посредством основного сервера.

Пример реализации бортовой сети с использованием *RCE* изображен на рис. 2.

Разрабатываемая ПП и формируемая на ее базе архитектура системы управления предполагаются к внедрению в одном из проектов ИПМТ ДВО РАН.

ЛИТЕРАТУРА

1. Инзарцев А.В., Львов О.Ю. Бортовые вычислительные сети автономных подводных роботов // Современные технологии автоматизации. 2005. №2. С. 68–74
2. Агеев М.Д., Киселев Л.В., Матвиенко Ю.В. и др. Автономные подводные роботы: системы и технологии / под общ. ред. М.Д. Агеева. М.: Наука, 2005.
3. Inter Process Communication (IPC) [Электронный ресурс] / IPC. URL: http://www.cs.cmu.edu/~ipc/, свободный. Загл. с экрана. Яз. англ.
4. The Internet Communications Engine [Электронный ресурс] / ZeroC, Inc. URL: http://www.zeroc.com/ice.html, свободный. Загл. с экрана. Яз. англ.
5. The official CORBA standard from the OMG group [Электронный ресурс] / OMG group. URL: http://www.omg.org/spec/CORBA/3.1/, свободный. Загл. с экрана. Яз. англ.
6. Carmen Robot Navigation Toolkit [Электронный ресурс] / CARMEN-Team. URL: http://carmen.sourceforge.net/, свободный. Загл. с экрана. Яз. англ.
7. Orca: Components for Robotics [Электронный ресурс] / Orca Robotics; Web-мастер Tobias Kaupp. URL: http://orca-robotics.sourceforge.net/, свободный. Загл. с экрана. Яз. англ.
8. Microsoft Robotics Developer Studio [Электронный ресурс] / Microsoft. URL: http://www.microsoft.com/robotics/, свободный. Загл. с экрана. Яз. англ.
9. Player [Электронный ресурс] / The Player Project. URL: http://playerstage.sourceforge.net/index.php?src=player, свободный. Загл. с экрана. Яз. англ.
10. MOOS-IvP [Электронный ресурс] / MOOS-IvP. URL: http://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php, свободный. Загл. с экрана. Яз. англ.
11. Performance of Orca Components [Электронный ресурс] / Orca Robotics; Web-мастер Tobias Kaupp. URL: http://orcarobotics.sourceforge.net/orca_doc_performance.html, свободный. Загл. с экрана. Яз. англ.
12. Gerkey B., Vaughan R. and Howard A. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems // Proc. of the 11th Int. Conf. on Advanced Robotics (ICAR 2003). 2003, June. Coimbra, Portugal, 2003. P. 317–323.
13. Горнак В.Е., Инзарцев А.В., Львов О.Ю., Матвиенко Ю.В., Щербатюк А.Ф. ММТ-3000 – новый малогабаритный автономный необитаемый подводный аппарат ИПМТ ДВО РАН // Подводные исследования и робототехника. 2007. №1(3). С. 12–20.
14. Пилигрим – новый русский подводный робот [Электронный ресурс] // DiveМир – мир дайвинга и приключений. URL: http://divemir.com/2010/10/piligrim-russian-underwater-robot/
15. Борейко А.А., Горнак В.Е, Матвиенко Ю.В. и др. Малогабаритный многофункциональный АНПА «МТ-2010» // В настоящем номере журнала.