

А. Д. Ковальчук

ПРОБЛЕМЫ НАВИГАЦИИ И КОММУНИКАЦИИ В КОСМИЧЕСКОМ ПРОСТРАНСТВЕ

Статья посвящена проблемам координирования деятельности летательных аппаратов, функционирующих в открытом космосе. Рассмотрены способы организации навигационно-коммуникационной сети, а также архитектура программного обеспечения для её узлов. В статье изложены требования к функционалу приложений на клиентских устройствах сети, позволяющему летательным аппаратам ориентироваться в космосе и вести диалог друг с другом. Учтены такие аспекты, как надёжность программного обеспечения, его автономность и перспективы усовершенствования.

В настоящее время насчитывается более 2700 действующих космических летательных аппаратов на орбите Земли и несколько десятков за её пределами. Это число будет неуклонно расти, что требует координирования деятельности устройств, обеспечения их средствами навигации и связи. Проблема решается путём создания навигационно-коммуникационной сети, по которой каждый аппарат будет получать достоверные сведения об объектах и пространстве в зоне его деятельности.

Сеть должна включать центральный узел, именуемый сервером, а также обсерватории и клиентские узлы. Сервер запрашивает данные о действующих устройствах и конфигурацию небесных тел у обсерваторий, которые ведут непрерывный мониторинг своих регионов. Информация проходит обработку, а затем по запросу передаётся узлам-клиентам в виде карт и списков [1, с. 16]. Базовая схема сети представлена на рисунке 1:

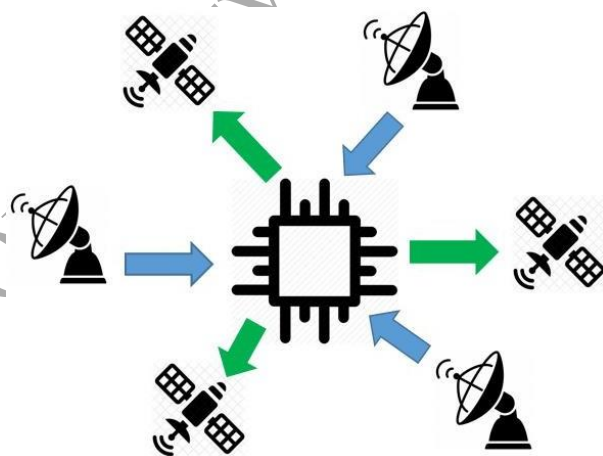


Рисунок 1 – Базовая схема навигационно-коммуникационной сети

С расширением пространства, в котором функционирует сеть, сбор данных сервером с удалённых обсерваторий и передача на клиентские узлы может вызвать существенные задержки. В этом случае целесообразно разбить пространство на регионы, в каждом из которых будет собственный сервер, отвечающий за снабжение сведениями узлов, относящихся к конкретному региону. Подход повысит устойчивость сети к перебоям на стороне центральных узлов, а также ускорит обслуживание летательных аппаратов.

Передача данных между обсерваториями, сервером и клиентскими узлами обеспечивается за счёт ретрансляторов. Это могут быть отдельные устройства или другие

обсерватории и клиентские узлы, расположенные на маршруте передачи. Помимо обмена информацией в пределах регионов, ретрансляторы применимы для синхронизации баз данных региональных серверов между собой [2, с. 12].

Серверная обработка обсерваторных сведений предполагает не только их форматирование перед отправкой на клиентские устройства, но и анализ материала. Например, обсерватории могут ошибочно идентифицировать один и тот же объект как два разных из-за неточного мониторинга или задержек сигнала. Анализ позволит выявить подобные проблемы перед обновлением информации на узлах сети.

Программное обеспечение клиентского устройства должно иметь возможности оперативного построения карт и отображения списков объектов, расположенных поблизости. Целесообразно строить карты по фрагментам с указанием координат каждого фрагмента и степени масштабирования. Это позволит передавать данные порционно и тем самым сохранять устойчивость работы системы при перебоях сигнала. Также фрагментарный подход даёт возможность детализации карты.

Космическое пространство характеризуется не только геометрическими параметрами, но и температурой, туманностью, уровнем радиации, а также магнитными полями. Данные характеристики могут влиять на планирование маршрута движения и деятельности космических летательных аппаратов. С расширением пространства, где работают устройства, и удалением от планетарных станций координация действий аппаратов с Земли будет затруднительна. Поэтому для обеспечения автономной работы клиентские узлы должны получать с сервера многомерные карты регионов.

Передача многомерных фрагментов в явном виде крайне затратна по времени и имеет высокие требования к памяти как на клиентских устройствах, так и на сервере. Альтернативой является хранение и передача данных в цифровых архивах, где конфигурация объектов в пределах фрагментов зашифрована по определённым принципам. Нужен механизм приведения архивированной информации в визуальную форму на клиентских устройствах. Такой механизм может быть реализован через процессор, на вход которого будут поступать массивы фрагментов, а на выход – готовые изображения для обновления карт в динамическом режиме.

Оптимизировать затраты оперативной памяти при построении карт можно путём частичного обновления. Клиентское приложение отправляет запрос на сервер с указанием даты последней загрузки. Сервер, в свою очередь, ведёт историю обновлений и при обработке запроса извлекает из собственного хранилища две версии карты: актуальную и ту, которая была отправлена клиентскому приложению последний раз. В ходе сравнения версий выявляются изменённые фрагменты. Они отправляются клиентской программе, которая обновляет карты точно, что позволяет сэкономить оперативную память.

Объекты, расположенные в зоне деятельности узла-клиента, подразделяются на две основные группы: небесные тела и станции. Первая группа подразумевает планеты, их естественные спутники, а также астероиды. Ко второй группе относятся орбитальные и планетарные станции, искусственные спутники и космические корабли. В таблице 1 представлена структура, в которой данные о небесных телах и станциях хранятся на сервере.

Таблица 1 – Структура данных о небесных телах и станциях

Наименование поля	Тип	Описание	Уникальное значение	Обязательное поле
name	текстовый	Название объекта	да (в рамках группы)	да
type	текстовый	Тип объекта в соответствии с классификацией по группе	нет	нет
distance	числовой	Расстояние до клиентского устройства	нет	да

Продолжение таблицы 1

Наименование поля	Тип	Описание	Уникальное значение	Обязательное поле
description	текстовый	Описание объекта	нет	нет
image_filename	текстовый (ссылка)	Ссылка на изображение объекта в графическом хранилище сервера	да	нет

Главное различие между небесными телами и станциями заключается в возможности установления сигнала со станцией для последующей коммуникации. Поэтому клиентское приложение должно иметь отдельный интерфейс для обновлений данных, относящихся к разным группам. Пример интерфейса для небесных тел представлен на рисунке 2.



Рисунок 2 – Пример интерфейса для небесных тел

Интерфейс для станций, помимо сведений, получаемых с сервера, должен включать данные о качестве сигнала, возможность отправки сообщений и просмотр истории сообщений. Клиентская программа самостоятельно рассчитывает качество сигнала как отношение числа успешных попыток соединения к общему их числу. Пример интерфейса для станций представлен на рисунке 3:



Рисунок 3 – Пример интерфейса для станций

Хранение истории сообщений целесообразно организовать в рамках легковесной базы данных, откуда данные по конкретной станции будут подгружаться по запросу пользователя. В качестве альтернативы клиентская программа может запрашивать историю у сервера, указав в запросе идентификатор своего узла. Это упростит архитектуру приложения и снизит затраты памяти на клиенте, но негативно скажется на автономности.

Литература

1 Ватутин, В. М. Навигация космических аппаратов при исследовании дальнего космоса / В. М. Ватутин. – Москва : Радиотехника, 2016. – 232 с.

2 Верба, В. С. Перспективные технологии цифровой обработки радиолокационной информации космических РСА / В. С. Верба. – Москва : Радиотехника, 2019. – 416 с.

УДК 004.4'2:004.774:331.108.26:004

А. В. Козлов

WEB-ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА ДАННЫХ СОТРУДНИКОВ ИТ-КОМПАНИИ

Описывается функционирование и работа приложения, позволяющего отслеживать карьерный рост сотрудников ИТ-компаний. Освещены вопросы удовлетворения специфичных для данной бизнес-модели требований заказчиков. Приложение разработано с использованием технологии React для клиентской части и Node.js для серверной, а также с использованием REST API, CSS, Material-UI и документо-ориентированной базы данных MongoDB.

Разработано web-приложение для учета данных о сотрудниках и событиях, произошедших в их карьере за время нахождения в ИТ-компаниях, с возможностью генерации отчета в виде PDF-файла. Актуальность данного проекта обусловлена заинтересованностью ИТ-компаний в контроле процесса развития сотрудника в удобном и простом виде. Приложение дает возможность формировать PDF-файл, где отображены события, произошедшие в карьере сотрудника, уровень навыков, которыми он располагает, список всех сотрудников фирмы и возможность просмотра их данных.

Приложение разработано с использованием технологии React для клиентской части и Node.js [1] для серверной, а также REST API, CSS, Material-UI и MongoDB. С помощью MongoDB реализовано хранение всех данных системы. В отличие от реляционных баз данных MongoDB предлагает документо-ориентированную модель данных, благодаря чему работает быстрее, обладает лучшей масштабируемостью, ее легче использовать [2].

Одной из главных особенностей React является использование JSX, который максимально приближен к HTML и компилируется в JavaScript [3]. С помощью Virtual DOM можно добиваться высокой производительности приложения. Также можно создавать изоморфные приложения. Созданные компоненты в технологии React могут быть с легкостью использованы и изменены заново в других проектах. Для безопасной передачи информации между клиентом и сервером используется JWT web-токен, который представляет собой зашифрованный формат упаковки данных [4].

Для входа в систему пользователю необходимо ввести логин и пароль, которые были установлены администратором (рисунок 1). При вводе логина и пароля система сравнит их с записями в базе данных.