Санкт-Петербургский государственный университет
Институт наук о Земле
Кафедра картографии и геоинформатики

МУХАМЕДЗЯНОВ Марк Русланович

Выпускная квалификационная работа

Разработка веб-сервисов для публикации пространственных данных на основе технологии WebRTC

Основная образовательная программа бакалавриата
«Картография и геоинформатика»
Профиль «Геоинформатика»

Научный руководитель:
д.т.н., доцент ПАНИДИ Евгений Александрович
Рецензент:
ведущий инженер, ООО «Мосинжпроект»
ДЕМЧЕНКО Евгений Викторович

Санкт-Петербург
2018

Содержание

Введение

В наше время наблюдается активная информатизацию и цифровизацию во всех сферах жизни. В соответствии с такой тенденцией стремительными темпами растет и число информационных проектов, программного обеспечения, инструментов разработки программного обеспечения, языков программирования и инфраструктурных технологий, растет и количество информации, которой обмениваются пользователи. Геоинформационные технологии также не стоят на месте. Появляется огромное количество пространственных данных которые требуется каким-то образом хранить, обрабатывать и налаживать обмен этой информацией между пользователями и производителями. Вслед за этим растет число геоинформационных систем, веб-картографических сервисов и других сервисов, решающих задачи, связанные с пространственными данными.

Ключевая роль в обмене пространственными данными и вообще данными в наши дни принадлежит сети Интернет.

Современная цифровая инфраструктура пространственных данных, в большинстве случаев, подразумевает централизованное хранение данных на серверах в базах данных или других хранилищах информации. Такой метод хранения и обработки обладает своими преимуществами и недостаткам. К преимуществам данного метода можно отнести единую точку доступа к ресурсам, централизованный контроль доступа к данным, а так же гибкость в создании резервных копий баз данных и реализации других методов сохранности данных.

В последнее время количество пространственной информации росло по экспоненциальному закону всвязи с развитием систем дистанционного зондирования Земли, включающим в себя спутниковые съемочные комплексы, атмосферные методы исследования (аэрофотосъемка), а так же развитием глобальной сети Интернет и сервисов обмена и обработки пространственных данных. Этот рост показал несколько недостатков централизованного хранения данных:

- увеличение расходов на поддержание серверов для обработки и хранения данных
- снижение номинальной скорости обмена информацией между сервером и клиентами за счет непосредственно роста количества информации, подлежащей обмену
- уязвимости сервера и хранилища данных перед хакерскими атаками
- увеличение нагрузки на канал передачи данных при росте числа пользователей сервиса
- Использование специализированной физической серверной инфраструктурой(датацентры)

В последние 5 лет выросла популярность самой децентрализации сети Интернет и идеи распределенного хранения и обработки данных. Появились технологии так называемых облачных вычислений, а также “туманных” вычислений (fog computing). Также стали популярными blockchain технологии и связанные с ними продукты. Однако сферу геоинформатики и пространственных данных всеобщее веяние децентрализации, особенно в веб-среде, почти не коснулось.

В данной работе рассматривается разработка и исследование потенциала архитектуры децентрализованного хранения, публикации и обработки пространственных данных в сети Интернет с помощью современного набора технологий Web 2.0. Предварительно обозреваются современные централизованные методы публикации и хранения пространственных данных. Приводится описание использованного набора технологий, использованных при разработке вышеупомянутой распределенной архитектуры. В итоге описывается попытка реализации одного из предложенных вариантов архитектур на языке JavaScript с помощью технологии WebRTC.

Проблематика

Перед выполнением этой работы были обозначены следующие цели и задачи:

- Рассмотреть все современные способы публикации, обработки и хранения пространственных данных
- Исследовать все методы децентрализованного и распределенного хранения и обработки информации и их реализации
- Изучить современные веб-технологии и освоить инструментарии для разработки веб-приложений, а также их возможности, в частности изучить возможности технологии WebRTC в создании децентрализованных систем
- На основании исследованного и изученного материала попытаться разработать архитектуру распределенного хранения и публикации пространственных данных, сформировать теоретическую основу для последующей реализации
- Реализовать прототип системы разработанной архитектуры
- Рассмотреть и предложить дальнейшие пути развития и улучшения полученного прототипа системы

Одноранговые и многоранговые сети

Существуют две основные модели сетевых взаимодействий компьютерной техники. В зависимости от того как распределены функции между узлами участниками сети выделяют одноранговые и многоранговые сети.

Одноранговые сети

В одноранговых или пиринговых(от англ. peer-to-peer) сетях все агенты общения равноправны и поэтому не имеется преимуществ у кого-либо из участников. Такие сети и являются основой всех децентрализованных архитектур сетевого обмена информацией. Компьютеры-участники этих сетей могут одновременно выступать клиентами и выполнять запросы к другим узлам сети и могут выполнять функции сервера, получая запросы от других участников сети, обрабатывая эти запросы и генерируя ответы. Структура такой сети показана на рис.1.

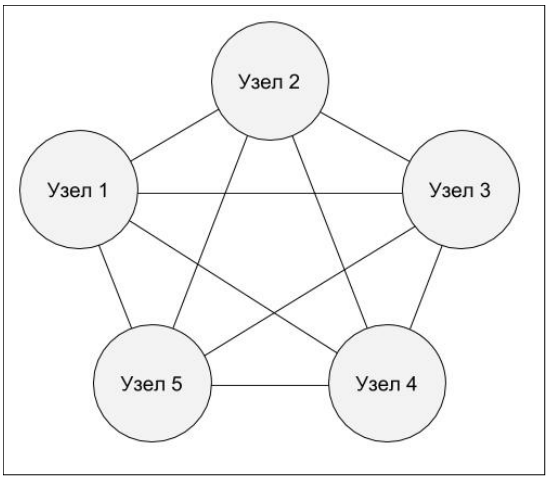

Рис 1. Структура одноранговой сети

Многоранговые сети

В отличии от одноранговых сетей, многоранговые подразумевают в своей структуре одного или несколько узлов сети с правами, большими, чем у остальных участников коммуникации. Самой распространенной многоранговой сетевой архитектурой является клиент-серверная архитектура. Она включает в себя два ранга участников общения, ранг клиент и ранг сервера, при этом ранг

сервера является главенствующим в сети. Эта модель – основа для централизованного обмена и хранения информацией, и самая распространенная сетевая архитектура в современной сети Интернет. Структура этой сети показана на рис. 2. Из рисунка видно, что “Узел 1” является сервером и к нему обращаются с запросами все клиенты сети. Также видно, что остальные узлы друг с другом не общаются напрямую, у них не подразумевается такая возможность. Все межклиентские взаимодействия происходят либо через посредничество сервера, либо не происходят вообще. При этом при выходе из строя сервера, по сути, выходит из строя вся сеть и узлы-клиенты абсолютно теряют возможность в получении услуги, предоставляемой узлом-сервером.

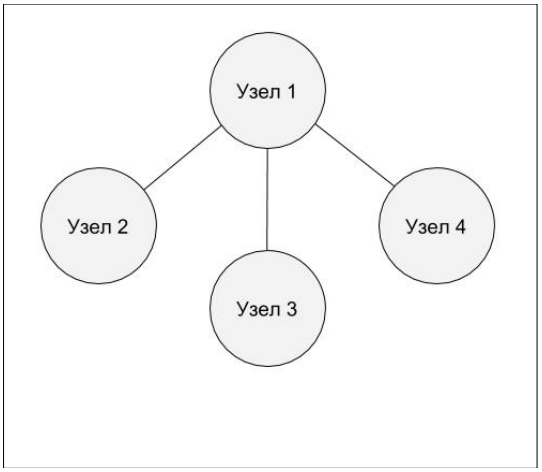

Рис 2. Структура многоранговой сети

Резюмируя вышесказанное, очевидными преимуществами обладает одноранговая сеть в условиях большого количества данных и пользователей, снижая удельную нагрузку на узел.

Существующие распределенные методы хранения, обработки и обмена информацией

Несмотря на главенствующее положение клиент-серверной(многоранговой) архитектуры в современной сети Интернет существуют децентрализованные модели коммуникаций, в которых нет единого сервера или другого авторитарного агента, ведущего надзор за сетью. К таким сетям относятся Torrent-трекеры, blockchain-сети, теневой интернет. Рассмотрим подробнее один из методов.

Протокол bittorrent

Протокол BitTorrent разработан специально для обмена большими файлами через сеть Интернет. Представляет из себя реализацию архитектуры одноранговой сети. Типичная сеть BitTorrent состоит из множества клиентов, которые будут осуществлять обмен файлами и так называемого торрент-трекера – сервера, который осуществляет отслеживание за текущими торрент-раздачами файлов и предоставляет пользователям сети информацию о других пользователях и файлах, которые можно загрузить.

Структура обмена файлами происходит по следующему механизму:

1. Пользователь 1, который хочет опубликовать файл, с помощью специального программного обеспечения создает torrent-файл, содержащий мета-информацию о публикуемом ресурсе, а также данные содержащие сетевые параметры подключения к Пользователю 1
2. Пользователь 1 выкладывает созданный файл на специальной платформе – торрент-трекере или создает раздачу и обычно сопровождает данную раздачу семантическим описанием
3. Пользователь 2 находит на торрент-трекере раздачу Пользователя 1, скачивает торрент-файл и открывает его в специальном программном обеспечении, начинается скачивание непосредственно файла напрямую с компьютера Пользователя 2

4. Пользователь 3 находит на торрент-трекере раздачу Пользователя 1, скачивает торрент-файл и открывает его в специальном программном обеспечении, начинается скачивание файла кусками и с компьютера Пользователя 1 и с компьютера Пользователя 2

Файл Пользователя 1 таким образом копируется или реплицируется на клиентах, запросивших этот файл и далее становится возможным скачивание этого файла со всех узлов этой сети, запросивших и скачавших хотя бы небольшой фрагмент данного файла. Преимущества очевидны – снижается нагрузка на канал между пользователями, скорость обмена информацией возрастает.

Обзор существующих методов публикации пространственных данных

Почти вся инфраструктура современного интернета представляет собой совокупность серверов и клиентов. Данные (не только пространственные) хранятся и обрабатываются централизованно на серверах.
В настоящее время централизованная Веб публикация пространственных данных производится в сети Интернет с помощью так называемых Веб-картографических сервисов. Эти сервисы и программные комплексы реализуют стандарты Open Geospatial Consortium – международной организации, занимающейся разработкой стандартов в области пространственных данных и геоинформатики.

Основные реализации стандартов OGC

Рассмотрим две основные реализации стандартов OGC о веб-сервисах.

GeoServer

GeoServer это программный продукт, написанный на языке программирования Java и реализующий стандарты Open Geospatial Consortium. GeoServer представляет собой веб-сервер, обеспечивающий стандартным клиентам, таким, как веб-браузеры и настольные ГИС, доступ к хранящимся в различных форматах картам и данным. Geoserver позволяет хранить пространственные данные почти в любом существую- щем формате, и пользователям не требуется ничего знать о ГИС-данных. 4 GeoServer — это реализация стандартов Open Geospatial Consortium (OGC) WFS (Web Feature Service) и WCS (Web Coverage Service), а также высокопроизводитель- ная сертифицированная реализация WMS (Web Map Service). 1.1.1 Возможности • Публикация данных из различных источников: Векторные форматы: Shape-файлы, внешний WFS,PostGIS, ArcSDE, DB2, Oracle Spatial, MySQL, SQL Server Растровые форматы: GeoTIFF, JPG и PNG (с файлами-заголовками), пирами- ды

изображений, форматы GDAL, Image Mosaic, Oracle GeoRaster. • Данные предоставляются пользователю в виде изображений через быстрый и безопасный протокол WMS: Поскольку информация передается в виде изображений, векторизованные данные защищены и находятся в полной безопасности. Единственный способ украсть эти данные — это выполнить их повторную векторизацию. Внешний вид каждого слоя карты может быть настроен при помощи дескрипторов стандарта SLD, позволяющего раскрашивать и подписывать объекты карты. Комбинируя эти правила с фильтрами OGC, можно реализовать зависимость символики от масштаба, позволяющую отображать более подробную карту при увеличении масштаба пользователем. Также реализованы управление размещением подписей, группировка и приоритеты.

- Полноценные векторные данные можно передавать клиенту по протоколу WFS: Клиент WFS может загрузить векторные данные и выполнять их отображение, пространственный анализ и другие операции. Авторизованный пользователь может также изменять данные и отсылать их обратно на сервер, обновляя хранящиеся там данные через протокол WFS. Данные могут передаваться в (сжатом) формате GML, равно как и в других стандартизированных форматах, например, в виде shape-файлов или JSON. 5
- Значения растоpовых данных можно передавать клиенту по протоколу WCS: Клиент ГИС может запросить актуальные растровые данные для простран- ственного анализа. Это даёт пользователю возможность создавать приложения для моделирования процесса, описываемого пользовательскими данными.
- Перепроецирование в реальном времени: GeoServer поддерживает большую часть проекций из базы данных организа- ции European Petroleum Survey Group(EPSG) и может при необходимости вы- полнять перепроецирование в любую из них, давая возможность клиентам с ограниченной поддержкой проекций перекладывать эту работу на сервер.
- Кэширование тайлов WMS GeoWebCache — тайловый клиент WMS. Он работает как прокси-сервер между клиентом и сервером карт, кэшируя запрашиваемые тайлы, предотвра- щая избыточные запросы и экономя

значительную часть времени, затрачивае- мого на обработку. GeoWebCache интегрирован в GeoServer.

MapServer

MapServer — написанная на языке программирования C система отрисовки географических данных с открытым исходным кодом. Помимо просмотра пространственных данных, MapServer позволяет создавать растровые географические карты, то есть карты, ссылающиеся на веб-контент. Например, веб-сайт Recreation Compass Департамента природных ресурсов штата Миннесота предоставляет пользователям более 10000 веб-страниц, отчётов и карт через единый интерфейс. Это же приложение служит картографическим движком для других частей сайта, предоставляя пространственное содержимое там, где это требуется. MapServer изначально был разработан в рамках проекта ForNet Университета Миннесоты (UMN) в сотрудничестве с NASA и Департаментом природных ресурсов Миннесоты (MNDNR). Позднее он размещался на TerraSIP, спонсируемом NASA совместном проекте UMN и консорциума по управлению земельными ресурсами. В настоящее время MapServer — проект OSGeo, поддерживаемый растущим (приближающимся к 20) числом разработчиков со всего мира. Он поддерживается группой разнообразных организаций, спонсирующих улучшения и поддержку. Управление разработкой в рамках OSGeo осуществляется Комитетом по управлению проектом MapServer, состоящим из разработчиков и контрибьюторов.

Возможности MapServer

- Развитые средства картографического отображения Отрисовка объектов и выполнение приложений, зависящие от масштаба Подписывание объектов, включая обработку случаев наложения подписей Поддержка шрифтов TrueType для подписей и условных знаков Автоматическое создание элементов карты (масштабной линейки, обзорной карты и легенды) Тематическое картографирование с выделением классов на основе логических и регулярных выражений Поддержка подключаемых рендереров с драйверами для AGG,

Cairo, GD, OpenGL и др. Специальные средства для генерации тайловых изображений

- Расширенная поддержка пространственных запросов Идентификация пространственных объектов с помощью атрибутов, точки, ограничивающего прямоугольника или геометрии в рамках одного или не- скольких слоёв Растровые запросы Полностью настраиваемый вывод на основе шаблонов 7
- Поддержка распространенных средств программирования расширений CGI/ FastCGI PHP, Python, Perl, Ruby, Java и .NET
- Кроссплатформенность MapServer запускается на таких операционных системах как Windows, Linux и Mac OS X
- Множество форматов растровых и векторных данных Как и Geoserver, Mapserver позволяет работать с большинством векторных и растро- вых форматов через GDAL и OGR
- Перепроецирование в реальном времени

Мы видим, что оба программных продукта имеют колоссальные возможности не только в публикации, но и в обработке пространственных данных, и, по сути, являются серверными геоинформационными системами, также данные решения являются программным обеспечением с открытым исходным кодом. Главным недостатком, помимо того, что это средства реализуют централизованный метод обмена ПД, является их немобильность и сложность развертывания — чтобы запустить и настроить данные решения, требуется выполнить набор нетривиальных для рядового пользователя-географа действий, включая настройку виртуального выделенного или физического сервера.

Обзор технологии WebRTC и её возможностей

WebRTC (Web RealTime Communication) – технология одноранговой коммуникации реального времени между веб-браузерами. Одноранговость выражается в отсутствии авторитарного сервера, как в классических клиент-серверных моделях сетевого обмена данными. Представлена публике в мае 2011 года и с тех пор продолжает развиваться. Эта технология включает в себя некоторое множество стандартов, направленных, в первую очередь, разработчикам браузеров. Данные стандарты разработанны специальной группой консорциума W3C (World Wide Web Consortium) при поддержке организации IETF (Internet Engineering Task Force). С помощью WebRTC становятся возможными передача медиа-содержимого в реальном времени прямо в веб-браузере. Технология позволяет разработать приложения для видео и аудио звонков через браузер.

В данный момент поддерживается сообществом на основаниях открытого программного обеспечения. WebRTC представляет возможность разработчикам Веб-приложений специальный интерфейс программирования (API), с помощью которого можно наладить обмен в реальном времени между двумя браузерами напрямую, минуя сервера посредники за некоторыми исключениям, которые будут рассмотрены ниже в работе. Интерфейс программирования приложений WebRTC реализован на языке JavaScript. В настоящее время поддерживается почти всеми современными веб-браузерами: Google Chrome, Mozila Firefox, Opera, Safari и др.

В настоящее время WebRTC используется, в основном для:

- Онлайновых трансляций
- Маркетинга реального времени
- Аудио и видео звонков
- Онлайн образования(вебинары, онлайн подкасты)
- Онлайн игр
- Наблюдения за каким-либо объектом с помощью медиа-оборудования, подключенного к компьютеру

Общая схема WebRTC

WebRTC позволяет настроить одноранговое соединение между двумя веб-браузерами с помощью WebRTC API. При этом данный API является низкоуровневым, т.е позволяет только осуществлять обмен сырыми пакетами данных. Контроль за доставкой пакетов, потерей данных, обработкой разрыва соединения или обхода технологии адресации NAT ложится на плечи разработчика. Базовая схема потока информации при использовании технологии WebRTC можно увидеть на рис.1 ниже.

Рис.1 Общая схема работы WebRTC

На схеме показано два уровня абстракции, один – уровень приложения, включает всё то, что разрабатывается разработчиками непосредственно веб-приложений для браузера. Второй уровень – уровень браузера, находится под контролем разработчиков браузеров. Уровень браузера разный для каждой реализации.

Описание типичного установления соединения с помощью WebRTC с точки зрения разработки веб-приложения

1. Происходит инициализация всех необходимых для работы веб-страницы JavaScript объектов и контекстов. Вызов WebRTC API метода createOffer(), возвращающим SDP (англ. Session Description Protocol) пакет, содержащий информацию о типе информации, которую разработчик хочет начать пересылать с помощью WebRTC. Эта информация может быть обычными данными, видео или аудиоконтентом и др.
2. Начинается процесс, который в стандарте WebRTC называется Signaling или оповещение адресата о предложении начать передавать данные. Происходит передача пакета SDP другому браузера каким-либо произвольным методом. Стандарт WebRTC, который выпускается W3C не возлагает никаких ограничений на метод передачи этого пакета. Разработчик должен сам решить как это сделать, используя свои сервера или как-то по другому. Сервера, которые обеспечивают Signaling называются Signaling серверами.
3. Получение SDP пакета браузером-адресатом от Signaling сервера. Полученный пакет нужно передать непосредственно WebRTC с помощью функции setRemoteDescription(). Теперь браузер знает о параметрах подключения WebRTC с первого браузера. Далее требуется сделать аналогичный SDP ответ для первого браузера. Это делается с помощью метода createAnswer(), который возвращает такой же текстовый SDP пакет, но уже для второго браузера, учтя данные из SDP пакета от первого браузера.
4. Передача второго SDP пакета с помощью произвольного метода передачи.
5. Получение SDP пакета первым браузером. Этот пакет следует передать WebRTC с помощью уже знакомого метода setRemoteDescription(). В результате проделанных 5 шагов браузеры обладают минимальной информацией друг о друге.
6. Далее происходит изучение сетевого подключения браузером, независимо от действий разработчика. На этом этапе происходит определение наличия NAT, между двумя браузерами и других параметров сети, которые будут важны

при последующем установлении соединения. Для определения наличия NAT используются так называемые STUN (от англ. Session Traversal Utilities for NAT) сервера. С помощью них происходит зондирование подключения в две стороны и сравнивание портов и IP-адресов, которые используются компьютерами-хостами браузеров при создании соединения.

7. Периодически WebRTC проверяет сеть о наличии изменений в соединении и создает на обоих браузерах так называмый пакет SIP (англ. Session Initiation Protocol), которй разработчику требуется передавать через свои Signaling сервера.
8. Далее происходит непосредственное peer-to-peer подключение по WebRTC в случае отсутствия NAT между двумя браузерами. Если же NAT имеется, то требуется настраивать так называемый TURN (Traversal Using Relay NAT) сервер, который, по сути, работает как обычный прокси-сервер и пропускает через себя трафик, создаваемый участниками соединения

Видно, что для установления прямого соединения между пользователями проходит достаточно большое количество шагов, которые требуется выполнить разработчику приложения.

Разработка архитектуры децентрализованного хранения и публикации пространственных данных

Применительно к пространственным данным в этой главе предлагается две архитектуры децентрализованного хранения, публикации и обработки пространственной информации. Учитывая специфику выбранной темы, предложенные архитектуры требовалось затачивать под среду Веб-приложений и выбранную для исследования технологию WebRTC. Ниже рассмотрены теоретические разработки предложенных архитектур.

Torrent-подобный метод

Как было рассмотрено выше, протокол bittorrent обладает большим потенциалом для разработки архитектуры распределенного хранения именно пространственных данных всвязи со своим уклоном под обмен большими файлами между пользователями. Общая схема обмена в такой архитектуре может быть реализована методом, сильно похожим на протокол bittorrent. Так как обмен пространственными данными и их отображение будет осуществляться в браузере, то и непосредственно роль torrent-клиента будет отведена веб-браузеру. Визуализировать пространственную информацию предполагается через использование веб-картографических библиотек, таких как Leaflet.JS, CesiumJS или OpenLayers. Исходя из того что, torrent-сеть подразумевает существование и работоспособность хотя бы двух элементов сети для своего существования, требуется обеспечить чтобы хотя бы одна копия единицы публикуемых данных была всегда в доступе. Беря во внимание то, что веб-браузер и компьютер-хост не могут обеспечить круглосуточной доступности файла, такая архитектура не сможет обойтись без сервера-трекера, который будет отслеживать всех текущих клиентов и проводить индексацию данных.

Общий алгоритм можно описать следующим образом:

1. Пользователь 1 заходит на страницу веб-приложения которое, в свою очередь подключается по протоколу HTTP к серверу-трекеру и получает необходимую мета-информацию о запрашиваемом ресурсе и начинает загрузку картографической информации с сервера-трекера и отображает её в странице веб-браузера. Также он оповещает сервер-трекер о фрагменте загруженных пространственных данных и сообщает о своих параметрах однорангового подключения по технологии WebRTC.
2. Пользователь 2 заходит на страницу веб-приложения и подключается по протоколу HTTP к серверу-трекеру за мета-информацией. Пусть Пользователь 2 запрашивает сервер-трекер такой же фрагмент пространственных данных, который в предыдущем шаге был загружен Пользователем 1. Сервер-трекер сообщает о наличии этого фрагмента у другого пользователя и сообщает ему параметры подключения по WebRTC
3. Далее Пользователь 2 может поступить двумя способами – запросить искомый фрагмент информации у сервера-трекера по протоколу HTTP или запросить этот фрагмент у Пользователя 1, используя возможности технологии WebRTC.
4. В случае выбора второго варианта начинается стадия соединения Пользователя 1 и Пользователя 2. В случае успеха искомый фрагмент пространственных данных скачивается с Пользователя 2 тем самым снимая нагрузку непосредственного обмена с сервера-трекера
5. В случае неуспеха соединения по WebRTC фрагмент загружается “классическим методом” – с сервера-трекера по протоколу HTTP.
6. Пользователь 2 оповещает сервер-трекер о наличии скачанного фрагмента пространственных данных

Наглядно данный алгоритм представлен на рисунке 3.

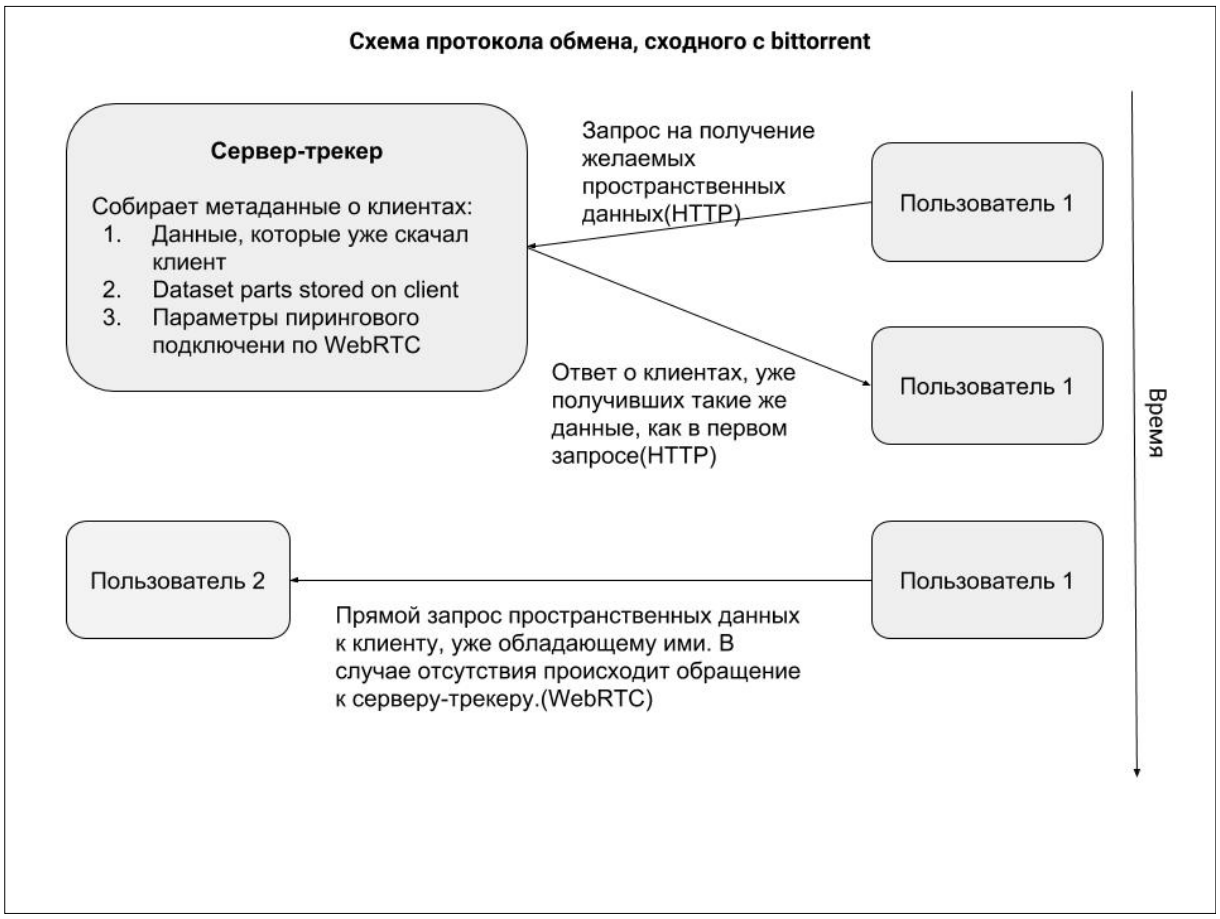

Рис 3. Схема торрент-подобного алгоритма.

Стоит упомянуть несколько важных особенностей данного алгоритма. При обрыве соединения или каких-либо других проблем связанных с подключением к серверу-трекеру у одного из пользователей возникает потребность в самостоятельном решении сервером-трекером об удалении записи о наличествовании каких-либо данных у отсоединенного пользователя. Эту проблему можно решить введя периодическое напоминание серверу-трекеру о своей жизнеспособности клиентами-получателями данных. Если в течении установленного интервала напоминание не произошло, то происходит удаление этого пользователя из индекса слежения.

Также можно отметить, что данных алгоритм не достигает полной децентрализации и распределенности в хранении данных. Всё еще существует

некий сервер-посредник, который, хоть и не в полной мере, но обладает правами, большими, чем у клиентов-потребителей информации. Вероятно такую архитектуру можно назвать гибридной, так как у неё присутствуют и признаки децентрализации и признаки многоранговых сетей. На начальном этапе данную архитектуру не отличить от классической модели “клиент-сервер”

Однако такой метод при большом количестве данных и клиентов позволит существенно снизить нагрузку на инфраструктуру, в следующих метриках

- *Совокупной пропускной способности* всех соединений топологии данной сети, так как удельный обмен данной инфраструктуры будет значительно ниже чем у клиент-серверной реализации
- *Удельной скорости обмена информации* в бит/сек/кол-во узлов

Также можно собирать данные о географическом местоположении клиентов, что позволяет сделать современный WebAPI. При наличии нескольких клиентов с одними и теми же данными можно производить ранжирование по географическом положении при выборе клиента, с которого будет скачиваться запрошенный кусок данных по географическому признаку. Т.е отдавать предпочтение тому клиенту, который будет географически ближе. Это позволит ещё увеличить скорость обмена информацией.

Резюмируя описанную архитектуру можно говорить о географически распределённых пространственных данных, хоть и не полностью. Такая архитектура может быть использована в веб-картографических сервисах, сходных с такими популярными продуктами как Yandex Maps, Google Maps, Bing Maps и других. Так как получается существенный выигрыш в скорости обмена данными за счет большого количества пользователей, опять же распределенных географически.

Попытка реализации архитектуры распределенного хранения и публикации пространственных данных

В данной главе описывается попытка реализации torrent-подобной архитектуры, описанной в предыдущей главе.

Выбор технологий, с помощью которых будет строиться архитектура

Как было сказано в предыдущей главе структура такой сети строится из одного или нескольких серверов-трекеров и нескольких клиентов. В качестве программных платформ для разработки сервера-трекера были выбрана платформа node.js. Такой выбор обусловлен тем, что программирование осуществляется на языке JavaScript, как и программирование непосредственно приложения для браузера. Для программирования приложения на стороне браузера в качестве выбранного набора технологий был выбран текущий современный стек технологий Web 2.0: HTML5, CSS3 и язык программирования JavaScript.

Итак, качестве готовых библиотек и фреймворков, которые будут существенно упрощать разработку были выбраны:
На серверной стороне:

- **Express.js**. Библиотека с открытым исходным кодом на языке JavaScript, которая упрощает создание HTTP сервера. Предназначена только для программной платформа node.js
- **Node.js**. Программная платформа для создания приложений обшего назначения на языке JavaScript. Этот продукт будет основой для создания сервера-трекера.

На клиентской стороне:

- **WebRTC API**. Данный интерфейс не является библиотекой или фреймворком, но он является центральным модулем, который будет осуществлять децентрализованный обмен пространственными данными.

- **PeerJS**. Библиотека – высокоуровневая абстракция над WebRTC. С помощью нее существенно облегчается низкоуровневая работа непосредственно с созданием соединения.
- **Axios**. Библиотека-абстракция протокола HTTP, позволяющая делать относительно высокоуровневые HTTP запросы из браузера и на стороне сервера на Node.js (нас интересует реализация библиотеки для браузера). Написана на языке JavaScript. Требуется нам для упрощения работы с HTTP на стороне браузера.
- **Leaflet.js**. Библиотека, которая позволяет отображать пространственные данные в браузере, как растровые, так и векторные.

Socket io

Также будут применена технология WebSockets в качестве транспортного протокола для SDP пакетов на сервере, выполняющем роль Signaling сервера. Эта технология будет применяться в виде библиотеки Socket io и на клиенте и на сервере. Выбор WebSocket обусловлен тем, что данный протокол служит для двухсторонней передачи между браузером и сервером в реальном времени. Т.е в отличии от классической модели “запрос-ответ” WebSockets подразумевает прямое живое подключение и соответственно имеется возможность передавать данные в любое время как клиентом серверу, так и сервером клиенту. Такая модель отлично ложится на описанный алгоритм создания соединения с помощью WebRTC. Также одним из аргументов в пользу WebSockets можно назвать то, что смежная ей технология – LongPolling может давать сбои на длительных периодах работы из-за обрыва соединения.

Также для упрощения процесса разработки был использован редактор кода WebStorm от компании JetBrains. Данный редактор обладает мощными функциями в рефакторинге, редактуре и обработки кодовой базы проекта. Также он позволяет разработчику легко работать с системой контроля версий Git.

Ссылки на все указанные библиотеки и технологии, которые были использованы при разработке, их документацию и описания будут даны в Приложении 1 в конце документа.

Следует напомнить, что одной из предыдущих глав был описан общий алгоритм подключения двух браузеров. На одном из шагов была видна потребность в так называемом Signaling сервере, который будет посредником в установлении соединения. Исходя из вышеописанной модели архитектуры таким сервером можно назначить сервер-трекер. Он будет выполнять функции непосредственно сервера-трекера и следить за файловым распределением в сети, также он будет выполнять функции Signaling сервера.

Разработка программной части инфраструктуры

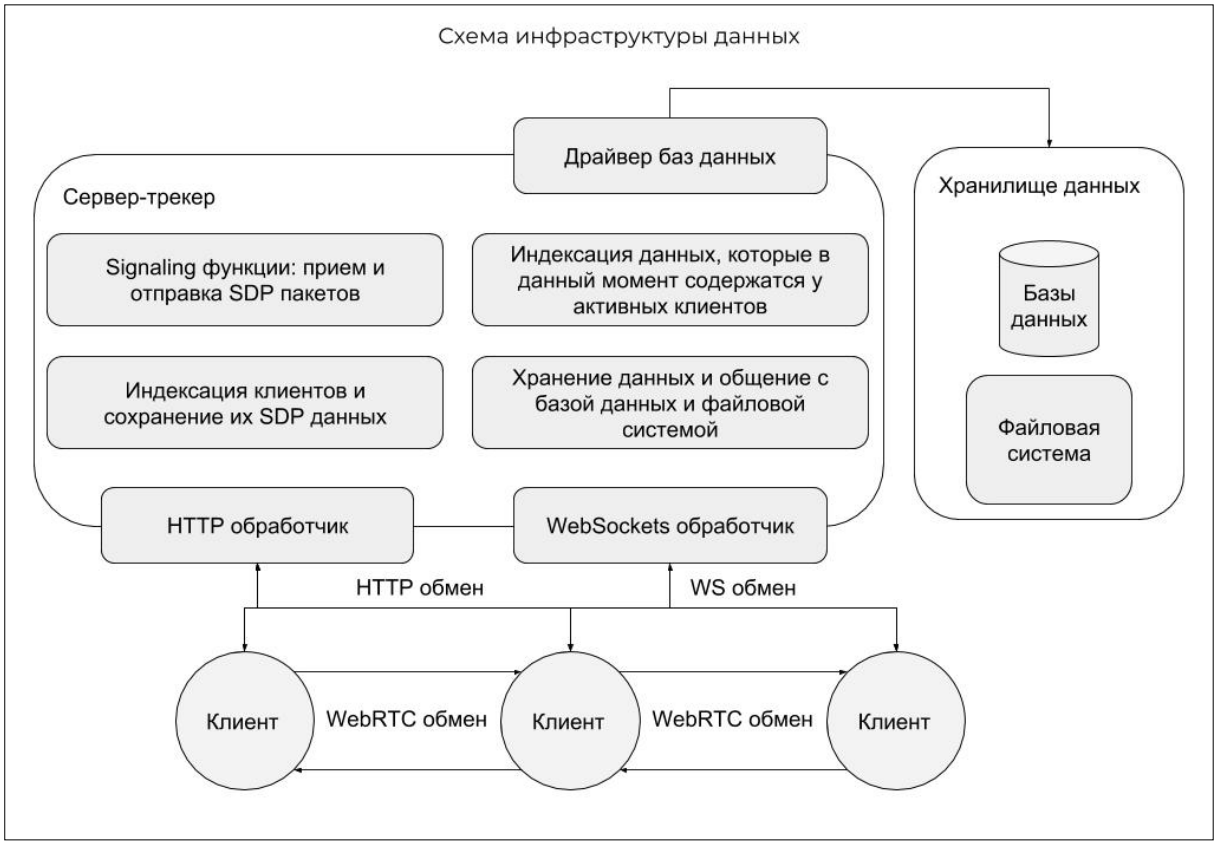

Рис 4. Схема инфраструктуры проекта

После всех необходимых теоретических изысканий была начата разработка прототипа архитектуры описанной в предыдущей главе. На рисунке 4 показана структура модулей проекта.

Первым делом была начата разработка сервера-трекера. Написаны обработчики HTTP запросов, которые этот сервер будет обрабатывать. В первую очередь требовалось реализовать именно функции Signaling сервера. С помощью библиотеки socket.io был написан модуль на сервера для общения с клиентами по технологии WebSockets для отправки SDP пакетов. Также написан модуль индексации всех клиентов и данных о их WebRTC соединениях. Эти два модуля тесно связаны друг с другом, так как именно они выполняют “инфраструктурную” функцию общения с клиентами. С программной точки зрения модуль индексирования клиентов можно представить как массив JavaScript объектов, содержащих поля, которые описывают клиента, включая данные о их WebRTC состоянии.

Следующим этапом происходила разработка клиентской или браузерной части инфраструктуры, написаны и налажены тестовые передачи данных по всем использованным протоколам общения: WebRTC, WebSockets и HTTP коммуникация. На этом этапе, фактически был налажен обмен между клиентом и сервером и можно было приступать непосредственно к реализации той распределенной архитектуры, которая была взята за основу.

Было принято решение сначала протестировать произвести торрент-подобный обмен на произвольных файлах (текстовых и растровых изображениях). Для этого была реализована следующая функциональность:

1. На сервере-трекере был написан модуль, который принимал запросы по HTTP на загрузку одного из нескольких произвольных файлов.
2. Написан модуль, который производит разбивку файлов на фрагменты по 64 килобайта каждый. Размер файла выбран произвольно – в дальнейшем можно настроить этот параметр. Далее модуль производит индексацию этих кусков и записывает это всё во временную память. Индексация производится путем присвоения каждому фрагменту уникального идентификатора UUID.

3. На клиенте написана функциональность, отвечающая за HTTP запросы к серверу-трекеру на получение файлов.
4. Написана функциональность, которая реализует поиск наличия файла у определенного клиента и методы, решающие вопрос о рекомендации клиенту загрузить недостающий фрагмент у другого пользователя по WebRTC.

Далее была произведена апробация реализованного функционала. Тестирование происходило следующим образом.

1. Открывалось окно браузера на компьютере и запускалось написаное веб-приложение.
2. Скачивался файл
3. Открывалось окно на втором компьютере в той же локальной сети, что и первое (чтобы избежать последствия работы NAT и в то же время протестировать WebRTC в боевых условиях)
4. Скачивался этот же файл и производилось тестирование трафика, анализировалось каким образом именно происходила загрузка. Желаемое поведение – браузер 2 скачивал файл из первого браузера.

Путем итеративного цикла “разработка-тестирование-отладка” были получены успешные результаты в обмене файлами с помощью технологии WebRTC.

Далее было принято решение попробовать произвести именно загрузку геоданных, используя WebRTC.

На сервере в качестве векторных пространственных данных был загружен GeoJSON файл с административными районами города Перми. В качестве растровых данных был использован GeoTIFF файл, содержащий информацию об индексе NDVI на произвольную территорию.

На следующем этапе появилась задача о разбивке этих данных на фрагменты, как было сделано с произвольными текстовыми файлами. С GeoTIFF файлом было всё довольно тривиально, файл разбивался на небольшие тайлы и

индексировался. С векторными данными ситуация может выглядеть посложнее. В отличие от растровых данных, векторные данные – дискретны и в своей структуре данные о каком-либо географическом объекте разбиты на фрагменты семантически и синтаксически, так как обычно векторные данные представлены в виде описания на каком-либо декларативном языке JSON, XML, SVG и др. Нужно было разработать алгоритм, который разбивает GeoJSON файл на меньшие GeoJSON файлы исходя из внутренней структуры файла. Этот алгоритм был разработан и работает следующим образом. Все точечные типы данных группируются в файлы по 10 штук и отделяются в свои GeoJSON файлы. С полигонами и линиями было решено поступить по-другому. Каждый полигон или каждый линейный объект выделялся в отдельный файл. Далее эти GeoJSON файлы индексировались. Таким образом была решена проблема фрагментации пространственных данных – векторных и растровых.

В клиентской части (в браузере) было решено сделать отображение пространственных данных. Это было сделано с помощью библиотеки Leaflet.js.

Далее был написан модуль, который скачивает пространственные данные по аналогии с модулем который скачивал произвольные данные. По мере загрузки частей какого-либо типа данных производилась отрисовка оных в браузере.

Таким образом был налажен обмен простейшими пространственными данными в браузере используя технологию WebRTC.

Все исходные файлы доступны в репозиториях по ссылкам, указанным в Приложении 1.

Пути развития

Несомненно, была произведена большая работа по теоретической разработке архитектуры, произведен анализ существующих решений распределенного хранения и обмена данными. Сделан прототип описанной архитектуры и получены некоторые результаты тестирования этого прототипа. В качестве путей развития разработки данного решения можно выделить несколько пунктов. Концептуально есть несколько путей развития:

- Разработать методику интеграции с существующими централизованными решениями по публикации, обмену и обработки пространственных данных.
- Разработать метод использования существующих популярных стандартов в области веб-картографирования смежно с предложенным решения
- Разработать систему менеджмента пространственными данными с помощью предложенной технологии
- Разработка стандарта обмена пространственными данными, распределенными географически.

К техническим пунктам можно отнести:

- Реализация обмена данными во всех возможных типах пространственных данных, векторных и растровых.
- Отладка и анализ всех крайних случаев, которые могут возникнуть при таком обмене данными
- Реализация инфраструктуры TURN и STUN серверов, которые будут обслуживать клиентов, которые спрятаны за технологией NAT

Заключение

В ходе выполнения этой работы было решено множество теоретических и практических задач. Получить большое количество знаний и умений, требуемых для решения поставленных изначально и возникающих в ходе выполнения этой работы задач. К теоретическим задачам можно отнести саму задачу разработки распределенной архитектуры хранения, обработки и публикации пространственных данных с помощью веб-технологий. Также требовалось более глубоко изучить устройство компьютерных сетей, язык программирования JavaScript и смежный с ним набор технологий, в частности WebRTC.

К практическим вызовам отнесём технические задачи, возникающие при непосредственном программировании элементов системы распределенной инфраструктуры пространственных данных.

В целом проделанная работа принесла большое количество опыта именно разработки сетевых систем и знаний, связанных с форматами пространственных данных. Расширено владение инструментарием веб-технологий. Изучено большое количество статей и книг по WebRTC, JavaScript, node.js и литературы, связанной с общей теорией компьютерных сетей. Также расширены знания в области распределенного хранения данных, в области дисциплины, занимающейся изучением современных инфраструктур пространственных данных.

Произведен анализ решения и изучены пути дальнейшего развития разработанной архитектуры и реализованного прототипа системы, использующую данную архитектуру.

Список литературы

Книги

1. Real-Time Communication with WebRTC. Сальваторе Лорето. O'Reilly Media. 2014г. 164с
2. Node.JS Web Development. Дэвид Хэрон. Packt Publishing. 2016г. 367с
3. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. Брендан Бёрнс. O’Reilly. 2018г. 149с

Интернет источники

Стандарты

1. Стандарт WebRTC. Ссылка: https://www.w3.org/TR/webrtc/
2. Стандарт IETF о peer-to-peer сетях. https://tools.ietf.org/html/rfc5694
3. Спецификация протокола BitTorrent. http://www.bittorrent.org/beps/bep_0003.html

Статьи из тематических ресурсов по исследованной теме

1. https://habr.com/company/Voximplant/blog/344794/
2. https://habr.com/company/flashphoner/blog/337670/
3. https://habr.com/post/163527/
4. https://medium.com/@jeremy.noring/webrtc-pro-tips-98f90acad007
5. https://medium.com/the-making-of-appear-in/what-kind-of-turn-server-is-being-used-d67dbfc2ff5d

Приложение 1

1. Библиотека Leaflet.js. https://leafletjs.com/
2. Библиотека Axios.js. https://github.com/axios/axios
3. Библиотека Peer.js. https://peerjs.com/
4. Фреймворк Express.js. http://expressjs.com/ru/
5. Платформа Node.JS. https://nodejs.org/en/