

Министерство образования и науки Российской Федерации
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

УДК 004.04, 004.82, 004.7
№ госрегистрации 114121750065



В.О. Никифоров
2015 г.

ОТЧЁТ
О ПРИКЛАДНЫХ НАУЧНЫХ ИССЛЕДОВАНИЯХ

Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)

по теме:

ТЕОРЕТИЧЕСКИЕ ИССЛЕДОВАНИЯ ПОСТАВЛЕННЫХ ПЕРЕД ПНИ ЗАДАЧ
(ОЧЕРЕДЬ 1)

(промежуточный)

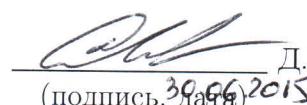
Шифр № 340745

Этап 2

ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы»

Соглашение о предоставлении субсидии от 24.11.2014 г. №14.575.21.0101

Руководитель проекта
к.т.н., доцент

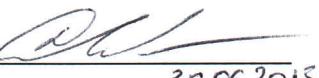

Д.И. Муромцев
(подпись, 30.06.2015)

Санкт-Петербург 2015

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель проекта:

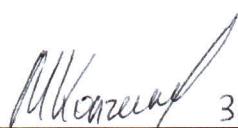
доцент,
канд. техн. наук.


(подпись, дата) 30.06.2015

Д.И.Муромцев
(введение, заключение)

Исполнители темы:

инженер


30.06.2015
(подпись, дата)

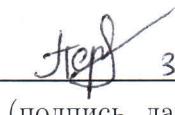
М.А. Колчин
(раздел 1, раздел 2)

ассистент,
канд. техн. наук


30.06.2015
(подпись, дата)

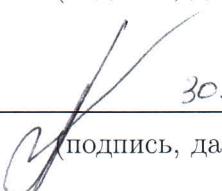
Р.Я. Лабковская
(раздел 3)

ассистент


30.06.2015
(подпись, дата)

О.В. Пархимович
(раздел 3)

профессор,
доктор техн. наук


30.06.2015
(подпись, дата)

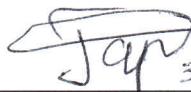
С.А. Арутамов
(раздел 1)

инженер


30.06.2015
(подпись, дата)

А.А. Андреев
(раздел 1, раздел 2)

инженер


30.06.2015
(подпись, дата)

Д.С. Гарайзуев
(раздел 2)

инженер


30.06.2015
(подпись, дата)

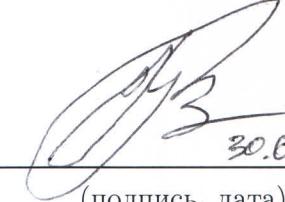
Н.В. Климов
(раздел 3)

инженер


30.06.2015
(подпись, дата)

И.А. Шилин
(раздел 1)

доцент,
канд. техн. наук


30.06.2013
(подпись, дата)

Д.А. Заколдаев
(раздел 1)

инженер


30.06.2015
(подпись, дата)

Д.О. Захаров
(раздел 2)

ведущий инженер,
канд. техн. наук


30.06.15
(подпись, дата)

И.А. Радченко
(раздел 2)

зав. лабораторией


30.06.15.
(подпись, дата)

О.А. Кураш
(раздел 2)

Нормоконтролер:


30.06.15
(подпись, дата)

В.В. Беззубик

РЕФЕРАТ

Отчет 131 с., 61 рис., 3 табл., 2 прил., 1 ч., 28 источника.

СЕМАНТИЧЕСКИЙ ИНТЕРНЕТ ВЕЩЕЙ, СЕМАНТИЧЕСКИЕ ТЕХНОЛОГИИ, ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ, ОНТОЛОГИИ, ОБРАБОТКА СЛОЖНЫХ СОБЫТИЙ, СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА, БЕСПРОВОДНЫЕ СЕНСОРНЫЕ СЕТИ, ОБРАБОТКА ПОТОКОВЫХ RDF ДАННЫХ, ИНТЕРНЕТ ВЕЩЕЙ, ВИЗУАЛИЗАЦИЯ ГЕТЕРОГЕННЫХ ДАННЫХ, ИНТЕГРАЦИЯ ГЕТЕРОГЕННЫХ ДАННЫХ, АНАЛИЗ ПОТОКОВЫХ ДАННЫХ, АНАЛИЗ ГЕТЕРОГЕННЫХ ДАННЫХ, ПУБЛИКАЦИЯ ПОТОКОВЫХ ДАННЫХ.

В отчете представлены результаты исследований, выполненных по этапу 2 ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)».

В данном отчете описаны результаты теоретических исследований процесса сбора, нормализации, анализа и визуализации гетерогенных данных распределенной сети электронных потребительских устройств.

Целью работ на данном этапу является разработка математических методов и алгоритмов направленных на решение следующих задач:

- а) агрегация больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба.
- б) нормализация и анализ больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things).

в) интерактивная визуализация больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data).

Кроме того, на данном этапе ПНИ целью работ так же является участие в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ (конференции, семинары, симпозиумы, выставки и т.п., в том числе, международные).

Для решения поставленных на данном этапе ПНИ задач были применены методы математического моделирования, методы онтологического инженеринга, методы логического вывода на знаниях, технологии построения веб-сервисов, технологии семантического веба, а также методы объектно-ориентированного программирования.

Результатами решения данных задач стали:

- а) методы и алгоритмы агрегации данных распределенной сети электронных потребительских устройств основанные на архитектурном подходе, который заключается в использовании драйверов (или адаптеров) для различных типов электронных потребительских устройств, отличающихся технологиями и протоколами передачи данных. В данном отчете рассматриваются методы и алгоритмы работы с устройствами поддерживающими протокол передачи данных CoAP (Constrained Application Protocol) или же проприетарный протокол передачи данных основанный на протоколе UDP (User Datagram Protocol). Новизна разработанных методов и алгоритмов заключается в: (а) применении протокола СоАР для электронных потребительских устройств с ограниченными вычислительными ресурсами (классы 0 и 1, в соответствии с RFC 7228¹) и (б) использование онтологических моделей и технологий семантического веба для аннотирования показаний (или же измерений) и характеристик устройств.
- б) методы и алгоритмы нормализации и анализа данных распределенной сети потребительских устройств основанные на применении высоконивневых онтологий для нормализации данных и методах анализа

¹См. <https://tools.ietf.org/html/rfc7228>

потоковых и статических данных на основе онтологического подхода. Новизна разработанных методов и алгоритмов заключается в: (а) разработке верхнеуровневых онтологий, которые дополняют существующие онтологии новыми сущностями и связями, и (б) применении методов анализа связанных данных в режиме "реального времени" с использованием логического вывода.

в) методы и алгоритмы визуализации данных распределенной сети электронных потребительских устройств основанные на интерактивной визуализации временных и пространственных данных в режиме "реального времени" и режиме временных исторических промежутков. Новизна разработанных методов и алгоритмов заключается в применении современных методов "нативной" визуализации, основанных на технологиях HTML5 и SVG, а так же использовании методов доступа данных на основе онтологий.

Разработанные методы и алгоритмы были закодированы и протестированы с использованием ПАС для моделирования распределенной сети электронных потребительских устройств, разработанного на предыдущем этапе ПНИ.

Результаты полученные на данном этапе прикладного научного исследования соответствуют Плану-графику работ и требованиям технического задания и будут использованы на последующих этапах ПНИ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	13
1 Разработка математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурированных и неструктурированных данных с применением приемов онтологического инжиниринга и технологий семантического веба	15
1.1 Назначение и состав математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурированных и неструктурированных данных	15
1.2 Метод агрегации данных с электронных потребительских устройств с ограниченными вычислительными ресурсами (ЭПУОВР), использующих прикладной протокол передачи данных CoAP (Constrained Application Protocol)	17
1.3 Алгоритм асинхронной передачи данных с ЭПУОВР, использующих протокол CoAP	21
1.4 Алгоритмы кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF (Resource Description Framework) и протокол CoAP	24
1.5 Метод агрегации данных с ЭПУОВР, использующих проприetaryный прикладной протокол передачи данных на основе протокола UDP (User Datagram Protocol)	32
1.6 Метод и алгоритм аннотирования данных с ЭПУОВР, не использующих модель данных RDF	35
1.7 Метод и алгоритмы хранения больших массивов данных распределенной сети электронных потребительских устройств	39
1.8 Результаты	44
2 Разработка математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурированных и неструктурированных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things)	45

2.1	Назначение и состав математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things)	45
2.2	Метод нормализации гетерогенных данных распределенной сети электронных потребительских устройств на основе верхнеуровневых онтологий	46
2.3	Алгоритм установления соответствия между предметно-ориентированными онтологиями и верхнеуровневыми онтологиями, используемыми для нормализации гетерогенных данных	53
2.4	Метод и алгоритм оценки производительности и корректности систем анализа связанных потоковых данных	57
2.5	Метод и алгоритмы анализа потоковых и статических данных распределенной сети электронных потребительских устройств	64
2.6	Метод обнаружения и генерации событий предметной области на основе анализа потоковых и статических данных	67
2.7	Результаты	71
3	Разработка математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data)	73
3.1	Назначение и состав математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data)	73
3.2	Метод и алгоритм доступа к статическим данным на основе семантических веб технологий и технологии связанных данных	74
3.3	Алгоритм доступа к историческим данными распределенной сети электронных потребительских устройств	78

3.4	Метод и алгоритм доступа к потоковым данным распределенной сети электронных потребительских устройств	80
3.5	Алгоритм визуализации потоковых и исторических данных . .	87
3.6	Результаты	94
ЗАКЛЮЧЕНИЕ		95
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ		99
Приложение А	Отчет о результатах исследования методов и алгоритмов	103
Приложение Б	Отчет об участии в мероприятиях	126

ОПРЕДЕЛЕНИЯ

В настоящем отчете о ПНИ применяют следующие термины с соответствующими определениями:

Интернет вещей (Internet of Things) — концепция сети однозначно идентифицируемых встраиваемых вычислительных устройств в рамках существующей инфраструктуры Интернета.

Технологии семантического веба (Semantic Web technologies) — технологии, которые предоставляют общий инструмент, позволяющий осуществлять обмен данными и их многократное использование между и за пределами программ, предприятий и сообществ.

Связанные данные (Linked Data) — методы публикации взаимосвязанных между собой наборов структурированных данных в рамках существующей инфраструктуры Интернета.

Онтология (Ontology) — попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью концептуальной схемы. Такая схема состоит из структур данных, содержащей все релевантные классы объектов, их связи и правила, принятые в этой области.

Логический вывод (Logical reasoning) — процесс рассуждения, в ходе которого осуществляется переход от некоторых исходных суждений (предпосылок) к новым суждениям - заключениям.

Гетерогенность — разнородность, наличие неодинаковых частей в структуре, в составе чего-либо.

Интерактивная визуализация данных — способ графического представления информации, позволяющий пользователю взаимодействовать с системой отображения информации и наблюдать ответную реакцию системы.

REST (Representational State Transfer) — метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP, CORBA и RPC.

Событие предметной области — какое-либо событие рассматриваемое в контексте конкретной предметной области. Например, превышение температуры теплоносителя при теплоснабжении жилого дома и т.п.

Система управления потоковыми данными — это программа для управления непрерывными потоками данных. Она аналогична системе управления базами данных (СУБД), которые, однако, предназначены для статических данных в обычных базах данных. СУПД также предлагает гибкую обработку запросов, так что информация может быть выражена с помощью запросов. Однако, в отличие от СУБД, в СУПД выполняет непрерывный запрос, который не только исполняется один раз, а исполняется постоянно. Таким образом, запрос непрерывно выполняется до тех пор, пока не будет явно удален.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

IoT — Internet of Things

WSN — Wireless Sensor Networks

ПО — программное обеспечение

URI — Unified Resource Identifier

UDP — User Datagram Protocol

HTTP — Hypertext Transfer Protocol

CoAP — Constrained Application Protocol

API — Application Programming Interface

ЭО ПАП — экспериментальный образец программно-аппаратной платформы

ПО ЭО ПАП — программное обеспечение экспериментального образца программно-аппаратной платформы

ПАС — программно-аппаратный стенд

ЭПУОВР — электронные потребительские устройства с ограниченными вычислительными ресурсами

SSN — Semantic Sensor Network Ontology

RDF — Resource Description Framework

OWL — Web Ontology Language

REST — Representational State Transfer

СУПД — Система управления потоковыми данными

Fuseki — Apache Jena Fuseki

ВВЕДЕНИЕ

В данном промежуточном отчете по 2-му этапу ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)» приведены результаты теоретических исследований на данном этапе ПНИ.

Общими целями выполнения данного прикладного научного исследования являются:

- а) создание комплекса научных/научно-технических решений в области разработки методов и алгоритмов, обеспечивающих повышение эффективности научных исследований посредством агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных распределенной сети электронных потребительских устройств (Internet of Things).
- б) получение значимых научных результатов в области разработки масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things).

Технологии Интернета вещей хоть и получили широкую известность, но все еще находятся на ранней стадии развития. Существует большое количество исследовательских и технических проблем, например такие как программная и аппаратная архитектура, стандартизация, безопасность и конфиденциальность. Научно-технические задачи решаемые на данном этапе ПНИ направлены на обеспечение синтаксической и семантической интеропе-

перабельности компонентов распределенной сети электронных потребительских устройств. А применение новых методов и инструментов технологий семантического веба определяют актуальность данной работы.

Целью работы на данном этапе ПНИ являются теоретические исследования направленные на решение следующих задач:

- а) разработка математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба.
- б) разработка математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things).
- в) разработка математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data).

В Приложении А данного отчета о ПНИ приводятся результаты исследования разработанных методов и алгоритмов.

Кроме того, так же целью работы является участии в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ (конференции, семинары, симпозиумы, выставки и т.п., в том числе, международные). В Приложении Б представлены соответствующие отчетные и справочные материалы.

1 Разработка математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба

1.1 Назначение и состав математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных

Сбор и хранение данных для последующей их обработки играет важную роль в области Интернета вещей так как на этом уровне возникают основные проблемы вызванные многообразием существующих типов электронных потребительских устройств и технологий, которые используются ими для обмена данными друг с другом и центрами обработки данных. В рамках данной работы поставлена задача разработать набор методов и алгоритмов решающих данную проблему поддержки различных типов электронных потребительских устройств.

На основе результатов аналитического обзора современной научно-технической литературы, а также анализа существующих приемов онтологического инжиниринга и технологий семантического веба для агрегации больших массивов гетерогенных данных, выполненных на 1-м этапе данного ПНИ [1], принято решение о разработке двух методов агрегации данных с электронными потребительскими устройствами. Первый метод заключается в том что такие устройства реализуют поддержку протокола СоАР и модель данных RDF для передачи данных. А второй метод заключается в поддержке агрегации данных с устройствами, которые не поддерживают протокол СоАР и модель данных RDF, но используют какой-либо другой протокол передачи данных, основанный на протоколе UDP. Данные методы позволяют реализовать подход к организации архитектуры ЭО ПО ПАП, использующий внешний подмодули (или драйвера), которые реализуют необходимый алгоритм для работы с конкретным типом электронного потребительского устройства.

Так же анализ существующих приемов онтологического инжиниринга и технологий семантического веба для агрегации больших массивов гетероген-

ных данных выявил преимущества использования технологий семантического веба и онтологий для нормализации данных электронных потребительских устройств в целях их агрегации. Данные технологии позволяют реализовать методы и алгоритмы аннотирования данных для решения проблем обработки гетерогенных данных распределенной сети электронных потребительских устройств.

Кроме сбора данных под агрегацией в данной работе понимается процесс хранения собранных данных для последующего доступа к ним и обработки. Соответственно в рамках данной работы так же необходимо реализовывать метод и алгоритмы хранения гетерогенных данных электронных потребительских устройств соответствующие требованиям технического задания, а именно требованию по времени хранения данных не менее 1000 устройств.

В итоге в рамках данной работы разработаны следующие методы и алгоритмы:

- а) метод агрегации данных с электронных потребительских устройств с ограниченными вычислительными ресурсами (ЭПУОВР), использующих прикладной протокол передачи данных СоАР (Constrained Application Protocol).
- б) алгоритм асинхронной передачи данных с ЭПУОВР, использующих протокол СоАР.
- в) алгоритм кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF (Resource Description Framework) и протокол СоАР.
- г) метод агрегации данных с ЭПУОВР, использующих проприетарный прикладной протокол передачи данных на основе протокола UDP (User Datagram Protocol).
- д) метод и алгоритм аннотирования данных с ЭПУОВР, не использующих модель данных RDF.
- е) метод и алгоритмы хранения больших массивов данных распределенной сети электронных потребительских устройств.

1.2 Метод агрегации данных с электронных потребительских устройств с ограниченными вычислительными ресурсами (ЭПУОВР), использующих прикладной протокол передачи данных CoAP (Constrained Application Protocol)

Согласно результатам аналитического обзора научно-технической литературы на 1-м этапе данного ПНИ [1] был сделан вывод о том что для передачи данных с электронных потребительских устройств необходимо использовать специализированные протоколы передачи данных, который бы эффективно использовать вычислительные ресурсы устройств. Выделяют 3 класса устройств с ограниченными вычислительными ресурсами [2], см. Таблицу 1.1. Исходя из описанных характеристики электронных потребительских устройств с ограниченными вычислительными ресурсами и обзору существующих протоколов передачи данных в данной работе предлагается использовать протокол CoAP (Constrained Application Protocol).

Таблица 1.1 — Классы электронных потребительских устройств с ограниченными вычислительными ресурсами

Название	размер памяти (напр. RAM)	размер кода (напр. Flash)
Класс 0, C0	<< 10 Кб	<< 100 Кб
Класс 1, C1	~ 10 Кб	~ 100 Кб
Класс 2, C2	~ 50 Кб	~ 250 Кб

Соответственно в данном подразделе описывается метод построения REST-интерфейса электронного потребительского устройства для агрегации данных с таких устройств.

Протокол CoAP – это результат работы рабочей группы Constrained RESTful Environments (CORE) в организации IETF. Это протокол передачи данных прикладного уровня, цель которого реализовать возможность взаимодействия по принципам REST, как это реализовано в протоколе HTTP, но

для устройств и в сетях с низкой пропускной способностью и ограниченными вычислительными ресурсами.

СоAP состоит из двух подуровней: уровень сообщений и уровень запросов/ответов. Уровень сообщений расширяет протокол UDP с поддержкой механизма определения дубликатов и, если необходимо, механизма гарантированной доставки сообщений, реализованный через простую ретрансляцию с экспоненциальной задержкой. Уровень запросов/ответов позволяет использовать REST-подобное взаимодействие через унифицированные интерфейсы, адресуемые по URI, широко известные методы, такие как GET, PUT, POST и DELETE, и передачу информации о самом устройстве через адресуемые ресурсы. Являясь более «легковесным» чем протокол HTTP, СоAP поддерживает только ограниченный набор функций HTTP и использует компактное, бинарное кодирование, которое было спроектировано для сериализации и разбора на устройствах с ограниченными вычислительными возможностями.

Подробный обзор и анализ существующих протоколов передачи данных для устройств с ограниченными ресурсами был проведен в промежуточном отчёте о ПНИ за 1-й этап работ [1].

Необходимо отметить связь, предоставляемую протоколом, между ресурсами и корнем. Согласно спецификации¹, сведения о ресурсах должно быть явно закреплены в корне устройства. Также, при наличии связи между ресурсами, она должна быть отмечена использованием специального слова *anchor*. Заголовки ссылок СоAP расширяют заголовки протокола HTTP, не требуя при этом какого-либо дополнительной специальной XML или бинарной обработки. При этом MIME-заголовок устанавливается как "*application/link-format*". Параметр заголовка *href* зарезервирован для использования в качестве параметра запроса и не может быть использован как параметр ссылки. Множественные описания ссылок разделяются кавычками. Стоит заметить, что кавычки встречаются в и в строках, и в URI – и не могут закачивать описание. Каждая ссылка передает конечный URI как URI-ссылку в угловых скобках.

Как было написано ранее, выбранный протокол предоставляет возможность использования REST-подобного взаимодействия. Достоинством данно-

¹Constrained RESTful Environments Link Format, <https://tools.ietf.org/html/rfc6690>

го метода является простота его использования, позволяя передавать данные в качестве параметров запроса. Также основными преимуществами представленного метода взаимодействия являются: надежность (поскольку отсутствует необходимость сохранять информацию о состоянии клиента, которая может быть утеряна), масштабируемость и простота разработки и внесения изменений.

Применяя СоАР к ЭПУОВР и, соответственно, используя REST взаимодействие, необходимо выделить применяемое API. Выделим используемое API, предоставляемое СоАР. Им являются GET и Observe запросы. Observe-запрос на определенный ресурс ЭПУОВР предоставляет возможность постоянно получать данные, генерируемые данным ресурсом, иначе говоря подписываться на обновление данных этого ресурса. При помощи такого запроса можно получать данные, генерируемые сенсорами, в виде потока, для последующей работы с ними.

По другому работает GET-запрос. При направлении запроса в корень устройства (по адресу */.well-known/core*), ответом служат информационные данные о самом устройстве и о его существующих сенсорах, называемых ресурсами. Пример ответа на данный запрос для датчика температуры при использовании специального расширения браузера для работы по СоАР-протоколу (Cooper¹) приведен на рисунке 1.1. При направлении запроса на ресурс, ответом служат последние сгенерированные этим ресурсом данные. При этом, в отличие от Observe-запроса, данные будут возвращены однократно, а не постоянно возвращаться при обновлении их на ресурсе. Пример ответа на такой запрос для датчика температуры приведен на рисунке 1.2.

```
1 </meter>,
2 </meter/heat>,
3 </meter/heat/obs>;obs ,
4 </meter/temperature>,
5 </meter/temperature/obs>;obs ,
6 </.well-known/core>
```

Рисунок 1.1 — Пример ответа на GET-запрос, используя Cooper, на
/.well-known/core

¹Copper (Cu), <https://addons.mozilla.org/En-us/firefox/addon/copper-270430/>

```

1 @prefix hmtr: <http://purl.org/NET/ssnext/heatmeters#> .
2 @prefix meter: <http://purl.org/NET/ssnext/meters/core#> .
3 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
5 @prefix om: <http://purl.org/ifgi/om#> .
6 @prefix : <coap://localhost:60002/meter/heat#> .
7
8 :1434821286881 a hmtr:HeatObservation ;
9   ssn:observationResultTime "2015-06-20T20:28:06"^^xsd:dateTime ;
10  ssn:observedBy <coap://localhost:60002/meter> ;
11  ssn:observationResult :1434821286881-result .
12
13 :1434821286881-result a hmtr:HeatSensorOutput ;
14   ssn:isProducedBy <coap://localhost:60002/meter> ;
15   ssn:hasValue :1434821286881-resultvalue .
16
17 :1434821286881-resultvalue a hmtr:HeatValue ;
18   meter:hasQuantityValue "0.9858575230550226"^^xsd:float ;
19   om:hasQuantityUnitOfMeasurement
      <http://purl.oclc.org/NET/muo/ucum/unit/energy/Joule> .

```

Рисунок 1.2 — Пример ответа на GET-запрос на ресурс *heat*

Используя данный протокол для получения данных с ЭПУОВР и для возможности полноценного предоставления метаданных, получаемых с устройств, было решено использовать описание устройств в формате RDF в совокупности с языком для описания онтологий OWL. Данный подход предоставляет возможность получать полный спектр данных, описывающих ЭПУОВР, таких как: установленные на устройстве сенсоры, по какому протоколу можно получать с них данные, на какие из сенсоров данного устройства можно совершить подписку и проч.

Пример получаемых об устройстве данных приведен на рисунке 1.3. В нем описаны префиксы для установления связи между системами ЭПУОВР, в качестве *\$HOST* и *\$PORT* выступают, соответственно, данные о имени хоста и порту, на котором доступно устройство. Также можно заметить, что приведено описание сенсоров как точек коммуникации и протоколы взаимодействия с ними.

В данном разделе был рассмотрен протокол СоAP, описаны его характеристики, возможности использования и отличия от протокола HTTP

```

1 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
3 @prefix hmtr: <http://purl.org/NET/ssnext/heatmeters#> .
4 @prefix ssncom: <http://purl.org/NET/ssnext/communication#> .
5
6 <coap://${HOST}:${PORT}/meter> a hmtr:HeatMeter ;
7   rdfs:label "Heat Meter #${PORT}" ;
8   ssn:hasSubsystem <coap://${HOST}:${PORT}/meter/temperature> ;
9   ssn:hasSubsystem <coap://${HOST}:${PORT}/meter/heat> .
10
11 <coap://${HOST}:/${PORT}/meter/temperature> a ssn:Sensor ;
12   ssn:observes hmtr:Temperature ;
13   ssncom:hasCommunicationEndpoint
14     <coap://${HOST}:${PORT}/meter/temperature/obs> .
15
16 <coap://${HOST}:${PORT}/meter/heat> a ssn:Sensor ;
17   ssn:observes hmtr:Heat ;
18   ssncom:hasCommunicationEndpoint <coap://${HOST}:${PORT}/meter/heat/obs> .
19
20 <coap://${HOST}:${PORT}/meter/temperature/obs> a ssncom:CommunicationEndpoint ;
21   ssncom:protocol "COAP" .
22 <coap://${HOST}:${PORT}/meter/heat/obs> a ssncom:CommunicationEndpoint ;
23   ssncom:protocol "COAP" .

```

Рисунок 1.3 — Пример получаемого описания устройства

и выделено предоставляемое протоколом API, применяемое в ПО ЭО ПАП. Также был разработан метод агрегации данных с ЭПУОВР с использованием описанного протокола СоAP.

1.3 Алгоритм асинхронной передачи данных с ЭПУОВР, использующих протокол СоAP

Для передачи данных с ЭПУОВР был выбран протокол СоAP – стандартизованный протокол передачи данных в сети с ограниченными вычислительными узлами RFC 7252¹.

Для реализации протокола СоAP для Arduino-совместимых устройств (широкий класс встраиваемых систем с ограниченными ресурсами, спроектированных для упрощения процесса быстрого прототипирования) за основу

¹RFC 7252 – The Constrained Application Protocol (CoAP) – <https://tools.ietf.org/html/rfc7252>

была взята существующая на языке C реализация microcoap¹. Она была выбрана по причинам низкой требовательности к ресурсам, высокой кросс-платформенности, возможности использования как в POSIX-совместимых системах, так и на широком наборе классов микроконтроллеров с ограниченными ресурсами. Однако, существующая реализация предполагала минимальный необходимый набор поддерживаемых функций, не включающий поддержку Observe-подписки на ресурсы, что было дополнительно реализовано согласно документации по стандарту RFC 7252.

Работа по протоколу состоит из следующих последовательных этапов:

- a) Ожидание новых сообщений от клиентов по протоколу UDP. В случае отсутствия на момент проверки, переход к обработке подписок.
- б) Получение нового сообщения от клиента. Обработка, поиск двоичного представления сообщения согласно СоAP-представлению.
- в) Проверка полученного сообщения на необходимость реагирования на него подготовленным образом, поиск в предустановленном списке ресурсов наличия запрашиваемого.
- г) Ответ согласно методам, подготовленным для найденного ресурса.
- д) В случае наличия в сообщении метки Observe для запроса на наблюдение, ответ согласно протоколу СоAP и сохранение клиента и запрашиваемого ресурса в список наблюдателей.
- е) В случае наличия списка наблюдателей, проверка для каждого на факт изменения наблюдаемого ресурса и ответ согласно СоAP-протоколу.

Алгоритм представлен на рисунке 1.4 в виде блок-схемы.

Таким образом, подготовленная реализация СоAP-протокола делает возможным ее использование на широком классе Arduino-совместимых устройств, добавляя полноценный функционал протокола, включающий подписку на обозреваемые ресурсы.

¹microcoap – реализация СоAP протокола для микроконтроллеров – <https://github.com/1248/microcoap>

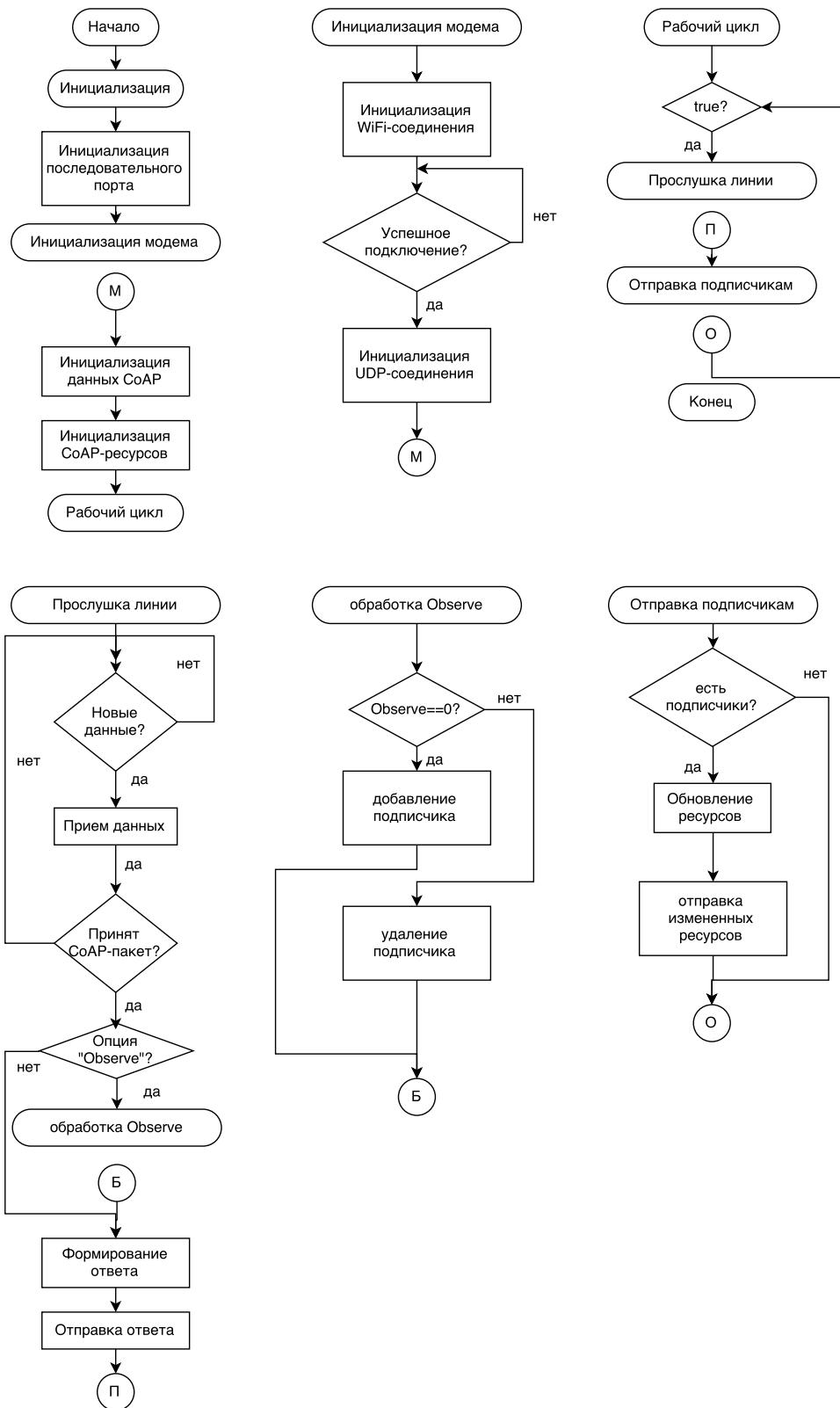


Рисунок 1.4 — Алгоритм реализации протокола CoAP для Arduino с поддержкой опции Observe

1.4 Алгоритмы кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF (Resource Description Framework) и протокол CoAP

Среди существующих подходов к кодированию и декодированию данных ЭПУОВР, использующих модель данных RDF, рабочей группы W3C RDF Stream Processing Community Group выделяются следующие способы представления данных в двоичном виде¹:

- а) HDT [3]
- б) SHDT [4]
- в) ERI [5]
- г) RDSZ [6]
- д) Эффективный RDF с применением технологий W3C EXI
- е) Двоичный RDF используя Apache Thrift²
- ж) Двоичный RDF от Sesame³
- з) RDF-3X [7]

Наиболее приемлемой и перспективной является описываемая технология «Эффективное и стандартизированное кодирование RDF-данных для сетей на встраиваемых системах» (EXI) [8], которая и была выбрана в качестве основы.

Формат EXI использует относительно простой, основанный на грамматиках, подход, который позволяет достичь очень эффективной кодировки EXI-потоков для широкого спектра задач. EXI-представление зачастую меньше XML-представления в более чем сотни раз [9]. Основанный на высокой степени сжатия с возможностью получать типизированные данные напрямую из EXI-потока, базирующийся на XML подход к передаче данных применим и в области встраиваемых систем с существенно ограниченными ресурсами [10]. Формат EXI распространяется в области встраиваемых систем (например, в области автомобилестроения – e-Mobility [11], и умных сетей энергоснабжения

¹См. http://www.w3.org/community/rsp/wiki/RSP_Serialization_Group

²RDF Binary using Apache Thrift, <http://afs.github.io/rdf-thrift/>

³Binary RDF in Sesame, <http://www.rivuli-development.com/2011/11/binary-rdf-in-sesame/>

- Smart Energy Profile¹) – все больше и больше благодаря своим ключевым преимуществам – таким, как: низкая требовательность к памяти, мощностям процессора и полосе пропускания.

Алгоритм кодирования EXI-данных для передачи по протоколу СоAP на встраиваемых системах с ограниченными ресурсами представлен на рисунке 1.5 в виде блок-схемы.

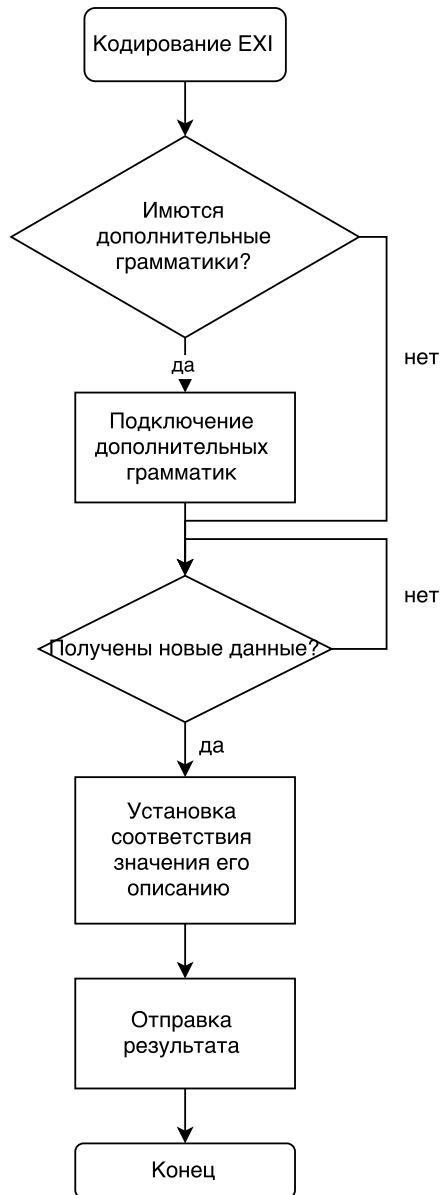


Рисунок 1.5 — Алгоритмы кодирования EXI-данных для передачи по протоколу СоAP на встраиваемых системах

¹ZigBee. Smart Energy Profile 2 (SEP 2) (2013) – <http://www.zigbee.org/>

Грамматики составляются согласно указанной *XML-схеме*, где каждый сложный объявленный тип данных представлен в качестве «предопределенного конечного автомата» (DFA). Кроме того, EXI-формат также имеет возможность работать в режиме «без схемы», что означает, что EXI-процессор использует общие грамматики, определяемые стандартом.

В случае с представлением RDF/XML, имеет смысл использовать определяемый схемой режим работы, предоставляя данные из RDF-схемы и то, как выглядят данные¹: каждый RDF-документ начинается с корневого RDF-элемента и несет в себе набор *Описаний* дочерних элементов для формулирования триплетов. Это позволяет описывать различные схемы RDF и EXI-грамматики с учетом их использования в конкретных задачах.

На рисунке 1.6 отражен пример EXI-грамматики G (набор автоматизаций), которая может быть использована для кодирования и декодирования RDF-данных. Эта грамматика опирается на схему XML, представленную инструментарием RDF с *корневым* RDF-элементом и встроенными *Описаниями* элементов в виде триплетов. Стоит заметить, что *корневая* грамматика – предопределенная, встречающаяся в каждом представлении EXI-грамматики любых XML-схем. Она содержит все точки входа ко всем корневым элементам выбранной схемы.

Здесь отмечается контекст связанных корневых RDF-элементов инструментария RDF XML. В основном, каждый DFA содержит одно начальное представление и одно конечное, отражающие соответственно начало и конец сложных объявлений типов.

Переходы на следующий уровень представления представляет собой последовательное объявление элементов и/или атрибутов сложных типов. Дополнительные объявления (например, *выбор*, *число вхождений* и т.д.) – отражаются множественными переходами и привязаны к коду события (EV).

Например, элемент *Описание* – зачастую повторяющийся элемент в RDF, в свою очередь объявляется как *цикл* в схеме XML с помощью установления неограниченного числа вхождений *maxOccurs = 'unbounded'* и отображен в EXI-грамматике двумя переходами: одним для повторного перехода к *Описанию* и одним – к конечному состоянию. Для отметки об этом одноби-

¹Синтаксис RDF 1.1 XML – <http://www.w3.org/TR/rdf-syntax-grammar/>

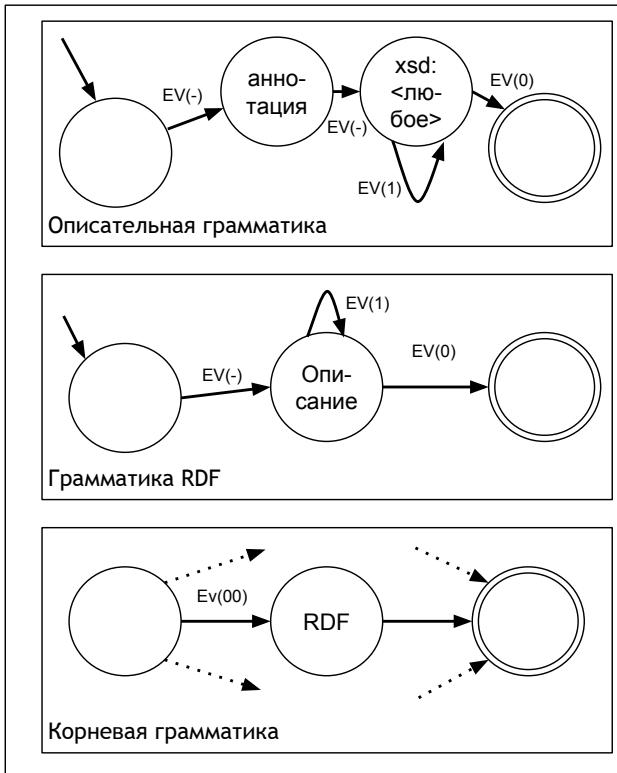


Рисунок 1.6 – Общая грамматика RDF EXI

товый код события (EV) привязывается к переходу ($EV(1)$ – для *Описания*, $EV(0)$ – для завершения). Обычно число бит, используемых для m переходов определяется, как: $[log_2 m]$. Код события $EV(-)$ означает, что больше кодов событий не потребуется.

Состояние *xsd:<любое>* означает описание предиката и объекта. Это состояние означает основную часть, так как названия предикатов и значения объектов зависят от текущего приложения.

Таким образом, пример RDF-XML представления, отраженный на рисунке 1.7, согласно описанию выше может быть представлен следующим EXI потоком (в целях удобства чтения, пространства имен сокращены): 00 'temperature' type resource 'sensor' value '28.5' 0 1 'humidity' 0 1 0 3 '59.6'...

Пример сразу демонстрирует, насколько компактнее становится запись. Синим цветом обозначены биты кодов событий, используемые для навигации в EXI-потоке при кодировании и декодировании. Зеленым цветом обозначены значения атрибутов (например, атрибут *about* со значениями температура и влажность: *temperature* и *humidity*) и элементов (например, элемент *значение* с показаниями 28.5 и 59.6).

```
1 <RDF>
2 <Description about= 'temperature'>
3 <type resource='sensor' />
4 <value>28.5</value>
5 </Description>
6 <Description about= 'humidity'>
7 <type resource='sensor' />
8 <value>59.6</value>
9 ...
```

Рисунок 1.7 — Пример RDF-XML представления

Таким образом, EXI-кодирование использует данные о типах, представляя эффективные механизмы кодирования для основных типов данных (например, *int*, *float*, *enumerations* и т.д.). В целях простоты, в примере EXI-потока значения отображены в форме, удобной для чтения человеком. Тем не менее, в случае с float-подобными значениями в контексте температуры, EXI-кодирование использует только 2 байта для представления значения *8.4*. Оранжевым на примере отмечены предикаты, которые не покрываются схемой *xsd:<любое>*. Механизм кодирования EXI в этом случае основан на следующем принципе: по первому появлению неизвестного элемента или атрибута в потоке. Например, наименование строкового типа запоминается и связывается с уникальным идентификатором ID, а после используется выбранный идентификатор.

Для избежания слишком обобщенного подхода, при котором все неизвестные наименования приходится дополнительном кодировать уникальными идентификаторами, предлагается использовать дополнительные XML-схемы и EXI-грамматики. Это имеет смысл в случае, когда семантическое представление заранее известно и может быть отражено в сторонней схеме. Такая схема может включать, например, все свойства данных и объектов, классы и диапазоны значений.

Рассмотрим, например, ситуацию, где требуется описать онтологию для встраиваемой системы с одним датчиком, получающим значения температуры и влажности, похожим на пример выше. Для этого можно добавить грамматику, представленную на рисунке 1.8, которая, кроме дополнительного верхнего уровня, отличается от рисунка 1.6 *Описательной* грамматикой

тем, что имеется более конкретное описание состояний *тип* и *значение* между состоянием-атрибутом *аннотация* и конечным состоянием.

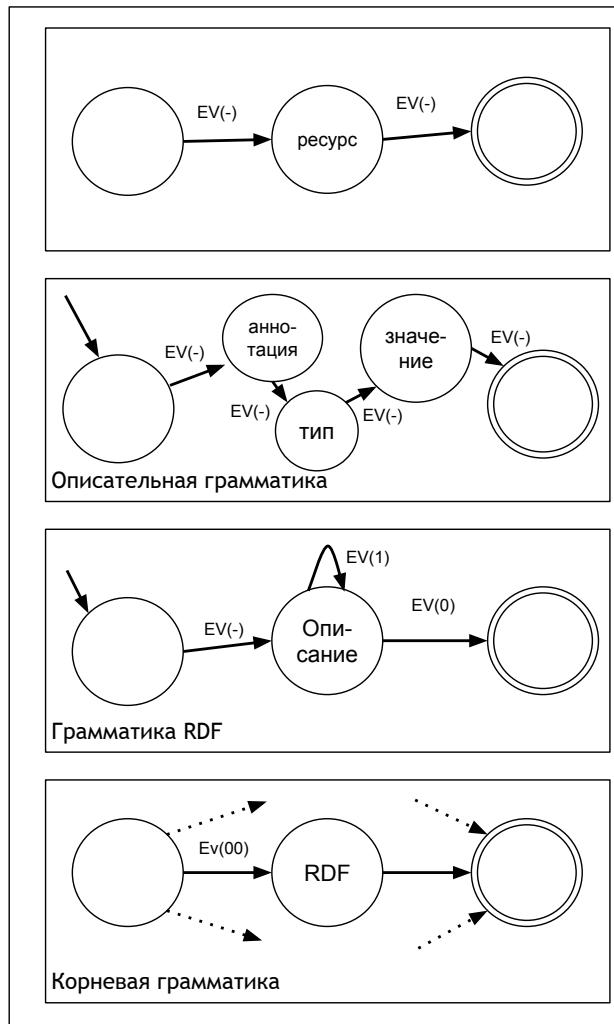


Рисунок 1.8 — Прикладная грамматика RDF/EXI

Применяя такой шаблон, получаем поток: 00 'temperature' 0 '8.4' 1 'humidity' 0 '79.2' ...

Очевидно, что представленный поток еще компактнее благодаря тому, что информация о типах, ресурсах и значениях элементов уже «вшита» в грамматику и не нуждается больше в отдельном представлении в потоке. Кроме того, т.к. онтология предоставляет определения всех классов, связанных с потоком, можно определить список всех имен классов в XML-схеме, при этом EXI-представление может использовать только порядковый номер в этом списке, как и показано красным цветом на примере (*0 – 'sensor'*).

Встраиваемые системы в промышленной области в основном обмениваются физическими значениями. Строковые типы при этом – редкость. По-

этому и предлагается использовать W3C EXI технологию для рационального использования ограниченной динамической памяти¹. Это позволяет оперировать значениями без строковых таблиц. В случае с повторяющимися строками (*температура, влажность*), они могут быть обработаны общим методом, описанным выше, как новые.

Алгоритм дедирования EXI-данных, передаваемых по протоколу СоAP, обрабатывается полноценной операционной системой шлюза, где ресурсов на прием данных от встраиваемых систем достаточно. Он аналогичен алгоритму кодирования и представлен на рисунке 1.9 в виде блок-схемы.

Оценка проверяемых результатов на встраиваемых системах с ограниченными ресурсами показывает существенное уменьшение размера RDF-представления применением технологий EXI-сериализации (возникших как необходимость преодоления проблем неэффективного использования формата XML при передаче потоковых данных [12]). Кроме того, выбранная технология позволяет реализовать семантическое хранилище на системах с ограниченными ресурсами.

Стандарт Консорциума Всемирной паутины W3C по сериализации RDF-данных требует больших затрат на предварительную обработку, основную обработку и хранение данных. Это становится наиболее критичным, когда речь идет о встраиваемых системах с ограниченными ресурсами, и существенно замедляет внедрение семантических технологий в IoT-область. Разрабатываемое решение основывается на формате «Эффективного обмена XML-данными» (EXI) – бинарного стандарта для XML от Консорциума W3C. Адаптация EXI позволяет также использовать его и для нужд описания данных в формате RDF. Для этого необходима обобщенная RDF-EXI грамматика и оценка ее эффективности по сравнению с другими возможными решениями.

Протокол СоAP допускает использование в качестве нагрузки в передаваемых сообщениях любых данных, что позволяет использовать сообщения в EXI-формате. Т.к. предполагается использование систем с ограниченными ресурсами, после анализа существующих работ [13] было решено использо-

¹Fablet, Y., Peintner, D.: Efficient XML Interchange (EXI) Profile for limiting usage of dynamic memory. W3C Recommendation, 09 September 2014 – <http://www.w3.org/TR/exi-profile/>

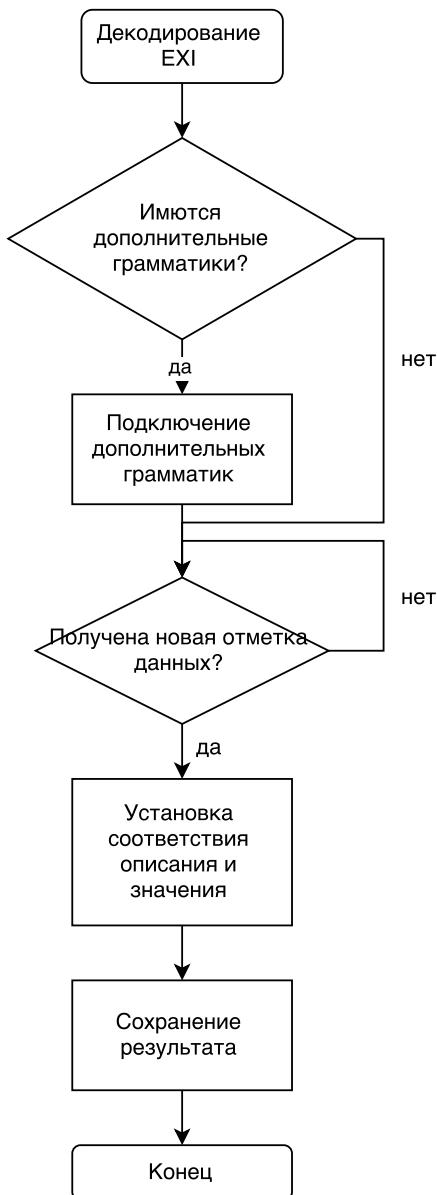


Рисунок 1.9 — Алгоритмы декодирования EXI-данных, передаваемых по протоколу СоAP

вать пре-обработку EXI-грамматики для эффективного ее использования в памяти устройства. Благодаря подобному подходу в памяти устройства создается структура для С-языка (на котором написана сама прошивка для устройства), экономно использующая ограниченные ресурсы для хранения EXI-грамматики. Решение предполагает предварительной подготовки словаря на этапе прошивки, но позволяет эффективно использовать EXI-сжатие на встраиваемых системах с ограниченными ресурсами.

Таким образом, EXI-представление данных с помощью предварительной обработки используемых грамматик для сохранения в памяти систем с ограниченными ресурсами эффективно при использовании в передаче данных по выбранному СоAP-протоколу, что позволяет добиться на встраиваемых системах необходимой гибкости в описании данных и эффективной потоковой передаче, а также обнаружении ресурсов и их изменения.

Также стоит отметить, что выбранная основа разрабатываемых методик базируется на установленных W3C стандартах, что важно в условиях промышленной разработки. Разрабатываемая технология спроектирована так, чтобы иметь возможность выполнять запросы на получения данных на SPARQL-подобном языке, адаптированным для встраиваемых систем, и быть пригодной для внедрения в общую разрабатываемую систему.

1.5 Метод агрегации данных с ЭПУОВР, использующих проприетарный прикладной протокол передачи данных на основе протокола UDP (User Datagram Protocol)

Каждый ЭПУОВР имеет свой API для передачи показаний. В разрабатываемой ЭО ПАП используется протокол СоAP для работы с устройствами составляющими распределенную сеть электронных потребительских устройств, но в мире существует множество других протоколов, таких как MQTT, XMPP и т.д., которые используются при передаче данных такими устройствами. В связи с этим возникает проблема подключения новых ЭПУОВР, не реализующих работу по протоколу СоAP, к разрабатываемой ЭО ПАП.

Для решения возникшей проблемы было принято решение разработать метод позволяющий работать с любыми ЭПУОВР. При разработке метода были выделены следующие требования:

- а) простота реализации;
- б) адаптируемость;
- в) масштабируемость.

Разработанный метод основывается на возможности работы с ЭПУОВР без изменения функционала ПО ЭО ПАП. Основной идеей метода яв-

ляется написание гибких драйверов реализующих один интерфейс. Данный интерфейс описывает основные функции необходимые системе для работы с устройством с ограниченными ресурсами. Для каждого типа ЭПУОВР описывается один драйвер.

В системе так же необходим конфигурационный файл отвечающий за работу системы с драйверами. Данный файл содержит перечисление следующих параметров для каждого устройства, использующего драйвер:

- а) URI устройства.
- б) полное имя класса драйвера.
- в) имя пакета с классом драйвера.
- г) список входных данных.

URI устройства будет идентифицировать используемое устройство. Полное имя класса драйвера необходимо для поиска конкретного класса в пакете. Имя пакета с классом драйвера описывает пакет в котором можно найти драйвер. Список входных данных содержит полный список посылаемых параметров при инициализации драйвера, к таким данным можно отнести IP, порт и т.д.

Все драйвера упакованы в один jar файл. Данный jar файл размещается в корневой папке системы. При запуске программы загружается конфигурационный файл, отвечающий за связь драйверов и устройств, далее система использует написанные драйвера в соответствии с описанным конфигурационным файлом. Для вступления в силу измененного конфигурационного файла систему необходимо остановить и запустить заного. Рассмотрим более подробно разработанный интерфейс, реализуемый при написании драйверов. Данный интерфейс имеет следующие основные методы:

- а) boolean initialize().
- б) void run().
- в) String getDriverName().

Метод initialize позволяет создать и подготовить полностью необходимые данные к работе с устройством. Одним из примеров подготовки данных можно привести инициализацию входных данных, например IP и порт с кото-

рыми необходимо будет производить работу в дальнем. Run будет выполнять необходимую работу по непосредственному запуску драйвера и началу получения данных с устройства. Так же был приведен в примере один из типовых методов getDriverName, который будет возвращать название драйвера.

Рассмотрим пример работы с устройством посылающим данные на основе протокола UDP. Данный пример представлен в виде фрагмента реализации класса драйвера на рисунке 1.10.

```
1 public class EmptyDriver implements SensorDriver {  
2  
3     private String driverName;  
4     private InetAddress addr;  
5     private String port;  
6  
7     public boolean initialize() {  
8         driverName = getName();  
9         addr = getAddr();  
10        port = getPort();  
11        return true;  
12    }  
13    public void run() {  
14        try {  
15            DatagramSocket data = new DatagramSocket(port);  
16            InetAddress address = InetAddress.getByName(addr);  
17            while(true) {  
18                byte[] buf = new byte[1000];  
19                DatagramPacket packet = new DatagramPacket(buf, buf.length);  
20                data.receive(packet);  
21                addResult(new String(packet.getData()));  
22            }  
23        } catch (IOException e) {  
24            e.printStackTrace();  
25        }  
26    }  
27    public String getDriverName() {  
28        return driverName;  
29    }  
30}
```

Рисунок 1.10 — Фрагмент реализации класса драйвера

Для работы с драйверами была разработана архитектура представленная на рисунке 1.11.

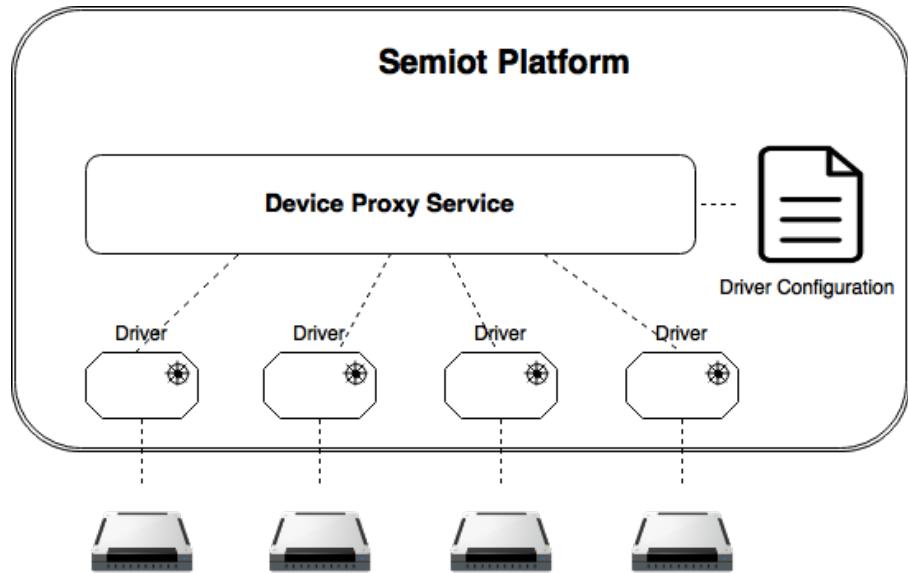


Рисунок 1.11 – Архитектура работы системы с драйверами

Разработанный метод позволяет работать с различными ЭВУОПР, при этом не изменяя функционал системы Semiot Platform. При добавлении нового устройства необходимо добавить описание драйвера и устройства в конфигурационный файл и перезагрузить систему. Данный подход обеспечивает простой подход для расширения поддерживаемых ЭВУОПР.

1.6 Метод и алгоритм аннотирования данных с ЭПУОВР, не использующих модель данных RDF

При работе с различными видами устройств необходимо рассматривать и проблему различных форматов данных. В рассматриваемом случае многие устройства не поддерживают модель данных RDF. Поэтому при использовании драйверов необходимо выполнить процесс преобразования от одной модели данных к другой. Обеспечить гибкость данному процессу позволяет использование конфигурационных файлов для каждого из драйверов, аннотирующих данные поступающие с устройств. Каждый тип устройств имеет свою заранее известную модель данных, что позволяет написать для него конфигурационный файл описывающий преобразование к модели данных RDF.

Для аннотирования данных использовались шаблоны в RDF формате, в которых будут вставляться поступающие данные с устройства.

Конфигурационный файл выполняющий аннотирование данных для устройств должен быть размещен в одной папке с классом драйвера и иметь такое же название, но с расширением *.ttl*. Рассмотрим небольшой пример поступающих данных по протоколу UDP. Пример шаблона описывающего данные в формате RDF силы тока для ЭПУОВР приведен на рисунке 1.12.

```
1 @prefix emtr: <http://purl.org/NET/ssnext/electricmeters#> .  
2 @prefix meter: <http://purl.org/NET/ssnext/meters/core#> .  
3 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .  
4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
5 @prefix om: <http://purl.org/ifgi/om#> .  
6 @prefix : <coap://{$HOST}:{$PORT}/meter/amperage#> .  
7  
8 :${TIMESTAMP} a emtr:AmperageObservation ;  
9   ssn:observationResultTime "${DATETIME}"^^xsd:dateTime ;  
10  ssn:observedBy <coap://{$HOST}:{$PORT}/meter> ;  
11  ssn:observationResult :${TIMESTAMP}-result .  
12  
13 :${TIMESTAMP}-result a emtr:AmperageSensorOutput ;  
14   ssn:isProducedBy <coap://{$HOST}:{$PORT}/meter> ;  
15   ssn:hasValue : ${TIMESTAMP}-resultvalue .  
16  
17 :${TIMESTAMP}-resultvalue a emtr:AmperageValue ;  
18   meter:hasQuantityValue "${VALUE}"^^xsd:float ;  
19   om:hasQuantityUnitOfMeasurement  
     <http://purl.oclc.org/NET/muo/ucum/unit/electric-current/Ampere> .
```

Рисунок 1.12 — Шаблон описания данных в формате RDF

В данном случае в шаблон будут подставлены значения *TIMESTAMP*, *HOST*, *PORT*, *VALUE*, *DATETIME*. Данные значения будут посыпаться ЭПУОВР. Где *TIMESTAMP* характеризует дату и время снятых показаний в количестве секунд прошедших с 00:00:00 UTC 1 января, 1970 года, *HOST* описывает адрес с которого был принят пакет с данными, *PORT* описывает порт с которого был принят пакет с данными, *VALUE* показатель значения ЭПУОВР, *DATETIME* дата и время снятых показаний в формате "yyyy-MM-dd'T'HH:mm:ss".

С ЭПУОВР силы тока посылающего данные в пакете по протоколу UDP, на IP адрес 192.45.42.03, порт 6521 приходят следующие данные, считываемые драйвером: *VALUE* = 5, *TIMESTAMP* = 1434956400. В данном

случае в переводе к формату DATETIME получаем 2015-06-22T10:00:00. После подстановки значений в шаблон описанный на рисунке 1.12. Получаем готовое показание ЭПУОВР в формате RDF описанный на рисунке 1.13.

```
1 @prefix emtr: <http://purl.org/NET/ssnext/electricmeters#> .  
2 @prefix meter: <http://purl.org/NET/ssnext/meters/core#> .  
3 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .  
4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
5 @prefix om: <http://purl.org/ifgi/om#> .  
6 @prefix : <coap://192.45.42.03:6521/meter/amperage#> .  
7  
8 :1434956400 a emtr:AmperageObservation ;  
9     ssn:observationResultTime "2015-06-22T10:00:00"^^xsd:dateTime ;  
10    ssn:observedBy <coap://192.45.42.03:6521/meter> ;  
11    ssn:observationResult :1434956400-result .  
12  
13 :1434956400-result a emtr:AmperageSensorOutput ;  
14     ssn:isProducedBy <coap://192.45.42.03:6521/meter> ;  
15     ssn:hasValue :1434956400-resultvalue .  
16  
17 :1434956400-resultvalue a emtr:AmperageValue ;  
18     meter:hasQuantityValue "5"^^xsd:float ;  
19     om:hasQuantityUnitOfMeasurement  
          <http://purl.oclc.org/NET/muo/ucum/unit/electric-current/Ampere> .
```

Рисунок 1.13 – Показания ЭПУОВР в формате RDF

Алгоритм и последовательность работы системы Semiot Platform с ЭПУОВР по разработанному методу представлены на рисунках 1.14 и 1.15.

Рассмотрим более подробно алгоритм работы в виде блок-схемы и диаграмму последовательности системы с ЭПУОВР, не использующих модель данных RDF. Device Proxy Service запрашивает у Driver Configuration конфигурационный файл, описывающий написанные драйвера для ЭПУОВР. Далее выполняется создание и инициализация объекта класса драйвера для ЭПУОВР. После инициализации вызывается run(), что означает запуск работы драйвера и начало приема данных драйвером. Далее драйвер после получения пакета с данными заправшивает шаблон аннотирующий полученные данные. Драйвер производит подстановку полученных данных в шаблон и возвращает их Device Proxy Service.

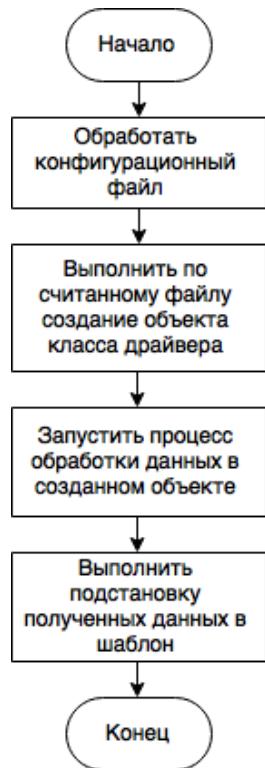
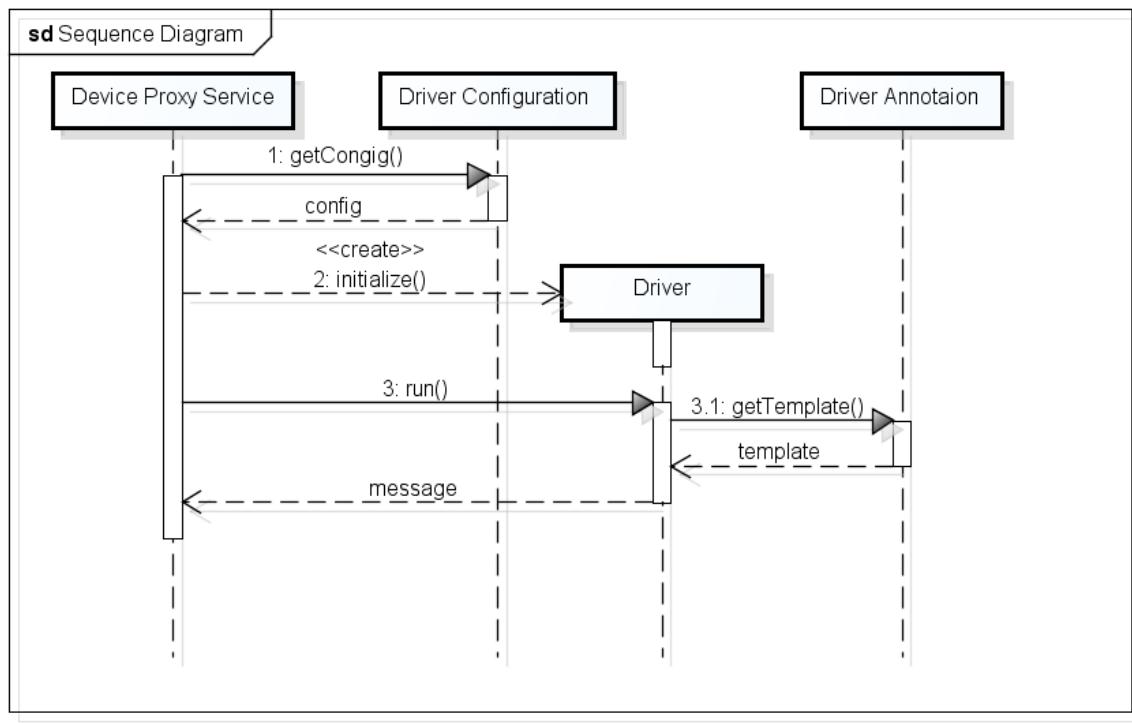


Рисунок 1.14 – Алгоритм работы системы с ЭПУОВР, не использующих модель данных RDF



powered by Astah

Рисунок 1.15 – Диаграмма последовательности работы системы с ЭПУОВР, не использующих модель данных RDF

В результате описанный метод аннотирования данных, позволяет преобразовать данные поступающие с ЭВУОПР к формату данных RDF. Данный метод прост в использовании за счет использования шаблонов, где незаполненные значения заполняются поступающими данными с устройства.

1.7 Метод и алгоритмы хранения больших массивов данных распределенной сети электронных потребительских устройств

По результатам анализа существующих приемов онтологического инжиниринга и технологий семантического веба для агрегации больших массивов гетерогенных данных, выполненного на 1-м этапе данного ПНИ [1], принято решение использовать модель данных RDF для кодирования данных электронных потребительских устройств в ЭО ПО ПАП.

Данные электронных потребительских устройств можно разделить на два типа: метаданные, описывающие характеристики устройств, такие как например точность измерения, диапазон измерений и т.д.; данные измерений (или показаний), например величина влажности воздуха или температуры. Данные первого типа изменяются редко и не привязаны ко времени, а данные второго типа являются временными, т.е. жестко связаны со временем, когда они были получены с устройства. Так же данные второго типа, являющиеся измерениями или показаниями утсойства, поступают в ПО ЭО ПАП часто, соответственно система хранения для данных такого типа должна отвечать следующим требованиям:

- а) высокая скорость записи,
- б) высокая пропускная способность,
- в) предоставлять дополнительные возможности по поиску данных по их временным меткам.

Для хранения данных такого типа используются специализированные базы данных для хранения временных рядов, примерами таких БД являются OpenTSDB¹, InfluxDB² и другие. Для хранения временных данных в данном проекте была выбрана база данных OpenTSDB по ряду причин:

¹<http://opentsdb.net/>

²<https://influxdb.com/>

- а) исходный код является открытым и распространяется по двойной лицензии (GPLv3+ и LGPLv2.1+), что позволяет использовать её как в проектах с открытым исходным кодом, так и в коммерческих проектах,
- б) база данных является распределенной и для хранения данных на каждом из узлов используется Apache HBase¹, которая превосходит существующие аналоги по производительности,
- в) для записи и чтения данных предоставляется HTTP REST API, который позволяет работать с БД без привязки к какому-либо определенному языку программирования.

Для записи данных обрабатываются показания ЭПУОВР в формате RDF, описанные на рисунке 1.12. Для записи показаний в OpenTSDB был разработан алгоритм представленный на рисунке 1.16 в виде блок-схемы.

Подключение к WAMP выполняется для возможности получения сообщений. После подключения к WAMP производится постоянная проверка существующего соединения, в случае его разрыва автоматически производится повторное подключение. Следующим важным этапом алгоритма является осуществление подписки на получение сообщений к топику, который в дальнейшем будет присыпать списки топиков сенсоров, по которым можно получить их показания.

При получении нового топика, по которому будут получены показания сенсора, осуществляется подписка на этот топик и дальнейшее ожидание новых показаний сенсора. При получении сообщения с показаниями, осуществляется преобразование полученного сообщения из формата RDF к схеме хранения данных в OpenTSDB. После преобразования выполняется запрос на запись нового значения в OpenTSDB по полученным показаниям.

Для оптимизации времени работы было решено осуществлять запись в OpenTSDB, сразу нескольких значений, которые были собраны за определенный промежуток времени. На текущий момент был выбран промежуток времени в 5 секунд. Это позволяет уменьшить количество запросов записи к OpenTSDB в течении 5 секунд до одного.

¹<http://hbase.apache.org/>

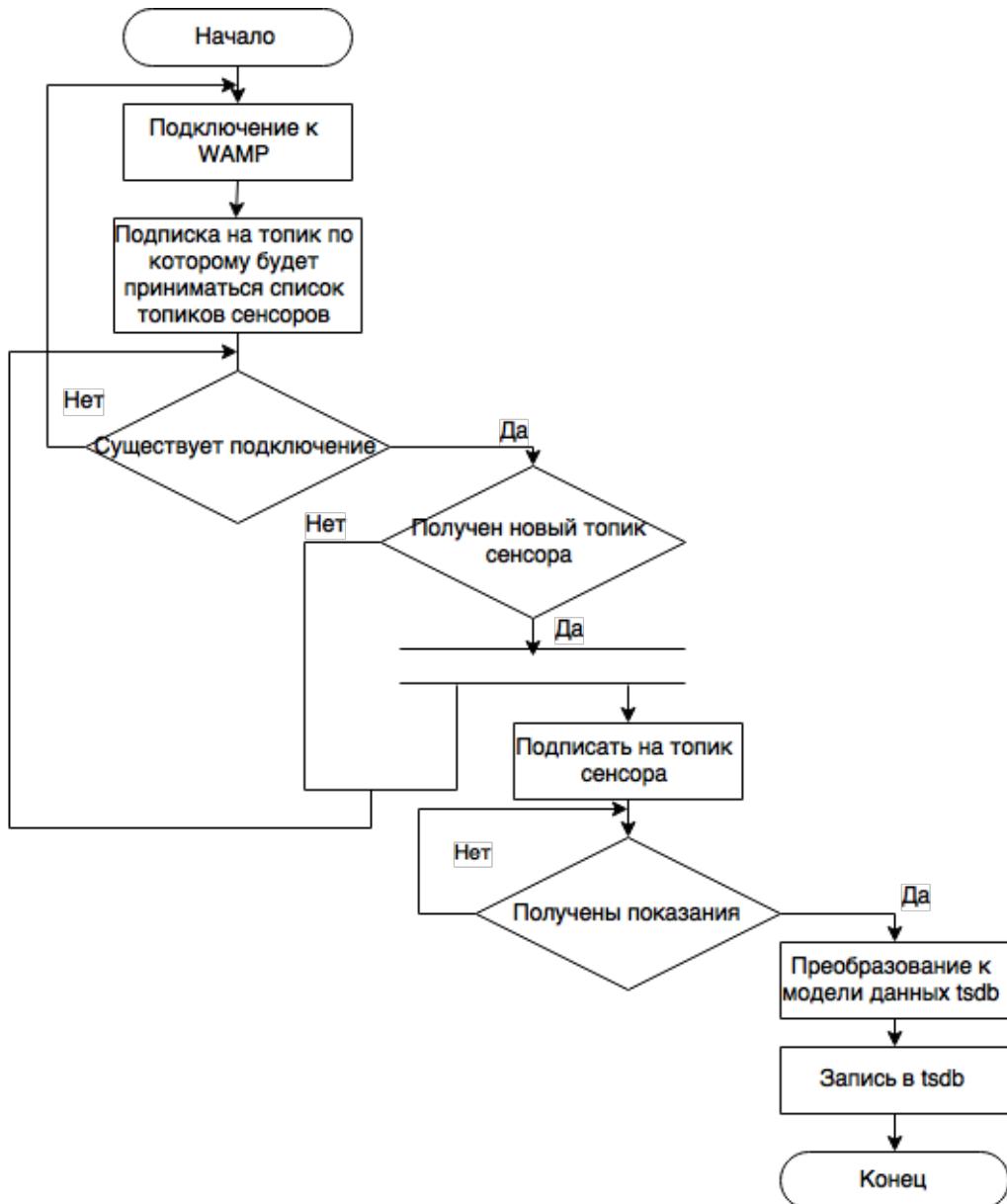


Рисунок 1.16 – Алгоритм записи показаний в OpenTSDB

OpenTSDB хранит данные в виде таблицы с множеством записей имеющих формат: <metric_uid><timestamp><tagk1><tagv1>[...<tagkN><tagvN>].¹

Данная схема позволяет записывать показания, где metric_uid - является идентификатором сенсора ЭПУОВР, timestamp описывает время дату и время снятых показаний в количестве секунд прошедших с 00:00:00 UTC 1 января 1970 года, tagkN описывает название тэга, tagvN описывает значение тэга tagkN, тэги позволяют записывать название и значение показания.

¹http://opentsdb.net/docs/build/html/user_guide/backends/hbase.html#data-table-schema/

Для осуществления записи в OpenTSDB необходимо преобразовать полученные показания сенсоров ЭПУОВР в формате RDF к приведенной выше схеме. Для преобразования выполняется запрос по получению необходимых значений из полученных показаний ЭПУОВР в формате RDF. В результате выполнения запроса, получаем следующие значения, позволяющие преобразовать к схеме OpenTSDB:

- а) metric_uid соответствует идентификатору сенсора полученному при осуществлении подписки на топик,
- б) timestamp соответствует времени и дате ssn:observationResultTime,
- в) tag1 val соответствует значение meter:hasQuantityValue,
- г) tag2 type соответствует типу устройства.

Например для показаний приведенных на рисунке 1.13, преобразованные показания будут иметь следующий вид: metric_uid = 192.45.42.03.6521.meter.amperage, timestamp = 1434967200, val = 5, type = emtr_AmperageObservation.

Для данного проекта в качестве базы данных хранения метаданных была выбрана Fuseki¹ по следующим причинам:

- а) исходный код является открытым,
- б) идет активная стадия разработки, что позволяет сотрудничать с разработчиками, задавая различного рода вопросы и указывая на существующие недостатки,
- в) удобный пользовательский интерфейс,
- г) простота развертывания.

Для записи метаданных ЭПУОВР в Fuseki был разработан алгоритм представленный на рисунке 1.16 в виде блок-схемы.

Первым этапом является получение сообщения с метаданными. Для осуществления получения сообщения, как было описано выше, необходимо выполнить подключение к WAMP и подписаться на топик, по которому в дальнейшем будут пересылаться сообщения с метаданными. Полученное со-

¹<http://jena.apache.org/documentation/fuseki2/index.html>

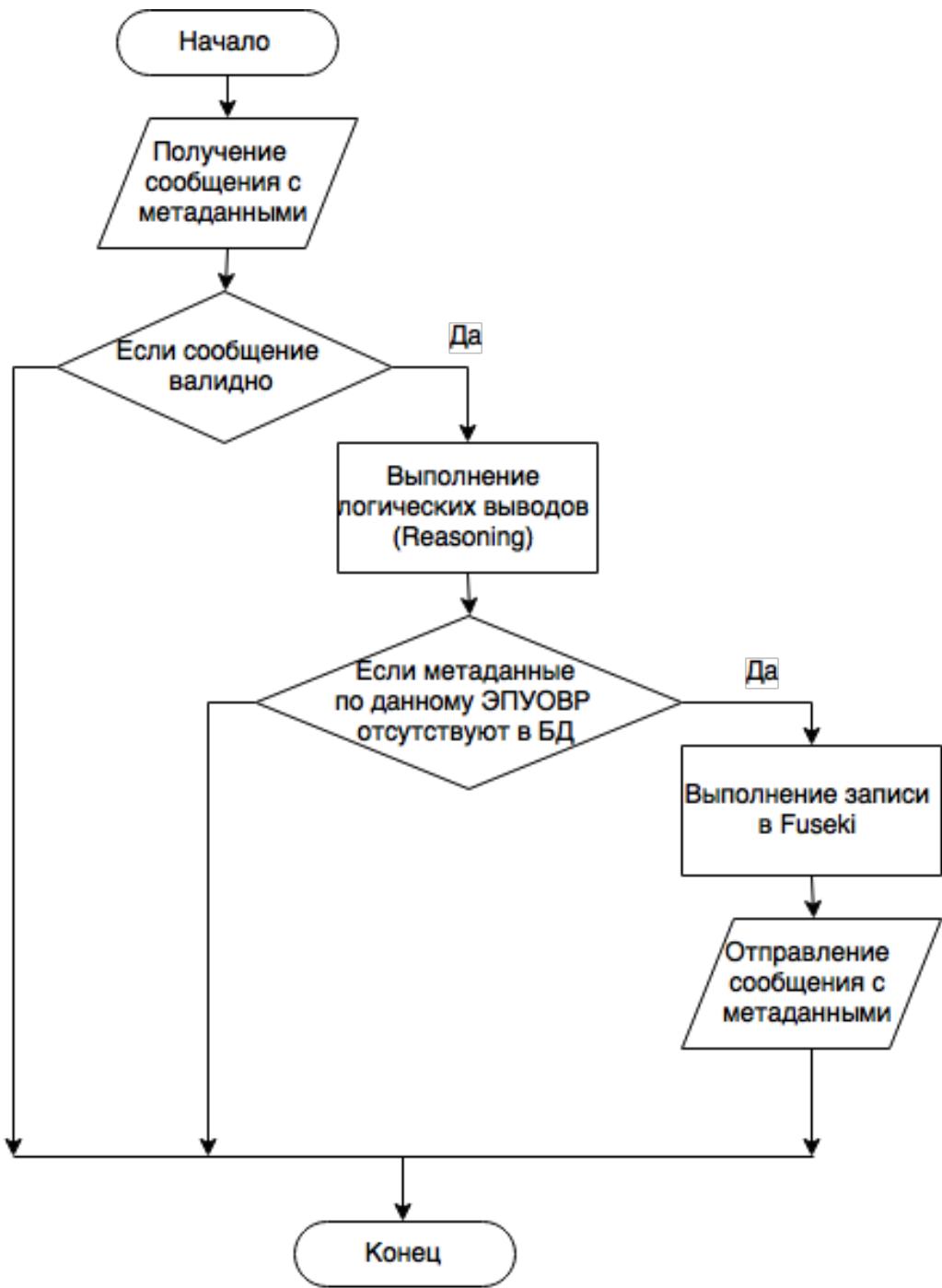


Рисунок 1.17 — Алгоритм записи метаданных в Fuseki

общение валидируется. В случае если сообщение не валидно, то далее сообщение не обрабатывается. Если сообщение валидно, то далее осуществляется формирование логических выводов (reasoning). Далее осуществляется проверка на существование метаданных для данного ЭПУОВР. Данная проверка необходима для исключения дублирования метаданных для одних и тех же ЭПУОВР. Если метаданные по данному ЭПУОВР отсутствуют в Fuseki, то далее выполняется запрос на запись в базу данных, полученного сообщения.

Далее данное сообщение пересыпается на топик, по которому осуществляется пересылка сообщений о новых ЭПУОВР.

1.8 Результаты

В данном разделе отчета описаны методы и алгоритмы агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба.

В соответствии с техническим заданием, разработанные в рамках данной работы методы и алгоритмы агрегации больших массивов гетерогенных данных должны: поддерживать работу по протоколам TCP/IP и/или UDP/IP; поддерживать синхронный и асинхронный режимы передачи данных; обеспечивать хранение данных с глубиной не менее 1 года для не менее чем 1000 электронных потребительских устройств. Все перечисленные требования ТЗ обеспечиваются техническими характеристиками разработанных методов и алгоритмов описанных в данном разделе, кроме того соответствие последнему требованию по хранению данных исследовано в Приложении А данного отчета о ПНИ.

Кроме того в Приложении Б представлены отчетные и справочные материалы по докладу на 17-й международной конференции FRUCT под названием «An implementation of CoAP protocol for Arduino and ESP8266», который освещает работу описанную в подразделе 1.2.

В результате работы были разработаны и исследованы необходимые методы и алгоритмы, соответствующие требованиям Технического задания.

2 Разработка математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things)

2.1 Назначение и состав математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things)

Под нормализацией гетерогенных данных понимается процесс трансформации исходных данных различных моделей в унифицированную модель данных, способной сохранить всю необходимую информацию. Нормализация гетерогенных данных является необходимым этапом обработки данных распределенной сети электронных потребительских устройств, который предшествует этапу анализа этих данных для выявления различных событий предметной области. Например, в области теплоснабжения таким событием является так называемый перетоп, который заключается в том что количества тепла, поступающего в дом, слишком много для текущей температуры воздуха на улице и в помещении. В рамках данной работы поставлена задача разработать методы и алгоритмы проблеме нормализации данных различных моделей и анализа событий предметной области.

На основе результатов аналитического обзора современной научно-технической литературы, а также анализа существующих приемов онтологического инжиниринга и технологий семантического веба для агрегации больших массивов гетерогенных данных, выполненных на 1-м этапе данного ПНИ [1], было решено в качестве унифицированной модели данных, используемой в процессе нормализации гетерогенных данных, использовать верхнеуровневые OWL-онтологии и модель данных RDF, который зарекомендовали себя как инструмент для обеспечения синтаксической и семантической интероперабельности. Так же преимущество данной модели данных является то, что она предоставляет расширенные возможности по анализу данных электрон-

ных потребительских устройств в режиме реального времени, используя как потоковые данные (показаний или измерения), так и метаданные устройств (точность измерений, местоположение и т.д.). Для анализа этих данных предлагается использовать технологии связанных потоковых данных описанный в подразделе 2.5.

Для анализа связанных потоковых данных используются специализированные программные системы, которыми являются системы CQELS и C-SPARQL. На данном этапе было принято решение разработать метод и реализовать алгоритмы оценки производительности и корректности существующих систем для анализа связанных потоковых данных, и для последующего выбора одной из существующих систем для работ на текущем и последующих этапах ПНИ.

В итоге в рамках данной работы должны быть разработаны следующие методы и алгоритмы:

- а) метод нормализации гетерогенных данных распределенной сети электронных потребительских устройств на основе верхнеуровневых онтологий,
- б) алгоритм установления соответствия между предметно-ориентированными онтологиями и верхнеуровневыми онтологиями, используемыми для нормализации гетерогенных данных,
- в) метод и алгоритмы оценки производительности и корректности систем анализа связанных потоковых данных,
- г) метод и алгоритмы анализа потоковых и статических данных распределенной сети электронных потребительских устройств,
- д) метод обнаружения и генерации событий предметной области на основе анализа потоковых и статических данных.

2.2 Метод нормализации гетерогенных данных распределенной сети электронных потребительских устройств на основе верхнеуровневых онтологий

Область семантической интеграции данных активно развивается в нескольких дисциплинах, таких как базы данных, интеграция информации

и онтологии. Исследователи баз данных и интеграции информации выполнили большое количество исследований для упрощения реализации интероперабельности между системами. Эти исследования описывают методы сопоставления схем баз данных для выполнения запросов с использованием нескольких источников данных. Методы онтологического инжиниринга являются другой областью, занимающейся решением задач обеспечения семантической интероперабельности структурированных данных. В статье [14] детально обсуждается использование онтологий, их отличие от схем баз данных и проблемы в интеграции данных, с которыми сталкиваются исследователи из этой области. Также в статье [15] приводится всеобъемлющий обзор текущего состояния области интеграции данных на основе онтологий.

Существует несколько определений понятия "онтология", согласно авторам статьи [16], можно выделить одно общее определение. "Онтология" - это некоторое формальное описание предметной области, которое предназначено для обмена этим описанием между приложениями и которое выражено на языке поддерживающий "логический вывод".

Этими возможностями, которые предоставляют онтологии, отличается семантическая интеграция данных с помощью онтологического инжиниринга:

- a) Во-первых, так как основополагающей целью разработки онтологий является создание артефактов, которыми приложения могут обмениваться, акцентируется внимание на создание верхнеуровневых онтологий, которые могут быть расширены для более специфичных областей и приложений. Если такие расширения ссылаются на одну и туже верхнеуровневую онтологию, то проблема из интеграции значительно упрощается.
- б) Во-вторых, так как онтологии разрабатываются для использования с движками логического вывода, и семантика языка представления онтологий подразумевает использование логического вывода, то рассуждения и логический вывод имеют важное значение в подходах к интеграции данных на основе онтологий.

В данной работе под нормализацией гетерогенных данных распределенной сети электронных потребительских устройств понимается процесс интеграции данных этой сети на основе верхнеуровневых онтологий.

Выделяют три задачи в интеграции данных на основе онтологий [17]:

- а) Определение соответствий между концептами: если даны две онтологии, то как определяются похожие концепты.
- б) Декларативное представление соответствий: если даны две онтологии, то как мы представляем соответствия между ними, чтобы позже можно было бы использовать логический вывод.
- в) Логический вывод с соответствиями: если соответствия определены, то как мы их используем.

Метод интеграции данных распределенной сети электронных потребительских устройств, предлагаемый в данной работе, основан на использовании верхнеуровневых онтологий, которые используются для создания предметно-ориентированных онтологий. Соответственно соответствие между концептами определяется на этапе создания данных онтологий. Соответствие между классами онтологий устанавливается с помощью конструкций *rdfs:subClassOf* и *owl:sameAs* определенных в языке описания онтологий OWL (Web Ontology Language)¹. А соответствия между свойствами онтологий устанавливаются с помощью конструкций *rdfs:subPropertyOf* и *owl:equivalentProperty*.

Большое количество очень общих онтологий, формализующих такие понятия, как процессы и события, время и пространство, физические объекты и другие, были разработаны и адаптированы, а некоторые из них стали стандартами. Основная цель таких онтологий = это предоставить предметно-ориентированным онтологиям базовый набор концепций. Можно заметить что такой сценарий отличается от традиционного сценария интеграции информации, когда глобальная схема обычно разрабатывается *после* того как предметные онтологии уже были адаптированы и задачей становиться интегрировать их в одну глобальную схему.

¹OWL 2 Web Ontology Language Quick Reference Guide (Second Edition), <http://www.w3.org/TR/owl2-quick-reference/>

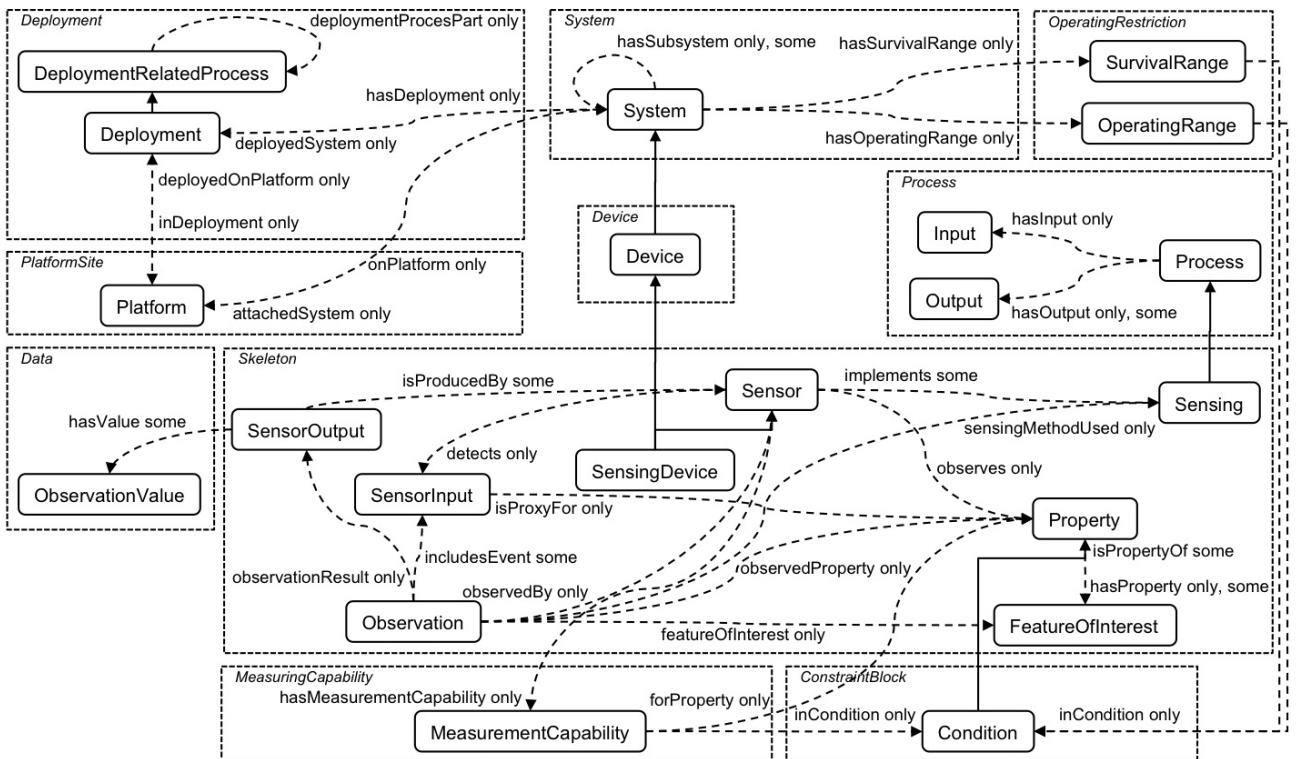


Рисунок 2.1 – Структура онтологии *SSN* с её основными классами и свойствами

В области сенсорных сетей (или сетей электронных потребительских устройств) уже существуют некоторое количество высокоуровневых онтологий, которые позволяют описать базовые концепции данной предметной области. В 2011-м году рабочая группа W3C Semantic Sensor Network Incubator Group¹ опубликовала финальный отчет [18], в котором представила верхнеуровневую онтологию для представления элементов сенсорных сетей, таких как *сенсор*, *показание*, *точность показаний* и т.д. Эта онтология получила название Semantic Sensor Network Ontology (*SSN*)², на рисунке 2.1 представлена структура онтологии с разделением на модули.

Описываемый метод основывается на использовании некоторых элементов онтологии *SSN* для представления сущностей принимающих участие в работе ЭО ПАП. Одной из таких сущностей является *устройство*, которое способно снимать какие-либо показания, может быть частью какого-либо составного устройства и т.д. Для этой сущности мы используем классы *ssn:System* и *ssn:Sensor*, где *ssn:System* представляет собой устройство, ко-

¹W3C Semantic Sensor Network Incubator group, <http://www.w3.org/2005/Incubator/ssn/>

²The Semantic Sensor Network Ontology, <http://purl.oclc.org/NET/ssnx/ssn#>

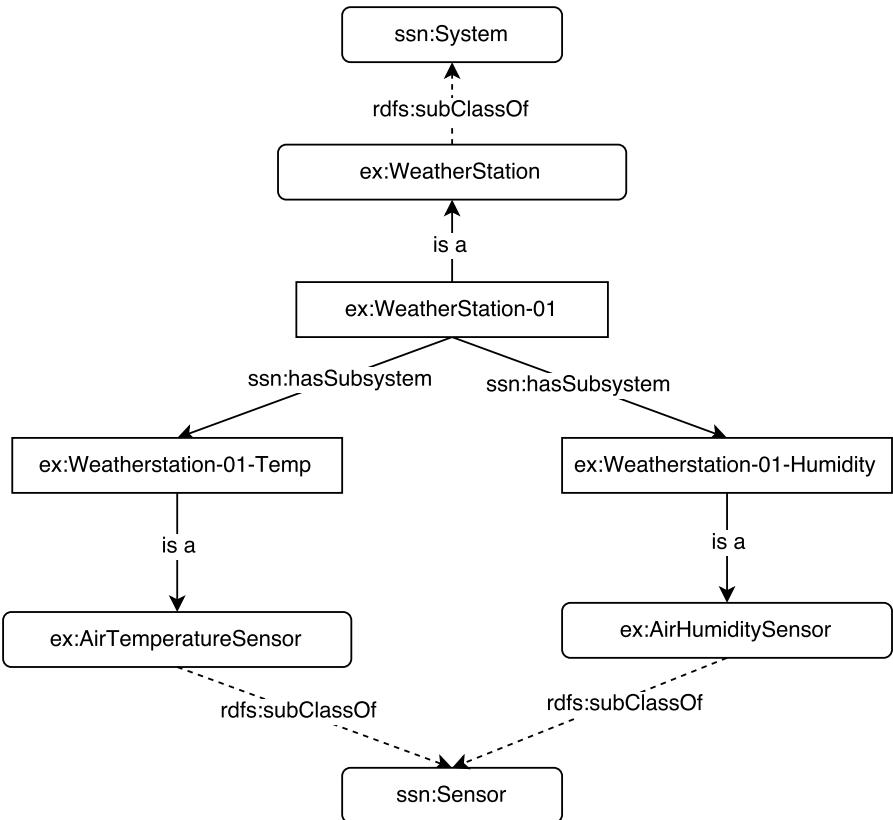


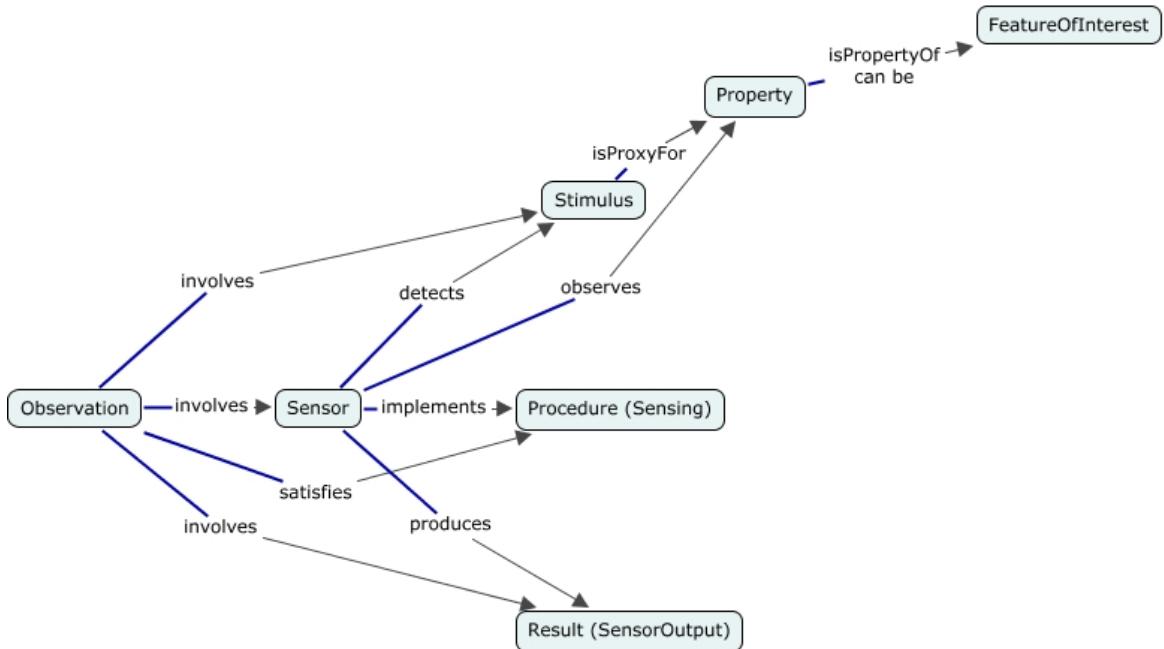
Рисунок 2.2 — Класс *ssn:System* и OWL-свойство *ssn:hasSubsystem*

торое может являться составным, и *ssn:Sensor* - устройство считающее или измеряющее какое-либо одно свойство окружающей среды, например, скорость ветра. Связи между составным устройством и его частью осуществляется через OWL-свойство *ssn:hasSubsystem*, см. рисунок 2.2. Здесь и далее для классов и свойств, которые описываются только для примера, используется префикс *ex*.

На рисунке 2.2 представлен пример устройства *ex:TempHum-01* измеряющего температуру и влажность окружающей среды. Это устройство является составным и включает в себя два сенсора/датчика: датчик температуры воздуха и датчик влажности. Для описания этого связи используются классы *ssn:System* и *ssn:Sensor*, а также объектное свойство *ssn:hasSubsystem*, которое устанавливает связь между составным устройством и его частями.

Другим базовым концептом в сенсорной сети является "наблюдение" или "измерение" состояния. В онтологии SSN для представления этого концепта существует класс *ssn:Observation*, который имеет свойства указывающие на то каким датчиков данное показание было снято, состояние какого

свойства окружающей среды датчик наблюдал (например скорость ветра или температура воздуха) и результат наблюдения. На рисунке 2.3 можно наблюдать схематичное описание связей класса *ssn:Observation* с другими классами данной онтологии.



Для описания свойств окружающей среды, которое наблюдается, предлагается создавать отдельные классы, которые наследуются от классов *ssn:FeatureOfInterest* и *ssn:Property*. В продолжении примера с датчиком температуры и влажности, на рисунке 2.4 схематически изображена иерархия классов и связей между ними, которая представляет такие свойства воздуха, как температура (*ex:AirTemperature*) и влажность (*ex:AirHumidity*). Свойство *ssn:hasProperty* используется для указать связь между предметом окружающей среды и его свойствами.

Кроме онтологии SSN предлагается также использовать концепты из онтологии QUDT¹, которая включается в описание единиц измерения, размеры и типы данных. Например, единицы измерения длины (*unit:Meter*), времени (*unit:SecondTime*), электричества (*unit:Ampere*) и т.д.

¹QUDT - Quantities, Units, Dimensions and Data Types Ontologies, <http://qudt.org/>

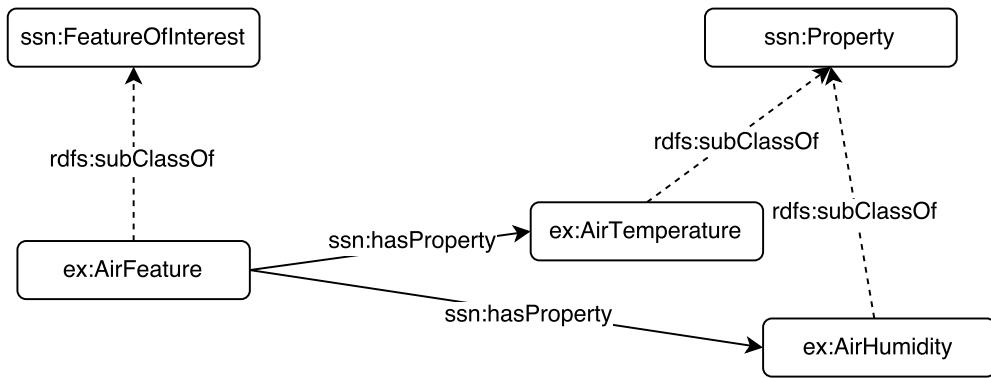


Рисунок 2.4 — Пример представления свойств воздуха, таких как температура и влажность на основе онтологии SSN

Метод подразумевает использование выше перечисленных базовых классов и свойств при разработке предметно-ориентированных онтологий. Ниже приводятся примеры предметно-ориентированных онтологий разработанных в рамках данного ПНИ.

В продолжении примера с датчиком температуры и влажности, на рисунке 2.5 приведено описание данного датчика в формате Turtle¹.

```

1 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
2 @prefix ex: <http://example.semiot.ru/> .
3
4 ex:WeatherStation-01 a ex:WeatherStation ;
5   ssn:hasSubsystem ex:WeatherStation-01-Temp ;
6   ssn:hasSubsystem ex:WeatherStation-01-Humidity ;
7
8 ex:WeatherStation-01-Temp a ex:AirTemperatureSensor ;
9   ssn:observes ex:AirTemperature .
10
11 ex:WeatherStation-01-Humidity a ex:AirHumiditySensor ;
12   ssn:observes ex:AirHumidity .
13
14 ex:AirFeature a ssn:FeatureOfInterest ;
15   ssn:hasProperty ex:AirTemperature ;
16   ssn:hasProperty ex:AirHumidity .

```

Рисунок 2.5 — Пример описания датчика температуры и влажность

¹Turtle, <http://www.w3.org/TR/turtle/>

Одним из рабочих примеров использования результатов работы в рамках данного ПНИ является сбор и обработка показаний тепловых и электрических счетчиков. Поэтому были разработаны онтологии для этих областей. На рисунке 2.6 изображена иерархия классов и свойств онтологии ElectricMeter, которая описывает счетчики электроэнергии.

Онтология ElectricMeter опубликована в репозитории GitHub по адресу <https://github.com/semitoproject/ontologies/blob/master/emtr/electricmeters.owl>.

Предложенный метод обеспечивает возможность работы с любым классом устройств распределенной сети электронных потребительских устройств, если его метаданные и показаний аннотированы с помощью предметно-ориентированной онтологий, которая является расширением онтологии SSN. В подразделе 2.3 описан алгоритм использующий логический вывод для восстановления связей между такими предметно-ориентированными онтологиями и верхнеуровневыми онтологиями, от которых они наследуются.

2.3 Алгоритм установления соответствия между предметно-ориентированными онтологиями и верхнеуровневыми онтологиями, используемыми для нормализации гетерогенных данных

Алгоритм установления соответствия между предметно-ориентированными онтологиями и верхнеуровневыми онтологиями основан на выполнении логического вывода на основе онтологий с предопределенным набором правил.

Логический вывод на основе онтологий - вывод логических следствий из набора известных фактов или аксиом, с использованием правил вывода, которые в свою очередь указываются с помощью онтологического языка (например OWL).

На рисунке 2.7 изображена архитектура части ПО ЭО ПАП отвечающая за сбор и нормализацию гетерошенных данных распределенной сети электронных потребительских устройств. Модуль Device Proxy Service отвечает за сбор данных (подробнее в разделах 1.2, 1.5 и 1.6), а модуль Device Directory Service - за хранение метаданных устройств и установление соот-

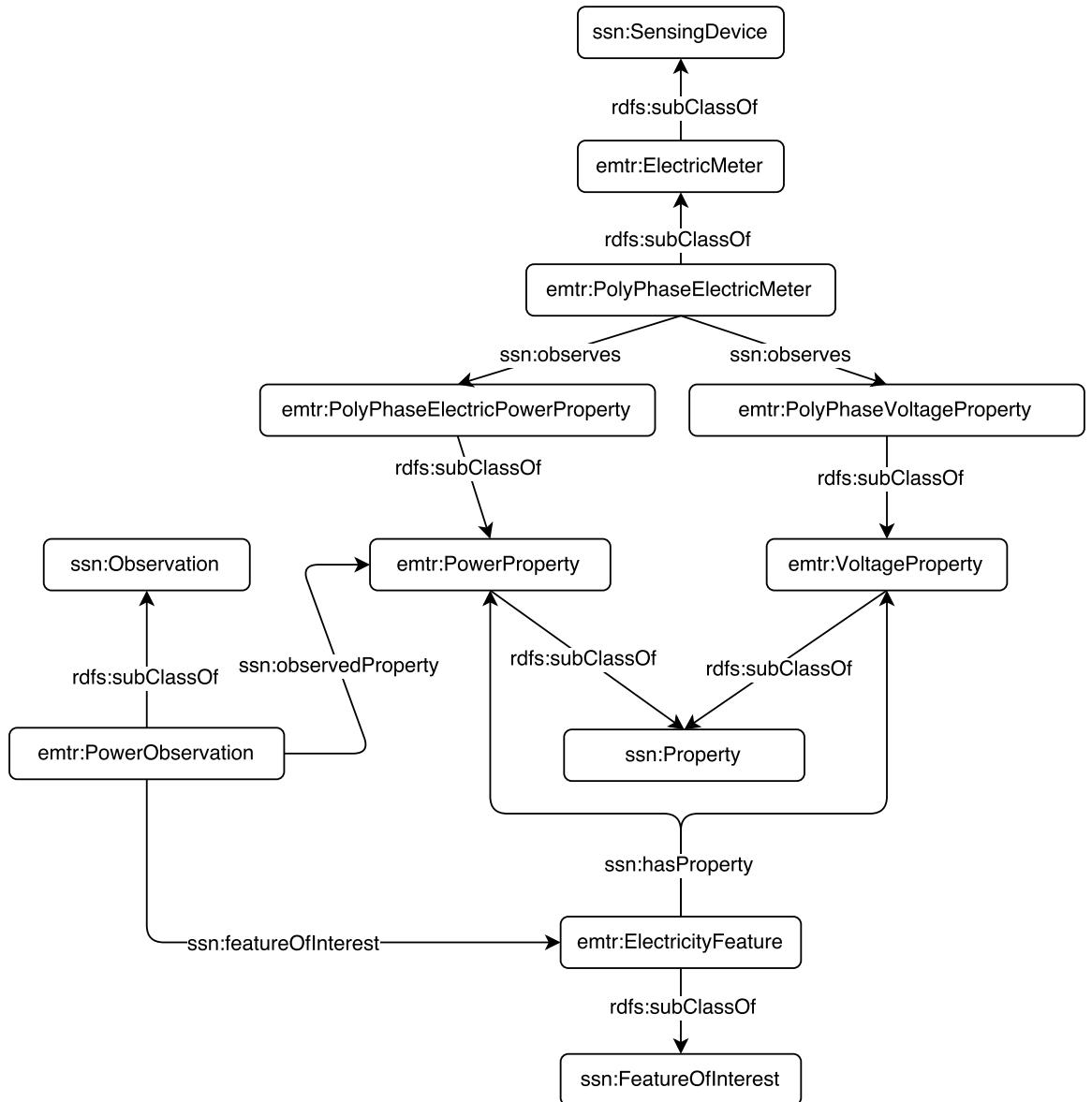


Рисунок 2.6 — Классы и свойства онтологии ElectricMeter

ветствия между предметно-ориентированным и верхнеуровневыми онтологиями с помощью логического вывода. Так же модуль Device Directory Service предоставляет доступ к метаданным устройств через SPARQL-точку доступа по средствам протокола HTTP.

На рисунке 2.8 представлен алгоритм подготовки и выполнения логического вывода для установления связей между предметно-ориентированной и верхнеуровневой онтологиями. Метаданные устройства получаются модулем Device Proxy Service в момент первичной регистрации устройства в ЭО ПАП, далее они валидируются и переотправляются через Message Bus в модуль Device Directory Service. В свою очередь метаданные устройства в фор-

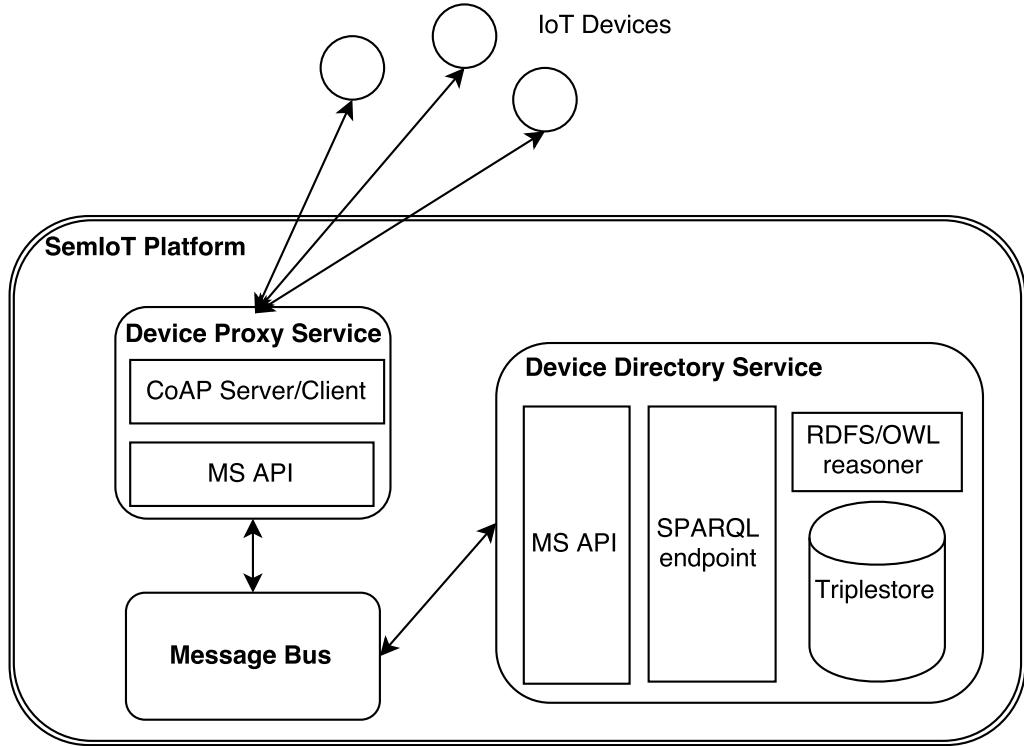


Рисунок 2.7 – Архитектура части ПО ЭО ПАП, отвечающей за сбор и нормализацию данных сенсорной сети

мате Turtle извлекаются из сообщения, пришедшее из Message Bus. И далее извлекаются URI всех онтологий участвующих в описании данного устройства и они позже подкачиваются из Интернета или если они уже есть во внутреннем кэше модуля, то берутся оттуда. На следующем шаге метаданные, непосредственно полученные с устройства, онтологии и правила вывода необходимые для вывода новых фактов подгружаются в движок логического вывода (reasoner). Этот движок в итоге выдает метаданные, которые уже содержат описание устройства в концептах, как предметно-ориентированной и так и верхнеуровневой онтологии.

Используются следующие правила вывода:

- Для свойства rdfs:subClassOf: [subClassOf: (?x rdfs:subClassOf ?y), (?a rdf:type ?x) -> (?a rdf:type ?y)]
- Для свойства rdfs:subPropertyOf: [subPropertyOf: (?p rdfs:subPropertyOf ?q) (?a ?p ?b) -> (?a ?q ?b)]

Рассмотрим пример метаданных устройства, которой измеряет показаний температуры и влажности воздуха, из 2.5. Предметно-ориентированная

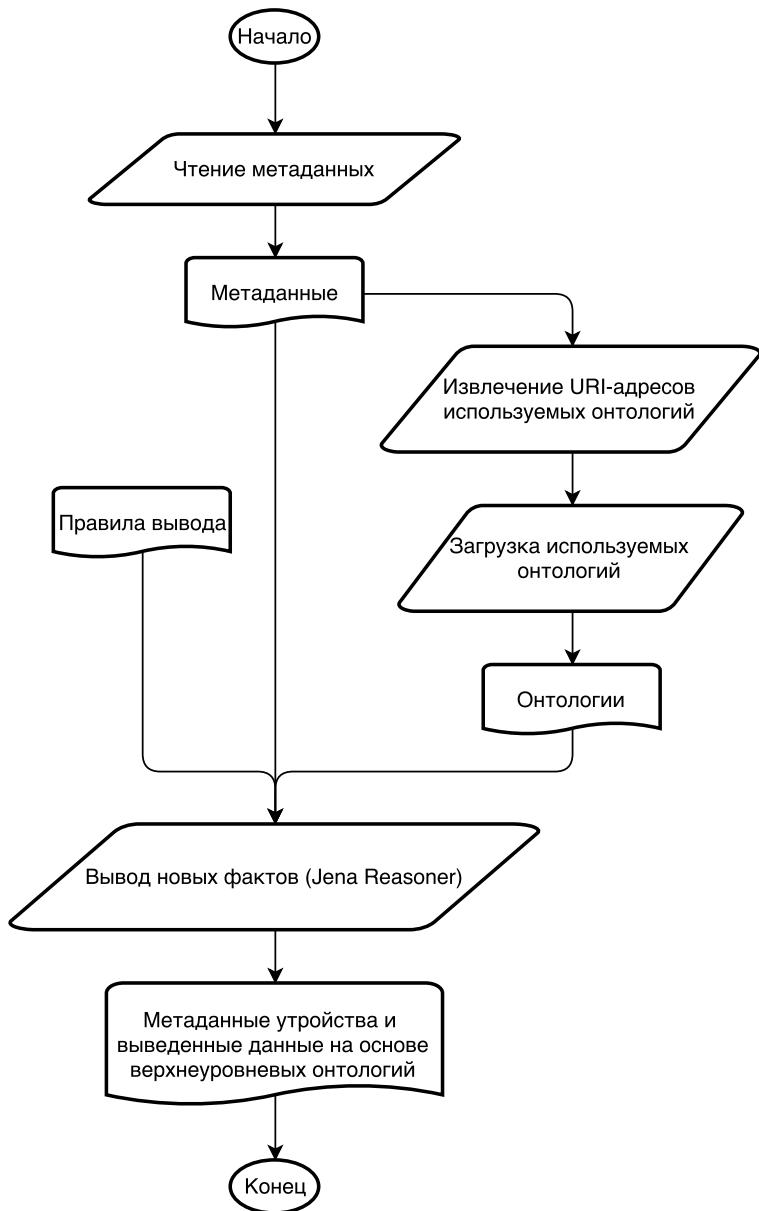


Рисунок 2.8 — Алгоритм вывода новых фактов из метаданных устройства и верхнеуровневых онтологий

онтология используемая этим устройством основана полностью на онтологии SSN, на рисунке 2.9 представлены RDF-данные в формате Turtle.

Этот пример отличается от предыдущего тем что в нем появляются факты об устройстве, которые связаны с верхнеуровневой онтологией, соответственно, это дает возможность выполнять запросы, которые до этого выполнить было нельзя. На рисунке 2.10 приведен простой SPARQL-запрос, который основан на онтологии SSN, и выводит список всех устройств в базе

```

1 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
2 @prefix ex: <http://example.semiot.ru/> .
3
4 ex:WeatherStation-01 a ex:WeatherStation , ssn:System ;
5     ssn:hasSubsystem ex:WeatherStation-01-Temp ;
6     ssn:hasSubsystem ex:WeatherStation-01-Humidity ;
7
8 ex:WeatherStation-01-Temp a ex:AirTemperatureSensor , ssn:Sensor ;
9     ssn:observes ex:AirTemperature .
10
11 ex:WeatherStation-01-Humidity a ex:AirHumiditySensor , ssn:Sensor ;
12     ssn:observes ex:AirHumidity .
13
14 ex:AirFeature a ssn:FeatureOfInterest ;
15     ssn:hasProperty ex:AirTemperature ;
16     ssn:hasProperty ex:AirHumidity .

```

Рисунок 2.9 — Пример описания датчика температуры и влажность

данных. Соответственно до логического вывода этот запрос вернул бы ноль результатов, а после возвращает URI погодного датчика *ex:WeatherStation-01*.

```

1 SELECT * WHERE {
2     ?system a ssn:System .
3 }
```

Рисунок 2.10 — Пример простого SPARQL-запроса на основе онтологии SSN

Метод описанный в разделе 2.2 и алгоритм описанный выше позволяют реализовать работу ПО ЭО ПАП на основе стандартных или де факто стандартных верхнеуровневых онтологий таким образом, что основные функции по сбору, хранению и анализу могут быть реализованы без привязки к определенной предметной области, в которой ЭО ПАП может быть использована.

2.4 Метод и алгоритм оценки производительности и корректности систем анализа связанных потоковых данных

Постоянное распространение и расширение сети Интернет приводит к все больше и большему распространению постоянно изменяющихся данных. Часто, для анализа данных требуется учитывать такую её характеристику

как время. Мы заинтересованы в том, что происходит прямо сейчас, чтобы сделать выводы о текущем или будущем состоянии системы. Области применения часто меняющихся потоков данных включают в себя, среди многих других, (а) интеграцию данных в контекст "умного" города и (б) экологический мониторинг, (в) управление общественным транспортом, (г) и здравоохранение.

Системы управления потоковыми данными (СУПД) и соответствующие языки запросов являются темой научных исследований уже достаточно продолжительное время, начиная с 2003 года [19]. СУПД берут своё начало с традиционных систем управления базами данных, но расширяют их тем, что работают с непрерывно изменяющимися данными. Так же они реализуют непрерывные запросы, которые выполняются непрерывно и предоставляют результаты каждый раз когда появляются новые данные [20].

Недавно сообщество исследователей технологий семантического веба заинтересовалось работой с потоковыми данными. Комбинирование потоковых данных с возможностями семантических технологий может открыть новые результаты и методы в управлении данными. Технологии семантического веба позволяют интегрировать различные источники данных с использованием федеративных запросов и логического вывода. Кроме того эти технологии позволяют работать с гетерогенными источниками данных, неполными данными и сложными моделями данных.

Уже существуют работы по реализации доступа к непрерывным потокам данных с помощью SPARQL-подобных запросов, но кроме только техническим различий в реализации, что приводит к различным характеристикам по производительности, главной проблемой является различия в понимании и реализации того как должны выполнять такие запросы. Более детально, различия появляются на уровне стратегий публикации результатов запросов, семантики операторов, операторов вывода и т.д. Все эти различия делают задачи по сравнению существующих СУПД достаточно нетривиально задачей.

Далее в этом подразделе описывается программный комплекс, который реализует метод и алгоритм оценки производительности и корректности систем анализа связанных потоковых данных.

В рамках данной работы, был разработан новый программный комплекс под названием YABench, который позволяет оценить как производительности СУПД, так и корректность результатов запросов. Он позволяет генерировать тестовый набор данных, определять тестовые сценарии и анализировать результаты этих тестовых сценариев. YABench предоставляет механизмы для создания воспроизводимых тестовых сценариев и для визуализации их результатов.

Исходный код и документация к данному программному комплексу опубликована под открытой лицензией в сети Интернет по следующему адресу: <https://github.com/YABench/yabench>.

YABench имеет модульную архитектуру, отделяющую программный комплекс от конфигурации тестового сценария, которая позволяет определять тестовые сценарии декларативным способом и одной командой запустить такой сценарий. Архитектура и алгоритм работы программного комплекса представлены на Рисунке 2.11.

Данный программный комплекс состоит из 4-х модулей:

- а) модуль генерации потоковых данных,
- б) система управления потоковыми данными, которая оценивается данным программным комплексом,
- в) модуль оценки корректности СУПД,
- г) модуль запуска программного комплекса, который контролирует весь цикл работы программного комплекса.

Конфигурация тестового сценария состоит из конфигурационного файла (config.json) и двух шаблонов запросов, а именно шаблон запроса для СУПД (engine.query) и шаблон запроса для модуля оценки корректности (oracle.query). Конфигурационный файл определяет набор тестов, который используют одни и те же шаблоны запросов, но с разными параметрами, такими как размер окна, шаг сдвига окна и другие.

Результатами каждого теста являются несколько файлов, а именно результаты проверки корректности (ORACLE_<имя теста>) и результаты производительности СУПД (P_<имя теста>).

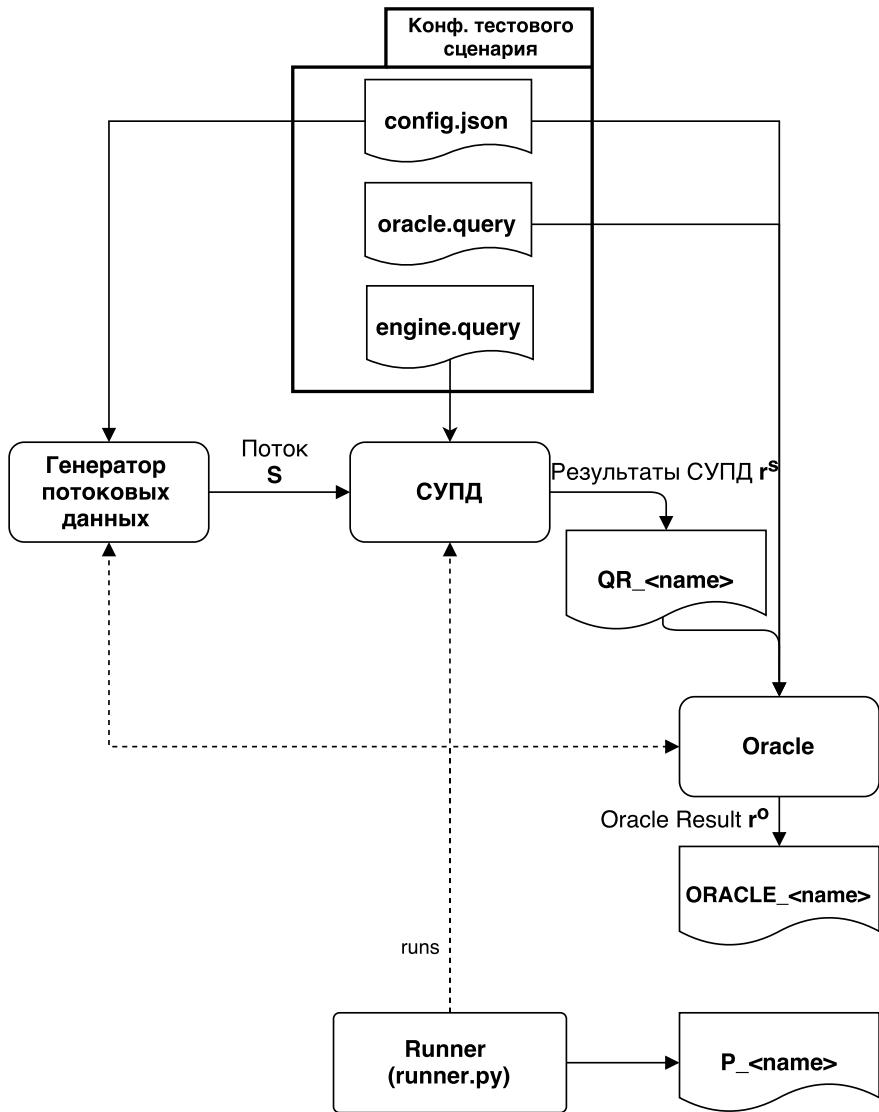


Рисунок 2.11 – Диаграмма архитектуры и алгоритма работы программного комплекса YABench

Модуль генерации потоковых данных создает набор данных, которые далее последовательно передаются в СУПД. Данная модель использует шаблоны в формате RDF и рандомизаторы для генерации потоковых данных, в качестве одного из шаблонов используются данные из датасета LinkedSensorDataset¹, которые так же были использованы в других работах. Эти данные состоят из измерений метеостанции во времена ураганов в США.

На Рисунке 2.12 представлена структура данных этого датасета. Основным элементом данной структуры является класс *weather:TemperatureObservation*, который описывает показание темпера-

¹http://wiki.knoesis.org/index.php/SSW_Datasets

туры. Это показание связано с непосредственно самим значением через свойство *om-owl:result*. Свойство *om-owl:observedProperty* указывает на свойство окружающей среды, которое было измерено элементом класса *ssw:System*. Физически этот элемент представляет реальное устройство, которое измеряет данные показания. Это устройство связано с показанием через свойство *om-owl:procedure*.

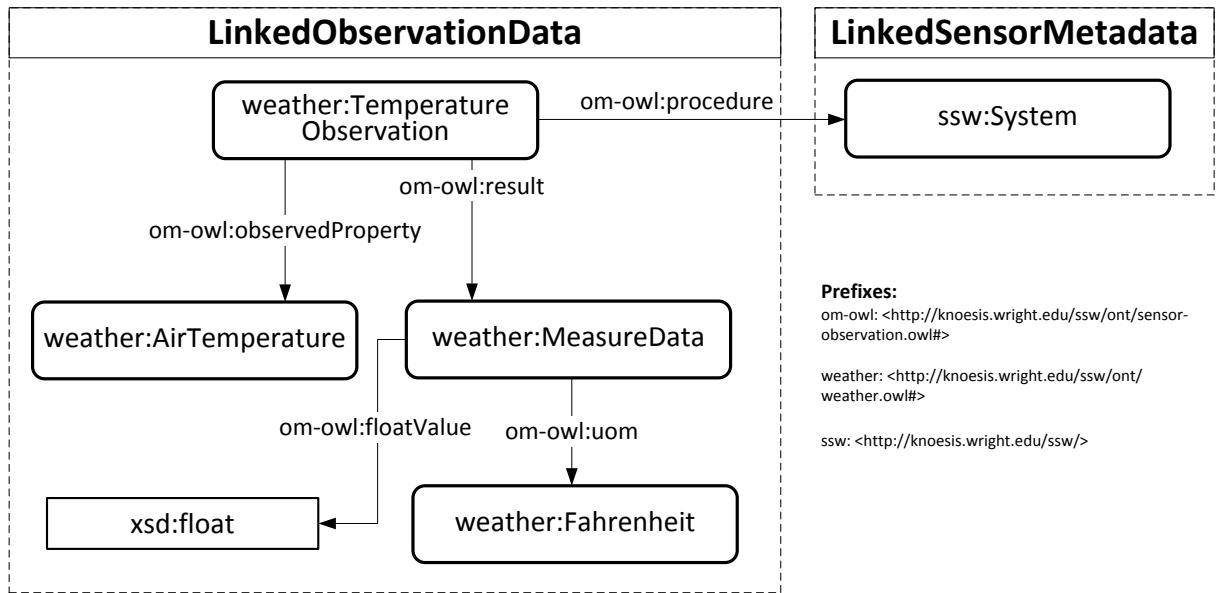


Рисунок 2.12 – Структура данных используемая модулем генерации потоковых данных на основе датасета LinkedSensorDataset

Процесс генерации потоковых данных полностью конфигурируем через несколько переменных: *s* количество метеостанций; *i* - интервал между показаниями одной станции; *d* - продолжительность генерируемых данных; *r* - начальное значение для рандомизатора. Соответственно нагрузка на СУПД может уменьшаться или повышаться за счет изменения параметров *s* и *i*. Комбинация переменных *s*, *i* и *r* гарантирует воспроизводимость набора данных, так как с одними и тем же значениями этих переменных наборы данных будут идентичными.

Для каждого СУПД требуется реализация адаптера, который реализует процесс передачи сгенерированных потоковых данных в СУПД. На данный момент реализованы адаптеры для следующих СУПД: C-SPARQL 0.9.5¹ и

¹<http://github.com/streamreasoning/CSPARQL-engine>

CQELS 1.0.0¹. Во время того как СУПД обрабатывает поступающие данные, YABench переодически замеряет показания производительности, такие как использование оперативной памяти, использование процессора и количество созданных потоков.

Модуль оценки корректности СУПД по результатам работы СУПД вычисляет корректности результатов запроса. Данный модуль считывает результаты (из файла QR_<имя теста>) запроса q , используя тот же самый набор потоковых данных, которые получала СУПД, и SPARQL запрос q' в соответствии с режимом вычисления результатов запроса [21], поддерживаемого данным СУПД. Кроме того на вход так же подаются значения размера и шага окна для данного запроса q .

Следующие режимы вычисления результатов запросов поддерживаются:

- а) По изменению содержимого: результат вычисляется только когда содержимое активного окна изменилось. Поддерживается СУПД CQELS.
- б) По закрытию окна: результат вычисляется только когда активное окно закрывается. Поддерживается СУПД C-SPARQL.

Модуль оценки корректности СУПД реализует следующий алгоритм, который представлен на рисунке 2.13:

- а) вычисляются границы активного окна, для которого новые результаты должны быть вычислены СУПД, на основе размера и шага окна, а так же режима вычисления результатов запроса,
- б) загрузка содержимого активного окна из входных потоковых данных, с учетом границ окна,
- в) вычисление ожидаемого результата запроса, на основе результата SPARQL запроса q' на содержимом активного окна,
- г) далее вычисляется схожесть (precision/recall) ожидаемого результата и результата СУПД,

¹<https://code.google.com/p/cqels>

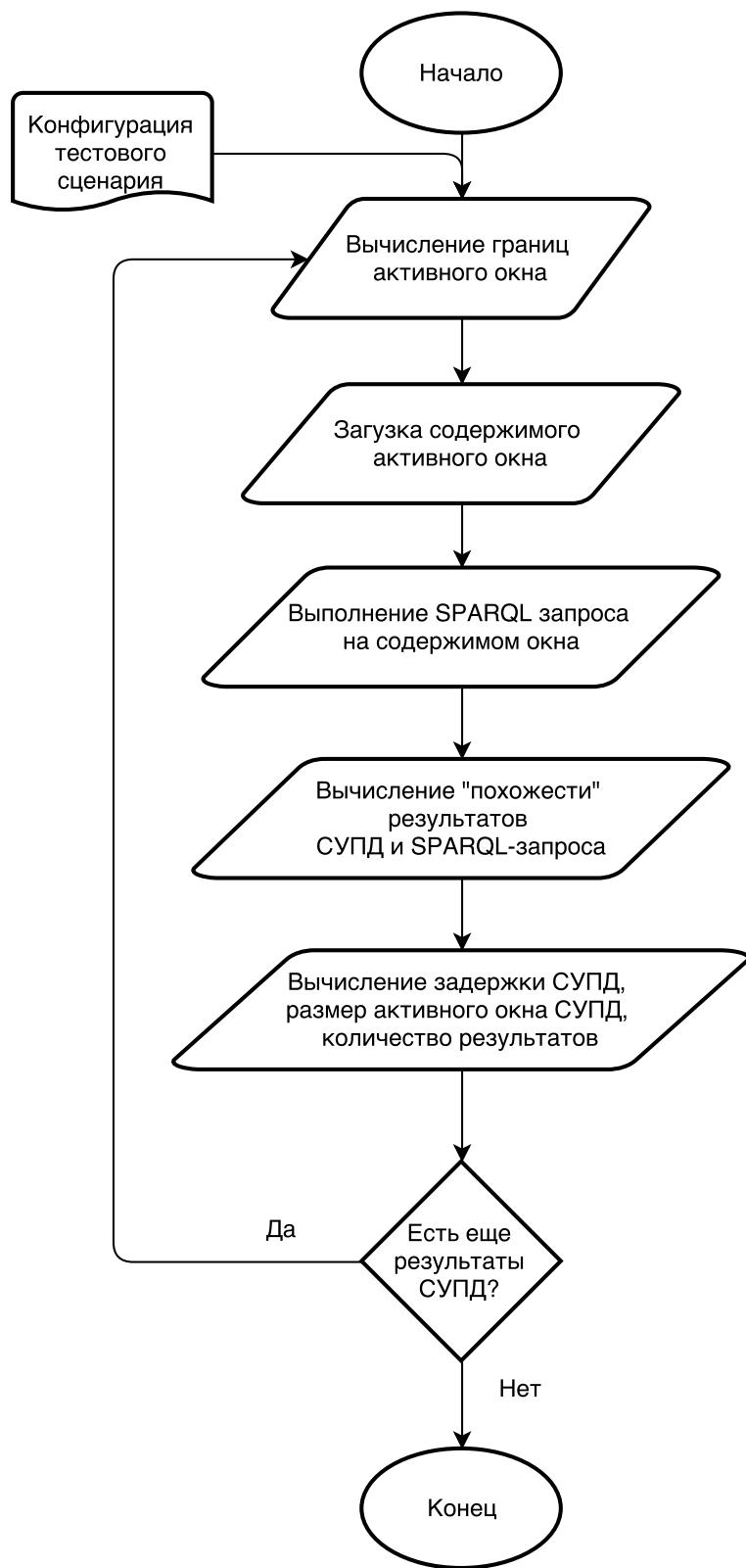


Рисунок 2.13 — Алгоритм оценки корректности СУПД

д) и наконец вычисляют остальные показатели, такие как время задержки, размер активного окна СУПД и количество результатов запроса.

2.5 Метод и алгоритмы анализа потоковых и статических данных распределенной сети электронных потребительских устройств

Процесс анализа данных необходим для выявления нештатных ситуаций как для самих ЭПУОВР, так и для умных сетей, включающих их в себя. Например, выявление в умных сетях электроснабжения проблем связанных с перепотреблением энергии поможет эффективно и быстро эту проблему устранить, сделав такую сеть отказоустойчивой.

Есть несколько готовых решений для анализа потоковых данных. Все они основаны на расширении языка SPARQL, но по-разному реализованы взаимодействия с данными. Ниже приведены основные технологии для анализа данных:

- a) C-SPARQL и C-SPARQL Engine. Continuous SPARQL (C-SPARQL)[22] - это язык для создания запросов к потоковым данным, расширяющий язык SPARQL. C-SPARQL Engine реализовывает язык C-SPARQL. Запросы, написанные на данном языке, позволяют наблюдать самые последние триплы в постоянно обновляющемся потоке. Суть работы C-SPARQL Engine в постоянном создании временных RDF-снимков (snapshot) и применения к ним стандартных SPARQL-запросов. C-SPARQL Engine использует в своей основе Esper и Apache Jena-ARQ. C-SPARQL использует принцип черного ящика, делегируя обработку запросов другим движкам.
- б) CQELS и CQELS-QL. CQELS-QL (Continuous Query Evaluation over Linked Streams)[23] – это декларативный язык запросов, основанный на грамматике языка SPARQL 1.1. Также как и C-SPARQL, расширяет стандартный язык SPARQL, добавляя новые операторы запросов потоковых данных. В отличие от C-SPARQL, CQELS использует принцип белого ящика, изначально реализуя все операторы, тем самым избегая закрытости системы и больших накладных расходов. Во время выполнения запроса CQELS постоянно изменяет порядок операторов в соответствии с эвристикой, улучшая тем самым скорость исполнения запроса и уменьшая его сложность. Также CQELS кеширует данные,

временно сохраняя их на носителе информации, для увеличения скорости исполнения и уменьшения размеров потребляемых ресурсов.

в) SPARQLstream и Morph-streams обработчик. SPARQLstream[24] – это расширение языка SPARQL, поддерживающих операторы работы над RDF потоками такими как временные окна. SPARQLstream очень похож на C-SPARQL за исключением двух моментов: SPARQLstream предоставляет синтаксис, полностью поддерживающий R2S оператор, в то время как C-SPARQL нет. Также C-SPARQL имеет функцию работы со временем, позволяя выполнять все основные временные сравнения потоковых триплов, а SPARQLstream такой возможности не имеет. Morph-streams обработчик – является обработчиком запросов к RDF-поток, использующий методы доступа к данным на основе онтологий [25] для непрерывного выполнения запросов языка SPARQLstream. SPARQLstream запросы переписываются в прямые запросы для потоков данных.

г) EP-SPARQL и ETALIS. EP-SPARQL расширяет язык SPARQL, добавляя возможности обработки событий (Event Processing). ETALIS (Event TrAnsaction Logic Inference System)[26] – это движок, основанный на правилах, для обработки событий и ризонинга потока. В качестве входных параметров он принимает RDF-утверждения, аннотированные двумя временными отметками. Пользователи могут уточнить задачи обработки событий, используя два декларативных языка: ETALIS Language for Events (ELE) и Event Processing SPARQL (EP-SPARQL)[27]. Оба языка имеют схожую семантику: комплексные события являются производными от простых событий, используя дедуктивные правила языка Пролог. ETALIS поддерживает не только типичные конструкции обработки событий, но и рассуждения о событиях.

Рассмотрев приведенные выше технологии, необходимо выбрать одну из них. Для этого были проанализированы несколько бенчмарков, например¹. По их результатам было выяснено, что CQELS обладает значительным преимуществом перед другими технологиями. Например, CQELS имеет одну из

¹A benchmark for Linked Stream Data processing engines, <https://code.google.com/p/lbench/>

самых высоких скоростей работы и она практически не меняется на значительном увеличении количества поступающих в движок триплов, в отличии от других технологий. За счет этого в нашей работе было принято решение воспользоваться готовой технологией CQELS. Проблемой при его использовании является протяженное время появления обновлений, поэтому некоторые моменты и проблемы, возможно, придется устранять самостоятельно.

Несмотря на все предоставляемые преимущества, необходимо отметить, что в исходном виде CQELS использоваться в ПО ЭО ПАП не может, поскольку необходимо создать прослойку, поставляющую в движок данные в требуемом формате, получала и добавляла в движок поступающие запросы и преобразовывала и отправляла в шину сообщений результат выполнения этих запросов. Поэтому необходим дополнительный сервис, интегрирующий в себе CQELS. Архитектура использования CQELS приведена на рисунке 2.14.

Алгоритм для анализа данных представляет из себя совокупность процессов интеграции данных с последующей передачей их в CQELS и добавления в него же запросов, написанных согласно расширенной грамматике языка SPARQL 1.1 и описывающих процессы нештатных ситуаций или те процессы, которые необходимо проанализировать и перехватить. Сам алгоритм для анализа данных приведен на рисунке 2.15.

Как можно заметить из приведенной архитектуры, потоковые данные собираются с ЭПУОВР специальным подписчиком, после чего адресуются по шине сообщений в CQELS. Если рассматривать этот процесс шире, то сервис для анализа данных обращается к SPARQL-сервису с запросом об активных ЭПУОВР, после чего осуществляет на них подпиську, обращаясь к подписчику на потоковые данные. В итоге все потоковые данные с ЭПУОВР поступают в сервис для анализа данных.

Для интеграции статических и потоковых данных и их последующего анализа, необходимо получить эти данные из мест их хранения. В данном случае, статические данные находятся в базе данных и извлекаются, используя SPARQL-запросы, а потоковые данные можно получить, совершив подпиську на ЭПУОВР. При этом все данные будут отправляться и, соответственно, поступать из шины сообщений. Сами данные представлены в расширенном

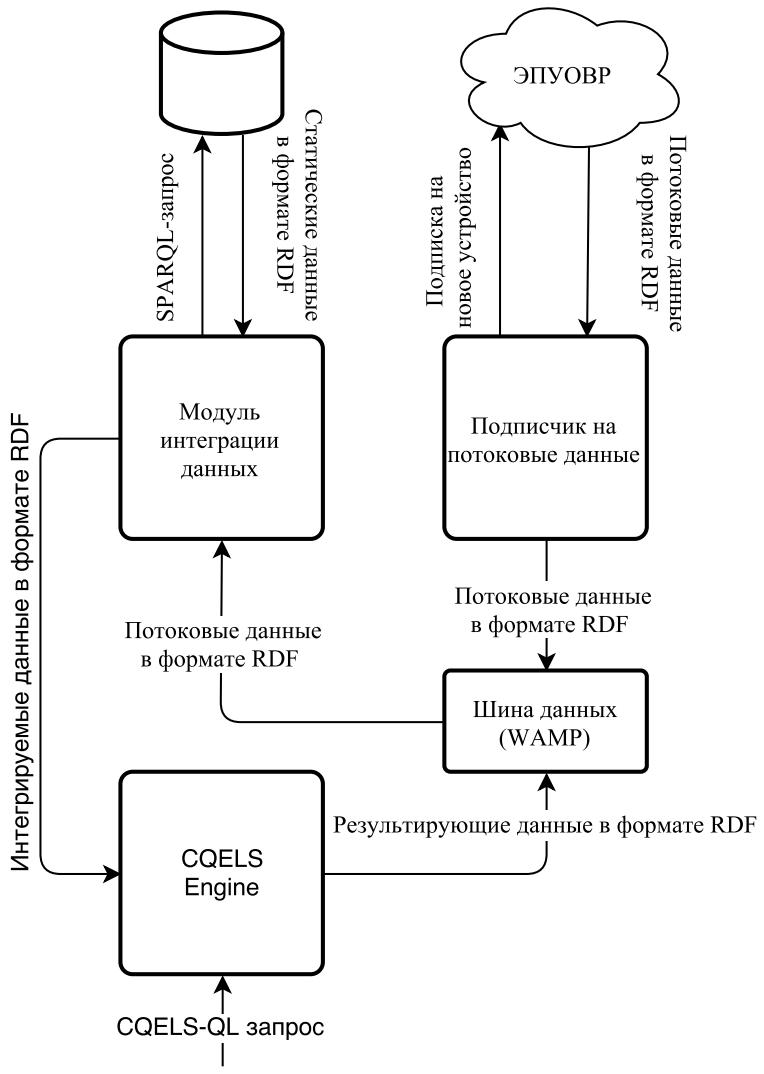


Рисунок 2.14 — Системная архитектура анализа данных

формате RDF, добавляющем временную метку. Алгоритм получения данных представлен на рисунке 2.16.

Пример SPARQL-запроса для получения списка активных устройств приведен на рисунке 2.17

2.6 Метод обнаружения и генерации событий предметной области на основе анализа потоковых и статических данных

Для обнаружения и генерации событий на основе анализа данных как было написано в разделе 2.5 используется CQELS. Использование этой технологии позволяет генерировать события, используя два вида SPARQL-запросов: SELECT и CONSTRUCT. SELECT-запрос извлекает необработан-

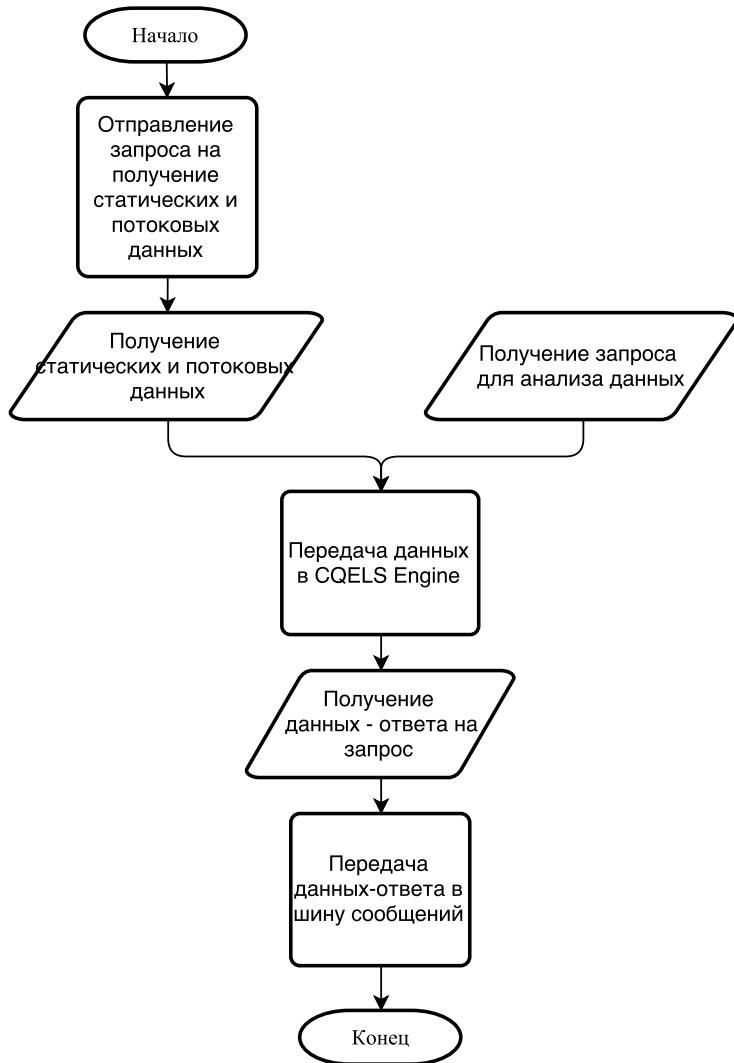


Рисунок 2.15 — Алгоритм анализа данных

ные значения из точки доступа SPARQL и возвращает результаты в формате таблицы. В отличие от него, CONSTRUCT-запрос извлекает информацию из точки доступа SPARQL в формате RDF и преобразовывает результаты к определенной форме. Таким образом, CONSTRUCT-запросы позволяют преобразовывать результирующие данные к определенному виду. То есть, описанные выше запросы передаются в движок и определяют формат выходных данных. Если в CQELS поступают данные, удовлетворяющие запросам, то происходит генерация URI события и полученные выходные данные записываются в переменную *alerts* и отправляются в шину сообщений. Пример простого SELECT-запроса для выявления так называемого эффекта "перетопа", когда температура превышает допустимый диапазон значений, приведен на рисунке 2.18.

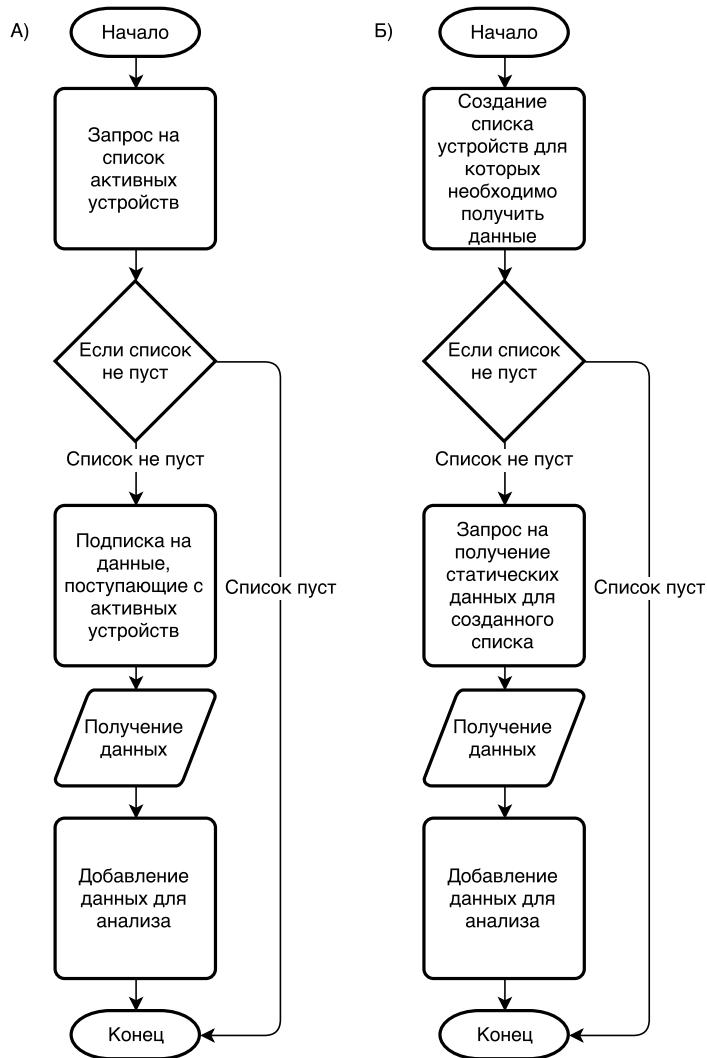


Рисунок 2.16 — Алгоритм интеграции данных: А) для потоковых данных; Б) для статических данных

```

1 prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
2 prefix ssnc: <http://purl.org/NET/ssnext/communication#>
3
4 SELECT ?q
5 WHERE{
6     ?x a ssn:Sensor ; ssnc:hasCommunicationEndpoint ?q .
7     ?q ssnc:protocol "WAMP" .
8 }

```

Рисунок 2.17 — Пример запроса активных устройств

Регистрация запроса в CQELS происходит следующим образом: сначала происходит проверка корректности синтаксиса запроса, после успешной проверки, начинается семантический разбор запроса, во время которого под-

```

1 PREFIX hmtr: <http://purl.org/NET/ssnext/heatmeters#>
2 PREFIX meter: <http://purl.org/NET/ssnext/meters/core#>
3 PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5
6 SELECT ?meter ?value {
7   STREAM ?stream [NOW] {
8     ?obs a emtr:HeatObservation ;
9     ssn:observedBy ?meter ;
10    ssn:observationResult ?result .
11    ?result ssn:hasValue ?v .
12    ?v meter:hasQuantityValue ?value .
13  }
14  FILTER(?value > 25)
15 }
```

Рисунок 2.18 — Пример запроса для выявления перетопа

гружаются объявленные в запросе источники для определения используемых онтологий. При необходимости извлечения дополнительных сведений из уже существующих данных (статических), выполняются дополнительные запросы к статическим данным и добавляются необходимые данные. После регистрации запроса, создается слушатель, отображающий результат выполнения запроса, то есть выдающий данные согласно схеме, объявленной в запросе. Соответственно, после всех описанных действий в движок начинают передаваться потоковые данные, которые сопоставляются запросу, при удовлетворении его условий через слушателя данные выводятся в требуемом формате из CQELS.

К сожалению, SELECT-запрос подходит не для всех ситуаций, например, через него нельзя обратиться к статическим данным и определить более сложный формат для выходных данных, а не выводить их в виде таблицы. Это необходимо, например, для подробного получения описания произошедшей ситуации, такой как превышение допустимого уровня напряжения в умных сетях электроснабжения. В такой ситуации необходимо выдать то, что произошло именно это событие, а не какое-либо другое и передать все необходимые данные для локализации произошедшего события. Для этого и применяется более сложные в построении CONSTRUCT-запросы. Они строятся подобно SELECT-запросам, но предлагают задать шаблон, согласно которо-

му будет возвращаться RDF граф. Пример подобного запроса приведен на рисунке 2.19.

```
1 CONSTRUCT {  
2     ?alert a :TooHighVoltageValue ;  
3     dul:hasEventDate ?time ;  
4     dul:involvesAgent ?meter .  
5 }  
6 FROM NAMED <http://ex.com/sparql>  
7 WHERE {  
8     GRAPH <http://ex.com/SmartMeters/> { ?meter em:hasStream ?stream }  
9     STREAM ?stream [NOW] {  
10         ?observation a em:PolyphaseVoltageObservation ;  
11         ssn:observationResultTime ?time .  
12         ?output a em:PolyphaseVoltageSensorOutput ;  
13         ssn:isProducedBy ?meter ;  
14         ssn:hasValue ?value .  
15         ?value em:hasQuantityValue ?qvalue .  
16     }  
17     FILTER(?qvalue >= (220 + 220*0.1))  
18     BIND (IRI(CONCAT(STR(?meter) , '/alerts /' , STR(?time))) AS ?alert)  
19 }
```

Рисунок 2.19 — Пример запроса для выявления перенапряжения

2.7 Результаты

В данном разделе отчета описаны методы и алгоритмы нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things).

Разработанные методы и алгоритмы соответствуют требованиям технического задания в части реализации процессов нормализации и анализа больших массивов гетерогенных данных электронных потребительских устройств. Использование онтологий и модели данных RDF в сочетании с методами и алгоритмами описанными в разделе 1 значительно упрощает доступ к данным электронных потребительских устройств от разных производителей и различных типов.

Кроме того в Приложении Б представлены отчетные и справочные материалы по докладу на 12-й международной конференции Extended Semantic Web Conference 2015 (ESWC 2015) под названием «Demo: YABench - Yet Another RDF Stream Processing Benchmark», который освещает работу описанную в подразделе 2.4.

В результате работы были разработаны и исследованы необходимые методы и алгоритмы, соответствующие требованиям Технического задания.

3 Разработка математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data)

3.1 Назначение и состав математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data)

Интерактивная визуализация комплексных и разнотипных данных распределенной сети электронных потребительских устройств позволяет эффективно представлять информацию для человека, тем самым упрощая её понимание. В рамках данного ПНИ, стоит задача по визуализации разнотипных данных, таких как: (а) обычные статические данные, например метаданные устройств; (б) временные данные, например показания сенсоров; (в) пространственные данные, например местоположение устройства. В рамках данной работы поставлена задача разработать методы и алгоритмы интерактивной визуализации обеспечивающей выполнение следующих требований:

- а) сбор данных для построения графических форм визуализации конфигурации сети пользовательских устройств не более, чем за 15 секунд,
- б) время отклика системы для операций навигации по интерактивным графическим формам – не более чем за 5 секунд,
- в) время построения визуальных форм для графического представления данных электронных потребительских устройств – не более 3 секунд.

На основе результатов аналитического обзора современной научно-технической литературы, а также анализа применимости технологии связанных данных (Linked Data) для задачи интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных, выполненных на 1-м этапе данного ПНИ [1], были

решено для доступа к данным электронных потребительских устройств использовать технологии связанных данных и современные технологии визуализации, с помощью технологии HTML5 и SVG.

Итого в рамках данной работы было решено разработать следующие методы и алгоритмы, обеспечивающие необходимую функциональность и соответствующих требованияния технического задания:

- а) метод и алгоритм доступа к статическим данным на основе семантических веб технологий и технологии связанных данных,
- б) алгоритм доступа к историческим данными распределенной сети электронных потребительских устройств,
- в) метод и алгоритм доступа к потоковым данным распределенной сети электронных потребительских устройств,
- г) алгоритм визуализации потоковых и исторических данных.

3.2 Метод и алгоритм доступа к статическим данным на основе семантических веб технологий и технологии связанных данных

В связи со все большим распространением технологий сети Интернет, донесение той или иной информации до конечного потребителя становится все менее ресурсозатратной задачей, а развитие сопутствующих веб-технологий - помогает делать это все более наглядными способами и в более полном виде. В частности, утверждение HTML5-стандартов графики в вебе, таких, как Canvas¹ или WebGL², адаптация и развитие векторной графики SVG в веб-документах, а также инструменты для работы с ними - как упрощающие и генерализующие API (Raphael.js, Paper.js, Three.js), так и предоставляющие готовые инструменты для реализации тех или иных подходов к визуализации (D3.js, Chart.js) - позволяют реализовать прикладные пользовательские приложения для работы с самыми различными данными и их аспектами.

Одним из преимуществ использования технологий семантического веба - в предоставлении открытых программных интерфейсов для работы со

¹HTML Canvas 2D Context, <http://www.w3.org/TR/2dcontext/>

²WebGL Specification, <https://www.khronos.org/registry/webgl/specs/1.0/>

структуризованными нормализованными данными посредством так называемых SPARQL Endpoints - точек входа, реализованных согласно некоторому унифицированному стандарту и доступных посредством протокола HTTP. Сформировав определенный SPARQL-запрос к таким точкам входа и соответствующий ему HTTP-запрос в REST-стиле на получение или изменение данных, сторонние системы могут выполнять над этими данными различные операции. Причем, данные запросы могут быть более сложными, чем, например, SQL-запросы к реляционным базам данных; в частности, при формировании ответа возможно задействие техники так называемого reasoning - получение информации из данных и описывающей их онтологии, не выраженной в них явным образом.

Наличие доступа к таким точкам входа посредством протокола HTTP является важным преимуществом над любыми другими способами организации транспорта между различными системами ввиду его очень широкой распространенности и наличии различных программных имплементаций; в частности, он является основным при работе с прикладными веб-приложениями. В 2000 году компанией Microsoft в программном обеспечении Outlook Web Access был внедрен новый API под названием XMLHttpRequest¹, позволяющий обмениваться данными между клиентом и сервером без перезагрузки страницы, позднее поддержанный прочими браузерными вендорами, такими, как Mozilla и Safari, а также оформленный в виде спецификации W3C. Адаптация данной технологии, получившей обобщенное название AJAX - Asynchronous JavaScript and XML - и прочих принципов построения интерактивных веб-приложений, объединенных концепцией DHTML (Dynamic Hyper Text Markup Language), привело к бурному развитию как прикладных веб-приложений, так и всей отрасли веб-разработки.

На сегодняшний день технология XMLHttpRequest до сих пор не является до конца кроссбраузерной; кроме того, сам браузерный API, предоставляемый посредством такой технологии, является громоздким и неудобным. Вследствие этого, широкое распространение получили библиотеки и фреймворки, содержащие функции-обертки для работы с AJAX. Одним из таких фреймворков для построения Rich Internet Application - современных

¹ XMLHttpRequest Level 1, <http://www.w3.org/TR/XMLHttpRequest/>

интерактивных веб-приложений с клиентской маршрутизацией, возможностью асинхронного обмена данными с удаленными серверами и работы в offline-режиме - является Angular.js¹, разработанный при непосредственном участии компании Google. Angular.js предоставляет возможность для построения клиентских JavaScript приложений с применением паттерна MVVM - Model-View-ViewModel, являющимся разновидностью паттерна MVC - Model-View-Controller, а также имеет встроенную поддержку таких инструментов , как Two-Way Data Binding, Promises, клиентскую маршрутизацию и является одним из самых популярных в своем роде на данный момент. Использование Angular.js позволяет ускорить разработку насыщенных пользовательских приложений.

Сценарий работы прикладных веб-приложений со статическими данными на основе семантических технологий в части работы с удаленными SPARQL endpoints в общем случае выглядит следующим образом: при инициализации приложения выполняются HTTP - запросы к точкам доступа, содержащие в качестве тела SPARQL-запрос на получение тех или иных сущностей. Пример такого запроса на получение всех датчиков, зарегистрированных в системе, приведен на рисунке 3.1.

```
1 SELECT ?label ?uri
2 WHERE {
3     ?uri a ssn:System ;
4         a ?type .
5     ?uri rdfs:label ?label .
6     FILTER NOT EXISTS {
7         ?subClass rdfs:subClassOf ?type .
8         ?uri a ?subClass .
9         FILTER (?subClass != ?type)
10    }
11 }
```

Рисунок 3.1 — SPARQL - запрос на получение списка зарегистрированных в системе ЭПУОВР

Протокол SPARQL 1.1² описывает три способа передать SPARQL-запрос посредством протокола HTTP:

¹<https://angularjs.org/>

²<http://www.w3.org/TR/sparql11-protocol/>

- a) Посредством GET-запроса, передав текст запроса в виде query-параметра.
- б) Посредством POST-запроса, передав URL-энкодированный текст запроса в теле с указанием типа application/x-www-form-urlencoded.
- в) Посредством POST-запроса, передав текст запроса в теле в указанием типа application/sparql-query.

В случае корректно сформированного запроса в ответ придет либо сериализованный в нативный для веб-разработки формат JSON(JavaScript Object Notation) результат (если выполненный запрос был типа SELECT или ASK), либо RDF - граф (в случае запросов типа DESCRIBE или CONSTRUCT). При разработке прикладных веб-приложений в подавляющем большинстве случаев SPARQL - точки входа рассматриваются в качестве источника данных, и большинство запросов к ним имеют тип SELECT.

Получив список сущностей при инициализации приложения, а также минимально необходимой метаинформации о них (например, URI для дальнейшей работы), можно отобразить их в виде некоторой таблицы.

Алгоритм получения статических данных представлен на рисунке 3.2 и описывается следующими шагами:

- а) *start* - начало временного периода просматриваемых показаний в формате Unix Timestamp.
- б) *end* - конец временного периода просматриваемых показаний в формате Unix Timestamp.
- в) *sum* - ключ, идентифицирующий конкретный счетчик конкретного ЭПУОВР.

Таким образом, используя технологии AJAX и удаленные SPARQL-точки доступа, мы имеем возможность создания различных прикладных клиент-серверных веб-приложений на основе семантических данных, предоставляемых данными точками доступа.

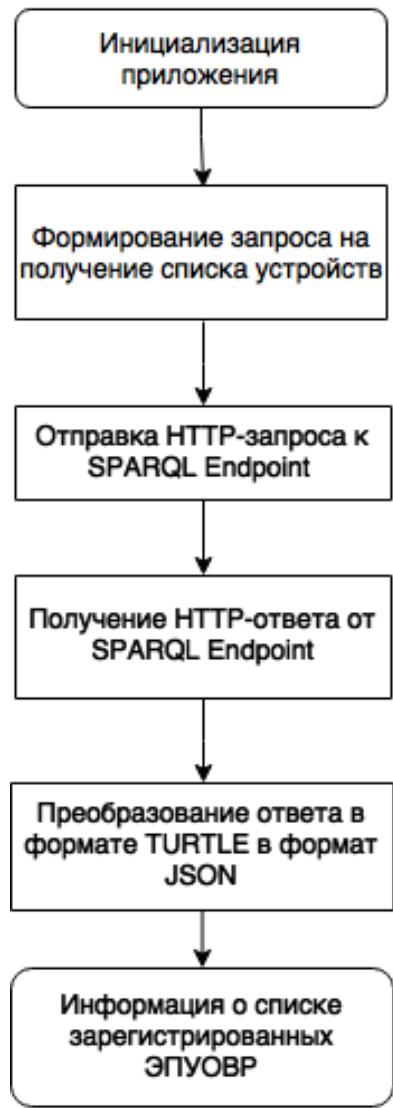


Рисунок 3.2 — Алгоритм получения статических данных о зарегистрированных ЭПУОВР

3.3 Алгоритм доступа к историческим данными распределенной сети электронных потребительских устройств

Помимо общей картины, создаваемой совокупностью ЭПУОВР в виде линейной таблицы или отображением их на карте, необходимо также обозревать каждый счетчик детально - в том числе, его архивные показания. Для работы с архивными данными из веб-приложения, необходимо некоторое внешнее хранилище, предоставляющее соответствующий HTTP API для получения данной информации.

Рассмотрим некоторые возможные реализации таких хранилищ. Наиболее распространенный подход к хранению и работы с данными на сего-

дня являются реляционные базы данных, такие, как MySQL, PostgreSQL, Oracle. Однако, такие инструменты не решают проблему масштабирования при больших объемах информации (каковыми являются непрерывные показания тысячи датчиков). Более подходящими для хранения и обработки больших объемов данных являются так называемые NoSQL - хранилища - такие, как MongoDB и CouchDB. Однако, лучше всего задача континуальной записи большого количества показаний и их времени ложится на так называемые Time Series Databases. В частности, одна из open - source реализаций - OpenTSDB¹ - предоставляет встроенный функционал горизонтального масштабирования, собственный веб-интерфейс для мониторинга, а также - HTTP API, заточенное под возможность выборки данных по временным параметрам. Конкретно для получения архивных показаний по конкретному датчику выбранного ЭПУОВР необходимо указать следующие параметры:

- а) *start* - начало временного периода просматриваемых показаний в формате Unix Timestamp.
- б) *end* - конец временного периода просматриваемых показаний в формате Unix Timestamp.
- в) *sum* - ключ, идентифицирующий конкретный счетчик конкретного ЭПУОВР.

Для предоставления удобного для пользователя метода выбора этих параметров были выбраны элементы управления в виде двух виджетов - полей для выбора даты и времени, внешний вид которых представлен на рисунке 3.3

Информация о TSDB-ключе, идентифицирующем счетчик, задается при регистрации ЭПУОВР в платформе и соответствует точке коммуникации по протоколу WAMP, соответствующей объекту предиката *sshcom:hasCommunicationEndpoint*. Чтобы получить все точки входа к сенсорам конкретного ЭПУОВР, зная его URI, достаточно выполнить запрос к SPARQL-точке доступа. Пример такого запроса на получение всех счетчиков ЭПУОВР с URI *coap://10.0.0.7:60000/meter* приведен на рисунке 3.4.

¹<http://opentsdb.net/>

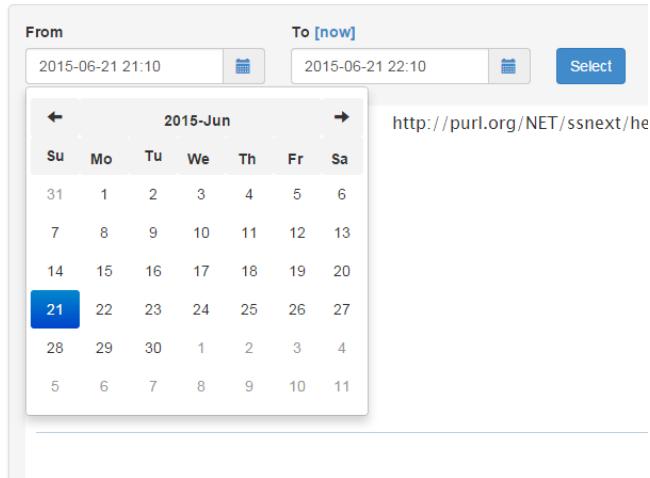


Рисунок 3.3 – Графические элементы управления выбора промежутка времени показаний датчиков ЭПУОВР

```

1 SELECT ?endpoint ?type {
2   <coap://10.0.0.7:60000/meter> ssn:hasSubsystem ?subsystem .
3   ?subsystem ssn:observes ?type .
4   ?subsystem ssncom:hasCommunicationEndpoint ?endpoint .
5   ?endpoint saref:hasState ?state .
6   ?endpoint ssncom:protocol 'WAMP' .
7 }
```

Рисунок 3.4 – SPARQL - запрос на получение всех точек входа к сенсорам конкретного ЭПУОВР

Сформировав GET-запрос с описанными выше данными параметрами и отправив его посредством HTTP в OpenTSDB, мы получим в ответ - при наличии показаний данного счетчика в этот промежуток времени - сообщение в формате JSON с маппингом времени и значений снятых с конкретного сенсора показаний.

Алгоритм получения исторических данных представлен на рисунке 3.5

3.4 Метод и алгоритм доступа к потоковым данным распределенной сети электронных потребительских устройств

В сфере веб-технологий после адаптации технологий асинхронного получения данных от сервера клиентом возникла актуальная задача оперативной доставки информации в другом направлении - нотификации клиента о

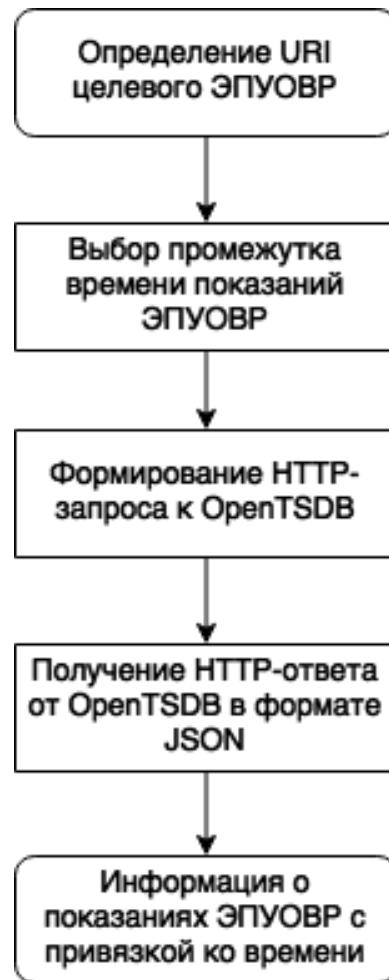


Рисунок 3.5 — Алгоритм получения исторических показаний с зарегистрированных ЭПУОВР

некоторых событиях на сервере. Первоначально эта задача решалась периодической отправкой клиентом HTTP-запросов на предмет изменений, произошедших на сервере. Это обеспечивало коммуникацию в псевдо - реальном времени, и так называемый overhead такого метода был прямо пропорционален количеству запросов в единицу времени (и обратно пропорционален – промежутку времени t , определяющему соответствие реальному времени).

С развитием клиентских приложений был разработан метод клиент-серверной коммуникации под названием СОМЕТ - первоначально ориентированный под взаимодействие браузерных Java - аплетов с J2SE - сервером, а позже - реализованный под множество программных языков и платформ. Суть метода - в установке "удерживаемых" (long-held) соединений (TCP или HTTP) между коммуницирующими сторонами, инициатором которого, как правило, выступает клиент. После возникновения какого-либо события на

сервере (например, регистрации новой сущности), сервер отсыпал данные в текущее соединение и закрывал его, а клиент после получения ответа инициировал коннекцию снова. Данная модель позволила приблизить время коммуникации к реальному, но проблема переизбытка траффика и вычислительных ресурсов, тратящихся на его поддержание, оставалась - в подавляющем большинстве случаев в 99% времени взаимодействия не производилось, в то время как технология COMET не предполагает долгосрочного удержания таких соединений - по истечении некоторого таймаута, клиент должен был переинициализировать соединение даже в том случае, если ответ не был получен.

Одним из нововведений HTML5 как набора стандартов стал протокол WebSocket¹, окончательно стандартизованный в 2011 году. Данный протокол описывает полнодуплексную коммуникацию между WebSocket-клиентом и WebSocket-сервером посредством единоразово устанавливаемого TCP - соединения, что значительно увеличивает его производительность в сравнении с технологией COMET.

Для установки WebSocket - соединения клиенту необходимо сформировать и отправить на сервер определенный HTTP - запрос (WebSocket Protocol Handshake), после чего будет инициализирована сессия, в рамках которой клиент и сервер могут обмениваться сообщениями.

Протокол WebSocket, в целом, является низкоуровневым и лишь предоставляет реализацию "Publish - Subscribe" паттерна. В то же время, для некоторых задач является актуальной поддержка на уровне протокола техники RPC (Remote Procedure Call) для формирования запросов не только на получение данных, но и на выполнение каких-либо операций, а также - маршрутизация сообщений по разным темам. Для этой цели были разработаны различные протоколы поверх WebSocket, такие, как WAMP (Web Application Messaging Protocol) или STOMP (Streaming Text Oriented Messaging Protocol). В рамках данной работы используется единая шина сообщений в рамках всей платформы, функционирующая по протоколу WAMP.

В то время, как протокол WebSocket реализуется на уровне браузерного API и на сегодняшний день поддержан всеми актуальными десктопными

¹<https://tools.ietf.org/html/rfc6455/>

или мобильными браузерами, протоколы типа WAMP или STOMP необходимо реализовывать на уровне языка JavaScript. Наиболее распространенной реализацией протокола WAMP на языке JavaScript является кроссплатформенная библиотека Autobahn.js¹. Пример установки соединения и подписки на регистрацию новых датчиков в системе приведен на рисунке 3.6.

```
1 var connection = new autobahn.Connection({
2     url: 'ws://localhost:8080/ws',
3     realm: 'realm1'
4 });
5
6 connection.onopen = function(session) {
7     session.subscribe('ru.semiot.devices.newandobserving', callback);
8 };
9
10 connection.open();
```

Рисунок 3.6 — Пример установки WAMP - соединения с сервером с помощью библиотеки Autobahn.js

В предложенной модели платформы, при регистрации нового ЭПУОВР в WAMP - шину сообщений транслируется его RDF - описание. Алгоритм Пример сообщения о зарегистрированном ЭПУОВР в формате Turtle приведен на рисунке 3.7.

На рисунке 3.7 видно, что при регистрации веб-приложение получает через WAMP информацию о URI ЭПУОВР, его названии, внутреннем и внешнем расположении, а также его сенсорах и точках подключения к ним по протоколу СоАР. Данной информации достаточно, чтобы отобразить новый ЭПУОВР в таблице уже зарегистрированных устройств.

Как уже было описано выше, при работе с конкретным ЭПУОВР необходимо получить дополнительную информацию о его сенсорах, выполнив определенный SPARQL-запрос к внешней SPARQL-точке доступа. Помимо прочего, в результате такого запроса присутствует специальный идентификатор *?endpoint*, являющийся также отдельным топиком вшине сообщений. По данному топику в системе транслируются в реальном времени показания конкретного сенсора. Получив этот идентификатор, клиентское приложение

¹<http://autobahn.ws/js/>

```

1 @prefix hmtr: <http://purl.org/NET/ssnext/heatmeters#> .
2 @prefix ssncom: <http://purl.org/NET/ssnext/communication#> .
3
4 <coap://10.1.1.1:6500/meter> a hmtr:HeatMeter ;
5   rdfs:label "Heat Meter #6500" ;
6   ssn:hasSubsystem <coap://10.1.1.1:6500:6500/meter/temperature> ;
7   ssn:hasSubsystem <coap://10.1.1.1:6500:6500/meter/heat> ;
8   limapext:isOccupantOf [ rdf:type limap:Room ;
9     limap:hasLocalCoordinates [ rdf:type limap:LocalCoordinates ;
10    geosparql:hasGeometry "POLYGON((3.976532 0,
11                               6.765645 0,
12                               6.765645 2.273458,
13                               3.976532 2.2734589))"^^geo:wktLiteral .
14   ] ;
15   limap:isLocated [ rdf:type limap:EscapePlan ;
16     limap:isEscapePlanOf [ rdf:type limap:Floor ;
17       dul:hasLocation [ geo:floor "4"^^xsd:int ] ,
18       limap:isFloorIn [ rdf:type limap:Building ;
19         geosparql:hasGeometry "POLYGON((-81.587906 45.336702,
20                               -81.148453 39.774769,
21                               -69.964371 39.30029,
22                               -70.403824 45.58329,
23                               -81.587906 45.336702))"^^geo:wktLiteral .
24     ] .
25   ] .
26 ] .
27 ] .
28 <coap://10.1.1.1:6500:6500/meter/temperature> a ssn:Sensor ;
29   ssn:observes hmtr:Temperature ;
30   ssncom:hasCommunicationEndpoint <coap://10.1.1.1:6500/meter/temp/obs> .
31 <coap://10.1.1.1:6500:6500/meter/heat> a ssn:Sensor ;
32   ssn:observes hmtr:Heat ;
33   ssncom:hasCommunicationEndpoint <coap://10.1.1.1:6500/meter/heat/obs> .
34 <coap://10.1.1.1:6500/meter/temp/obs> a ssncom:CommunicationEndpoint ;
35   ssncom:protocol "COAP" .
36 <coap://10.1.1.1:6500:6500/meter/heat/obs> a ssncom:CommunicationEndpoint ;
37   ssncom:protocol "COAP" .

```

Рисунок 3.7 – Сообщение о новом зарегистрированном ЭПУОВР

может в рамках открытой WAMP-сессии подписаться на топик, соответствующий данному идентификатору. Алгоритм получения потоковых данных представлен на рисунке 3.8.

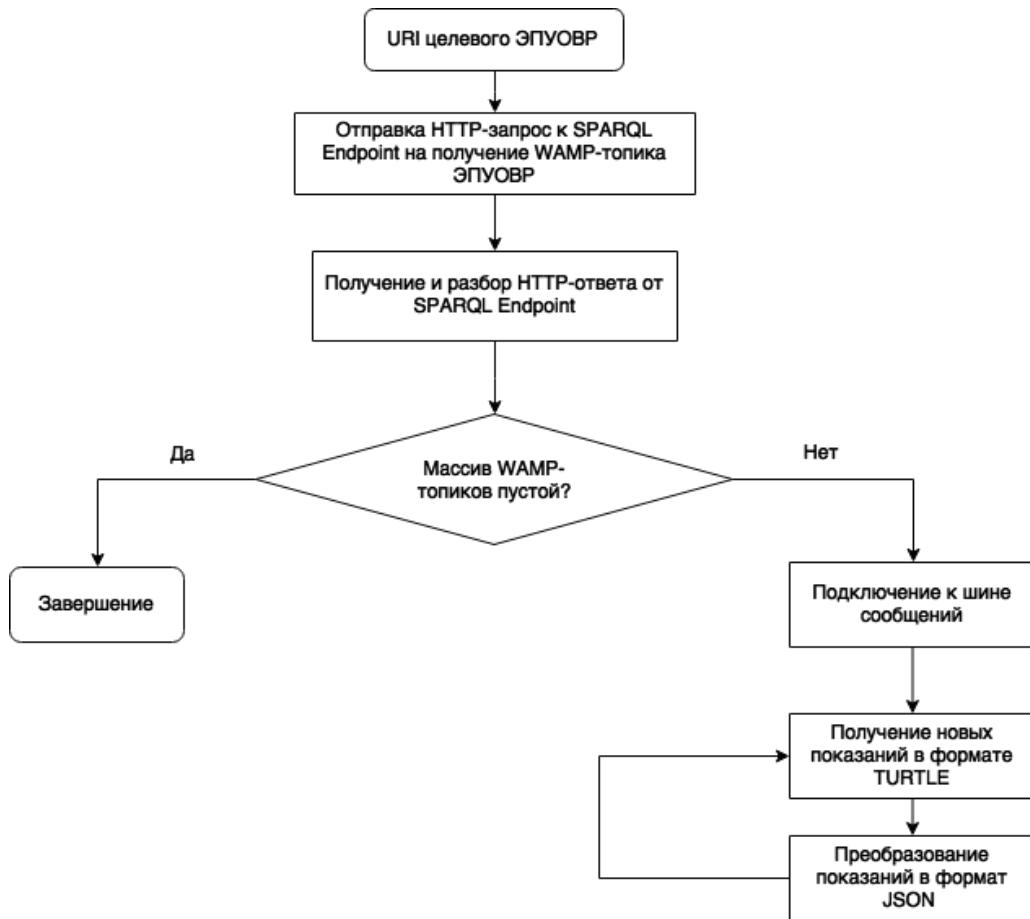


Рисунок 3.8 — Алгоритм получения текущих (потоковых) показаний ЭПУОВР

Формат сообщений в формате Turtle, транслирующихся по таким топикам, приведен на рисунке 3.9.

Форматы RDF и Turtle не являются родными для клиентского языка JavaScript, как, например, формат JSON. Существует ряд библиотек для разбора и работы с данными форматами, написанных на языке JavaScript:

- а) *rdfstore.js*
- б) *rdfquery*
- в) *N3.js*
- г) *SPARQL.js*
- д) *jsonld.js*
- е) *green-turtle*
- ж) *rdflib.js*

```

1 @prefix hmtr: <http://purl.org/NET/ssnext/heatmeters#> .
2 @prefix meter: <http://purl.org/NET/ssnext/meters/core#> .
3 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
4 @prefix xsd: <http://www.example.org/> .
5 @prefix : <coap://10.0.0.7:60000/meter/heat#> .

6
7 :1434916797482 a hmtr:HeatObservation ;
8   ssn:observationResultTime "2015-02-05T13:15:30Z"^^xsd:dateTime ;
9   ssn:observedBy <coap://10.0.0.7:60000/meter> ;
10  ssn:observationResult :1434916797482-result .
11
12 :1434916797482-result a hmtr:HeatSensorOutput ;
13   ssn:isProducedBy <coap://10.0.0.7:60000/meter> ;
14   ssn:hasValue :1434916797482-resultvalue .
15
16 :1434916797482-resultvalue a hmtr:HeatValue ;
17   meter:hasQuantityValue "0.636"^^xsd:float .

```

Рисунок 3.9 — Пример сообщения о новом зарегистрированном показании сенсора тепла

В рамках данной работы была использована библиотека N3.js как наиболее популярная, кроссплатформенная, поддерживающая наибольшее количество доступных форматов и поддерживаемая сообществом.

Применяя данную библиотеку и некоторые надстройки над ней, мы имеем возможность преобразовать данные формата Turtle в удобные для последующей обработки и визуализации. N3.js позиционирует себя как потоковый и асинхронный парсер; пример разбора показаний в формате Turtle приведен на рисунке 3.10.

```

1 rdfUtils.parseTTL(message).then(function(triples) {
2   var resource = rdfUtils.parseTriples(triples);
3   var observationResult = parseFloat(
4     resource.get(
5       "http://purl.org/NET/ssnext/meters/core#hasQuantityValue"
6     )
7   );
8 });

```

Рисунок 3.10 — Получение значения показания тепла из сообщения в формате Turtle

Преобразовывая с помощью библиотеки N3.js сообщения о регистрации новых показаний, клиентское приложение может в дальнейшем использовать их непосредственно для визуализации.

Таким образом, используя библиотеки и утилиты для подключения ко внешней шине сообщений, а также для разбора RDF-данных в JavaScript-приложениях, мы имеем возможность обрабатывать потоковые данные показаний системы ЭПУОВР.

3.5 Алгоритм визуализации потоковых и исторических данных

Основной информационной ценностью сенсорных сетей являются данные, предоставляемые ими, и для конечных пользователей программно-аппаратной платформы агрегации, нормализации и анализа таких данных важной целью является их наглядная визуальная презентация. Существуют различные способы визуализации статических, потоковых и исторических данных в зависимости от рассматриваемого аспекта.

Выделяют четыре основных аспекта данных и, соответственно, их атомарных типов, исходя из которых можно применить ту или иную модель визуализации:

- а) Количествоенный, или N - мерный - отображает набор характеристик некоторой сущности.
- б) Темпоральный - позволяет представить количественный аспект в его временной динамике.
- в) Пространственный - отвечает за расположение (в том числе географическое) сущностей.
- г) Коммуникативный - определяет взаимосвязь между сущностями.

Используя по отдельности или комбинируя, можно добиться приемлемого уровня информативности визуализации данных в том или ином контексте и применительно к той или иной задаче.

При работе с показаниями счетчиков ЭПУОВР в части визуализации возможно рассмотрение различных аспектов информации как о самих устрой-

ствах, их расположении, типе и состоянии, так и о самих показаниях - архивных и реального времени.

В частности, устройства, являющиеся частью распределенной системы, такой, как система отопления городских жилых домов, как правило имеют четко означенную географическую локализацию. Более того, информация о местоположении таких устройств может играть важную роль при работе с ними для получения пространственного представления о тех или иных событиях, идентифицируемых в ходе работы с такими устройствами. Визуализация системы таких устройств в виде карты с нанесенными на нее динамическими объектами, меняющими свое состояние в реальном времени, является достаточно емкой и информативной для данного класса задач.

В то же время, визуальное представление архивных данных за какой-либо период выходит за рамки описанной визуализации и решает другой класс задач, такой, как историчностное инспектирование тех или иных устройств на предмет выявления тех или иных динамик или тенденций - например, на основе его показаний. Данные задачи могут быть решены с применением визуализаций типа "временная линия событий" или двумерных линейных графиков.

Для реализации тех или иных визуализаций в прикладных пользовательских веб-приложениях на сегодня существует достаточно большое количество высокоуровневых инструментов или библиотек, использующих такие технологии, как API Canvas и WebGL для растровой графики или Scalable Vector Graphics для векторной графики. Рассмотрим наиболее популярные инструменты для создания графики и визуализаций в браузере:

- a) *Raphael.js* - предоставляет низкоуровневый кросбраузерный API для манипулирования SVG(в современных браузерах) и VML(в старых версиях браузера Internet Explorer). Дополняется функциями тех или иных визуализаций посредством плагинов.
- б) *D3.js* - аналогично *Raphael.js*, работает с SVG-элементами, однако содержит помимо низкоуровневого API инструменты для работы с форматами CSV и TSV, а также набор стандартных элементов - как примитивов(фигуры, линии), так и макетов диаграмм и графиков.

- в) *Leaflet.js* - популярная open-source библиотека для работы с картами. Имеет обширную базу плагинов.
- г) *Timeline.js* - предназначена в первую очередь для динамических интерактивных визуализаций континуальных процессов.
- д) *vis.js* - содержит набор атомарных макетов визуализаций, таких, как графы, сети, временные линии и графики.
- е) *Highcharts.js* - библиотека, ориентированная на визуализацию линейных данных при помощи графиков, гистограмм и диаграмм.

Рассмотрим подробней визуализацию архивных и текущих показаний датчиков устройств. После описанных выше шаго по получению архивных показаний от внешнего хранилища, а также подписки на новые показания посредством шины сообщений WAMP и парсинга Turtle-сообщений для получения искомых значений, на стороне клиентского веб-приложения имеется набор структурированных данных о показаниях конкретных счетчиков с их привязкой ко времени регистрации. Предложенный в данной работе метод визуализации таких данных заключается в построении графика зависимости значений показаний датчиков ЭПОУВР от времени регистрации. Для реализации данной задачи была использована библиотека *Highcharts.js*, предоставляющая наиболее простой и удобный API для работы. Пример конфигурации графика представлен на рисунке 3.11.

В результате, подав на вход функции инициализации графика с такой конфигурацией данные, нормализованные после получения от внешнего хранилища архивных показаний, формируется динамическая кривая корреляции показаний и времени их регистрации, пример которых приведен на рисунке 3.12. Также, в процессе работы с конкретным ЭПУОВР имеется возможность установить другой временной интервал для обозрения показаний.

Данные графики являются, во-первых, интерактивными, так как позволяют выбрать подинтервал на мини-карте графика внизу, и, во-вторых, динамическими, так как при получении и обработки новых сообщений новые показания автоматически вырисовываются на графике, что позволяет мониторить их в реальном времени.

Как уже было сказано выше, иногда бывает необходимо обозреть информационную картину не детализируя отдельные сущности, а представив

```

1  {
2      useHighStocks: true ,
3      xAxis: { type: "datetime" },
4      options: {
5          chart: { type: "spline" },
6          navigator: { enabled: true },
7          rangeSelector: { enabled: false },
8          useUTC: false ,
9          timezoneOffset: 3 * 60
10     },
11     series: [
12         {
13             pointStart: ( new Date() ).getTime() ,
14             name: "Temperature , C",
15             data: data
16         }
17     ],
18     title: { text: "Temperature , C" }
19 }
```

Рисунок 3.11 — Декларативное описание графика Highcharts.js для визуализации архивных показаний

более общий обзор. Используя географические методы визуализации распределенных в масштабах района или города совокупности ЭПУОВР, возможно визуально отобразить статус генерируемых показаний или же события, идентифицируемые на их основе - например, с помощью цветовой окраски мест, являющихся зонами ответственности конкретных устройств или с применением техники "heatmap" - тепловой карты, аккумулирующей события в виде цветовых пятен.

С развитием веб-картографии, обширной "векторизации" геоданных и появлением таких сервисов по работе с ними, как OpenStreetMap¹, появилось множество инструментов, и особенно - веб-инструментов для работы с картографическими данными. Одним из наиболее популярных на сегодня является библиотека Leaflet.js благодаря ее расширяемости и легковесному API. В данной работе библиотека Leaflet.js использована для географической визуализации распределенной совокупности ЭПУОВР.

¹<https://www.openstreetmap.org/>

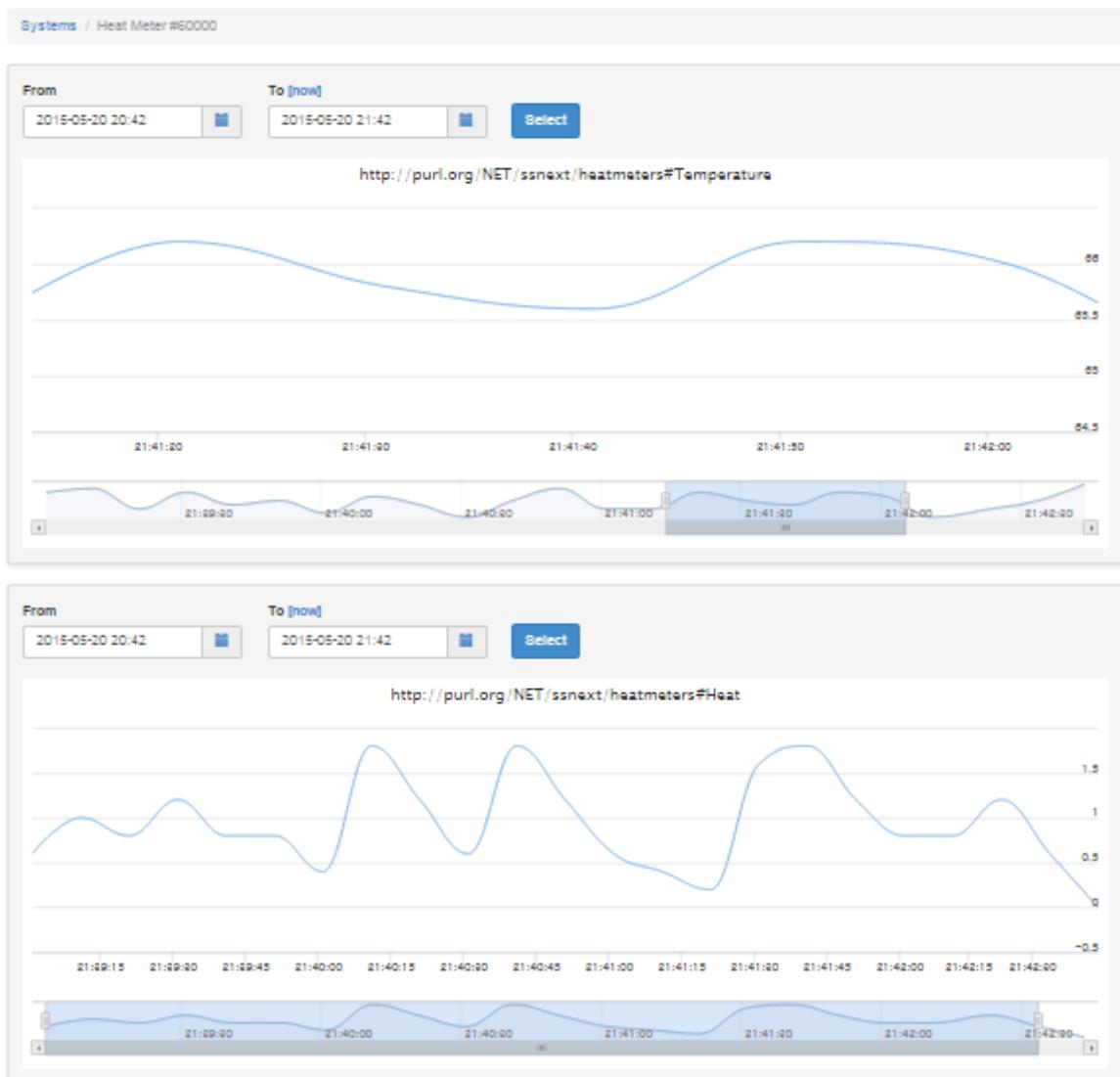


Рисунок 3.12 — Графики архивных показаний датчиков тепла и температуры ЭПУОВР

После получения списка зарегистрированных устройств пользователь может выбрать режим карты, на которой отобразятся все устройства, имеющие информацию о географическом местоположении. После этого ко всем таким устройствам осуществляются подписки в рамках WAMP-шины сообщений для получения их показаний, которые, в зависимости от интенсивности и близости друг от друга, отображаются в виде цветовых пятен тепловой карты. Пример данной визуализации представлен на рисунке 3.13

Также, предполагая, что в качестве зоны ответственности таких устройств выступают отдельные дома, возможна их цветовая окраска. Зная координаты устройства, можно определить, в каком доме он находится. Так-

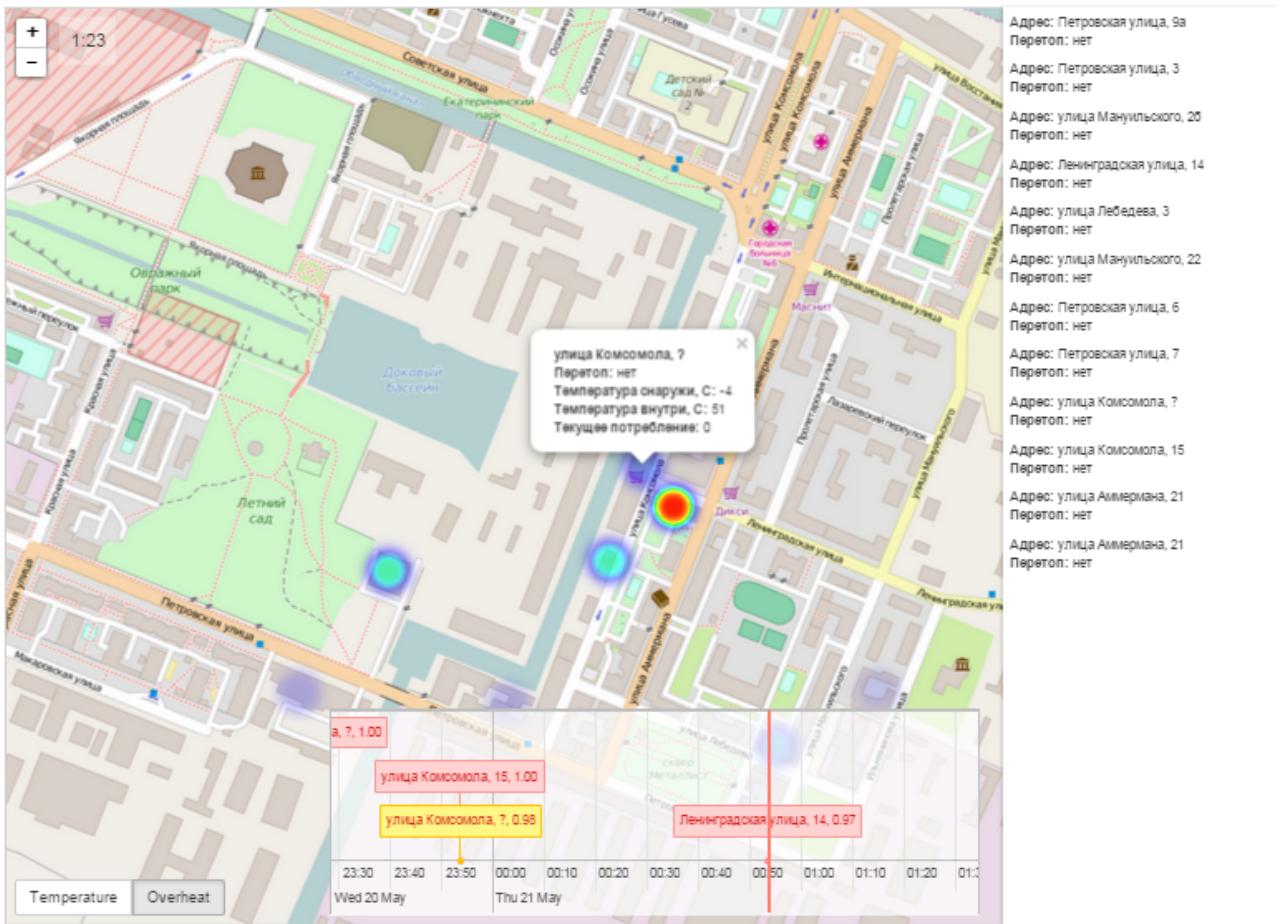


Рисунок 3.13 — Пример визуализации показаний посредством технологии heatmap

же, с помощью сторонних сервисов - например, Open Street Map Buildings¹ - есть возможность получить информацию о параметрах самих домов - точных координатах, этажности и так далее. После чего, каждый дом в рамках карты можно представить в векторном отображении, используя плагин к библиотеке Leaflet.js под названием *OSMBuildings-Leaflet.js* и, соответственно, окрасить его соразмерно интенсивности показаний устройства, расположенного внутри данного дома. Пример данной визуализации продемонстрирован на рисунке 3.14.

Однако, такая парадигма визуализации не предполагает какой-либо исторической информации; для этой цели был разработан пользовательский элемент управления в виде временной линии, на которой отображаются в реальном времени с возможностью просмотра истории те или иные события, возникающие на основе показаний.

¹<http://wiki.openstreetmap.org/wiki/Buildings>



Рисунок 3.14 — Пример цветовой дифференциации домов в качестве зон ответственности счетчиков

Алгоритм визуализации потоковых и исторических данных представлен на рисунке 3.15

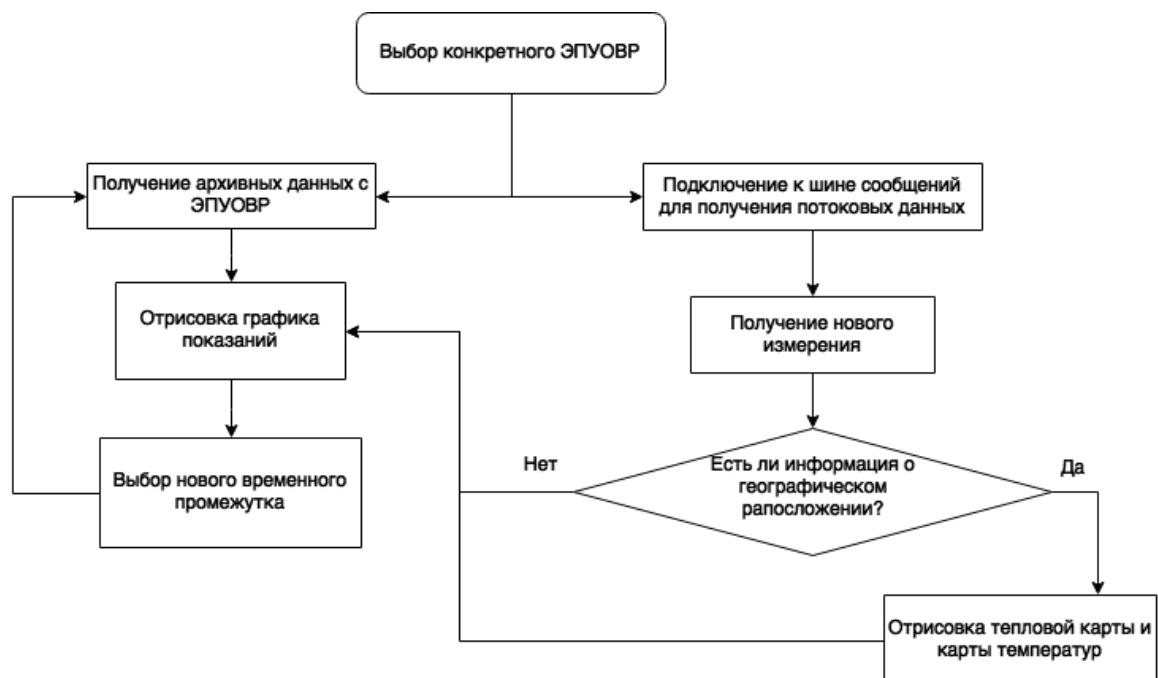


Рисунок 3.15 — Алгоритм визуализации потоковых и исторических данных с зарегистрированных ЭПУОВР

Таким образом, используя современные библиотеки для работы с картами и построения графиков, такие, как Leaflet.js и Highcharts.js, мы можем визуализовать различные аспекты статических, архивных и потоковых данных, генерируемых ЭПУОВР, алгоритм получения которых описан в предыдущих подразделах.

3.6 Результаты

В данном разделе отчета описаны методы и алгоритмы интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data).

В соответствии с техническим заданием, разработанные в рамках данной работы методы и алгоритмы интерактивной визуализации данных распределенной сети электронных потребительских устройств отвечают следующим требованиям технического задания: (а) сбор данных для построения графических форм визуализации конфигурации сети пользовательских устройств не более, чем за 15 секунд; (б) время отклика системы для операций навигации по интерактивным графическим формам – не более чем за 5 секунд; (в) время построения визуальных форм для графического представления данных электронных потребительских устройств – не более 3 секунд. Все перечисленные требования ТЗ обеспечиваются техническими характеристиками разработанных методов и алгоритмов, описанных в данном разделе, и исследованиями описанными в Приложении А данного отчета о ПНИ.

В результате данной работы были разработаны и исследованы необходимые методы и алгоритмы, соответствующие требования Технического задания.

ЗАКЛЮЧЕНИЕ

В данном отчете представлены результаты работ по теоретическим исследованиям на 2 этапе ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)».

В данном отчете представлены результаты следующих работ, в соответствии с Планом-графиком работ (далее, ПГ):

- а) разработка математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба.
- б) разработка математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things).
- в) разработка математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологий связанных данных (Linked Data).
- г) участие в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ (конференции, семинары, симпозиумы, выставки, и т.п., в том числе, международные).

Далее перечисляются основные результаты, полученные за отчетный период и описанные в данном отчете о ПНИ.

В рамках работы по п.2.1 ПГ разработаны математические методы и алгоритмы агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением при-

емов онтологического инжиниринга и технологий семантического веба в составе:

- а) метода агрегации данных с электронных потребительских устройств с ограниченными вычислительными ресурсами (ЭПУОВР), использующих прикладной протокол передачи данных СоАР (Constrained Application Protocol).
- б) алгоритм асинхронной передачи данных с ЭПУОВР, использующих протокол СоАР.
- в) алгоритма кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF (Resource Description Framework) и протокол СоАР.
- г) метода агрегации данных с ЭПУОВР, использующих проприetary прикладной протокол передачи данных на основе протокола UDP (User Datagram Protocol).
- д) метода и алгоритма аннотирования данных с ЭПУОВР, не использующих модель данных RDF.
- е) метод и алгоритмы хранения больших массивов данных распределенной сети электронных потребительских устройств.

Разработанные в рамках данной работы методы и алгоритмы соответствуют требованиям технического задания, в частности требованиям: по поддержки протокола UDP; по поддержки синхронного и асинхронного режимов передачи данных; по глубине хранения не менее 1 года для не менее чем 1000 электронных потребительских устройств.

В рамках работы по п.2.2 ПГ разработаны математические методы и алгоритмы нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things) в составе:

- а) метода нормализации гетерогенных данных распределенной сети электронных потребительских устройств на основе верхнеуровневых онтологий.

- б) алгоритма установления соответствия между предметно-ориентированными онтологиями и верхнеуровневым онтологией, используемыми для нормализации гетерогенных данных.
- в) метода и алгоритма оценки производительности и корректности систем анализа связанных потоковых данных.
- г) метода и алгоритма анализа потоковых данных распределенной сети электронных потребительских устройств.
- д) алгоритма интеграции потоковых и статических данных для анализа данных распределенной сети электронных потребительских устройств.
- е) метода обнаружения и генерации событий предметной области на основе анализа потоковых и связанных данных.

Разработанные в рамках данной работы методы и алгоритмы полностью соответствуют требованиям технического задания, в частности требования по приведению данных распределенной сети электронных потребительских устройств к нормальному виду и препроцессингу структурированных, полученных структурированных и неструктурированных данных.

В рамках работы по п.2.3 ПГ разработаны математические методы и алгоритмы интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data) в составе:

- а) метода и алгоритма доступа к статическим данным на основе семантических технологий и технологии связанных данных.
- б) алгоритма доступа к историческим данным распределенной сети электронных потребительских устройств.
- в) метода и алгоритма доступа к потоковым данным распределенной сети электронных потребительских устройств.
- г) алгоритма визуализации потоковых и исторических данных.

Разработанные методы и алгоритмы соответствуют требованиям перечисленным в техническом задании, в частности требованиям: по времени построения графических форм визуализации конфигурации сети устройств; по времени отклика системы для операции навигации по интерактивным графическим

формам; по времени построения визуальных форм для графического представления данных устройств.

На отчетном этапе были достигнуты все необходимые результаты в полном объеме, предусмотренные Планом-графиком работ, которые удовлетворяют требования, сформулированным в техническом задании. Команда проекта проявила способность оперативно справляться с исследовательскими и техническими задачами и продемонстрировала готовность к выполнению серьезных прикладных научно-технических исследований.

На специализированном сайте в сети Интернет, доступном по адресу <http://semiot.ru>, размещена информацию о ходе выполнения и результатах промежуточных работ по данному ПНИ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Отчет о ПНИ по теме: «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полу-структурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)». Этап 1 «Выбор направления исследований»: Отчет о ПНИ (промежуточный) / Университет ИТМО. — М.: ВИНИТИ, 2014. — 303 с.

2 C. Bormann M. Ersue, A. Keranen. Terminology for Constrained-Node Networks / A. Keranen C. Bormann, M. Ersue; Internet Engineering Task Force (IETF): 2014. — May.

3 Binary RDF representation for publication and exchange (HDT) / Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez et al. // *Web Semantics: Science, Services and Agents on the World Wide Web*. — 2013. — Vol. 19, no. 0. — Pp. 22 – 41.

4 Henning, Hasemann. RDF provisioning for the Internet of Things / Hasemann Henning, Kröller Alexander, Pagel Max // 3rd IEEE International Conference on the Internet of Things, IOT 2012, Wuxi, Jiangsu Province, China, October 24-26, 2012. — 2012. — Pp. 143–150.

5 Fernández, Javier D. Efficient RDF Interchange (ERI) Format for RDF Data Streams / Javier D. Fernández, Alejandro Llaves, Oscar Corcho // The Semantic Web – ISWC 2014. — Springer International Publishing, 2014. — Vol. 8797 of *Lecture Notes in Computer Science*. — Pp. 244–259.

6 RDSZ: An Approach for Lossless RDF Stream Compression / Norberto Fernández, Jesús Arias, Luis Sánchez at al. // The Semantic Web: Trends and Challenges. — Springer International Publishing, 2014. — Vol. 8465 of *Lecture Notes in Computer Science*. — Pp. 52–67.

7 Neumann, Thomas. The RDF-3X engine for scalable management of RDF data / Thomas Neumann, Gerhard Weikum // *The VLDB Journal*. — 2009. — sep. — Vol. 19, no. 1. — Pp. 91–113.

8 *Käbisch, Sebastian*. Standardized and Efficient RDF Encoding for Constrained Embedded Networks / Sebastian Käbisch, Daniel Peintner, Darko Anicic // The Semantic Web. Latest Advances and New Domains. — Springer Science + Business Media, 2015. — Pp. 437–452.

9 *Bournez, Carine*. Efficient XML Interchange Evaluation / Carine Bournez; World Wide Web Consortium (W3C): XML Binary Characterization (XBC) Working Group, 2009. — April.

10 The SSN ontology of the W3C semantic sensor network incubator group / Michael Compton, Payam Barnaghi, Luis Bermudez et al. // *Web Semantics: Science, Services and Agents on the World Wide Web*. — 2012. — Vol. 17. — Pp. 25–32.

11 Road vehicles – Vehicle-to-Grid Communication Interface – Part 2: Network and application protocol requirements / The International Organization for Standardization: International Organization for Standardization, 2014. — ISO 15118-2:2014.

12 Efficient XML Interchange (EXI) Format 1.0 (Second Edition) / Schneider John, Kamiya Takuki, Peintner Daniel, Kyusakov Rumen; World Wide Web Consortium (W3C): EXI Working Group, 2014. — February.

13 Web Services for the Internet of Things through CoAP and EXI / Angelo P Castellani, Mattia Gheda, Nicola Bui et al. // Communications Workshops (ICC), 2011 IEEE International Conference on / IEEE. — 2011. — Pp. 1–6.

14 *Uschold, Michael*. Ontologies and Semantics for Seamless Connectivity / Michael Uschold, Michael Gruninger // *SIGMOD Rec.* — 2004. — December. — Vol. 33, no. 4. — Pp. 58–64.

15 *Kalfoglou, Yannis*. Ontology mapping: the state of the art / Yannis Kalfoglou, Marco Schorlemmer // *The Knowledge Engineering Review*. — 2003. — 1. — Vol. 18. — Pp. 1–31.

16 *Welty, Christopher*. Ontology Research / Christopher Welty // *AI Magazine*. — 2003. — Vol. 24. — Pp. 11–12.

17 *Noy, Natalya F.* Semantic Integration: A Survey of Ontology-based Approaches / Natalya F. Noy // *SIGMOD Rec.* — 2004. — December. — Vol. 33, no. 4. — Pp. 65–70.

18 Semantic Sensor Network XG Final Report / Barnaghi Payam, Compton Michael, Corcho Oscar et al.; World Wide Web Consortium (W3C): Semantic Sensor Network Incubator Group, 2011.

19 STREAM: The Stanford Stream Data Manager / Arvind Arasu, Brian Babcock, Shivnath Babu et al. // *IEEE Data Eng. Bull.* — 2003. — Vol. 26, no. 1. — Pp. 19–26.

20 *Cugola, Gianpaolo*. Processing flows of information: From data stream to complex event processing / Gianpaolo Cugola, Alessandro Margara // *ACM Comput. Surv.* — 2012. — Vol. 44, no. 3. — P. 15.

21 SECRET: A Model for Analysis of the Execution Semantics of Stream Processing Systems / Irina Botan, Roozbeh Derakhshan, Nihal Dindar et al. // *PVLDB*. — 2010. — Vol. 3, no. 1. — Pp. 232–243.

22 Querying RDF streams with C-SPARQL / Davide Francesco Barbieri, Daniele Braga, Stefano Ceri et al. // *SIGMOD Record*. — 2010. — Vol. 39, no. 1. — Pp. 20–26.

23 A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data / Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, Manfred Hauswirth // Proceedings of the 10th International Conference on The Semantic Web - Volume Part I. — ISWC'11. — Berlin, Heidelberg: Springer-Verlag, 2011. — Pp. 370–388.

24 *Calbimonte, Jean-Paul*. Enabling Ontology-based Access to Streaming Data Sources / Jean-Paul Calbimonte, Oscar Corcho, Alasdair J. G. Gray // Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I. — ISWC'10. — Berlin, Heidelberg: Springer-Verlag, 2010. — Pp. 96–111.

25 Enabling Query Technologies for the Semantic Sensor Web / Jean-Paul Calbimonte, Hoyoung Jeung, Oscar Corcho, Karl Aberer // *International Journal On Semantic Web and Information Systems (IJSWIS)*. — 2012. — Vol. 8, no. 1. — Pp. 43–63.

26 ETALIS: Rule-Based Reasoning in Event Processing / Darko Anicic, Paul Fodor, Sebastian Rudolph et al. // Reasoning in Event-Based Distributed Systems / Ed. by Sven Helmer, Alexandra Poulovassilis, Fatos Xhafa. — Springer Berlin / Heidelberg, 2011. — Vol. 347 of *Studies in Computational*

Intelligence. — Pp. 99–124.

27 EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning / Darko Anicic, Paul Fodor, Sebastian Rudolph, Nenad Stojanovic // Proceedings of the 20th International Conference on World Wide Web. — WWW '11. — New York, NY, USA: ACM, 2011. — Pp. 635–644.

Приложение А

Отчет о результатах исследования методов и алгоритмов

Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев

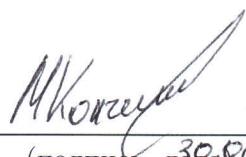
«30 июня 2015 г.

ОТЧЕТ О РЕЗУЛЬТАТАХ ИССЛЕДОВАНИЯ МЕТОДОВ И АЛГОРИТМОВ

ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы»

Соглашение о предоставлении субсидии от 24.11.2014 г. №14.575.21.0101

Ответственный исполнитель работ


М.А. Колчин
(подпись, дата)

Санкт-Петербург 2015

A.1 Аннотация

В данном отчете представлены результаты исследования методов и алгоритмов разработанных в рамках этапа 2 данного ПНИ. Целью исследования является определение технических характеристик разработанных методов и алгоритмов и оценка их соответствия требованиям технического задания.

Исследования представленные в данном отчете были выполнены на компьютере со следующими характеристиками:

- а) объем оперативной памяти: 8 Гб,
- б) объем дискового пространства: 180 Гб,
- в) тип жесткого диска: SSD (Solid-state drive),
- г) модель процессора: Intel Core i7,
- д) операционная система: Fedora 21 x86_64 (Linux 4.1.3).

A.2 Исследование математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба

A.2.1 Исследование метода агрегации данных с ЭПУОВР, использующих протокол СоАР, и алгоритма кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF

Метод агрегации данных, описанный в подразделе 1.2, и алгоритм кодирования и декодирования данных с ЭПУОВР, использующих модель данных RDF и протокол СоАР, описанный в подразделе 1.4 были реализованы в симуляторах электронных потребительских устройств, которые являются частью программно-аппаратной системы моделирования сети электронных потребительских устройств (Internet Of Things) [1].

Данный метод и алгоритм были реализованы в двух симуляторах на языке программирования Java: симулятор электрических счетчиков и сети, которая состоит из них, а также симулятор общедомовых счетчиков тепла. Исходный код симуляторов является открытым и опубликован под открытой лицензией по адресам <http://github.com/semitproject/simulator-electric-meters> и <http://github.com/semitproject/simulator-heat-meters> соответственно.

Для оценки данного метода и алгоритма были выполнены несколько экспериментов при количестве счетчиков равному 100, 500 и 1000: (а) оценка времени ответа на GET-запроса; (б) максимальный объем потребляемой оперативной (динамической) памяти.

Оценка времени ответа на GET-запрос проводилась с помощью утилиты *coap-client*, которая поставляется в комплекте с библиотекой *libcoap*¹, и команды *time*. Пример команды: *time coap-client -p 60100 -m get <URI-ресурса>*.

¹<https://github.com/obgm/libcoap>

На рисунке А.1 представлены результаты оценки времени ответа на GET-запрос с 100, 500 и 1000 счетчиками. Среднее значение вычислено на основе пяти измерений. Результаты в порядке увеличения количества счетчиков: 104 мс, 120 мс, 116 мс.

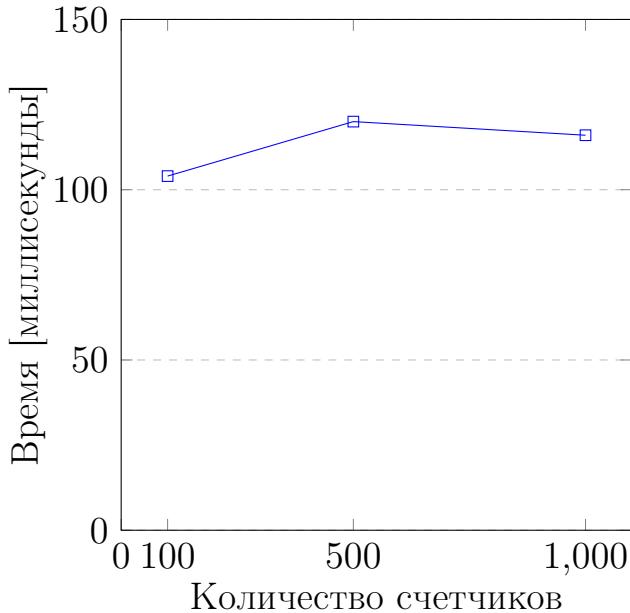


Рисунок А.1 — Среднее время ответа на GET-запрос

Из результатов данного эксперимента можно сделать вывод, что количество одновременно симулируемых счетчика не влияет на время отклика на GET-запросы к одному из них.

На рисунке А.2 представлены результаты оценки максимального объема потребляемой (динамической) оперативной памяти с 100, 500 и 1000 счетчиками. Под динамической памятью понимается объем памяти отводимый на хранение экземпляров классов, так называемая Heap Memory. Результаты в порядке увеличения количества счетчиков: 25.3 Мб, 54.1 Мб, 76.7 Мб.

Из результатов данного эксперимента можно сделать вывод, что оперативная память используется достаточно эффективно. Это позволяет сделать вывод, что симуляторы можно запускать на компьютере с меньшим количеством ОЗУ, чем на компьютере, на котором производились измерения.

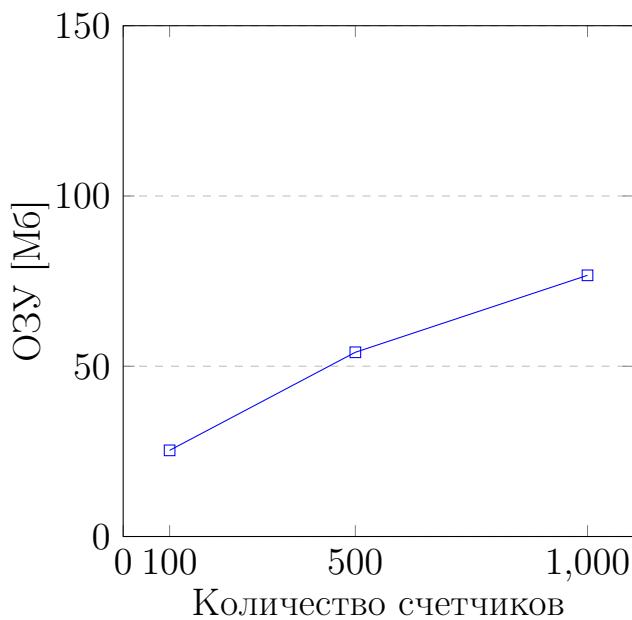


Рисунок А.2 — Максимальный объем потребляемой ОЗУ

A.2.2 Исследование алгоритма асинхронной передачи данных с ЭПУОВР, использующих протокол СоАР

В данном подразделе представлено исследование алгоритма асинхронной передачи данных с ЭПУОВР, использующих протокол СоАР. В данном исследовании представлены следующие характеристики алгоритма: размер кода, который записывается на ЭПУОВР; требуемый размер оперативной памяти; максимально количество одновременных подписчиков; среднее время ответа на GET-запрос.

Данный алгоритм был реализован на языке программирования С для контроллера Arduino Mega 2560 Rev3¹ и Wi-Fi модуля ESP8266².

Размер кода прошивки составляет 292.46 Кб, который кроме самого алгоритма включает все необходимые сторонние библиотеки. Так же данная прошивка использует от 64.12 Кб до 69.05 КБ оперативной памяти в качестве динамической памяти в зависимости от количества одновременных подписок и 15.87 Кб для локальных переменных. На рисунке А.3 представлен график зависимости размера динамической памяти от количества одновременных подписчиков.

¹<https://store.arduino.cc/product/A000067>

²<http://www.espruino.com/ESP8266>

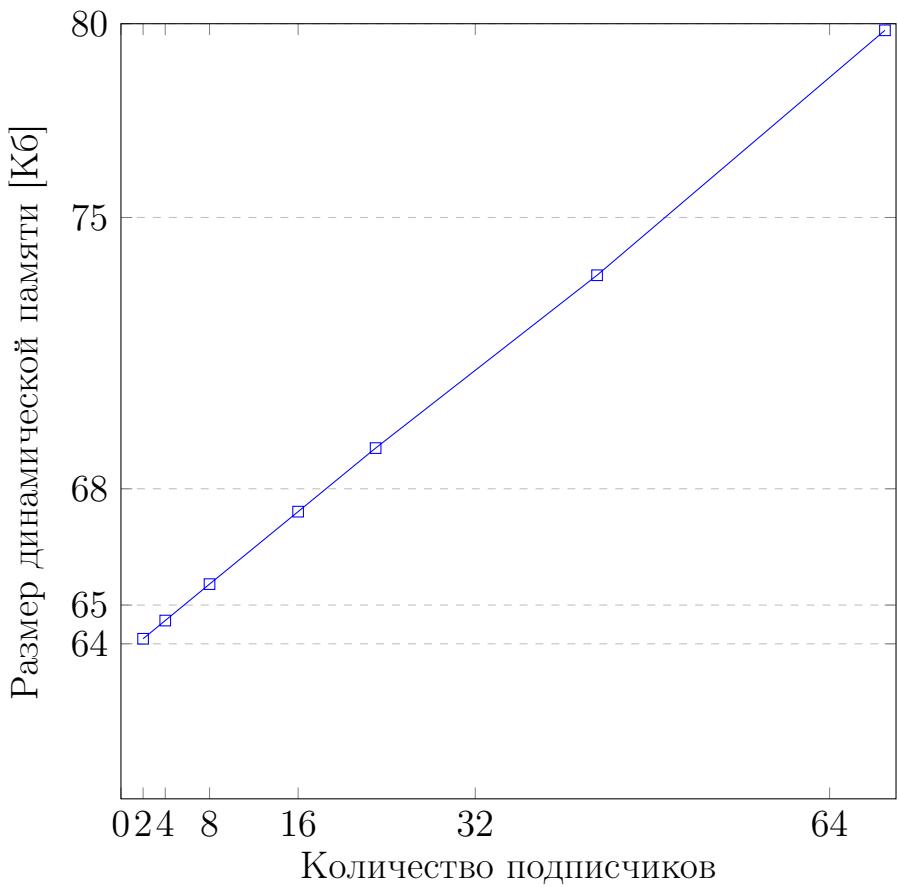


Рисунок А.3 — Размер динамической памяти в зависимости от количества подписчиков

И последней характеристикой данного алгоритма является среднее время ответа на GET-запрос. После 5-ти измерений, из которых наименьшим временем ответа стало 4 мс, а наибольшим 117 мс, получилось что средним временем ответа на GET-запрос составляет 7 мс.

A.2.3 Исследование метода агрегации данных с ЭПУОВР, использующих проприетарный прикладной протокол передачи данных на основе UDP, и метода и алгоритма аннотирования данных с ЭПУОВР, не использующих модель данных RDF

Метод агрегации данных с ЭПУОВР, использующий проприетарный протокол передачи данных на основе UDP, описанный в подразделе 1.5, и метод и алгоритм аннотирования данных с ЭПУОВР, не использующих модель данных RDF, описанных в подразделе 1.6, были реализованы в одном из модулей ПО ЭО ПАП.

В данном подразделе описано исследование методов и алгоритмов агрегации данных с ЭПУОВР работающих по протоколу UDP и не использующих модель данных RDF, которое заключается в оценке максимального объема динамической (кучи) памяти потребляемой модулем ПО ЭО ПАП реализующим их.

Так же как в предыдущих подразделах для исследования методов и алгоритмов применяется симулятор общедомовых счетчиков тепла, являющийся частью ПАС. Для оценки максимального объема динамической (кучи) память данный модуль про работал со 100, 500 и 1000 одновременно подключенными счетчиками. Результаты представлены на рисунке А.4 и равны следующим оценкам перечисленным в порядке возрастания количества счетчиков: 85.1 Мб, 90.3 Мб и 92.9 Мб.

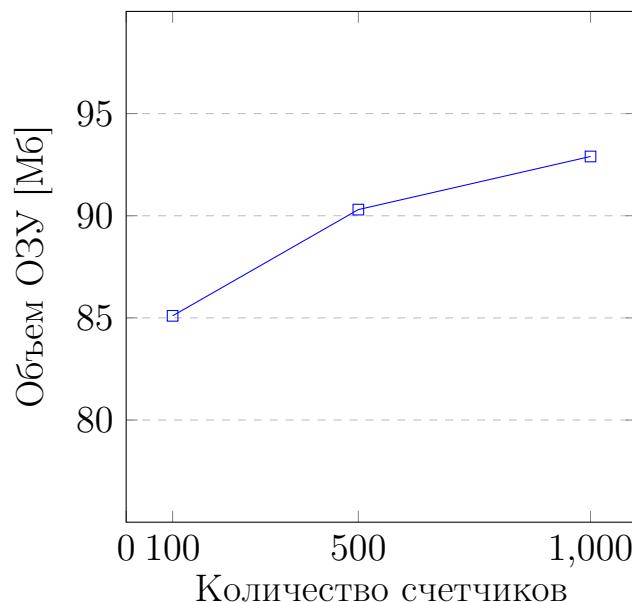


Рисунок А.4 — Максимальный объем динамической (кучи) памяти

A.2.4 Исследование метода и алгоритмов хранения больших массивов данных распределенной сети электронных потребительских устройств

В данном подразделе описаны исследования метода и алгоритмов хранения данных электронных потребительских устройств, которые заключаются в оценке объема занимаемого дискового пространства данной БД и оценке

максимального объема динамической (кучи) памятиу потребляемое реализацией алгоритмов.

Для оценки объема занимаемого дискового пространства БД Fuseki в нее были загружены метаданные 100, 500 и 1000 общедомовых счетчиков тепла. В рисунке А.5 представлен пример метаданных общедомового счетчика тепла в формате RDF/Turtle.

```
1 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
2 @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .  
3 @prefix hmtr: <http://purl.org/NET/ssnext/heatmeters#> .  
4 @prefix ssncom: <http://purl.org/NET/ssnext/communication#> .  
5  
6 <coap://localhost:90000/meter> a hmtr:HeatMeter ;  
7   rdfs:label "Heat Meter #90000" ;  
8   ssn:hasSubsystem <coap://localhost:90000/meter/temperature> ;  
9   ssn:hasSubsystem <coap://localhost:90000/meter/heat> .  
10  
11 <coap://localhost:90000/meter/temperature> a ssn:Sensor ;  
12   ssn:observes hmtr:Temperature ;  
13   ssncom:hasCommunicationEndpoint  
14     <coap://localhost:90000/meter/temperature/obs> .  
15  
16 <coap://localhost:90000/meter/heat> a ssn:Sensor ;  
17   ssn:observes hmtr:Heat ;  
18   ssncom:hasCommunicationEndpoint <coap://localhost:90000/meter/heat/obs> .  
19  
20 <coap://localhost:90000/meter/temperature/obs> a ssncom:CommunicationEndpoint ;  
21   ssncom:protocol "COAP" .  
22 <coap://localhost:90000/meter/heat/obs> a ssncom:CommunicationEndpoint ;  
23   ssncom:protocol "COAP" .
```

Рисунок А.5 — Пример метаданных общедомового счетчика тепла

Изначально в БД Fuseki загружены верхнеуровневые онтологии, в результате чего изначальный размер БД на диске составляет 292Кб. В последующих замерах размера БД на диске её изначальный размер будет учитываться.

На рисунке А.6 представлены результаты оценки объема занимаемого дискового пространства БД Fuseki с 100, 500 и 1000 счетчиками. Результаты в порядке возрастания количества счетчиков: 608Кб, 2.0Мб и 3.7Мб.

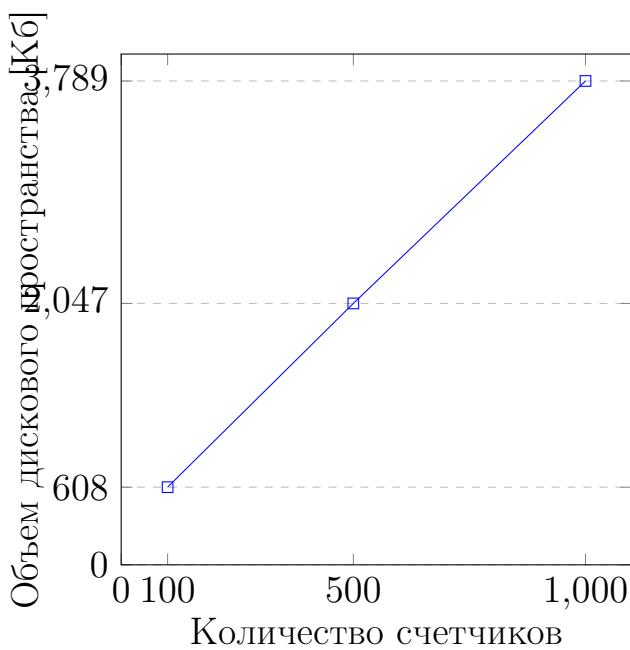


Рисунок А.6 — Объем занимаемого дискового пространства БД Fuseki

Из результатов этого эксперимента можно сделать вывод, что БД Fuseki занимает небольшое количество дискового пространства для хранения метаданных тысяч устройств.

Алгоритм хранения данных электронных потребительских устройств в БД Fuseki реализован в виде одного из модулей ПО ЭО ПАП. На рисунке А.7 представлены результаты эксперимента по оценке максимального объема динамической (кучи) памяти потребляемое данным алгоритмом. Далее эти результаты в порядке возрастания количества счетчиков: 73.84 Мб, 142.18Мб и 264.56 Мб.

Стоит отметить, что замеренные максимальные объемы динамической памяти в каждом из случаев являются пиковыми значениями, средний же объем динамической памяти значительно за все время работы значительно ниже. См. рисунок А.8.

Оценка объема занимаемого дискового пространства БД OpenTSDB будет вычислена теоретически на основе данных размера структур данных используемых для хранения данных в этой БД, представленных на сайте разработчика¹. Вычислим размер дискового пространства занимаемого одним показаниям общедомового счетчика тепла на основе размера соответствую-

¹<http://opentsdb.net/faq.html>

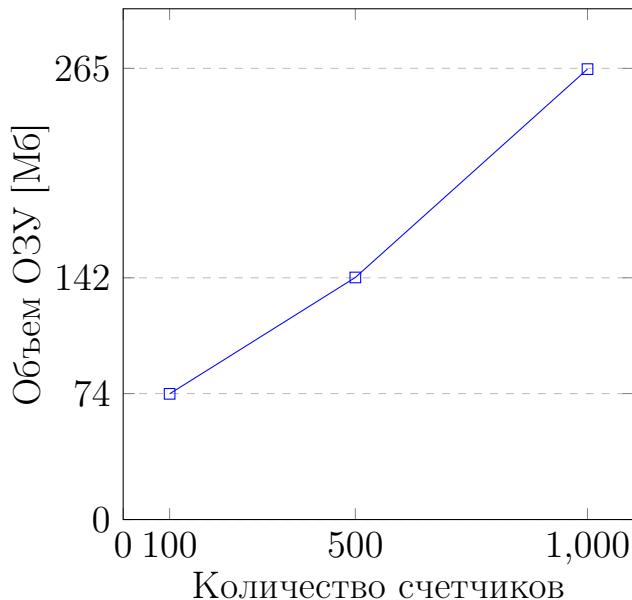


Рисунок А.7 — Макс. объем динамической (кучи) памяти

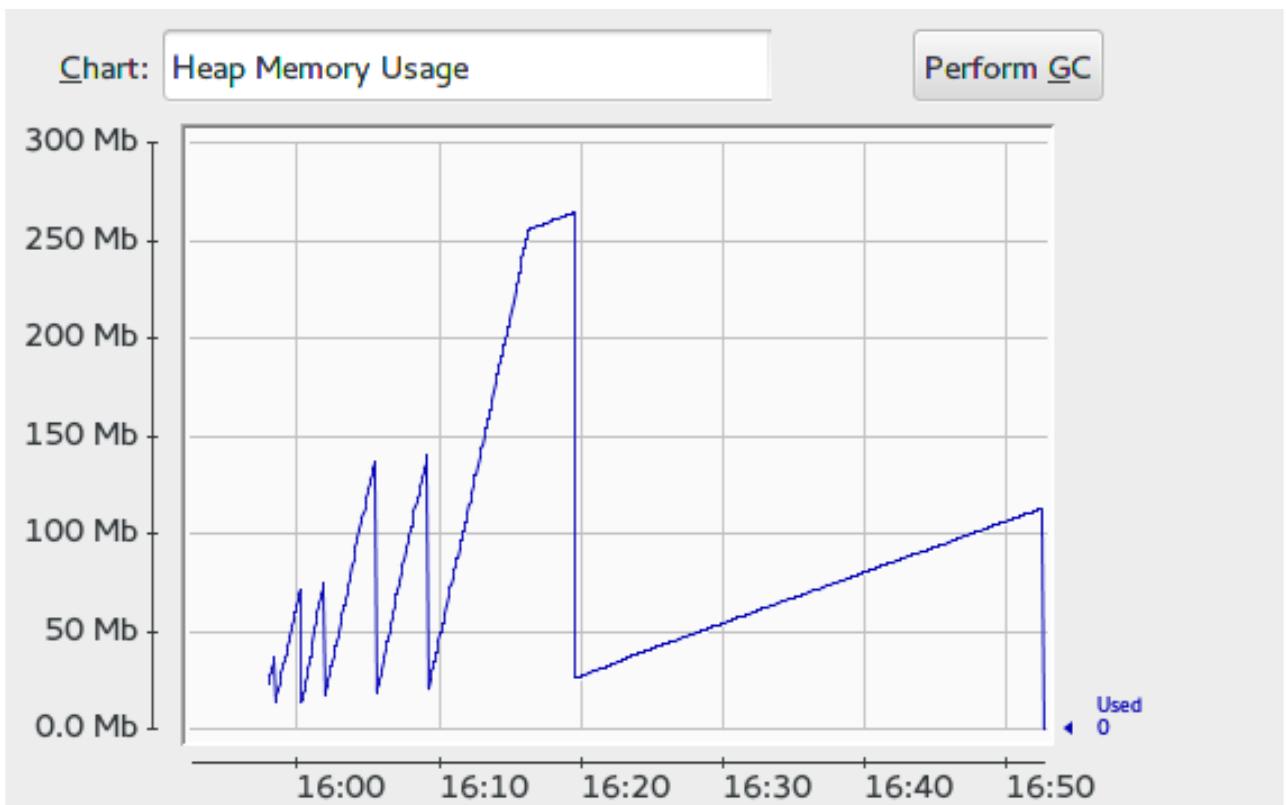


Рисунок А.8 — График объема динамической (кучи) памяти

щих структур данных, а именно: затраты HBase равны 16 байт, 3 байта на метрику, 4 байта на временную метку, 6 байт на один тег и 2 байта на затраты OpenTSDB. Так как на одно показание счетчика используется 2 тега, тогда размер требуемого дискового пространства составляет: $16 + 3 + 4 + 6*2 + 2 = 37$ байт.

На основе вычисленного выше размера одного показания и того что каждый счетчик снимает по два показания в 1 минуту, делаем следующие выводы по объему дискового пространства необходимые для 100, 500 и 1000 счетчиков: 3709.2 Мб, 18546.3Мб и 37092.6Мб соответственно.

Алгоритм хранения данных электронных потребительских устройств в БД OpenTSDB реализован в виде одного из модулей ПО ЭО ПАП. На рисунке А.9 представлены результаты эксперимента по оценке максимального объема динамической (кучи) памяти потребляемое данным алгоритмом. Далее эти результаты в порядке возрастания количества счетчиков: 73 Мб, 73.23 Мб и 80 Мб.

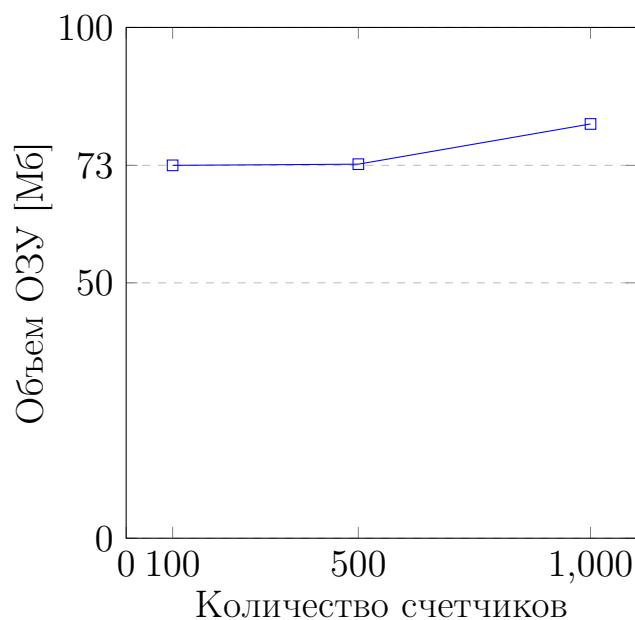


Рисунок А.9 — Макс. объем динамической (кучи) памяти

А.3 Исследование математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств (Internet of Things)

А.3.1 Исследование методов и алгоритмов нормализации и анализа больших массивов гетерогенных данных генерируемых в распределенной сети электронных потребительских устройств

В данном подразделе исследуются характеристики метода нормализации гетерогенных данных, описанного в подразделе 2.2, и алгоритма установления соответствия между предметно-ориентированными и верхнеуровневыми онтологиями. Для этого проведены два эксперимента: оценка работы логического вывода (reasoning) на основе SPARQL-запросов и оценка среднего времени ответа на SPARQL-запрос.

Для оценки работы логического вывода в БД Fuseki, в которая используется для хранения метаданных устройств загружаются метаданные общедомового счетчика тепла, которые используют предметно-ориентированные онтологии. Далее выполняется SPARQL-запрос, использующий классы и свойства одной из верхнеуровневых онтологий, например Semantic Sensor Network Ontology (SSN), если ответ на такой запрос получен и он корректен, то логический вывод работает.

На рисунке А.5 представлен пример метаданных общедомового счетчика тепла, который был загружен в БД Fuseki с помощью веб-интерфейса этой БД. Далее выполнен SPARQL-запрос представленный на рисунке А.10.

```
1 PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
2
3 SELECT ?x WHERE {
4     ?x a ssn:System .
5 }
```

Рисунок А.10 — Пример запроса активных устройств

В метаданных счетчика явно не указано, что он относиться к классу *ssn:System*, но в онтологии HTMR¹ указано, что класс *htmr:HeatMeter* является подклассом *ssn:System*. Соответственно результатом данного запроса должен быть URI данного счетчика. Без логического вывода такой запрос не вернет результатов.

Так как логический вывод реализован, результатом запроса на рисунке A.10 стал URI данного счетчика.

На рисунке A.11 представлены результаты оценки среднего времени ответа на SPARQL-запрос, в котором используются выводимые логическим выводом концепты, для 100, 500 и 1000 устройств. Среднее время ответа вычислено на основе 5 запросов. Результаты в порядке возрастания количества счетчиков: 42.6 мс, 101 мс и 187.4 мс.

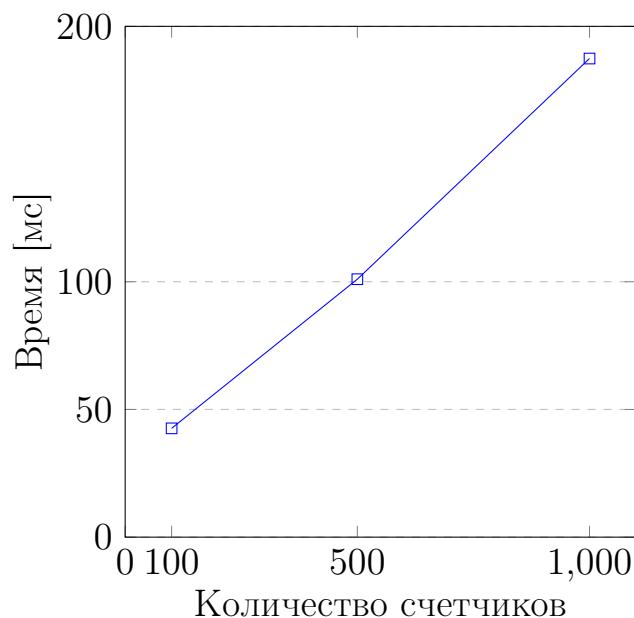


Рисунок A.11 — Среднее время ответа на SPARQL-запрос

¹<http://purl.org/NET/ssnext/heatmeters#>

A.3.2 Исследование метода и алгоритмов оценки производительности и корректности систем анализа связанных потоковых данных

В данном подразделе исследуется метод и алгоритмы оценки производительности и корректности систем анализа связанных потоковых данных, описанных в подразделе 2.4.

Исследование корректности работы разработанных метода и алгоритмов реализовано за счет повторения эксперимента CSRBench¹ с СУПД C-SPARQL и CQELS и сравнения результатов этого эксперимента и результатов полученных программным комплексом YABench.

Методология сравнения результатов следующая:

- а) исходные данные использованные CSRBench были конвертированы в формат N-Triples и расширены с помощью временных меток для имитации потока данных в течении 1000 мс. Тем самым поток данных выглядит так же как выходной поток генератора используемого в YABench.
- б) для каждого из семи запросов CSRBench были подготовлены конфигурации тестовых сценариев. Эти конфигурации определяют такие параметры как размер и шаг окна, параметры фильтра, которые используют для регистрации запроса в СУПД.
- в) для каждого СУПД и каждого из семи запросов были запущены тестовые сценарии, т.е. передачей потоковых данных в СУПД и регистрацией запроса с предопределенными параметрами.
- г) после завершения всех тестовых сценариев, результаты каждого из них были сравнены с результатами CSRBench, которые опубликованы в сети Интернет по адресу <https://github.com/dellaglio/csrbench-oracle-engines>.

В таблице А.1 представлено сравнение результатов эксперимента CSRBench и того же самого эксперимента, но повторенного с помощью программного комплекса YABench. Хотя время выполнения тестовых сценариев

¹<http://www.w3.org/wiki/CSRBench>

достаточно короткое, всего 1 секунда, этого достаточно для того чтобы эффективно исследовать и сравнить результаты вручную. Галочка (\checkmark) означает что YABench получил абсолютно такие же результаты как в эксперименте CSRBench. Ячейки с крестиками (\times) в колонках CSRBench означают что в эксперименте с CSRBench СУПД вернул некорректные результаты. То же самое относится и к ячейкам в колонках YABench.

Таблица A.1 — Сравнение результатов эксперимента CSRBench с результатами YABench

Query	C-SPARQL		CQELS	
	CSRBench	YABench	CSRBench	YABench
Q1	\checkmark	\checkmark	\checkmark	\checkmark
Q2	\checkmark	\checkmark	\checkmark	\checkmark
Q3	\checkmark	\checkmark^*	\checkmark	\checkmark
Q4	\checkmark	\checkmark	\times	\times
Q5	\times	\times	\checkmark	\checkmark
Q6	\checkmark	\checkmark^*	\times	\times
Q7	\checkmark	\checkmark^*	\times	\times^{**}

Галочки со звездами (\checkmark^*) означают, что результаты YABench по большей части равны результатам CSRBench, однако некоторых результатов нехватает. После глубокого изучения мы обнаружили, что недостающие результаты были очень близки к границам окна и появились уже в результатах следующего окна. Эта проблема объясняется временными расхождениями, которые случаются от того что тестовые сценарии запускались на разных машинах и несколько раз. Далее результаты следующего окна были изучены

и было обнаружено, что потерянные результаты верны, но появились уже в последующем окне.

Крестики в ячейках YABench означают, что соответствующая СУПД не прошла тест, и результаты отличны от результатов CSRBench. Однако результаты этих же тестов но с CSRBench так же отрицательны. Крестики с двойными звездами (\times^{**}) означают, что во время тест провалился из-за ошибке в работе СУПД. Такой случай произошел с запросом Q7, где СУПД CQELS остановило свою работу с ошибкой, что говорит об ошибке в коде СУПД.

Все конфигурации тестовых сценариев для этого исследования и исходные коды доступны в сети Интернет по адресу <https://github.com/YABench/csrbench-validation>.

В результате этого исследования, приходим к выводу, что за исключением мелких различий, который объяснены выше, YABench повторяет результаты эксперимента CSRBench, поэтому может быть рассмотрен как корректный программный комплекс для тестирования СУПД на корректность и производительность. Стоит заметить, что в отличии от существующих решений YABench предоставляет дополнительные возможности по тестированию СУПД: конфигурирование тестовых сценариев; выполнение тестовых сценариев независимо от операционной семантики СУПД; глубокий анализ результатов экспериментов на основе новых метрик, учитывающих как производительность, так и корректность СУПД.

A.3.3 Исследование метода и алгоритма анализа потоковых и статических данных, и метода обнаружения и генерации событий предметной области

В данном разделе описаны исследование метода и алгоритма анализа потоковых и статических данных, и метода обнаружения и генерации событий предметной области. Данные методы и алгоритмы реализованы в виде отдельного модуля ПО ЭО ПАП, для которого был замерен максимальный объем динамической (кучи) памяти во время анализа данных с 100, 500 и 1000 электронных потребительских устройств, а также среднее время задержки

между временем, когда событие произошло и когда оно было идентифицировано. Как и в предыдущих исследованиях, в качестве таких устройств использовался симулятор общедомовых счетчиков тепла.

На рисунке А.12 представлены результаты замеров максимального объема динамической (кучи) памяти в зависимости от количества подключенных счетчиков. Так же стоит указать, что интервал между показаниями каждого из счетчиков составляет 1 минуту. Результаты в порядке возрастания количества счетчиков: 173.2 Мб, 205 Мб и 251.4 Мб.

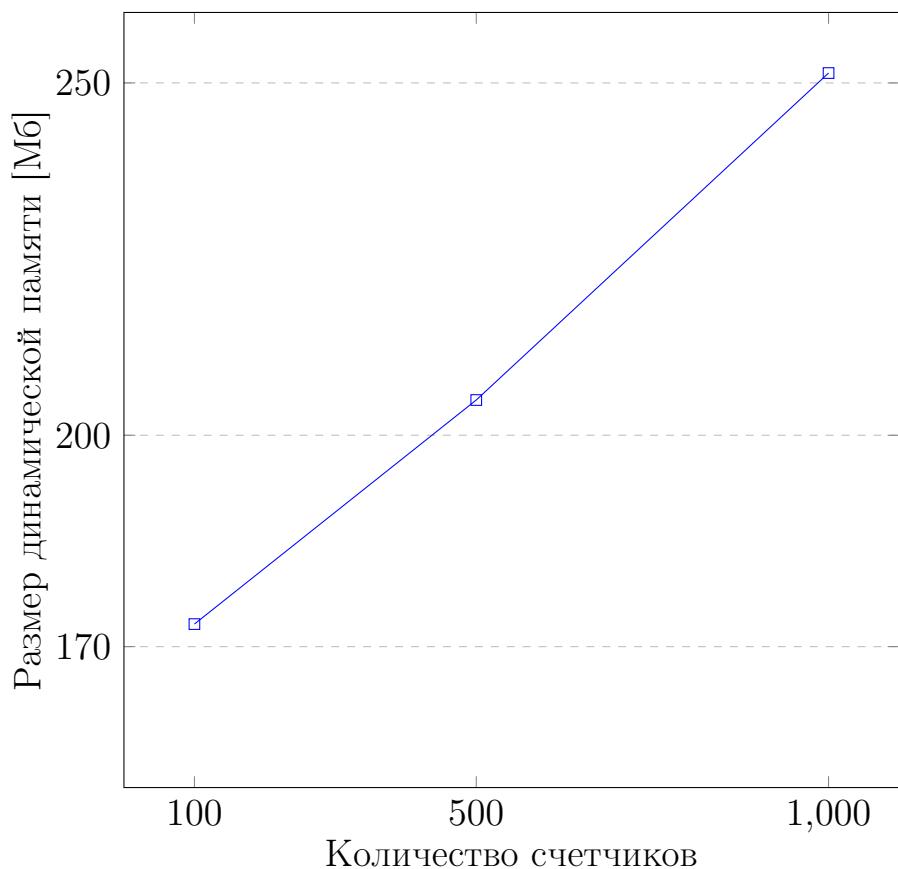


Рисунок А.12 — Максимальный объем динамической (кучи) памяти во время анализа данных с 100, 500 и 1000 счетчиков

Среднее время задержки между временем когда событие произошло и когда оно было идентифицировано так же было замерено для 100, 500 и 1000 счетчиков. В качестве события, которое должно было быть идентифицировано, служило превышение температуры энергоносителя порога в 80 градусов. На рисунке А.13 представлен CQELS-запрос, который использовался для идентификации этого события.

```

1 PREFIX hmtr: <http://purl.org/NET/ssnext/heatmeters#>
2 PREFIX meter: <http://purl.org/NET/ssnext/meters/core#>
3 PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5
6 SELECT ?meter ?value {
7   STREAM ?stream [NOW] {
8     ?obs a emtr:HeatObservation ;
9     ssn:observedBy ?meter ;
10    ssn:observationResult ?result .
11    ?result ssn:hasValue ?v .
12    ?v meter:hasQuantityValue ?value .
13  }
14  FILTER(?value > 80)
15 }
```

Рисунок А.13 – CQELS запрос на идентификацию превышения температуры энергоносителя

На рисунке А.14 представлен график среднего времени задержки и количества подключенных счетчиков. Среднее значение вычислено на основе 5 замеров. Результаты в порядке возрастания количества счетчиков: 3.2 сек, 5.2 сек и 10.9 сек.

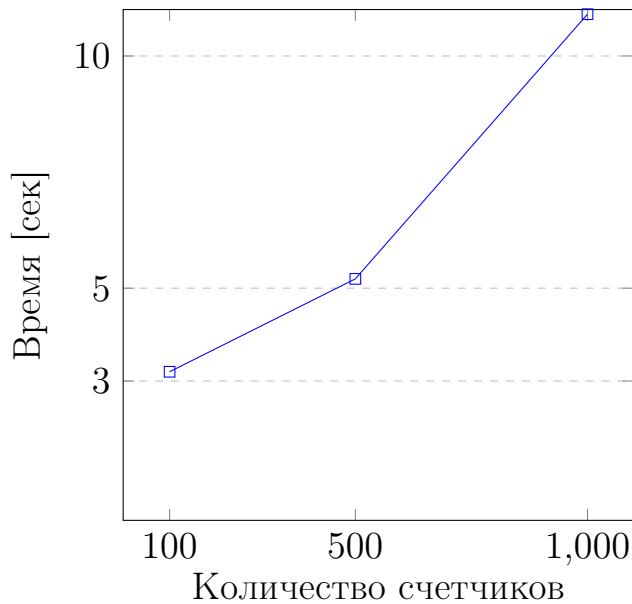


Рисунок А.14 – Среднее время задержки идентификации события с 100, 500 и 1000 счетчиками

A.4 Исследование математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data)

В данном разделе описаны исследования технических характеристик методов и алгоритмов описанных в разделе 3, а именно:

- а) метода и алгоритма доступа к статическим данным на основе семантических веб технологий и технологии связанных данных,
- б) алгоритма доступа к историческим данным распределенной сети электронных потребительских устройств,
- в) метода и алгоритма доступа к потоковым данным распределенной сети электронных потребительских устройств,
- г) алгоритма визуализации потоковых и исторических данных.

В соответствии с пунктом 4.2.4 Технического задания, методы и алгоритмы исследуемые в данном разделе должны отвечать трем требованиям, на основе которых проведены соответствующие эксперименты.

Аналогично предыдущим разделам в качестве электронных потребительских устройств используются общедомовые счетчики тепла. Для 300 и 1000 счетчиков проведены три эксперимента, которые оценивают среднее время сбора данных для построения графических форм, среднее время отклика системы для операций навигации и среднее время построения визуальных форм для представления данных. Каждый эксперимент был проведен не менее 5-ти раз и выбран результат эксперимента наиболее близкий к среднему значению.

На рисунках А.15, А.16 и А.17 представлены результаты трех перечисленных экспериментов для 300 одновременно подключенных счетчиков. Соответственно среднее время сбора данных для построения графических форм равно 747 мс, среднее время отклика системы для операций навигации равно 59 мс, а среднее время построения визуальных форм для представления данных равно 93 мс.

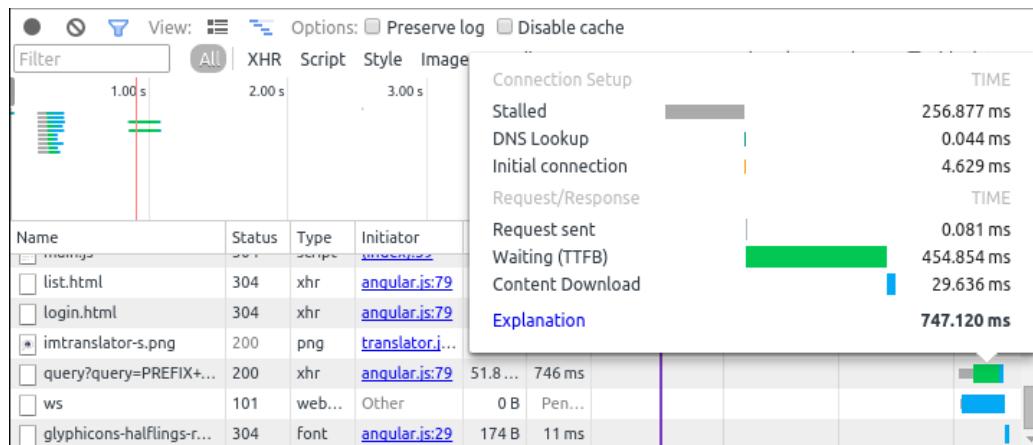


Рисунок А.15 – Среднее время сбора данных для построения графических форм (300 счетчиков)

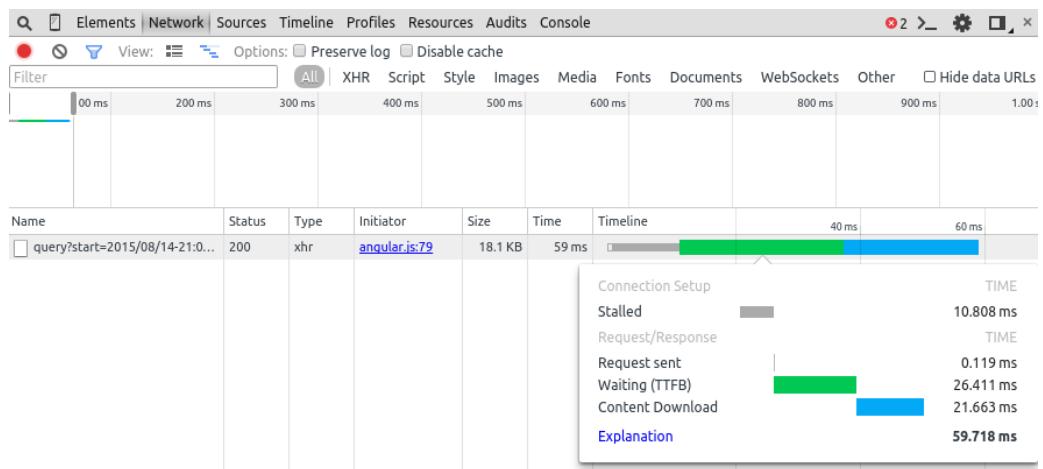


Рисунок А.16 – Среднее время отклика системы для операций навигации (300 счетчиков)

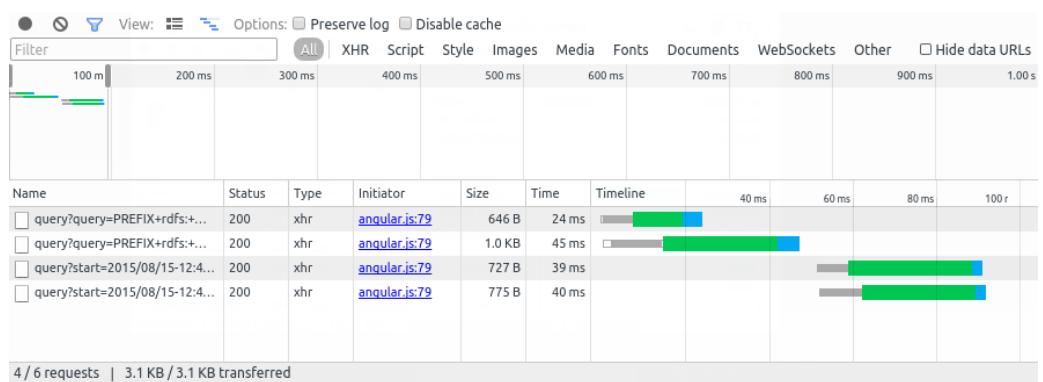


Рисунок А.17 – Среднее время построения визуальных форм для представления данных (300 счетчиков)

На рисунках А.18, А.19 и А.20 представлены результаты трех перечисленных экспериментов для 1000 одновременно подключенных счетчиков.

Соответственно среднее время сбора данных для построения графических форм равно 816 мс, среднее время отклика системы для операций навигации равно 51 мс, а среднее время построения визуальных форм для представления данных равно 178 мс.

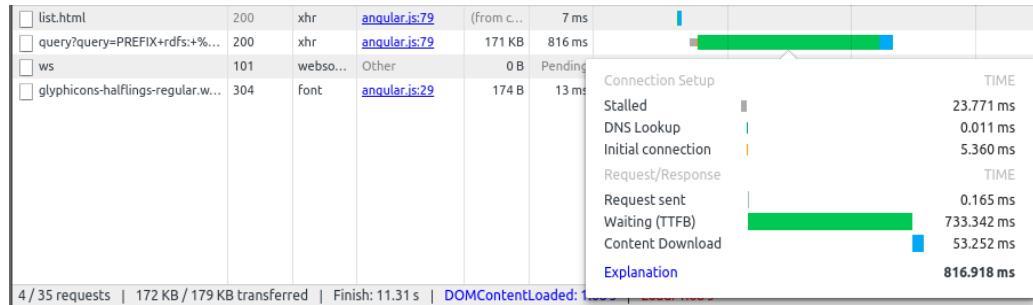


Рисунок А.18 – Среднее время сбора данных для построения графических форм (1000 счетчиков)

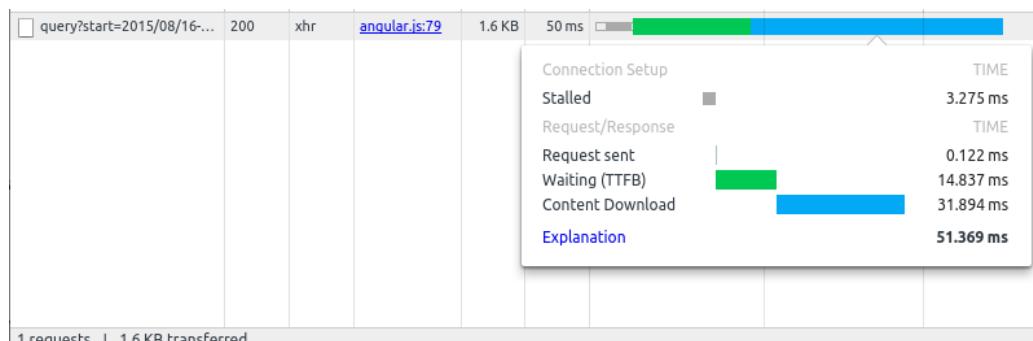


Рисунок А.19 – Среднее время отклика системы для операций навигации (1000 счетчиков)

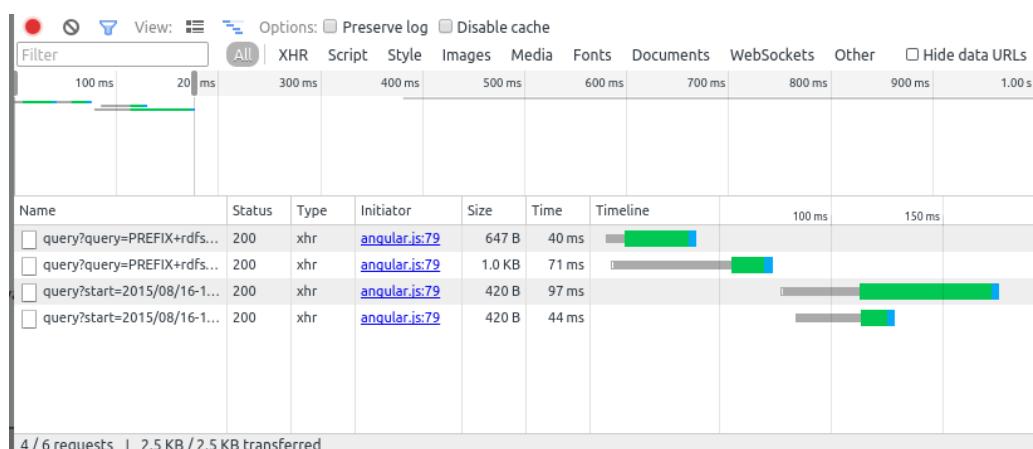


Рисунок А.20 – Среднее время построения визуальных форм для представления данных (1000 счетчиков)

На основе представленных результатов трех экспериментов соответствующих требованиям Технического задания можно сделать вывод, что методы и алгоритма визуализации данных описанные в разделе 3 данного отчета о ПНИ отвечают требованиям Технического задания.

A.5 Результаты

В данном отчете представлены результаты исследований методов и алгоритмов разработанных на данном этапе ПНИ относящиеся к следующим работам по Плану графику:

- а) п.2.1 ПГ Разработка математических методов и алгоритмов агрегации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных с применением приемов онтологического инжиниринга и технологий семантического веба.
- б) п.2.2 ПГ Разработка математических методов и алгоритмов нормализации и анализа больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных генерируемых в распределенной сети электронных потребительских устройств.
- в) п.2.3 ПГ Разработка математических методов и алгоритмов интерактивной визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных на основе технологии связанных данных (Linked Data).

Результаты исследований разработанных методов и алгоритмов показали их полное соответствие требованиям Технического задания.

Приложение Б

Отчет об участии в мероприятиях

Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев

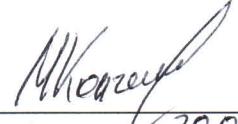
«30» июня 2015 г.

ОТЧЕТ ОБ УЧАСТИИ В МЕРОПРИЯТИЯХ

ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы»

Соглашение о предоставлении субсидии от 24.11.2014 г. №14.575.21.0101

Ответственный исполнитель работ


М.А. Колчин
(подпись, дата)

Санкт-Петербург 2015

Б.1 Аннотация

В данном отчете представлены отчетные и справочные материалы об участии в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ (конференции, семинары, симпозиумы, выставки и т.п., в том числе, международные).

Б.2 Об участии в 17-й международной конференции FRUCT, г. Ярославль, Россия

Международная конференция FRUCT в 2015-году проходила с 20-го по 24-е апреля 2015 года в городе Ярославль, Россия. Официальный сайт конференции с программой мероприятия и другой информацией доступен по адресу <http://fruct.org/program17>. Копия программы мероприятия прилагается к данному отчету.

В рамках данного мероприятия Исполнителями проекта был представлен доклад «An implementation of CoAP protocol for Arduino and ESP8266». Авторы доклада: Андреев Алексей, Муромцев Дмитрий, Колчин Максим, Климов Николай, Гарайзуев Даниил, Шилин Иван. Презентация данного доклада с отметкой о финансовой поддержки работы Министерством Образования и Науки в рамках гранта №RFMEFI57514X0101 доступна по адресу <https://fruct.org/sites/default/files/files/conference17/02.pdf>, а также прилагается к данному докладу.

Данный доклад отражает промежуточные результаты работы по данному ПНИ в рамках работы 2.1 по Плану-графику работ, которые также отражены в разделе 1.2 данного отчета о ПНИ.

Б.3 Об участии в 12-й международной конференции Extended Semantic Web Conference (ESWC 2015), г. Порторож, Словения

В рамках международной конференции Extended Semantic Web Conference 2015¹, которая проходила с 31-го мая по 4-е июня 2015 года в городе Порторож, Словения, так же прозодил семинар под названием «RDF Stream Processing Workshop», на котором был представлен доклад «Demo: YABench - Yet Another RDF Stream Processing Benchmark». Авторы доклада: Колчин Максим и Питер Витц.

Программа семинара и другая информацию представлена на официальном сайте мероприятия доступному по адресу <https://www.w3.org/community/rsp/rsp-workshop-2015>. Копия программы с докладом Исполнителей проекта прилагается к данному отчету. Так же на сайте мероприятия опубликована презентация доклада (https://www.w3.org/community/rsp/files/2015/07/RSP_Workshop_2015_submission_14_slides.pdf), копия этой презентации прилагается к отчету.

Данный доклад отражает промежуточные результаты работы по данному ПНИ в рамках работы 2.2 по Плану-графику работ, которые также отражены в разделе 2.5 данного отчета о ПНИ.

¹Cf. <http://2015.eswc-conferences.org/>

Б.4 Результаты

В данном отчете представлены отчетные и справочные материалы об участии в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ (конференции, семинары, симпозиумы, выставки и т.п., в том числе, международные).

В рамках этапа 2 данного ПНИ было принято участие в следующих мероприятиях:

- а) 17-я международная конференция FRUCT, г. Ярославль, Россия,
- б) 12-я международная конференция Extended Semantic Web Conference (ESWC 2015), г. Порторож, Словения.

Работа по п.2.4 Плана графика работ по участию в мероприятиях, направленных на освещение и популяризацию промежуточных результатов ПНИ выполнена в полном объеме.