МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Системное программирование и компьютерные технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Применение алгоритмов интеллектуального анализа данных в системе мониторинга и контроля сетевого оборудования»

Студент А.Е. Исайков
(И.О. Фамилия) (личная подпись)

Руководитель доцент, к.п.н., О.Ю. Копша
(ученая степень, звание, И.О. Фамилия)

Консультант старший преподаватель, М.А. Четаева
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема бакалаврской работы: «Применение алгоритмов интеллектуального анализа данных в системе мониторинга и контроля сетевого оборудования».

Выпускная квалификационная работа посвящена разработке системы, занимающейся контролем и мониторингом сетевого оборудования пользователя и провайдера. В данной работе используются следующие алгоритмы: алгоритм ближайших соседей (k-NN – Nearest Neighbours), алгоритм k-средних (k-means), алгоритм линейной регрессии.

Объект исследования – сетевое оборудование, система мониторинга, искусственный интеллект.

Предмет исследования – система мониторинга и контроля сетевого оборудования, алгоритмы искусственного интеллекта.

Цель бакалаврской работы – применение алгоритмов интеллектуального анализа в разработанной программной реализации системы мониторинга.

Для достижения цели работы необходимо решить следующие задачи:

1. Изучить существующее сетевое оборудование;

2. Проанализировать рынок и выявить необходимость в системах мониторинга;

3. Разработать эмуляцию сетевого оборудования со стороны провайдера и пользователя;

4. Разработать систему, собирающую информацию о сетевом оборудовании и позволяющую производить манипуляцию с ним;

5. Применить алгоритмы интеллектуального анализа данных для автоматизации системы.

Бакалаврская работа представлена на 67 страницах, включает 22 иллюстраций, 14 формул, список используемой литературы, состоящий из 22 источников.

Abstract

The title of the bachelor’s thesis is "Application of algorithms of data mining in the monitoring and control system of network equipment".

The bachelor’s thesis is devoted to the development of a system that monitors the user and provider’s network equipment.

The following algorithms are used in this work: the nearest neighbors algorithm (k-NN - Nearest Neighbors), the k-means algorithm, the linear regression algorithm.

The object of the study is network equipment, monitoring systems, artificial intelligence.

The subject of the research is a monitoring and control system of network equipment, artificial intelligence algorithms.

The goal of the bachelor’s thesis is to use algorithms of data mining in the developed software implementation of the monitoring system.

To achieve the goal of the work it is necessary to solve the following tasks:

1. Examine existing network equipment;

2. To analyze the market and identify the need for monitoring systems;

3. Develop network equipment emulation by the provider and user;

4. Develop a system that collects information about network equipment and allows you to manipulate it;

5. To apply algorithms of data mining to automate the system.

The bachelor’s thesis consists of an explanatory note on 67 pages, includes 22 figures, 14 formulas, a list of 22 references.

Оглавление

Введение 5
1 Изучение предметной области 8
1.1 Сетевое оборудование 8
1.1.1 Виды сетевого оборудования 9
1.1.2 Уровни взаимодействия сетевой модели OSI 20
1.2 Состояние рынка продуктовых решений 26
2 Разработка объектной модели сети 31
2.1 Анализ возможных способов взаимодействия сетевых устройств 31
2.2 Выбор сетевых устройств, для мониторинга и выделение их характерных признаков 37
2.3 Алгоритмы искусственного интеллекта 40
2.3.1 Алгоритм k-NN 40
2.3.2 Алгоритм k-means 45
2.3.3 Алгоритм линейной регрессии 47
3 Разработка системы мониторинга и анализ результатов 49
3.1 Описание общей схемы взаимодействия системы 49
3.2 Описание модулей программы 55
3.3 Описание базы данных 61
3.4 Интерфейс системы 62
3.5 Анализ результатов 65
Заключение 67
Список используемой литературы и используемых источников 68
Приложение А Листинг кода устройства CPE 70
Приложение Б Листинг кода устройства PE 76
Приложение В Листинг кода системы мониторинга 82
Приложение Г Листинг кода страницы интерфейса 95

Введение

Сетевые технологии – это совокупность программно-аппаратных средств, основанных на согласованном наборе стандартных протоколов, достаточных для построения локальной вычислительной сети.

С момента появления первой локальной сети прошло более пятидесяти лет и с тех пор развитие информационных технологий этой области не становилось ни на секунду. Появилось огромное количество фирм, выпускающих сетевое оборудование для организации локальной сети, выхода в глобальную сеть или подключения беспроводных точек доступа. Раз в несколько лет выпускаются новые стандарты для сетей, сетевых кабелей, протоколов и т.д.

Система мониторинга и управления сети – это совокупность технических средств, которые на постоянной основе осуществляют наблюдение, сбор информации, автоматическое или ручное управление оборудованием с целью выявления и устранения неработоспособных узлов сети.

Опираясь на исследование компании, которая специализируется на составлении аналитических отчетов «Persistence Market Research», общемировой рынок управления и мониторинга ИТ-инфраструктуры к 2024 году будет оцениваться более чем в 34 миллиарда долларов, что в 3 раза больше, чем его оценка в 2016 году.

Данная область является перспективной с точки зрения развития как физических устройств (сетевых адаптеров, коммутаторов, маршрутизаторов, кабелей), так и различного программного обеспечения для работоспособности активных элементов сети (драйвера, прошивки), куда можно отнести и системы мониторинга и контроля за сетевым оборудованием.

Система мониторинга сетевого оборудования может заблаговременно сообщить о неполадке в сети, чем поможет системным администраторам

устранить ее, не дожидаясь отказа устройств или всей системы. Данная особенность позволит не только сохранить стабильную работу сети, но и сэкономит бюджет, будь то покупка нового оборудования или стоимость простоя для крупной организации.

Современные системы позволяют строить графы и просматривать графики собранных данных в реальном времени и накапливать статистические данные в распределенные базы данных, для последующего анализа, поиска неисправностей или наличия дыр в безопасности. Мониторинг сетевого оборудования входит в комплексный мониторинг ИТ-инфраструктуры, являясь его важной составляющей, от качественного выполнения которой зависит надежное функционирование всей системы в целом.

Объект исследования: процесс мониторинга и контроля сетевым оборудованием, искусственный интеллект.

Предмет исследования: системы мониторинга и контроля сетевого оборудования, алгоритмы искусственного интеллекта.

Цель данной работы — применение алгоритмов интеллектуального анализа в разработанной программной реализации системы мониторинга.

Для реализации поставленной цели, необходимо сформулировать следующие задачи:

Задачи:

1. Изучить существующее сетевое оборудование;
2. Проанализировать рынок и выявить необходимость в системах мониторинга;
3. Разработать эмуляцию сетевого оборудования со стороны провайдера и пользователя;
4. Разработать систему, собирающую информацию о сетевом оборудовании и позволяющую производить манипуляцию с ним;
5. Применить алгоритмы интеллектуального анализа данных для автоматизации системы.

Обзор по главам:

1. В первой главе приводиться общая информация о существующем сетевом оборудовании, сетевой модели OSI и существующих продуктовых решениях в сфере систем мониторинга;

2. Во второй главе рассматриваются возможные взаимодействия сетевых устройств, создается модель исследуемой сети и анализируются алгоритмы автоматизации контроля над сетью;

3. В третьей главе представлена программная реализация объектов сети и системы мониторинга, описание общей схемы взаимодействия и производится анализ полученной системы и возможности ее дальнейшего развития.

1 Изучение предметной области

1.1 Сетевое оборудование

На сегодняшний день в мире существует более 130 миллионов компьютеров, и более 80% из них объединены в различные информационно-вычислительные сети, от малых локальных сетей в офисах, до глобальных сетей.

Острую необходимость ускорения передачи информационных сообщений, обмена информационными сообщениями, не отходя от рабочего места (факс, E-mail письма), быстрого обмена информацией между людьми, обменом информацией между компьютерами в любой точке мира обуславливает всемирная тенденция к объединению компьютеров в сети, начиная от маленьких локальных, и, заканчивая сетями, насчитывающими тысячи вычислительных машин.

Всего этого невозможно достичь без устройств, предназначенных для создания вычислительных сетей. Эти устройства определяются общим понятием «сетевое оборудование».

Сетевое оборудование — совокупность устройств, необходимых для создания сети, например: коммутатор, маршрутизатор, концентратор, мост, сетевой адаптер, патч-панель и многие другие. Всё сетевое оборудование можно разделить на две большие категории: активное сетевое оборудование и пассивное сетевое оборудование[2].

Активное сетевое оборудование (АСО) – содержащее электронные схемы, получающее питание от электрической сети и выполняющее функции преобразования сигналов, усиления сигналов и другие. В эту категорию попадают такие устройства, как: маршрутизатор, коммутатор (свитч) и т.д. Повторитель (репитер) и концентратор (хаб) не являются активным сетевым оборудованием, так как их задача сводится к повторению электрического сигнала для продолжения его перемещения без коллизий. Такие устройства

не дополняют электрический сигнал никакими битами информации, и, следовательно, не считаются за активное сетевое оборудование[7].

Под пассивным сетевым оборудованием подразумевается оборудование, не наделенное особенностями модификации сигнала. Например, кабель (вилка/розетка (RG58, RJ11, GG45), коаксиальный кабель и витая пара (UTP/STP)), патч-панель, повторитель (репитер), концентратор (хаб), балун (balun) для коаксиальных кабелей (RG-58) и т.д.

1.1.1 Виды сетевого оборудования

Выделяют следующие виды сетевого оборудования:

– сетевые карты,

– повторители,

– концентраторы,

– мосты,

– коммутаторы,

– маршрутизаторы,

– брандмауэры,

– беспроводные сетевые адаптеры,

– точки беспроводного доступа,

– и т.д;

Сетевые карты.

Устройства, связывающие пользователей с сетью, называются станциями, сетевыми адаптерами или оконечными узлами (host). Одним из примеров такого устройства является персональный компьютер пользователя или рабочая станция (мощная вычислительная машина, выполняющая определенные функции, требующая большой вычислительной мощности. Например, обработка потоков видео, аудио, рендер объектов, моделирование физических процессов и т.д.). Для работы в сети каждая станция оснащена платой сетевого интерфейса (Network Interface Card — NIC), другими словами – сетевым адаптером[1].

Сетевой адаптер является печатной платой, вставляемой в слот материнской платы компьютера, или внешнее устройство, подключаемое к порту. Каждое такое устройство (NIC) имеет уникальный код, присваиваемый ему на этапе производства, называемый MAC-адресом. Этот адрес используется для организации работы этих устройств в сети и служит уникальным идентификатором этого устройства. NIC обеспечивает передачу данных, которые необходимо передавать между персональными компьютерами в сети. Они удлиняют и объединяют кабельные соединения, преобразуют данные из одного формата в другой и управляют передачей данных.

Примерами устройств, выполняющих перечисленные функции, являются повторители, концентраторы, мосты, коммутаторы и маршрутизаторы.

Повторители.

Повторители (repeater) – это сетевые устройства, которые функционируют на первом (физическом) уровне эталонной модели OSI, разбор которого будет представлен в главе 1.1.2. Изначально данные покидают устройство отправителя и выходят в сеть в виде электрических или световых импульсов, которые передаются по сетевой передающей среде. Такие импульсы называются сигналами (signals). В начале передачи сигналы являются четкими и легко распознаваемыми, однако, по прошествии определенного расстояния сигналы начинают затухать, теряя свою частоту. Чем больше длина кабеля, тем более слабым и менее различимым становится сигнал по мере прохождения по сетевой передающей среде. Целью использования повторителя является регенерация и ресинхронизация сетевых сигналов на битовом уровне, что позволяет передавать их по среде на большее расстояние. Изначально термин повторитель (repeater) означал отдельный порт «на входе» некоторого активного сетевого оборудования и отдельный порт на его «выходе». В настоящее время используются повторители с несколькими портами, для регенерации сигналов с нескольких

сетевых сред. Повторители классифицируются как устройства первого уровня (в эталонной системе OSI), поскольку они функционируют только на битовом уровне, не просматривая содержащуюся в пакете информацию и не производя с ней каких-либо операций[6].

Концентраторы.

Концентратор — это простое сетевое оборудование, не оборудованное электронными схемами для передачи сообщений между узлами сети. Концентратор не должен определять, какому узлу сети предназначается конкретное сообщение, в его задачу входит прием электронного сигнала с одного порта и воспроизведение (ретрансляция) этого же сообщения на всех остальных портах.

Для получения и отправки сообщений все порты концентратора должны быть подключены к одному и тому же каналу. Из-за того, что все узлы концентратора работают на одной полосе одного канала, концентратор так же называют устройством с общей полосой пропускания.

Концентраторы имеют схожий функционал с повторителями, поэтому их еще называют – многопортовые повторители (multiport repeater). Разница между повторителем и концентратором состоит лишь в количестве кабелей, подсоединенных к устройству. В то время как повторитель может иметь только два порта, у концентраторов количество портов варьируется от 4 до 20 и более[12].

Свойства концентраторов:

– усиление сигналов,

– распространение сигнала по сети,

– не нуждаются в фильтрации сигнала,

– не нуждаются в определении маршрутов и определения границ и содержания пактов,

– объединяют трафик сети;

Функции концентраторов.

Концентраторы считаются устройствами первого уровня, поскольку они регенерируют сигнал и дублируют его на все подключенные порты (на выходные сетевые соединения). Сетевые карты принимают только те сообщения, которые адресованы именно на их MAC-адрес. Только узел, которому предназначалось сообщение, обрабатывает информацию и отвечает отправителю.

Концентратор дает возможность одновременной отправки только одного сообщения. Если два или более узла, подключенные к одному устройству, попытаются одновременно отправить сообщение, то произойдет столкновение электронных сигналов, из которых состоит сообщение. Результирующее сообщение будет искажено и узлы не смогут прочесть содержащуюся в них информацию[9].

Поскольку концентратор не декодирует сообщение – он не в состоянии обнаружить, что информация в сообщении искажена, и дублирует его всем портам сети. Область сети, в которой узел может получить искаженное при столкновении сообщение, называется доменом коллизий.

Внутри этого домена каждый узел, получивший искаженное сообщение, обнаруживает, что произошла коллизия и, по прошествии определенного времени, пытается заново отправить или переправить своё сообщение. По мере увеличения конечных узлов сети – растет и вероятность столкновений. Чем больше столкновений, тем больше будет повторов. При этом сеть перегружается, скорость передачи сетевого трафика падает, что приводит к большим проблемам всей сети. Теоретически существует количество узлов сети, при которых столкновения будут происходить на каждое сообщение, поэтому размер домена коллизий необходимо ограничивать[10].

Мосты.

Мост (bridge) представляет собой устройство второго уровня, предназначенное для создания двух или более сегментов локальной сети LAN, каждый из которых является отдельным коллизионным доменом.

Иными словами, мосты используются для более точечного отправления сигнала получателю. Главной целью моста является фильтрация потоков данных в LAN-сети с тем, чтобы локализовать внутрисегментную передачу данных и, вместе с тем, сохранить связи с другими частями (сегментами) LAN-сети для перенаправления туда потоков данных. Каждое сетевое устройство имеет связанный с NIC-картой уникальный MAC-адрес.

Мост собирает информацию о том, на каком из его портов находится тот или иной MAC-адрес, и принимает решение о пересылке данных. Для моста отличительным признаком узлов сети являются только их MAC-адреса. По этой причине они могут быстро пересылать данные, не смотря на протокол передачи. Мосты принимают решение о пересылке сообщения только на уровне: отправить или не отправить; которое основывается только на MAC-адресе получателя[17].

Ниже приведены наиболее важные свойства мостов:

– мосты, в отличие от концентраторов являются более «интеллектуальными» устройствами. Это обуславливается тем, что мост занимается анализом входящих фреймов при их пересылке, в то время как концентратор лишь реплицирует сигналы без анализа;

– мосты собирают и передают пакеты между двумя или более сегментами LAN-сети;

– мосты увеличивают количество доменов коллизий (и уменьшают их размер за счет сегментации локальной сети), что позволяет нескольким устройствам передавать данные одновременно, не вызывая коллизий;

– в абстрактном сравнении мосты работают медленнее, чем концентраторы именно из-за их аналитической работы, однако этот недостаток компенсируется значительным уменьшением коллизий;

– мосты составляют и поддерживают таблицу MAC-адресов сети.

Функция моста.

Отличительными функциями моста являются фильтрация фреймов на втором уровне и используемый при этом способ обработки трафика. Для

фильтрации или выборочной доставки данных мост создает таблицу всех MAC-адресов, расположенных в данном сетевом сегменте и в других известных ему сетях, и преобразует их в соответствующие номера портов. На рисунке 1 подробно описаны этапы работы моста с поступающими ему данными[17].

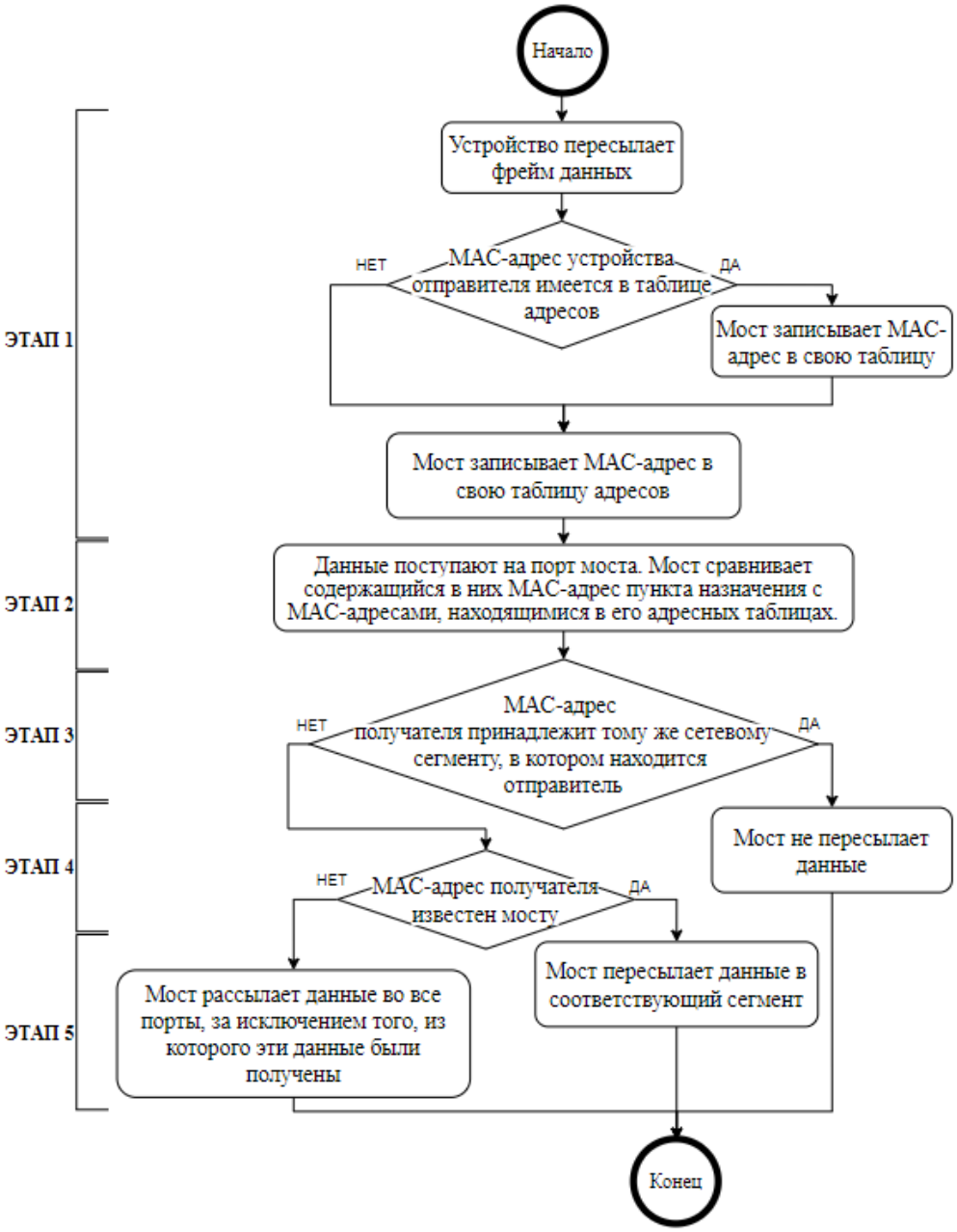

Рисунок 1 – Этапы обработки трафика мостами

Коммутаторы.

Коммутатор является улучшенной модификацией моста и концентратора. В самом простом использовании коммутатор можно назвать многопортовым мостом, однако в большинстве случаев функциональность коммутатора выходит за пределы данного упрощения[13].

Коммутатор представляет собой устройство второго уровня. Наподобие концентратора, коммутатор агрегирует несколько узлов в сеть, однако, в отличие от концентратора, коммутатор в состоянии передать сообщение конкретному узлу, определяя который по MAC-адресу получателя.

В таблице адресов коммутатора, которая называется таблицей MAC-адресов, находится список активных портов устройства и MAC-адреса подключенных к ним узлов. Для реализации обмена сообщениями коммутатор проверяет, есть ли в его таблице MAC-адрес получателя, если да, то коммутатор устанавливает между портом источника и получателя временное соединение, которое называется каналом связи. Данный канал представляет собой назначенный канал, по которому два узла могут обмениваться данными. Другие узлы, подключенные к коммутатору, работают на других полосах пропускания и не принимают сообщения, адресованные не им. Для каждой новой пары общающихся устройств коммутатор создает новый канал. Такие каналы позволяют устанавливать несколько соединений одновременно без возникновения коллизий на конкретном участке сети.

Поскольку коммутация осуществляется на аппаратном уровне, это дает значительный прирост производительности, по сравнению с мостом, у которого коммутация происходит посредством программного обеспечения (следует обратить внимание, что мост считается устройством с программной, а коммутатор с аппаратной коммутацией.). Каждый порт коммутатора можно рассматривать как отдельный микромост, который предоставляет каждой

рабочей станции всю полосу пропускания передающей среды. Такой процесс называется микросегментацией[14].

Микросегментация (microsegmentation) позволяет создавать, выделенные или частные сегменты, в которых присутствует только одна рабочая станция. Каждая такая станция захватывает полный доступ ко всей полосе пропускания в единицу времени, без конкуренции с остальными станциями за право доступа к передающей среде. У коммутаторов в дуплексном режиме отсутствуют коллизии, поскольку на каждый порт коммутатора приходится только одно устройство.

Коммутатор, как и мост, пересылает широковещательные пакеты всем узлам сети, поэтому в сети, которая использует коммутаторы, все сегменты должны рассматриваться как один широковещательный домен.

Некоторые современные коммутаторы, главным образом коммутаторы уровня предприятия, способны выполнять операции сразу на нескольких уровнях. Например, устройства серий Cisco восемьдесят пятой и шестьдесят пятой серии выполняют некоторые функции третьего уровня.

В некоторых случаях к порту коммутатора подключают другое сетевое устройство, например, концентратор. Это увеличивает количество узлов, доступных для подключения. Если к порту коммутатора подключен концентратор, то MAC-адреса всех узлов, подключенных к концентратору, начинают ассоциироваться с портом подключения. В данной сети возможен сценарий, когда одна рабочая станция концентратора шлет пакет другой рабочей станции этого же концентратора. В этом случае концентратор, по своей функциональности шлет сообщение всем портам, в том числе и коммутатору. Коммутатор же, принимая пакет – расшифровывает его, проверяет MAC-адреса получателя и отправителя, и, проверяя их по своей таблице, отклоняет сообщение, так как оба этих адреса ассоциируются с одним и тем же портом. При таком подключении, есть сценарии, при которых возникают коллизии. Концентратор, принимая одновременно несколько сообщений, теряет их различия и передает поврежденные при

столкновении сообщения всем портам. Коммутатор принимает поврежденное сообщение, но при расшифровке не может однозначно определить адреса получателя и отправителя, вследствие чего не отправляет сообщение дальше. В итоге у каждого порта коммутатора создается отдельный домен коллизий, дальше которой эти коллизии выйти не в состоянии. Чем больше узлов в домене коллизий, тем вероятность их возникновения увеличивается[1].

Маршрутизаторы.

Маршрутизатор (router) представляет собой активное сетевое устройство, которое занимается пересылкой пакетов между сетями на основе адресов третьего уровня.

Маршрутизаторы, функционирую на третьем уровне, способен выбирать наилучший путь в сети для передаваемых данных на основе сетевых адресов устройств, вместо использования индивидуальных MAC-адресов второго уровня, как это делают коммутаторы, концентраторы и т.д. В возможности маршрутизаторов так же входит возможность соединения сетей с различными технологиями, такими как: Token Ring, Ethernet и FDDI (Fiber Distributed Data Interface). FDDI — это распределенный интерфейс передачи данных, основанный на протоколе Token Ring. Передача данных происходит по волоконно-оптическим каналам. Основной задачей маршрутизатора является соединение сетей, использующих ATM (Asynchronous Transfer Mode), то есть асинхронную передачу данных, и последовательных соединений. Благодаря своей особенности в пересылке пакетов, на основе хранящихся в них адресов третьего уровня, маршрутизаторы стали основным устройством, используемым для подключения к интернету, используя стек протоколов TCP/IP[2].

Основной задачей маршрутизатора является инспектирование входящих пакетов, проверяя адресата, выбор кратчайшего или быстрейшего пути, по которым пакет будет доставлен на выходной порт. Маршрутизаторы делают возможным обмен информацией любым компьютерам. В большинстве сетей, в которых количество персональных компьютеров

достигает двухсот и свыше, маршрутизаторы являются главными устройствами, регулирующими перемещение по сети потоков данных.

В задачи маршрутизатора входит определение необходимости пересылки данных в другую сеть. Каким же образом происходит определение? В пересылаемой информации содержится множество пакетов, каждый из которых отвечает за свою часть работы. Первым делом устройство считывает IP-адрес получателя из надлежащего пакета. Следом происходит определение, по маске сети и подсети, какой из подключенных сетей передать информацию, чтобы она дошла до адресата. После этого считанный пакет снова запечатывается в сообщение и пересылается дальше по сети, до следующего узла[3].

Маршрутизатор используется, если сетевые части IP-адресов у отправителя и получателя не совпадают. Если узел, который находится, допустим, в сети 3.3.3.0, должен доставить пакет узлу в сети с маской 1.1.1.0, то такой пакет будет передан маршрутизатору, потому что это работа для сетевого оборудования третьего уровня.

Брандмауэры.

Обычно, под термином брандмауэр (firewall) понимают установленное программное обеспечение на компьютере, сервере или маршрутизаторе, однако существует отдельный компонент сите под таким же названием[20].

Целью брандмауэра является защита сети, компьютера или сервера от проникновения извне. Путем исследования каждого пакета, проходящего через него – брандмауэр анализирует содержание и, опираясь на определенный набор правил фильтрует входящий и исходящий трафик. Работу этого устройства можно сравнить с личным секретарем, который занят прочтением всех входящих писем и пакетов, отсеивая ненужные или опасные.

Брандмауэры имеют несколько политик, которые доступны для настройки. Существуют полностью пропускающие политики, при которых совершенно отсутствует фильтрация трафика. Данный режим является

опасным, и включать его не рекомендуется. Существует режим, который блокирует трафик и ничего не возвращает пользователю, либо блокирует и сообщает о возможной угрозе. Данные режимы полезны для анализа работы сети и срочного реагирования на возможные проникновения.

Голосовые устройства, DSL-устройства, кабельные модемы и оптические устройства.

Возникший в последнее время спрос на интеграцию голосовых и обычных данных и быструю передачу данных от конечных пользователей в сетевую магистраль привел к появлению следующих новых сетевых устройств:

а) голосовых шлюзов, используемых для обработки интегрированного голосового трафика и обычных данных;

б) мультиплексоров DSLAM, используемых в главных офисах провайдеров служб для концентрации соединений DSL»модемов от сотен индивидуальных домашних пользователей;

в) терминальных систем кабельных модемов (Cable Modem Termination System — CMTS), используемых на стороне оператора кабельной связи или в головном офисе для концентрации соединений от многих подписчиков кабельных служб;

г) оптических платформ для передачи и получения данных по оптоволоконному кабелю, обеспечивающих высокоскоростные соединения.

Беспроводные сетевые адаптеры.

Для работы с беспроводной сетью Wi-Fi необходим беспроводной сетевой адаптер NIC. Данные устройства могут быть представлены в форме плат PCMCIA, либо по виду напоминать флэш накопитель, который использует внешний интерфейс USB. Беспроводные адаптеры могут использоваться как на стационарный персональных компьютерах, так и на переносных ноутбуках[8].

Беспроводные адаптеры для шин PCI (Peripheral Component Interconnect — 32-разрядная системная шина для подключения периферийных устройств) и ISA (Industry-Standard Architecture — структура, соответствующая промышленному стандарту) для настольных рабочих станций позволяют добавлять к локальной сети LAN конечные станции легко, быстро и без особых материальных затрат. При этом не требуется прокладки дополнительных кабелей. Все адаптеры имеют антенну: карты PCMCIA обычно выпускаются со встроенной антенной, а PCI-карты комплектуются внешней антенной. Эти антенны обеспечивают зону приема, необходимую для передачи и приема данных.

Точки беспроводного доступа.

Точка доступа (Access Point — AP), называемая также базовой станцией, представляет собой беспроводной приемопередатчик локальной сети LAN, который выполняет функции концентратора, т.е. центральной точки отдельной беспроводной сети, или функции моста — точки соединения проводной и беспроводной сетей. Использование нескольких точек AP позволяет обеспечить выполнение функций роуминга (roaming), что предоставляет пользователям беспроводного доступа свободный доступ в пределах некоторой области, поддерживая при этом непрерывную связь с сетью[11].

1.1.2 Уровни взаимодействия сетевой модели OSI

В данном параграфе представлено краткое описание сетевой модели OSI (The Open Systems Interconnection model). Данная модель является сетевой моделью стека сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом, в зависимости от уровня устройства. Модель определяет различные уровни взаимодействия систем, где каждый уровень выполняет определённые функции. На рисунке 2 представлено графическое представление модели OSI.

Рисунок 2 – Уровни взаимодействия The Open Systems Interconnection model

Физический уровень (Physical Layer).

На физическом уровне происходит работа с битами, а точнее – передача битов по каналу связи (витая пара, оптоволоконный кабель). На данном уровне задаются атрибуты электрических сигналов, которые содержат в себе дискретную информацию, такую как: частота сигнала, амплитуда, фаза, и т.д. К физическому уровню взаимодействия также относится и физическая среда, её характеристики: волновое сопротивление, полоса пропускания, помехозащищенность[3].

На физическом уровне базируются следующие стандарты: 100BASE-T или 1000BASE-X, где цифра в начале обозначает максимальную скорость передачи (100 – 100 Мегабит, 1000 – Гигабит), а буквы в конце означают физическую среду (T – витая пара, X – оптоволокно).

Функции физического уровня реализуются сетевым адаптером или последовательным портом.

Канальный уровень (Data link Layer)

На канальном уровне происходит передача данных между соседними узлами, расположенными в одной сети. Узлом является любое устройство, подключенное к сети[2].

Канальный уровень производит адресацию информации по MAC-адресам, которые задаются на этапе производства, компанией, выпускающей устройство. Каждый сетевой адаптер имеет свой уникальный MAC-адрес.

Этот уровень преобразует поступившую с верхних уровней информацию в биты, которые разбиваются на кадры (фрагменты данных). Эти кадры отдаются на нижний, физический, уровень, где будут передаваться по физическому каналу связи. На рисунке 3 представлена схема работы канального уровня.

Для взаимодействия между этими двумя уровнями используются кадры.

Примерная схема пересылки кадра:

– Канальный уровень отправляет кадр физическому уровню, который отправляет кадр в сеть;

– Каждый узел сети получает кадр;

– Каждый узел проверяет, соответствует ли адрес получателя в кадре с его собственным адресом;

– Если адреса не совпадают, кадр игнорируется. Если адреса совпадают то канальный уровень этого узла принимает кадр и передает его в сетевой уровень[2].

Канальный уровень делится на два подуровня: LLC (Logical Link Control) - подуровень логической передачи данных и MAC (Media Access Control) - подуровень управления доступом к среде. Каждый из этих подуровней определяет работу уровня с верхним, или нижним уровнями.

Подуровень LLC необходим для связи с верхним уровнем. На этом слое реализуются особые алгоритмы (процедуры на основе логики), которые позволяют определить необходимо ли установление связи, для передачи

кадров, или имеется ли необходимость восстановления кадров, при возникновении ошибок передачи[7].

Подуровень MAC определяет особенности доступа к физическим каналам связи, когда используются различные технологии локальных сетей. Протоколы MAC-уровня ориентированы на совместное использование физической среды абонентами. Разделяемая среда (shared media) используется в таких широко распространенных в локальных сетях технологиях как Ethernet, Fast Ethernet, Gigabit Ethernet, Token Ring, FDDI. Использование разделяемой между пользователями среды улучшает загрузку канала связи, удешевляет сеть, но ограничивает скорость передачи данных между двумя узлами.

Сетевой уровень (Network Layer)

Предыдущий уровень обеспечивает связь по локальной сети только между компьютерами, которые соединены индивидуальными линиями связи. Соединение сетей обеспечивается на сетевом уровне модели OSI[17].

Для соединения локальных сетей используются специальные устройства третьего уровня – маршрутизаторы. Задачей маршрутизатора является сбор информации о топологии межсетевых соединений. На основании собранных данных происходит передача информационного сообщения сетевого уровня в сеть получателя. Сообщение формируется из фреймов и собирается в пакет. Для передачи такого пакета от отправителя к получателю, находящемуся в другой локальной сети, необходимо провести этот пакет через несколько сетей, встречающихся на пути (транзитных сетей). Такие передачи называются хопы (от англ. hop — прыжок). Для каждой пары отправитель – получатель каждый раз выбирается новый, оптимальный, маршрут.

Сетевой уровень использует некоторое количество видов протоколов, таких как: сетевые протоколы, обеспечивающие передвижение пакетов по сети; протоколы разрешения адреса ARP (Address Resolution Protocol); протоколы маршрутизация RIP и OSPF.

Классические примеры протоколов сетевого уровня: IP (стек TCP/IP), IPX (стек Novell).

Транспортный уровень (Transport Layer)

Транспортный уровень занимается обеспечением, требуемого приложению, или верхнему уровню (сеансовому или прикладному), надлежащего уровня надежности соединения, так как на пути пересылки сообщения, от отправителя к получателю, пакет может быть искажен или вовсе утерян. Некоторые приложения берут на себя работу над этой задачей, но в большинстве случаев этим занимается данный уровень[19].

На транспортном уровне определены пять основных классов сервиса, которые характеризуют его:

1. Срочность.
2. Восстановление прерванной связи.
3. Наличие средств мультиплексирования нескольких соединений.
4. Обнаружение ошибок.
5. Исправление ошибок.

Начиная с этого уровня, в модели OSI, все последующие уровни реализуются на программном уровне специально разработанные для этого компоненты операционной системы, так как имеют большой функционал и сложную логику работы.

Примеры протоколов транспортного уровня: TCP и UDP (стек TCP/IP), SPX (стек Novell).

Сеансовый уровень (Session Layer)

Сеансовый уровень занимается установкой и разрывом соединения между компьютерами, управлением диалога между ними, а также предоставлением средств синхронизации. Данный уровень позволяет производить передачу информации в полудуплексном или дуплексном режиме. Дуплексный режим – это режим, в котором две стороны одновременно могут как принимать, так и отправлять информацию.

Полудуплексный режим, это режим, в котором в единицу времени происходит передача информации только в одном направлении[9].

Благодаря средствам синхронизации становится возможным выставление определенные контрольные информационные точки, при работе с длинными передачами, чтобы, в случае возникновения непредвиденной ситуации, не нужно было начинать передачу информационного сообщения с начала, а просто вернуться к последней точке, до которой соединение считалось стабильным.

Сеансом называют логическое соединение между двух компьютеров. У каждого сеанса имеется три фазы:

1. Установление соединения. Здесь узлы «договариваются» между собой о протоколах и параметрах связи.

2. Передача информации.

3. Разрыв связи.

Сеанс, на пятом уровне модели OSI – не тоже самое, что и сеанс связи, при выходе в интернет. Для сеанса связи пользователь устанавливает соединение с Интернетом, но может и начинать принимать или передавать данные, то есть не иметь ни с кем логического соединения.

Представительный уровень (Presentation Layer)

Представительный уровень – это уровень, который занимается только изменением формы передаваемой информации. На этом уровне не происходит изменение содержаний пакетов. Средствами этого уровня можно изменить кодировку информации из одной в другую, зашифровать или дешифровать данные, сжать или распаковать данные.

Примером протокола представительного уровня служит протокол SSL (Secure Socket Layer), который обеспечивает зашифрованный обмен данными.

Прикладной уровень (Application Layer)

Прикладной уровень работает на наборе разнообразных протоколов, с помощью которых пользователи сети получают доступ к совместно

используемым ресурсам. Единица данных, в этом протоколе, называется сообщением[15].

Примеры протоколов:

- HTTP – протокол гипертекстовых документов,
- FTP – протокол передачи данных по сети,
- TFTP – протокол передачи данных по сети без аутентификации,
- SMTP – протокол отправки почтовых сообщений,
- POP – протокол почтового отделения,
- SMB – протокол удаленного доступа к сетевым ресурсам,
- NFS – протокол для подключения (монтирования) файловых систем по сети.

1.2 Состояние рынка продуктовых решений

Программа для мониторинга сети – позволяет отслеживать и оперативно реагировать на деятельность внутри определенной локальной сети, позволяет отслеживать различные сетевые процессы, собирать вся информацию, с помощью которой возможно произвести автоматизацию части работы, которой занимался сетевой администратор. В первую очередь это касается обеспечение сетевой безопасности, системные сбои и так далее.

В данном разделе будет разбор нескольких систем, которые уже имеются на рынке в этой области, выявление их достоинств и недостатков.

Вот некоторые из них:

Total Network Monitor 2 (TNM 2) – доступное программное решение для отслеживания состояния серверов на рынке. Обладает обширным функционалом, графическим интерфейсом для пользователя. Главным инструментом программы являются мониторы, которые выполняют проверки системы с заданной периодичностью. Данные мониторы могут отслеживать огромное количество параметров сети, начиная от проверки состояния

сервисов на разных портах и заканчивая обычное проверкой работоспособности сервисов.

Система оснащена автоматическими скриптами, которые реагируют на происходящие в системе аномалии. Скрипты умеют перезагружать отдельные службы, активировать или деактивировать антивирус, вести журнал событий и так далее. Большинство из этих действий происходит автоматически, то есть без участия системного администратора.

В журналах событий хранится вся информация по всем проверкам системы, каждым из мониторов. Стоимость одной копии приложения, без настройки и поддержки стоит примерно пять тысяч рублей.

Достоинства:

– самая низка цена среди конкурентов,

– для установки не требуется специальных навыков,

– имеется графический интерфейс;

Недостатки:

– отсутствие дашбордов,

– не поддерживает многопоточность,

– нет поддержки и обновлений;

Observium – программное решение, которое работает на основе протокола SNMP. Главной его особенностью является упор на анализ производительности системы. Это решение интегрируется с оборудованием от Cisco, Windows, Linux, HP, Juniper, Dell, FreeBSD, Brocade, Netscaler, NetApp и прочих вендоров[11].

Имея богатый графический интерфейс настроек системы, данное ПО позволяет системному администратору подстроить настройки решения под каждую сеть, начиная от диапазонов для автообнаружения и заканчивая данными протокола SNMP, необходимыми для сбора информации о сети.

Использую Observium администратор получает полный доступ к описанию всех характеристик оборудования, которое подключено к локальной сети.

Все отчеты, формируемые на основе журналов событий, Observium представляет в виде диаграмм, что позволяет даже не опытному пользователя понять все "слабые" стороны сети.

Данное решение предоставляет на выбор демо-версию своей программы (месячная подписка), либо платную лицензию. Стоимость годовой подписки варьируется в районе восемнадцати тысяч рублей.

Достоинства:

– имеется бесплатная пробная версия,

– функции автоматического обнаружения нового оборудования,

– интегрирована со многим оборудованием;

Недостатки:

– нет мобильного приложения,

– для установки нужна инструкция и знание терминала Linux,

– не поддерживает сети с большим количеством оборудования;

Nagios – еще одно решение мониторинга и управление сети, которое основано на веб-интерфейсе. Данная программа сложна в освоении, однако благодаря хорошей документации, интернет-сообществу и активной поддержки пользователей, со стороны компании – обучение пройдет за несколько недель[13].

С помощью данного продуктового решения системные администраторы получают возможность отслеживать объем нагрузки на оборудование в режиме реального времени, регулируя данный объем. Так же система позволяет отслеживать степенью загруженности резервов памяти в базах данных и следить за показателями отдельных частей сетевого оборудования, таких как температура на материнской плате и т.д.

Nagios предоставляет, в течение 60-ти дней бесплатную демо-версию, а полная стоимость самой дешевой версии составляет сорок две тысячи рублей, в которую входит годовая подписка с обслуживанием, и ограничение до 20 объектов в сети. Самая дорогая будет стоит около 315.000 рублей, однако она не имеет ограничение на количество устройств сети.

Достоинства:

– более высокая гибкость, по сравнению с предыдущими решениями,

– имеется интеграция с другими приложениями, в том числе мобильными;

Недостатки:

– самая трудоемкая настройка из всех представленных решений,

– целевой аудиторией являются мелкие и средние компании, из-за ограниченности функционала;

PRTG – продуктовое решение, которое занимается анализом и сохранением статистики в базу данных, анализом входящих пакетов, просмотром карты сети в режиме реального времени, с возможностью просмотра предыдущих карт сети. В достоинства данной программы так же можно записать сбор технических параметров об устройствах, подключенных к сети, а также анализ уровня нагрузки на сетевое оборудование.

Благодаря удобному пользовательскому интерфейсу (UI) очень удобен в использовании и настройке. UI, так же, как и у Nagios основан на веб-интерфейсе.

Доступна бесплатная 30-ти дневная лицензия. Подписки выдаются на неограниченное количество времени, но на 1 копию. Самой дешевой является бессрочная лицензия, стоимостью 122.500 рублей, имеющая ограничение, до пятисот устройств сети.

Достоинства:

– обладает большим количеством функций, чем Nagios,

– удобный интерфейс настроек,

– гибкий настраиваемый мониторинг, которого нет в Nagios,

– умеет строить карты сети и хранить их в базе данных;

Недостатки:

– самая высокая цена на рынке,

– громоздкий интерфейс системы,

– не имеет собственной базы данных;

Kismet – последний обозреваемый продукт. Это open-source приложение, гораздо меньше, чем все предыдущие. Его особенностью является возможность всесторонне анализировать сетевой трафик и обнаруживать в нем аномалии[2].

Данное решение позволяет найти некорректно сконфигурированные или нелегальные точки доступа, которые злоумышленники могут использовать для перехвата трафика сети.

Для этих целей в приложении проработана возможность обнаружения различных типов сетевых атак – как на уровне сети, так и на уровне каналов связи. Существует система обнаружения сетевых атак с последующим уведомление системного администратора, для предотвращения угрозы.

Однако данным решением пользуются не только системные администраторы, но также и злоумышленники, которые подключили к беспроводной сети, и занимаются анализом трафика, составлением карты сети и т.д.

Достоинства:

– бесплатный open-source проект,

– минималистичный интерфейс, в отличии от PRTG;

Недостатки:

– сложна в использовании, по сравнению с Total Network Monitor 2,

– сканер сети в несколько раз медленнее, чем у PRTG;

Здесь приведены лишь некоторые программные решения. Существует еще несколько продуктов, которые не так популярны, многофункциональны или удобны.

2 Разработка объектной модели сети

2.1 Анализ возможных способов взаимодействия сетевых устройств

Современные сети огромны. Большая часть из них насчитывает сотни компьютеров и еще столько же сетевых устройств для налаживания работы, разграничения видимости и доступа.

Взаимодействие большинства происходит на самых низких уровнях модели OSI: физический, канальный, сетевой и транспортный. На этих уровнях происходит работа с пакетами, ip адресами, маркерами и т.д. Именно основываясь на этих данных существующие продуктовые решения собирают информацию, стоят карты сети, сканируют атаки.

К сожалению, решений с открытым кодом очень мало, так как каждый из них является самостоятельным коммерческим проектом, приносящим прибыль своим компаниям за разовую покупку, месячные подписки или постоянное обслуживание.

Для создания модели сети были выделены несколько высокоуровневых (относительно вышеперечисленных алгоритмов) протоколов: HTTP, HTTPS, REST и SOAP[6].

Далее приведен краткий обзор протоколов:

SOAP (Simple Object Access Protocol) – это протокол обмена сообщениями, состоящих из объектов в распределённой вычислительной среде. Протокол SOAP был создан в 1998 году программистом, работавшим в Microsoft Дейвом Винером (Dave Winer), после чего был передан в консорциум W3C. Последняя версия стандарта на сегодняшний день - SOAP 1.2.

На рисунке 3 представлена схематическая структура сообщения SOAP, которая содержит следующие теги:

– Envelope – корневой элемент, который определяет сообщение и пространство имен, использованное в документе,

– Header – необязательный элемент заголовка, может содержать информацию о сетевой маршрутизации, безопасности и т.д.,

– Body – элемент тела сообщения, содержит набор объектов, которым обмениваются приложения,

– Fault – необязательный элемент в структуре, который содержит информацию об ошибках;

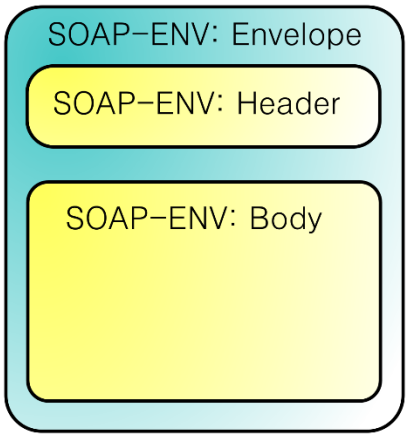

Рисунок 3 – Схематическая структура SOAP

До 2007 года SOAP расшифровывался как Simple Object Access Protocol — простой протокол доступа к объектам. Это название было отражением его первоначального замысла — обращаться к методам удаленных объектов. Однако после 2007 года вышла версия SOAP 1.2 и данную аббревиатуру было решено никак не расшифровывать, потому что протокол SOAP не умеет различать вызов процедуры и ответ на него. Протокол только определяет формат послания (message) в виде документа XML[6].

Послание может содержать совершенно любую информацию: вызов какой-либо процедуры у объекта, ответ на такой вызов, запрос на выполнение каких-то других действий или просто текст. Спецификацию SOAP не может разобрать содержимое послания, она задает только его оформление. На рисунке 4 представлено простейшее сообщение SOAP. Оно состоит из тега Envelope, в котором определены пространства имен xsi, xsd и soap, следом за ним идет обязательный тег Body, в котором хранится

структура передаваемого объекта, состоящего из одного параметра – productID.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>12345</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

Рисунок 4 – Простейшее сообщение SOAP

SOAP основан на языке разметки XML и расширяет некоторый протокол прикладного уровня — HTTP, FTP, SMTP и т.д. В большинстве случаев используется HTTP. Заменяя протокол HTTP для запроса web-страницы, которая будет отображена в браузере, SOAP отправляет XML-сообщение и получает результат в HTTP-отклике. Для корректной работы с XML-сообщениями процесс-слушатель должен реализовать публичный SOAP-процессор, который будет обрабатывать входящие сообщения, с разметкой XML[10].

Следующим возможным взаимодействием между приложениями был выбран REST (от англ. Representational State Transfer — «передача состояния представления»). Данная аббревиатура означает архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST является согласованным набором ограничений, учитываемым при проектировании распределённой системы или структурно связанных микро-сервисах. В некоторых случаях, таких как поисковые системы, интернет-магазины и прочие системы, которые оперируют данными, это приводит к повышению производительности и упрощению архитектуры.

REST архитектура подразумевает под собой простые правила, которые предоставляют простой CRUD интерфейс для внешних приложений, работающих на протоколах HTTP и HTTPS.

На следующем рисунке 5 приведены соответствие интерфейса CRUD и REST архитектуры. Данное соответствие является общепринятой нормой, но не обязательным условием. При необходимости, можно использовать только POST запросы, однако тогда система станет менее гибкой и для работы с ней внешним системам придется изменить свой подход[17].

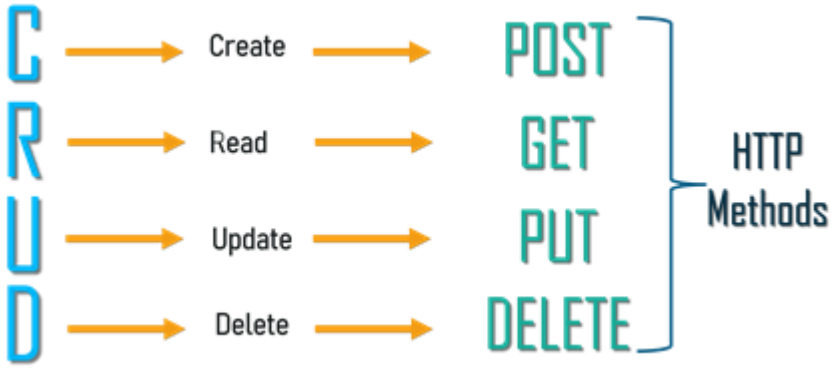

Рисунок 5 – Соответствие CRUD интерфейса и REST архитектуры

В интернет сети вызов удаленных процедур может представлять собой обычный HTTP или HTTPS запрос, обычно такими запросами являются GET или POST. Такие запросы называют «REST-запрос». Все необходимые данные передаются либо в качестве параметров запроса, как в случае с GET, либо в теле запроса, в случае POST.

Главные принципы написания REST-сервисов:

– Client-Server. Необходимо отделять пользовательские интерфейсы и хранилища данных;

– Stateless (без состояния). Вся информация о сессии клиента с сервером должна храниться у клиента;

– Cachebility (кэшируемость). Если данные в запросе пользователя могут быть закэшированные, тогда, при следующем обращении пользователя, данные должны достаться из кэша, а не из базы данных;

– Uniform interface (единообразие интерфейса). Использование общепринятых практик по представлению публичных интерфейсов;

Веб-сервисы, которые не нарушающих накладываемых ограничений архитектурой REST, называют RESTful сервисами[19].

В отличие от веб-сервисов на основе SOAP, не существует общепринятого стандарта для RESTful публичного интерфейса. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют такие стандарты, как HTTP, URL, JSON и XML.

После данного сравнения было решено выбрать именно REST-архитектуру, как более удобную, гибкую и популярную.

К выбранной архитектуре необходимо выбрать разработать публичный интерфейс, который будет читать, создавать, изменить и удалять данные. Согласно REST архитектуре, чтение происходит путем GET-запроса, а посылка данных на создание POST-запросом. Для работы с POST-запросами необходимо передавать всю информацию в теле запроса. Встает вопрос про формат сообщения, в котором будет происходить передача будущих объектов сети. Проанализировав возможные форматы передачи сообщений был выбран формат JSON.

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. За счёт своей лаконичности и краткости по сравнению с XML форматом, JSON признан более подходящим для сериализации сложных структур. На рисунке 6 представлено наглядное сравнение одной и той же структуры данных в двух разных форматах.

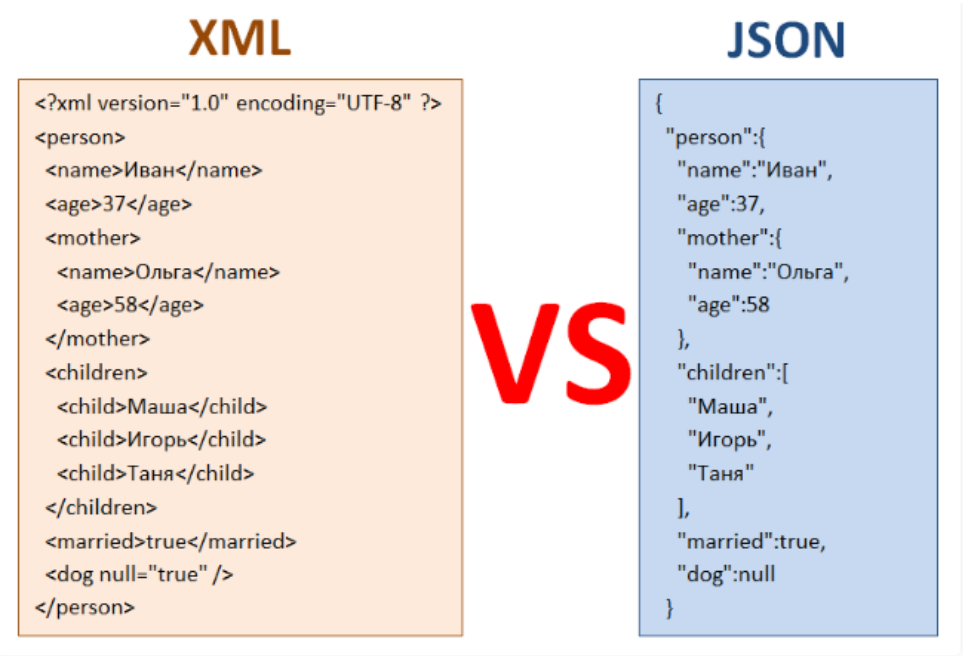

Рисунок 6 – Сравнение форматов XML и JSON

Из картинки наглядно видно, насколько JSON является более экономичным, в плане количества символов, форматом, по сравнению с XML. Именно из-за этого было решено отказаться от SOUP протокола передачи данных, дабы не смешивать форматы JSON и XML, и взять архитектуру REST протокола HTTP[11].

Данный формат применяется в веб-сервисах для обмена данными между клиентом и сервером или между серверами (программные HTTP-сопряжения). Во многих языках программирования имеются библиотеки для автоматического преобразования структуры JSON в объект базы данных или в объект программы.

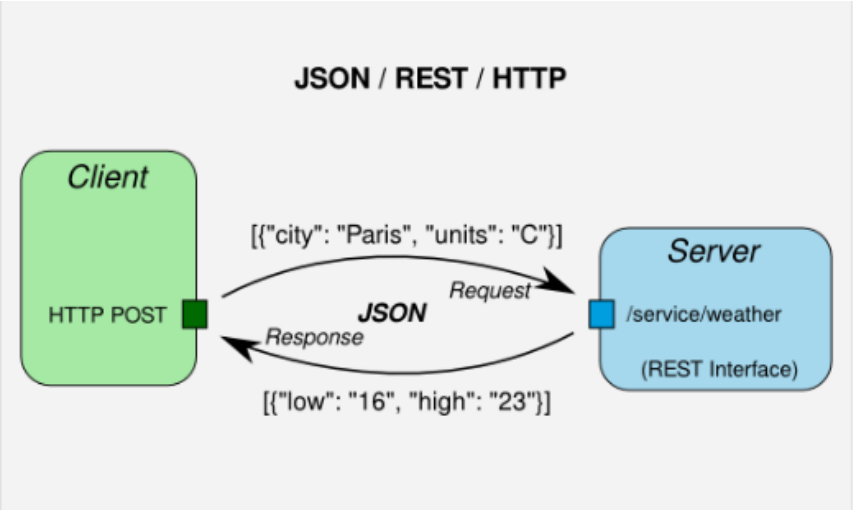

Рисунок 7 – Структура взаимодействия клиента и сервера

На рисунке 7 представлена общая схема работы системы, основанной на REST архитектуре и HTTP протоколе, в теле запроса которого структура из JSON объектов. Клиент шлет на сервер HTTP POST запрос, передавая в теле запроса параметры с именем города и секцией, в ответ сервер, после обработки запроса, шлет численную структуру.

2.2 Выбор сетевых устройств, для мониторинга и выделение их характерных признаков

Из раздела 1.1.1 были выделены два основных устройства, на базе которых и будет строиться модель сети:

1) маршрутизатор со стороны узла клиента (CPE),
2) граничный маршрутизатор со стороны оператора (PE);

Данные устройства были выбраны по тому, что с их помощью можно построить простейшую сеть обычного жилого здания, где CPE будут выступать маршрутизаторами обычных жителей, а PE – устройствами провайдера, который предоставляет услуги в здание. Граничных маршрутизаторов со стороны оператора может быть несколько на одно здание, так как количество провайдеров, предоставляющих услуги в здании

часто превышает единицу, или здание может быть многоподъездным домом, в котором один и тот же провайдер выставляет несколько своих устройств[7].

Следующая задача, которая возникла в ходе построения модели: выделение главных характеристик для устройств сети. Выделенные характеристики пойдут в основу объектов программного кода, структур таблиц реляционных баз данных и передаваемых текстовых данных в формате JSON.

В объекте CPE выделены следующие характеристики:

1) координаты (*X, Y*),
2) имя CPE (IP),
3) тип,
4) скорость интернета,
5) состояние интернета на устройстве (ON/OFF);

Остальные характеристики были отброшены для упрощения абстрактной модели устройства.

В объекте PE выделены следующие характеристики:

1) координаты (*X, Y*),
2) имя PE (IP),
3) тип,
4) температура,
5) состояние охлаждения устройства (ON/OFF);

Остальные характеристики были отброшены для упрощения абстрактной модели устройства.

Для данных моделей необходимо сформировать ряд правил, которые будут определять их поведение:

– по умолчанию, скорость CPE равна максимально возможной;

– если CPE имеет выключенный интернет, то его скорость приема становится равной 0;

– если на PE температура поднимается выше 70 градусов, скорость отдачи падает в 2 раза, и при увеличении на каждые 10 градусов, падает в 2 раза от текущей;

– если PE оборудование имеет выключенное охлаждение, то температура растет на 1 градус каждую секунду, и наоборот;

– с одним PE может быть связано несколько CPE, но суммарная максимальная скорость CPE не должна превышать максимальную скорость PE (например, если максимальная скорость отдачи PE в текущий момент времени равна 10000, а скорость приема CPE 1000, то к одному устройству PE можно одновременно подключить 10 CPE, без потери скорости);

– если к PE подключают CPE и включают интернет, то скорость отдачи PE увеличивается (если она уже не на максимуме) и наоборот;

– если скорость отдачи PE падает от перегрева, то скорость приема связанных CPE должна пропорционально падать.

Данные модели являются сильно упрощенными моделями реальных устройств. Для реализации этих двух видов оборудования было решено программно реализовать два микро-сервиса, каждый из которых будет заниматься эмуляцией реального оборудования, по описанным выше моделям. Все взаимодействия между устройствами будут посредством REST запросов, описанных в пункте 2.1[13].

Для отправки своих данных системе мониторинга, было принято решение, что каждый из микро-сервисов обновляет скорость и температуру и публикует информацию о своем состоянии в формате JSON в топик Apache Kafka, на которую подписан DataFlow Apache NiFi, агрегирующий данные в единый топик, приводя их в общий формат и публикует их в собственный топик Kafka, который в свою очередь будет принимать система мониторинга.

Подробнее схема работы микро-сервисов будет описана в пункте 3.1.

2.3 Алгоритмы искусственного интеллекта

Термин искусственного интеллекта (ИИ) используется для обозначения большого направления научных и прикладных исследований. Искусственный интеллект используется там, где условия и циклы не в состоянии решить задачу.

В системе мониторинга и управления оборудованием сети искусственный интеллект играет роль автоматизации. В данной сфере огромный выбор работ, которые можно и нужно автоматизировать, чтобы исключить человеческие ошибки, снизить объем информации, которую человек должен получать из системы и т.д.[10].

В данном подразделе будет рассмотрено два алгоритма:

– алгоритм k-NN, который необходим для автоматизации распределения нагрузки между оборудованием провайдера,

– алгоритм k-means, который облегчит добавление новых устройств в систему, автоматически распределяя их по категориям, таким как: принадлежность к одному району или распределения оборудования провайдера по степени загруженности.

2.3.1 Алгоритм k-NN

Метод ближайших соседей (англ. k-nearest neighbors algorithm, k-NN) — метрический алгоритм для автоматической классификации объектов.

Алгоритм k-NN необходим в системе мониторинга, для того, чтобы система самостоятельно идентифицировала новые элементы сети по различным классам[11].

Если алгоритм используется для классификации, объект присваивается той группе объектов, которые являются наиболее распространёнными среди K близлежащих соседей данного элемента, классы которых уже известны. В случае, если алгоритм используется для регрессии, то объекту будет присвоено среднее значение по K ближайшим к нему объектам, значения которых уже известны.

Различные параметры будут иметь различную размерность и диапазон значений, например, параметр *P1* будет представлен в диапазоне от 10 до 25, а параметр *P2* представлен в диапазоне от 100 до 1000. В этом случае значение расстояния до объекта с известным классом сильно искажается, из-за большой разницы диапазонов значений. Для устранения этого недостатка все значения параметров требуется нормализовать. Существует 2 способа нормализации данных:

– минимакс-нормализация,

– Z-нормализация;

Минимакс-нормализация производится следующим образом:

$$x' = \frac{(x - min|X|)}{(max|X| - min|X|)}, \quad (1)$$

где x' – нормированное значение;

x – значение параметра до нормирования;

$min|X|$, $max|X|$ – минимальное и максимальное значение, среди всех значений параметра X.

При использовании данной нормализации все значения будут лежать в пределах от 0 до 1.

Z-нормализация производится следующим образом:

$$x' = \frac{(x - M|X|)}{\sigma|X|}, \quad (2)$$

где σ — среднеквадратичное отклонение.

При использовании данной нормализации значения будут располагаться в диапазоне $(-3 \cdot \sigma;\ 3 \cdot \sigma)$.

Встречаются ситуации, когда некоторые параметры могут быть важнее остальных, для данного алгоритма, поэтому для каждого параметра может быть задан в соответствие определённый вес. Чем больше выбран вес, по отношению к другим веса, там более значимым становится параметр. Таким образом, каждому параметру k будет задан в соответствие вес Z_k так, что значение параметра будет попадать в диапазон $\lceil 0;\ Z_k \cdot \max(k) \rceil$ (данный

диапазон характерен только для способа нормализации минимакс-нормализация). К примеру, если параметру присвоить вес равный 3,5, то его нормализованное значение будет лежать в диапазоне ⌈0; 3,5⌉.

Далее будет рассмотрен пример работы системы мониторинга с алгоритмом k-NN, который используется для идентификации объектов, добавленные в сеть, по отношению к объектам, которые уже имеют определенные классы.

Пусть имеется *m* объектов, каждому из которых соответствует запись в реляционной базе данных системы мониторинга. Все записи, на начало работы программы, принадлежат какому-либо классу. Необходимо определить класс для новой записи[17].

Алгоритм определения класса для объекта:

На первом шаге алгоритма должно быть задано число ближайших соседей *k*. Если взять *k* равным слишком малому числу, то смысл алгоритма теряется, он будет находить ближайших соседей и принимать их класс, даже не подозревая о существовании остальных. Отсюда можно сделать вывод, что чем больший *k* взять, тем лучшим будет результат. Однако у всего есть предел, если количество объектов в сети будет огромным, то для определения класса нашего объекта уйдет слишком много времени, и, было бы проще проставить класс самостоятельно, при добавлении объекта. К тому же нет необходимости оценивать значения параметров слишком удаленных объектов, так как к их классу наш объект точно не будет принадлежать. В нашей системе примем значение *k=10,* так как модель системы не будет слишком большой.

Например, если для объекта B, ближайшем соседом является A, то это не означает, что B – ближайший сосед *A*. Данный пример представлен на рисунке 8. При *k=1* ближайшей для точки B будет точка A, а для A – X. При увеличении коэффициента до *k=7,* точка B так же не будет входить в число соседей.

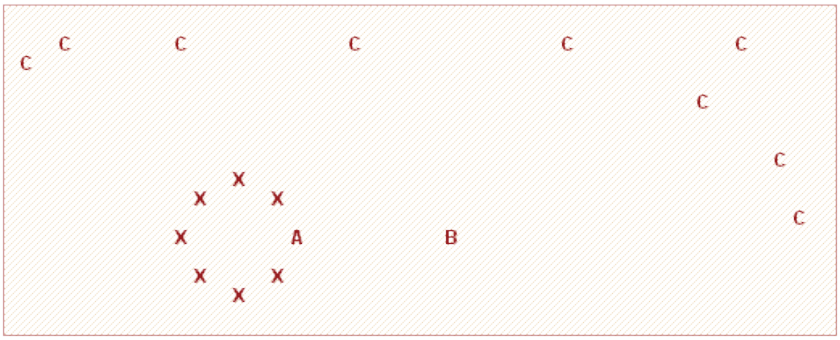

Рисунок 8 – Ближайшие соседи A и B

На втором шаге необходимо найти k записей с минимальным расстоянием до вектора признаков нового объекта. Функция для расчета расстояния должна отвечать следующим правилам:

$$d(x, y) \geq 0, d(x, y) = 0, \quad (3)$$

тогда и только тогда, когда $x = y$.

$$d(x, y) = d(y, x), \quad (4)$$

$$d(x, z) \leq d(x, y) + d(y, z). \quad (5)$$

где x, y, z - векторы признаков сравниваемых объектов и не лежат на одной прямой.

Для упорядоченных значений атрибутов находится Евклидово расстояние:

$$D_E = \sqrt{\sum_i^n (x_i - y_i)^2}, \quad (6)$$

где n – количество атрибутов.

На следующем шаге, когда были найдены объекты, наиболее похожие на новый объект, необходимо решить, как они повлияют на класс устройства. Для этого используется функция сочетания (combination function). Существуют несколько способов определить класс объекта, воспользуемся вариантом под названием «взвешенное голосование» (weighted voting)[16].

Взвешенное голосование.

При использовании метода взвешенного состояния учитывается расстояние до нового объекта. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по следующей формуле:

$$votes(class) = \sum_{i=1}^{n} \frac{1}{d^2(X,Y_i)}, \quad (7)$$

где $d^2(X, Y_i)$ – квадрат расстояния от известной записи Y_i до новой X;

n – количество известных записей класса, для которого рассчитываются голоса;

class - наименование класса.

Новой записи ставится класс, набравший наибольшее количество голосов.

Рассмотрим пример, для системы мониторинга и контроля оборудования сети.

Пусть в сети на момент подключения имеются 8 объектов с определенными классами и двумя параметрами. На рисунке 9 представлено графическое распределение на координатной плоскости существующих устройств и нового устройства:

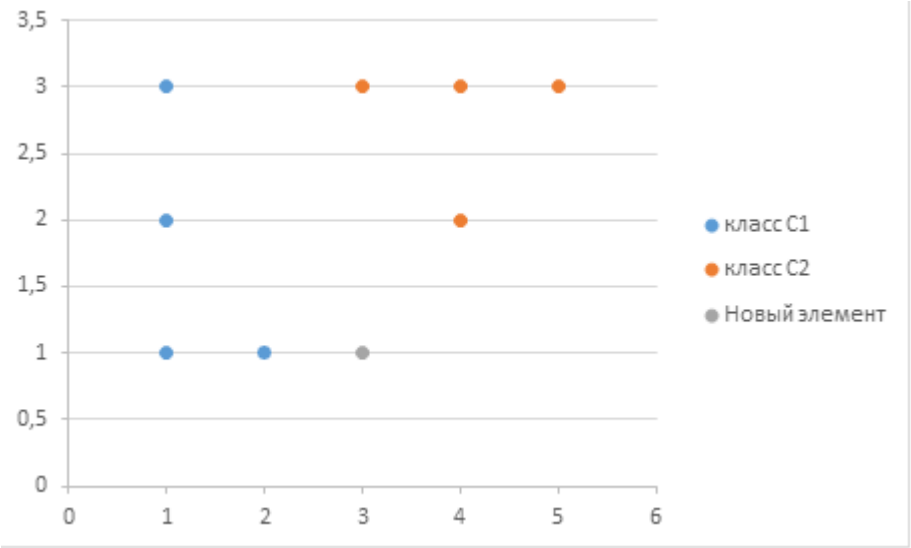

Рисунок 9 – Существующие объекты в системе

Необходимо определить в какому классу оно будет относиться новое устройство, на основании имеющихся распределений объектов[11].

Первым делом необходимо рассчитать, по формуле Евклида, расстояние от каждого имеющегося объекта до нового и расположить их в

порядке возрастания, после чего рассчитать методом взвешенного голосования результирующий класс объекта

Таблица 1 – Расчет результирующего класса

k	Объект	Оценка близости с классом C1	Оценка близости с классом C2	Результат
1	H	$\frac{1}{1^2}$	---	C1
2	F	$\frac{1}{1^2}$	$\frac{1}{1{,}41^2}$	C1
3	B	$\frac{1}{1^2}$	$\frac{1}{1{,}41^2}+\frac{1}{2^2}=0{,}75$	C1
4	G	$\frac{1}{1^2}+\frac{1}{2^2}=1{,}25$	$\frac{1}{1{,}41^2}+\frac{1}{2^2}=0{,}75$	C1
5	C	$\frac{1}{1^2}+\frac{1}{2^2}=1{,}25$	$\frac{1}{1{,}41^2}+\frac{1}{2^2}+\frac{1}{2{,}23^2}=0{,}95$	C1
6	E	$1{,}25+\frac{1}{2{,}23^2}=1{,}45$	$\frac{1}{1{,}41^2}+\frac{1}{2^2}+\frac{1}{2{,}23^2}=0{,}95$	C1
7	A	$1{,}45+\frac{1}{2{,}82^2}=1{,}57$	$\frac{1}{1{,}41^2}+\frac{1}{2^2}+\frac{1}{2{,}23^2}=0{,}95$	C1
8	D	$1{,}45+\frac{1}{2{,}82^2}=1{,}57$	$0{,}95+\frac{1}{2{,}82^2}=1{,}07$	C1

Из таблицы видно, что результирующий класс – класс C1.

Данный пример имел абстрактную форму. В ходе реализации системы мониторинга и контроля оборудованием сети, алгоритм k-NN будет использоваться для автоматического переключения устройств пользователей (CPE) на менее нагруженные устройства провайдера (PE).

Новое оборудование будет выбираться по очереди из имеющихся в системе. Классом, которое необходимо определить станет устройство PE, к которому будет подключен CPE. Основными характеристиками для расчета расстояния будут выбраны: координаты (*X*, *Y*), максимально возможная скорость отдачи на маршрутизаторе провайдера[18].

2.3.2 Алгоритм k-means

Метод k-средних (англ. k-means) — наиболее популярный метод кластеризации. Был изобретён в 1950-х годах математиком Гуго Штейнгаузом и почти одновременно Стюартом Ллойдом.

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров, по формуле 8.

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2, \quad (8)$$

где k – число кластеров;

S_i – полученные кластеры;

μ_i – центры масс всех векторов *x* из кластера S_i.

По аналогии с методом главных компонент центры кластеров называются также главными точками, а сам метод называется методом главных точек и включается в общую теорию главных объектов, обеспечивающих наилучшую аппроксимацию данных.

Для расчета расстояния между объектами используется формула (6) Евклидово расстояние.

Алгоритм выполняет работу и разбивает объекты на кластеры до тех пор, пока центры не перестанут двигаться, то есть в соседних итерациях координаты центров останутся неизменными.

Алгоритм k-means может использоваться в огромном множестве областей, так как он не привязан к сути данных, а занимается анализом только отдельных параметров. Однако он имеет ряд недостатков:

– кластеризация некластеризованных данных – даже там, где данные однородны, алгоритм занимается разбиением,

– даже на совершенных наборах данных алгоритм может вычисляться до бесконечности или неверно разбить данные;

Изначально планировалось использовать k-means для кластеризации объектов по нескольким критериям, таким, как: координаты, что позволило бы делить объекты на дома, районы и т.д. без участия системного администратора; нагруженность объектов, что делило бы устройства провайдера на высоко, средне и низко нагруженные[17].

Однако данная система не оправдала себя, намного более логично было бы воспользоваться более простой логикой, такой как: поиск среднего

значения среди нагрузки устройств с допустимыми интервалами, который бы сортировал объект по категориям. Для разделения по районам так же подходит логика сортировки относительно координат других объектов. В связи с этим, и, учитывая все описанные выше недостатки алгоритма, от него было решено отказаться.

Искусственный интеллект уже успел укрепиться во многих сферах науки. Его алгоритмы используются повсеместно, начиная от анализа возможных сочетаний различных препаратов, с оценкой потенциальной эффективности, для разработки вакцины от болезни, и заканчивая распознаванием лиц через уличные камеры.

В случае со сферой систем мониторинга искусственный интеллект позволяет автоматизировать некоторые действия, делая услуги провайдера сети более приятными для пользователя.

2.3.3 Алгоритм линейной регрессии

Линейная регрессия — метод восстановления зависимости между двумя переменными. Ниже приведен пример программы, которая строит линейную модель зависимости по заданной выборке и показывает результат на графике.

Для заданного множества из m пар (x, y), $i=1,..,m$, значений свободной и зависимой переменной требуется построить зависимость. Назначена линейная модель:

$$y_i = f(\omega, x_i) + \epsilon_i, \quad (9)$$

где ϵ_i – адаптивная случайная величина;

Переменные x, y принимают значения на числовой прямой R. Предполагается, что случайная величина распределена нормально с нулевым математическим ожиданием и фиксированной дисперсией σ_ϵ^2, которая не зависит от переменных x, y. При таких предположениях параметры ω регрессионной модели вычисляются с помощью метода наименьших квадратов.

Например, требуется построить зависимость скорости интернета от времени. Для этого необходимо выстроить таблицу, где будут храниться пары значений: время и скорость интернета. Для корректности данных – их необходимо нормировать[13].

Одномерная регрессия

Определим модель зависимости как:

$$y_i = \omega_1 + \omega_2 x_i + \epsilon_i \quad (10)$$

Согласно методу наименьших квадратов, искомый вектор параметров вычисляется по формуле:

$$\omega = (\omega_1, \omega_2)^T \quad (11)$$

Столбцы матрицы A есть подстановки значений свободной переменной $x_i^0 \to a_{i1}$ и $x_i^1 \to a_{i2}, i = 1, .., m$. Матрица имеет вид

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_m \end{pmatrix} \quad (12)$$

Зависимая переменная восстанавливается по полученным весам и заданным значениям свободной переменной

$$y_i{}^* = \omega_1 + \omega_2 x_i \quad (13)$$

Для оценки качества модели используется критерий суммы квадратов регрессионных остатков, SSE — Sum of Squared Errors.

$$SSE = \sum_{i=1}^{m} (y_i - y_i{}^*)^2 = (y - y^*)^T (y - y^*) \quad (14)$$

Алгоритм линейной регрессии используется для прогнозирования скорости интернета для CPE в зависимости от времени в стабильной модели сети. Однако, если хаотично добавлять или удалять устройства из сети – прогнозы становятся слишком далеки от реальной скорости.

3 Разработка системы мониторинга и анализ результатов

В данной главе происходит описание общей схемы взаимодействия системы мониторинга и управление оборудованием сети с двумя микро-сервисами, которые выполняют роль эмуляции реального оборудования (RE и CPE). Так же приводится программный код систем с описанием классов и методов, описанием пользовательского интерфейса.

3.1 Описание общей схемы взаимодействия системы

В предыдущих пунктах были разработаны модели, отвечающие за устройства пользователей и устройства провайдера, предоставляющего услуги. Данные модели будут реализованы в отдельных микро-сервисах[4].

Важнейшей характеристикой любой системы является ее быстродействие. Потенциально система мониторинга может иметь огромное множество устройств, которые будут слать тысячи запросов в единицу времени на один единственный мониторинг сервер. Для устранения этой проблемы, есть несколько решений:

– использовать несколько серверов, которые будут делить нагрузку между собой,

– использовать очереди на сервере, с помощью которых будут обрабатываться запросы,

– использовать брокеры сообщений;

У первого решения есть явный минус, стоимость. И даже в этом случае слишком большой поток запросов может привести к зависанию сервера и последующему отказу.

Использование очередей является только временным решением, так как количество запросов на сервер не сократится, а будет либо копиться в ожидании, либо отбрасываться по ограничению времени ожидания.

Apache Kafka — распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache. Написан на языке программирования Scala и Java.

Согласно исследованиям, проведенным программистами-разработчиками социальной сети «LinkedIn» – максимальная скорость передачи информации, с использованием продукта компании Apache – миллион сообщений в секунду (данная скорость была достигнута на современных серверах, настройкой которых занимались высококлассные программисты, а также была сконфигурирована специальная архитектура под данные замеры). Поэтому выбор был сделан в пользу Apache Kafka.

Данный продукт работает с потоками данных, которые помещаются в специальные топики (отдельно созданные темы, в которые складываются и из которых читаются сообщения) издателем и читаются подписчиком этого топика[6].

Данная схема очень удобна, так как она нивелирует лишние затраты времени на создание и какой-либо лишней информации, которую бы пришлось создавать при работе с HTTP REST или SOUP.

Для работы с данным продуктом необходимо создать среду и отдельные топики на сервере, на котором будет производиться работа.

```
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
.\bin\windows\kafka-server-start.bat .\config\server.properties

.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181
--replication-factor 1 --partitions 1 --topic javainuse-topic

.\bin\windows\kafka-console-produser.bat --broker-list localhost:9092
--topic javainuse-topic

.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092
--topic javainuse-topic --form-beginning
```

Рисунок 10 – Команды для запуска Kafka

На рисунке 10 представлены команды для запуска брокера сообщений на локальной машине. На первой и второй строчках запускаются серверы, в

среде которых и существуют топики. На последующих строках создается сам топик, издатель и подписчик.

Каждый микро-сервис будет издателем своего топика, отсылая в него сообщениями информацию о своём состоянии. Однако системе мониторинга в этом случае придется быть подписчиком сразу на две темы, что является крайне неудобным фактом, в случае расширения общей модели сети, добавления всё больших устройств в модель.

Для решения этой проблемы необходимо агрегирование данных из потоков Apache Kafka, приведения их к общему формату и помещению их в выходной поток Kafka, на который и будет подписчиком мониторинг сервис. Вся эта работы должна быть выполнена за пределами сервиса, так как не имеет ничего общего с его функционалом[7].

Со всеми задачами блестяще справился компонент от компании Apache – NiFi. Данный продукт имеет пользовательский интерфейс, который позволяет собирать логику работы с потоками данных из блоков, которые уже имеются в стандартном наборе, а также писать свои на любом широко-известном языке программирования.

Для выбора блоков из уже созданных имеется удобный поиск по названию или тегу блока, показанному на рисунке 11.

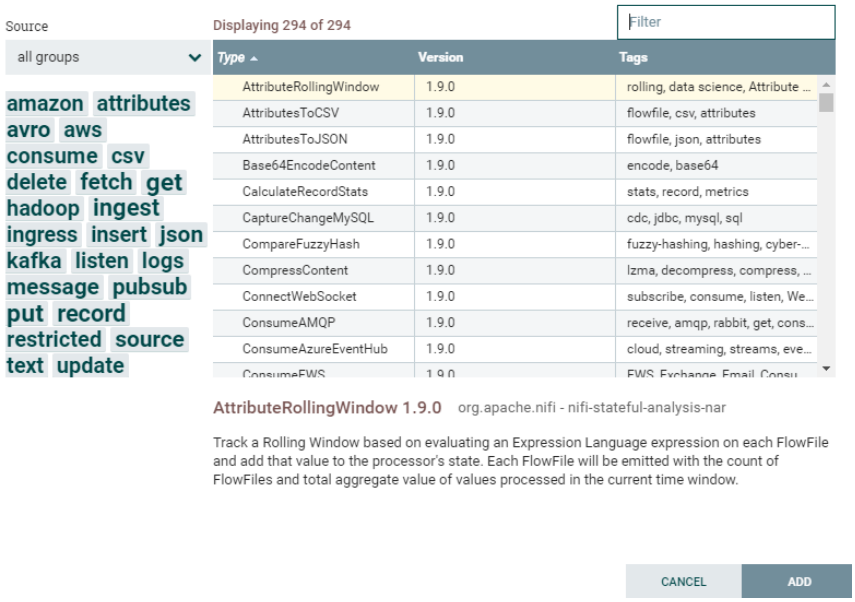

Рисунок 11 – Поиск и добавление блоков NiFi

Используя несколько стандартных блоков, которые принимают данные из входящих потоков Kafka и помещают данные в выходящий поток Kafka был написан агрегирующий блок, который соединяет потоки с принимающих блоков, проводит над ними работу по приведению к общему формату и направляет на выходной блок.

Выходной блок в свою очередь помещает собранные данные в поток, который принимает на вход система мониторинга и отображает в пользовательском интерфейсе[11].

На рисунке 12 показана схема NiFi. Верхние два блока являются подписчиками на топики микро-сервисов PE и CPE. Последующий блок является агрегирующим блоком, который соединяет сообщения. Следующий блок является издателем в новом топике Apache Kafka, слушателем которого является система мониторинга и контроля оборудования сети. В самый нижний блок помещаются все сообщения которые были переданы в выходящем топике.

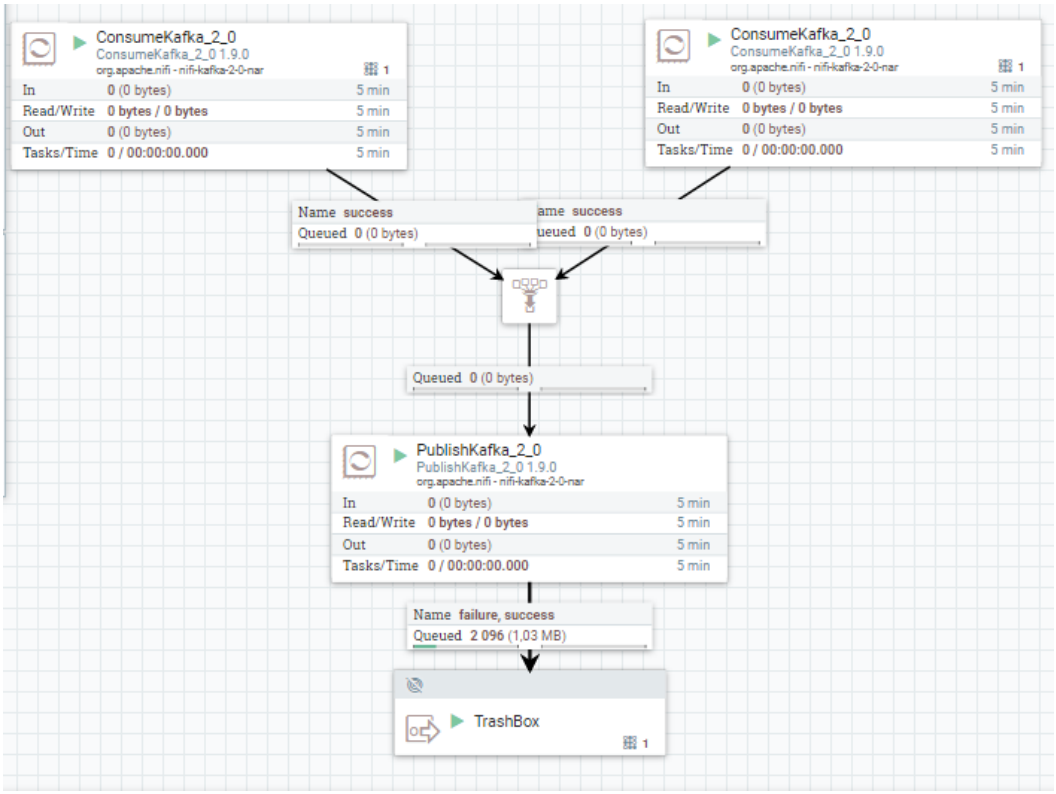

Рисунок 12 – Схема блоков NiFi

Каждый блок из данной схемы настраивается индивидуально, и имеет большое количество свойств, начиная от имени и цвета блока, заканчивая выходными/выходными данными, шифрованием потоков, адресами и именами топиков Kafka. На рисунке 13 представлена настройка выходящего блока, подписчиком которого является система мониторинга.

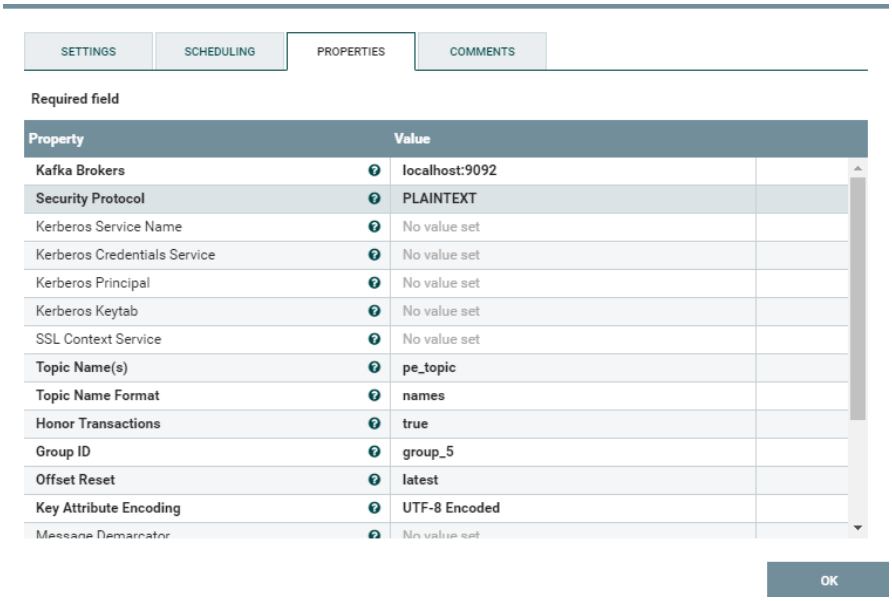

Рисунок 13 – Настройки блока в NiFi

Написание блоков осуществляется на любом языке программирования высокого уровня. Для этого был скачан шаблон блока и заполнен логикой агрегирования данных, после чего был собран с помощью Maven, упакован в JAR и залит на сервер NiFi[10].

Общая схема взаимодействия всей системы представлена на рисунке 14.

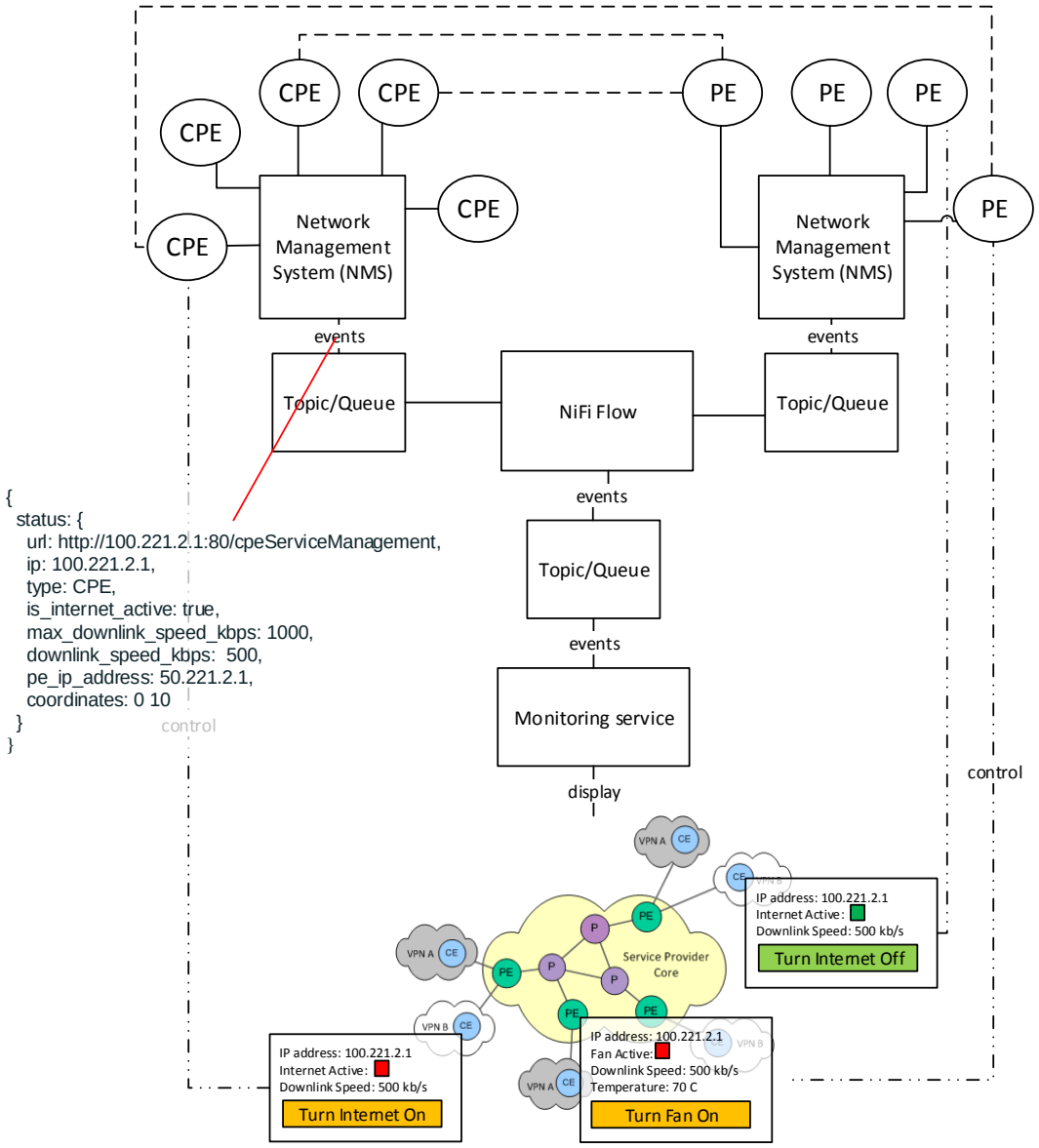

Рисунок 14 – Общая схема взаимодействия микро-сервисов и системы мониторинга и контроля управления сети

На данной схеме представлено схематическое представление взаимодействия микро-сервисов между собой и с топиками Apache Kafka. Штрихпунктирные линии означают REST запросы, которые исходят от системы мониторинга для управления оборудованием.

3.2 Описание модулей программы

В данном разделе будет рассмотрена структура двух микро-сервисов PE и CPE, а также системы мониторинга и управления оборудованием сети. Каждый модуль выполняет свою работу и запускается отдельно, однако самостоятельное существование их невозможно. Маршрутизаторам пользователя необходимы краевые маршрутизаторы провайдера, к которым они подключаются. Маршрутизаторам провайдера в свою очередь необходимы пользовательские устройства для отдачи интернет трафика. Системе мониторинга необходимы все эти устройства для их контроля и отображения[13].

Все отдельные модули написаны на языке Java с использованием фреймворка Spring Boot в среде разработки IntelliJ IDEA.

Первым устройством, которое будет разобрано, является CPE.

На рисунке 15 представлена структура пакетов с классами, интерфейсами и файлом настроек.

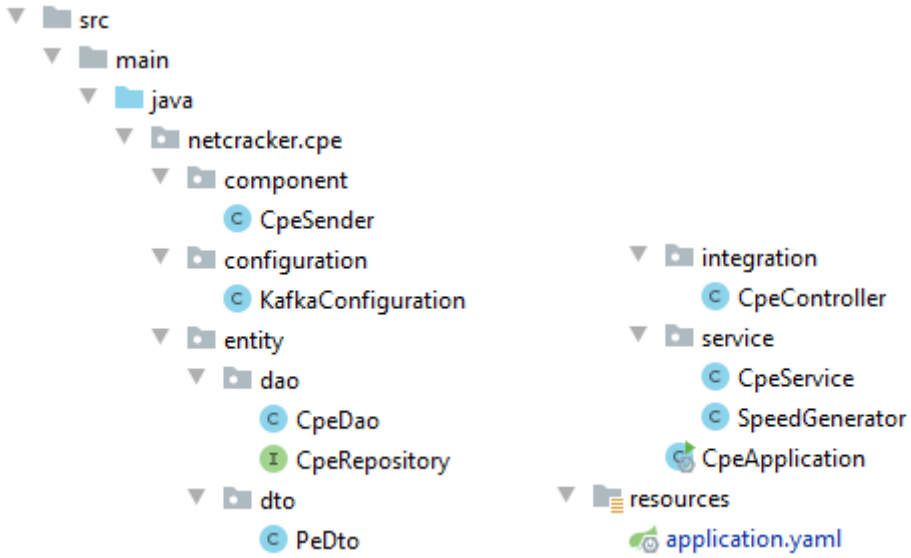

Рисунок 15 – Структура пакетов микро-сервиса CPE

Весь сервис разбит на пакеты, каждый из которых несет свою функциональность, отраженную в названии. В пакете java.netcracker.cpe

находятся классы, реализующие функциональность, описанную в модели в разделе 2.2.

Для подробного разбора всех классов, на рисунке 16 представлена UML-диаграмма классов.

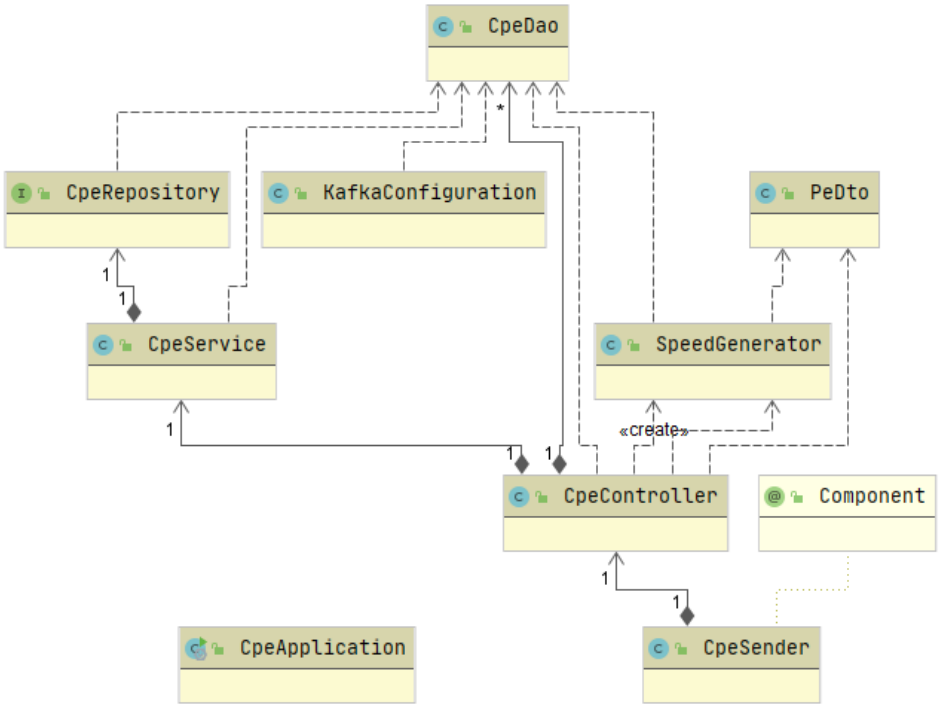

Рисунок 16 – UML-диаграмма классов CPE

Пунктирные линии на диаграмме отражают работу с данным классом, то есть он не инициализирует его напрямую, используя оператор new(), а работает с уже имеющимися объектами, переданными в методах. Сплошные линии обозначают направленные связи между классами, где класс, от которого идет связь инициализирует класс, к которому направлена связь, цифры рядом обозначают количественное отношение (один к одному, один ко многим).

Далее рассмотрим функционал каждого класса:

– CpeApplication – точка входа в программу. Содержит метод main, инициализируя приложение Spring;

– CpeDao – (Data Access Object) – класс сущности CPE, который состоит из атрибутов, описанных в модели и конструктора;

– PeDto – (Data Transfer Object) – класс сущности PE, предназначенный для маппинга входящей сущности;

– CpeRepository – интерфейс, который наследуется от интерфейса JpaRepository, предназначен для работы с базой данных;

– SpeedGenerator – класс, который генерирует скорость для всех объектов CPE, в зависимости от возможной скорости отдачи устройства PE, с которым они связаны;

– CpeService – класс, который формирует SQL запросы для работы с базой данный PostgreSQL, используя интерфейс CpeRepository;

– KafkaConfiguration – класс, отвечающий за работу с топиком Apache Kafka, который использует настройки для инициализации микро-сервиса как поставщика сообщений в свой собственный топик;

– CpeController – REST контроллер, который занимается приемом входящих запросов управления от системы мониторинга, и запросов информирования от микро-сервиса PE.

Весь код по данному микро-сервису находится в приложении А.

Так же данный сервис содержит файл под названием application.yaml, содержащий свойства, необходимы для работы программы. К этим свойствам относится: порт, на котором будет доступен сервер, URL базы данных, имя пользователя, пароль, драйвер для работы с базой данных, вид работы с базой данных на этапе запуска приложения (создание таблиц, создание таблиц в начале и удаление таблиц по окончанию работы, изменение записей в таблицах).

Второй микро-сервис схож по структуре и функционалу с сервисом CPE, поэтому рассмотрены будут только ключевые различия этих двух модулей.

На рисунке 17 представлена UML-диаграмма классов микро-сервиса PE.

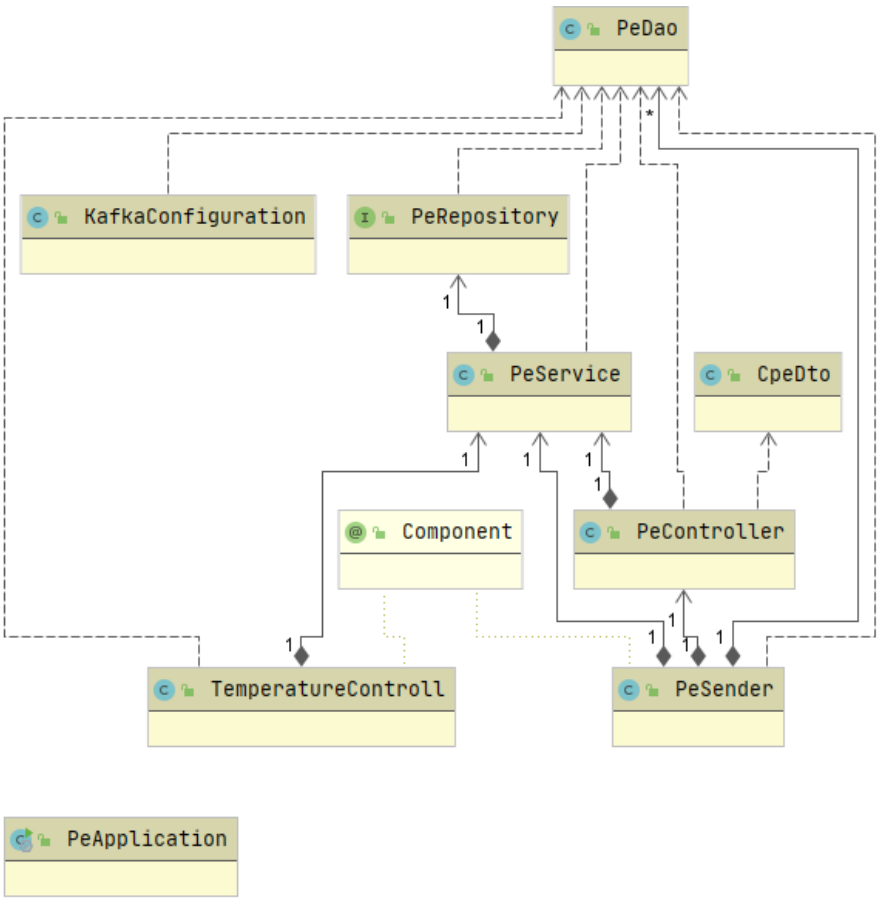

Рисунок 17 – UML-диаграмма классов PE

Отличительной особенностью данного модуля является класс TemperatureControll, который содержит методы, повышающие и понижающие температуру на оборудовании, в зависимости от включенности охлаждения. Остальные классы выполняют схожий функционал, за исключением особенностей связанных с объектами PeDao.

В файле свойств отличием является только порт, на котором поднимается приложение, имя пользователя в базе данных и пароль.

Весь код приложения находится в приложении Б.

Следующей частью программной разработки является его самая значимая часть – система мониторинга и контроля сетевого оборудования.

Данная программа имеет бэкэнд часть в виде совокупности контроллеров, объектов данных, сервисов и классов конфигурации, а так же фронтэнд часть, представленную в виде веб-страницы в браузере.

Веб-страница представлена с помощью технологии jsp (java server page), и использует языки верстки HTML, язык стилей CSS, а так же JavaScript для асинхронной отправки запросов устройствам сети и без загрузочного обновления страницы в режиме реального времени.

На рисунке 18 представлена UML-диаграмма классов системы мониторинга.

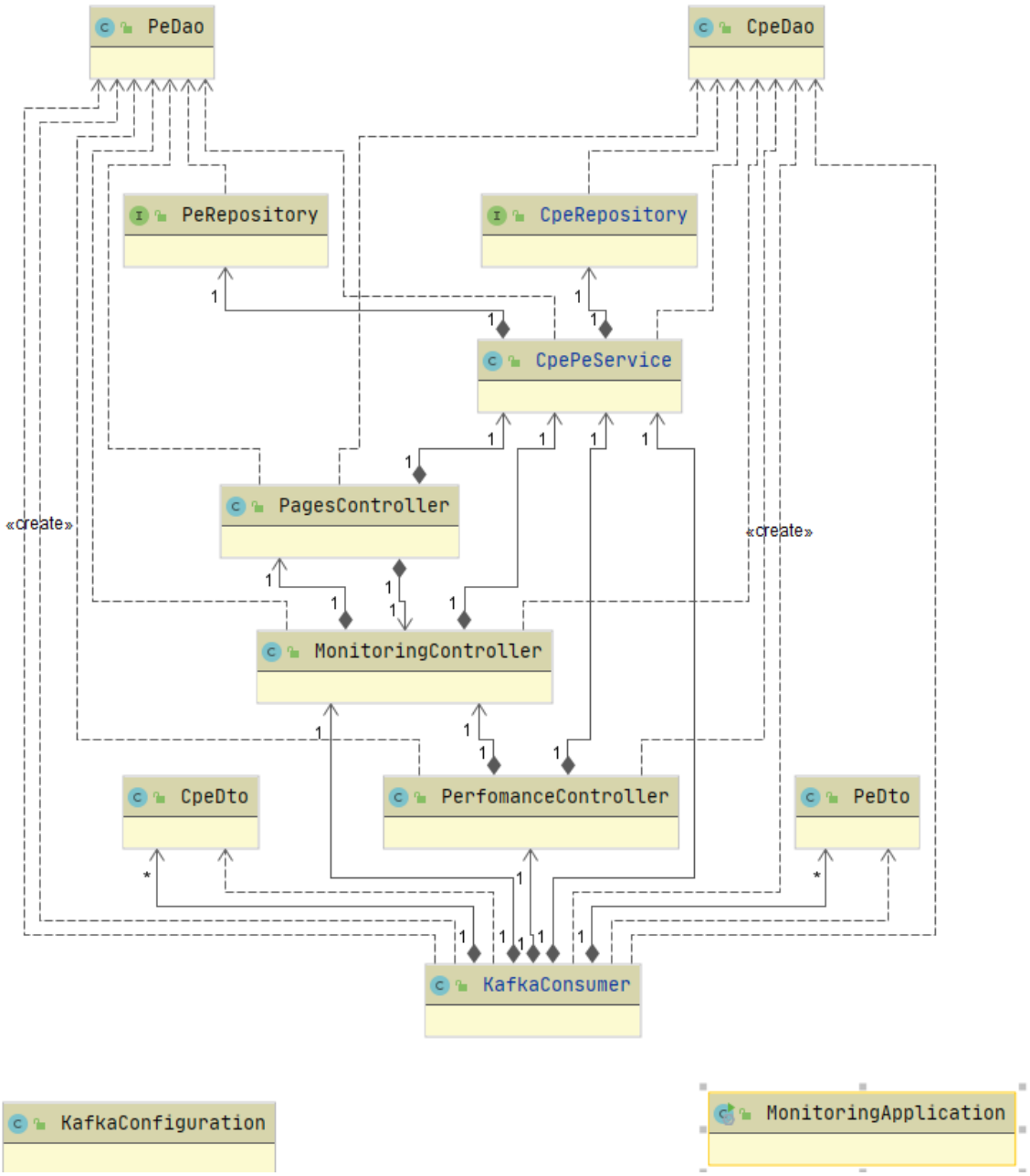

Рисунок 18 – UML-диаграмма классов системы мониторинга

Структура программы схожа со структурами устройств, однако имеет свои отличительные особенности. Далее рассмотрим функционал каждого класса:

– MonitoringApplication – содержит метод main, инициализируя приложение Spring,

– KafkaConfiguration – класс, настраивающий соединение с Apache Kafka, который использует настройки для инициализации микро-сервиса как слушателя сообщений из топика, в который NiFi агрегирует сообщения;

– CpeDao, PeDao, CpeDto, PeDto – это объекты, хранящие информацию для отрисовки сущностей на пользовательском интерфейсе и для передачи по каналам связи;

– PeRepository, CpeRepository – интерфейсы, наследующие JpaRepository, предназначены для работы с базой данных, каждый со своей таблицей;

– CpePeService – класс, который формирует SQL запросы для работы с базой данный PostgreSQL;

– PerfomanceController – класс, отвечающий за алгоритм k-NN, благодаря которому происходит автоматическое переключение устройств CPE на менее нагруженные PE;

– KafkaConsumer – класс, который взаимодействует со своим топиком Kafka, слушая из него сообщений, и производя вызовы остальных контроллеров;

– MonitoringController – REST контролер, с помощью которого веб страница управляет отображением объектов и происходит отправка управляющих команд на устройства сети;

– PagesController – контроллер, с помощью которого асинхронными запросами происходит анимация на странице пользователя без перезагрузки страницы.

Данная система получилась сложной в разработке как со стороны фронтэнда, так и со стороны бэкэнда. Большое количество асинхронных

запросов, событий, приходящих из топиков Kafka, работы с базой данных, произведение расчетов по улучшению скорости входящего трафика с помощью алгоритмов искусственного интеллекта. Код бэкэнд части находится в приложении В, а фронтэнд части в приложении Г.

3.3 Описание базы данных

Любая программа работает с данными. Для полноценной работы программе необходимо место, куда эти данные будут сохраняться, откуда будут извлекаться, при необходимости.

Для данных целей отлично подходит sql база данных, основой которой являются объекты и отношения между объектами. Существует огромное множество баз данных, от разных компаний, у каждой имеются свои особенности, преимущества и недостатки. Так как микро-сервисы, эмулирующие устройства, и система мониторинга являются самостоятельными единицами – для каждой нужно собственное место хранения.

Выбор пал на базу данных Postgres. На локальной машине разворачивание разных баз данных было ресурсозатратно, поэтому для каждой системы был создан пользователь со своим логином и паролем, с помощью которых и был организован доступ.

На рисунке 19 изображены таблицы, соответствующие таблицам в базах данных микро-сервисов PE и CPE.

PE							
IP	type	tempepature	maxSpeed	speed	fanActive	coordinateX	coordinateY

CPE							
IP	type	internateActive	maxSpeed	speed	pelpAddress	coordinateX	coordinateY

Рисунок 19 – Структура таблиц в базе данных

В данных таблицах:

– «IP» является основным ключом (PK), по которому происходит проверка, на уровне базы данных, – существует такой устройство в системе или нет,

– «Type» отображает тип устройства,

– «MaxSpeed» содержит значения максимальной скорости отдачи и приема для PE и CPE соответственно,

– «CoordinateX» и «CoordinateY» отображает местоположение данного устройства в системе и на визуальной карте системы мониторинга,

– «InternetActive» и «FanActive» являются boolean значениями-индикаторами включенности охлаждения и интернета на устройстве,

– «Speed» отображает текущую скорость отдачи и приема для PE и CPE соответственно,

– «peIpAddress» содержит значение IP адреса устройства, к которому оно подключено;

Система мониторинга и управления оборудованием сети содержит такие же таблицы, хранящие активные устройства в своей базе данных.

3.4 Интерфейс системы

Важной частью любой программы является ее пользовательский интерфейс, так как именно с ним работают пользователи, даже не подозревая что скрывается на сервере.

На рисунке 20 представлен пользовательский интерфейс.

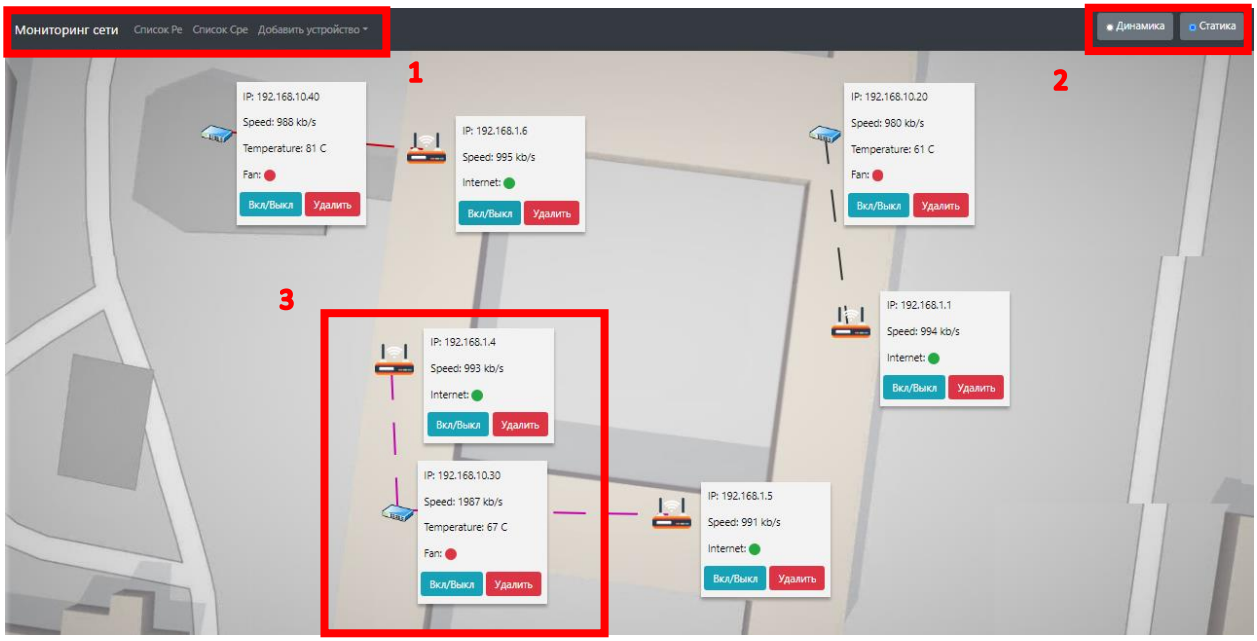

Рисунок 20 – Пользовательский интерфейс

На рисунке изображена карта здания с GoogleMaps, на территории которой расположены устройства PE (голубые) и CPE (рыжие) выделенные в блоке 3. Устройства расположены на карте в соответствии с координатами, которые имеются в их атрибутах. Каждый CPE связан со своим PE анимированной линией, эмулирующую связь 2х объектов. Рядом с каждым из устройств имеется табличка, отображающая его характеристики: IP, Speed, Internet, а так же кнопки управления, позволяющие удалить устройство из базы данных или включить/выключить охлаждение или интернет.

С помощью блока 2 можно выключить отображение блоков с информацией, что бывает удобно, если на карте расположено множество устройств. Для просмотра информации об устройстве можно навести на него мышку и тогда блок вновь появится, как показано на рисунке 21.

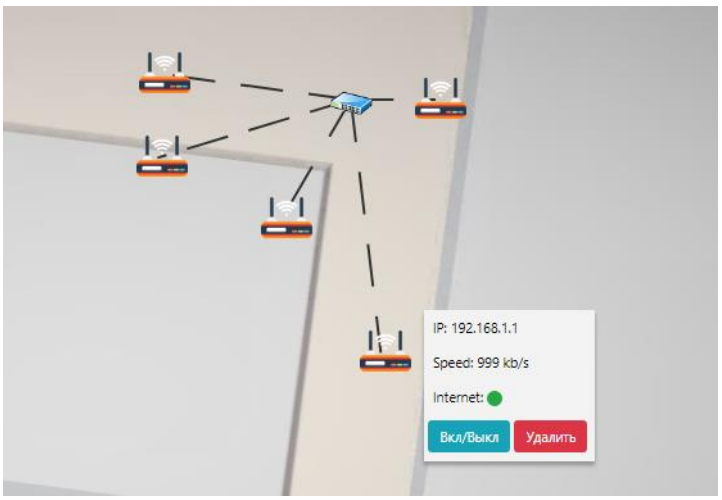

Рисунок 21 – Динамическое отображение информации

С помощью блока 1 на рисунке 20 происходит просмотр всех активных устройств в сети и добавление новых. В данном блоке так же находятся кнопки управления устройствами сети, если необходимо массово выключить оборудование или удалить. При добавлении устройства появляется диалоговое окно, при заполнении которого отправляется асинхронный REST запрос на один из микро-сервисов, где и происходит его создание. На рисунке 22 представлены оба вида окон.

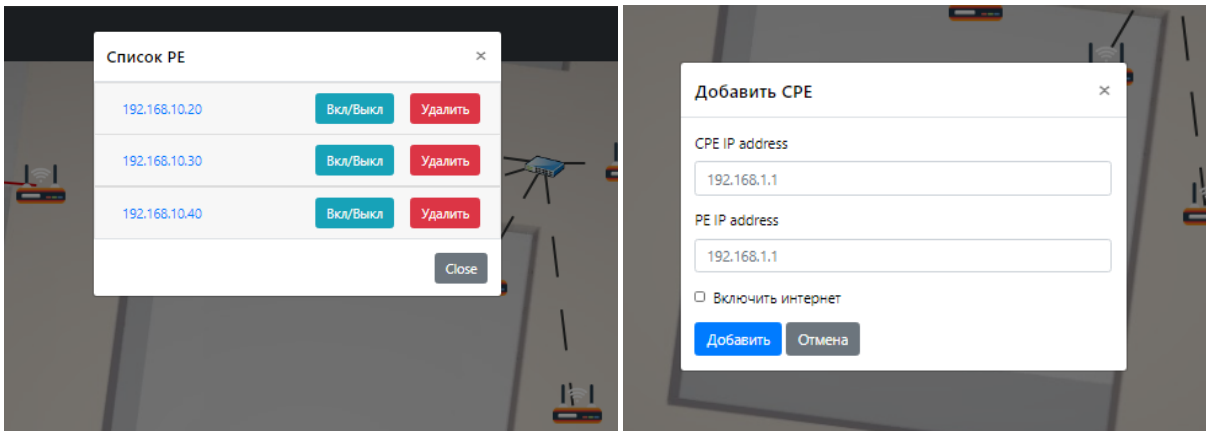

Рисунок 22 – Окно просмотра активных устройств
и окно добавления нового устройства

Следом приведен алгоритм работы всей системы, первым делом будут описаны процессы, которые присутствуют в системе постоянно:

1) Каждые 2 секунды веб страница шлет асинхронные запросы на систему мониторинга, для того, чтобы произвести обновление информации о места нахождении объектов, их значениях температуры и скоростей;

2) Каждые 3 секунды происходит обновление списков, находящихся в блоке 1 на рисунке 20 тем же алгоритмом;

3) Каждые 0,2 секунды происходит обновление анимации линий связи между устройствами;

4) Каждые 3,5 секунды система мониторинга выбирает устройство CPE и, используя k-NN пытается выбрать ему новое устройство для связи;

5) Каждые 3,5 секунды каждый микро-сервис отсылает свои данные в свой топик Kafka, где агрегирующий модуль собирает их сообщения и отправляет в систему мониторинга;

6) После приема сообщений из топика, система мониторинга записывает данные в базу данных, вся остальная работа с данными перекладывается на асинхронные запросы от jsp;

7) Каждое действие пользователя, такое как перенос оборудования по веб интерфейсу, удаление устройства, добавление устройства – сопровождается асинхронным запросом на соответствующий микро-сервис для обновление параметров объекта.

3.5 Анализ результатов

Разработанная система мониторинга и микро-сервисы запускаются на разделенных серверах Tomcat средствами Spring Boot.

Система показала себя отказоустойчивой и тестировалась ручным тестированием. Каждая эмуляция устройств создавала по сто устройств, которыми система мониторинга могла управлять и отображать.

Данные проблемы связаны с железом, на котором запускался весь проект. Нагрузка приходилась в основном на процессор. Для запуска всего проекта необходимо единовременно запустить на компьютере:

– 3 сервера Tomcat, каждый из которых работает со своим приложением,

– базу данных PostgreSQL для трех пользователей,

– программу NiFi, агрегирующую данные из входных топиков и передающая в выходной,

– сервер zookeeper, на котором разворачивается сервер брокера сообщений;

Все эти приложения расходуют вычислительную мощность, поэтому, если бы система работала с реальными устройствами, то не было бы необходимости запускать эмуляционные микро-сервисы.

Система обладает перспективным потенциалом в следующих направлениях разработки:

– добавление новых устройств. Система не зависит от устройств, которые с ней работают. Для расширения приложения необходимо написать новый микро-сервис для эмуляции и несколько классов в системе мониторинга, работающих с объектами, а так же методы их контроля;

– анализ данных. Для наглядности пользователя система может собирать данные в базе данных и преобразовывать их в графики по команде пользователя, для выявления тенденций в работе устройств;

– удобство отображения. Добавление новой страницы, где бы выводились все объекты в табличном виде со всеми параметрами, или древовидном виде, отображая связи объектов;

– горизонтальное расширение. Используя связи, Kafka+NiFi, горизонтальное расширение становится максимально доступным. Добавив новый параллельный модуль – пропускная способность системы вырастет, без затрат на расширение. Используя данные модули, создаются новые топики Kafka, не требующие никаких затрат ресурсов.

Заключение

В ходе выполнения бакалаврской работы были рассмотрены виды сетевого оборудования, возможные алгоритмы из взаимодействия, существующие продуктовые решения систем мониторинга.

В ходе работы были выполнены следующие задачи:

1. Изучить существующее сетевое оборудование;
2. Проанализировать рынок и выявить необходимость в системах мониторинга;
3. Разработать эмуляцию сетевого оборудования со стороны провайдера и пользователя;
4. Разработать систему, собирающую информацию о сетевом оборудовании и позволяющую производить манипуляцию с ним;
5. Внедрить алгоритмы интеллектуального анализа в систему для автоматизации некоторых работ.

В ходе работы были разработаны два микро-сервиса, отвечающие за эмуляцию сетевого оборудования моделируемой сети и система мониторинга и контроля сетевого оборудования, которые взаимодействуют посредством топиков и HTTP REST запросов.

Алгоритмы искусственного интеллекта были использованы для автоматизации переключения маршрутизаторов пользователя между маршрутизаторами провайдера, кластеризации имеющихся устройств и прогнозирования скорости интернета для устройств CPE.

Полученная система показала себя как отказоусточивая и обладающая перспективным потенциалом для дальнейшей модернизации по многим направлениям разработки.

Список используемой литературы и используемых источников

1. Андреев А. Е. Дискретная математика: прикладные задачи и сложность алгоритмов : учебник и практикум для академического бакалавриата / А. Е. Андреев, А. А. Болотов, К. В. Коляда, А. Б. Фролов. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 317 с.

2. Аверина, Т. А. Численные методы. Верификация алгоритмов решения систем со случайной структурой : учебное пособие для вузов / Т. А. Аверина. — Москва : Издательство Юрайт, 2019. — 179 с.

3. Игошин, В.И. Теория алгоритмов: Учебное пособие / В.И. Игошин. - М.: ИНФРА-М, 2013. - 318 с.

4. Канцедал, С.А. Алгоритмизация и программирование : Учебное пособие / С.А. Канцедал. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. - 352 с.

5. Крупский, В.Н. Математическая логика и теория алгоритмов: Учебное пособие для студентов учреждений высшего проф. образования / В.Н. Крупский, В.Е. Плиско. - М.: ИЦ Академия, 2013. - 416 с.

6. Семакин, И.Г. Основы алгоритмизации и программирования. Практикум: Учебное пос. для студ. учреждений сред. проф. образования / И.Г. Семакин, А.П. Шестаков. - М.: ИЦ Академия, 2013. - 144 с.

7. Судоплатов, С. В. Математика: математическая логика и теория алгоритмов : учебник и практикум для среднего профессионального образования / С. В. Судоплатов, Е. В. Овчинникова. — 5-е изд., стер. — Москва : Издательство Юрайт, 2019. — 255 с.

8. Семакин, И.Г. Основы алгоритмизации и программирования: Учебник для студ. учреждений сред. проф. образования / И.Г. Семакин, А.П. Шестаков. - М.: ИЦ Академия, 2013. - 304 с.

9. Трофимов, В. В. Алгоритмизация и программирование : учебник для академического бакалавриата / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2019. — 137 с.

10. Черняк А. А. Методы оптимизации: теория и алгоритмы : учебное пособие для академического бакалавриата, 2019

11. Васильев А. Научные вычисления в Microsoft Excel. Москва: Вильяме, 2004

12. Водяхо А., Горнец Н., Пузанков Д. Высокопроизводительные системы обработки данных. Москва: Высшая школа, 1997

13. Гарнаев А. Visual Basic .NET Разработка приложений. Санкт-Петербург: БХВ-Петербург, 2002

14. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания. Москва: URSS, 2005

15. Семакин, И.Г. Основы алгоритмизации и программирования: Учебник для студ. учреждений сред. проф. образования / И.Г. Семакин, А.П. Шестаков. - М.: ИЦ Академия, 2013. - 304 с.

16. Гук М. Аппаратные средства локальных сетей. Энциклопедия. Санкт-Петербург: Питер, 2000

17. Водяхо А Аппаратные средства 1BM PC: Энциклопедия. Санкт-Петербург: Питер, 2004

18. MestriaM. New hybrid heuristic algorithm for the clustered traveling salesman problem. Computers & Industrial Engineering // 116: 1-12. –2018.

19. Padberg M.W. The travelling salesman problem and a class of polyhedra of diameter two / M.W. Padberg, M. R.Rao // Math. Program. –1974. –7(1), 32

20. Potvin J.Y. A Genetic Algorithm for the Clustered Traveling Salesman Problem with a Prespecified Order on the Clusters / J.Y. Potvin, F. Guertin //Computer Science Interfaces Series. – vol 9. – Springer, Boston, MA, 1998.

21. Raheem, N. (2019). Big Data : A Tutorial-Based Approach (Vol. First edition). Boca Raton: Chapman and Hall/CRC. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&site=eds

22. Simon, P. (2014). The Visual Organization : Data Visualization: Wiley. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=707200

Приложение А

Листинг кода устройства CPE

```
package netcracker.cpe;

@EnableScheduling
@SpringBootApplication
public class CpeApplication {

     public static void main (String[] args) {
          Locale.setDefault(Locale.ENGLISH);
          SpringApplication.run(CpeApplication.class, args);
     }
}

package netcracker.cpe.entity.dao;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table ( name = "CPE" )
public class CpeDao {

     @Id
     private String ip;
     private String type;
     private boolean internetActive;
     private Integer maxSpeed;
     private Integer speed;
     private String peIpAddress;
     private Integer coordinateX;
     private Integer coordinateY;

}

package netcracker.cpe.entity.dao;

public interface CpeRepository extends JpaRepository<CpeDao,
String> {

     CpeDao getOneByIp (String ip);

     List<CpeDao> findAllByPeIpAddressAndInternetActive (String
peIp, boolean internet);
}

package netcracker.cpe.entity.dto;

@Data
@AllArgsConstructor
```

```
@NoArgsConstructor
public class PeDto {

     private String ip;
     private String type;
     private Integer temperature;
     private Integer maxSpeed;
     private Integer speed;
     private boolean fanActive;
     private Integer coordinateX;
     private Integer coordinateY;
}

package netcracker.cpe.integration;

@RestController
@RequestMapping ( "/cpe" )
public class CpeController {

     private static final String TOPIC = "cpe_topic";

     @Autowired
     private CpeService service;

     @Autowired
     private KafkaTemplate<String, List<CpeDao>> kafkaTemplate;

     @Autowired
     private KafkaTemplate<String, String> kafkaTemplateString;

     @GetMapping ( "/all" )
     public List<CpeDao> getAll ( ) {
          return service.findAll();
     }

     @GetMapping ( "/add/{cpeStr}" )
     public void addOne (@PathVariable ( "cpeStr" ) String
cpeStr) {
          Gson gson = new Gson();
          CpeDao cpe = gson.fromJson(cpeStr, CpeDao.class);
          cpe.setType("CPE");
          cpe.setSpeed(1000);
          cpe.setSpeed(0);
          cpe.setCoordinateX(100);
          cpe.setCoordinateY(100);
          service.saveCpe(cpe);
     }

     @GetMapping ( "/delete/{ip}" )
     public void deleteByIp (@PathVariable String ip) {
          service.deleteCpe(ip);
     }
```

```
     @GetMapping ( "/internet/{ip}" )
     public @ResponseBody
     ResponseEntity<HttpStatus> changeInternetStatus
(@PathVariable ( "ip" ) String ip) {
          CpeDao cpeDao = service.getCpeByIp(ip);
          if (cpeDao.isInternetActive()) {
               cpeDao.setInternetActive(false);
               cpeDao.setSpeed(0);
          } else {
               cpeDao.setInternetActive(true);
          }
          service.saveCpe(cpeDao);
          return new ResponseEntity(HttpStatus.OK);
     }

     @PostMapping ( "/peData" )
     public void generateCpeSpeed (@RequestBody List<PeDto> list)
{
          List<CpeDao> listCpe;
          SpeedGenerator generator = new SpeedGenerator();
          for (PeDto peDto : list) {
               listCpe =
service.findAllByPeIpAddressAndInternetActive(peDto.getIp());
               listCpe = generator.generate(peDto, listCpe);
               for (CpeDao cpe : listCpe) {
                    service.saveCpe(cpe);
               }
          }
          listCpe = service.findAll();
          kafkaTemplate.send(TOPIC, listCpe);

          String linkToAddCpe = "http://localhost:8080/cpe/add";
          String linkToDeleteCpe =
"http://localhost:8080/cpe/delete";
          String linkToInternet =
"http://localhost:8080/cpe/internet";
          String totalString = linkToAddCpe + ',' +
linkToDeleteCpe + ',' + linkToInternet;
          kafkaTemplateString.send(TOPIC, totalString);
     }

     public void sendCpe ( ) {
          try {
               RestTemplate rt = new RestTemplate();
               String url =
"http://localhost:8081/pe/cpeData2";//URL to PE adoption
               List<CpeDao> list = service.findAll();
               rt.postForEntity(url, list, List.class);
          } catch (Exception e) {
               System.out.println("No connecting to: " +
"http://localhost:8081/pe");
          }
     }
```

```
}

package netcracker.cpe.service;

@Service
public class CpeService {

     @Autowired
     CpeRepository cpeRepository;

     public CpeDao getCpeByIp (String ip) {
          return cpeRepository.getOneByIp(ip);
     }

     public void saveCpe (CpeDao cpe) {
          cpeRepository.save(cpe);
     }

     public void deleteCpe (String ip) {
          cpeRepository.deleteById(ip);
     }

     public List<CpeDao> findAll ( ) {
          return cpeRepository.findAll();
     }

     public List<CpeDao> findAllByPeIpAddressAndInternetActive
(String peIp) {
          return
cpeRepository.findAllByPeIpAddressAndInternetActive(peIp, true);
     }
}

package netcracker.cpe.service;

public class SpeedGenerator {

     public List<CpeDao> generate (PeDto peDto, List<CpeDao>
list) {

          Random random = new Random();
          Integer temperature = peDto.getTemperature();
          Integer speed;
          Integer maxSpeed = peDto.getMaxSpeed();
          try {

               if (temperature < 70) {
                    maxSpeed = maxSpeed / list.size();
                    if (maxSpeed > 1000) maxSpeed = 1000;
                    for (CpeDao cpe : list) {
                         speed = random.nextInt(21) + (maxSpeed -
20);
                         cpe.setSpeed(speed);
```

```
                    }
                } else if (temperature < 80) {
                    maxSpeed = maxSpeed / 2 / list.size();
                    if (maxSpeed > 1000) maxSpeed = 1000;
                    for (CpeDao cpe : list) {
                        speed = random.nextInt(21) + (maxSpeed -
20);
                        cpe.setSpeed(speed);
                    }
                } else if (temperature < 90) {
                    maxSpeed = maxSpeed / 4 / list.size();
                    if (maxSpeed > 1000) maxSpeed = 1000;
                    for (CpeDao cpe : list) {
                        speed = random.nextInt(21) + (maxSpeed -
20);
                        cpe.setSpeed(speed);
                    }
                } else if (temperature < 100) {
                    maxSpeed = maxSpeed / 8 / list.size();
                    if (maxSpeed > 1000) maxSpeed = 1000;
                    for (CpeDao cpe : list) {
                        speed = random.nextInt(21) + (maxSpeed -
20);
                        cpe.setSpeed(speed);
                    }
                }
            } catch (Exception e) {
                System.out.println("У PE: " + peDto.getIp() + "
нет CPE со включенным интернетом");
            }
            return list;
        }
}
```

```
package netcracker.cpe.configuration;

@Configuration
public class KafkaConfiguration {

    @Bean
    public ProducerFactory<String, List<CpeDao>> producerFactory
( ) {
        HashMap<String, Object> config = new HashMap<>();

        config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"127.0.0.1:9092");
        config.put(ConsumerConfig.GROUP_ID_CONFIG, "group_1");
        config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);

    config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
        return new DefaultKafkaProducerFactory<>(config);
```

```
    }

    @Bean
    public KafkaTemplate<String, List<CpeDao>> kafkaTemplate ( )
{
        return new KafkaTemplate<>(producerFactory());
    }

    @Bean
    public ProducerFactory<String, String> producerFactoryString
( ) {
        HashMap<String, Object> config = new HashMap<>();

        config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"127.0.0.1:9092");
        config.put(ConsumerConfig.GROUP_ID_CONFIG, "group_2");
        config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);

    config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
        return new DefaultKafkaProducerFactory<>(config);
    }

    @Bean
    public KafkaTemplate<String, String> kafkaTemplateString ( )
{
        return new KafkaTemplate<>(producerFactoryString());
    }
}

package netcracker.cpe.component;

@Component
public class CpeSender {

    @Autowired
    CpeController cpeController;

    @Scheduled ( fixedRate = 3500 )
    public void dialogWithPe ( ) {
        cpeController.sendCpe();
    }
}
```

Приложение Б

Листинг кода устройства PE

```
package netcrackerpe.pe.entity.dao;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table ( name = "PE" )
public class PeDao {

     @Id
     private String ip;
     private String type;
     private Integer temperature;
     private Integer maxSpeed;
     private Integer speed;
     private boolean fanActive;
     private Integer coordinateX;
     private Integer coordinateY;
}

package netcrackerpe.pe.entity.dto;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CpeDto {

     private String ip;
     private String type;
     private boolean internetActive;
     private Integer maxSpeed;
     private Integer speed;
     private String peIpAddress;
     private Integer coordinateX;
     private Integer coordinateY;

}

package netcrackerpe.pe.repository;

public interface PeRepository extends JpaRepository<PeDao,
String> {

     PeDao getOneByIp (String ip);
}
```

```
package netcrackerpe.pe.service;

@Service
public class PeService {
     @Autowired
     private PeRepository peRepository;

     public PeDao getPeByIp (String ip) {
          return peRepository.getOneByIp(ip);
     }

     public void savePe (PeDao pe) {
          peRepository.save(pe);
     }

     public void deletePe (String ip) {
          peRepository.deleteById(ip);
     }

     public List<PeDao> findAll ( ) {
          return peRepository.findAll();
     }
}

package netcrackerpe.pe.service;

@Component
public class TemperatureControll {

     @Autowired
     private PeService peService;

     @Scheduled ( fixedRate = 3500 )
     public void controllTemperature ( ) {
          List<PeDao> pe = peService.findAll();

          for (int i = 0; i < pe.size(); i++) {
               generateTemperature(pe.get(i));
               if (pe.get(i).isFanActive()) {
                    fan(pe.get(i));
               }
               peService.savePe(pe.get(i));
          }

     }

     public void generateTemperature (PeDao pe) {
          Integer newTemperature;
```

```
            newTemperature = pe.getTemperature() + 2;
            pe.setTemperature(newTemperature);

            if (pe.getTemperature() < 45 && pe.isFanActive()) {
                 pe.setTemperature(45);
                 pe.setFanActive(false);
            }
            if (pe.getTemperature() > 99 && !pe.isFanActive()) {
                 pe.setFanActive(true);
            }
       }

       public void fan (PeDao pe) {
            Integer cooling = 5;
            Integer currentTemperature = pe.getTemperature();
            pe.setTemperature(currentTemperature - cooling);
       }
}

package netcrackerpe.pe.kafka.config;

@Configuration
public class KafkaConfiguration {

       @Bean
       public ProducerFactory<String, List<PeDao>> producerFactory
( ) {
            HashMap<String, Object> config = new HashMap<>();

            config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"127.0.0.1:9092");
            config.put(ConsumerConfig.GROUP_ID_CONFIG, "group_1");
            config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);

       config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
            return new DefaultKafkaProducerFactory<>(config);
       }

       @Bean
       public KafkaTemplate<String, List<PeDao>> kafkaTemplate ( )
{
            return new KafkaTemplate<>(producerFactory());
       }

       @Bean
       public ProducerFactory<String, String> producerFactoryString
( ) {
```

```
          HashMap<String, Object> config = new HashMap<>();

          config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"127.0.0.1:9092");
          config.put(ConsumerConfig.GROUP_ID_CONFIG, "group_1");
          config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);

     config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
          return new DefaultKafkaProducerFactory<>(config);
     }

     @Bean
     public KafkaTemplate<String, String> kafkaTemplateString ( )
{
          return new KafkaTemplate<>(producerFactoryString());
     }
}

package netcrackerpe.pe.controller;

@RestController
@RequestMapping ( "/pe" )
public class PeController {

     @Autowired
     private PeService peService;

     @GetMapping ( value = "/all" )
     public List<PeDao> getAll ( ) {
          return peService.findAll();
     }

     @GetMapping ( "/add/{peStr}" )
     public void addOne (@PathVariable ( "peStr" ) String peStr)
{
          Gson gson = new Gson();
          PeDao pe = gson.fromJson(peStr, PeDao.class);
          pe.setType("PE");
          pe.setTemperature(40);
          pe.setMaxSpeed(10000);
          pe.setSpeed(0);
          pe.setFanActive(false);
          pe.setCoordinateX(200);
          pe.setCoordinateY(100);
          peService.savePe(pe);
     }
```

```
     @GetMapping ( "/delete/{ip}" )
     public void deleteByIp (@PathVariable String ip) {
          peService.deletePe(ip);
     }

     @GetMapping ( "/{ip}" )
     public PeDao getPeByIp (@PathVariable ( "ip" ) String ip) {
          return peService.getPeByIp(ip);
     }

     @GetMapping ( "/fan/{ip}" )
     public @ResponseBody
     ResponseEntity<HttpStatus> changeFanStatus (@PathVariable (
"ip" ) String ip) {
          PeDao peDao = peService.getPeByIp(ip);
          if (peDao.isFanActive()) {
               peDao.setFanActive(false);
          } else {
               peDao.setFanActive(true);
          }
          peService.savePe(peDao);
          return new ResponseEntity(HttpStatus.OK);
     }

     @RequestMapping ( value = "/cpeData2", method =
RequestMethod.POST )
     public ResponseEntity adoption (@RequestBody List<CpeDto>
list) {

          List<PeDao> listPe = peService.findAll();
          Integer speed = 0;

          for (int i = 0; i < listPe.size(); i++) {
               for (int s = 0; s < list.size(); s++) {
                    if
(list.get(s).getPeIpAddress().equals(listPe.get(i).getIp())) {
                         speed += list.get(s).getSpeed();
                    }
               }

     peService.getPeByIp(listPe.get(i).getIp()).setSpeed(speed);

     peService.savePe(peService.getPeByIp(listPe.get(i).getIp()))
;
               speed = 0;
          }

          return new ResponseEntity(HttpStatus.OK);
     }
```

```
     @PostMapping ( "/sendPe" )
     public void sendPe ( ) {
          try {
               RestTemplate restTemplate = new RestTemplate();
               String url = "http://localhost:8080/cpe/peData";
               List<PeDao> list = peService.findAll();
               restTemplate.postForEntity(url, list, List.class);
          } catch (Exception e) {
          }
     }
}

package netcrackerpe.pe.component;

@Component
public class PeSender {

     private static final String TOPIC = "pe_topic";

     @Autowired
     private KafkaTemplate<String, List<PeDao>> kafkaTemplate;

     @Autowired
     private KafkaTemplate<String, String> kafkaTemplateString;

     @Autowired
     private PeService peService;

     @Autowired
     private PeController peController;

     @Scheduled ( fixedRate = 3500 )
     public void conversationWithCpe ( ) {
          peController.sendPe();
          List<PeDao> listPe = peService.findAll();
          kafkaTemplate.send(TOPIC, listPe);

          String linkToAddCpe = "http://localhost:8081/pe/add";
          String linkToDeleteCpe =
"http://localhost:8081/pe/delete";
          String linkToInternet = "http://localhost:8081/pe/fan";
          String totalString = linkToAddCpe + ',' +
linkToDeleteCpe + ',' + linkToInternet;
          kafkaTemplateString.send(TOPIC, totalString);
     }
}
```

Приложение В

Листинг кода системы мониторинга

```
package com.netcracker.edu.configuration;

@Configuration
public class KafkaConfiguration {

   @Bean
   public ConsumerFactory<String, String> consumerFactory ( ) {
      Map<String, Object> config = new HashMap<>();

      config.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"127.0.0.1:9092");
      config.put(ConsumerConfig.GROUP_ID_CONFIG, "group_id");
      config.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class);
      config.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class);

      return new DefaultKafkaConsumerFactory<>(config);
   }

   @Bean
   public ConcurrentKafkaListenerContainerFactory<String,
String> kafkaListenerContainerFactory ( ) {
      ConcurrentKafkaListenerContainerFactory<String, String>
factory = new ConcurrentKafkaListenerContainerFactory();
      factory.setConsumerFactory(consumerFactory());
      return factory;
   }
}

package com.netcracker.edu.contoller;

@RestController
@RequestMapping ( "/service" )
public class MonitoringController {

     @Autowired
     CpePeService cpePeService;
     @Autowired
     private PagesController pagesController;

     private HashMap<Integer, String> peLinks = new HashMap<>();
     private HashMap<Integer, String> cpeLinks = new HashMap<>();

     public void setLinksDto (HashMap<Integer, String> peLinks,
HashMap<Integer, String> cpeLinks) {
```

```
        this.peLinks = peLinks;
        this.cpeLinks = cpeLinks;
    }

    public HashMap<Integer, String> getPeLinks ( ) {
        peLinks.put(1, "http://localhost:8081/pe/add");
        peLinks.put(2, "http://localhost:8081/pe/delete");
        peLinks.put(3, "http://localhost:8081/pe/fan");
        return peLinks;
    }

    public HashMap<Integer, String> getCpeLinks ( ) {
        cpeLinks.put(1, "http://localhost:8080/cpe/add");
        cpeLinks.put(2, "http://localhost:8080/cpe/delete");
        cpeLinks.put(3, "http://localhost:8080/cpe/internet");
        return cpeLinks;
    }

    @GetMapping ( "/deleteCpe/{ip}" )
    public void deleteCpe (@PathVariable ( "ip" ) String ip) {
        cpePeService.deleteCpe(ip);
    }

    @GetMapping ( "/deletePe/{ip}" )
    public void deletePe (@PathVariable ( "ip" ) String ip) {
        cpePeService.deletePe(ip);
    }

    @GetMapping ( "/welcome" )
    public ModelAndView welcomePage ( ) {
        ModelAndView model = new ModelAndView("WelcomePage");
        model.addObject("linksCpe", getCpeLinks());
        model.addObject("linksPe", getPeLinks());
        return model;
    }

    @ModelAttribute ( "CpeDaoList" )
    public List<CpeDao> getAllCpe ( ) {
        return cpePeService.findAllCpe();
    }

    @ModelAttribute ( "PeDaoList" )
    public List<PeDao> getAllPe ( ) {
        return cpePeService.findAllPe();
    }

    @GetMapping ( "/getListPe" )
    public String getListPe ( ) {
        return pagesController.refreshListPe();
```

```
    }

    @GetMapping ( "/getListCpe" )
    public String getListCpe ( ) {
        return pagesController.refreshListCpe();
    }

    @GetMapping ( "/refreshPeAndCpe" )
    public String refreshPeAndCpe ( ) {
        return pagesController.refreshPeAndCpe();
    }

    @GetMapping ( "/refreshXY/{ip}/{x}/{y}" )
    public void refreshXY (@PathVariable ( "ip" ) String ip,
@PathVariable ( "x" ) String x, @PathVariable ( "y" ) String y)
{
        CpeDao cpe = cpePeService.getCpeByIp(ip);
        PeDao pe = cpePeService.getPeByIp(ip);
        Integer intX;
        Integer intY;
        try {
            intX = Integer.parseInt(x.substring(0,
x.indexOf(".")));
        } catch (Exception e) {
            intX = Integer.parseInt(x);
        }
        try {
            intY = Integer.parseInt(y.substring(0,
y.indexOf(".")));
        } catch (Exception e) {
            intY = Integer.parseInt(y);
        }
        if (intX > 0 && intY > 0) {
            try {
                cpe.setCoordinateX(intX);
                cpe.setCoordinateY(intY);
                saveCpe(cpe);
            } catch (Exception e) {
                pe.setCoordinateX(intX);
                pe.setCoordinateY(intY);
                savePe(pe);
            }
        }
    }

    @GetMapping ( "/lines" )
    public String lines ( ) {
        return pagesController.lines();
    }
```

```
    private void saveCpe (@RequestBody CpeDao cpeDao) {
        cpePeService.saveCpe(cpeDao);
    }

    private void savePe (@RequestBody PeDao peDao) {
        cpePeService.savePe(peDao);
    }
}

package com.netcracker.edu.contoller;

import com.netcracker.edu.entity.dao.CpeDao;
import com.netcracker.edu.entity.dao.PeDao;
import com.netcracker.edu.service.CpePeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.io.Serializable;
import java.util.HashMap;
import java.util.List;

@Service
public class PagesController implements Serializable {


    @Autowired
    private CpePeService cpePeService;

    @Autowired
    private MonitoringController monitoringController;

    private int kindOfSwitch = 0;

    public String refreshListPe ( ) {
        String str = "";
        List<PeDao> listPe = cpePeService.findAllPe();
        for (PeDao pe : listPe) {
            str += " <div class=\"accordion\"
id=\"accordionExample\">\n" +
                    "  <div class=\"card\">\n" +
                    "    <div class=\"card-header\"
id=\"headingTwo\">\n" +
                    "      <h5 class=\"mb-0\">\n" +
                    "        <button class=\"btn btn-link\"
type=\"button\" data-toggle=\"collapse\" data-
target=\"#collapseTwo\" aria-expanded=\"false\" aria-
controls=\"collapseTwo\">\n" +
                    "          " + pe.getIp() + "\n" +
```

```
                        "        </button>\n" +
                        "<button style=\"margin-left: 120px;\"
type=\"submit\" class=\"btn btn-info\"
onclick=\"changePeInternet('" +
monitoringController.getPeLinks().get(3) + "/" + pe.getIp() +
"')\">Вкл/Выкл\n" +
                        "                    </button>" +
                        "<button style=\"margin-left: 20px;\"
type=\"submit\" class=\"btn btn-danger\" onclick=\"deletePe('" +
monitoringController.getPeLinks().get(2) + "/" + pe.getIp() +
"')\">Удалить\n" +
                        "                    </button>" +
                        "      </h5>\n" +
                        "    </div>" +
                        "</div>" +
                        "</div>" +
                        "</div>";
          }
          return str;
     }

     public String refreshListCpe ( ) {
          String str = "";
          List<CpeDao> listCpe = cpePeService.findAllCpe();
          for (CpeDao cpe : listCpe) {

               str += " <div class=\"accordion\"
id=\"accordionExample\">\n" +
                        "  <div class=\"card\">\n" +
                        "    <div class=\"card-header\"
id=\"headingTwo\">\n" +
                        "      <h5 class=\"mb-0\">\n" +
                        "        <button class=\"btn btn-link\"
type=\"button\" data-toggle=\"collapse\" data-
target=\"#collapseTwo\" aria-expanded=\"false\" aria-
controls=\"collapseTwo\">\n" +
                        "          " + cpe.getIp() + "\n" +
                        "        </button>\n" +
                        "<button style=\"margin-left: 120px;\"
type=\"submit\" class=\"btn btn-info\"
onclick=\"changeCpeInternet('" +
monitoringController.getCpeLinks().get(3) + "/" + cpe.getIp() +
"')\">Вкл/Выкл\n" +
                        "                    </button>" +
                        "<button style=\"margin-left: 20px;\"
type=\"submit\" class=\"btn btn-danger\" onclick=\"deleteCpe('"
+ monitoringController.getCpeLinks().get(2) + "/" + cpe.getIp()
+ "')\">Удалить\n" +
                        "                    </button>" +
```

```
                        "      </h5>\n" +
                        "    </div>" +
                        "</div>" +
                        "</div>" +
                        "</div>";
          }
          return str;
     }

     public String refreshPeAndCpe ( ) {
          List<PeDao> listPe = cpePeService.findAllPe();
          List<CpeDao> listCpe = cpePeService.findAllCpe();

          String str = "";
          for (int i = 0; i < listPe.size(); i++) {
               str += " <div class=\"item two \"
style=\"position: absolute; top: " +
listPe.get(i).getCoordinateX() + "px;" +
                        "left: " +
listPe.get(i).getCoordinateY() + "px; z-index: 10\" id=\"" +
listPe.get(i).getIp() + "\">\n" +

                        "                <div class=\"descrPE
descr\" name=\"test1\">\n" +
                        "                    <p class=\"pcl\">"
+ "IP:          " + listPe.get(i).getIp() + "</p>\n" +
                        "                    <p class=\"pcl\">"
+ "Speed:       " + listPe.get(i).getSpeed() + " kb/s</p>\n" +
                        "                    <p class=\"pcl\">"
+ "Temperature: " + listPe.get(i).getTemperature() + " C
</p>\n";
               if (listPe.get(i).isFanActive()) {
                    str += "                    <p
class=\"pcl\">" + "Fan:         " + "<span class=\"badge badge-
pill badge-success\"><span style=\"visibility:
hidden\">.</span></span>" + "</p>\n";
               } else {
                    str += "                    <p
class=\"pcl\">" + "Fan:         " + "<span class=\"badge badge-
pill badge-danger\"><span style=\"visibility:
hidden\">.</span></span>" + "</p>\n";
               }
               str += "<button style=\"margin-left: 5px;\"
type=\"submit\" class=\"btn btn-info\" onclick=\"changePeFan('"
+ monitoringController.getPeLinks().get(3) + "/" +
listPe.get(i).getIp() + "')\">Вкл/Выкл\n" +
                        "                    </button>" +
                        "<button style=\"margin-left: 5px;\"
type=\"submit\" class=\"btn btn-danger\" onclick=\"deletePe('" +
```

```
monitoringController.getPeLinks().get(2) + "/" +
listPe.get(i).getIp() + "')\">Удалить\n" +
                        "                    </button>";
                str +=
                        "                </div>\n" +
                        "                </div>";
        }

        for (int i = 0; i < listCpe.size(); i++) {
             str += " <div class=\"item three \"
style=\"position: absolute; top: " +
listCpe.get(i).getCoordinateX() + "px;" +
                        "left: " +
listCpe.get(i).getCoordinateY() + "px;\" id=\"" +
listCpe.get(i).getIp() + "\">\n" +
                        "                <div class=\"descrCPE
descr\" name=\"test2\">\n" +
                        "                    <p class=\"pcl\">"
+ "IP:          " + listCpe.get(i).getIp() + "</p>\n" +
                        "                    <p class=\"pcl\">"
+ "Speed:       " + listCpe.get(i).getSpeed() + " kb/s</p>\n";
               if (listCpe.get(i).isInternetActive()) {
                    str += "                    <p
class=\"pcl\">" + "Internet: " + "<span class=\"badge badge-pill
badge-success\"><span style=\"visibility:
hidden\">.</span></span>" + "</p>\n";
               } else {
                    str += "                    <p
class=\"pcl\">" + "Internet: " + "<span class=\"badge badge-pill
badge-danger\"><span style=\"visibility:
hidden\">.</span></span>" + "</p>\n";
               }
               str += "<button style=\"margin-left: 5px;\"
type=\"submit\" class=\"btn btn-info\"
onclick=\"changeCpeInternet('" +
monitoringController.getCpeLinks().get(3) + "/" +
listCpe.get(i).getIp() + "')\">Вкл/Выкл\n" +
                        "                    </button>" +
                        "<button style=\"margin-left: 5px;\"
type=\"submit\" class=\"btn btn-danger\" onclick=\"deleteCpe('"
+ monitoringController.getCpeLinks().get(2) + "/" +
listCpe.get(i).getIp() + "')\">Удалить\n" +
                        "                    </button>" +
                        "                </div>\n" +
                        "            </div>";
        }
        return str;

    }
```

```
    public String lines ( ) {

        List<PeDao> listPe = cpePeService.findAllPe();
        List<CpeDao> listCpe = cpePeService.findAllCpe();
        HashMap<Integer, String> colors = new HashMap<>(7);
        String str = "";
        colors.put(0, "#323433");
        colors.put(1, "#C300AE");
        colors.put(2, "#C30008");
        colors.put(3, "#00C60B");
        colors.put(4, "#0100C4");
        colors.put(5, "#24657A");
        colors.put(6, "#C37C00");
        for (int j = 0; j < listPe.size(); j++) {
            for (int i = 0; i < listCpe.size(); i++) {
                if
(listPe.get(j).getIp().equals(listCpe.get(i).getPeIpAddress()))
{
                    switch (kindOfSwitch) {
                        case 0: {
                            str += "<line x1=\"" +
(listPe.get(j).getCoordinateY() - 25) + "\" y1=\"" +
(listPe.get(j).getCoordinateX() + 75) + "\" x2=\""
                                    +
(listCpe.get(i).getCoordinateY() - 25) + "\" y2=\"" +
(listCpe.get(i).getCoordinateX() + 75) + "\" style=\"stroke: " +
colors.get(j) + ";stroke-width:3; filter: brightness(0.6)\"
stroke-dasharray=\"50px 40px\"  />\n";
                            kindOfSwitch++;
                        }
                        break;
                        case 1: {
                            str += "<line x1=\"" +
(listPe.get(j).getCoordinateY() - 25) + "\" y1=\"" +
(listPe.get(j).getCoordinateX() + 75) + "\" x2=\""
                                    +
(listCpe.get(i).getCoordinateY() - 25) + "\" y2=\"" +
(listCpe.get(i).getCoordinateX() + 75) + "\" style=\"stroke: " +
colors.get(j) + ";stroke-width:3; filter: brightness(0.6)\"
stroke-dasharray=\"50px 35px\" />\n";
                            kindOfSwitch++;
                        }
                        break;
                        case 2: {
                            str += "<line x1=\"" +
(listPe.get(j).getCoordinateY() - 25) + "\" y1=\"" +
(listPe.get(j).getCoordinateX() + 75) + "\" x2=\""
```

```
                                                    +
(listCpe.get(i).getCoordinateY() - 25) + "\" y2=\"" +
(listCpe.get(i).getCoordinateX() + 75) + "\" style=\"stroke: " +
colors.get(j) + ";stroke-width:3; filter: brightness(0.6)\"
stroke-dasharray=\"50px 30px\" />\n";
                                    kindOfSwitch++;
                                }
                                break;
                                default: {
                                    str += "<line x1=\"" +
(listPe.get(j).getCoordinateY() - 25) + "\" y1=\"" +
(listPe.get(j).getCoordinateX() + 75) + "\" x2=\""
                                                    +
(listCpe.get(i).getCoordinateY() - 25) + "\" y2=\"" +
(listCpe.get(i).getCoordinateX() + 75) + "\" style=\"stroke: " +
colors.get(j) + ";stroke-width:3; filter: brightness(0.6)\"
stroke-dasharray=\"50px 25px\" />\n";
                                    kindOfSwitch = 0;
                                }
                            }
                        }
                    }
                }
                return str;
            }
}

package com.netcracker.edu.service;

import com.netcracker.edu.entity.dao.CpeDao;
import com.netcracker.edu.entity.dao.CpeRepository;
import com.netcracker.edu.entity.dao.PeDao;
import com.netcracker.edu.entity.dao.PeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CpePeService {

    @Autowired
    CpeRepository cpeRepository;

    @Autowired
    PeRepository peRepository;

    public void saveCpe(CpeDao cpe) {
        cpeRepository.save(cpe);
```

```
    }

    public void deleteCpe(String ip) {
        cpeRepository.deleteById(ip);
    }

    public void savePe(PeDao pe) {
        peRepository.save(pe);
    }

    public CpeDao getCpeByIp(String ip) {
        return cpeRepository.getOne(ip);
    }

    public PeDao getPeByIp(String ip) {
        return peRepository.getOne(ip);
    }

    public CpeDao getCpe(Integer number) {
        return cpeRepository.getOneBySpeed(number);
    }

    public void deletePe(String ip) {
        peRepository.deleteById(ip);
    }

    public List<CpeDao> findAllCpe() {
        List<CpeDao> list = cpeRepository.findAll();
        list.sort((o1, o2) ->
o1.getIp().compareToIgnoreCase(o2.getIp()));
        return list;
    }

    public List<PeDao> findAllPe() {
        List<PeDao> list = peRepository.findAll();
        list.sort((o1, o2) ->
o1.getIp().compareToIgnoreCase(o2.getIp()));
        return list;
    }

}

package com.netcracker.edu.service;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.reflect.TypeToken;
import com.netcracker.edu.contoller.MonitoringController;
import com.netcracker.edu.contoller.PerformanceController;
```

```
import com.netcracker.edu.entity.dao.CpeDao;
import com.netcracker.edu.entity.dao.PeDao;
import com.netcracker.edu.entity.dto.CpeDto;
import com.netcracker.edu.entity.dto.PeDto;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;

import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

@Service
public class KafkaConsumer {

    @Autowired
    MonitoringController monitoringController;

    @Autowired
    CpePeService cpePeService;

    @Autowired
    PerformanceController performanceController;

    private HashMap<Integer, String> peLinks = new HashMap<>();
    private HashMap<Integer, String> cpeLinks = new HashMap<>();
    private Type listCpeType = new
TypeToken<ArrayList<CpeDto>>() {
    }.getType();
    private Type listPeType = new TypeToken<ArrayList<PeDto>>()
{
    }.getType();

    @KafkaListener(topics = "total_topic")
    public void consume(String message) {

        GsonBuilder builder = new GsonBuilder();
        Gson gson = builder.create();
        if (message.contains("fanActive")) {
            List<PeDao> listPe = cpePeService.findAllPe();
            List<PeDto> listPeFromMess = gson.fromJson(message,
listPeType);
            savePe(listPeFromMess, listPe);
        } else if (message.contains("internetActive")) {
            List<CpeDao> listCpe = cpePeService.findAllCpe();
            List<CpeDto> listCpeFromMess =
gson.fromJson(message, listCpeType);
            saveCpe(listCpeFromMess, listCpe);
```

```
        } else if (message.contains("8081")) {
            splitLinks(message, peLinks);
        } else {
            splitLinks(message, cpeLinks);
            monitoringController.setLinksDto(peLinks, cpeLinks);
        }
        performanceController.controlPerformanceOnCpe();
    }

    public void splitLinks(String mes, HashMap<Integer, String>
links) {
        String[] arrLinks = mes.split(",");
        links.put(1, arrLinks[0].substring(1));
        links.put(2, arrLinks[1]);
        links.put(3, arrLinks[2].substring(0,
arrLinks[2].length() - 1));
    }

    public void savePe(List<PeDto> listPeFromMess, List<PeDao>
listPe) {
        boolean isFind = false;
        for (PeDto peDto : listPeFromMess) {
            for (PeDao peDao : listPe) {
                if (peDao.getIp().equals(peDto.getIp())) {
                    peDao.setFanActive(peDto.isFanActive());
                    peDao.setSpeed(peDto.getSpeed());

peDao.setTemperature(peDto.getTemperature());
                    cpePeService.savePe(peDao);
                    isFind = true;
                    break;
                }
            }
            if (!isFind) {
                PeDao pe = new PeDao();
                pe.setIp(peDto.getIp());
                pe.setCoordinateX(peDto.getCoordinateX());
                pe.setCoordinateY(peDto.getCoordinateY());
                pe.setFanActive(peDto.isFanActive());
                pe.setMaxSpeed(peDto.getMaxSpeed());
                pe.setSpeed(peDto.getSpeed());
                pe.setTemperature(peDto.getTemperature());
                pe.setType(peDto.getType());
                cpePeService.savePe(pe);
            }
            isFind = false;
        }
    }
```

```
    public void saveCpe(List<CpeDto> listCpeFromMess,
List<CpeDao> listCpe) {
        boolean isFind = false;
        for (CpeDto cpeDto : listCpeFromMess) {
            for (CpeDao cpeDao : listCpe) {
                if (cpeDao.getIp().equals(cpeDto.getIp())) {

cpeDao.setInternetActive(cpeDto.isInternetActive());
                    cpeDao.setSpeed(cpeDto.getSpeed());
                    cpePeService.saveCpe(cpeDao);
                    isFind = true;
                    break;
                }
            }
            if (!isFind) {
                CpeDao cpe = new CpeDao();
                cpe.setIp(cpeDto.getIp());
                cpe.setCoordinateX(cpeDto.getCoordinateX());
                cpe.setCoordinateY(cpeDto.getCoordinateY());

cpe.setInternetActive(cpeDto.isInternetActive());
                cpe.setMaxSpeed(cpeDto.getMaxSpeed());
                cpe.setSpeed(cpeDto.getSpeed());
                cpe.setType(cpeDto.getType());
                cpe.setPeIpAddress(cpeDto.getPeIpAddress());
                cpePeService.saveCpe(cpe);
            }
            isFind = false;
        }
    }
}
```

Приложение Г

Листинг кода страницы интерфейса

```
<!DOCTYPE html>
<html lang="ru">

<%@ page import="java.util.HashMap" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="spring"
uri="http://www.springframework.org/tags/form" %>
<% HashMap<Integer, String> linksCpe = (HashMap<Integer,
String>) request.getAttribute("linksCpe");%>
<% HashMap<Integer, String> linksPe = (HashMap<Integer, String>)
request.getAttribute("linksPe");%>
<head>
    <meta http-equiv="content-Type" content="text/html"
charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css"
          integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T
" crossorigin="anonymous">
    <style src="..//css/line.css"></style>
    <title>Главная страница</title>
    <!-- CSS -->
    <style>
        body {
            background: #c7b39b url(/images/map.jpg);
            backdrop-repeat: no-repeat;
            color: black;
            background-size: cover;
            width: 1920px;
            height: 980px;
        }
    </style>
    <!-- DRAGIN CSS -->
    <style>
        #container {
            width: 2020px;
            height: 980px;
            display: flex;
            align-items: center;
```

```
        justify-content: center;
        overflow: hidden;
        border-radius: 7px;
        touch-action: none;
        position: relative;
    }

    .two {
        width: 50px;
        height: 50px;
        background: url(/images/PE.png);
    }

    .three {
        width: 60px;
        height: 48px;
        background: url(/images/CPE.png);
    }

    .item:active {
        opacity: .75;
    }

    .item:hover {
        cursor: pointer;
    }

    h1 {
        margin-bottom: 10px;
    }
    .pcl{
        margin-left: 10px;
    }

</style>
<!-- /DRAGIN CSS -->
<style>
    .descr {
    display: none;
    }

    .descrCPE {
    padding-top: 10px;
    height: 370%;
    width: 200px;
    margin-left: 75px;
    margin-top: -25px;
    background: #f3f3f3;
    -moz-box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
```

```
        -webkit-box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
        box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
        }

        .descrPE {
            padding-top: 10px;
            height: 440%;
            width: 200px;
            margin-left: 55px;
            margin-top: -50px;
            /*margin-top:-75px;*/
            background: #f3f3f3;
            -moz-box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
            -webkit-box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
            box-shadow: 0 5px 5px rgba(0, 0, 0, 0.3);
        }

        .two:hover .descrPE{
        display: block;
        margin-left: 55px;
        margin-top: -50px;
        height: 440%;
        z-index: 9999;
        width: 200px;
        }

        .three:hover .descrCPE {
        display: block;
        margin-left: 75px;
        margin-top: -25px;
        height: 370%;
        z-index: 9999;
        width: 200px;
        }

        .pcl{
            margin-left: 10px;
        }
    </style>

    <style id="dark">
        .descr{
            display: block;
        }
    </style>

    <!-- /CSS -->
</head>
<body>
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.m
in.js"></script>
<script src="../js/refresh.js"></script><!-- Script for generate
PE/CPE-->
<script async=""
src="../welcomeBootstrapFiles/watch.js.download"></script>
<script async=""
src="../welcomeBootstrapFiles/analytics.js.download"></script>
<script>


</script>

<!-- NAVIGATION BAR-->
<nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-
top">

    <a class="navbar-brand" href="">Мониторинг сети</a>
    <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarsExampleDefault"
            aria-controls="navbarsExampleDefault" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse"
id="navbarsExampleDefault">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" data-toggle="modal" data-
target="#PeList">Список Pe</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" data-toggle="modal" data-
target="#CpeList">Список Cpe</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle"
href="https://bootstrap-4.ru/docs/4.3.1/examples/starter-
template/#"
                   id="dropdown01" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">Добавить
                    устройство</a>
                <div class="dropdown-menu" aria-
labelledby="dropdown01">
                    <a class="dropdown-item" data-toggle="modal"
data-target="#addCPE">Добавить Cpe</a>
```

```
                    <a class="dropdown-item" data-toggle="modal"
data-target="#addPE">Добавить Pe</a>
                </div>
            </li>
        </ul>
            <label class="btn btn-secondary"><input type="radio"
name="chooseStyle" value="block" checked=""> <span
style="vertical-align: 3px;">Динамика</span></label>
        <label class="btn btn-secondary" style="margin-left:
10px"><input type="radio" name="chooseStyle" value="none"><span
style="vertical-align: 3px;"> Статика</span> </label>
            <script>
                (function() {
                    var styles = {
                        block: $("#light"),
                        none:  $("#dark").detach()
                    };

$("input[name=chooseStyle]").click(function() {
                        var other = this.value === "block" ?
"none" : "block";
                        styles[this.value].appendTo('head');
                        styles[other].detach();
                    });
                })();
            </script>
    </div>
</nav>
<!-- /NAVIGATION BAR-->

<!-- Modal for List Cpe -->
<div class="modal fade" id="CpeList" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLongTitle"
     aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="CpeListTitle">Список
CPE</h5>
                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <!-- List Cpe -->
            <div id="cpeListId">
                <div class="list-group">
                </div>
            </div>
```

```
            <!-- /List Cpe -->
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary"
data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
<!-- /Modal for List Cpe -->

<!-- Modal for List Pe -->
<div class="modal fade" id="PeList" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLongTitle"
     aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="PeListTitle">Список
PE</h5>
                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <!-- List Pe -->
            <div id="peListId">
                <div class="list-group">
                </div>
            </div>
            <!-- /List Pe -->
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary"
data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
<!-- /Modal for List Pe -->

<!-- Modal for add CPE-->
<div class="modal fade" id="addCPE" tabindex="-1" role="dialog"
aria-labelledby="exampleModalCenterTitle"
     aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered"
role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title"
id="CPEModalLongTitle">Добавить CPE</h5>
```

```
                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <!-- Form for data CPE-->
                <form name="addCpeForSend">
                    <div class="form-group">
                        <label for="inputCpeIp">CPE IP
address</label>
                        <input type="text" class="form-control"
id="inputCpeIp" name="inputCpeIp"
                               placeholder="192.168.1.1">
                    </div>
                    <div class="form-group">
                        <label for="inputPeIpAddressForCpe">PE
IP address</label>
                        <input type="text" class="form-control"
id="inputPeIpAddressForCpe" placeholder="192.168.1.1">
                    </div>

                    <div class="form-group row">
                        <div class="col-sm-10">
                            <div class="form-check">
                                <label class="form-check-label"
for="InternetActive">
                                    <input class="form-check-
input" type="checkbox" id="InternetActive">
                                    Включить интернет
                                </label>
                            </div>
                        </div>
                    </div>
                    <button type="submit" class="btn btn-
primary" onclick="sendCpe('<%=linksCpe.get(1)%>')">Добавить
                    </button>
                    <button type="button" class="btn btn-
secondary" data-dismiss="modal">Отмена</button>
                </form>
                <!-- /Form for data CPE-->
            </div>
        </div>
    </div>
</div>
<!-- /Modal for add CPE-->

<!-- Modal for add PE-->
```

```
<div class="modal fade" id="addPE" tabindex="-1" role="dialog"
aria-labelledby="exampleModalCenterTitle"
     aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered"
role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title"
id="PEModalLongTitle">Добавить PE</h5>
                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <!-- Form for data PE-->
                <form>
                    <div class="form-group">
                        <label for="inputPeIp">PE IP
address</label>
                        <input type="text" class="form-control"
id="inputPeIp" placeholder="192.168.1.1">
                    </div>
                    <button  type="submit" class="btn btn-
primary" onclick="sendPe('<%=linksPe.get(1)%>')">Добавить
                    </button>
                    <span style="visibility: hidden">.</span>
                    <button  type="button" class="btn btn-
secondary" data-dismiss="modal">Отмена</button>
                </form>
                <!-- /Form for data PE-->
            </div>
        </div>
    </div>
</div>
<!-- /Modal for add PE-->

<!-- MAP -->
<div id="container">
    <svg width="1920px" height="1080px" id="svgId" style="z-
index: 5; pointer-events: none;"></svg>
    <div id="refreshPeAndCpe" style="z-index: 10"></div>
</div>
<!-- /MAP -->
<script>
    setInterval('refresh1()', 3000);
    setInterval('refresh2()', 3000);
    setInterval('refresh3()', 2000);
    setInterval('lines()', 200);
```

```
</script><!-- call refresh.js-->
<script>
    var container = document.querySelector("#container");
    var activeItem = null;
    var active = false;

    container.addEventListener("touchstart", dragStart, false);
    container.addEventListener("touchend", dragEnd, false);
    container.addEventListener("touchmove", drag, false);

    container.addEventListener("mousedown", dragStart, false);
    container.addEventListener("mouseup", dragEnd, false);
    container.addEventListener("mousemove", drag, false);

    function dragStart(e) {
        if (e.target !== e.currentTarget) {
            active = true;
            activeItem = e.target;
            if (activeItem !== null) {
                if (!activeItem.xOffset) {
                    activeItem.xOffset = 0;
                }
                if (!activeItem.yOffset) {
                    activeItem.yOffset = 0;
                }
                if (e.type === "touchstart") {
                    activeItem.initialX = e.touches[0].clientX -
activeItem.xOffset;
                    activeItem.initialY = e.touches[0].clientY -
activeItem.yOffset;
                } else {
                    console.log("doing something!");
                    activeItem.initialX = e.clientX -
activeItem.xOffset;
                    activeItem.initialY = e.clientY -
activeItem.yOffset;
                }
            }
        }
    }

    function dragEnd(e) {
        if (activeItem !== null) {
            activeItem.initialX = activeItem.currentX;
            activeItem.initialY = activeItem.currentY;

            var coords = activeItem.getBoundingClientRect();
            var strElementId = activeItem.id;
            refreshXY(strElementId, (coords.top), coords.left);
```

```
        }

        active = false;
        activeItem = null;
    }

    function drag(e) {
        if (active) {
            if (e.type === "touchmove") {
                e.preventDefault();
                activeItem.currentX = e.touches[0].clientX -
activeItem.initialX;
                activeItem.currentY = e.touches[0].clientY -
activeItem.initialY;
            } else {
                activeItem.currentX = e.clientX -
activeItem.initialX;
                activeItem.currentY = e.clientY -
activeItem.initialY;
            }
            activeItem.xOffset = activeItem.currentX;
            activeItem.yOffset = activeItem.currentY;
            setTranslate(activeItem.currentX,
activeItem.currentY, activeItem);
        }
    }

    function setTranslate(xPos, yPos, el) {
        el.style.transform = "translate3d(" + xPos + "px, " +
yPos + "px, 0)";
    }
</script><script src="https://code.jquery.com/jquery-
3.3.1.slim.min.js"
        integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
"crossorigin="anonymous"></script><script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd
/popper.min.js"integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1
"crossorigin="anonymous"></script><script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/boots
trap.min.js"
        integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM
"
        crossorigin="anonymous"></script>
</body>
</html>
```