ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения заочная

«К ЗАЩИТЕ ДОПУЩЕН(А)»
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 20___

Выпускная квалификационная работа

Обучающегося Попова Вадима Игоревича
(фамилия, имя, отчество)
Вид работы выпускная квалификационная работа бакалавра
(выпускная квалификационная работа бакалавра, специалиста, магистра)

Пояснительная записка

Тема Разработка приложения тестирования доступности конечных терминалов (на примере МО РФ в/ч 28916)
(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

Руководитель работы к. ф.-м. н., доцент Черняева С. Н.
(должность, подпись, фамилия, инициалы, дата)
Консультант _____
(при наличии) (должность, подпись, фамилия, инициалы, дата)
Консультант _____
(должность, подпись, фамилия, инициалы, дата)
Обучающийся Попов В. И.
(подпись, фамилия, инициалы, дата)

Воронеж
2024

ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения заочная

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 2024

Задание
на выпускную квалификационную работу

Вид работы ВКР бакалавра
(ВКР бакалавра, ВКР специалиста, ВКР магистра)

Обучающемуся Попову Вадиму Игоревичу
(фамилия, имя, отчество)

Тема Разработка приложения тестирования доступности конечных терминалов (на примере МО РФ в/ч 28916)

Утверждена приказом ректора университета от «27» апреля 2024г., № 173-Ф

Срок сдачи законченной работы 21 июня 2024г.

Исходные данные (или цель ВКР):
Разработать приложение тестирования доступности конечных терминалов».
Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР):
– Введение. Актуальность выбранной темы, цель и задачи ВКР
(наименование вопроса, раздела и его краткое содержание)
– Исследовательский раздел.
(наименование вопроса, раздела и его краткое содержание)

Общая характеристика организации, описание существующих моделей мониторинга компьютерной сети, анализ существующих разработок и обоснование выбора технологии проектирования, выводы по первому разделу

– Проектный раздел.

(наименование вопроса, раздела и его краткое содержание)

Определение функциональных требований, составление концептуальной и логической схемы, разработка пользовательского интерфейса, реализация системы, расчет показателей экономической эффективности работы

– Заключение. Выводы по работе в целом. Оценка степени решения поставленных задач

(наименование вопроса, раздела и его краткое содержание)

Практические рекомендации

Перечень графического материала (или презентационного материала):

1. Титульный лист
2. Цель и задачи ВКР
3. Обзор предметной области
4. Существующие модели мониторинга компьютерной сети
5. Анализ существующих разработок
6. Концептуальная и логическая схема
7. Разработка пользовательского интерфейса
8. Реализация системы
9. Расчет показателей экономической эффективности работы
10. Результаты ВКР

Консультанты по разделам ВКР (при наличии):

1. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

2. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

3. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

Дата выдачи задания: _____ 20___

Задание согласовано и принято к исполнению: _____ 20___

Руководитель ВКР: к. ф.-м. н., доцент Черняева С. Н. _____

(должность, ученая степень, ученое звание, ФИО) (подпись)

Обучающийся: Попов В. И. _____

(учебная группа, ФИО) (подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 7
1.1 Общая характеристика организации 7
1.2 Описание существующих моделей мониторинга компьютерной сети 11
1.3 Анализ существующих разработок и обоснование выбора технологии проектирования 17
1.4 Выводы по первому разделу 34
2 ПРОЕКТНЫЙ РАЗДЕЛ 36
2.1 Определение функциональных требований 36
2.2 Составление концептуальной и логической схемы 37
2.3 Разработка пользовательского интерфейса 38
2.4 Реализация системы 41
2.5 Расчет показателей экономической эффективности работы 44
ЗАКЛЮЧЕНИЕ 47
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 49
ПРИЛОЖЕНИЕ А 52
ПРИЛОЖЕНИЕ Б 53
ПРИЛОЖЕНИЕ В 54

ВВЕДЕНИЕ

Система мониторинга в сети передачи данных имеет большое значение для обеспечения безопасности и управления сетями связи. Мониторинг в сети передачи данных включает в себя отслеживание событий в сети передачи данных для обеспечения безопасности и стабильной работы сетей связи. Однако не все инструменты соответствуют требованиям организаций, так как они могут представлять угрозу утечки конфиденциальных данных или создавать высокую нагрузку на оборудование и сеть передачи данных. В некоторых компаниях запрещено использование программного обеспечения без должной проверки или несоответствующего внутренним правилам, так как могут содержать в себе скрытые модули или просто будут осуществлять избыточную нагрузку на сеть передачи данных и оборудование. А некоторые программы попросту слишком сложны своим интерфейсом, что затрудняет их настройку и сам процесс работы с ними.

В рамках ВКР предлагается разработать приложение для мониторинга в локально вычислительной сети, которое будет иметь простой минималистичный графический интерфейс, который будет интуитивно прост в использовании и предоставит необходимые функции для мониторинга конечных терминалов и промежуточных узлов в сети передачи данных, а также будет соответствовать установленным требованиям безопасности. Таким образом, преимуществом этого приложения будет возможность его применения в любых сетях, так как оно будет имеет открытый исходный код, простой графический интерфейс, не будет создавать значительной нагрузки на сеть и оборудование и позволит осуществлять контроль доступности конечных терминалов, даже если их в сети большое количество.

Цель ВКР заключается в проектировании и реализации приложения для мониторинга доступности конечных терминалов и промежуточных узлов в локально-вычислительной сети организации.

Для достижения поставленной цели необходимо решить следующие задачи:

– описать объект исследования;

– описать существующие модели мониторинга компьютерной сети;

– проанализировать существующие разработки и обосновать выбор технологии проектирования;

– спроектировать приложение для мониторинга сети;

– осуществить концептуальное проектирование системы локальной вычислительной сети организации;

– выполнить логическое моделирование приложения для мониторинга сети;

– построить физическую модель вычислительной сети организации;

– реализовать приложения для мониторинга вычислительной сети организации;

– рассчитать показатели экономической эффективности работы.

Объект исследования – МО РФ в/ч 28916.

Предмет исследования – организация процесса мониторинга в локально-вычислительной сети.

Практическая значимость ВКР заключается в возможности отслеживать доступность большого количества конечных терминалов и промежуточных узлов в режиме реального времени.

Результатом ВКР будет разработка и реализация приложения для мониторинга конечных терминалов и промежуточных узлов в локально-вычислительной сети организации, которое будет соответствовать установленным требованиям безопасности, будет иметь простой графический интерфейс и позволит единовременно осуществлять контроль большого количества терминалов в режиме реального времени.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Общая характеристика организации

МО РФ в/ч 28916 (далее – организация) была сформирована 25.12.2009г.

Адрес регистрирующего органа – 188643, Ленинградская обл., Всеволожск г. Колтушское ш,138а.

В/ч 28916 является организацией Министерства обороны Российской Федерации, входит в состав Центрального военного округа, содержится в численности Центрального военного округа Вооруженных Сил Российской Федерации и подчиняется командующему войсками Центрального военного округа.

Войсковая часть осуществляет операции по расходованию бюджетных средств в соответствии с бюджетной сметой, утвержденной в установленном порядке распорядителем бюджетных средств.

Войсковая часть, соответственно, подлежит постановке на налоговый учёт в установленном порядке по месту его постоянной дислокации, а в случае его нахождения за пределами Российской Федерации – по месту дислокации Учреждения.

Войсковая часть в своей деятельности руководствуется Конституцией Российской Федерации, федеральными конституционными законами, федеральными законами, актами Президента Российской Федерации, в том числе издаваемыми им как Верховным Главнокомандующим Вооруженными

Силами Российской Федерации, актами Правительства Российской Федерации, международными договорами Российской Федерации, правовыми актами Министерства обороны Российской Федерации, Учреждения, а также Положением.

Организационная структура войсковой части 28916 определяется штатом, утвержденным в порядке, которое установило Министерство обороны Российской Федерации.

Организация осуществляет в соответствии с законодательством

Российской Федерации и в порядке, установленном в Министерстве обороны Российской Федерации, следующие виды деятельности:

1) обеспечение организации боевого дежурства, боевой и мобилизационной подготовки, проведения учений, оперативных тренировок и иных мероприятий в соответствии с утвержденными планами и в порядке, установленном в Министерстве обороны Российской Федерации;

2) обеспечение функционирования закрепленных и предоставленных в пользование объектов материально-технической базы;

3) ведение договорной и договорного-претензионной работы;

4) организацию и осуществление всех видов боевого, технического, тылового, финансового, медицинского и морально-психологического обеспечения иных видов деятельности в соответствии со своим функциональным предназначением.

Организация не имеет права осуществлять приносящую доход деятельность, за исключением тех видов деятельности, которые включены в генеральное разрешение Министерства обороны Российской Федерации на осуществление приносящей доход деятельности в соответствии с законодательными и иными нормативными правовыми актами Российской Федерации.

Войсковая часть 28916 выполняет работы, связанные с использованием сведений, составляющих государственную тайну, и обеспечивает защиту указанных сведений в соответствии с законодательством Российской Федерации и в порядке, установленном в Министерстве обороны Российской Федерации.

Организация может в соответствии с законодательством Российской Федерации выступать государственным заказчиком на поставки товаров, выполнение работ, оказание услуг для государственных нужд исключительно на основании доверенности и Положения.

Организационная структура организации определяется штатом, утверждённым в установленном порядке. Структура прописана в Положение.

В структуру входят:

– командование (начальник, заместитель начальника – начальник хранения, инспектор по кадрам, инженер по охране труда и технике безопасности);

– организационно-плановое отделение (начальник отделения, офицер (по обеспечению безопасности информации);

– секретная часть (начальник секретной части, делопроизводитель);

– отделение материально-технического обеспечения (начальник отделения, инженер энергетик, техник);

– производственно-техническое отделение (начальник отделения,

– инженер, техник (по нормированию труда);

– отделение технического контроля (начальник отделения, инженер);

– отделение (операционное и комплектации) (начальник отделения, ведущий экономист, экономист);

– отдел (сборки, регламента и ремонта) (начальник отдела, помощник начальника отдела, инженер, такелажник, водитель автомобиля, сборщик боеприпасов, контролер-приемщик, машинист крана);

– отдел хранения (начальник отдела, помощник начальника отдела, техник, заведующий хранилищем, такелажник, контролер-приемщик, укладчик-упаковщик);

– отдел механизации и перевозок (и инженерных работ) (начальник отдела, машинист крана, водитель автомобиля, машинист тепловоза, составитель поездов, машинист бульдозера, водитель боевых и специальных машин);

– отделение (деревообделывающее) (начальник отделения, техник, станочник деревообрабатывающих станков);

– энергомеханическое отделение (начальник отделения, электрогазосварщик, токарь);

– взвод обеспечения (командир взвода, заместитель командира взвода – командир отделения, командир отделения, старший водитель, старший мастер, аккумуляторщик, водитель, водитель-крановщик, мастер);

– технический взвод (командир взвода, заместитель командира взвода - командир отделения, старший водитель, старший лаборант-такелажник, старший электрокарщик-такелажник, лаборант-такелажник, электрокарщик-такелажник, водитель-такелажник);

– контрольно-технический пункт (техник (по безопасности движения-начальник КТП);

– пожарная команда (начальник команды, командир отделения, пожарный, водитель автомобиля, водитель боевых и специальных машин);

– склады (заведующий складом);

– медицинский пункт (начальник медпункта, врач-терапевт, фельдшер, медицинская сестра);

– отряд военизированной охраны (начальник отряда, заместитель начальника отряда, старшина).

Таким образом, делопроизводитель относится к отделу «секретная часть» в которую так же входит начальник секретной части. Начальник секретной части является руководителем, которому непосредственно подчиняется делопроизводитель. Наглядную схему организационной структуры можно увидеть на рисунке 1.1.

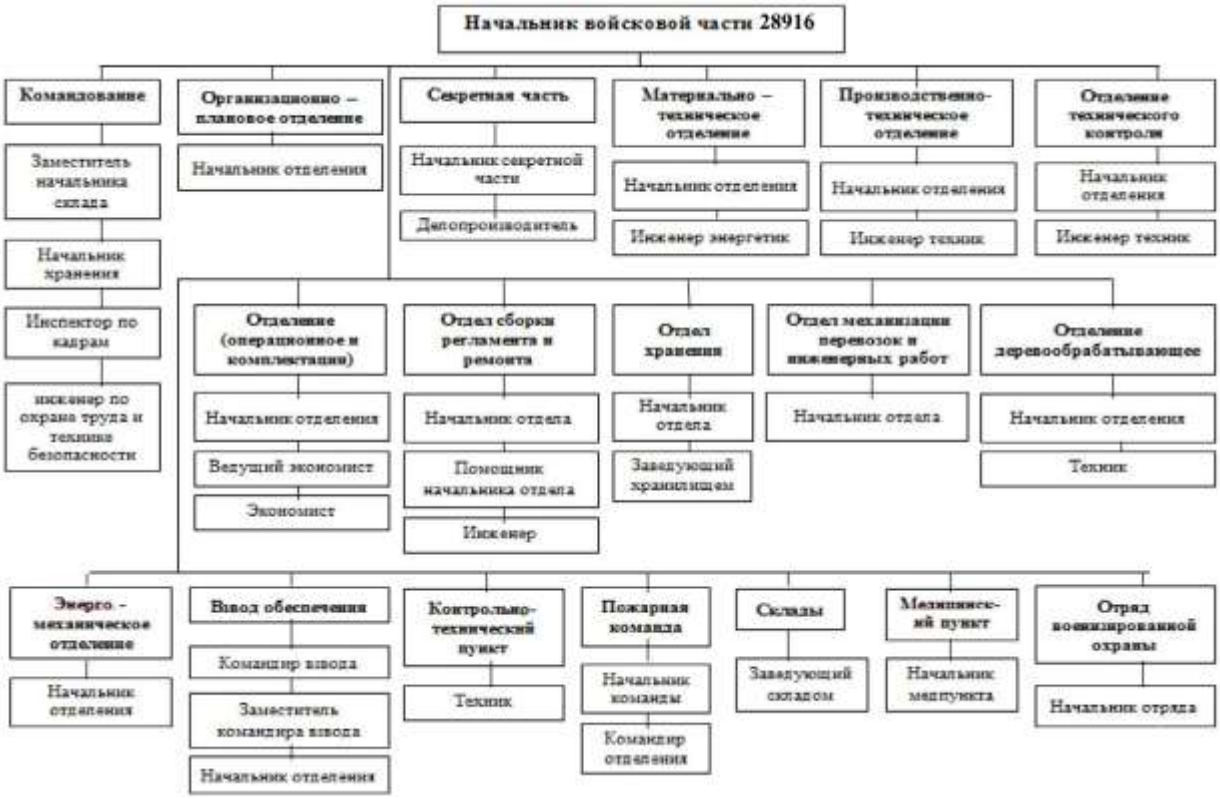


Рисунок 1.1 - Организационная структура войсковой части

Войсковая часть в своей деятельности руководствуется Конституцией Российской Федерации, федеральными законами, законами, Указами Президента Российской Федерации, Постановлениями Правительства Российской Федерации, приказами и директивами Министра обороны Российской Федерации, начальника Генерального штаба Вооруженных Сил Российской Федерации, указаниями заместителей Министра обороны Российской Федерации, приказами и директивами командующего войсками Центрального военного округа и его заместителя по материально-техническому обеспечению, а также настоящим Положением.

1.2 Описание существующих моделей мониторинга компьютерной сети

Методы мониторинга, основанные на маршрутизаторе – жёстко заданы (вшиты) в маршрутизаторах и, следовательно, имеют низкую гибкость. Краткое описание наиболее часто используемых методов такого мониторинга приведены ниже. Каждый метод развивался много лет, прежде чем стать стандартизованным способом мониторинга [12].

Протокол простого сетевого мониторинга (SNMP), определённый в RFC 1157. SNMP – это протокол прикладного уровня, часть стека протоколов TCP/IP. Он позволяет администраторам контролировать производительность сети, выявлять и устранять сетевые проблемы, а также планировать развитие сети. SNMP собирает статистику трафика до конечного хоста через пассивные датчики, работающие вместе с маршрутизаторами. Хотя существуют две версии SNMP (SNMPv1 и SNMPv2), в данном контексте рассматривается только SNMPv1. SNMPv2 основан на SNMPv1 и предлагает улучшения, такие как поддержка дополнительных операций с протоколами. Ожидается стандартизация ещё одного варианта версии SNMP – SNMPv3. Для протокола SNMP присущи три ключевых компонента:

- управляемые устройства (Managed Devices),
- агенты (Agents),
- системы управления сетью (Network Management Systems – NMSs).

Они показаны на рисунке 1.2.

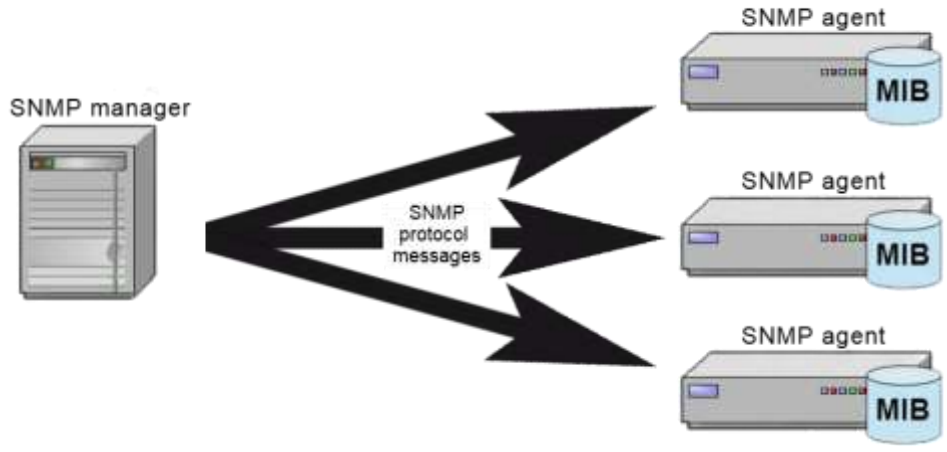


Рисунок 1.2 – Компоненты протокола простого сетевого мониторинга

Удалённый мониторинг (RMON), RFC 1757. RMON включает в себя различные сетевые мониторы и консольные системы для изменения данных, полученных в ходе мониторинга сети [20]. Это расширение для SNMP информационной базы данных по управлению (MIB). В отличии от SNMP, который должен посылать запросы о предоставлении информации, RMON

может настраивать сигналы, которые будут «мониторить» сеть, основанную на определённом критерии. RMON предоставляет администраторам возможности управлять локальными сетями также хорошо, как удалёнными от одной определённой локации/точки. Его мониторы для сетевого уровня приведены ниже. RMON имеет две версии RMON и RMON2. Однако в данной статье говорится только о RMON. RMON2 позволяет проводить мониторинг на всех сетевых уровнях. Он фокусируется на IP-трафике и трафике прикладного уровня.

Хотя существует три ключевых компонента мониторинговой среды RMON, здесь приводятся только два из них. Они показаны на рисунке 1.3.

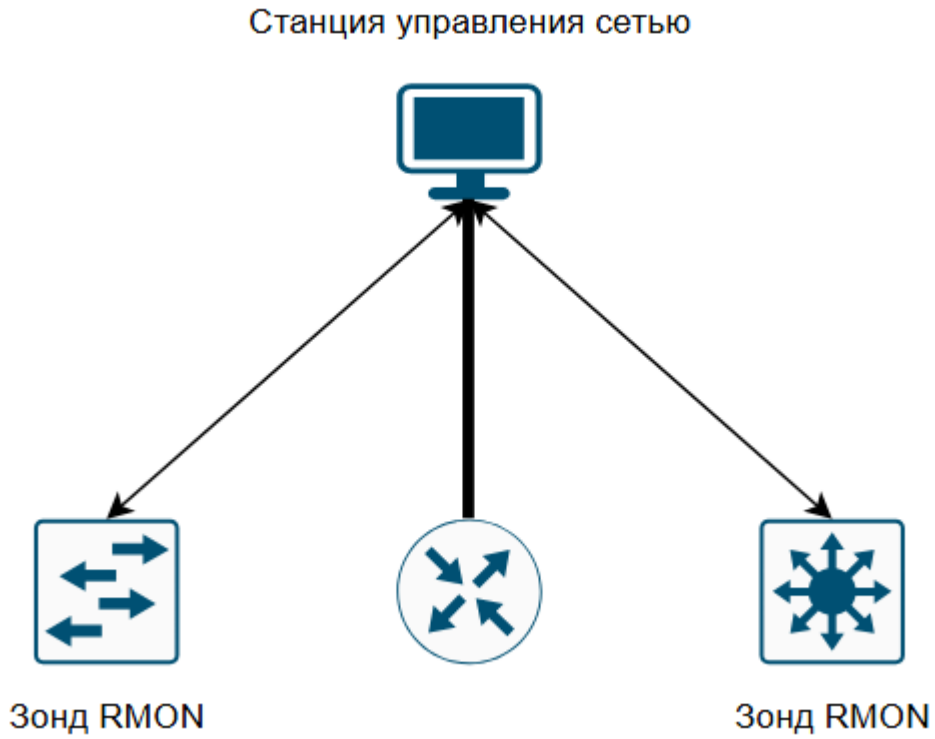


Рисунок 1.3 – Компоненты удаленного мониторинга

Два компонента RMON это датчик, также известный как агент или монитор, и клиент, также известный как управляющая станция (станция управления). В отличии от SNMP датчик или агент RMON собирает и хранит сетевую информацию. Датчик – это встроенное в сетевое устройство (например, маршрутизатор или переключатель) программное обеспечение. Датчик может запускаться также и на персональном компьютере. Датчик

должен помещаться для каждого различного сегмента локальной или глобальной сети, так как они способны видеть трафик, который проходит только через их каналы, но они не знают о трафике за их приделами. Клиент – это обычно управляющая станция, которая связана с датчиком, использующим SNMP для получения и коррекции RMON-данных [13].

RMON использует 9 различных групп мониторинга для получения информации сети:

– Statistics – статистика, измеренная датчиком для каждого интерфейса мониторинга для данного устройства,

– History – учёт периодических статистических выборок из сети и хранение их для поиска,

– Alarm – периодически берёт статистические образцы и сравнивает их с набором пороговых значений для генерации события,

– Host – содержит статистические данные, связанные с каждым хостом, обнаруженным в сети,

– HostTopN – готовит таблицы, которые описывают вершину хостов

– (главный хост),

– Filters – включает фильтрацию пакетов, основываясь на фильтровом уравнении для захвата событий,

– Packet capture – захват пакетов после их прохождения через канал,

– Events – контроль генерации и регистрация событий от устройства,

– Token ring – поддержка кольцевых лексем.

Как уже упоминалось ранее, RMON основан на протоколе SNMP. Несмотря на возможность мониторинга трафика с помощью этого метода, аналитические данные, полученные с использованием SNMP и RMON, имеют низкую производительность.

Утилита Netflow успешно работает с разными аналитическими программами, упрощая работу администратора. Она соответствует стандарту

RFC 3954. Netflow – это расширение, представленное в маршрутизаторах Cisco, позволяющее собирать IP-трафик, если это настроено в интерфейсе [24].

Анализируя данные Netflow, администратор определяет источник и получателя трафика, класс сервиса и причины перегрузок. Netflow состоит из трёх компонентов: кеширование потока, сбор информации о потоках и анализ данных. Рисунок 1.4 показывает инфраструктуру Netflow. Каждый компонент, показанный на рисунке, объясняется ниже.

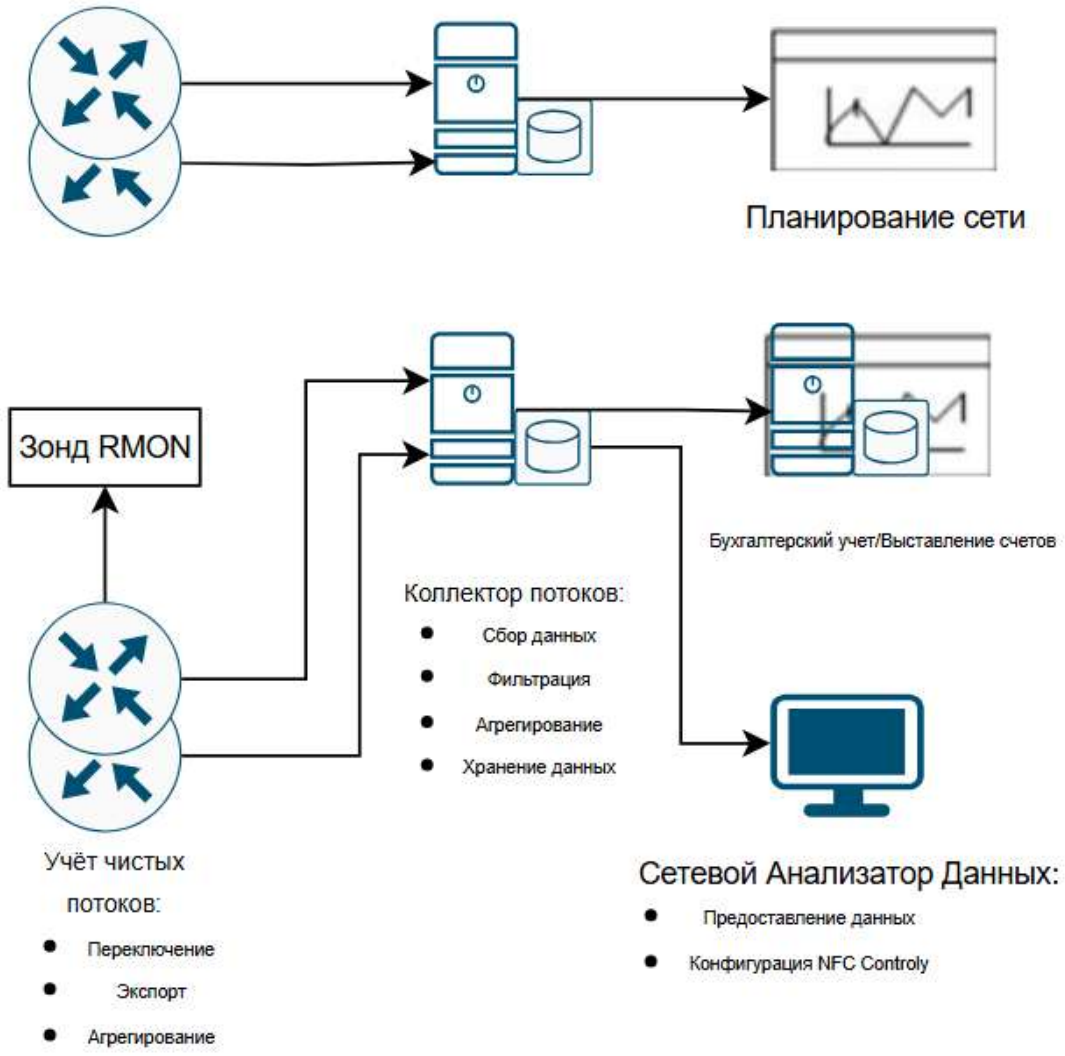


Рисунок 1.4 – Инфраструктура NetFlow

FlowCaching анализирует и собирает данные о IP потоках, которые входят в интерфейс, и преобразует данные для экспорта.

WMI (Windows Management Instrumentation или Инструментарий управления Windows) – это технология для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows представлено на рисунок 1.5.

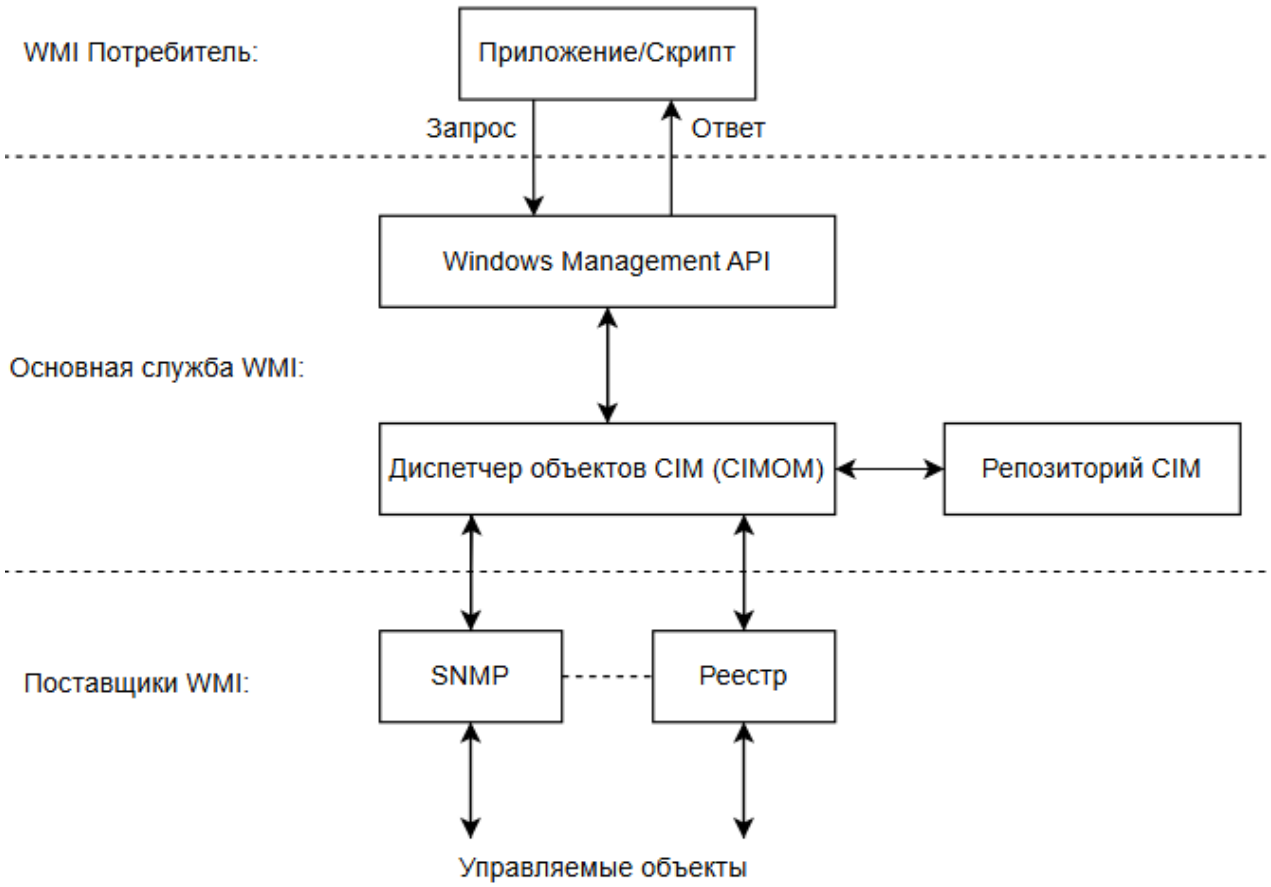


Рисунок 1.5 – Архитектура инструментария управления Windows

Технология WMI – это расширенная и адаптированная под Windows реализация стандарта WBEM, которая позволяет создавать универсальный интерфейс мониторинга и управления различными системами и компонентами распределённой информационной среды предприятия. WMI использует объектно-ориентированные идеологии и протоколы HTML и XML. Эта технология является мощным инструментом для администрирования операционных систем семейства Windows с помощью скриптов. С её помощью можно управлять устройствами, учётными записями, сервисами, процессами, сетевыми интерфейсами и другими программами, которые расширяют базовую структуру WMI своими классами.

1.3 Анализ существующих разработок и обоснование выбора технологии проектирования

Novell ZENworks Configuration Management – это комплексное решение для управления IT-системами, которое помогает снизить общую стоимость владения. Интуитивно понятный веб-интерфейс, безопасное дистанционное управление и делегированное администрирование позволяют техническим специалистам быстро оказывать поддержку пользователям, представлено на рисунке 1.6.

Рисунок 1.6 – Окно программы Novell ZENworks Configuration Management

Основные функции:

– централизованная установка и обновление ПО и ОС,

– инвентаризация программного и аппаратного обеспечения,

– удаленное управление рабочими станциями и серверами.

Управление осуществляется на основании системных политик, единых в масштабе всей сети. Управление станциями осуществляется из общей консоли.

SolarWinds Network Performance Monitor – система мониторинга сетей и активного сетевого оборудования, представлено на рисунке 1.7.

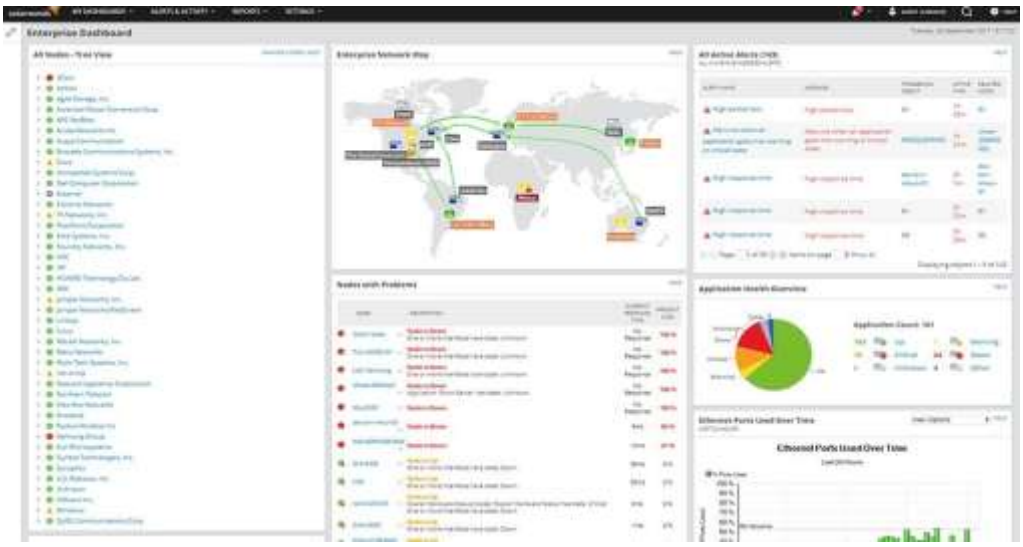


Рисунок 1.7 – Окно программы SolarWinds Network Performance Monitor

SolarWinds Network Performance Monitor – это мощное решение для мониторинга сетевой производительности в режиме реального времени. Оно помогает выявлять и диагностировать проблемы быстродействия, обеспечивая бесперебойную работу сети. Благодаря динамическим картам топологии сети и автоматическому обнаружению сетевых компонентов, администраторы могут легко масштабировать сеть и адаптировать её к изменениям. Это гарантирует соответствие важных процессов потребностям бизнеса и обеспечивает стабильность работы сети.

Основные функции:

– автоматическое обнаружение сети. Программа позволяет легко планировать процессы автоматического сканирования сети, обнаруживать новые сетевые устройства и осуществлять мониторинг критически важного оборудования,

– поддержка устройств от различных вендоров. Network Performance Monitor поддерживает гетерогенные сети и устройства от ведущих производителей оборудования,

– мониторинг производительности и доступности сети. Программа позволяет отслеживать доступность и индикаторы производительности интерфейса и сетевых устройств, такие как нагрузка на полосу пропускания,

центральный процессор и память, задержки, ответы, потеря пакетов – по каждому оборудованию с поддержкой SNMP и WMI,

– интеллектуальные уведомления. Продукт поддерживает оповещения о событиях, условиях и состояниях сетевых устройств. При необходимости администратор может заблокировать уведомления на основе зависимостей и топологии и получать оповещения только по важным сетевым проблемам,

– быстрое развертывание.

Network Performance Monitor загружается и устанавливается менее чем за час в три простых шага.

Программно-аппаратное решение Dell KACE K1000 Management Appliance представляет собой комплексное устройство для управления сетевыми системами IT-инфраструктуры предприятия. Оно предназначено для компаний с количеством конечных узлов от 100 до 20 тысяч единиц, представлено на рисунке 1.8.

Рисунок 1.8 – Внешний вид комплекса Dell KACE K1000 Management Appliance

Устройство позволяет управлять серверами, настольными компьютерами и ноутбуками с помощью простого пользовательского интерфейса. Благодаря централизованному управлению через единую консоль, можно работать с сетями, созданными на базе оборудования разных производителей, с различной архитектурой, версиями и поколениями.

Основные функции:

– обнаружение и инвентаризация всего оборудования и программного обеспечения во всей сети,

– управление патчами для автоматического анализа уязвимых мест и установки обновлений,

– управление ресурсами для осуществления комплексного отслеживания ресурсов и отчетов о соответствии нормативными требованиями,

– групповые оповещения для уведомлений пользователей о важных событиях, например, таких как перебой в работе службы электронной почты,

– распространение программного обеспечения включает возможности удаленного распределения и установки приложений и цифровых ресурсов,

– удаленное управление позволяет централизованно заниматься устранением неисправностей без необходимости выезда на место.

CiscoWorks LMS – это набор программных решений для упрощения настройки, мониторинга и диагностики сетей, основанных на продуктах Cisco. LMS представляет собой интегрированную систему, которая предоставляет информацию о сетевых устройствах другим информационным системам, автоматизирует рутинные задачи по управлению сетью, собирает и предоставляет данные о нагрузке устройств, а также предлагает инструменты для определения и диагностики сетевых проблем. Все компоненты системы используют общий модуль хранения данных об устройствах, что облегчает

настройку и интеграцию с существующими информационными системами компании, представлено на рисунке 1.9.

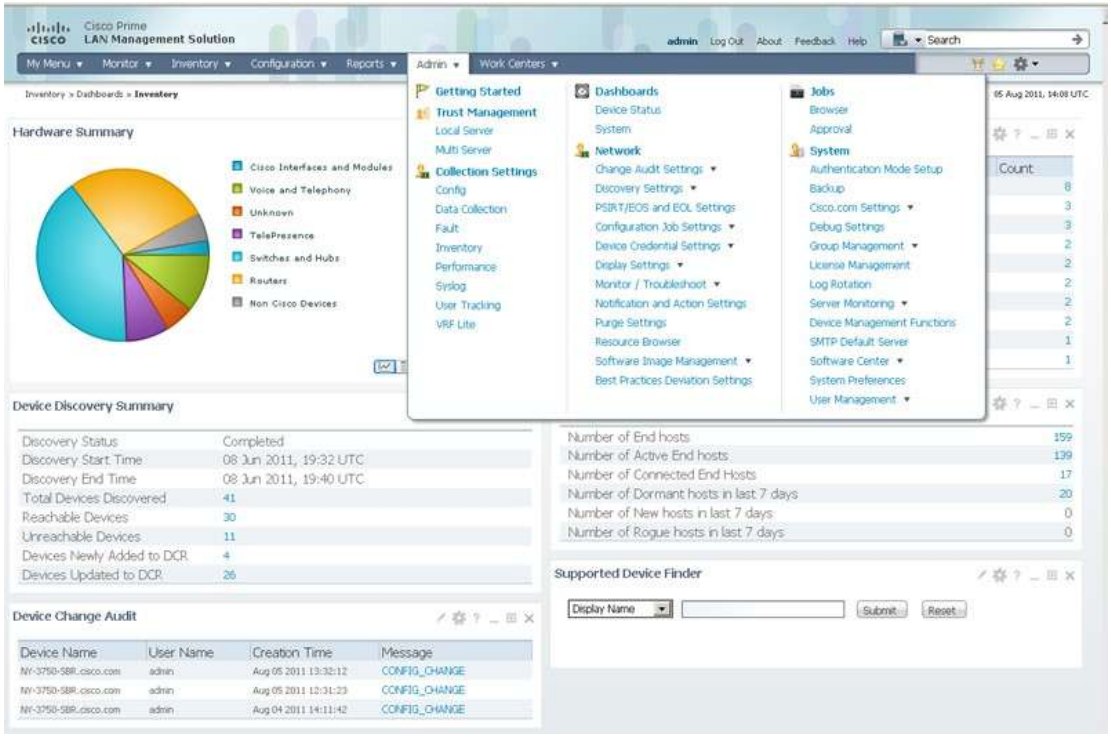


Рисунок 1.9 – Окно программы CiscoWorks

Решение CiscoWorks LMS включает в себя несколько приложений. Пользователи могут установить все приложения или выбрать только те, которые им необходимы. Все приложения поставляются в едином пакете и работают на основе общей лицензии (кроме приложения Health and Utilization Monitor, для которого нужна отдельная лицензия).

Naumen Network Manager – комплексное решение для мониторинга ИТ-инфраструктуры и управления сетями любого масштаба. Naumen Network Manager сканирует сеть, отслеживая состояние физических и виртуальных серверов, рабочих станций, маршрутизаторов, коммутаторов, других устройств, сервисов и приложений. Платформа объединяет несколько сотен компонентов для сетевого управления, позволяющих работать с тревогами, применять оповещения и обработчики событий, создавать отчеты и диаграммы, строить карты сети, выполнять задачи по составленному расписанию и многое другое, представлено на рисунке 1.10.

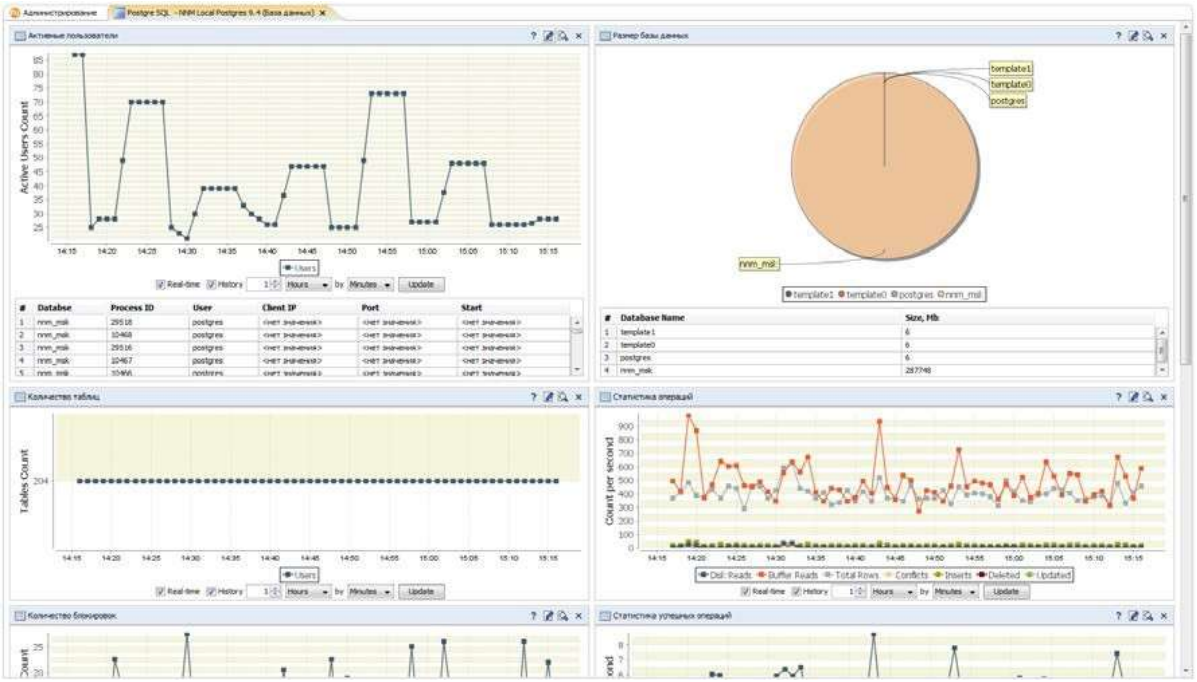


Рисунок 1.10 – Окно программы Naumen Network Manager

Основные возможности:

– мониторинг сетей,

– мониторинг серверов,

– мониторинг приложений и сервисов,

– мониторинг маршрутизаторов,

– мониторинг виртуальной инфраструктуры,

– мониторинг VOIP и проверка IP SLA,

– мониторинг баз данных,

– мониторинг и управление по SNMP,

– управление IT-активами.

Ivanti Endpoint Manager (ранее LANDesk Management Suite) – комплексная система инвентаризации/управления парком ПК и серверов. Решение позволяет ИТ-администраторам получить комплексную инвентарную информацию о компьютерном парке компании, автоматизировать задачи по распространению программного обеспечения и операционных систем, быстро устранять проблемы пользователей при

помощи технологий удаленного подключения, отслеживать программные и аппаратные активы представлено на рисунок 1.11.

Рисунок 1.11 – Окно программы Ivanti Endpoint Manager

Основные функции:

– активное, пассивное и безагентное обнаружение и инвентаризация,

– конфигурация и управление политиками,

– удаленное управление,

– автоматизированная установка ОС и ПО,

– система оповещений о событиях,

– создание отчетов,

– портал самообслуживания,

– сканирование и исправление угроз безопасности (при наличии модуля Endpoint Security for Endpoint Manager),

– оптимизация используемого ПО, удаление неиспользуемых продуктов, оптимизация затрат на закупку лицензий (при наличии модуля Asset Intelligence).

System Center Configuration Manager (SCCM) – это продукт для управления ИТ-инфраструктурой на основе Microsoft Windows и смежных устройств. Он предоставляет возможности для управления обновлениями,

развёртывания программного обеспечения и операционных систем, интеграции с NAP, инвентаризации аппаратного и программного обеспечения, удалённого управления и управления виртуальными и мобильными системами на базе Windows.

В SCCM технологии и функциональные возможности, реализованные ранее в SMS, были значительно переработаны. Среди основных функций можно выделить управление обновлениями, развёртывание ПО и операционных систем, интеграция с NAP, инвентаризация аппаратного и программного обеспечения, удалённое управление, управление виртуальными и мобильными системами на базе Windows.

SCCM помогает обеспечить продуктивную работу пользователей, предоставляя оптимальные возможности для работы в любом месте и на любом устройстве [5].

Благодаря консолидации всех возможностей управления клиентами и функций безопасности в единой инфраструктуре, обеспечиваемой Configuration Manager, организация сможет оптимизировать свою деятельность по управлению ИТ представлено на рисунке 1.12.

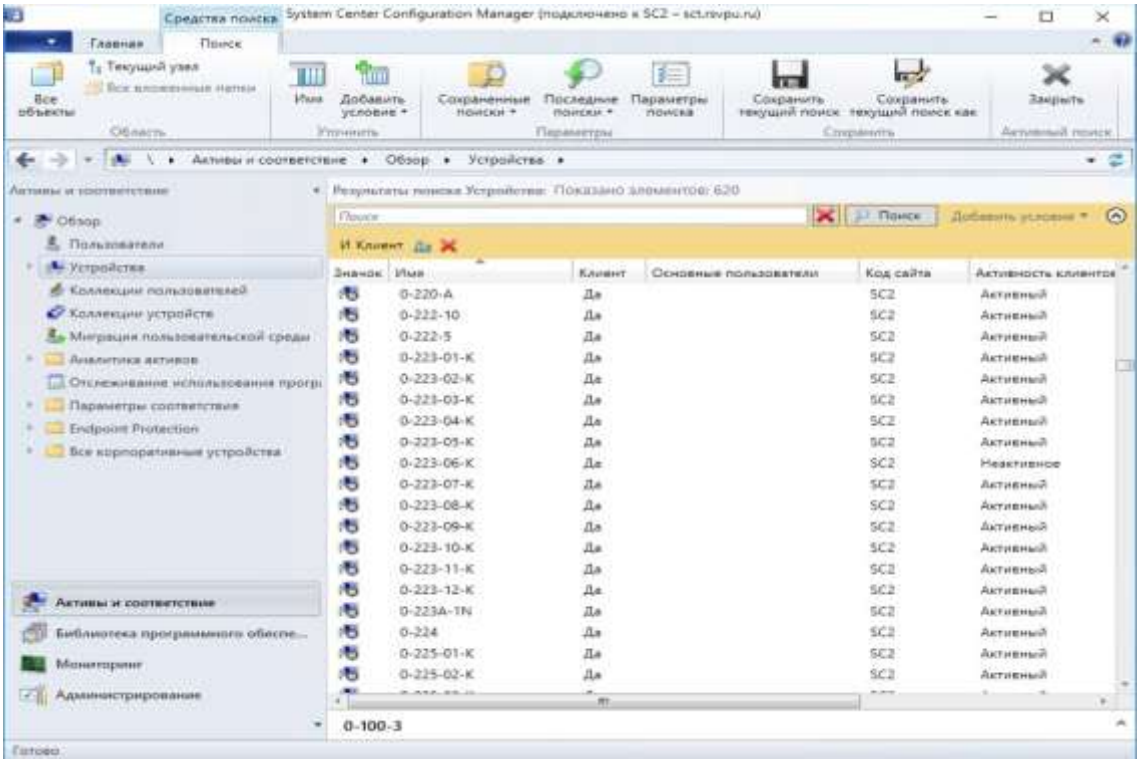


Рисунок 1.12 – Окно программы System Center Configurations Manager Zabbix

Zabbix – это бесплатная система мониторинга, которая отслеживает статусы сервисов компьютерной сети, серверов и сетевого оборудования. Для хранения данных Zabbix использует MySQL, PostgreSQL, SQLite или Oracle [14] представлен на рисунке 1.13. Веб-интерфейс написан на PHP. ZABBIX поддерживает несколько видов мониторинга:

– Simple checks – может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP без установки какого-либо программного обеспечения на наблюдаемом хосте;

– ZABBIX agent – может быть установлен на UNIX-подобных или Windows хостах для получения данных о нагрузке процессора, использования сети, дисковом пространстве и т.д.;

– External check – выполнение внешних программ. ZABBIX также поддерживает мониторинг через SNMP.

Возможности:

– распределенный мониторинг вплоть до 1000 узлов, конфигурация младших узлов полностью контролируется старшими узлами, находящихся на более высоком уровне иерархии,

– сценарии на основе мониторинга,

– автоматическое обнаружение,

– централизованный мониторинг лог-файлов,

– поддержка SNMP ловушек,

– поддержка IPMI,

– поддержка мониторинга JMX приложений из коробки,

– поддержка выполнения запросов в различные базы данных без

– необходимости использования скриптовой обвязки,

– расширение за счет выполнения внешних скриптов,

– возможность создавать карты сетей.

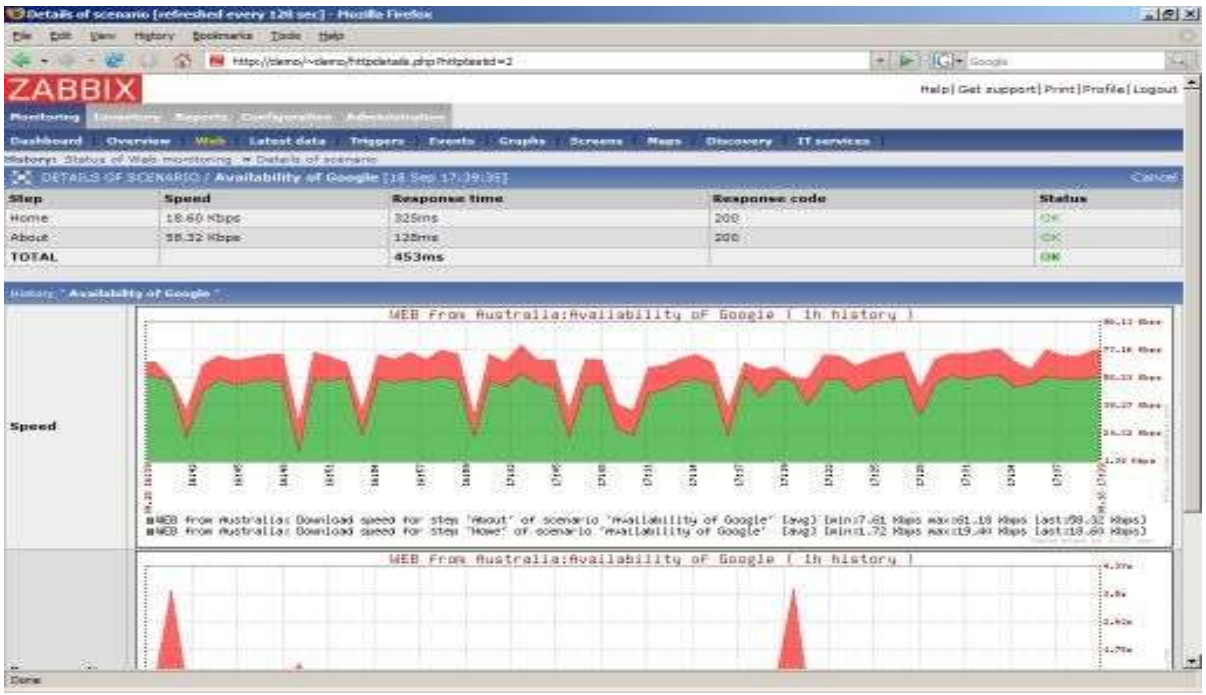


Рисунок 1.13 – Окно программы Zabbix

Friendly Pinger – это бесплатное приложение для администрирования, мониторинга и инвентаризации компьютерных сетей представлено на рисунке 1.14 [13].

Возможности Friendly Pinger:

- визуализация компьютерной сети в красивой анимационной форме,
- отображение, какие компьютеры включены, а какие нет,
- проверка доступности всех устройств за раз,
- оповещение в случае остановки/запуска серверов,
- инвентаризация программного и аппаратного обеспечения всех компьютеров в сети,
- слежение за доступом к Вашему компьютеру и его файлам,
- назначение внешних команд (например, telnet, tracert, net.exe) устройствам,
- поиск HTTP, FTP, e-mail и других сетевых служб,
- отображение состояния сети на рабочем столе или Web странице,
- графический TraceRoute,

– открытие компьютеров в проводнике, в Total Commander'e или в FAR'e,

– функция «Создать дистрибутив» позволяет создать облегченную версию с Вашими картами и настройками [31].

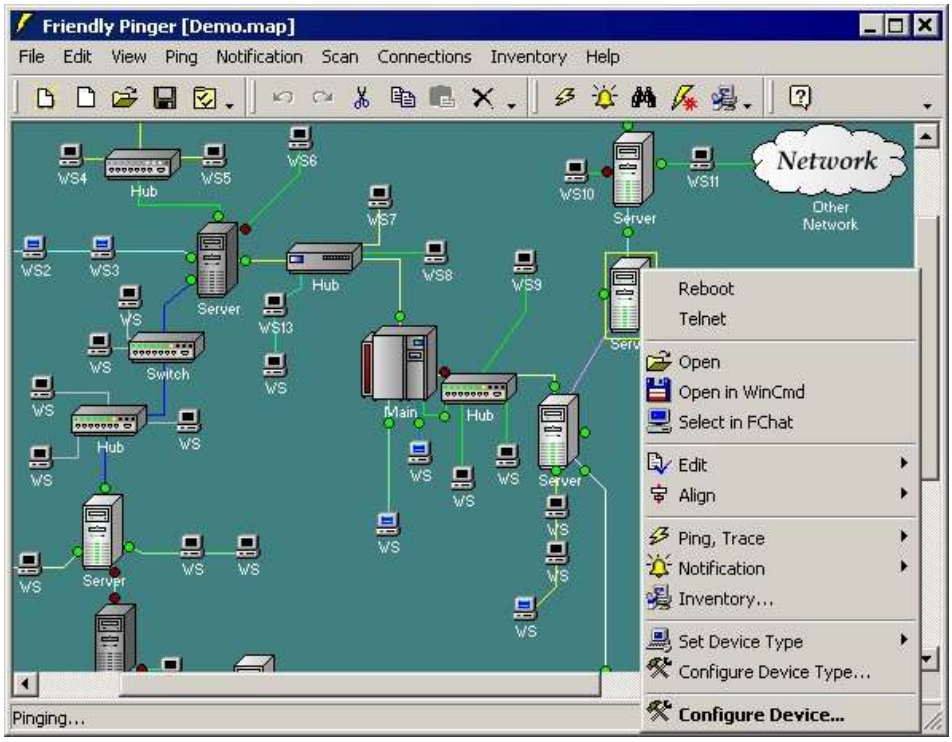


Рисунок 1.14 – Окно программы Friendly Pinger

Friendy Pinger – бесплатное средство мониторинга и инвентаризации небольших локальных сетей с поддержкой русского языка.

Помимо стандартных возможностей подобных программ Friendly Pinger для мониторинга и инвентаризации компьютерных сетей позволяет использовать встроенные команды и передавать им внутренние переменные. Это превращает программу в центр управления локальной вычислительной сетью (ЛВС), позволяя выполнять такие действия, как включение, выключение и перезагрузка удалённых компьютеров, получение доступа к удалённым рабочим столам, командным строкам и сетевым службам., позволяющей легко и быстро выполнить, например, следующие действия:

– включить, выключить или перезагрузить удаленный компьютер,

– получить доступ к удаленному рабочему столу Windows или графической подсистеме Linux,

– получить доступ к удаленной командной строке Windows или Linux,

– выполнить подключения к различным удаленным сетевым службам,

– подключиться к серверу терминалов или серверу SSH,

– принудительно завершить любой процесс на отдельно взятом компьютере или на всех включенных компьютерах [15].

В целом, встроенные команды Friendly Pinger позволяют каждому системному администратору расширить возможности программы на свой вкус, и получить в свое распоряжение удобный инструмент управления локальной сетью [3].

Network Olympus – это многофункциональная система мониторинга сетевых устройств, которая не требует установки агентов. Она взаимодействует с сетевыми администраторами и поддерживает производительность сети и её компонентов представлено на рисунке 1.15.

Конструктор сценариев в Network Olympus позволяет создавать гибкие и сложные схемы мониторинга, учитывающие различные аспекты работы устройств. Это помогает точно определять проблемы и неисправности, а также автоматизировать процесс их устранения.

Network Olympus также предлагает полный мониторинг серверов через протокол WMI, запись и сравнение значимых параметров с заданными значениями. При отклонении параметров от заданных значений программа автоматически исправляет ситуацию с помощью запуска скриптов, приложений или перезагрузки затронутых серверов.

Программа также предоставляет датчики для получения информации о сетевых службах и параметрах их работы, а также карту сети для визуализации инфраструктуры. Network Olympus непрерывно собирает и записывает информацию о сети, следит за производительностью, проверяет доступность служб и наблюдает за состоянием трафика.

Так же имеется возможность получать доступ к журналам проверок, просматривать статистические данные о производительности и времени работы сети и её компонентов.

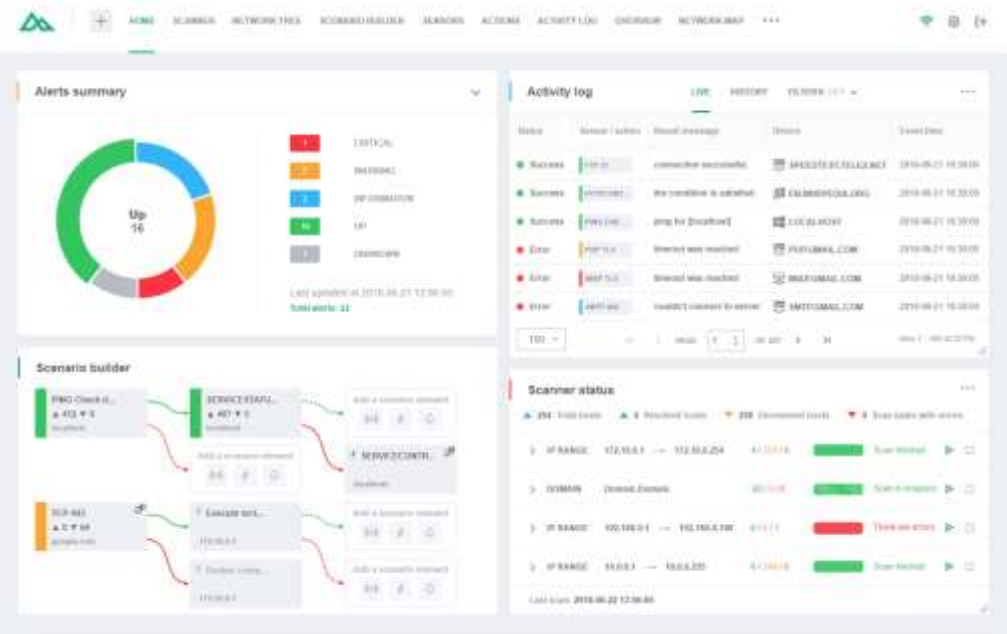


Рисунок 1.15 – Окно программы Network Olympus

Преимущества данной программы:

– Групповые сенсоры
– Простота настройки
– Несложно освоить
– Конструктор сценариев

Недостатки данной программы:

– Только веб-интерфейс
– Установка только под Windows
– Нет многопользовательского доступа

PRTG Network Monitor – это программный компонент для мониторинга сетей, совместимый с устройствами на базе ОС Windows. PRTG помогает обнаруживать сетевые атаки и предоставляет ряд полезных сетевых сервисов, таких как инспекция пакетов, анализ статистических данных, просмотр карты сети в реальном времени и сбор информации об устройствах и уровне нагрузки на сетевое оборудование.

Среди самых полезных сетевых сервисов PRTG: инспекция пакетов, анализ и сохранение статистических данных в базу, просмотр карты сети в режиме реального времени (также доступна возможность получения

исторических сведений о поведении сети), сбор технических параметров об устройствах, подключенных к сети, а также анализ уровня нагрузки на сетевое оборудование. Заметим, что он очень удобен в использовании – прежде всего, благодаря интуитивно понятному графическому интерфейсу, который открывается при помощи любого браузера. В случае необходимости, системный администратор может получить и удаленный доступ к приложению, через веб-сервер. На рисунке 1.16 показан интерфейс программы PRTG.

Рисунок 1.16 – Программный интерфейс программы PRTG Network Monitor

Преимущества данной программы:

– множество функций;

– настраиваемые панели;

– гибкий мониторинг;

– карта сети.

Недостатки данной программы:

– высокая цена;

– нет отдельных БД;

– нет групповых датчиков.

WireShark – это бесплатный анализатор трафика с открытым исходным кодом, предоставляющий пользователям продвинутые функции и признанный

образцовым решением в области сетевой диагностики. Он отлично интегрируется с операционными системами UNIX, Windows и macOS.

Вместо не слишком хорошо понятных для новичков веб-интерфейсов и CLI, в которых нужно вводить запросы на специальном программном языке, WireShark использует графический пользовательский интерфейс (GUI). Однако если вам потребуется расширить стандартные возможности WireShark, вы можете легко написать программы на языке Lua.

Развернув и настроив его один раз на своём сервере, вы получите централизованный инструмент для мониторинга мельчайших изменений в работе сети и сетевых протоколах. Это позволит вам выявлять и идентифицировать сетевые проблемы на ранней стадии. На рисунке 1.17 показан программный интерфейс данной программы.

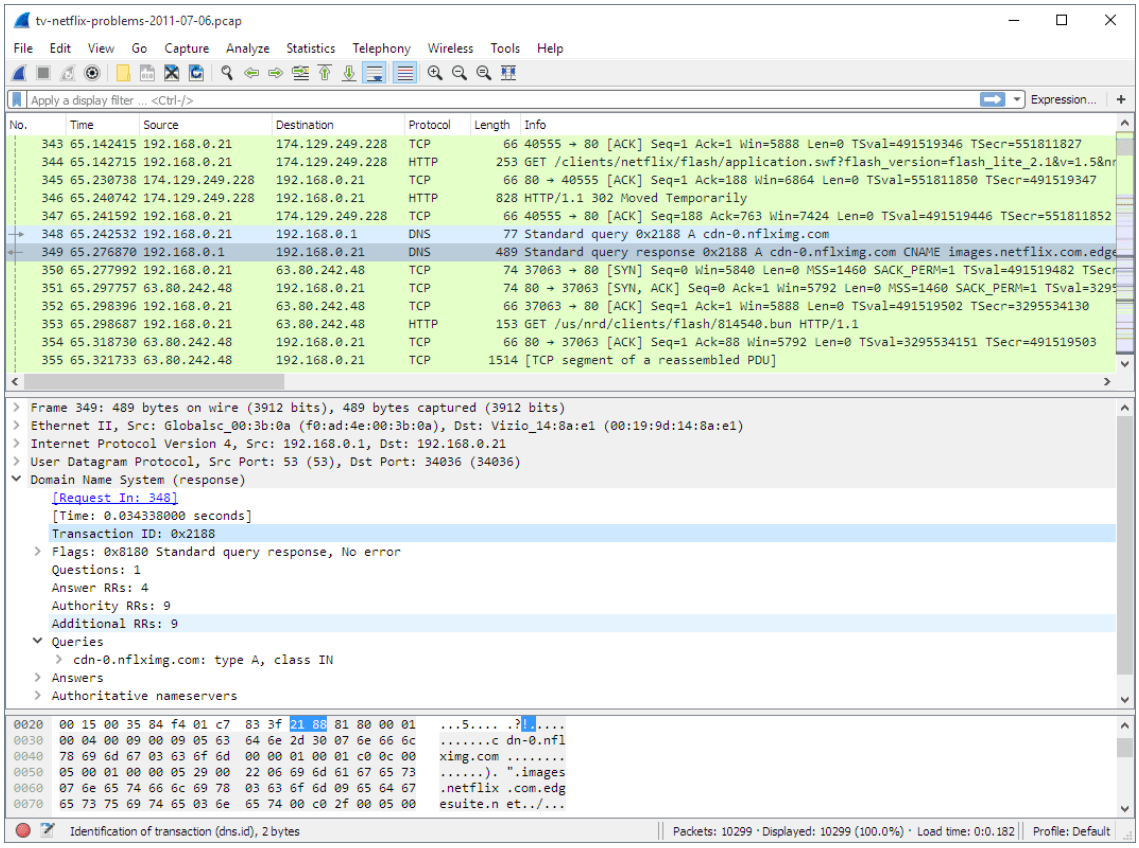


Рисунок 1.17 – Программный интерфейс программы WireShark

Преимущества данной программы:

– бесплатное ПО;

– простота в установке и настройке;

– анализатор пакетов IP;

– гибкий интерфейс.

Недостатки данной программы:

– не интуитивна;

– нет мобильной поддержки;

– не для крупных компаний.

Nagios – известный бесплатный инструмент, разработанный в 1996 году в США для мониторинга, над созданием которого работали с целью сделать его максимально удобным для мониторинга. Его ядро является самой важной частью инструмента, а поверх ядра можно создавать плагины для мониторинга определенных элементов.

Его широкое применение обусловлено тем, что он первым охватил незаменимые функции в сетевом мониторинге.

Nagios – один из самых популярных инструментов для мониторинга сетей и серверов. Он был создан в 1996 году в США и с тех пор постоянно развивается и совершенствуется.

Он состоит из двух основных компонентов: Nagios-сервера и Nagios-клиента. Первый отвечает за сбор информации о состоянии сетевых устройств и серверов, а второй – за отображение этой информации пользователю. Данный инструмент поддерживает различные типы мониторинга, включая мониторинг доступности, производительности, использования ресурсов и событий. Он также может использоваться для оповещения администраторов о проблемах и событиях, требующих их внимания. Nagios имеет открытый исходный код и распространяется бесплатно. Он доступен для множества операционных систем, включая Linux, Windows и macOS. По этой причине Nagios был очень популярен. Более того, благодаря своему высокому первоначальному проникновению на рынок он до сих пор широко используется. На рисунке 1.18 представлен его программный интерфейс.

Рисунок 1.18 – программный интерфейс Nagios

Преимущества данной программы:

– есть много профессионалов с опытом работы с Nagios;

– если вы хорошо знакомы с инструментом, ручная настройка может дать вам много возможностей, когда речь идет о мониторинге изолированных и конкретных случаев;

– он предлагает множество плагинов для адаптации Nagios к потребностям пользователя;

– базовая конфигурация очень проста в использовании.

Недостатки данной программы:

– сложная настройка и редактирование из-за необходимости вносить изменения вручную, чтобы инструмент был готов к работе;

– графический интерфейс не отличается удобством;

– высокая стоимость обучения;

– в итоге, каждая установка – это «паззл», в котором вместо стандартного продукта мы имеем собственную его версию, с сотнями патчей, собственным или сторонним кодом и сложным для развития или поддержки сторонними компаниями;

– неполные отчеты;

– очень плохая работа с SNMP, как опрос, так и управление ловушками.

1.4 Выводы по первому разделу

Преимущества внедрения системы управления ИТ-инфраструктурой:

– повышение доступности и надежности критически важных IT-ресурсов,

– повышение эффективности работы сотрудников, за счет снижения временных затрат IT-персонала на рутинные операции,

– возможность оперативного анализа и контроля оборудования организации,

– сокращение времени ввода нового оборудования,

– снижение временных затрат на обновление, развертывание и настройку ПО.

Централизованный контроль устройств позволяет повысить безопасность сети, сократить время простоя систем и оборудования, удовлетворить требования, связанные с условиями лицензионных соглашений на ПО.

Программные продукты такие как: Friendly Pinger, 10-Strike LANState – представляют собой простые инструменты для мониторинга сети и инвентаризации аппаратного обеспечения. Но функционал данных инструментов весьма ограничен и не позволяет в полной мере управлять всей инфраструктурой организации.

Система Zabbix является универсальной системой мониторинга корпоративной сети. Однако в данном продукте отсутствует возможность удаленного подключения к рабочим станциям и серверам. Инвентаризационные данные аппаратного обеспечения, содержат лишь основную информацию о системе: загрузка процессора, использование

дисков, использование памяти, информация о входящем/исходящем трафике, а также состояние основных служб Windows. Функция инвентаризации ПО – отсутствует.

Программный продукт CiscoWorks LMS предназначен в первую очередь для сетей, построенных на базе оборудования Cisco. Т. к. содержит множество инструментов, специально предназначенных для работы с данным оборудованием. В университете имеется ряд оборудования Cisco, но все же большая часть оборудования – других производителей.

Программные продукты, такие как Naumen Network Manager, Ivanti Endpoint Manager, SolarWinds Network Performance Monitor и Novell ZENworks Configuration Management, а также программно-аппаратный комплекс Dell KACE K1000 Management Appliance – являются мощными инструментами мониторинга ИТ-инфраструктуры предприятия, однако требуют весьма больших финансовых затрат на их приобретение.

Ключевым фактором выбора системы SCCM является доступность данного продукта для университета. Университет ежегодно приобретает лицензии на программные продукты Microsoft по подписке Microsoft Desktop Education ALNG LicSAPk OLVS. В данную подписку входят последние версии программных продуктов таких как: Windows, Windows Server, Office, SQL-server и т. д., в том числе пакет продуктов System Center. В связи с чем использование SCCM экономит средства университета на приобретение программного обеспечения для мониторинга и управления ИТ-инфраструктурой.

2 ПРОЕКТНЫЙ РАЗДЕЛ

2.1 Определение функциональных требований

Для грамотной разработки системы мониторинга сети первым делом нужно определиться с функциональными требованиями приложениями, это позволит чётко понимать то, что нужно разработчику от программы и что должен будет видеть в конечном итоге пользователь при открытии проекта, какой функционал он должен будет предоставлять и чего будет ожидать потребитель при открытии программы.

Создаваемое приложение будет базироваться на проверке доступности узла сети относительно сетевого интерфейса устройства, на котором оно будет запущено. Предназначено оно будет для удобного мониторинга нескольких десятков, а возможно и сотен узлов одновременно, что позволяет применять её в малых и средних компаниях, программа также должна будет иметь простой и понятный интерфейс с целью возможности внедрения в компании без прохождения курса обучения пользования программой.

Функциональные требования к такому приложению:

– Определение IP-адреса или URL проверяемого узла сети. Приложение должно иметь возможность указать адрес или URL узла сети, доступность которого нужно проверить.

– Проверка доступности узла сети с использованием ping-запросов или HTTP-запросов. Приложение должно уметь отправлять запросы к указанному узлу сети и анализировать ответы. Если ответы получены успешно, узел считается доступным, иначе – недоступным.

– Отображение результата проверки (успешно/недоступно). После завершения проверки приложение должно отобразить результат пользователю, например, сообщением «узел доступен» или «узел недоступен» или графическим отображением путём изменения цвета узла сети и интуитивно понятного результата по ассоциативному признаку.

– Возможность настройки периодичности проверок. Приложение должно предоставлять пользователю возможность выбрать интервал времени между проверками доступности узла сети.

2.2 Составление концептуальной и логической схемы

Исходя из данных, что были изложены о предполагаемом приложении, можно составить простую концептуальную схему того, как проверяется доступность узла сети и меняется его отображение представлено на рисунке 2.1.

Рисунок 2.1 – Концептуальная схема проверки узла

Также немаловажно разработать логическую схему работы приложения, она изображена на рисунке 2.2, в ней отражено то, что должны содержать определённые блоки данных.

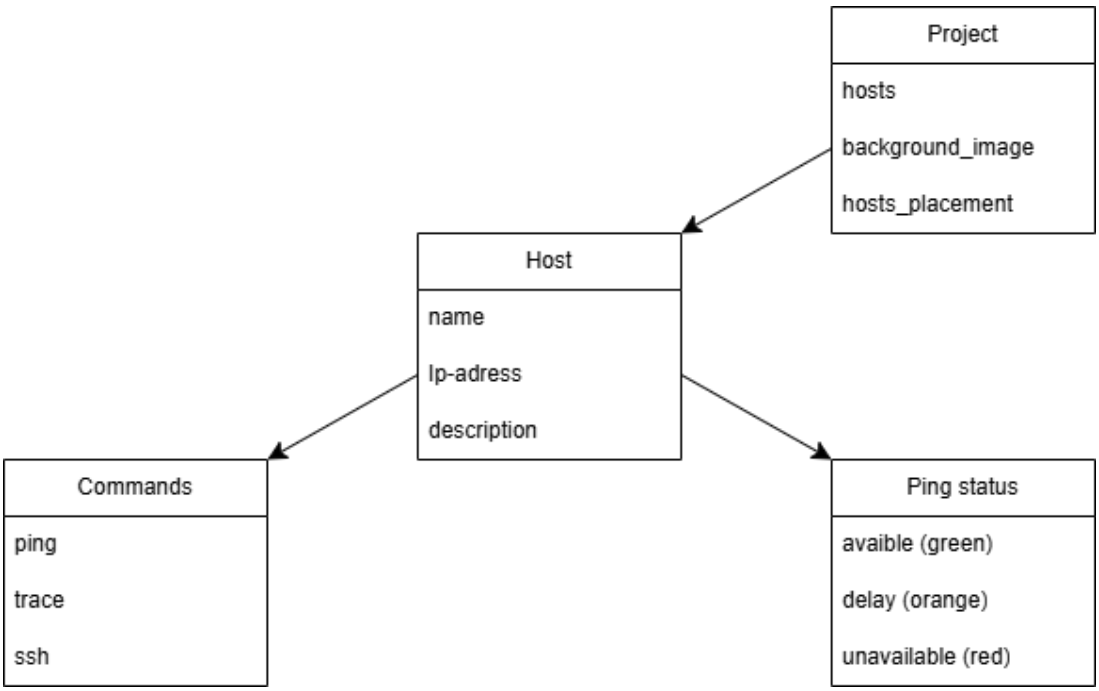


Рисунок 2.2 – Логическая схема работы приложения

2.3 Разработка пользовательского интерфейса

Программа (Pinghost) должна представлять собой удобное и функциональное приложение, предназначенное для управления и мониторинга сетевых ресурсов. Она должна иметь интуитивно понятный интерфейс, который облегчит работу пользователей.

Pinghost будет состоять из нескольких основных компонентов:

– мониторинг сети: здесь пользователи могут видеть список доступных узлов, их названия, IP-адреса и описания. Узлы будут отображаться в виде отдельных блоков, содержащих информацию о доступности адреса в цветовом формате (узлы после добавления на страницу мониторинга изначально приобретут серый цвет, когда проверка доступности ещё не осуществлялась, а далее, в зависимости от ответа, они будут становятся зелёными, когда доступность стабильна, или жёлтыми при высоком времени ответа, если ответа не последует, то узел будет становится красным, а значит недоступным для устройства, с которого будет запущена программа “Pinghost”);

– кнопка «Добавить»: позволит пользователям выбрать, добавить новый узел или отредактировать существующий. При добавлении нового узла необходимо будет указать его название, IP-адрес и описание;

– кнопка «Проект»: она будет использоваться для создания, загрузки и сохранения проектов, а также изменения подложки (заднего фона). Проекты можно будет сохранять в файл, содержащий информацию о расположении узлов из базы данных и о подложке, установленной для этого проекта. Проекты, аналогично можно будет и запускать из этого файла, что упростит перенос информации о множестве узлов и их расположении между устройствами;

– кнопка «Настройки»: позволит выбрать интервал проверки доступности узлов, расположенных на главной странице приложения.

Интерфейс начального окна Pinghost отображён на рисунке 2.3.

Рисунок 2.3 – Главное меню программы Pinghost

Кнопка «Проект» должна содержать кнопки «Создать», «Загрузить», «Сохранить» и «Изменить подложку» в соответствии с описанным ранее предполагаемым оформлением программы, на рисунке 2.4 отображено содержание кнопки «Проект» после нажатия на неё, по ходу написания функционала программы будет добавлена возможность взаимодействия со сторонними файлами для управления проектами, их сохранением и загрузкой.

Кнопка «Изменить подложку» будет использоваться для добавления фотографии на задний фон мониторинга сети, это позволит наглядно

расположить узлы сети для мониторинга, что устранит беспорядочное их расположение и увеличит интуитивную понятность того, что будет отображено на компоненте «Мониторинг сети» даже для пользователя без особых навыков владения компьютером и понимания его функционирования.

Рисунок 2.4 – Развернутая кнопка «Проект»

Следующая кнопка «Добавить» должна иметь возможность добавления не только узла на экран мониторинга, но и простых текстовых блоков для упрощения оформления подложки под свои конкретные задачи, что позволит использовать шаблонные подложки вместо постоянной её перерисовки вследствие необходимости добавления текста на мониторинг.

На рисунке 2.5 отображен функционал кнопки «Добавить»

Рисунок 2.5 – Кнопка «Добавить» после нажатия

Последняя в меню кнопка, которую необходимо добавить - это кнопка настроек, которая позволит настраивать интервалы отправки Ping – запроса и число неудачных попыток, после которых узел можно будет считать недоступным, наглядный пример нажатия кнопки «Настройки» изображён на рисунке 2.6.

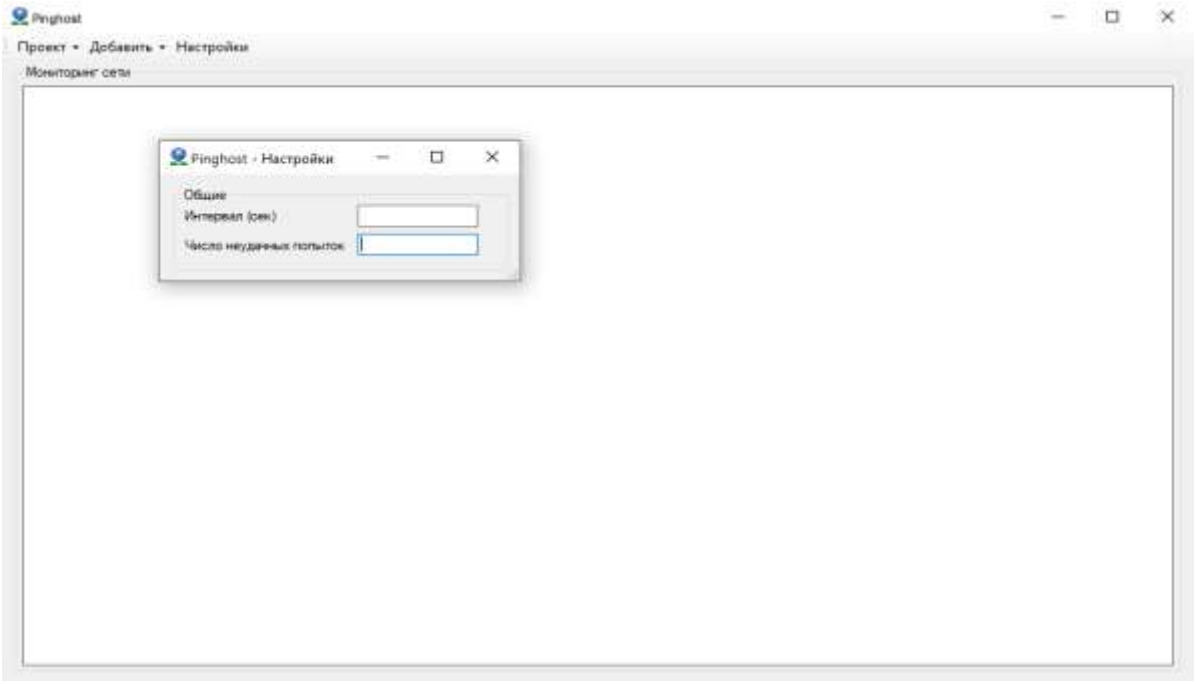


Рисунок 2.6 – Оформление кнопки «Настройки»

2.4 Реализация системы

В данной главе представлена реализация предлагаемой системы мониторинга сети. Эта система была реализована с использованием языка программирования C# в среде VisualStudio [3][6]. Платформа Microsoft .Net — это хорошая среда для сетевых программистов. Причина выбора C# заключается в том, что этот язык программирования позволяет программистам разрабатывать сетевые приложения с использованием сетевых функций Windows. Таким образом, что касается использования. Netframework и языка программирования C#, реализованное приложение сможет работать в операционной системе Windows [7][11][19].

Функциональная реализация предъявленных требований включила в себя добавление взаимодействия приложения с базой данных sqlite,

добавление и изменение записей в базе данных, а также сохранение проекта в файл формата «.hconf», подробное описание кода, используемого при работе системы мониторинга ЛВС Pinghost можно увидеть в приложении.

На рисунке 2.7 показано, что в разделе трассировки сети администратор сервера может ввести IP-адрес узла в специальное поле. После нажатия на кнопку запуска вся информация о маршруте между сервером и указанным узлом будет показана в рабочей области.

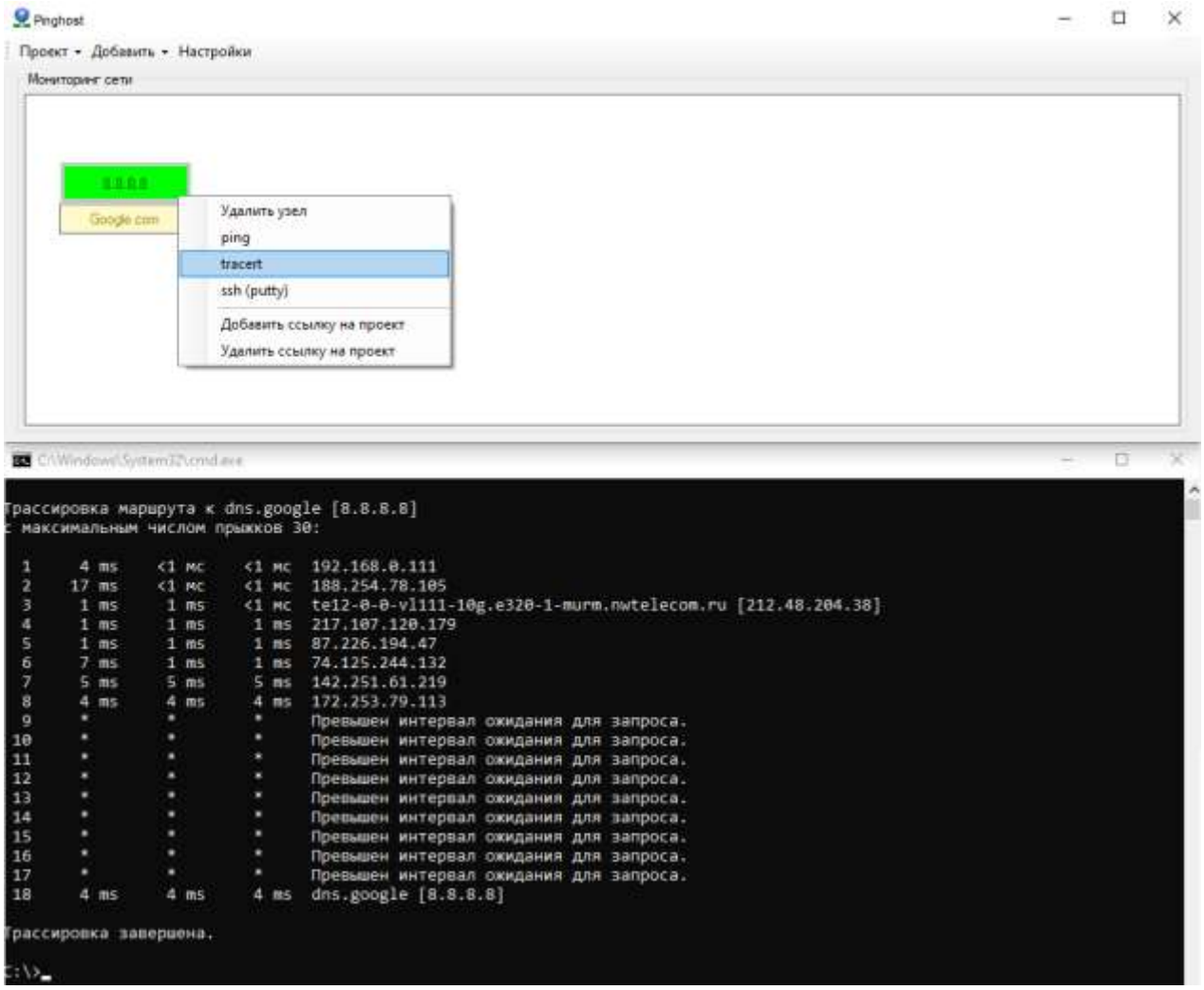


Рисунок 2.7 – Трассировка сети

В приведённом примере на рисунке 2.8 показан путь между серверным устройством и IP-адресом www.google.com. Информация включает маршрутизаторы и точки доступа, расположенные на пути между сервером и веб-сайтом Google. Обратите внимание, что IP-адреса международных маршрутизаторов не указаны из соображений безопасности. Одна из предоставленных данных – IP-адрес сетевых устройств в середине пути между

сервером и пунктом назначения. Также указана информация о времени, которая измеряется в миллисекундах и связана с расчётным временем получения пакета каждым сетевым устройством на маршруте.

В разделе серверной части следует учитывать, что устройство сервера должно иметь статический IP-адрес. В случае наличия динамического IP-адреса в разделе мониторинга сети IP-адрес клиента не будет отображаться в раскрывающемся списке. Причина в том, что эта система реализована на третьем уровне TCP/IP, то есть на сетевом уровне.

На рисунке 2.8 показана работа команды «ping». Эта утилита командной строки предназначена для проверки соединения с другим компьютером на уровне IP-адреса.

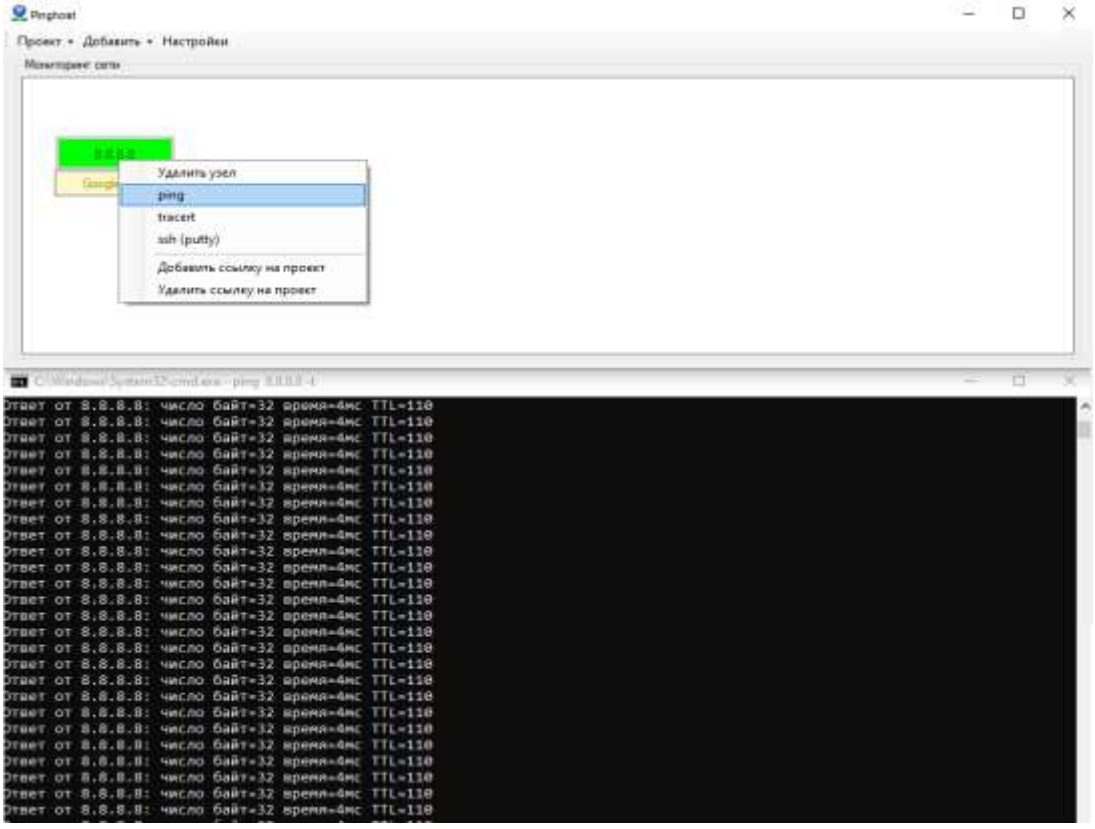


Рисунок 2.8 – Пример проверки одного из адресов

Работа утилиты основана на отправке серии небольших пакетов данных на указанное устройство и последующем отображении времени ответа. Эта основная команда TCP/IP используется для диагностики проблем с подключением, доступностью и разрешением имён. Также она позволяет определить имя и IP-адрес компьютера.

2.5 Расчет показателей экономической эффективности работы

Экономическая эффективность даёт возможность обосновать и поставить оценку о нуждаемости предприятия или компании в внедряемом программном продукте. Первичным вариантом технического процесса принималась работа сотрудников технической поддержки в обработке запросов без автоматизированной информационной системы. А в качестве внедряемого варианта предлагалась использовать разработанную автоматизированную информационную систему в рамках бакалаврской работы.

В таблице 2.1 представлены капитальные (единовременные) затраты проектирования ИС

Таблица 2.1 – Капитальные затраты проектирования ИС

Затраты	Состав затрат	Планируемая сумма (руб.)
Затраты на проектирование ИС	Затраты на заработную плату проектировщиков	18000
	Затраты на инструментальные программные средства для проектирования	1200
	Затраты на средства вычислительной технически для проектирования	2000
	Прочие затраты на проектирование	500
Затраты на технические средства управления		600
Затраты на создание линий связи локальных сетей		0
Затраты на программные средства		0
Затраты на формирование информационной базы	ЗП разработчика	10000
	Работа ЭВМ	700
Затраты на обучение персонала	ЗП обучаемого	1700
Затраты на опытную эксплуатацию	ЗП разработчика	7000
	Работа ЭВМ	500

Итого**:** сумма эксплуатационных затрат составляет 42361 рублей.

На рисунке 2.98 представлена диаграмма, отображающая соотношение статей капитальных затрат по проекту.

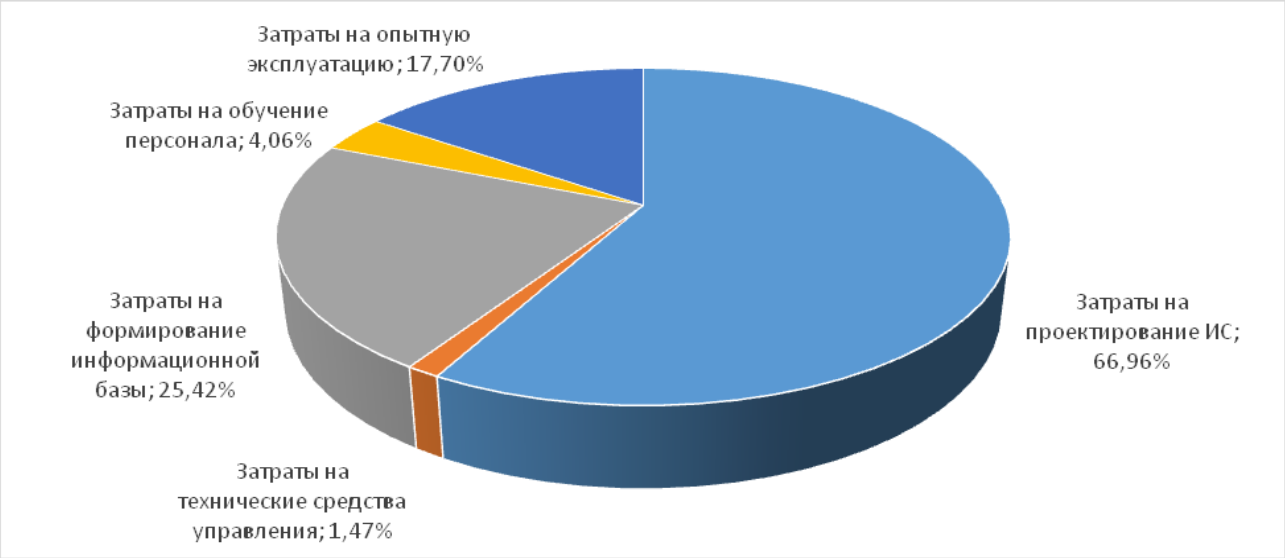


Рисунок 2.9 – Соотношение статей капитальных затрат по проекту

В таблице 2.2 представлены эксплуатационные затраты.

Таблица 9 – Эксплуатационные затраты проектирования системы мониторинга

Затраты	Сумма, руб
Зарплата сетевого администратора	0
Амортизационные отчисления	7000
Затраты на техническое обслуживание	5650
Затраты, связанные с использованием глобальных вычислительных сетей Internet	4000
Затраты на носители информации	0
Прочие затраты	3300

Итого**:** сумма эксплуатационных затрат составляет 21240 рублей в год.

На рисунке 2.10 представлена диаграмма, отображающая соотношение статей эксплуатационных затрат по проекту .

Эксплуатационные затраты, в отличие от капитальных, являются повторяющимися. Они повторяются в каждом цикле производства, а рассчитываются в сумме за год. Эксплуатационные затраты осуществляются синхронно с производством. Эксплуатационные затраты составляют себестоимость продукции или услуг.

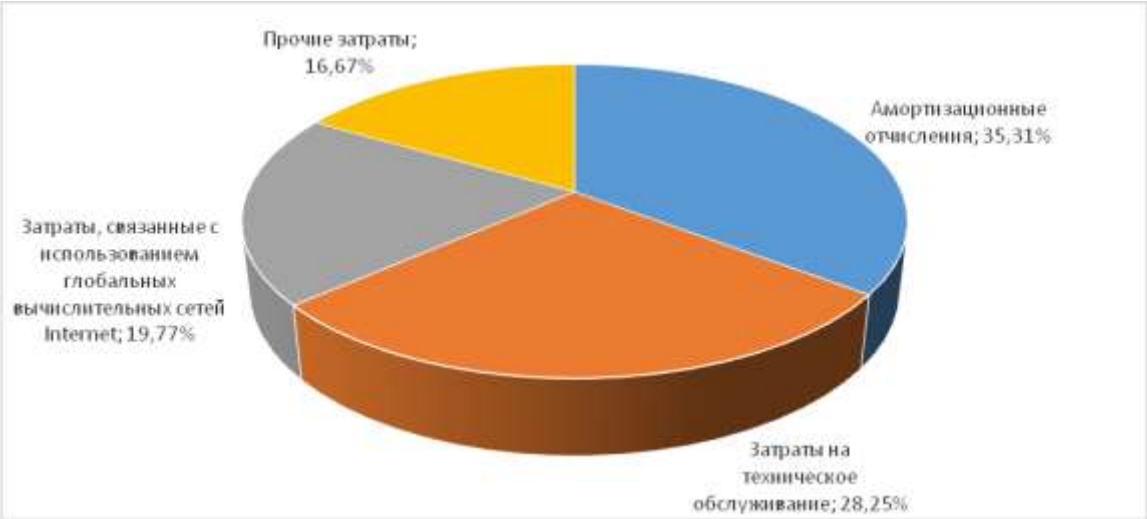


Рисунок 2.10 – Соотношение статей эксплуатационных затрат

Таким образом, видно, что использование второго варианта (проектного) может привести к немалому уменьшению рабочих и стоимостных издержек. Данный результат в общей сложности даёт понять, что предлагаемый вариант решает поставленные задачи в два раза быстрее и эффективнее базового варианта, что для предприятия или для компании очень выгодно. Данный продукт может поддерживать внедрение для снижения расходов и повышения качества и эффективности работы сотрудников.

Основываясь на материале, описываемым в данном разделе, можно сделать вывод, что было построено приложение, основанное на архитектуре клиент сервер.

Была реализована система мониторинга локально-вычислительной сети в организации, которая показала способность оценивать работоспособность за заданный период времени работоспособность серверов и другого оборудования.

А оценка затрат на проект показывает, что решение эффективное: ускоряет процесс выполнения задач и минимизирует возможные ошибки, что приводит к уменьшению рисков в процессе функционирования локально-вычислительной сети в организации.

ЗАКЛЮЧЕНИЕ

В ВКР были отражены концепции мониторинга сети, удаленного доступа и другие концепции, связанные с этой работой. Кроме того, были исследованы аналогичные работы, которые ранее проводились в этой области, и дано сравнение между реализованным приложением и существующими инструментами. Кроме того, процесс, процедуры и условия внедренного приложения, шаг за шагом, были подробно объяснены.

Анализ существующих систем мониторинга производительности сетевых систем показал, что подобные системы позволяют пользователю просматривать и управлять качеством производительности, анализом данных и ошибками сети Интернет-протокола (IP). Было выявлено, что инструмент мониторинга производительности является важным активом для системного администратора, позволяющим постоянно отслеживать работу сети на основе данных, собираемых через регулярные промежутки времени.

Основываясь на проведенном анализе, можно сделать вывод, что типичная сеть состоит из различных аппаратных устройств, таких как мосты, повторители, коммутаторы, маршрутизаторы и т. д. Мониторинг сети включает в себя проверку правильности функционирования этих устройств, а также обеспечение доступности связующей среды и нужен, чтобы обеспечивать бесперебойную работу вычислительной системы в организации. А предлагаемое решение будет аналогично существующим системам, но будет иметь интерактивный интерфейс и возможность быстрого обмена сообщениями через чат.

В рамках данной ВКР реализована упрощенная система мониторинга сети. Анализ исследований показал, что все существующие на рынке инструменты мониторинга сети имеют сложный пользовательский интерфейс, если они не работают на основе структуры командной строки.

Основной целью проекта ВКР было создание простой в использовании системы мониторинга сети, которая содержит большинство необходимых

функций. Мониторинг и анализ пакетов позволит администратору пользователя контролировать безопасность всей сети.

Система отвечала всем заданным функциональным требованиям. В том числе:

– система способна решить проблему, с которой столкнулся клиент, и позволить клиенту опубликовать проблему,

– информация сохраняется в базе данных, к которой уполномоченный персонал может получить удаленный доступ,

– лучшее управление IP-адресами.

Онлайн-система управления сетью может решить все поставленные перед ней задачи. В том числе:

– безопасная работающая СУБД для отправки сведений о проблеме,

– разработанная универсальная база данных для обмена данными о записях проблем,

– система, которая будет запрашивать информацию о проблемах и решениях по клиентам, отделам и зданиям из любого места.

Была реализована система мониторинга локально-вычислительной сети в организации, которая показала способность оценивать работоспособность за заданный период времени работоспособность серверов и другого оборудования. А оценка затрат на проект показывает, что решение эффективное: ускоряет процесс выполнения задач и минимизирует возможные ошибки, что приводит к уменьшению рисков в процессе функционирования локально-вычислительной сети в организации.

Система мониторинга сети может использоваться как студентами, так и начинающими пользователями. Таким образом, это приложение может быть использовано в образовательных и обучающих целях.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Артюшенко, В. В. Компьютерные сети и телекоммуникации: учебно-методическое пособие / В. В. Артюшенко, А. В. Никулин. – Новосибирск: Новосибирский государственный технический университет, 2020. – 72 с.

2. Архитектуры и топологии многопроцессорных вычислительных систем: учебник / А. В. Богданов, В. В. Корхов, В. В. Мареев, Е. Н. Станкова. – 3-е изд. – Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 135 с.

3. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, VisualStudio 2008): учебник / В. А. Биллиг. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 409 с.

4. Виноградов, Г. П. Компьютерные сети. Работа в сети Интернет: учебное пособие / Г. П. Виноградов, Е. Е. Фомина, Г. В. Кошкина. — Тверь: ТвГТУ, 2022. – 116 с.

5. Галас, В. П. Вычислительные системы, сети и телекоммуникации. Часть 1. Вычислительные системы: электронный учебник / В. П. Галас. – Владимир: Владимирский государственный университет им. А.Г. и Н.Г. Столетовых, 2019. – 232 с. [электронный ресурс] Режим доступа: https://www.iprbookshop.ru/57363.html (дата обращения: 17.05.2024)

6. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 томах. Т.II: учебник / С. В. Горелов; под редакцией П. Б. Лукьянова. – Москва: Прометей, 2019. – 378 с.

7. Казанский, А. А. Объектно-ориентированное программирование на языке Microsoft Visual C# в среде разработки Microsoft Visual Studio и

.NET Framework. 4.3: учебное пособие и практикум / А. А. Казанский. – Москва: Московский государственный строительный университет, ЭБС АСВ, 2021. – 180 с.

8. Ковган, Н. М. Компьютерные сети: учебное пособие / Н. М. Ковган. – Минск: Республиканский институт профессионального образования (РИПО), 2019. – 179 с. [электронный ресурс] Режим доступа: https://www.iprbookshop.ru/93384.html (дата обращения: 17.05.2024)

9. Компьютерные сети: учебник / В. Г. Карташевский, Б. Я. Лихтциндер, Н. В. Киреева, М. А. Буранова. – Самара: Поволжский государственный университет телекоммуникаций и информатики, 2021. – 267 с.

10. Кукарцев, В. В. Проектирование и архитектура информационных систем: учебник / В. В. Кукарцев, Р. Ю. Царев, О. А. Антамошкин. – Красноярск: Сибирский федеральный университет, 2021. – 192 с.

11. Мацкевич, А. Г. Лекции по курсу: Информационные технологии с изложением основ программирования на языке C#. Ч. 1 : учебное пособие / А. Г. Мацкевич. – Москва: Московский технический университет связи и информатики, 2021. – 81 с.

12. Минакова, О. В. Надежность информационных систем: учебник / О. В. Минакова. – Саратов: Вузовское образование, 2020. – 283 с.

13. Моделирование вычислительных сетей: методические указания / составители С. А. Олейникова, Т. И. Сергеева. – Воронеж: ВГТУ, 2022. – 40 с.

14. Нужнов, Е. В. Компьютерные сети. Часть 2. Технологии локальных и глобальных сетей: учебное пособие / Е. В. Нужнов. – Таганрог: Издательство Южного федерального университета, 2021. – 176 с.

15. Оливер, Ибе Компьютерные сети и службы удаленного доступа / ИбеОливер; перевод И. В. Синицын. – 2-е изд. – Саратов: Профобразование, 2020. – 335 с.

16. Проскуряков, А. В. Компьютерные сети. Основы построения компьютерных сетей и телекоммуникаций: учебное пособие / А. В. Проскуряков. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2020. – 201 с. [электронный ресурс] Режим доступа: https://www.iprbookshop.ru/87719.html (дата обращения: 17.05.2024)

17. Ракитин, Р. Ю. Компьютерные сети: учебное пособие / Р. Ю. Ракитин, Е. В. Москаленко. – Барнаул: Алтайский государственный педагогический университет, 2019. – 338 с. [электронный ресурс] Режим доступа: https://www.iprbookshop.ru/102731.html (дата обращения: 17.05.2024)

18. Сергеев, М. Ю. Компьютерные сети: практикум / М. Ю. Сергеев, Т. И. Сергеева, С. А. Олейникова. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2019. – 154 с. [электронный ресурс] Режим доступа: https://www.iprbookshop.ru/93261.html (дата обращения: 17.05.2024)

19. Bipin Joshi. Beginning XML with C# 7: XML Processing and Data Access for C# Developers. – 301 Pitruchhaya, Thane, India – 2022. – 464 p.

20. Douglas E. Comer. The Internet Book. Everything You Need to Know about Computer Networking and How the Internet Works: Fifth Edition. – CRC Press – 2021. – 405 p.

21. Irv Englander. The architecture of computer hardware, Systems software, & networking. – Bentley University – 2020. – 699 p.

22. Mike O'Leary. Cyber Operations: Building, Defending, and Attacking Modern Computer Networks. – Towson, MD, USA – 2019. – 1151 p.

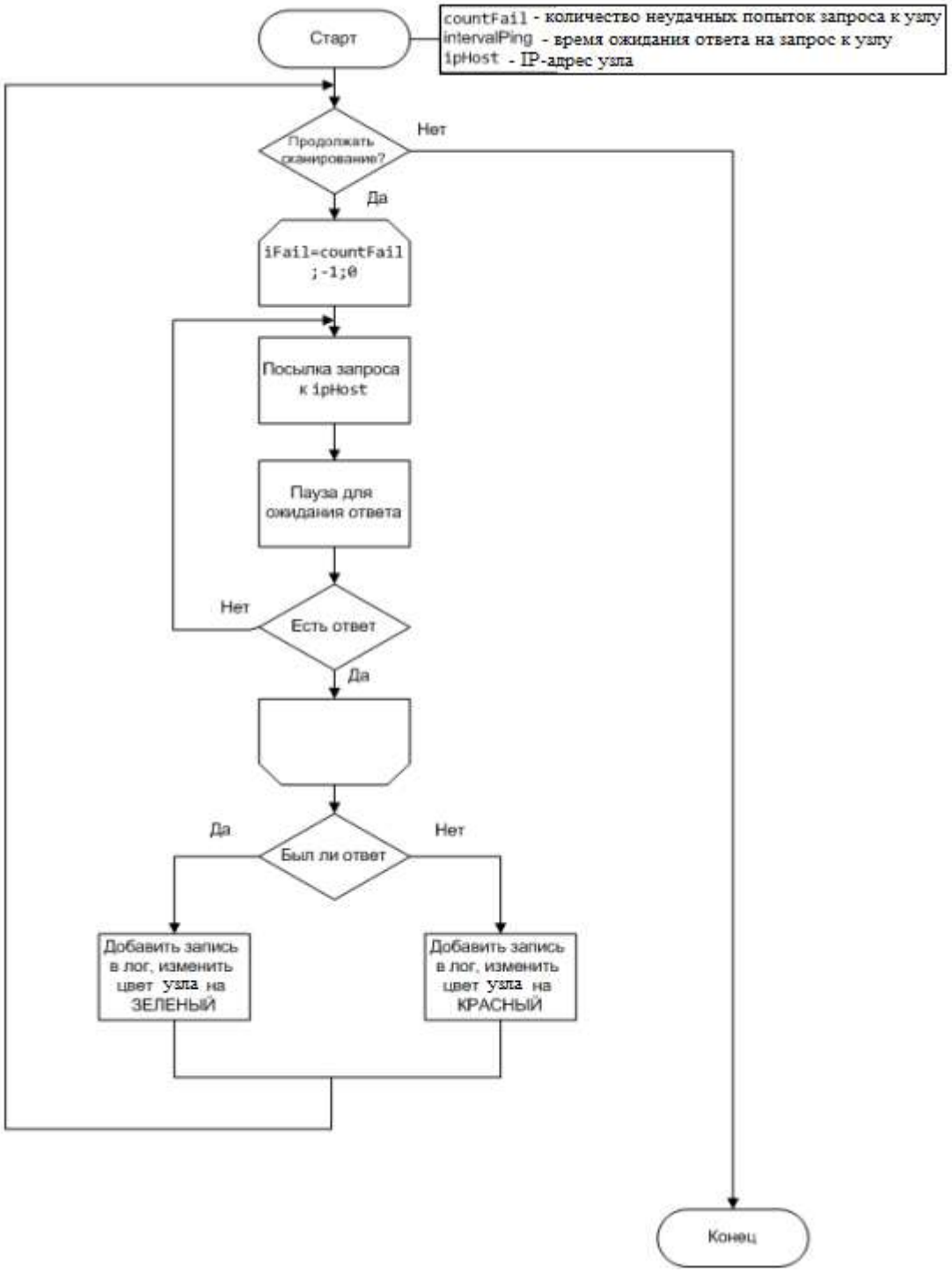
23. Morris Sloman. A survey of trust in internet applications. - IEEE Communications Surveys & Tutorials. – 2020. – Т3 (№4). – p. 2-16

24. Richard Fox, Wei Hao. Internet Infrastructure: Networking, Web Services, and Cloud Computing. – CRC Press – 2021. – 633 p.

25. Trimintzios P., Polychronakis and over. DiMAPI: An Application Programming Interface for Distributed Network Monitoring. – 2021. – p. 382-393.

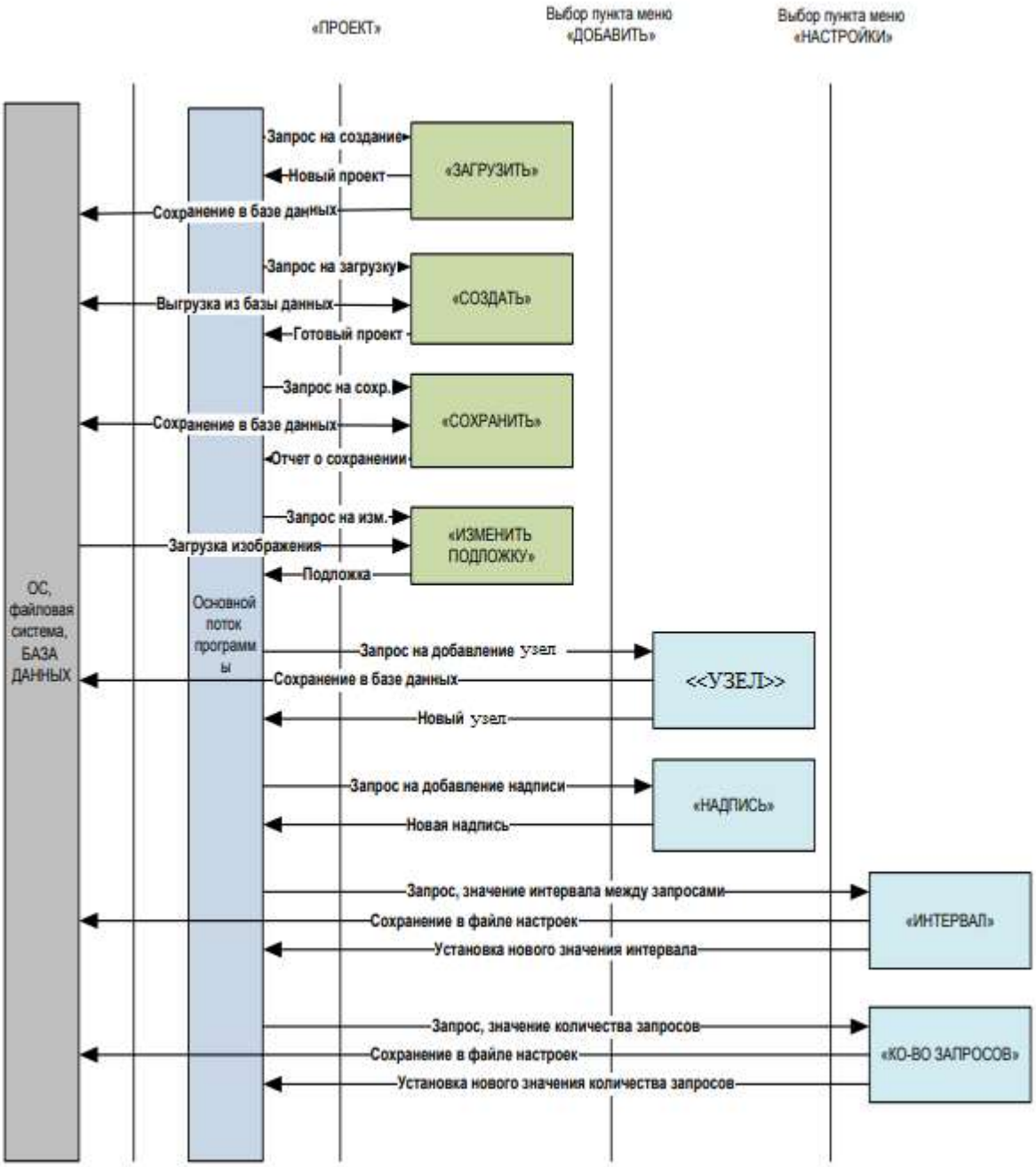
ПРИЛОЖЕНИЕ А

Схема алгоритма опроса хоста программы Ping_Host



ПРИЛОЖЕНИЕ Б

Диаграмма использования основного потока программы



ПРИЛОЖЕНИЕ В

Листинг программного кода

Файл “Logger.cs” – класс, описывающий систему логгирования в файл

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Windows.Forms;



namespace PingHost {

    //Класс логгера на C#
    class Logger {

        //Потоковый объект
        private StreamWriter loggerSW;

        //Путь до файла
        private string loggerPath;

        //Возможность дозаписи в файл
        private bool loggerAppendFlag;

        //Конструктор класса
        public Logger(string path, bool appendFlag)
        {
            string[] listElement = File.ReadAllLines(path, System.Text.Encoding.GetEncoding(1251));
            if (listElement.Length > 400)
                File.Delete(path);

            loggerPath = path;
            loggerAppendFlag = appendFlag;
            loggerSW = new StreamWriter(loggerPath, loggerAppendFlag);

        }

        //Запись лога в файл
        public void WriteLog(string strLog)
        {
            loggerSW.WriteLine(strLog);
            loggerSW.Flush();
        }

        //Запись дата_время_лог в файл
        public void WriteLogUtc(string strLog)
        {
           var sysdate = System.DateTime.UtcNow;
           sysdate = sysdate.AddHours(3);

           WriteLog(String.Concat(sysdate, " ", strLog));
        }

    }
}
```

Файл “AddHost.cs” – класс, описывающий добавление нового хоста

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SQLite;
using System.IO;

namespace PingHost
{
    //Класс-форма добавления нового хоста
    public partial class AddHost : Form
    {
        //имя базы данных
        private string dbName;
        //обьект для работы с базой данных
        private SQLiteConnection dbConnect;
        //обьек для формирования команд
        private SQLiteCommand dbCommand;

        //Конструктор класса
        public AddHost()
        {
            InitializeComponent();
```

```
            SQLConnect();

            //Выгрузка данных из БД
            string[,] arrayHost = getSQLHost();
            //Печать списка ip-хостов
            for (int i = 0; i < arrayHost.Length/4; i++)
            {
                DataGridViewRow row = new DataGridViewRow();
                object[] temp = new object[4];
                temp[0] = arrayHost[i, 0];
                temp[1] = arrayHost[i, 2];
                temp[2] = arrayHost[i, 1];
                temp[3] = arrayHost[i, 3];
                row.CreateCells(dataGridView1, temp);
                dataGridView1.Rows.AddRange(row);
            }
        }

        //Подключение к БД
        private void SQLConnect()
        {
            dbConnect = new SQLiteConnection();
            dbCommand = new SQLiteCommand();

            dbName = "resources/hostdb.sqlite";

            if (!File.Exists(dbName))
            {
                SQLiteConnection.CreateFile(dbName);
            }
            try
            {
                dbConnect = new SQLiteConnection("Data Source=" + dbName + ";Version=3;");
                dbConnect.Open();
                dbCommand.Connection = dbConnect;
                dbCommand.CommandText = "CREATE TABLE IF NOT EXISTS Host (id INTEGER PRIMARY KEY AUTOINCREMENT, ip TEXT, name TEXT, description
TEXT, id group INTEGER)";
                dbCommand.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Error");
            }
        }

        //Добавление нового хоста в базу данных
        private void addSQLHost(String ip, String name, String description)
        {
            if (dbConnect.State != ConnectionState.Open)
            {
                MessageBox.Show("Open connection with database");
                return;
            }
            try
            {
                dbCommand.CommandText = "INSERT INTO Host ('ip', 'name', 'description') values ('"
                                        + ip + "', '"
                                        + name + "', '"
                                        + description
                                        + "')";
                dbCommand.ExecuteNonQuery();
            }
            catch (SQLiteException ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }

        //Формирование запроса с фильтром(Вытаскивает из БД отфильтрованные данные)
        private string[,] filterDB(String fDB) {
            string[,] array;
            try {
                SQLConnect();
                dbCommand.CommandText = "SELECT id FROM Host WHERE ip LIKE "+"'%" + fDB + "%'";
                SQLiteDataReader dbRead = dbCommand.ExecuteReader();
                int count = 0;
                while (dbRead.Read()) {
                    count++;
                }
                dbRead.Close();

                dbCommand.CommandText = "SELECT * FROM Host WHERE ip LIKE " + "'%" + fDB + "%'";

                dbRead = dbCommand.ExecuteReader();
                int i = 0;
                Console.WriteLine(dbRead);
                array = new string[count, 4];
                while (dbRead.Read()) {
                    array[i, 0] = dbRead["id"].ToString();
                    array[i, 1] = dbRead["ip"].ToString();
                    array[i, 2] = dbRead["name"].ToString();
                    array[i, 3] = dbRead["description"].ToString();
                    i++;
                }
                dbRead.Close();
                return array;
            } catch (SQLiteException ex) {
                MessageBox.Show("Error: " + ex.Message);
                return new string[0, 0];
            }
```

```
}

private string[,] getSQLHost(string where = "")
{
    string[,] array;
    try
    {
        if (where.Length > 0)
        {
            where = " WHERE " + where;
        }
        dbCommand.CommandText = "SELECT id FROM Host" + where;
        SQLiteDataReader dbRead = dbCommand.ExecuteReader();
        int count = 0;
        while (dbRead.Read())
        {
            count++;
        }
        dbRead.Close();

        dbCommand.CommandText = "SELECT * FROM Host" + where;

        dbRead = dbCommand.ExecuteReader();
        int i = 0;
        Console.WriteLine(dbRead);
        array = new string[count, 4];
        while (dbRead.Read())
        {
            array[i, 0] = dbRead["id"].ToString();
            array[i, 1] = dbRead["ip"].ToString();
            array[i, 2] = dbRead["name"].ToString();
            array[i, 3] = dbRead["description"].ToString();
            i++;
        }
        dbRead.Close();
        return array;
    }
    catch (SQLiteException ex)
    {
        MessageBox.Show("Error: " + ex.Message);
        return new string[0,0];
    }

}

//Реакция на нажате кнопки добавления хоста
private void buttonAdd_Click(object sender, EventArgs e)
{
    var reg = new System.Text.RegularExpressions.Regex(@"\b(?:\d{1,3}\.){3}\d{1,3}\b");

    bool res = reg.IsMatch(ipTextBox.Text);

    if (nameTextBox.Text != "" && res == true)
    {
        if (getSQLHost("ip = '" + ipTextBox.Text + "'").Length <= 0)
        {
            addSQLHost(ipTextBox.Text, nameTextBox.Text, descriptionTextBox.Text);
        }
        this.Close();
    }
    else
    {
        errorProvider.SetError(ipTextBox, "Значение должно быть IP-адрессом!");
        ipTextBox.ResetText();
    }
}


private void listBoxHost_SelectedIndexChanged(object sender, EventArgs e)
{
}

//Реакция на нажате кнопки добавления хоста
private void buttonAdd_Click_1(object sender, EventArgs e)
{
    nameTextBox.Text = dataGridView1.SelectedRows[0].Cells[1].Value.ToString();
    ipTextBox.Text = dataGridView1.SelectedRows[0].Cells[2].Value.ToString();
    descriptionTextBox.Text = dataGridView1.SelectedRows[0].Cells[3].Value.ToString();
    this.Close();
}

//Реакция на нажате кнопки создания хоста
private void buttonCreate_Click(object sender, EventArgs e)
{
    AddHostCreate formCreate = new AddHostCreate();
    formCreate.ShowDialog();

    dataGridView1.Rows.Clear();

    string[,] arrayHost = getSQLHost();
    for (int i = 0; i < arrayHost.Length / 4; i++)
    {
        DataGridViewRow row = new DataGridViewRow();
        object[] temp = new object[4];
        temp[0] = arrayHost[i, 0];
        temp[1] = arrayHost[i, 2];
        temp[2] = arrayHost[i, 1];
        temp[3] = arrayHost[i, 3];
        row.CreateCells(dataGridView1, temp);
        dataGridView1.Rows.AddRange(row);
    }
```

```
        }

        //Реакция на нажате кнопки редактирования хоста
        private void buttonEdit_Click(object sender, EventArgs e)
        {
            AddHostCreate formCreate = new AddHostCreate();
            formCreate.Controls.Find("textBoxID", false)[0].Text = dataGridView1.SelectedRows[0].Cells[0].Value.ToString();
            formCreate.ShowDialog();

            dataGridView1.Rows.Clear();

            string[,] arrayHost = getSQLHost();
            for (int i = 0; i < arrayHost.Length / 4; i++)
            {
                DataGridViewRow row = new DataGridViewRow();
                object[] temp = new object[4];
                temp[0] = arrayHost[i, 0];
                temp[1] = arrayHost[i, 2];
                temp[2] = arrayHost[i, 1];
                temp[3] = arrayHost[i, 3];
                row.CreateCells(dataGridView1, temp);
                dataGridView1.Rows.AddRange(row);
            }
        }

        //Реакция на изменение строки-фильтра
        private void textBox1_TextChanged(object sender, EventArgs e) {
            dataGridView1.Rows.Clear();
            string[,] arrayHost = filterDB(textBox1.Text);

            for (int i = 0; i < arrayHost.Length / 4; i++) {
                //listBoxHost.Items.Add(arrayHost[1,1]);
                DataGridViewRow row = new DataGridViewRow();
                object[] temp = new object[4];
                temp[0] = arrayHost[i, 0];
                temp[1] = arrayHost[i, 2];
                temp[2] = arrayHost[i, 1];
                temp[3] = arrayHost[i, 3];
                row.CreateCells(dataGridView1, temp);
                dataGridView1.Rows.AddRange(row);
            }
        }


    }
}
```

Файл “AddHostCreate.cs” – класс, описывающий занесение хоста в базу данных

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SQLite;
using System.IO;


namespace PingHost {

    //Класс представляющий форму для сохранения нового хоста
    public partial class AddHostCreate : Form {
        private string dbName;
        private SQLiteConnection dbConnect;
        private SQLiteCommand dbCommand;

        //Конструктор класса
        public AddHostCreate() {
            InitializeComponent();
            SQLConnect();
        }

        //Метод для подключения к базе данных
        private void SQLConnect() {
            dbConnect = new SQLiteConnection();
            dbCommand = new SQLiteCommand();

            dbName = "resources/hostdb.sqlite";

            if (!File.Exists(dbName)) {
                SQLiteConnection.CreateFile(dbName);
            }

            try {
                dbConnect = new SQLiteConnection("Data Source=" + dbName + ";Version=3;");
                dbConnect.Open();
                dbCommand.Connection = dbConnect;
                dbCommand.CommandText = "CREATE TABLE IF NOT EXISTS Host (id INTEGER PRIMARY KEY AUTOINCREMENT, ip TEXT, name TEXT, description
TEXT, id_group INTEGER)";
                dbCommand.ExecuteNonQuery();
```

```
    } catch {
        MessageBox.Show("Error");
    }
}

//Метод, реализующий SQL-запрос добавления нового хоста
private void addSQLHost(String ip, String name, String description) {
    if (dbConnect.State != ConnectionState.Open) {
        MessageBox.Show("Open connection with database");
        return;
    }

    try {
        dbCommand.CommandText = "INSERT INTO Host ('ip', 'name', 'description') values ('"
                                + ip + "', '"
                                + name + "', '"
                                + description
                                + "')";
        dbCommand.ExecuteNonQuery();
    } catch (SQLiteException ex) {
        MessageBox.Show("Error: " + ex.Message);
    }
}

//Метод, реализующий SQL-запрос изменения параметров хоста
private void editSQLHost(int id, String ip, String name, String description) {
    if (dbConnect.State != ConnectionState.Open) {
        MessageBox.Show("Open connection with database");
        return;
    }

    try {
        dbCommand.CommandText = "UPDATE Host SET "
                                + "ip = '" + ip + "',"
                                + "name = '" + name + "',"
                                + " description = '" + description
                                + "' WHERE id = "
                                + id;
        dbCommand.ExecuteNonQuery();
    } catch (SQLiteException ex) {
        MessageBox.Show("Error: " + ex.Message);
    }
}

//Метод, реализующий выгрузку данных о хостахиз базы данных
private string[,] getSQLHost(string where = "") {
    string[,] array;
    try {
        if (where.Length > 0) {
            where = " WHERE " + where;
        }
        dbCommand.CommandText = "SELECT id FROM Host" + where;
        SQLiteDataReader dbRead = dbCommand.ExecuteReader();
        int count = 0;
        while (dbRead.Read()) {
            count++;
        }
        dbRead.Close();

        dbCommand.CommandText = "SELECT * FROM Host" + where;

        dbRead = dbCommand.ExecuteReader();
        int i = 0;
        Console.WriteLine(dbRead);
        array = new string[count, 4];
        while (dbRead.Read()) {
            array[i, 0] = dbRead["id"].ToString();
            array[i, 1] = dbRead["ip"].ToString();
            array[i, 2] = dbRead["name"].ToString();
            array[i, 3] = dbRead["description"].ToString();
            i++;
        }
        dbRead.Close();
        return array;
    } catch (SQLiteException ex) {
        MessageBox.Show("Error: " + ex.Message);
        return new string[0, 0];
    }
}

//Метод, реагирующий на на событие сохранения
private void buttonSave_Click(object sender, EventArgs e) {
    if (textBoxID.Text.Length > 0) {
        var reg = new System.Text.RegularExpressions.Regex(@"\b(?:\d{1,3}\.){3}\d{1,3}\b");

        bool res = reg.IsMatch(textBoxIP.Text);

        if (res == true) {
            editSQLHost(int.Parse(textBoxID.Text), textBoxIP.Text, textBoxName.Text, textBoxDescription.Text);
            this.Close();
        } else {
            errorProvider.SetError(textBoxIP, "Значение должно быть IP-адрессом!");
            textBoxIP.ResetText();
        }
    } else {
        var reg = new System.Text.RegularExpressions.Regex(@"\b(?:\d{1,3}\.){3}\d{1,3}\b");

        bool res = reg.IsMatch(textBoxIP.Text);
        if (res == true) {
            if (getSQLHost("ip = '" + textBoxIP.Text + "'").Length <= 0) {
                addSQLHost(textBoxIP.Text, textBoxName.Text, textBoxDescription.Text);
            }
```

```
                this.Close();
            } else {
                errorProvider.SetError(textBoxIP, "Значение должно быть IP-адрессом!");
                textBoxIP.ResetText();
            }
        }
    }

    //Метод выгрузки хоста
    private void AddHostCreate_Load(object sender, EventArgs e) {
        if (textBoxID.Text.Length > 0) {
            string[,] arrayHost = getSQLHost("id = '" + textBoxID.Text + "'");
            textBoxIP.Text = arrayHost[0, 1];
            textBoxName.Text = arrayHost[0, 2];
            textBoxDescription.Text = arrayHost[0, 3];
        }
    }
  }
}
```

Файл “AddLabel.cs” – класс, описывающий добавление подписи к хосту

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PingHost {
    public partial class AddLabel : Form {
        public AddLabel(string defaultLabelText = "") {
            InitializeComponent();
            labelTextBox.Text = defaultLabelText;
        }
    }
}
```

Файл “AddProject.cs” – класс, описывающий загрузку проекта

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PingHost
{
    public partial class AddProject : Form
    {
        public AddProject()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBoxNameProject.Text != "")
            {
                this.Close();
            }
        }
    }
}
```

Файл “Main.cs” – класс, описывающий работу основного потока ПО

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SQLite;
using System.IO;
using System.Threading;
using System.Net.NetworkInformation;
using System.Diagnostics;
```

```
using System.Media;


namespace PingHost {
    //Класс, описывающий работу основного потока программы
    public partial class Main : Form {
        //Флаг остановки потока
        bool stopThr = false;

        //Имя базы данных
        private string dbName;
        //Имя базы данных для хранения статистики
        private string dbStatName;
        //Объект для подключения к базе данных
        private SQLiteConnection dbConnect;
        //Объект для формирования команд
        private SQLiteCommand dbCommand;

        //Объект для подключения к базе данных, хранящей статистику
        private SQLiteConnection dbStatConnect;
        //Объект для формирования команд к базе данных, хранящей статистику
        private SQLiteCommand dbStatCommand;

        //Поле, хранящее интервал запросов к статистической базе данных
        private int intervalPingStat = 15;
        //Поле, хранящее интервал запросов к хостам
        private int intervalPing = 1;
        //Поле, хранящее количество неудачных попыток запросов
        private int countError = 3;

        //Объект для нотификации
        private static NotifyIcon noty = new NotifyIcon();
        //Объект для проигрывания звука
        private SoundPlayer soundErr = new SoundPlayer();

        //Имя картинки
        string imgFileName = "";
        //Имя проекта
        string projectName = "";

        //Объект для записи лога
        Logger failPingLog;
        //Делегат для записи лога
        delegate void printLog(string str);
        //Ссылка на функцию
        private printLog printLogFunc;
        private bool flagLog = false;

        //Метод, записывающий лог на главную форму программы
        private void printingLog(string str) {
            if (flagLog) richTextBox1.AppendText(str);
        }

        //Главный метод программы
        public Main(string projectPath = "") {
            //Инициализация перед началом работы
            InitializeComponent();
            SQLConnect();
            failPingLog = new Logger(@"..\..\..\PingHostLog.txt", true);
            printLogFunc = new printLog(printingLog);

            //Разбор файла конфигурации для проекта
            if (projectPath != "") {
                if (File.Exists(projectPath)) {
                    string[] listElement = File.ReadAllLines(projectPath, System.Text.Encoding.GetEncoding(1251));

                    foreach (string stringConf in listElement) {
                        string stringConfNew = stringConf.Replace(@"\x\s", "\r\n");
                        string[] confSplit = stringConfNew.Split(';');
                        string configType = confSplit[0];

                        switch (configType) {
                            //Конфигурирование имени проекта
                            case "name":
                                projectName = confSplit[1];
                                this.Text = "Рыжабикс: " + projectName;
                                break;

                            //Конфигурирование подложки
                            case "background":
                                imgFileName = confSplit[1];
                                if (imgFileName != "")
                                    try {
                                        panelHosts.BackgroundImage = Image.FromFile(imgFileName);
                                    } catch (FileNotFoundException) {
                                        MessageBox.Show("Ошибка при загрузке подложки: " + imgFileName + "\nПодложка не найдена");
                                    }
                                break;

                            //Конфигурирование хоста
                            case "host":
                                string ipAddress = confSplit[1];
                                string name = confSplit[2];
                                string description = confSplit[3];
                                int xOffSet = int.Parse(confSplit[4]);
                                int yOffSet = int.Parse(confSplit[5]);
                                string linkProject = confSplit[6];
                                AddHostIco(ipAddress, name, description, xOffSet, yOffSet, linkProject);
                                break;

                            //Конфигурирование метки
```

```
                        case "label":
                            AddLabelIco(confSplit[1], int.Parse(confSplit[2]), int.Parse(confSplit[3]));
                            break;

                        //Если неизвестна инструкция в файле конфигурации
                        default:
                            MessageBox.Show("Неопознанная опция в файле конфигурации:\n" + stringConf);
                            break;
                    }
                }
            }
        }

        //Разбор файла конфигурации для программы
        string fileName = "settings.conf";
        if (File.Exists(fileName)) {
            string[] listElement = File.ReadAllLines(fileName, System.Text.Encoding.GetEncoding(1251));
            foreach (string stringConf in listElement) {
                string stringConfNew = stringConf.Replace(@"\x\s", "\r\n");
                string[] confSplit = stringConfNew.Split('=');

                switch (confSplit[0]) {
                    case "ALL_INTERVAL":
                        intervalPing = int.Parse(confSplit[1]);
                        break;

                    case "COUNT_ERROR":
                        countError = int.Parse(confSplit[1]);
                        break;

                    case "STATUS_INTERVAL":
                        intervalPingStat = int.Parse(confSplit[1]);
                        break;

                    case "LOGGING":
                        flagLog = confSplit[1] == "ON";
                        break;
                }
            }
        }

        //Загрузка звукового файла для нотификации
        noty.Icon = global::PingHost.Properties.Resources.noty;
        soundErr.SoundLocation = "resources/win_error.wav";
        try {
            soundErr.Load();
        } catch {
            MessageBox.Show("Не удалось загрузить звуковой файл: " + soundErr.SoundLocation);
        }

    }

    //Подключение к базе данных
    private void SQLConnect() {
        dbConnect = new SQLiteConnection();
        dbCommand = new SQLiteCommand();
        dbStatConnect = new SQLiteConnection();
        dbStatCommand = new SQLiteCommand();

        dbName = "resources/hostdb.sqlite";
        dbStatName = "resources/statist_" + System.DateTime.Now.Month.ToString() + "_" + System.DateTime.Now.Year.ToString() + ".sqlite";

        if (!File.Exists(dbName)) {
            SQLiteConnection.CreateFile(dbName);
        }

        try {
            dbConnect = new SQLiteConnection("Data Source=" + dbName + ";Version=3;");
            dbConnect.Open();
            dbCommand.Connection = dbConnect;

            dbCommand.CommandText = "CREATE TABLE IF NOT EXISTS Host (id INTEGER PRIMARY KEY AUTOINCREMENT, ip TEXT, name TEXT, descriptic
TEXT, id_group INTEGER)";
            dbCommand.ExecuteNonQuery();

            dbCommand.CommandText = "CREATE TABLE IF NOT EXISTS GroupHost (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT)";
            dbCommand.ExecuteNonQuery();
        } catch {
            MessageBox.Show("Error");
        }

        if (!File.Exists(dbStatName)) {
            SQLiteConnection.CreateFile(dbStatName);
        }


        try {
            dbStatConnect = new SQLiteConnection("Data Source=" + dbStatName + ";Version=3;");
            dbStatConnect.Open();
            dbStatCommand.Connection = dbStatConnect;
            dbStatCommand.CommandText = "CREATE TABLE IF NOT EXISTS StatisticHost (id INTEGER PRIMARY KEY AUTOINCREMENT, id_host INTEGER,
data TEXT, status LONG)";
            dbStatCommand.ExecuteNonQuery();
        } catch {
            MessageBox.Show("Error");
        }
    }


    //Вставка нового хоста в базу данных
    private void addSQLHost(String ip, String name, String description) {
```

```
            if (dbConnect.State != ConnectionState.Open) {
                MessageBox.Show("Open connection with database");
                return;
            }

            try {
                dbCommand.CommandText = "INSERT INTO Host ('ip', 'name', 'description') values ('" + ip + "', '" + name + "', '" + description
+ "')";
                dbCommand.ExecuteNonQuery();
            } catch (SQLiteException ex) {
                MessageBox.Show("Error: " + ex.Message);
            }
        }

        //Выгрузка информации из базы данных
        private string[,] getSQLHost(string where = "") {
            string[,] array;
            try {
                if (where.Length > 0) {
                    where = " WHERE " + where;
                }
                dbCommand.CommandText = "SELECT id FROM Host" + where;
                SQLiteDataReader dbRead = dbCommand.ExecuteReader();
                int count = 0;
                while (dbRead.Read()) {
                    count++;
                }
                dbRead.Close();

                dbCommand.CommandText = "SELECT * FROM Host" + where;

                dbRead = dbCommand.ExecuteReader();
                int i = 0;
                Console.WriteLine(dbRead);
                array = new string[count, 4];
                while (dbRead.Read()) {
                    array[i, 0] = dbRead["id"].ToString();
                    array[i, 1] = dbRead["ip"].ToString();
                    array[i, 2] = dbRead["name"].ToString();
                    array[i, 3] = dbRead["description"].ToString();
                    i++;
                }
                dbRead.Close();
                return array;
            } catch (SQLiteException ex) {
                MessageBox.Show("Error: " + ex.Message);
                return new string[0, 0];
            }
        }

        //Добавление хоста в базу данных статистики
        private void addStatisticHost(int id, string data, long status) {
            SQLiteConnection dbStatConnect;
            SQLiteCommand dbStatCommand;
            dbStatCommand = new SQLiteCommand();
            dbStatConnect = new SQLiteConnection("Data Source=" + dbStatName + ";Version=3;");
            dbStatConnect.Open();
            dbStatCommand.Connection = dbStatConnect;

            if (dbStatConnect.State != ConnectionState.Open) {
                MessageBox.Show("Open connection with database");
                return;
            }

            try {
                dbStatCommand.CommandText = "INSERT INTO StatisticHost ('id_host', 'data', 'status') values ('"
                                          + id + "', '"
                                          + data + "', '"
                                          + status
                                          + "')";
                dbStatCommand.ExecuteNonQuery();
            } catch { }
            dbConnect.Close();
        }


        //Реакция на нажатие кнопки добавления хоста на форме
        private void toolStripMenuItemAddHost_Click(object sender, EventArgs e) {
            Form addHostForm = new AddHost();
            addHostForm.ShowDialog();
            Control.ControlCollection formControl = addHostForm.Controls;

            string nameForm = formControl.Find("nameTextBox", false)[0].Text;
            string ipForm = formControl.Find("ipTextBox", false)[0].Text;
            string descriptionForm = formControl.Find("descriptionTextBox", false)[0].Text;


            string[,] arrayHost = getSQLHost("ip like '" + formControl.Find("ipTextBox", false)[0].Text + "'");

            if (arrayHost.Length > 0) {
                AddHostIco(arrayHost[0, 1], arrayHost[0, 2], arrayHost[0, 3]);
            }

        }

        //Добавление графического представления хоста на форму
        private void AddHostIco(string ip, string name, string description, int offsetX = 30, int offsetY = 60, string linkProject = "") {
            Button btHostButton = new Button();
            btHostButton.Size = new Size(110, 35);
            btHostButton.Name = "new_host_button";
            btHostButton.Location = new Point(0, 0);
            btHostButton.Margin = new Padding(0, 0, 0, 0);
```

```
    btHostButton.Enabled = false;
    btHostButton.Text = ip;
    btHostButton.Font = new Font("Microsoft Sans Serif", 9, (linkProject == "" ? FontStyle.Regular : FontStyle.Underline));
    btHostButton.BackColor = Color.Gray;

    Label btHostLabel = new Label();
    btHostLabel.Name = "new_host_label";
    btHostLabel.Size = new Size(110, 25);
    btHostLabel.MaximumSize = new Size(110, 25);
    btHostLabel.Location = new Point(0, 55);
    btHostLabel.Enabled = false;
    btHostLabel.Text = name;
    if (name.Length <= 0) {
        btHostLabel.Visible = false;
    }

    btHostLabel.AutoSize = false;
    btHostLabel.AutoEllipsis = true;
    btHostLabel.Margin = new Padding(0, 0, 0, 0);
    btHostLabel.BackColor = Color.LemonChiffon;
    btHostLabel.BorderStyle = BorderStyle.FixedSingle;
    btHostLabel.TextAlign = ContentAlignment.MiddleCenter;
    btHostLabel.Tag = linkProject;
    btHostLabel.Font = new Font("Microsoft Sans Serif", 8);


    HostButton btHost = new HostButton();
    btHost.Name = "new_host";
    btHost.Location = new System.Drawing.Point(offsetX, offsetY);
    btHost.Controls.Add(btHostButton);
    btHost.Controls.Add(btHostLabel);
    btHost.Size = new System.Drawing.Size(110, 60);
    if (name.Length <= 0) {
        btHost.Size = new System.Drawing.Size(110, 45);
    }

    btHost.TabIndex = 1;
    btHost.Text = "123";
    btHost.ContextMenuStrip = this.contextMenuStripHostButton;
    btHost.MouseDoubleClick += new MouseEventHandler(HostIco_DoublClick);
    btHost.MouseHover += new EventHandler(HostIco_Hover);
    btHost.MouseLeave += new EventHandler(HostIco_Leave);
    btHost.AccessibleDescription = description;
    this.toolTip1.SetToolTip(btHost, btHost.AccessibleDescription);
    panelHosts.Controls.Add(btHost);

    Thread thr = new Thread(checkPing);
    thr.Start(btHost);
    //btHost.ControlRemoved += new ControlEventHandler();
}

//Добавление графического изображения подписи на форму
private void AddLabelIco(string text, int offsetX = 30, int offsetY = 60, string linkProject = "") {
    Label btHostLabel = new HostLabel();
    btHostLabel.Name = "new_host_label";
    btHostLabel.AutoSize = true;
    btHostLabel.AutoEllipsis = true;
    btHostLabel.Margin = new Padding(0, 0, 0, 0);
    btHostLabel.BackColor = Color.LemonChiffon;
    btHostLabel.BorderStyle = BorderStyle.FixedSingle;
    btHostLabel.TextAlign = ContentAlignment.MiddleCenter;
    btHostLabel.Text = text;
    btHostLabel.Location = new System.Drawing.Point(offsetX, offsetY);
    btHostLabel.ContextMenuStrip = contextMenuStripHostLabel;
    btHostLabel.Font = new Font("Microsoft Sans Serif", 10);
    panelHosts.Controls.Add(btHostLabel);
}


//Старт всех потоков, прослушивающих хосты
private void startAllCheckPing() {
    foreach (Control host in panelHosts.Controls) {
        if (host.GetType().FullName == "PingHost.HostButton") {
            Thread thr = new Thread(checkPing);
            thr.Start(host);
        }
    }
}


//Старт записи данных для статистики
private void startCheckStatisticPing(object array) {
    string[] arrayHost = (string[])array;
    var ping = new Ping();
    int iFail = 0;
    int countFail = countError;
    PingReply pingRes;
    stopThr = false;
    while (!stopThr) {
        if (buttonIndicator.BackColor == Color.LimeGreen) {
            buttonIndicator.BackColor = Color.Lime;
        } else {
            buttonIndicator.BackColor = Color.LimeGreen;
        }

        Thread.Sleep(1000 * intervalPingStat);
        try {
            pingRes = ping.Send(arrayHost[1]);
            if (pingRes.Status == IPStatus.Success) {
                long pingVal = pingRes.RoundtripTime;
                if (pingVal < 1) {
                    pingVal = 1;
```