МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительные системы и информационная безопасность»

И.о. зав. каф. «ВСиИБ»

_____ А.Р. Газизов

подпись И.О. Фамилия

«___» _____ 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема «СИСТЕМА ЦЕНТРАЛИЗОВАННОГО УПРАВЛЕНИЯ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТЬЮ»

Направление подготовки	10.05.02 Информационная безопасность телекоммуникационных систем
Специализация	Защита информации в системах связи и управления
Обозначение ВКР	10.05.02.730000.000 группа ВИБТ61

Обучающийся _____ (подпись, дата) В.В. Фомичев И.О.Ф.

Руководитель ВКР _____ (подпись, дата) доц., к.т.н. В.В. Галушка должность, И.О.Ф.

Консультанты по разделам:

Безопасность и экологичность проекта _____ (подпись, дата) проф., д.т.н. В.Л. Гапонов (должность, И.О.Ф.)

Технико-экономическое обоснование _____ (подпись, дата) ст. преп. А.В. Белоусова (должность, И.О.Ф.)

Нормоконтроль _____ (подпись, дата) ст. преп. М.А. Ганжур (должность, И.О.Ф.)

Ростов-на-Дону

2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительные системы и информационная безопасность»

Зав. кафедрой «ВСиИБ»

_____ В.А. Фатхи

подпись И.О. Фамилия

«___» _____ 2022 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

Тема «СИСТЕМА ЦЕНТРАЛИЗОВАННОГО УПРАВЛЕНИЯ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТЬЮ»

Обучающийся Фомичев Владимир Викторович

Обозначение ВКР 10.05.02.730000.000 группа ВИБТ61

Тема утверждена приказом по ДГТУ от «23» июня 2022 г. № 2831-ЛС-О

Срок представления ВКР к защите «1» февраля 2023 г.

Исходные данные для выполнения выпускной квалификационной работы:

Требования к функциональности и безопасности телекоммуникационных систем, методы управления устройствами в сети, протоколы удалённого доступа, средства хранения истории версий, типовые варианты политик безопасности организации, перечень команд для настройки сетевого оборудования, библиотеки для программной реализации взаимодействия по протоколу telnet.

Содержание выпускной квалификационной работы

Введение: Во введении необходимо изложить актуальность выбранной темы, обозначить объект и предмет исследования, цель и задачи выпускной квалификационной работы, теоретическую и практическую значимость работы, структуру работы.

Наименование и краткое содержание разделов:

1. Проблема централизованного управления сетью. Технологии компьютерных сетей. Задачи администрирования компьютерных сетей. Требования к безопасности сетей. Централизованное управление доступом как одна из задач информационной безопасности.
2. Методы и средства управления сетевым оборудованием. Общая схема построения системы управления сетью. Методы конфигурирования сетевых устройств. Протоколы взаимодействия с сетевым оборудованием. База данных. Представление схемы сети в базе данных.
3. Практическая реализация системы. Средства разработки. Средства программного моделирования сетей. Схема сети. Алгоритм поиска пути.
4. Безопасность и экологичность проекта. Расчёт уровня шума на рабочем месте. Экологичность работы. Расчёт электромагнитных излучений. Организация и обеспечение пожаробезопасности. Выводы.
5. Технико-экономическое обоснование. План-график проекта. Расчёт затрат на разработку проекта.

Заключение: Заключение должно содержать обобщенные результаты проведенной работы в соответствии с поставленной целью и задачами, необходимо указать чем завершается работа – усовершенствованием, модернизацией, дать свои предложения.

Перечень графического и иллюстративного материалов:

1. Структурная схема системы
2. Схема базы данных
3. Общая схема сети
4. Модель сети в GNS3
5. Представление схемы сети в виде графа связей между сетевыми интерфейсами

Руководитель ВКР _____ доц., к.т.н. В.В. Галушка
(подпись, дата)

Задание к исполнению принял _____ В.В. Фомичев
(подпись, дата)

Аннотация

Выпускная квалификационная работа посвящена вопросам построения системы для централизованного управления сетевым оборудованием телекоммуникационной системы. В ней рассматриваются методы и протоколы удалённой передачи команд устройствам сети, хранения списка этих устройств и связей между ними в базе данных, а также автоматизации действий администратора по заданию настроек конфигураций сетевого оборудования. В работе предложен проект программной системы, позволяющей формировать правила фильтрации пакетов для цепочки маршрутизаторов для пропускания определённого трафика между частями телекоммуникационной сети.

Объем текстового материала 82 листа (А4), количество иллюстраций 13, таблиц — 11, использованных источников — 20.

Abstract

The final qualifying work is devoted to the issues of building a system for centralized management of network equipment of a telecommunications system. It discusses methods and protocols for remote transmission of commands to network devices, storage of a list of these devices and links between them in a database, as well as automation of administrator actions to set network equipment configuration settings. The paper proposes a software system project that allows you to create packet filtering rules for a chain of routers to pass certain traffic between parts of a telecommunications network.

The volume of textual material 82 pages (A4), the number of images 13, tables — 11, used sources — 20.

Содержание

Введение

Современные компьютерные сети представляют собой сложную систему со множеством компонентов, которые включают в себя конечные устройства, коммуникационное оборудование и линии связи. Функционирование такой системы полностью зависит от коммуникационного оборудования, которое составляет основу инфраструктуры любой сети передачи данных. В свою очередь данное оборудование требует правильного подбора, конфигурирования, периодического мониторинга состояния, а также внесения изменений в настройки, отражающих изменения самой телекоммуникационной системы.

Перечисленные функции выполняются системным администратором исходя из потребностей в уровне качества сетевых сервисов, финансовых и других ограничений, а также требований к информационной безопасности. При этом спектр сетевого оборудования широк, а методы его настройки значительно отличаются и зависят от типа устройства, его производителя, версии операционной системы и других, менее значимых характеристик. Работа администратора может также осложняться из-за территориальной распределённой сети, отсутствия проектной документации и ошибок, допущенных на этапе монтажа сетевого оборудования и линий связи, затрудняющих физический доступ к ним.

В результате актуальной становится проблема создания системы, позволяющей компенсировать описанные выше недостатки за счёт автоматизации части рутинных операций и процедур поиска активного сетевого оборудования. Внедрение такой системы должно упростить и, соответственно, ускорить работу администратора по управлению как информационно-телекоммуникационной сетью в целом, так и каждым конкретным сетевым устройством, входящим в неё.

Таким образом целью данной выпускной квалификационной работы является сокращение времени, затрачиваемого на задание параметров конфигурации сетевого оборудования.

Общая задача работы — создание проекта системы, позволяющей осуществлять централизованное управление сетью на основе автоматизации процессов задания настроек для коммутационного оборудования, хранения их резервных копий, а также мониторинга состояния сети, и предоставляющей администратору удобный интерфейс для выполнения указанных действий.

Для её достижения были решены следующие частные задачи:

— проведён анализ существующих средств конфигурирования устройств в сети в том числе с использованием протоколов удалённого доступа;

— выполнен обзор литературы командам для настройки сетевого оборудования;

— проведён анализ технологий хранения историй версий для создания резервных копий конфигураций оборудования;

— предложены методы программного формирования и передачи команд сетевому оборудованию;

— описан вариант программной реализации автоматического задания параметров для маршрутизаторов;

— опробованы и протестированы предложенные методы.

Объектом исследования в выпускной квалификационной работе является сетевое оборудование, предметом — методы его настройки с использованием средств удалённого доступа.

1 Проблема централизованного управления сетью

1.1 Технологии компьютерных сетей

Рассматривая в данной работе задача централизованного управления сетью в первую очередь ориентирована либо на локальные сети, принадлежащие какой-либо одной организации, либо на сети несколько большего масштаба, которые, однако, могут быть представлены как совокупность небольших локальных сетей, каждая из которых находится под собственным административным управлением, в рамках которого и осуществляются функции по контролю за её функционированием.

В соответствии с моделью OSI за функционирование локальных сетей отвечает канальный уровень [1]. Он описывает аспекты сетевого взаимодействия, касающиеся процессов организации связи между устройствами, соединёнными напрямую, то есть без других, промежуточных, устройств.

В процессе развития компьютерных сетей существовало большое количество протоколов канального уровня, которые реализовывали совершенно разные технологии. У каждой из них были свои достоинства и недостатки, которые определяли её популярность и степень распространённости. Однако, наличие большого числа различных, а значит несовместимых, технологий усложняло процесс объединения сетей, что привело к необходимости использования единого стандарта для обеспечения совместимости оборудования. В силу различных технических и исторических причин таким стандартом стал Ethernet, который к настоящему времени вытеснили все остальные сетевые технологии.

Ethernet — это наиболее распространённая на сегодняшний день технология организации локальных сетей [2]. Она описывает реализацию двух первых уровней модели OSI — проводные соединения и электрические сигналы (физический уровень), а также форматы блоков данных и протоколы управления доступом к сети (канальный уровень).

Название Ethernet произошло от двух английских слов — ether (эфир) и net (сеть). Первые реализации Ethernet использовали концепцию общего эфира. Каждый компьютер посылает данные в общий кабель и указывает, кому они адресованы. Данные могут дойти до всех компьютеров сети, но обрабатывает их только тот, которому они предназначены. Остальные узлы чужие данные игнорируют. Такая работа аналогична эфиру радиостанций и была заимствована из существовавших на момент разработки Ethernet технологий радиосетей с пакетной коммутацией.

В ранних версиях Ethernet коаксиальный кабель являлся носителем общего электромагнитного эфира. Компьютеры подключались к общему кабелю с помощью специальных коннекторов по топологии общая шина. Каждый компьютер отправлял в шину электрические сигналы, а все остальные узлы их получали. При получении, компьютер должен был определить, кому реально этот сигнал адресован, и, соответственно, свои сигналы обработать, а чужие — проигнорировать.

Первая версия Ethernet была рассчитана на скорость 3Мбит/с. Она использовала толстый коаксиальный кабель, метод управления доступом — CSMA/CD и полудуплексный режим работы, то есть узел не мог одновременно передавать и принимать информацию. Размер пакета от 72 до 1526 байт Количество узлов в одном разделяемом сегменте сети ограничено предельным значением в 1024 рабочих станции. Однако сеть, построенная на одном разделяемом сегменте, становится неэффективной задолго до достижения предельного значения количества узлов, в основном по причине полудуплексного режима работы.

Несмотря на то, что Ethernet на коаксиальном кабеле уже давно не используется, механизм адресации данных и концепция общего доступа к разделяемой среде сохранились без изменений.

К сегодняшнему дню разработано большое количество стандартов Ethernet сетей, которые отличаются скоростью передачи, типами кабелей и оборудованием. Их можно разделить на несколько групп:

— 10 Мбит Ethernet;

— Fast Ethernet;

— Gigabit Ethernet;

— Прочие (перспективные) реализации Ethernet.

Актуальными на сегодняшний день являются сетевые технологии из семейств Fast Ethernet и Gigabit Ethernet:

— 100BASE-T — общий термин для обозначения стандартов, использующих в качестве среды передачи данных витую пару с длиной сегмента до 100 метров. Включает в себя стандарты 100BASE-TX, 100BASE-T4 и 100BASE-T2.

— 100BASE-TX (другое обозначение IEEE 802.3u) — развитие стандарта 10BASE-T для использования в сетях топологии «звезда». Предполагает использование витой пары пятой категории, хотя фактически в ней задействованы только две пары проводников из четырёх, по которым поддерживается одновременная передача данных в обоих направлениях.

— 1000BASE-T, IEEE 802.3ab — основной гигабитный стандарт, опубликованный, также использующий витую пару категории 5e. При этом, в передаче данных участвуют уже все четыре пары, каждая пара используется одновременно для передачи данных в обоих направлениях со скоростью 250 Мбит/с.

Перечисленные стандарты и технологии реализуются сетевым оборудованием. При этом, как показано на рисунке 1.1, основу локальной сети составляют коммутаторы, а на её границе используются маршрутизаторы.

Сетевой коммутатор (жарг. свитч от англ. switching hub, switch) — устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного или нескольких сегментов сети [3].

В отличие от концентратора, который распространяет трафик от одного подключенного устройства ко всем остальным, коммутатор передаёт данные только непосредственно получателю (исключение составляет широковещательный трафик всем узлам сети и трафик для устройств, для

которых не известен исходящий порт коммутатора). Это повышает производительность и безопасность сети, избавляя остальные сегменты сети от необходимости (и возможности) обрабатывать данные, которые им не предназначались.

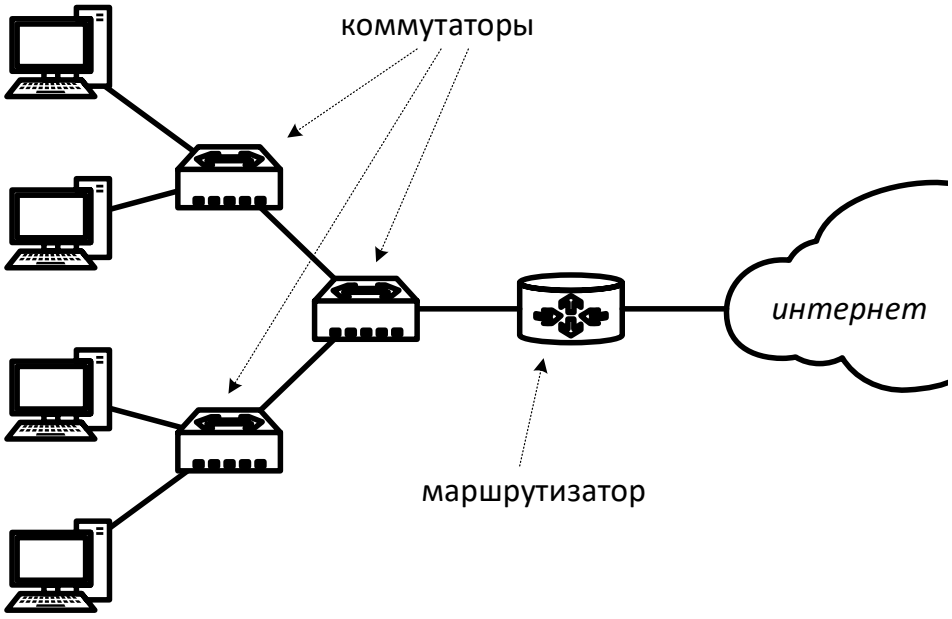


Рисунок 1.1 — Типовая схема локальной сети

Коммутатор хранит в памяти таблицу коммутации, в которой указывается соответствие MAC-адреса узла порту коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения. В этом режиме поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. При этом коммутатор анализирует кадры (фреймы) и, определив MAC-адрес хоста-отправителя, заносит его в таблицу на некоторое время. Впоследствии, если на один из портов коммутатора поступит кадр, предназначенный для хоста, MAC-адрес которого уже есть в таблице, то этот кадр будет передан только через порт, указанный в таблице. Если MAC-адрес хоста-получателя не ассоциирован с каким-либо портом коммутатора, то кадр будет отправлен на все порты, за исключением того порта, с которого он был получен. Со временем коммутатор строит таблицу для всех активных MAC-адресов, то есть узнаёт обо всех устройствах, подключённых к сети, и в результате трафик локализуется.

При объединении в сеть трех и более компьютеров возникает проблема их адресации, а точнее адресации отдельных сетевых интерфейсов, так как у одного компьютера может их может быть несколько.

Каждое конечное устройство в сети имеет 2 адреса: MAC и IP.

На канальном уровне, который обеспечивает доступ к среде и передачу кадра, для идентификации сетевых интерфейсов узлов сети используются регламентированные стандартом IEEE 802.3 уникальные 6-байтовые адреса, называемые MAC-адресами [4].

Старшие 3 байта MAC-адреса представляют собой так называемый уникальный идентификатор организации (Organizationally Unique Identifier, OUI), который централизованно выделятся производителям сетевого оборудования IEEE (Институтом инженеров электроники и электротехники). На сайте IEEE имеется возможность поиска информации о производителе по значению OUI.

Младшие 3 байта представляют собой организационно уникальный адрес (Organizationally Unique Address, OUA), который назначается производителем каждому выпущенному им контроллеру сетевого интерфейса.

MAC-адреса являются примером так называемой плоской адресации, при которой множество адресов никак не структурировано. В противоположность ей, при иерархической организации адресное пространство структурируется в виде вложенных друг в друга подгрупп, которые, последовательно сужая адресуемую область, в конце концов, определяют отдельный сетевой интерфейс. Примером иерархической адресации являются IP-адреса.

На практике обычно применяют сразу несколько схем адресации, так что сетевой интерфейс компьютера может одновременно иметь несколько адресов-имен. Каждый адрес задействуется в той ситуации, когда соответствующий вид адресации наиболее удобен. А для преобразования адресов из одного вида в другой используются специальные вспомогательные протоколы, которые называют протоколами разрешения адресов.

В заголовке IP-пакета для хранения IP-адресов отправителя и получателя отводятся два поля, каждое из которых имеет фиксированную длину 4 байта (32 бита). IP-адрес состоит из двух логических частей — номера сети и номера узла в сети.

Наиболее распространенной формой представления IP-адреса является запись в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками.

Другое рассматриваемое устройство — маршрутизатор, в классическом представлении нужен для трансляции пакетов между раздельными IP сетями [5]. Он позволяет решить вопрос объединения разрозненных LAN и предотвращения роста широковещательного трафика в одной большой локальной сети разделением её на сегменты. Разумеется, для правильного перенаправления трафика необходимо знать, куда его отправлять, то есть выстраивать маршрут.

Современные модели маршрутизаторов работают выше 3-го уровня модели OSI. Помимо трансляции IP пакетов из одной сети в другую, эти устройства часто имеют функции управления трафиком, например, возможность закрывать/открывать TCP или UPD порты, выполнять функции NAT, перенаправления портов и так далее.

Однако по умолчанию данные технологии не активированы на большинстве маршрутизаторов, а для их правильной работы требуется задание некоторого количества параметров, согласованных с параметрами других (чаще всего соседних) устройств. Решение данной задачи осуществляется в процессе администрирования сети.

1.2 Задачи администрирования компьютерных сетей

Администрирование — это процесс, целью которого является приведение системы (в том числе телекоммуникационной сети) в соответствие целям и задачам предприятия или организации [6].

Для достижения этой цели системное управление должно быть построено таким образом, чтобы минимизировать необходимое время и ресурсы, направляемые на управление системой и, в то же время, максимизировать доступность, производительность и продуктивность системы.

Управление сетью — целенаправленное воздействие на сеть, осуществляемое для организации её функционирования по заданной программе. Оно включает следующие процедуры:

— включение и отключение системы, каналов передачи данных, терминалов;

— диагностика неисправностей;

— сбор статистики;

— подготовка отчётов и т.п.

С точки зрения модели OSI управление сетью подразделяется на управление:

— конфигурацией;

— отказами;

— безопасностью;

— трафиком;

— учётом.

Традиционные методы управления основаны на использовании правил. Они предписывают системе управления в компьютерной сети предпринимать определённые действия (например, выдать предупреждающее сообщение на управляющую консоль) при наступлении определённых событий (превышение интенсивностью трафика заранее определённого порогового значения и др.).

Приемлемая в небольших сетях, методология управления на основе правил сталкивается с множеством препятствий в крупных сетях: сетях вычислительных центров и корпоративных информационных сетях. Основная трудность обусловлена тем, что функционирование мощной вычислительной среды может описываться многими тысячами параметров.

Управление локальной вычислительной сетью становится необходимым, когда у сетевого администратора возникает необходимость, а также возможность оперировать ее общим представлением. Обычно это касается сетей, имеющих сложную архитектуру. Администрирование компьютерных сетей подразумевает переход от управления работой отдельных устройство к анализу трафика на различных сетевых участках, управлению логической конфигурацией сети, ее рабочими параметрами. Таким образом, задачи администрирования можно разбить на две основные группы:

— контроль над функционированием сетевого оборудования;

— управление работой сети в целом.

Ключевой целью администрирования локальной вычислительной сети становится достижение и поддержание параметров работы информационной системы, наиболее точно соответствующих потребностям пользователя, который может оценить ее работу не по характеристикам трафика, используемым протоколам, скоростью отклика сервера на запросы и особенностям сценариев, а по работе программного обеспечения, постоянно запускаемого на его персональном компьютере.

Общепринято считать, что управление сетью наиболее целесообразно производить с одного рабочего места. Необходимость контроля за работой сетевых устройств с использованием одного компьютера способствовала разработке разных архитектур платформ и программного обеспечения для администрирования. Наиболее распространенным среди них стала распределенная двухуровневая архитектура «менеджер-агенты». Приложение-менеджер работает с использованием управляющей консоли, непрерывно взаимодействуя с запускаемыми в различных сетевых устройствах модулями (агентами). Агенты отвечают за сбор информации о параметрах функционирования ресурсов, а также за внесение в конфигурацию определенных изменений по запросу менеджера и предоставление ему различных административных сведений.

Система управления сетью (Network management system) — аппаратные и (или) программные средства, применяемые для мониторинга и управления узлами сети [7]. Программное обеспечение системы управления сетью состоит из агентов, локализующихся на сетевых устройствах и передающих информацию сетевой управляющей платформе.

Платформа управления сетью (Network management platform) — комплекс программ, предназначенных для управления сетью и входящими в неё системами [7]. Для работы с платформой администратору предоставляется одна или несколько абонентских систем (консолей). Обычно платформа создаётся на базе протокола SNMP. Платформа обеспечивает:

— контроль работы устройств и состояния кабелей;

— контроль деловых процедур;

— контроль других аспектов функционирования сети.

Чтобы компьютерная сеть могла эффективно выполнять свои функции, необходимо централизованно контролировать состояние основных её элементов, выявлять и разрешать возникающие проблемы, выполнять анализ производительности и планировать развитие сети и др. Эти виды работ являются основными задачами администрирования сетей.

Сетевое администрирование (Network Management) возникает, когда у администратора сети появляется потребность и возможность оперировать единым представлением сети, как правило, это относится к сетям со сложной архитектурой.

При этом осуществляется переход от управления функционированием отдельных устройств к анализу трафика в отдельных участках сети, управлению её логической конфигурацией и конкретными рабочими параметрами, причём все эти операции целесообразно выполнять с одной управляющей консоли. Задачи, решаемые в данной области, разбиваются на две группы [7]:

1) контроль за работой сетевого оборудования,

2) управление функционированием сети в целом.

Конечной целью управления сетью является достижение параметров функционирования ИС, соответствующих потребностям пользователей. Пользователи оценивают работу ИС не по характеристикам сетевого трафика, применяемым протоколам, времени отклика серверов на запросы определённого типа и особенностям выполняемых сценариев управления, а по поведению приложений, ежедневно запускаемых на их настольных компьютерах.

Интегрированная система управления сетью (Integrated network management system, INMS) – система управления, обеспечивающая объединение функций, связанных с анализом, диагностикой и управлением сетью. Таким образом, эволюция средств и систем администрирования непосредственно связана с развитием основных информационных технологий.

Проекты развития административных механизмов обычно включают в себя задачи постановки стратегического управления, разработки политики информационного обеспечения и доступа к информационным ресурсам, а также программно-аппаратным устройствам, системам и комплексам, постановки и развития системы, совершенствование непрерывного управления.

1.3 Требования к безопасности сетей

На сегодняшний день многие организации в значительной степени полагаются на компьютерные сети для эффективного и продуктивного обмена информацией. Организационные компьютерные сети в настоящее время становятся большими и повсеместными. Предполагая, что у каждого сотрудника есть выделенная рабочая станция, в крупной компании будет несколько тысяч рабочих станций и много серверов в сети.

Вполне вероятно, что эти рабочие станции не могут управляться централизованно и не имеют защиты по периметру. Они могут иметь различные операционные системы, аппаратные средства, программное обеспечение и протоколы с разным уровнем осведомленности среди пользователей. Если эти тысячи рабочих станций в сети компании будут напрямую подключены к

Интернету, то такой вид незащищенной сети становится целью для атаки, которая содержит ценную информацию и отображает уязвимости.

Для защиты современных бизнес-сетей и ИТ-инфраструктуры требуется комплексный подход, четкое понимание всех уязвимых мест, а также соответствующих необходимых мер защиты [8]. И хотя одно такое знание не сможет предотвратить все попытки проникновения в сеть или атаки на систему, но оно даст возможность сетевым администраторам устранить определенные глобальные проблемы, значительно сократить возможный ущерб и позволит быстро находить пробелы в защите. Из-за постоянного растущего числа и сложности атак злоумышленников, необходимы бдительные методы обеспечения безопасности систем, как для больших, так и для малых предприятий.

Для эффективной работы защищенное предприятие, независимо от того, большое оно или малое, должно иметь собственный универсальный и всеобъемлющий подход к обеспечению безопасности.

Сетевая безопасность — список требований, рекомендаций и политик, которые используются в сетевой инфраструктуре для повышения ее уровня защиты и отказоустойчивости.

Вторая важная функция анализа работы инфраструктуры компании и предотвращения несанкционированного доступа (НСД) к информационным ресурсам со стороны злоумышленников [9].

Независимо от масштаба и типа бизнеса (малый, средний или крупный) использование сетевой инфраструктуры подразумевает интеграцию аппаратных и программных решений, которые обеспечивают работоспособность и безопасность сети.

Выделяют 4 основных принципа проектирования сетевой безопасности на объекте информатизации [10]:

— Защита оборудования, подключенного к сетевой инфраструктуре. В качестве защитных мер используют антивирусные решения с регулярным

обновлением баз, межсетевые экраны с фильтрацией трафика и блокировкой нежелательных абонентов и т. д.

— Оборудование должны быть отказоустойчивым и предусматривать возможность быстрого восстановления. Подразумевается наличие дублирующих компонентов в критически важных узлах.

— Систематический мониторинг всей инфраструктуры компании для обнаружения уязвимых точек. Также система должна предоставлять подробную информацию о любом программном или аппаратном компоненте оборудования.

— Постоянный мониторинг пропускной способности сетевого канала. Это гарантирует своевременную блокировку нежелательного трафику, а также позволяет осуществить балансировку нагрузки в ручном режиме.

— Критически важные узлы инфраструктуры организации должны обеспечивать высокую доступность при любой угрозе либо атаке на компанию. Это достигается за счет создания второй независимой площадки (ЦОДа), которая реплицирует данные с первой в синхронном режиме.

При разработке сетевой архитектуры необходимо учитывать использование различных средств защиты информации.

Защитные методы делятся на четыре группы:

— организационные;

— технические или аппаратные;

— программные;

— аппаратно-программные.

Все эти инструменты предназначены для создания трудностей для НСД к ЛС.

Основные барьеры для злоумышленников:

— физическое препятствие, которое не позволяет третьему лицу взаимодействовать с элементами сети;

— система контроля и управления доступом, регулирующая уровни прав пользователей;

— использование криптографических средств защиты информации (шифрование данных);

— регламентация действий персонала;

— принятие дисциплинарных, гражданских и даже уголовных мер для защиты конфиденциальной информации.

Средства контроля и управления доступом в целях защиты информации включают [11]:

— механизмы идентификации пользователей и элементов системы, основанные на текстовых (логин, пароль) или технических (смарт-карта, токен) принципах;

— распределение прав доступа в зависимости от служебного ранга пользователя;

— регламентирование разрешенных работ в сети для каждой категории пользователей;

— фиксацию действий пользователей;

— определение реакций (отключение системы, сигнализация) при выявлении попыток НСД.

Организационные методы традиционно включают в себя внутренние нормативные акты, регламентирующие порядок работы с информацией [12]. Это положения о коммерческой тайне, о том, как работать с информационными ресурсами, и о том, как получить доступ к документам. Однако организационные меры не ограничиваются нормативными актами и положениями, они также могут носить характер действий.

Организационные средства защиты информации (СЗИ) включают в себя:

— ограничение доступа в рабочие помещения, внедрение системы пропусков;

— разграничение прав пользователей в работе с массивами данных;

— выделение специальных автоматических рабочих станций (АРМ) без подключения к Интернету для обработки ценной информации;

— специальная процедура учета и хранения съемных носителей информации;

— размещение АРМ таким образом, чтобы экран компьютера и клавиатура не были видны другим сотрудникам и посторонним лицам;

— контроль за выводом информации на принтер, создание защищенных зон для печати;

— контроль за распечатанными экземплярами документов, содержащими критичную информацию;

— в случае выхода оборудования из строя уничтожение данных на жестких дисках перед отправкой его в ремонт;

— установку запирающих устройств на корпус компьютера.

1.4 Централизованное управление доступом как одна из задач информационной безопасности

В самом простом случае управление доступом служит для определения того, имеет ли разрешение пользователь на доступ к некоторому элементу сети. При повышении избирательности управления доступом реально добиться разрешения/запрета доступа к отдельным элементам для отдельных пользователей независимо от других. И наконец, имеются все возможности для расширения механизмов управления так, чтобы были охвачены объекты внутри сетевого элемента, например, файлы или процессы.

Нарушение полномочий выражается в следующем:

— когда отправитель запрашивает и выдает команды, которые не включены в список получателей сетевого элемента;

— несоответствие между значениями представленного и хранимого на объекте-получателе;

— получение зашифрованной информации, которую невозможно расшифровать, и т. д.

Во всех вышеперечисленных случаях все дальнейшие действия прекращаются, и объект управления безопасности получает сообщение о попытке НСД, в котором указывается имя объекта-отправителя, дата и время события и его характер. Как только служба безопасности получает подобное сообщение, она сразу же реагирует на него и начинает расследование произошедшего, выясняет причину события. В случае, если причина оказалась случайной, то дальнейшее решение вопроса поручается службе обеспечения надежности, если преднамеренная, то согласно должностной инструкции, разработанной организацией, выполняются соответствующие действия.

Существует три типа управления доступом:

— централизованное управление. Администрация организации устанавливает, а ввод и контроль полномочий осуществляется с соответствующего объекта управления представителем службы безопасности информации;

— иерархическое децентрализованное управление. Центральная организация, которая устанавливает полномочия, может передать часть своих полномочий организациям, находящимся в подчинении, имея при этом возможность отменять или пересматривать решения подчиненной организации или лица;

— индивидуальное управление. Отдельное лицо может создавать свою информацию, защищенную от НСД. Владелец информации по своему усмотрению может разрешить другим пользователям доступ к ней и передавать право собственности.

В последнее время российский рынок все больше и больше наполняется корпоративными (частными) цифровыми сетями связи, которые в прошлом использовались лишь в оборонных отраслях для передачи секретной информации. Целью таких сетей является обеспечение закрытой связи между пользователями, имеющих общие корпоративные интересы.

Единое централизованное управление поможет реализовать общую политику безопасности, а также обеспечивает своевременное и оперативное отражение вносимых в неё изменений на реальное оборудование сети.

Под политикой безопасности понимается совокупность норм и правил, регламентирующих процесс обработки информации, выполнение которых обеспечивает защиту от определенного множества угроз и составляет необходимое условие безопасности системы. Формальное выражение политики безопасности называют моделью безопасности.

Основная цель создания политики безопасности системы и описания ее в виде формальной модели — это определение условий, которым должно подчиняться поведение системы, выработка критерия безопасности и проведение формального доказательства соответствия системы этому критерию при соблюдении установленных правил и ограничений.

Технологии защиты телекоммуникационных систем начали развиваться относительно недавно, но сегодня уже существует значительное число теоретических моделей, позволяющих описывать различные аспекты безопасности и обеспечивать средства защиты формально подтвержденной алгоритмической базой.

Кроме того, формальные модели безопасности позволяют решить еще целый ряд задач, возникающих в ходе проектирования, разработки и сертификации защищенных систем, поэтому их используют не только теоретики информационной безопасности, но и другие категории специалистов, участвующих в процессе создания и эксплуатации защищенных информационных систем (производители, потребители, эксперты).

Модели безопасности обеспечивают системотехнический подход, включающий решение следующих важнейших задач:

— выбор и обоснование базовых принципов архитектуры защищенных автоматизированных систем, определяющих механизмы реализации средств и методов защиты информации;

— подтверждение свойства защищенности разрабатываемых систем путем формального доказательства соблюдения политики безопасности (требований, условий, критериев);

— составление формальной спецификации политики безопасности как важнейшей составной части организационного и документационного обеспечения разрабатываемых защищенных компьютерных систем.

2 Методы и средства управления сетевым оборудованием

2.1 Общая схема построения системы управления сетью

Рассматриваемая в данной выпускной квалификационной работе система в соответствии со сложившимся на сегодняшний день подходом к проектированию приложений должна иметь модуль структуру, при которой каждый компонент независим от остальных и отвечает за выполнение заданного набора функций. Взаимодействие между модулями осуществляется через определённый разработчиком интерфейс.

Общая схема разрабатываемой системы представлена на рисунке 2.1. Она включает в себя следующие модули:

— пользовательский интерфейс;

— модуль взаимодействия с устройствами;

— база данных.

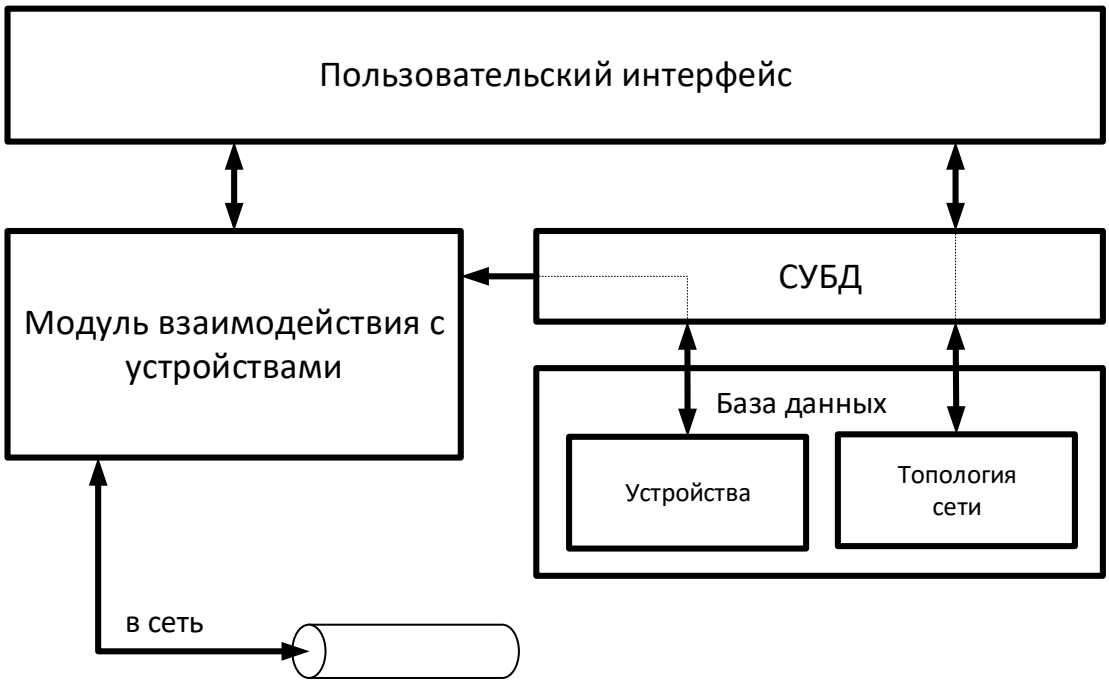


Рисунок 2.1 — Структурная схема системы

Основным модулем системы является интерфейс, который помимо обеспечения взаимодействия с пользователями, также выполняет функции организации взаимодействия остальных модулей, такие как:

— извлечение информации из базы данных для формирования на её основе наглядного представления схемы сети;

— приём от пользователя команд на изменение параметров функционирования сети;

— формирование на основе полученных команд списка оборудования, для которого должна выполняться переконфигурация, а также необходимых для этого операций в зависимости от типа и модели каждого устройства;

— передача операций следующему модулю, отвечающему за взаимодействие с сетевым оборудованием.

Модуль взаимодействия с сетевым оборудованием предназначен для формирования команд, которые должны отправляться каждому устройству. Сами команды формируются исходя из перечня операций, которые необходимо выполнить на устройстве для того, чтобы привести его настройки в соответствие требуемым параметрам функционирования сети в целом.

Основные функции данного модуля включают в себя:

— реализация одного из протоколов удалённого доступа;

— осуществление, при необходимости, аутентификации и авторизации на сетевом оборудовании;

— контроль выполнения команд и обработка результатов.

Для хранения списка устройств, их характеристик, а также доступных для настройки параметров в системе предусмотрена база данных. Она предоставляет наиболее эффективный способ представления данных в структурированном виде не только для их хранения, но и дальнейшего использования, включая обновление и извлечение.

Базу данных предлагаемой системы можно условно разделить на 2 части: первая, описанная ранее, предназначена для хранения информации об отдельных

устройствах сети, а вторая — для представления топологии сети, то есть хранения связей между её узлами, которые представляют собой граф.

Среди проблем, связанных с представлением и обработкой графов в базах данных, в практике разработки информационных систем чаще всего возникают задачи хранения деревьев, предназначенных для описания иерархических систем и структур. Рассмотренные в 1 главе технологии Ethernet также предполагают использование топологии иерархическая звезда (звезда из звёзд), которая полностью соответствует математическому определению дерева как одного из видов графов.

Сложность задачи хранения графов в базах данных обусловлена в первую очередь тем, что реляционные базы не приспособлены к хранению иерархических структур и представляют собой простые списки. Иерархические же данные имеют связь родитель-наследники, которая напрямую не реализована в реляционной структуре. Однако существует несколько моделей, позволяющих отразить древовидную структуру данных на таблицы реляционной БД. К ним относятся:

— список смежности (adjacency list);

— материализованный путь (materialized path);

— таблица связей (closure table);

— вложенные множества (nested sets).

Отдельно следует упомянуть возможность использования нереляционных баз данных для хранения графов.

Нереляционные базы данных (NoSQL от англ. not only SQL — не только SQL) — термин, обозначающий ряд подходов, направленных на реализацию систем управления базами данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. В базах данных такого рода применяются модели хранения, оптимизированная под конкретные требования типа хранимых данных.

Одной из моделей данных для NoSQL баз данных является графовая модель, являющаяся обобщением сетевой модели данных, использовавшейся на ранних этапах развития систем управления базами данных. В соответствии с этой моделью, база данных состоит из наборов записей, которые связаны между собой так, что записи могут содержать явные ссылки на другие наборы записей. Тем самым наборы записей образуют сеть.

Современные графовые базы данных предоставляют те же возможности по реализации произвольных связей между данными различной структуры, что и сетевые, но с использованием более новых технологий программирования и в расчёте на более производительное аппаратное обеспечение. Однако графовые базы данных предназначены не столько для хранения самих графов, особенно с большим числом вершин и рёбер (это можно осуществлять и в реляционных БД с использованием одного из перечисленный ранее способов), сколько для структуризации данных со сложной структурой на основе методов теории графов.

Хранение в базе данных графа, представляющего топологию сети, позволяет осуществляет, в зависимости от задачи, поиск и выбор одного или нескольких необходимых устройств для их дальнейшего централизованного конфигурирования.

2.2 Методы конфигурирования сетевых устройств

Большинство моделей сетевого оборудования различных производителей поддерживает несколько вариантов подключения для выполнения настроек:

— подключение через консольный порт,

— web-интерфейс,

— протоколы удалённого доступа.

Консольный порт — это отдельный физический разъём в сетевом оборудовании, который используется для доступа к интерфейсу командной

строки устройства и его конфигурированию с помощью программ эмуляции терминала.

Для подключения через консольный порт необходим специальный консольный кабель, имеющий с одного конца стандартный сетевой разъём 8P8C, а с другой — разъём DB9m для подключения к COM-порту.

Через консольный порт можно получать доступ к терминальным линиям IOS устройств Cisco. Однако это может быть потенциальной угрозой сети, поскольку любой может получить свободный доступ к устройству или пользователь может получить доступ к устройству, используя тот же пароль, который локально хранится на устройстве. Аутентификации пользователей нет.

Подключение через консольный порт используется в следующих ситуациях:

— первичная настройка оборудования, когда другие способы подключения только предстоит настроить;

— если другие способы подключения в данный момент недоступны по причине неисправности оборудования или инфраструктуры;

— вы находитесь непосредственно рядом с оборудованием и вам в данный момент наиболее удобен этот способ.

Веб-интерфейс (Web UI) — это веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством протокола HTTP и веб-браузера. Веб-интерфейсы получили широкое распространение в связи с ростом популярности всемирной паутины и, соответственно, повсеместного распространения веб-браузеров.

Пользовательский веб-интерфейс предоставляет встроенный инструмент управления устройствами и является одним из видов графического интерфейса. Он обеспечивает возможность инициализировать устройство, упростить развертывание сети и управление ею, а также расширить возможности пользователя. Для многих устройств он активен по умолчанию, благодаря чему нет необходимости что-либо активировать или устанавливать какую-либо

дополнительные средства или модули. Главным преимуществом использования веб-интерфейса является возможность создания конфигураций, а также осуществления мониторинга и устранения неполадок в работе сетевых устройств без необходимости работы в командной строке (CLI), которая традиционно считает гораздо более сложной.

Классическим и наиболее популярным методом создания веб-интерфейсов является использование HTML с применением CSS и JavaScript'a. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах вызывает проблемы при разработке веб-приложений и их последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе интерфейса.

Основным преимуществом веб-интерфейсов является отсутствие необходимости установки дополнительного программного обеспечения, так как популярные операционные системы поставляются с уже установленным браузером.

В свою очередь недостатком веб-интерфейсов являются бóльшие требования к вычислительной мощности, больший объём передаваемого трафика, необходимость поддержки графического интерфейса на клиентском устройстве (как было отмечены выше веб-интерфейс является одним из видов графического интерфейса), а также большое разнообразие вариантов реализации веб-интерфейсов, значительно отличающихся друг от друга.

Таким образом, наиболее универсальным средством взаимодействия с сетевыми устройствами являются протоколы удалённого доступа, которые будут более подробно рассмотрены далее.

2.3 Протоколы взаимодействия с сетевым оборудованием

Существующее многообразие сетевого оборудования определяет наличие большого числа способов управления им: как локально, так и удалённо. Данные

способы включают в себя разнообразные пользовательские интерфейсы для локального администрирования и сетевого протоколы для удалённого.

Локальные настройки могут выполняться через графический пользовательский интерфейс (GUI — graphical user interface) или интерфейс командной строки (CLI — command line interface). В свою очередь сетевые протоколы предоставляют доступ к одному из указанных типов интерфейсов, а следовательно, их тоже можно разделить на протоколы, предоставляющие доступ в режиме удалённого рабочего стола, и протоколы для доступа в терминальном режиме, или режиме командной строки. К первой группе относятся RPD, VNC и множество других подобных протоколов, а ко второй — telnet и SSH.

Протоколы VNC и RDP используются для подключения к удаленному устройству, находящемуся, как правило в локальной сети. Оба протокола обеспечивают удаленного доступа к графическому пользовательскому интерфейсу устройства, позволяя выполнять действия, как если бы подключающийся пользователь находился прямо перед удаленным устройством.

И для VNC, и для RDP требуется программное обеспечение на стороне сервера и клиента. Сервер — это устройство, к которому нужно подключиться, а клиент — это подключающееся устройство. Чтобы клиент мог получить доступ к серверу, необходимо настроить серверное программное обеспечение для удаленного доступа. Программное обеспечение VNC-сервера работает на всех ОС. Программное обеспечение RDP-сервера установлено по умолчанию во всех поддерживаемых в настоящее время версиях Windows Server, а также в Windows 10 и более поздних версиях. Уже существует программное обеспечение RDP-сервера для Linux и macOS. Клиентское программное обеспечение VNC и RDP ещё более широко.

VNC использует протокол RFB (Remote Frame Buffer) для удаленного доступа к другим устройствам. RFB работает на уровне фреймбуфера или той части оперативной памяти, которая используется видеокартами для представления данных, отображаемых на экранах компьютеров. Фреймбуфер

присутствует во всех оконных операционных системах и приложениях, включая macOS, Windows и систему X Window, распространенную в Unix/Linux. Типичный сеанс VNC начинается с подключения клиента к порту 5900 на сервере. Как только соединение установлено, экран на сервере отправляется обратно клиенту, пиксель за пикселем. Нажатия клавиш и щелчки мыши также распределяются между клиентом и сервером.

RDP — это проприетарный протокол Microsoft, основанный на протоколе совместного использования приложений. Устройство Windows с RDP-сервером может получать запросы на удаленное подключение от устройств с RDP-клиентами либо через порт 3389 протокола управления передачей (TCP), либо через порт 3389 протокола пользовательских дейтаграмм (UDP). Вместо кадровых буферов RDP передает клиенту наборы инструкций для создания экрана сервера. После того, как инструкции будут полностью отправлены, на клиенте появится экран сервера.

Главным недостатком рассмотренных протоколов является необходимость наличия графического пользовательского интерфейса как на клиентском, так и на серверном устройстве. Соответственно они могут быть использованы только на очень ограниченном спектре сетевых устройств, а именно серверах с операционными системами Windows и десктопными дистрибутивами Linux. К большей части сетевого оборудования данные протоколы не применимы, в связи с чем, наиболее предпочтительными для реализации централизованного управления сетью является другая группа протоколов, обеспечивающих взаимодействие устройств путём передачи текстовых команд, а именно telnet и SSH.

Telnet (сокр. от англ. teletype network) — сетевой протокол для реализации текстового терминального интерфейса по сети (в современной форме — при помощи протокола TCP) [13]. Telnet является одним из самых старых протоколов, он был разработан в момент зарождения сети интернет, но используется и по сей день. Telnet — это прикладной протокол, который помогает пользователям общаться с удаленной системой. Он использует

текстовый интерфейс для создания виртуального терминала, позволяя администраторам получать доступ к приложениям на других устройствах.

Telnet является одним из стандартных протоколов стека TCP/IP для реализации службы виртуального терминала. Он позволяет устанавливать соединения с удаленной системой таким образом, чтобы она отображалась как локальная система.

SSH (англ. Secure Shell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений. С помощью SSH помимо стандартного подключения для управления удаленными устройствами можно выполнять команды, перемещать файлы между и многое другое.

Протокол SSH шифрует трафик в обоих направлениях, что помогает предотвратить прослушивание канала связи и кражу учётных записей. То есть он выполняет ту же основную функцию, что и Telnet, но делает это более безопасным способом. Этот протокол обеспечивает безопасный доступ даже в незащищенных сетях, устраняя многие уязвимости Telnet. Поскольку Telnet передает данные в виде открытого текста, он уязвим для атак типа «человек посередине», что позволяет злоумышленникам получать логины и пароли учетных записей пользователей и читать данные, которые передаются по сети.

Преимущества SSH:

— доступен бесплатно для некоммерческого использования;

— имеет с открытый исходный код, который постоянно дорабатывается и улучшается;

— через SSH могут предоставляться несколько сервисов, используя одно и то же соединение;

— позволяет безопасно туннелировать небезопасные приложения, такие как SMTP, IMAP, POP3 и CVS;

— позволяет осуществлять туннелирование портов работает для простых VPN-соединений;

— имеет надежную аутентификацию и безопасную связь по незащищенным каналам;

— позволяет пользователям безопасно входить на другой компьютер через небезопасную сеть;

— обеспечивает конфиденциальность и целостность передаваемых данных с помощью надежного шифрования;

— позволяет пересылать или шифровать другие сеансы связи на основе протоколов TCP/IP.

— позволяет просматривать содержимое каталогов, редактировать файлы и удаленно получать доступ к пользовательским приложениям базы данных.

Обобщённая сравнительная характеристика протоколов telnet и SSH приведена в таблице 2.1.

Таблица 2.1 — Сравнение протоколов telnet и SSH

Telnet	SSH
Стандартный протокол TCP/IP для службы виртуального терминала. Позволяет устанавливать соединение с удаленной системой таким образом, чтобы она отображалась как локальная система.	Программа для входа на другой компьютер по сети для выполнения команд на удаленном компьютере.
Всегда использует порт 23, специально выделенный для него.	По умолчанию работает на порту 22, который можно изменить.
Не имеет встроенных средств для аутентификации пользователей.	Безопасный протокол, использующий шифрование с открытым ключом для аутентификации он использует.
Подходит для частных сетей.	Подходит для публичных сетей.
Передает данные в виде обычного текста.	Для отправки данных использует зашифрованный формат.

Низкие требования к пропускной способности канала связи.	Требуется высокая пропускная способность.
Используется в операционных системах Linux и Windows.	Поддерживается всеми популярными операционными системы.

2.4 Средства управления доступом в телекоммуникационных системах

Access Control List или ACL — список управления доступом, который определяет, кто или что может получать доступ к объекту (программе, процессу или файлу), и какие именно операции разрешено или запрещено выполнять субъекту (пользователю, группе пользователей) [14].

Списки доступа содержат просто набор инструкций какие порты и адреса блокировать, а какие наоборот разрешить. Этих инструкций может от нескольких единиц до десятков.

При поступлении трафика проверка списка доступа начинается сверху вниз, то есть с первой инструкции. Как только будет найдено совпадение проверка списка прекратится и будет выполнено действие, указанное в инструкции (заблокировать или пропустить).

ACL разделяются на два типа:

– Стандартные (Standard): могут проверять только адреса источников.

– Расширенные (Extended): могут проверять адреса источников, а также адреса получателей, в случае IP ещё тип протокола и TCP/UDP порты.

Стандартный список доступа имеет следующий вид:

Router(config)#access-list <номер списка от 1 до 99> {permit | deny | remark} {address | any | host} [source-wildcard] [log]

– permit: разрешить;

– deny: запретить;

– remark: комментарий о списке доступа;

– address: запрещаем или разрешаем сеть;

- any: разрешаем или запрещаем всё;
- host: разрешаем или запрещаем хосту;
- source-wildcard: WildCard маска сети;
- log: включаем логгирование пакеты проходящие через данную запись ACL.

Расширенный список доступа имеет вид: Router(config)#access-list <номер списка от 100 до 199> {permit | deny | remark} protocol source [source-wildcard] [operator operand] [port <порт или название протокола> [established]

- protocol source: какой протокол будем разрешать или закрывать (ICMP, TCP, UDP, IP, OSPF и т.д);
- deny: запретить;
- operator:
  - A.B.C.D — адрес получателя;
  - any — любой конечный хост;
  - eq — только пакеты на этом порте;
  - gt — только пакеты с большим номером порта;
  - host — единственный конечный хост;
  - lt — только пакеты с более низким номером порта;
  - neq — только пакеты не на данном номере порта;
  - range — диапазон портов;
- port: номер порта (TCP или UDP), можно указать имя;
- established: разрешаем прохождение TCP-сегментов, которые являются частью уже созданной TCP-сессии.

2.5 База данных

В соответствии с представленной ранее общей схемой системы, её неотъемлемой частью является база данных, обеспечивающая функции хранения информации в структурированном виде, а также её поиска и извлечения.

Проектирование базы данных на первом этапе включает в себя построение её концептуальной модели, которая содержит сущности, атрибуты и связи между сущностями. На следующих этапах эта концептуальная модель должна быть преобразована в физическую, учитывающую особенности конкретной системы управления базами данных, а затем в схему данных реальной БД.

Концептуальная модель данных для разрабатываемой системы представлена на рисунке 2.2.

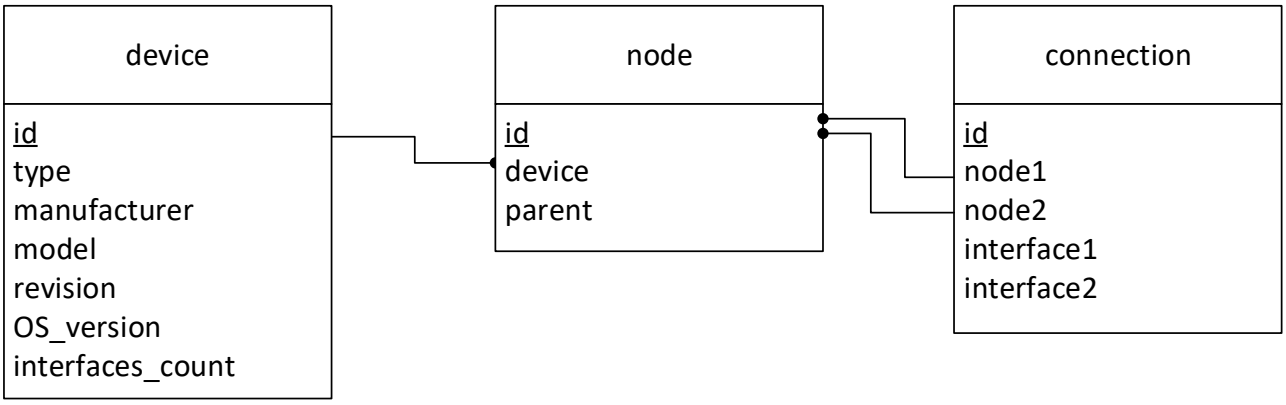


Рисунок 2.2 — Концептуальная модель данных

Построенная модель данных включает в себя перечисленные далее сущности.

Устройство (device) — представляет множество экземпляров сетевого оборудования одного вида, например, определенные модели коммутаторов конкретного производителя. Данная сущность имеет следующие атрибуты:

— id — уникальный идентификатор устройства, используемый в качестве первичного ключа. Здесь и далее все поля id являются суррогатными ключами, то есть представляют собой дополнительную служебную информацию, единственное назначение которой — служить первичным ключом. Значения этого поля не связаны с какими-либо характеристиками идентифицируемых объектов, а генерируется искусственно, как правило в виде последовательных значений.

— type — определяет тип оборудования: коммутатор, маршрутизатор, аппаратный сетевой экран и т.д.;

— manufacturer — фирма-производитель устройства;

— model — модель конкретного устройства (например, для маршрутизаторов Cisco может иметь значения 2811, 2800 и т.д., для коммутаторов: 2450, 2460 и т.д.);

— revision — ревизия, или версия аппаратной конфигурации, которая у некоторых производителей может отличаться в том числе для устройств одной модели;

— OS_version — версия операционной системы, установленной на устройстве (например, Cisco IOS);

— interfaces_count — количество сетевых интерфейсов, которое может использоваться для проверки правильности отображения топологии сети в других таблицах.

Следующие описываемые сущности предназначены для хранения в базе данных топологии сети.

Узел (node) — представляет собой физическое устройство, входящее в сеть. Атрибутами данной сущности являются:

— id — уникальный идентификатор узла;

— device — внешний ключ, ссылающийся на запись в таблице device (устройство), то есть показывающий какой именно тип оборудования, а также какая модель, версия и т.д., является данным узлом.

— parent — идентификатор родительского элемента, представляет собой ссылку на другую запись в той же таблице и позволяет отразить взаимосвязь элементов дерева.

Связь (connection) — определяет физическое сетевое соединение между двумя узлами (node). Она включает в себя следующие атрибуты:

— id — уникальный идентификатор соединения;

— node1 — внешний ключ, ссылающийся на элемент Node (узле), то есть указывающий на один конец (одну сторону) соединения;

— node2 — внешний ключ, ссылающийся на элемент Node (узле), то есть указывающий на другой конец (другую сторону) соединения;

— interface1 — номер или идентификатор сетевого интерфейса (порта) на оборудовании, соответствующем, указанному в node1;

— interface2 — номер или идентификатор сетевого интерфейса (порта) на оборудовании, соответствующем, указанному в node2.

2.6 Представление схемы сети в базе данных

Как было описано в п. 1.1 наиболее распространённой на сегодняшний день сетевой топологией является дерево. В соответствии с формальным определением дерево представляет собой ориентированный связный граф, который не имеет циклов. Однако на практике даже сети стандарта Ethernet могут иметь циклы при условии использования протокола STP, а значит известные методы представления деревьев в базах данных, такие как список смежности (Adjacency List), материализованный путь (Materialized Path), вложенные множества (Nested Sets) и таблица связей (Closure Table) нельзя использовать для хранения топологии сети.

Вместо перечисленных можно использовать какой-либо из универсальных способов представления графов, например, матрицу смежности. Проблема при этом заключается в том, что такие способы представления не соответствуют принципам хранения данных в реляционных базах данных, в результате чего появляется необходимость выполнения дополнительных преобразований над данными при их передаче из базы данных в приложение для дальнейшей обработки.

Наиболее подходящим для хранения в базе данных является реляционный список рёбер [15]. Он представляет собой отношение *R*, со следующей схемой:

$$R = (\underline{id}, v1, v2, len),$$

где *id* — суррогатный первичный ключ отношения,

v1 — номер одной из вершин,

v2 — номер вершины, с которой связана вершина v1,

len — длина ребра, связывающего вершины v1 и v2.

Таким образом, каждый кортеж данного отношения является каким-либо ребром графа.

При этом большинство стандартны алгоритмов компьютерной обработки графов рассчитано на работу с матрицей смежности, а значит необходим алгоритм выполняющий взаимные преобразована: матрицу смежности в реляционный список рёбер, и наоборот.

Общий подход такого алгоритма заключается в поэлементном обходе матрицы смежности, на каждом шаге которого проверяется значение в соответствующей ячейке матрицы (см. рис. 2.3). Нулевые значения пропускаются. Если значение в ячейке матрицы отлично от нуля, то формируется запрос к на добавление новой строки в таблицу базы данных, данные для которой вычисляются на основе индексов текущей ячейки, её содержимого и номера текущей итерации в соответствии с приведёнными выше формулами.

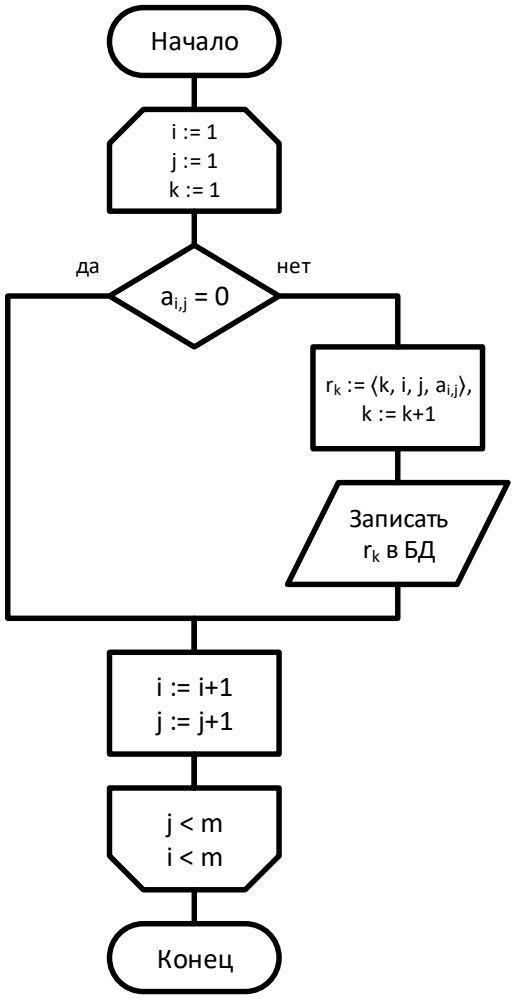


Рисунок 2.3 — Схема алгоритма преобразования матрицы смежности в реляционный список рёбер

Алгоритм обратного преобразования сводится к последовательному перебору всех строк r_i таблицы R (см. рис. 2.4). Информация в строке содержит номера двух вершин (в столбцах *v1* и *v2*), определяющих индексы элемента матрицы смежности, в который и будет записана длина ребра (*len*) также представленная в строке r_i.

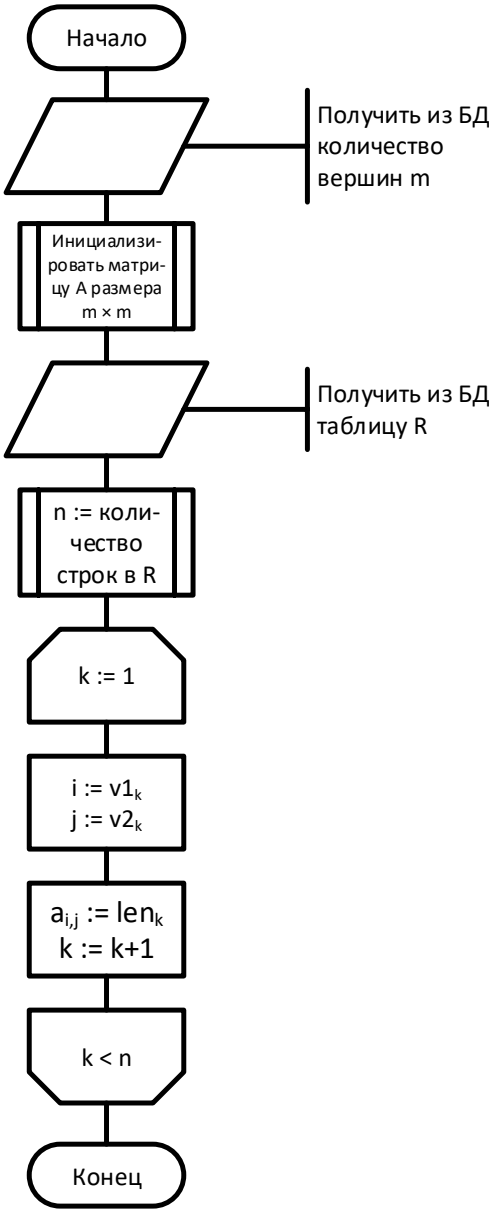
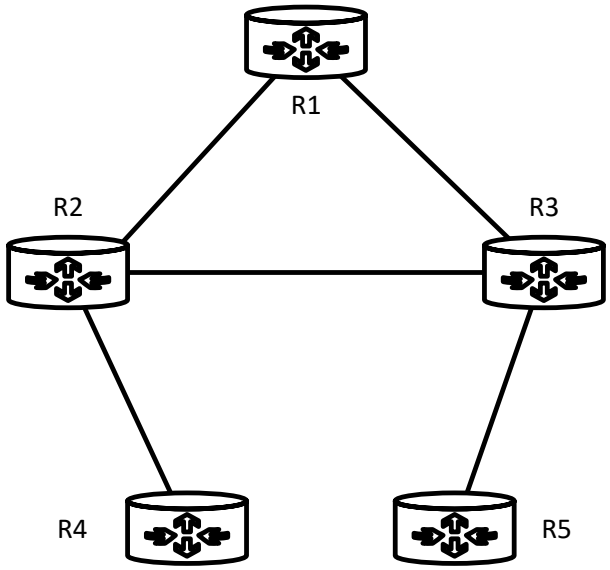


Рисунок 2.4 — Схема алгоритма преобразования реляционного списка рёбер в матрицу смежности

На рисунке 2.5 показан пример представления топологии сети в базе данных с использованием рассматриваемого метода. При этом все значения в столбце «len», обозначающем длину ребра равны 1, однако в более сложных реальных сетях величина условной длины ребра может быть заменена на его

«вес», обозначающий то на сколько эффективно будет передавать данных именно через ту линию связи, которую отражает ребро. Вес может рассчитываться на основе скорости линии связи, её загруженности, надёжности и т.д.



id	v1	v2	len
1	R1	R2	1
2	R1	R3	1
3	R2	R3	1
4	R2	R4	1
5	R3	R5	1

Рисунок 2.5 — Отображение сетевой топологии в базе данных в виде списка рёбер

3 Практическая реализация системы

3.1 Средства разработки

Большинство технологий программирования предусматривают абстрагирование, которое позволяет разработчикам сосредоточиться на решении предметных задач, меньше внимания уделяя особенностям аппаратного обеспечения и операционных систем.

Несмотря на то, что эти технологии значительно облегчают работу, они требуют от программиста осваивать массу материала. Также различные технологии разрабатывались без расчета на совместное использование, и разработчики сталкиваются с необходимостью решать непростые проблемы интеграции.

В отличие от этого, платформа .NET Framework ставит своей целью предоставить разработчикам возможность создавать код на любом языке по собственному выбору. При этом платформа обеспечивает максимальную интеграцию всех компонентов, даже если они были написаны на разных языках.

При проектировании платформы .NET Framework были учтены недостатки существующих Windows-платформ. .NET Framework состоит из двух частей: общеязыковой исполняющей среды (common language runtime, CLR) и библиотеки классов (Framework Class Library, FCL). CLR предоставляет модель программирования, используемую во всех типах приложений. У CLR собственный загрузчик файлов, диспетчер памяти (сборщик мусора), система безопасности (безопасность доступа к коду), пул потоков и многое другое. Кроме того, CLR предоставляет объектно-ориентированную модель программирования, определяющую, как выглядят и ведут себя типы и объекты. FCL предоставляет объектно-ориентированный API-интерфейс, используемый всеми моделями приложений. В ней содержатся определения типов, которые позволяют разработчикам выполнять ввод/вывод, планирование задач в других потоках, создавать графические образы, сравнивать строки и т. п.

Естественно, что все эти определения типов соответствуют существующей модели программирования в CLR. Ниже представлен список возможностей и преимуществ платформы NET:

— полное и абсолютное межъязыковое взаимодействие;

— общая среда выполнения для любых приложений .NET, вне зависимости от того, на каких языках они были созданы;

— единая программная модель;

— упрощенная модель программирования;

— упрощенное развертывание;

— удаление же приложений сводится к удалению файлов;

— работа на многих платформах;

— автоматическое управление памятью (сбор мусора);

— интеграция языков программирования;

В платформу .NET входят различные языки программирования: C#, Visual Basic .NET, F#. Совместимыми с ней могут быть также и другие языки. Среди перечисленных языков для практической реализации описанного в предыдущем разделе метода решения задачи был выбран C#.

Язык программирования C# был создан в конце 1990-х годов и стал частью общей .NET-стратегии Microsoft. C# непосредственно связан с C, C++ и Java. Эти три языка — самые популярные и самые распространённые языки программирования в мире. Более того, почти все профессиональные программисты сегодня знают C и C++, и большинство знает Java. Поскольку C# построен на прочном, понятном фундаменте, то переход от этих "фундаментальных" языков к "надстройке" происходит без особых усилий со стороны программистов.

От C язык C# унаследовал синтаксис, многие ключевые слова и операторы. Кроме того, C# построен на улучшенной объектной модели, определенной в C++. C# и Java связаны между собой несколько сложнее. Как упоминалось выше, Java также является потомком C и C++. У него тоже общий с ними синтаксис и сходная объектная модель. Подобно Java C# предназначен для создания

переносимого кода. Однако C# — не потомок Java. Скорее C# и Java можно считать двоюродными братьями, имеющими общих предков.

C# обладает многими полезными особенностями — простота, объектная ориентированность, типовая защищенность, "сборка мусора", поддержка совместимости версий и многое другое. Данные возможности позволяют быстро и легко разрабатывать приложения, особенно COM+ приложения и Web сервисы. При создании C#, его авторы учитывали достижения многих других языков программирования: C++, C, Java, SmallTalk, Delphi, Visual Basic и т.д. Надо заметить, что по причине того, что C# разрабатывался с чистого листа, у его авторов была возможность (которой они явно воспользовались), оставить в прошлом все неудобные и неприятные особенности (существующие, как правило, для обратной совместимости), любого из предшествующих ему языков. В результате получился действительно простой, удобный и современный язык, по мощности не уступающий C++, но существенно повышающий продуктивность разработок.

В C#, как в несомненно современном языке, также существуют характерные особенности для обхода возможных ошибок. Например, помимо упомянутой выше "сборки мусора", там все переменные автоматически инициализируются средой и обладают типовой защищенностью, что позволяет избежать неопределенных ситуаций в случае, если программист забудет инициализировать переменную в объекте или попытается произвести недопустимое преобразование типов. Также в C# были предприняты меры для исключения ошибок при обновлении программного обеспечения.

В C# была унифицирована система типов, теперь вы можете рассматривать каждый тип как объект. Несмотря на то, используете вы класс, структуру, массив или встроенный тип, вы можете обращаться к нему как к объекту. Объекты собраны в пространства имен (namespaces), которые позволяют программно обращаться к чему-либо. Это значит, что вместо списка включаемых файлов заголовков в своей программе вы должны написать какие пространства имен, для доступа к объектам и классам внутри них, вы хотите использовать. В C#

выражение using позволяет вам не писать каждый раз название пространства имен, когда вы используете класс из него. Например, пространство имен System содержит несколько классов, в том числе и Console. И вы можете писать либо название пространства имен перед каждым обращением к классу, либо использовать using как это было показано в примере выше.

При написании программ на C/C++ необходимо принимать во внимание не только тип данных, но его размер в конкретной реализации компилятора. В C# все упрощено — теперь символ Unicode называется просто char (а не wchar_t, как в C++) и 64-битное целое теперь — long (а не __int64). Также в C# нет знаковых и беззнаковых символьных типов.

Серьёзной проблемой при программировании на C++ является механизм ссылок и указателей, традиционно считающийся источником большинства ошибок и утечек памяти. В C# указателей нет. В действительности нетривиальность указателей соответствовала их полезности. Например, порой, трудно себе представить программирование без указателей на функции. В соответствии с этим в C# присутствуют Delegates — как прямой аналог указателя на функцию, но их отличает типовая защищенность, безопасность и полное соответствие концепциям объектно-ориентированного программирования.

Современность C# проявляется и в новых шагах к облегчению процесса отладки программы. Традиционным средством для отладки программ на стадии разработки в C++ является маркировка обширных частей кода директивами #ifdef и т.д. В C#, используя атрибуты, ориентированные на условные слова, вы можете куда быстрее писать отлаживаемый код.

В наше время, когда усиливается связь между миром коммерции и миром разработки программного обеспечения, и корпорации тратят много усилий на планирование бизнеса, ощущается необходимость в соответствии абстрактных бизнес процессов их программным реализациям. К сожалению, большинство языков реально не имеют прямого пути для связи бизнес логики и кода. Например, сегодня многие программисты комментируют свои программы для объяснения того, какие классы реализуют какой-либо абстрактный бизнес

объект. C# позволяет использовать типизированные, расширяемые метаданные, которые могут быть прикреплены к объекту. Архитектурой проекта могут определяться локальные атрибуты, которые будут связанны с любыми элементами языка — классами, интерфейсами и т.д. Разработчик имеет возможность проверить атрибуты какого-либо элемента из программы. Это существенно упрощает работу, к примеру, вместо того чтобы писать автоматизированный инструмент, который будет проверять каждый класс или интерфейс, на то, является ли он действительно частью абстрактного бизнес объекта, можно просто воспользоваться сообщениями основанными на определенных в объекте локальных атрибутах.

C#, являясь последним из широко распространенных языков программирования, должен впитать в себя весь имеющийся опыт и вобрать лучшие стороны существующих языков программирования, при этом являясь специально созданным для работы в .NET. Сама архитектура .NET продиктовала ему (как и многим другим языкам, на которых можно писать под .NET) объектно-ориентированную направленность. Конечно, это не является правилом, возможно создание компиляторов даже функциональных языков по .NET, на эту тему существуют специальные работы.

Из вещей, включенных в спецификацию языка, но не являющихся чисто "программистскими" необходимо отметить возможность использование комментариев в формате XML. Если комментарии отвечают специально описанной структуре, компилятор по ним может сгенерировать единый XML-файл документации.

Ввиду удобного объектно-ориентированного дизайна, C# является хорошим выбором для быстрого конструирования различных компонентов — от высокоуровневой бизнес логики до системных приложений, использующих низкоуровневый код. Также следует отметить, что C# является и Web ориентированным — используя простые встроенные конструкции языка любые компоненты могут быть легко превращены в Web сервисы, к которым можно

будет обращаться из Internet посредством любого языка на любой операционной системе.

Указанные средства разработки будут использованы для реализации функций программных модулей в соответствии со схемой построения системы, рассмотренной в предыдущей главе.

3.2 Средства программного моделирования сетей

Создание моделей сетевых топологий имеет важное значение в процессе проектирования сети, отладки функционирования её устройств, оценки защищённости и выявлении потенциальных рисков, а также полезно при внедрении новых технологий или изменении настроек в качестве тестовой системы.

Сетевые симуляторы позволяют создавать топологии в виртуальной среде и настраивать сетевые устройства так же, как настраиваются их реальные физические аналоги. Тремя наиболее часто используемыми сетевыми смуляторами являются: Cisco Packet Tracer, GNS3 и EVE-NG. Рассмотрим их более подробно.

Одним из старейших, наиболее известных и популярных сетевых симуляторов является Graphical Network Simulator 3 или GNS3. Вопреки своему названию, это не просто симулятор, но и эмулятор. В большинстве случаев он эмулирует аппаратное обеспечение устройств Cisco, таких как, например, маршрутизатор, и запускает реальные файлы образов Cisco IOS на этом виртуальном оборудовании [16]. Для некоторых устройств, таких как коммутаторы, он имитирует функции и функции этих устройств. В случае симуляции используются не настоящие файлы образов IOS, а симулированное устройство, которое было спроектировано и разработано командой GNS3 так, чтобы оно вело себя как сетевое устройство. GNS3 использует Dynamips для эмуляции оборудования Cisco. Эта виртуальная машина используется в основном для версий операционной системы IOS 12.X.

GNS3 может быть установлен на компьютерах с Windows, Mac или Linux и может использовать один из трех вариантов установки. Его можно установить непосредственно на локальную машину, на виртуальную машину, работающую локально, или на виртуальную машину, работающую удаленно. Эта гибкость полезна в зависимости от имеющихся у вас ресурсов. Еще одно преимущество GNS3 заключается в том, что он может соединять виртуальные устройства с реальными устройствами. Таким образом, можно создать топологию, которая существует частично виртуально, а частично физически. Вы даже можете использовать GNS3 как часть вашей производственной сети. GNS3 — это инициатива с открытым исходным кодом, поэтому ее можно использовать бесплатно. Однако образы Cisco должны быть получены по контракту на обслуживание или по программе, проводимой учебным заведением.

EVE-NG (Emulated Virtual Environment Next Generation) — это симулятор сети от нескольких производителей, который предоставляет функции, аналогичные GNS3. Основное отличие от GNS3 заключается в том, что EVE-NG не имеет клиента. По сути, это означает, что он работает как автономная виртуальная машина и для работы не требует установки дополнительных программных компонентов на локальном устройстве. Как и GNS3, EVE-NG для работы требует использования образов Cisco IOS или образов Cisco VIRL. Однако, в отличие от GNS3, у EVE-NG есть бесплатная версия, поддерживаемая сообществом, и профессиональная версия, которую можно приобрести по разумной цене. Дополнительные преимущества профессиональной версии включают лабораторные таймеры, интеграцию с Wireshark и другие ценные инструменты.

Packet Tracer — это проприетарное программное обеспечение Cisco для моделирования сети. В отличие от двух предыдущих вариантов, Packet Tracer — это чистый симулятор. Он запрограммирован так, чтобы имитировать работу коммутаторов, маршрутизаторов, брандмауэров и точек беспроводного доступа, а не запускать их исходные файлы образов IOS. Поскольку это симулятор, он не полностью поддерживает все функции, доступные для физических аналогов

устройств в программном обеспечении. Например, несмотря на поддержку протокола BGP, в Packet Tracer невозможно создавать пиринговые соединения iBGP с внешними устройствами за пределами модели. Программное обеспечение было разработано с учетом функций, необходимых для сертификации на уровне CCNA. Любые расширенные функции помимо этого не обязательно поддерживаются.

Исходя из приведённых выше описаний особенностей различных программных средств можно сделать вывод, что наиболее подходящим для использования в работе является GNS3, так как он предоставляет возможность доступа к эмулируемому оборудованию с реального компьютера, при этом проще в использовании, а используемая в нём виртуальная машина для запуска операционных систем сетевого оборудования менее требовательна к ресурсам компьютера по сравнению с EVE-NG.

3.3 Схема сети

Для экспериментальной оценки выполнения разрабатываемой системой своих функций будем использовать модель сети, состоящую из нескольких подсетей и включающую в себя набор разнообразного оборудования. Схема сети, используемой в работе представлена на рисунке 3.1. На ней представлены следующие устройства:

— маршрутизаторы (R1, R2, R3);

— коммутаторы (Sw1, Sw2, Sw3 и т.д.);

— конечные устройства (PC1, PC2 и т.д.);

— «облако», обозначающее выход в интернет средствами реального компьютера, на котором будет реализована данная модель.

Сеть разделена на 6 подсетей: 192.168.1.0, 192.168.2.0, 192.168.3.0, 192.168.4.0, 172.16.0.0 и 172.17.0.0. Между подсетями располагаются маршрутизаторы, обеспечивающие связь как между соседними, так и между удалёнными сетями, контроль трафика и управление доступом. На

маршрутизаторе R1 для обеспечения выхода всей сети в интернет используется технология трансляции адресов (NAT).

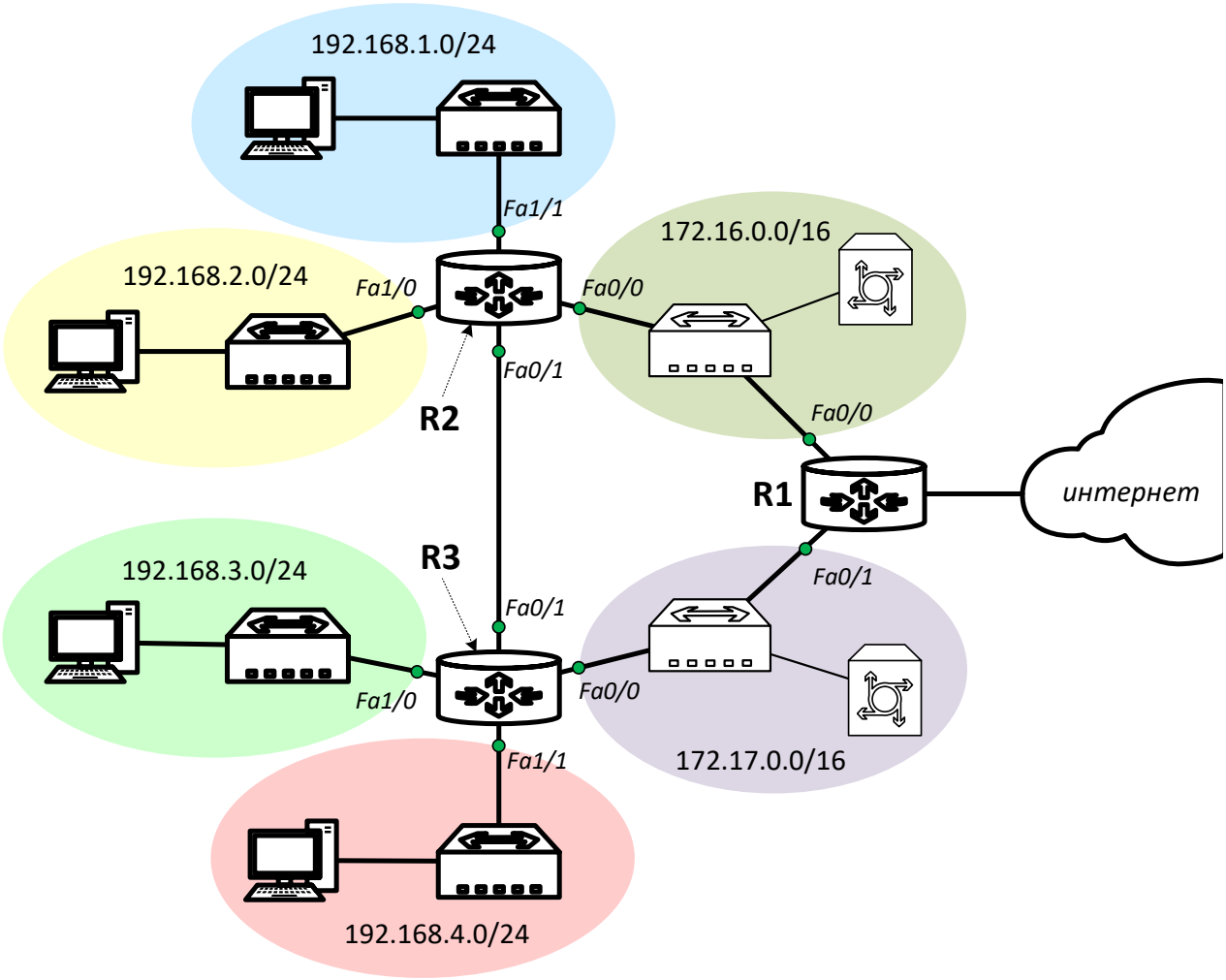


Рисунок 3.1 — Общая схема сети

Представленная схема на практике была реализована с помощью рассмотренного ранее средства программной эмуляции сетей — GNS3. На рисунке 3.2 представлен скриншот из данной программы. В качестве маршрутизаторов выбраны модели Cisco 3725, а для симуляции работы коммутаторов и конечных устройств использованы встроенные в GNS3 виртуальные устройства. Слева на рисунке изображена схема соединения сетевого оборудования, а справа показан список устройств, а также указаны параметры подключения к каждом из них: IP-адрес, порт, а также используемые протокол.

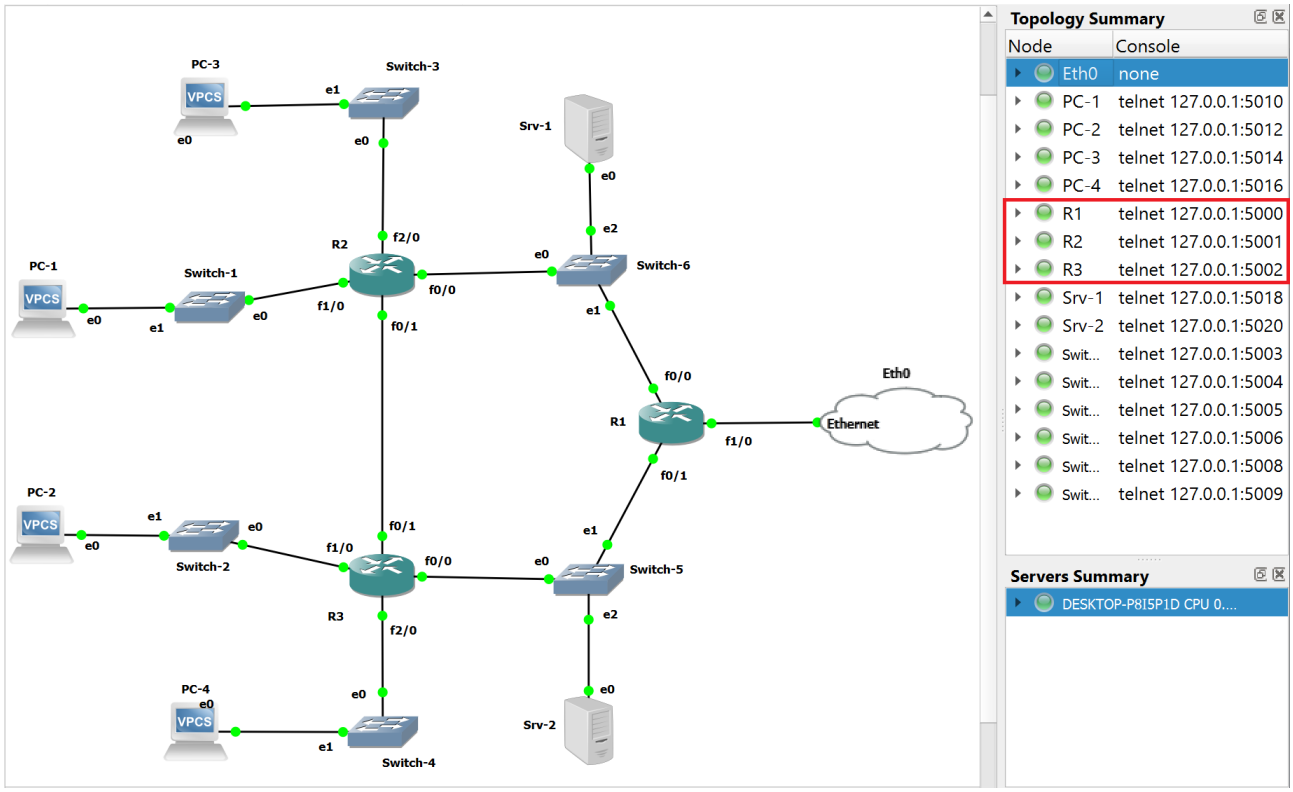


Рисунок 3.2 — Модель сети в GNS3

Так как при моделировании сети в GNS3 все используемые в ней устройства запускаются в виде виртуальных машин на компьютере, то IP-адрес для подключения к ним будет одинаковым и соответствовать локальному адресу компьютера — 127.0.0.1. При этом разные устройства будут использовать разные порты. Для маршрутизаторов R1, R2 и R3 это порты 5000, 5001 и 5002 соответственно. На рисунке также видно, что для подключения необходимо использовать протокол telnet рассмотренный ранее. В реальной сети для подключения к ним можно было бы использовать реальные IP-адреса и одинаковые, стандартные для telnet порты (23).

3.4 Алгоритм поиска пути

Процесс автоматизированной настройки фильтрации трафика и разрешения доступа, реализуемый в работе состоит из 2-х частей:

1) формирование текста команды для создания списка контроля доступа;

2) поиск интерфейсов, входящих в путь передачи данных, и определение их направлений (in/out).

Такое разделение определяется особенностями работы маршрутизаторов со списками контроля доступа. Сразу после создания список существует в памяти маршрутизатора сам по себе и не оказывает никакого влияния на работу устройства. Для того, чтобы правила из списка начали применяться к трафику, необходимо прикрепить список к интерфейсу и направлению. Направления в команде прикрепления списка задаются ключевыми словами in и out. Вход (in) означает что фильтроваться будет только трафик, приходящий на интерфейс извне. Правила списка, прикреплённого к выходу (out) будут применяться к трафику, который прошёл через маршрутизатор и должен быть отправлен дальше через указанный интерфейс.

Так как списки контроля доступа задаются именно для интерфейса, а не для сетевого устройства в целом, то на графе сети необходимо отображать каждый интерфейс как отдельную вершину, независящую от других интерфейсов того же устройства. На рисунке 3.3 показано представление схемы сети, описанной в предыдущем пункте, в виде графа. При этом следует учитывать, что внутри любого устройства трафик может передаваться между любыми интерфейсами, в результате чего сетевое устройство с несколькими интерфейсами должно отображаться на графе как полносвязный фрагмент графа (на рисунке 3.3 такие фрагменты обведены пунктирными линиями).

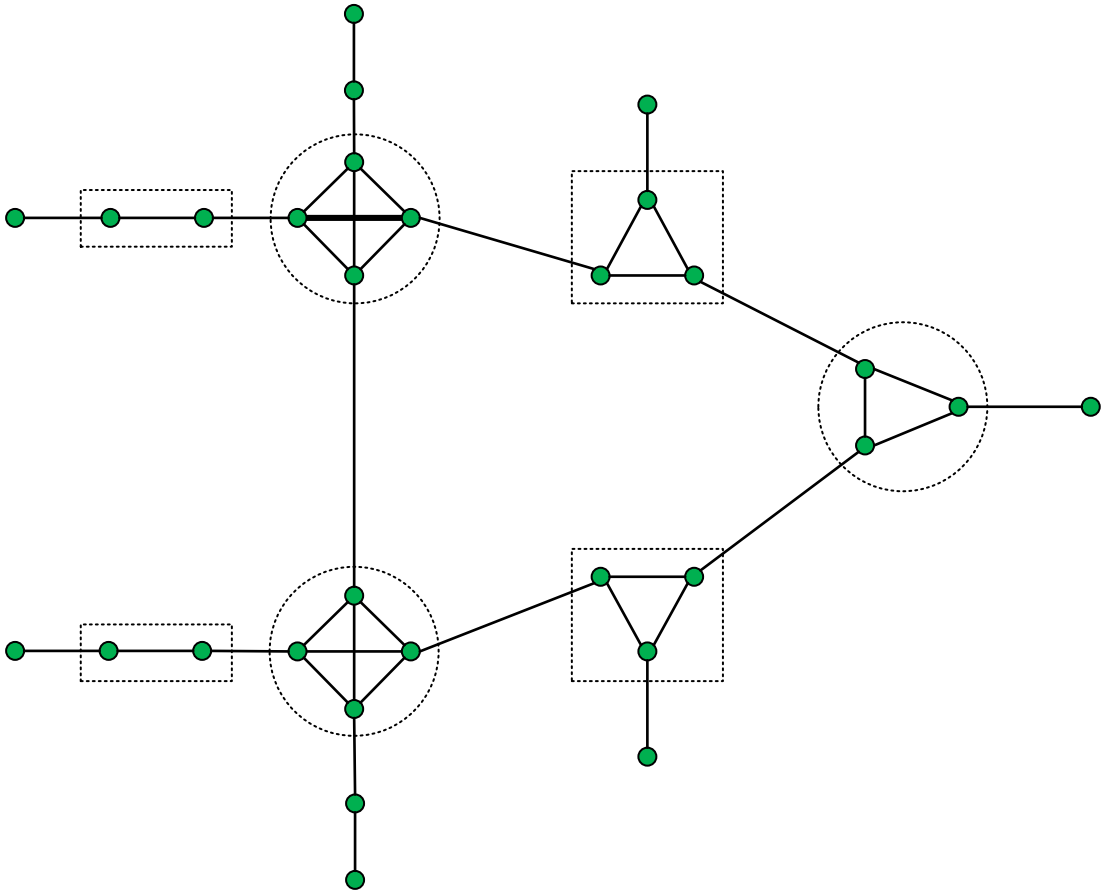


Рисунок 3.3 — Представление схемы сети в виде графа связей между сетевыми интерфейсами

Для определения промежуточных устройств, на которых необходимо настроить списки контроля доступа для разрешения или запрета передачи трафика между выбранными пользователем узлами необходимо найти путь в графе между заданными вершинами. Для этого существует несколько известных алгоритмов, однако наиболее вычислительно эффективным является алгоритм Дейкстры [17].

Данный алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины-источника до конкретной вершинs. При назначении меток и переборе вершин необходимо придерживаться следующих правил:

1) на каждой следующей итерации выбирается вершина с наименьшим известным расстоянием;

2) для выбранной вершины вычисляется расстояние от начальной вершины до всех её соседних;

3) если на каком-либо этапе работы алгоритма было найдено расстояние до вершины меньше минимального известного расстояния, то минимальное известное расстояние заменяется на найденное.

Работу алгоритма Дейкстры можно условной разделить на несколько этапов:

— этап инициализации,

— этап обновления вершин,

— восстановление маршрута.

Рассмотрим их более подробно начиная с этапа инициализации.

Метка начальной вершины полагается равной 0, метки остальных вершин — недостижимо большое число (в идеале — бесконечность, в реальности в большинстве языков программирования существуют константы, хранящие максимально возможное для конкретного типа данных число, например, в C# это Int.MaxValue или Double.MaxValue). Это отражает то, что расстояния от вершины 1 до других вершин пока неизвестны. Все вершины графа помечаются как непосещенные.

На первом шаге алгоритма рассматриваются вершины, соседние с начальной и для них рассчитываются расстояния от начальной на основе веса рёбер графа (для невзвешенного графа веса всех рёбер принимаются равными 1). После того как все соседние вершины проверены, полученное на данном этапе минимальное расстояние до неё считается окончательным и дальнейшему пересчёту не подлежит. Классический алгоритм Дейкстры предполагает вычёркивание такой вершины из списка, однако с использованием средств объектно-ориентированного программирования для неё можно просто указать соответствующее свойство, и в дальнейшем игнорировать.

Описанный шаг повторяется для той вершины из списка оставшихся, текущее расстояние до которой минимально: пересчитываются расстояния до её соседних и, если найден более короткий путь, то обновляются метки, а сама вершина исключается из списка.

Данные действия повторяются пока все вершины не будут отмечены как посещённые, после чего начинается этап восстановления пути. Зная длину кратчайшего пути до каждой вершины начинают их просмотра с конца, то есть с той, которая рассматривалась на последнем шаге алгоритма. Для всех вершин, с которыми она связана находят длину пути, вычитая вес соответствующего ребра из длины пути конечной вершины. Если в результате получается значение, которое совпадает с длиной пути рассматриваемой вершины, то именно из неё был осуществлен переход в конечную вершину. Эта вершина добавляется в путь и далее маршрут восстанавливается уже от неё аналогичным образом.

Учитывая специфику задачи работы, если две вершины графа принадлежат одному устройству, то первая из них в маршруте является входной, а вторая выходной. Соответственно для входной вершины при прикреплении списка контроля доступа нужно указывать in, а для выходной — out.

3.5 Пример работы системы

Для запуска системы имеется исполняемый файл NetContorl.exe. Необходимыми требованиями для её работы является наличие файла базы данных NetControl.mdb, драйвера ADO.NET для подключения к ней и установленный Microsoft .NET Framework 4.5.

За отображение пользовательского интерфейса отвечает библиотека WindowsForms, а сам интерфейс представлен в виде одной формы (рисунок 3.4). На нём имеются несколько групп элементов. Первая из них предназначена для определения маршрута передачи информации от заданного отправителя к получателю. В ней также предусмотрено отображение промежуточных точек, или, правильнее, промежуточных интерфейсов, которые входят в маршрут.

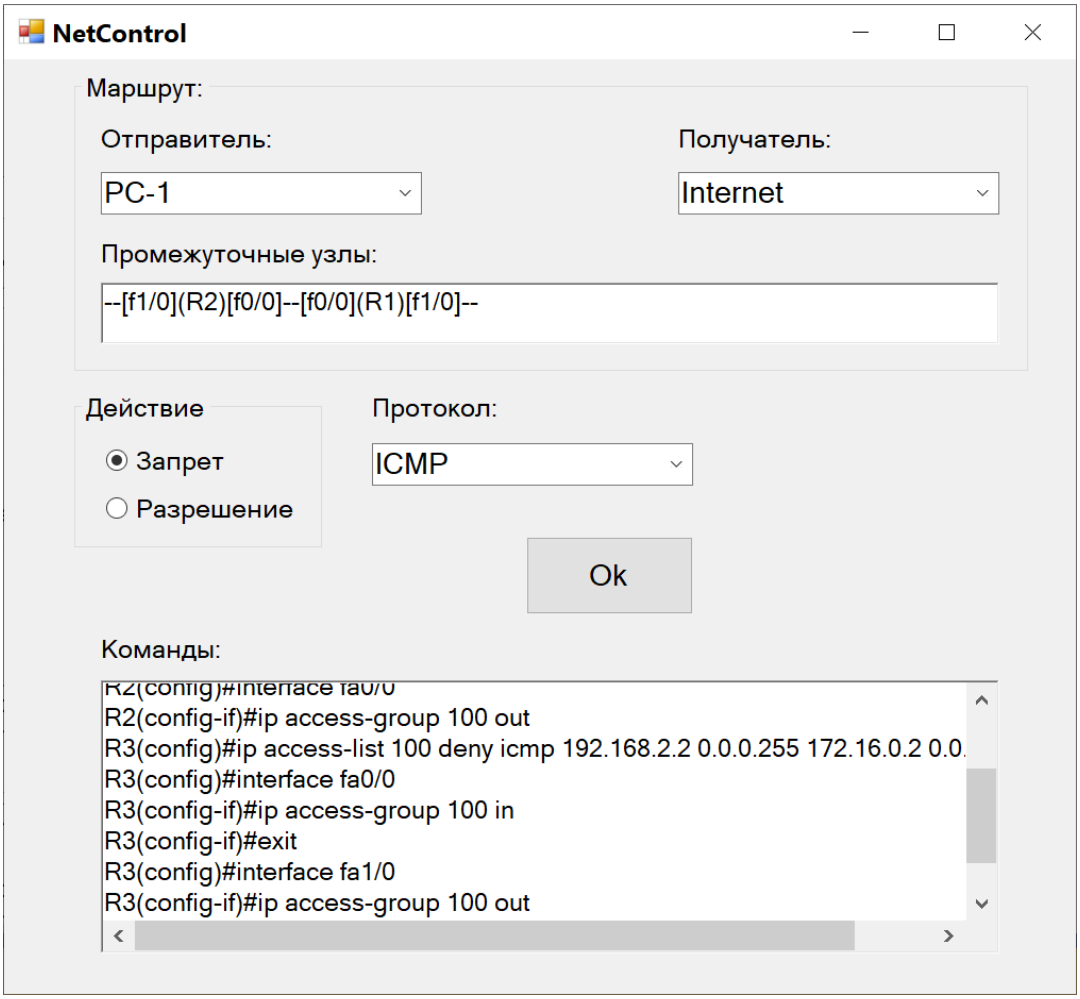


Рисунок 3.4 — Пользовательский интерфейс системы

Следующая группа элементов управления определяет параметры правила, добавляемого в список контроля доступа, а именно действие и протокол. Действие выбирается из элементов radioButton, представляющих собой переключатели с выбором одного варианта, а протокол из выпадающего списка с заранее заданным набором элементов.

В нижней части окна выводится список команд, отвечающих за формирование конфигурацию устройств, входящих в маршрут. Данные команды последовательно отправляются по протоколу telnet на каждый из маршрутизаторов, формируя на нём списки контроля доступа, обеспечивающие выполнение заданного выше правила.

4 Безопасность и экологичность проекта

4.1 Расчёт уровня шума на рабочем месте

Одним из неблагоприятных факторов производственной среды в ИВЦ является высокий уровень шума, создаваемый печатными устройствами, оборудованием для кондиционирования воздуха, вентиляторами систем охлаждения в самих ЭВМ.

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума [выше 80 дБ(А)] на слух человека приводит к его частичной или полной потере.

В таблице 4.1 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 4.1 — Предельные уровни звука, дБ, на рабочих местах

Категория напряженности труда	Категория тяжести труда			
	I. Легкая	II. Средняя	III. Тяжелая	IV. Очень тяжелая
I. Мало напряженный	80	80	75	75
II. Умеренно напряженный	70	70	65	65
III. Напряженный	60	60	-	-
IV. Очень напряженный	50	50	-	-

Уровень шума на рабочем месте программистов и операторов видеоматериалов не должен превышать 50 дБА, а в залах обработки информации на вычислительных машинах – 65 дБА.

Для решения вопросов о необходимости и целесообразности снижения шума необходимо знать уровни шума на рабочем месте оператора.

Согласно ГОСТ 12.1.003-2014 [18] уровень шума, возникающий от нескольких некогерентных источников, работающих одновременно, подсчитывается на основании принципа энергетического суммирования излучений отдельных источников:

$$L_{\Sigma} = 10 \lg \sum_{i=1}^{i=n} 10^{0,1 L_i},$$

где L_i – уровень звукового давления i-го источника шума;

n – количество источников шума.

Полученные результаты расчета сравнивается с допустимым значением уровня шума для данного рабочего места. Если результаты расчета выше допустимого значения уровня шума, то необходимы специальные меры по снижению шума. К ним относятся: облицовка стен и потолка зала звукопоглощающими материалами, снижение шума в источнике, правильная планировка оборудования и рациональная организация рабочего места оператора.

Уровни звукового давления источников шума, действующих на оператора на его рабочем месте представлены в табл. 4.2.

Таблица 4.2 — Уровни звукового давления различных источников

Источник шума	Уровень шума, дБ
Жесткий диск	48
Вентилятор	45
Монитор	16
Клавиатура	14
Принтер	45
Сканер	34

Обычно рабочее место оператора оснащено следующим оборудованием: винчестер в системном блоке, вентиляторы систем охлаждения ПК, монитор, клавиатура, принтер и сканер.

Подставив значения уровня звукового давления для каждого вида оборудования (исходные данные в табл. 13) в формулу, получим:

$$L_{\Sigma} = 10 \cdot \log\left(10^{4,8} + 10^{4,5} + 10^{1,6} + 10^{1,4} + 10^{4,5} + 10^{3,4}\right) = 51,1 \text{ дБ}.$$

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ. И если учесть, что вряд ли такие периферийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже.

Для снижения шума следует:

– ослабить шум самих источников, в частности, предусмотреть применение в их конструкции акустических экранов, кожухов и т.д.;

– снизить эффект суммарного воздействия на рабочие места отраженных звуковых волн за счет звукопоглощения энергии прямых звуковых волн поверхностями ограждающих конструкций;

– применять рациональное расположение оборудования;

– использовать архитектурно-планировочные и технологические решения, направленные на изоляцию источников шума.

4.2 Экологичность работы

Санитарные правила и нормы СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» [19] устанавливают предельно-допустимые уровни (ПДУ) воздействия на людей электромагнитных излучений в диапазоне частот 30 кГц – 300 ГГц (таблица 4.3).

При работе радио и теле и иных передающих станций магнитная составляющая по своей величине не имеет существенного значения, поэтому

интенсивность ЭМИ оценивается только по величине напряженности электрического поля (Е, в/м).

Таблица 4.3 — Предельно допустимые уровни ЭМИ, создаваемые передающими станциями

Частота, МГц	ПДУ, в/м
30-60	5
60-120	4
120-240	3
240-300	2,5

При одновременном облучении от нескольких источников, для которых установлены разные ПДУ, должно соблюдаться следующее условие:

$$\alpha = \sum_{i=1}^{n} \left(\frac{E_i}{\text{ПДУ}_i} \right)^2 \leq 1$$

где E_i — напряженность электрического поля, создаваемого i-источником,

ПДУ_i — предельно-допустимый уровень для i-источника, в/м.

Для защиты населения от ЭМИ мощных передающих станций (свыше 100 кВт) КВ диапазона, они должны размещаться за пределами населенных мест, вдали от жилой застройки.

Вокруг передающих станций создают санитарно-защитные зоны, размеры которых должны обеспечивать предельно-допустимый уровень ЭМИ в населенных местах (таблица 4.4).

Санитарная зона разделяется на зону строгого режима (50-100 м) и зону ограниченного пользования в зависимости от мощности передатчика. В зоне строгого режима допускается пребывание только работников передающей станции, и ограниченное время. В зоне ограниченного пользования можно располагать объекты, в которых граждане могли бы находиться менее 8 час (гаражи, хозяйственно-бытовые помещения и др.)

Таблица 4.4 — Размеры санитарных зон

Суммарная мощность передатчика, кВт	Размеры санитарной зоны, м
до 10	в пределах технической территории
10-75	200-300
75-160	400-500
более 160	500-1000

Определим напряженности электрического поля на разном расстоянии от передающей антенны.

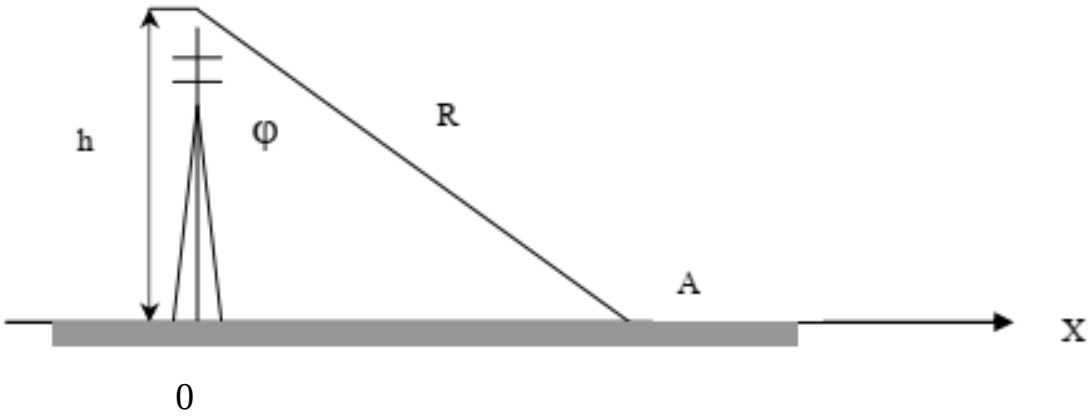


Рисунок 4.1 — Напряжённость электрического поля на расстоянии от антенны

Электрическая напряженность ЭМИ в расчетной точке А определяется по формуле:

$$W = \bar{E} \cdot \bar{H} = \frac{E^2}{377} = \frac{P \cdot \varphi}{4 \cdot \pi \cdot R^2}$$

$$E = \sqrt{\frac{30 \cdot P \cdot \varphi}{h^2 + x^2}}$$

где P — мощность источника, Вт,

φ — коэффициент направленности антенны, рад.

$$\varphi = \text{arctg}\frac{x}{h}$$

где R — расстояние от антенны до расчетной точки, м,

h — высота антенны, м,

x — расстояние от основания антенны до расчетной точки, м.

Электрическая напряженность ЭМИ в жилом помещении определяется по формуле:

$$E_{ж} = k \cdot E$$

где k — ослабление ЭМИ стенами здания.

$k = 1$ — для кирпичных стен;

$k = 0{,}2$ — для панельных стен.

Исходные данные для расчётов приведены в таблице 4.5.

Таблица 4.5 — Исходные данные для расчётов

h, м	1 канал		2 канал		3 канал	
	f_1	P_1	f_2	P_2	f_3	P_3
220	39	1900	69	3900	129	6900

Где h — высота антенны;

f_i — частота, МГц;

p_i — мощность передатчика, Вт.

Определим ПДУ для каждого канала по таблице 4.3 и занесем в таблицу 4.4.

Определим электрическую напряженность в расчетных точках и результаты расчета сведем в таблицу 4.6 и рисунок 4.2.

Для $x = 50$ м:

$$E_1 = \sqrt{\frac{30 \cdot P \cdot \operatorname{arctg}\frac{x}{h}}{h^2 + x^2}} = \sqrt{\frac{30 \cdot 1900 \cdot \operatorname{arctg}\frac{50}{220}}{220^2 + 50^2}} = 0{,}46 \text{ Вт/м},$$

$$E_2 = \sqrt{\frac{30 \cdot 3900 \cdot \operatorname{arctg}\frac{50}{220}}{220^2 + 50^2}} = 0{,}66 \text{ Вт/м},$$

$$E_3 = \sqrt{\frac{30 \cdot 6900 \cdot \operatorname{arctg}\frac{50}{220}}{220^2 + 50^2}} = 0{,}89 \text{ Вт/м},$$

$$\alpha = \left(\frac{0{,}46}{5}\right)^2 + \left(\frac{0{,}66}{4}\right)^2 + \left(\frac{0{,}89}{3}\right)^2 = 0{,}12.$$

Таблица 4.6 — Мощности электромагнитного излучения на разном расстоянии от передающей станции

x	arctg(x/h)	E_1	E_2	E_3	α
0	0	0	0	0	0
50	0,43	0,65	0,92	1,23	0,24
100	0,60	0,69	0,99	1,32	0,28
150	0,74	0,69	0,99	1,31	0,27
200	0,85	0,66	0,95	1,26	0,25
250	0,94	0,62	0,89	1,18	0,22
300	0,43	0,65	0,92	1,23	0,24
ПДУ	–	5	4	3	1

Суммарная мощность передатчиков: 1900 + 3900 + 6900 = 12700 Вт = 12,7 кВт. Отсюда по таблице 4.2 определяем размер санитарной зоны — 200 м.

Находим по таблице 4.4 величину E для $x = 200$ м и рассчитываем напряженность электрического поля в кирпичном и панельном домах.

Таблица 4.5 — Напряженность электрического поля в кирпичном и панельном домах

	E_1	E_2	E_3	α
x = 200	0,85	0,66	0,95	1,26
Кирпичный дом	0,85	0,66	0,95	1,26
Панельный дом	0,138	0,198	0,263	0,011
ПДУ	5	4	3	1

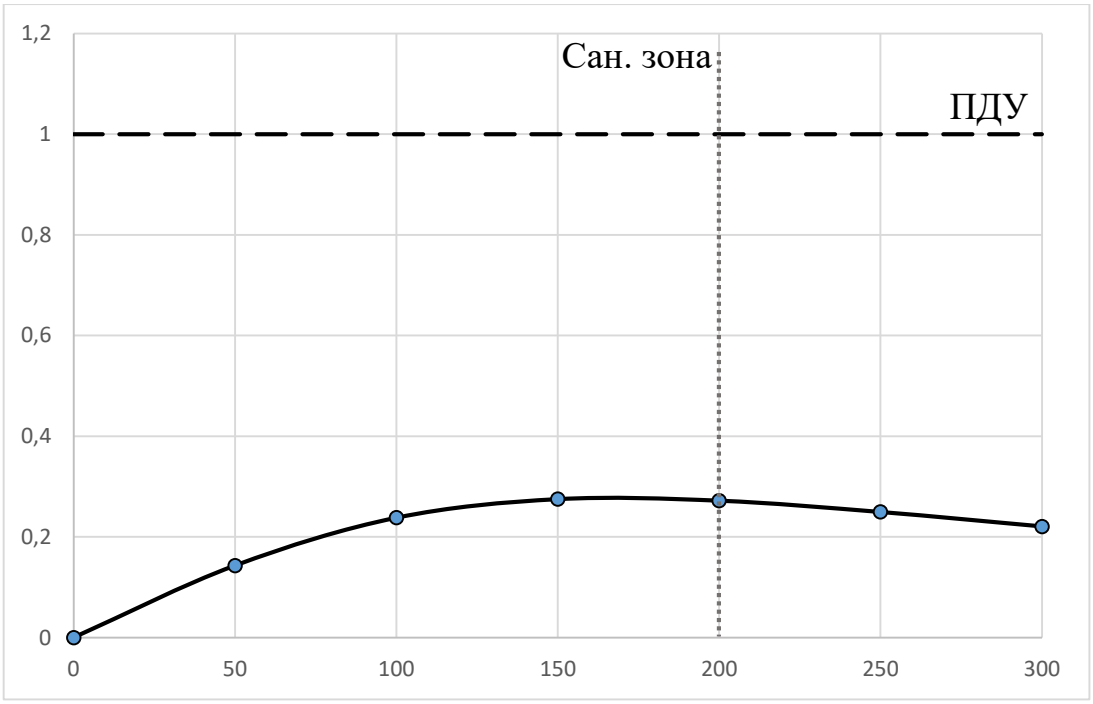


Рисунок 4.2 — Мощности электромагнитного излучения на разном расстоянии от передающей станции

На границе санитарной зоны ЭМИ в кирпичных и панельных домах не превышает допустимые значения.

4.3 Организация и обеспечение пожаробезопасности

Пожарная безопасность обеспечивается противопожарной системой и системой противопожарной защиты. В соответствии с техническим регламентом о требованиях пожарной безопасности [20] все служебные помещения должны иметь «План эвакуации людей при пожаре», в котором регламентируются действия персонала при возникновении пожара и указывается местонахождение пожарной техники.

Пожар в рабочем помещении представляет особую опасность, так как сопряжен с большими материальными потерями. Как известно, пожар может возникнуть при взаимодействии горючих веществ, окисления и источников зажигания. На анализируемом рабочем месте присутствуют все три основные фактора, необходимые для возникновения пожара.

Источниками зажигания могут быть электронные схемы, приборы, применяемые для технического обслуживания, устройства электропитания, кондиционирования воздуха, где в результате различных нарушений образуются перегретые элементы, электрические искры и дуги, способные вызвать загорания горючих материалов.

В современном электронном оборудовании очень высокая плотность размещения элементов электронных схем. В непосредственной близости друг от друга располагаются соединительные провода, кабели. При протекании по ним электрического тока выделяется значительное количество теплоты. При этом возможно оплавление изоляции. Для отвода избыточной теплоты служат системы вентиляции и кондиционирования воздуха. При постоянном действии эти системы представляют собой дополнительную пожарную опасность.

Рассмотрим аналитический прогноз горения зданий для исходных данных, представленных в таблице 4.6.

Таблица 4.6 — Исходные данные для аналитического прогноза горения зданий

№ варианта	Характеристика горящего промышленного объекта				Характеристика объекта или вещества подверженного тепловому воздействию
	Деревянное здание		Резервуар с нефтепродуктами		
	Высота, м	Длина, м	Диаметр, м	Вещество	
9	10	95	20	мазут	возгорание древесины через 5 минут

Расчет протяженности зон теплового воздействия R, м, при горении зданий и промышленных объектов производится по формуле:

$$R = 0{,}282 \cdot R^{*} \cdot \sqrt{\frac{q^{\text{соб}}}{q_{\text{кр}}}}$$

где $q^{\text{соб}}$ — плотность потока собственного излучения пламени пожара, кВт/м^2 (260 кВт/м^2 для древесины, 1 300 кВт/м^2 для мазута);

$q_{\text{кр}}$ — критическая плотность потока излучения пламени пожара, падающего на облучаемую поверхность и приводящую к тем или иным последствиям, кВт/м^2 (для возгорания древесины через 5 минут — 17,5 кВт/м^2);

R* — приведенный размер очага горения, м, равный:

$\sqrt{lh}$ — для горящих зданий,

$(1{,}75 \ldots 2{,}0)\cdot\sqrt{lh}$ —для штабеля пиленого леса,

$0{,}8\cdot D_{\text{рез}}$ — для горения нефтепродуктов в резервуаре;

l, h — длина и высота объекта горения, м;

$D_{\text{рез}}$ — диаметр резервуара, м.

Задавая ту или иную степень поражения человека, сооружения и т.п., по формуле определяют искомое расстояние от очага пожара. Определим расстояние от очага пожара до границы зоны появления ожогов II степени и возгорания горючих материалов.

Рассчитаем протяженность зоны теплового воздействия R, м безопасного нахождения людей при горении деревянного здания и резервуара с керосином.

В случае возгорания в деревянном здании высотой 10 м и длиной 95 м, граница зоны возгорания древесины через 5 минут будет равняться:

$$R = 0{,}282 \cdot R^* \cdot \sqrt{\frac{q^{\text{соб}}}{q_{\text{кр}}}} = 0{,}282 \cdot \sqrt{10 \times 95} \cdot \sqrt{\frac{260}{17{,}5}} = 33{,}5 \text{ м}.$$

Определим расстояние от очага пожара, возникшего в резервуаре с мазутом диаметром 20 м до границы зоны возгорания древесины через 5 минут:

$$R = 0{,}282 \cdot R^* \cdot \sqrt{\frac{q^{\text{соб}}}{q_{\text{кр}}}} = 0{,}282 \times (0{,}8 \times 20) \times \sqrt{\frac{1\,300}{17{,}5}} = 38{,}9 \text{ м}.$$

Рекомендовано оборудовать рассматриваемое помещение установками стационарного автоматического пожаротушения. Наиболее целесообразно применять установки газового тушения пожара, действие которых основано на быстром заполнении помещения огнетушащим газовым веществом с резким сжижением содержания в воздухе кислорода.

В необходимых местах размещены ручные огнетушители (углекислотные ОУ-8 в количестве 2 шт).

Средствами обнаружения и оповещения о пожаре являются автоматические датчики-сигнализаторы о пожаре типа ДТП, реагирующие на

повышение температуры. Средством оповещения сотрудников о пожаре служит внутрифирменное радио.

4.4 Выводы

В соответствии с принятыми нормами в отделе информационных технологий обеспечивается необходимый микроклимат, минимальный уровень шума, созданы удобные и правильные с точки зрения эргономики рабочие места, соблюдены требования технической эстетики.

Для сотрудников отдела в процессе работы одним из важнейших факторов, влияющих на производительность труда при длительной зрительной работе, является достаточное освещение рабочего места. Это достигается правильным выбором и расположением осветительных приборов.

Специальные мероприятия обеспечивают электробезопасность и пожаробезопасность сотрудников.

В целом условия труда инженера соответствуют установленным нормам, сотрудникам обеспечены комфорт и благоприятные условия труда.

5 Технико-экономическое обоснование

В данной главе проводится составление план-графика разработки системы централизованного управления информационно-телеком-муникационной сетью, а также составление сметы затрат на выполнение проекта.

Затраты на разработку включают затраты на оплату труда, материалы, затраты на использование машинного времени, общехозяйственные расходы.

Принимается, что для разработки задействовано два человека: исполнитель (инженер-программист) и руководитель проекта.

Руководитель выполняет постановку задачи, курирует ход работ и дает необходимые консультации при разработке системы.

Исполнитель отвечает за проектирование информационного обеспечения, разработку структур баз данных, реализацию вычислительных алгоритмов завершенного продукта, разработку интерфейсных блоков и отладку программы.

5.1 План-график проектирования и разработки системы

Выбор комплекса работ по разработке проекта производится в соответствии со стандартом «ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств», устанавливающим стадии разработки программных продуктов, и приведен в таблице 5.1.

Таблица 5.1 — Комплекс работ по разработке системы централизованного управления информационно-телекоммуникационной сетью

Содержание работ	Исполнители	Длительность в календарных днях	Загрузка	
			в днях	в %
1 Разработка технического задания				
1.1 Исследование и обоснование разработки				

1.1.1 Постановка задачи	Руководитель	3	3	100
	Программист		3	100
1.1.2 Сбор исходных данных	Руководитель	5		0
	Программист		5	100
1.2 Поиск аналогов и прототипов				
1.2.1 Анализ существующих методов организации управления сетью	Руководитель	5	2	40
	Программист		5	100
1.2.2 Обоснование необходимости разработки и внедрения	Руководитель	5	3	60
	Программист		5	100
1.3 Анализ требований				
1.3.1 Определение и анализ требований к разрабатываемой системе	Руководитель	4	2	50
	Программист		4	100
1.3.2 Анализ методов централизованного управления сетевым оборудованием	Руководитель	4	2	50
	Программист		4	100
1.3.3 Согласование и утверждение технического задания	Руководитель	4	4	100
	Программист		4	100
Итого по этапу 1	Руководитель	30	16	53
	Программист		30	100
2 Проектирование				
2.1 Проектирование архитектуры системы	Руководитель	14	4	29
	Программист		14	100
2.2 Проектирование базы данных топологии сети	Руководитель	14	5	36
	Программист		14	100
2.3 Построение модели сети	Руководитель	14	2	14
	Программист		14	100
Итого по этапу 2	Руководитель	42	6	14
	Программист		42	100
3 Разработка и тестирование модулей системы защиты информации				
3.1 Реализация алгоритма централизованного формирования и отправки команд	Руководитель	14	4	29
	Программист		14	100
3.2 Реализация алгоритма поиска маршрута	Руководитель	14	4	29
	Программист		14	100
3.3 Сборка и тестирование системы	Руководитель	14	4	29
	Программист		14	100
	Руководитель	10	2	20

3.4 Проведение экспериментов, формулирование выводов	Программист		10	100
Итого по этапу 3	Руководитель	52	14	27
	Программист		52	100
4 Оформление рабочей документации				
4.1 Проведение расчетов показателей безопасности жизнедеятельности	Руководитель	8		0
	Программист		8	100
4.2 Проведение экономических расчетов	Руководитель	8		0
	Программист		8	100
4.3 Оформление пояснительной записки	Руководитель	12		0
	Программист		12	100
Итого по 4 этапу	Руководитель	28	0	0
	Программист		28	100
Итого по проекту	Руководитель	152	36	24
	Программист		152	100

На основании вышеописанных таблиц, был составлен план-график по проектированию и разработке системы централизованного управления информационно-телекоммуникационной сетью, показывающий последовательность и взаимосвязь выполнения комплекса работ.

Таблица 5.2 — Календарный график выполнения работ

Содержание работ	Исполнители	Длительность в календарных днях	График работ	
			начало 1 сентября	окончание 31 января
Постановка задачи	Руководитель	3	1 сен	4 сен
	Программист	3	1 сен	4 сен
Сбор исходных данных	Руководитель		4 сен	4 сен
	Программист	5	4 сен	9 сен
Анализ существующих методов организации управления сетью	Руководитель	2	9 сен	11 сен
	Программист	5	9 сен	14 сен
Обоснование необходимости разработки и внедрения	Руководитель	3	14 сен	17 сен
	Программист	5	14 сен	19 сен
Определение и анализ требований к разрабатываемой системе	Руководитель	2	19 сен	21 сен
	Программист	4	19 сен	23 сен

Анализ методов централизованного управления сетевым оборудованием	Руководитель	2	23 сен	25 сен
	Программист	4	23 сен	27 сен
Согласование и утверждение технического задания	Руководитель	4	27 сен	1 окт
	Программист	4	27 сен	1 окт
Проектирование архитектуры системы	Руководитель	4	1 окт	5 окт
	Программист	14	1 окт	15 окт
Проектирование базы данных топологии сети	Руководитель	5	15 окт	20 окт
	Программист	14	15 окт	29 окт
Построение модели сети	Руководитель	2	29 окт	31 окт
	Программист	14	29 окт	12 ноя
Реализация алгоритма централизованного формирования и отправки команд	Руководитель	4	12 ноя	16 ноя
	Программист	14	12 ноя	26 ноя
Реализация алгоритма поиска маршрута	Руководитель	4	26 ноя	30 ноя
	Программист	14	26 ноя	10 дек
Сборка и тестирование системы	Руководитель	4	10 дек	14 дек
	Программист	14	10 дек	24 дек
Проведение экспериментов, формулирование выводов	Руководитель	2	24 дек	26 дек
	Программист	10	24 дек	3 янв
Проведение расчетов показателей безопасности жизнедеятельности	Руководитель		3 янв	3 янв
	Программист	8	3 янв	11 янв
Проведение экономических расчетов	Руководитель		11 янв	11 янв
	Программист	8	11 янв	19 янв
Оформление пояснительной записки	Руководитель		19 янв	19 янв
	Программист	12	19 янв	31 янв

На основе данных, приведённых в таблице 5.2 формируется график этапов работ.

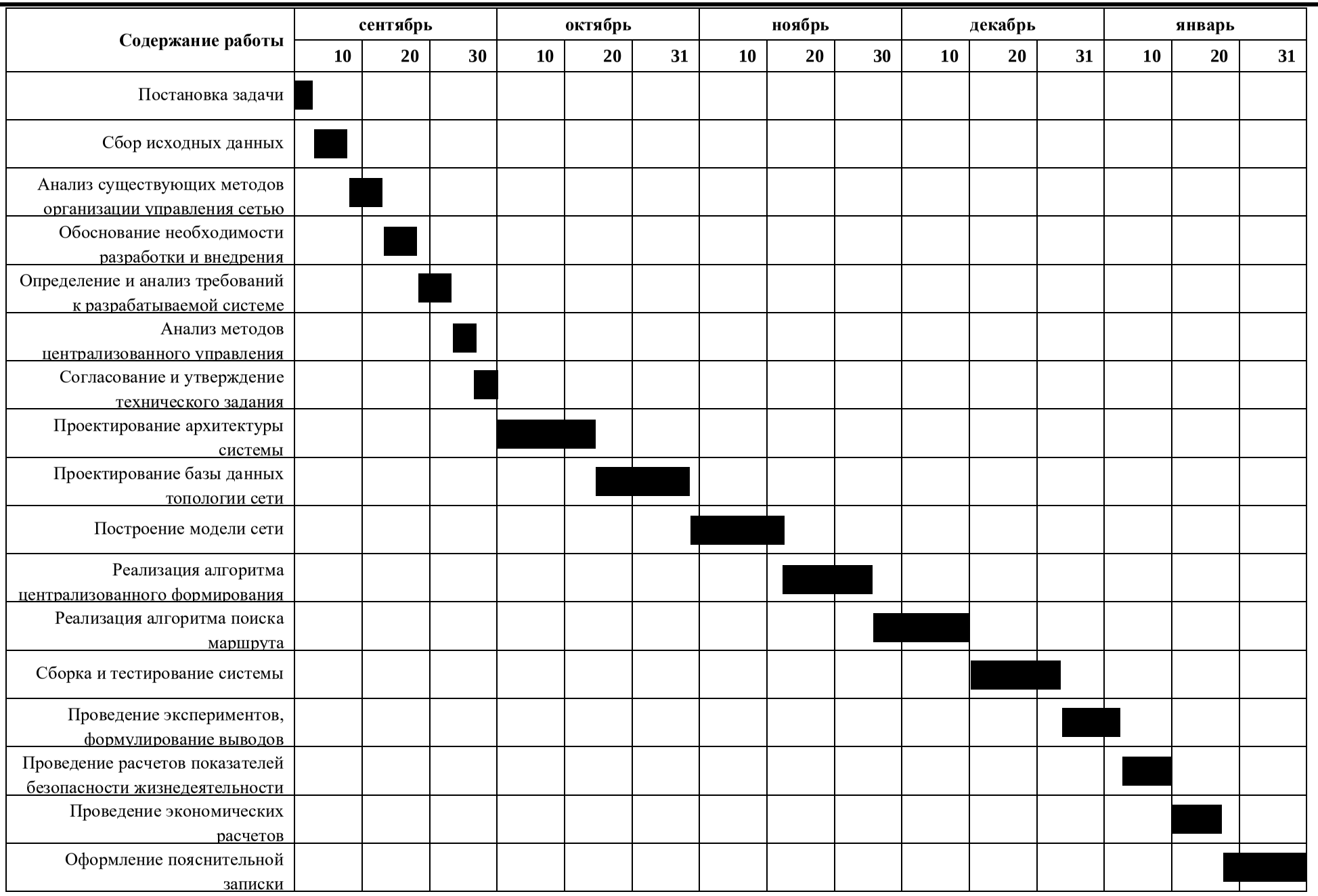


Рисунок 5.1 — План-график разработки проекта

5.2 Расчет затрат на разработку проекта

Предпроизводственные затраты представляют собой единовременные расходы на разработку обеспечивающих или функциональных систем и элементов на всех этапах проектирования, а также затраты на обработку материалов исследования, разработку технического задания, проверки. Сюда включаются затраты на разработку алгоритмов и программ, разработку технического и рабочего проекта системы и её опытной проверки.

Основная заработная плата разработчиков определяется по формуле:

$$E_o = \frac{12 * O * T}{D_p},$$

где O — должностной оклад, руб;

D_p — число рабочих дней в году;

T— затраты времени на разработку, рабочие дни.

Число рабочих дней на разработку определяется по формуле:

$$T = f * T_k,$$

где T_k — календарные дни;

f — коэффициент перевода календарных дней в рабочие;

Данный коэффициент f равен отношению рабочих дней в году к общему числу календарных дней.

$$f = \frac{247}{365} = 0{,}677$$

Согласно данным из таблицы 5.1, рассчитаем трудозатраты руководителя:

$$T_p = 36 * 0{,}677 = 24{,}36 \text{ дн.}$$

Рассчитаем трудозатраты программиста:

$$T_\text{п} = 152 * 0{,}677 = 102{,}86 \text{ дн.}$$

В компании оклад руководителя равен 90 000 руб., а оклад программиста 60 000 руб.

Рассчитаем заработную плату программиста и руководителя по представленной ниже формуле.

$$З_{ор} = \frac{12 * О * Т}{D_р} = \frac{12 * 90\,000 * 24{,}36}{247} = 106\,520{,}55 \text{ руб.}$$

$$З_{оп} = \frac{12 * О * Т}{D_р} = \frac{12 * 60\,000 * 102{,}68}{247} = 299\,835{,}62 \text{ руб.}$$

Отсюда следует, что основная заработная плата на разработку:

$$З_о = 106\,520{,}55 + 299\,835{,}62 = 406\,356{,}16 \text{ руб.}$$

Рассчитаем дополнительную зарплату:

$$З_{доп} = З_о * k_д = 406\,356{,}16 * 0{,}11 = 44\,699{,}18 \text{ руб.},$$

где $k_д$ — коэффициент начисления на дополнительную зарплату.

Премия рассчитывается согласно следующей формуле:

$$З_{пр} = З_0 * k_{пр} = 406\,356{,}16 * 0{,}4 = 162\,542{,}47 \text{ руб.},$$

где $k_{пр}$ — коэффициент начисления на премию.

Рассчитаем начисления на единый страховой сбор:

$$ОСН = (З_0 + З_{доп} + Пр) * k_{осн}$$

$$ОСН = (406\,356{,}16 + 44\,699{,}18 + 162\,542{,}47) * 0{,}3\% = 184\,079{,}34 \text{ руб.}$$

Ввиду того, что проектируемая система должна быть разработана и отлажена с помощью компьютеров, к суммарным затратам на разработку добавляются затраты на их использование.

Доля амортизационных отчислений на компьютерное оборудование, приходящаяся на разработку проекта, определяется по формуле:

$$A_к = \frac{Б_к * T_п}{T_{сл\ ком} * Ф_{д.к} * z}$$

где $Б_к$ — балансовая стоимость компьютерного оборудования;

$T_п$ — продолжительность использования компьютера программистом, час;

$T_{сл.к}$ — срок службы компьютерного оборудования. $T_{сл.к}$ = 6 лет;

z — число одновременно выполняемых проектов (z = 1…3).

$Ф_{д.к}$ — действительный годовой фонд времени компьютерного оборудования;

Действительный годовой фонд компьютерного оборудования рассчитывается по формуле:

$$\Phi_{д.к} = \Phi_н * (1 - y),$$

где $\Phi_н$ — номинальный фонд времени работы компьютерного оборудования;

y — коэффициент потерь времени; y = 3 – 5 %.

$$\Phi_{д.к} = 2\,000 * (1 - 0{,}05) = 1\,900 \text{ час.}$$

Рассчитаем амортизационные отчисления для компьютерного оборудования:

$$A_к = \frac{60\,000 * 1\,216}{6 * 1\,900 * 2} = 3\,200 \text{ руб.}$$

Используемое при конструкторской подготовке системы анализа радиоэфира относится к нематериальным активам предприятия. Амортизационные отчисления на программное обеспечение, приходящиеся на конструкторскую подготовку производства, определяются по формуле:

$$A_{ПО} = \frac{Б_{по} * T_п}{T_{сл.ПО} * \Phi_{д.к}},$$

Где $Б_{по}$ — балансовая стоимость программного обеспечения, руб.;

$T_{сл.ПО}$ — срок службы программного обеспечения (при отсутствии фактических данных применяется равным 10 годам).

$$A_{ПО} = \frac{15\,000 * 1\,216}{10 * 1\,900} = 960 \text{ руб}.$$

Рассчитаем затраты электроэнергии, необходимые на проектирование и разработку системы.

$$Э = N * k * T_п * Ц_э,$$

где N — установленная мощность компьютера, кВт;

k — коэффициент загрузки установленной мощности.

$Ц_э$ — цена электроэнергии, руб/кВт.

Итого затраты на энергию будут следующими:

$$Э = 0{,}6 * 0{,}5 * 1\,216 * 6{,}18 = 2\,254{,}46 \text{ руб.}$$

Итого затраты на машинное время:

$$З_м = 3\,200 + 960 + 2\,254{,}46 = 6\,414{,}46 \text{ руб.}$$

Общехозяйственные расходы:

$$\text{ОХР} = 270\% * \ 406\,356{,}16 = 1\,097\,161{,}63 \text{ руб.}$$

Рассчитаем также затраты на материалы (таблица 5.3).

Таблица 5.3 — Затраты на материалы

Материалы	Единица измерения	Требуемое количество	Цена за единицу руб.	Сумма, руб.
Доступ в интернет	месяц	5	850	4 250
Тонер для лазерного принтера	шт.	1	500	500
Бумага офисная	пачка	1	500	500
ИТОГО				5 250

В таблице 5.4 представлены затраты на разработку и материалы в соответствии со спецификацией, приведённой в п. 5.1.

Таблица 5.4 — Затраты на разработку

Статьи затрат	Затраты
Основная заработная плата	406 356,16
Дополнительная зарплата	44 699,18
Премия	162 542,47
Отчисления на социальные нужды	184 079,34
Затраты на материалы	5 250,00
Затраты на машинное время	6 414,46
Общехозяйственные расходы	1 097 161,63
ИТОГО	1 901 253,24

В результате проведенных расчетов определено, что себестоимость разработки составляет 1 901 253,24 руб. С учетом нормативной рентабельности $R = 20\%$ планируемая прибыль $П$ составит:

$$П = C \cdot R = 1\,901\,253{,}24 \cdot 20\% = 380\,250{,}65 \text{ руб.}$$

А цена продукта составит:

$$Ц = C + П = 1\,901\,253{,}24 + 380\,250{,}65 = 2\,281\,503{,}89 \text{ руб.}$$

Цена продукта с учётом НДС составит:

$$Ц_{НДС} = 1{,}18 \cdot Ц = 1{,}18 \cdot 2\ 281\ 503{,}89 = 2\ 692\ 174{,}59 \text{ руб.}$$

Потенциальными покупателями разработанной системы являются предприятия среднего и крупного бизнеса, использующие в качестве инфраструктуры передачи данных разветвлённые телекоммуникационные сети, требующие периодического контроля и управления.

Заключение

В результате выполнения выпускной квалификационной работы был разработан и реализован проект системы по централизованному управлению сетевым оборудованием телекоммуникационных систем. Такая система позволяет осуществлять конфигурирование устройств с рабочего места администратора, а также автоматически формировать команды и отправлять их через один из протоколов управления удалёнными устройствами.

Система является универсальной, использует стандартные, наиболее распространённые протоколы управления оборудованием, имеет базу данных для хранения топологии сети, поддерживает управление оборудованием различных производителей и может быть применена на широком спектре предприятий и организаций.

Применение методов и средств, предложенных в работе, позволит сократить время на конфигурирование сетевых устройств за счёт упрощения части работы системного администратора, а также снизить вероятность ошибки благодаря автоматическому формированию команд.

Перечень использованных информационных ресурсов

1. Галушка В.В. Сети и системы передачи: учебное пособие. — Ростов н/Д: Издательский центр ДГТУ, 2016. — 105 с.

2. Попов И.И. Компьютерные сети / И.И. Попов, Н.В. Максимов. — М.: Форум, 2004. — 336 с.

3. Айвенс К. Компьютерные сети. Хитрости. — СПб.: Питер, 2006. — 298 с.

4. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации. М. 2008. — 284 с.

5. Олифер В. Компьютерные сети. Принципы, технологии, протоколы: Учебник для ВУЗов / В. Олифер. - СПб.: Питер, 2012. - 944 с.

6. Лимончелли Т., Хоган К., Чейлап С. Системное и сетевое администрирование. Практическое руководство, 2-е издание. — Пер. с англ. — СПб: Символ-Плюс, 2009. — 944 с.

7. Немет Э. Unix и Linux. Руководство системного администратора. 5-е издание / Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли, Д. Макни // — М: Диалектика, 2020. — 1168 с.

8. Лапонина О.Р. Основы сетевой безопасности. Учебное пособие. — 2005. URL: http://master.cmc.msu.ru/files/Laponina-1.pdf.

9. Родичев Ю. А. Информационная безопасность: нормативно-правовые аспекты: Учебное пособие. – СПб.: Питер, 2008. – 272с.: ил.

10. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа / А.Ю. Щеглов. – СПб.: Издательство «Наука и Техника», 2009. – 384 с.

11. Анин Б.А. Защита компьютерной информации. – СПб.: БХВ-Петербург. 2000. – 384 с.

12. Галатенко В.А. Стандарты информационной безопасности: курс лекций: учебное пособие / В.А. Галатенко Под ред. Академика РАН В.Б.

Бетелина / – М.:ИНТУИТ.РУ «Интернет-университет Информационных Технологий», 2006. – 264 с.

13. Как подключиться к Telnet из C#. — URL: https://qna.habr.com/q/661874 (дата обращения: 26.11.2022).

14. НОУ ИНТУИТ. Лекция «Списки контроля доступа». —– URL: https://www.intuit.ru/studies/courses/3646/888/lecture/31159 (дата обраще-ния: 14.14.2022).

15. Galushka V.V. Representation of graphs for storing in relational databases /V. Galushka, V. Fathi, D. Fathi [et al.] // E3S Web of Conferences, 2020. — Vol. 164. — Article number 09014. — 6 p. (Topical Problems of Green Architecture, Civil and Environmental Engineering 2019 (TPACEE 2019), Moscow, Russia, November 19-22, 2019).

16. NetSkills: Online Network School. Курс "Основы GNS3" URL: https://blog.netskills.ru/p/gns3.html (дата обращения: 22.12.2022).

17. Новиков Ф.А. Дискретная математика для программистов: Учебник для вузов. 3-е изд. — СПб.: Питер, 2019. — 384 с. мл.

18. ГОСТ 12.1.003-2014 МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ. Система стандартов безопасности труда. ШУМ. Общие требования безопасности.

19. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания».

20. Технический регламент о требованиях пожарной безопасности: федер. закон: [принят Гос. Думой Федер. собрания РФ 22 июля 2008 г. №123-ФЗ]; О противопожарном режиме [утв. и введен в действие постановления Правительства Российской Федерации от 25.04.2012 г. №390 с изм. на 30.12.2017 г.] – Введ. 2012–05–02; НПБ 105–03. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности. Введ. 2003-08-01.

ПРИЛОЖЕНИЕ А. Листинг программы

```
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.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace Telnet
{

    public partial class Form1 : Form
    {
        public static string GetAnswer(TcpClient tc)
        {
            NetworkStream netStream = tc.GetStream();
            byte[] bytes = new byte[tc.ReceiveBufferSize];
            netStream.Read(bytes, 0, (int)tc.ReceiveBufferSize);
            string returndata = Encoding.ASCII.GetString(bytes);
            return ("Это что хост вернул Вам: " + returndata);
        }

        static void SendCmd(TcpClient tc, string cmd)
        {
            NetworkStream netStream = tc.GetStream();
            if (netStream.CanWrite)
            {
                Byte[] sendBytes = Encoding.ASCII.GetBytes(cmd + "\r");
                netStream.Write(sendBytes, 0, sendBytes.Length);
            }
            Thread.Sleep(1000);
        }
        public Form1()
        {
            InitializeComponent();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
                dataGridView1.Rows.Clear();
                dataGridView1.AutoSizeColumnsMode =
   DataGridViewAutoSizeColumnsMode.AllCells;
                dataGridView1.AutoResizeColumns();
```

```
                dataGridView1.AutoSizeRowsMode =
                    DataGridViewAutoSizeRowsMode.AllCells;
                dataGridView1.AutoResizeRows(
                    DataGridViewAutoSizeRowsMode.AllCellsExceptHeaders);
                dataGridView1.Rows.Add("1");
                dataGridView1.Rows.Clear();
                dataGridView1.AutoSizeColumnsMode =
   DataGridViewAutoSizeColumnsMode.AllCells;
                dataGridView1.AutoResizeColumns();
        }
        private void Button1_Click(object sender, EventArgs e)
        {
            IPAddress address = IPAddress.Parse(ip)
            IPEndPoint host = new IPEndPoint(address, 23);// Параметры для
подключения по telnet
            TcpClient tcpClient = new TcpClient();
            tcpClient.Connect(host);
            Thread.Sleep(100);
            string ip = textBox1.Text;
            string otdel = listBox1.Text;
            if (otd != "")
            {
                string dolzhnost =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                if (ip != "")
                {
                    if (otdel == "Бухгалтерия" && dolzhnost == "Старший
бухгалтер")
                    {
                        progressBar1.Value = 0;
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, "user");
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, "user");
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, "conf t");
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, "ip access-list extended ");//
Добавление нового правила
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, ("permit host "+ip));
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, ("exit"));
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        timer1.Enabled = true;
                    }
                    progressBar1.Value = 0;
                    Console.WriteLine(Form1.GetAnswer(tcpClient));
                    SendCmd(tcpClient, "user");// Логин
                    Console.WriteLine(Form1.GetAnswer(tcpClient));
                    SendCmd(tcpClient, "user");// Пароль
                    Console.WriteLine(Form1.GetAnswer(tcpClient));
```

```
                    SendCmd(tcpClient, "conf t");
                    Console.WriteLine(Form1.GetAnswer(tcpClient));
                    SendCmd(tcpClient, "ip access-list standard ");// Добавление
нового правила
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, ("permit host " + ip));           // в
список для FTP-сервера
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                        SendCmd(tcpClient, ("exit"));
                        Console.WriteLine(Form1.GetAnswer(tcpClient));
                }
                else
                {
                    MessageBox.Show(
        "Пустое поле ввода айпи",
        "Ошибка",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error,
        MessageBoxDefaultButton.Button1,
        MessageBoxOptions.DefaultDesktopOnly);
                }
            }
            else
            {
                MessageBox.Show(
    "Ошибка",
    MessageBoxButtons.OK,
    MessageBoxIcon.Error,
    MessageBoxDefaultButton.Button1,
    MessageBoxOptions.DefaultDesktopOnly);
                }
            }
            private void Button2_Click(object sender, EventArgs e)
            {
                IPAddress address = IPAddress.Parse(ip);
                IPEndPoint host = new IPEndPoint(address, 23);
                TcpClient tcpClient = new TcpClient();
                tcpClient.Connect(host);
                Thread.Sleep(100);
                progressBar1.Value = 0;
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, "user");
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, "user");
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, "conf t");
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, "ip access-list standard");
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("exit"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
```

```
                SendCmd(tcpClient, ("interface fa0/0.6"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("ip access-group " + addr + " out"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("exit"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("permit host ") + ip);
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("exit"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("interface fa0/0.9"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                SendCmd(tcpClient, ("exit"));
                Console.WriteLine(Form1.GetAnswer(tcpClient));
                timer1.Enabled = true;
            }


            private void Timer1_Tick(object sender, EventArgs e)
            {
                progBar1.Increment(+5);
                if (progressBar1.Value == 100)
                {
                    timer1.Enabled = false;
                }
            }

        }
    }
```