Федеральное агентство связи

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

Утверждаю

Зав. Кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра Информационных систем и технологий

Разработка графического приложения для составления недельного рациона питания

Курсовой проект

09.03.01.000056 П.956 КП

Выполнил

студент (ка) гр. Такшеев К.А.

ФИО подпись

Руководитель Кислицин Е.В.

ФИО подпись

г. Екатеринбург. 2020г.

Отзыв руководителя

Федеральное агентство связи

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовой проект

по дисциплине: «Объектно-ориентрованное программирование»

на тему: «Разработка графического приложения для составления недельного рациона питания»

студента(ки) группы ПЕ-71б ФИО Такшеев К.А.

ИСХОДНЫЕ ДАННЫЕ

Разработать программное обеспечение с графическим интерфейсом на предпочтительном языке программирования с использованием объектно-ориентированного подхода:

* с использование специальных средств языка;
* с использованием паттернов программирования;
* с использованием принципов объектно-ориентированного подхода.

Содержание

[Введение 5](#_Toc57281843)

[1 Спецификация 10GBASE-W 6](#_Toc57281844)

[2 Протокол TCP 7](#_Toc57281845)

[3 IP-адресация 11](#_Toc57281846)

[3.1 Адресация по протоколу IPv4 11](#_Toc57281847)

[3.2 Адресация по протоколу IPv6 12](#_Toc57281848)

[3.3 Решение задач 15](#_Toc57281849)

[4 Проектирование ЛВС 20](#_Toc57281850)

[4.1 Задание на проектирование ЛВС 20](#_Toc57281851)

[4.2 Планирование проекта ЛВС 21](#_Toc57281852)

[4.3 Определение эффективного трафика 21](#_Toc57281853)

[4.4 Построение схем учебной компьютерной сети 25](#_Toc57281854)

[4.5 Коммутационный шкаф 26](#_Toc57281855)

[4.6 Расчет полезной пропускной способности сети Ethernet 28](#_Toc57281856)

[5 Экономическая часть 30](#_Toc57281857)

[Заключение 33](#_Toc57281858)

[Список использованных источников 34](#_Toc57281859)

[Приложение А 35](#_Toc57281860)

[Приложение Б 39](#_Toc57281861)

Введение

Компьютерные технологии развиваются все больше и больше с каждым днем, и уметь пользоваться компьютером, в наше время, по большей части определяет степень квалифицированности специалиста. Разработчики программ ставят для себя каждый раз все большиецели. Сначала это было, уменьшение занимаемой памяти программой из-за не сильно мощного компьютерного оборудования, сейчас же, с нынешней компьютерной техникой, задача состоит в том, чтобы как можно сильнее сократить время разработки и исполнения программой своей задачи. А также простота чтения написанного кода, так как, в нынешнее время, очень редко, одну программу пишет один человек. Разработкапрограмм происходит в команде. А для того, чтобы каждый разработчик мог разобраться в написанном ранее коде другим разработчиком, нужно упрощать написание программы, разбивая ее на блоки.

Для достижения этих целей в отрасли создания программных комплексов используют методы и подходы управления процессом разработки.

На разных этапах развития программной инженерии использовались различные технологии программирования:

• императивное программирование;

• модульное программирование;

• структурное программирование;

• программирование, управляемое данными;

• программирование, управляемое событиями;

• функциональное программирование;

• логическое программирование и т.п.

Теперь невозможно принять участие в дискуссии, посвященной программированию, если не использовать термин "объектно-ориентированное программирование".

Целькурсовой работы: закрепить и углубить навыки в использовании объектно-ориентированного подхода к программированию, полученные в процессе изучения дисциплины «Основы и объектно-ориентированного программирования».

Задачи курсовой работы: для выполнения поставленной задачи, предлагается использовать средство объектно-ориентированного языка C#. Предполагается разработать консольное приложение, которое осуществит определенную сортировку, для упорядочения заданных данных по условию задачи, которая будет производить сортировка по одному из заданных ключей, в итоге упорядочив всю полученную информацию.

1 Анализ и описание предметной области

1.1 Предметная область

Правильное питание – залог хорошего самочувствия, здорового внешнего вида и крепкого иммунитета. Сбалансированное здоровое питание позволяет получать все необходимые минералы и микроэлементы. Если же стоит задача снизить вес, соблюдение принципов правильного питания позволит сделать это без дискомфорта и голода.

Суточная калорийность рассчитывается индивидуально, исходя из веса, возраста, пола и активности. Например, для сотрудницы офиса 25-30 лет среднего телосложения, которая ходит в день менее 30 минут, для поддержания веса достаточно будет 1600-1700 ккал, а для её ровесника, который 3 раза в неделю интенсивно тренируется, оптимальным будет дневной рацион калорийностью около 2500 ккал.

Для здорового снижения веса дневную калорийность достаточно снизить на 15-20% - этого вполне достаточно, чтобы постепенно терять лишние килограммы. Часто к правильному питанию относят диеты, которые предполагают снижение дневной калорийности до 1000-1200 калорий, резкое сокращение количества жиров или углеводов. Это большая ошибка! Полуголодный рацион не может быть здоровым.

Резкое снижение дневной нормы калорий заставляет организм работать в авральном режиме, сигнализирует о том, что еда кончилась, и, чтобы выжить, нужно экономить ресурсы. Организм при этом переходит в режим выживания – замедляет обмен веществ, сокращает расход калорий на питание мозга, а также старается запасти побольше жира. После такой диеты велик шанс набрать куда больше веса, чем было сброшено. Кроме того, голодание может нарушить важные процессы в организме и даже оказаться смертельно опасным.

Вода – это основной элемент правильного питания. Без воды организм не способен функционировать. Поэтому очень важно пить столько, сколько требует организм. Для поддержания оптимального уровня водного баланса нужно выпивать как минимум два литра в сутки. Полезнее всего пить чистую воду, а не чай кофе или сок. Делать это лучше отдельно от приема пищи, чтобы не разбавлять желудочный сок.

Если вы почувствовали голод, попробуйте сначала выпить воды. Иногда мозг ошибочно принимает сигналы о жажде за сигналы голода. Если чувство голода при этом прошло – значит, прием пищи можно отложить.

1.2 Словарь здорового питания

Веганы — исключают из своего рациона все продукты животного происхождения (мясо, рыба, птица, молочные продукты и иногда мёд).

Вегетарианцы — едят всё за исключением мяса, птицы, рыбы.

Калорийность (от калории — старой единицы измерения энергии): Энергетическая ценность пищи — количество энергии, получаемой организмом из продукта питания. Теплота сгорания — количество энергии, выделяющейся при полном сгорании вещества.

Белки (протеины, полипептиды) — высокомолекулярные органические вещества, состоящие из альфа-аминокислот, соединённых в цепочку пептидной связью. В живых организмах аминокислотный состав белков определяется генетическим кодом, при синтезе в большинстве случаев используется 20 стандартных аминокислот. Множество их комбинаций создают молекулы белков с большим разнообразием свойств. Кроме того, аминокислотные остатки в составе белка часто подвергаются посттрансляционным модификациям, которые могут возникать и до того, как белок начинает выполнять свою функцию, и во время его «работы» в клетке. Часто в живых организмах несколько молекул разных белков образуют сложные комплексы, например фотосинтетический комплекс.

Жиры, также триглицериды, триацилглицериды (сокр. ТАГ) — органические вещества, продукты этерификации карбоновых кислот и трёхатомного спирта глицерина. В живых организмах выполняют, прежде всего, структурную и энергетическую функции: они являются основным компонентом клеточной мембраны, а в жировых клетках сохраняется энергетический запас организма.

Углеводы являются неотъемлемым компонентом клеток и тканей всех живых организмов представителей растительного и животного мира, составляя (по массе) основную часть органического вещества на Земле. Источником углеводов для всех живых организмов является процесс фотосинтеза, осуществляемый растениями.

Здоровое питание — это питание, обеспечивающее рост, нормальное развитие и жизнедеятельность человека, способствующее укреплению его здоровья и профилактике заболеваний. Соблюдение правил здорового питания в сочетании с регулярными физическими упражнениями сокращает риск хронических заболеваний и расстройств, таких как ожирение, сердечно-сосудистые заболевания, диабет, повышенное давление и рак. Современная наука проводит многочисленные исследования, чтобы оптимизировать рацион питания для профилактики основных хронических заболеваний.

1.3 Анализ существующих программных решений

В современном мире эта ниша очень развита и имеет множество программных и аналитических решений.

На просторах интернета можно найти множество фитнес программ считалок калорий. Как правило, это простые в исполнении программки, считающие калорийность рациона и содержание основных нутриентов (белков, жиров и углеводов), но есть отдельные программы, которые подходят к вопросу питания более детально.

1.3.1 «Калькулятор рациона питания» (АСПОН - Питание)

Пакет программных средств «Калькулятор рациона питания» (АСПОН - Питание) Предназначен для анализа рациона питания на предмет качественной и количественной оценки его сбалансированности и для выработки рекомендуемого сбалансированного рациона питания. С его помощью могут быть составлены сбалансированные индивидуальные и коллективные рационы питания в семье, в воспитательных и образовательных учреждениях, в учреждениях общественного питания и др.

Пакет программных средств (ППС) также может быть использован во врачебной практике, в том числе в комплексе Аспон-Д, с целью профилактического обследования и диагностики. Пользователями могут быть врач-диетолог, практикующий врач, а также другие лица, имеющие отношение к технологии питания, обученные работе с программой и получающие советы врачадиетолога. Субъектом программы является Пациент, чей конкретный рацион анализируется и корректируется с помощью программы. Пациентом может быть отдельный человек или целый коллектив. В тех случаях, когда требуется оценить качество кулинарных изделий или меню школьного, спортивного, ресторанного т.д. или когда для них подбирается оптимальный состав продуктов, с ними надо работать как с пациентами. Программные возможности: анализ представленного существующего рациона пациента на содержание нутриентов; выбор для пациента норм потребления нутриентов (Стандарта).

Сравнение рациона со Стандартом и выявление недостаточности или избыточности потребления требуемых по Стандарту нутриентов; корректировка представленного рациона с выработкой рекомендуемого рациона, отвечающего Стандарту. Программа позволяет формировать индивидуальные рационы, рационы для детских, спортивных и др. коллективов, для учебных и воспитательных учреждений, для учреждений общественного питания, в практике семейного питания и т.п. Важным назначением программы является формирование рационов для различных диет.

Вс определяется тем, какие в программе используются стандарты. Сильными сторонами программы являются: глубокий научный подход к подбору рациона питания, учет дефицита\избытка витаминов, микроэлементов и минералов, поступающих с пищей. Возможность применения программы на различном контингенте пользователей. Большая база данных по продуктам питания. Минусами программы являются: Необходимость работы диетолога с программой. Отсутствие учета двигательной активности и соотношения с потребляемыми калориями. Отсутствие адаптации к использованию спортсменами, нет четкой специализации.

1.3.2 "Калькулятор калорий"

Программа "Калькулятор калорий" Это программа для расчета калорийности еды и расхода потраченных калорий. Программа вычисляет баланс калорийности, сопоставляя потребленные с пищей калории и калории, израсходованные при физической активности. Программа основывается на ведении ежедневного хронологического дневника, куда пользователь заносит измерения объемов тела, вес, съеденные блюда и физические нагрузки. Вычисляется калорийность и пищевая ценность (белки, жиры, углеводы) потребленных продуктов, потраченные при физической активности калории, баланс калорийности за день.

Особенностью программы является то, что данные для просмотра могут быть отобраны за произвольный период; при этом автоматически рассчитываются итоговые и средние показатели за выбранный период - потребленные и потраченные калории в сумме, итоговый баланс калорий за период, баланс белков, жиров и углеводов. Динамика изменений ключевых показателей отображается на диаграммах. Также в программе имеется возможность составления, сохранения и загрузки в дневник рационов питания.

По каждому рациону рассчитывается его пищевая ценность и калорийность. В программе существует возможность добавления своих собственных продуктов или готовых блюд, удаления и редактирования блюд, а также возможность редактирования/добавления/удаления физических нагрузок. Сильными сторонами программы являются: программа имеет возможность производить подсчет энергозатрат по видам деятельности.

Имеет неплохую базу данных по продуктам питания и блюдам Минусами программы являются: база данных продуктов и блюд не учитывает содержания в них микро- и макроэлементов. Отсутствие суплиметов в базе данных. Нет четкой специализации программы. Отсутствие анализа рациона и выдачи рекомендаций. Программы выполнена на программной базе Microsoft Access, что подразумевает наличие у пользователя предустановленного расширенного пакета Microsoft Office. Программа работает только на 32х разрядных системах. Нет продуманного интерфейса и диалога с пользователем.

1.4 Техническое задание

Для разработки технического задания определим, что мы хотим по итогу получить.

Хочется получить компьютерное приложение с графическим интерфейсом.

Подойдем к разработке графического интерфейса.

Требуется что бы он был понятен, так же требуется что бы был простой доступ к каждому дню недели и к каждому приему пищи.

Экранные формы должны быть выполнены с использованием стандартных компонентов, предоставленных средой разработки.

Для самого же приложения требуется ввести ограничения. В первоначальной версии данного программного обеспечения предполагается ввод набора правил, который можно изменить, не нарушая всю целостность кода, то есть не требуется изменения больше одного класса, в данном случает класса правил (Rules).

Требуется составить так же расписание приемов пищи:

1. На один прием пищи
2. на один день
3. на одну неделю

Требуется ввести считывание баз данных существующих продуктов из файла или из удаленной базы данных.

Данные должны иметь собственные модели, от которых должны быть реализованы стандартные реализации.

Для достижения правильной архитектуры должны использоваться паттерны, структуры и принципы, стандартные для приложений использующие принцип объектно-ориентированного подхода.

2 Разработка программного обеспечения

2.1 Планирование объектов

Начиная разработку программного обеспечения с использованием объектно-ориентированного подхода стоит начать с выделения основных объектов (классов).

Основные объекты, которые мы будем использовать можем разделить на 2 вида:

1. Базовые;
2. Инкапсулирующие;
3. Объекты, использующие в себе базовые или инкапсулирующие объекты.

К базовым объектам можно отнести:

1. Структура для хранения продуктов;
2. Продукт;
3. Правила для алгоритма генерации рациона питания.

К инкапсулирующим отнесем:

1. Правила с дополнительными свойствами.

Объекты, использующие в себе базовые или инкапсулирующие объекты:

1. Блюдо;
2. День;
3. Неделя;
4. База данных;
5. Класс для создания расписания.

Класс продуктов содержит в себе такие характерные показатели как:

1. Содержание килокалорий;
2. Содержание жиров;
3. Содержание белков;
4. Содержание углеродов;
5. Масса.

Класс блюдо содержит набор продуктов.

Класс дня содержит набор блюд.

Класс недели содержит набор дней.

Правила с дополнительными свойствами содержит продукт, которые является ограничителем и контрольным продуктов, который определяет оптимальное содержание белков, жиров, углеводов и т.д.

Класс базы данных будет содержать универсальные методы для работы будь то удалённая или локальная (самодельная) база данных.

2.2 Программная реализация спланированных классов

2.2.1 Класс продуктов

Подойдем к реализации нашего программного продукта создав класс продуктов.

Создадим интерфейс IProduct, который будет содержать все выше перечисленные поля. Типы полей определяется характеристикой, которую они определяют. Так поле имя продукта будет типом строки (string), а все остальные поля будут иметь тип числа с плавающей запятой (double). Дополнительно так же расширим данный интерфейс интерфейсом ICloneable, который показывает, что можно создавать копию данного объекта и определим метод Clone с помощью которого и будет создавать клон объекта. Программная реализация интерфейса представлена на рисунке 2.2.1.

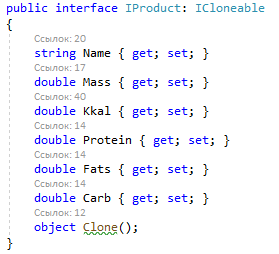


Рисунок 2.2.1 – Интерфейс продуктов

Создадим стандартную реализацию данного интерфейса. Для этого создадим класс DefaultProduct, унаследуем интерфейс IProduct, и реализуем все методы, описанные в данном интерфейсе, применяя к всем полям из интерфейса модификатор доступа publick.

Добавим так же конструктор 2 конструктора для данного класса. Один без параметров, а второй с определением всех полей, которые содержатся в классе. Реализацию данных конструкторов можно увидеть на рисунке 2.2.2.

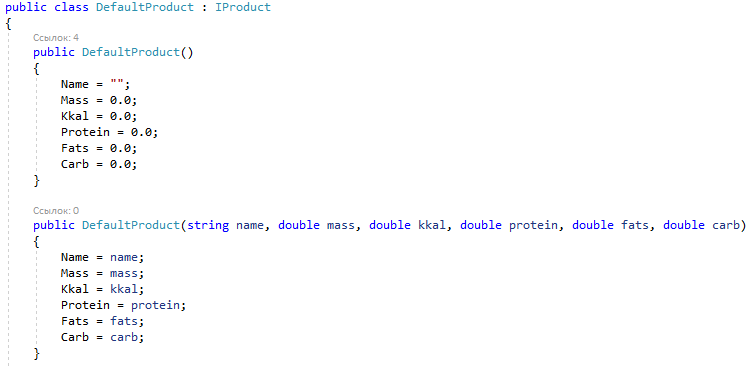


Рисунок 2.2.2 – Реализация конструкторов

2.2.2 Класс правил

Создадим следующий базовый класс – класс правил.

Создадим интерфейс IRules, который содержит одно лишь поле, это поле правил имеющее тип созданного ранее интерфейса IRules. Интерфейс представлен на рисунке 2.2.3.

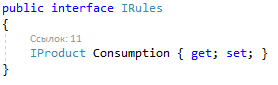


Рисунок 2.2.3 – Реализация интерфейса правил

Создадим стандартную реализацию данного интерфейса. Создадим класс DefaultRules и унаследуем интерфейс IRules. Создадим общедоступный конструктор, в котором определить наши стандартные правила, которые в последствии будет использоваться в создание рациона питания. Реализацию класса можно увидеть на рисунке 2.2.4.

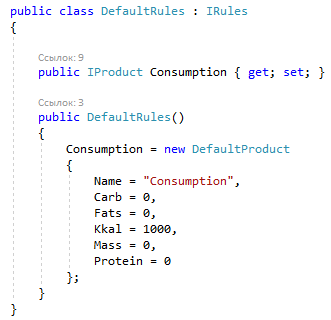


Рисунок 2.2.4 – Реализация стандартного класс для правил

2.2.3 Класс базы данных

Перейдем к реализации базы данных.

Создадим интерфейс IDB с тремя входными параметрами, которые обозначают типы данных которые будем хранить в нашей базе данных, а также разработаем интерфейс так, чтобы можно было легко подстроить под него любую базу данных.

Определим основные методы интерфейса.

Метод поиска данных по ключу (GET), который будет принимать на вход ключ и возвращать массив всех данных которые мы найдем в нашей базе данных.

Метод добавления данных по ключу (SET), который на вход будет принимать ключ и значение ключа, которые будут хранится по этому ключу.

Метод обновления данных по ключу (UPDATE), который будет на вход принимать ключ записи, которую надо изменить и новые данные для данного ключа.

Метод получения всех данных из базы данных (GETALL), который будет возвращать все данных из базы данных.

Реализацию данного интерфейса продемонстрирована на рисунке 2.2.5.

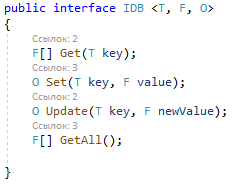


Рисунок 2.2.5 – Реализация интерфейса

Создадим абстрактный класс, в котором определим все входные типы данных для нашей стандартной реализации. Кроме того добавим некоторые поля, которые понадобятся для нашей стандартной реализации, это поле переменной в которой будем хранить нашу базу данных, обозначению методом доступа protected, потому что мы не хотим что бы класса которые наследовались от нашей стандартной реализации напрямую получали доступ к базе данных, таким образом мы абстрагируемся от реализации базы данных в классах наследниках. Так как у нас нет готовой базы данных, мы создадим свою и будем хранить ее в файле, для того что бы поместить ее в нашу программу нам понадобится метод для заполнения нашей базы данных. Определим же метод, который по имени файла, будет заполнять нашу базу данных.

Чтение из файла будем производить с помощью встроенного функционала языка C#. И считываемые строки будем преобразовывать в экземпляры класса продуктов, которые уже и будем хранить в нашей базе данных.

Реализацию данного абстрактного класса можно увидеть на рисунке 2.2.6.

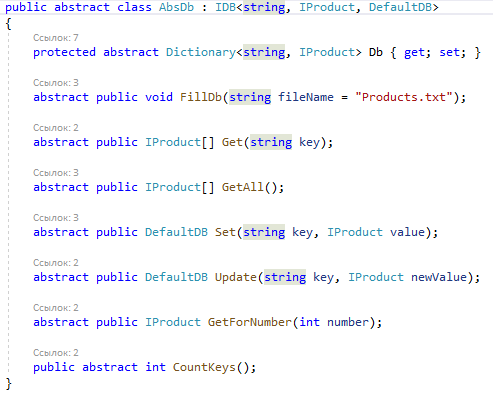


Рисунок 2.2.6 – Реализация абстрактного класс

Создадим стандартную реализацию наше базы данных. Как структуру данных будем использовать Словарь (Dictionary) с ключами в виде строк (string), а значениями данных ключом будет в виде продуктов, но введем абстракцию и укажем интерфейс продуктов IProduct, а все методы по их логическому значению реализуем в унаследованных методах.

2.2.4 Класс блюда

Создадим структуру для хранения блюда. Для этого создадим интерфейс IDish. В котором будут, поля для хранения имени блюда, а также будет поле в которой будем хранить все продукты, которые входят в блюдо, чтобы не привязывать данную структуру к определенной реализации продуктов, укажем как тип хранимых объектов интерфейс IProduct. И расширим нашу реализацию интерфейсом ICloneable и определим соответствующий метод Clone.

Создадим стандартную реализацию определив все поля, метод для генерации имению блюда определим, как строка, состоящая из первых двух букв имени всех продуктов, входящих в блюдо.

2.2.5 Класс дня

Структура для не отличается принципиальными изысками от класса блюд. Поэтому так же создадим интерфейс IDay, в котором определим поля, такие как, поле для хранения блюд, так как у нас интерфейс блюд является генерируем укажем передающийся тип данных в интерфейсе дня и укажем его в интерфейсе объектов нашего хранилища. И расширим наш интерфейс интерфейсом ICloneable, что бы была возможность клонировать объект.

Создадим стандартную реализацию в виде класса DefaultDay. Для это унаследуем класс от созданного ранее нами интерфейса, как параметр передав тип double. И реализовав все метода интерфейса.

2.2.6 Класс недели и использование паттерна Proxy

Как и в прошлые разы начнем создание нашего класса с универсального интерфейса IWeek, который будет принимать типы данных которые принимает интерфейс IDay, потому что одно из полей этого класса будет словарь из всех дней и советующие их дни с блюдами и продуктами. И сделаем возможность клонировать это объём расширив наш интерфейс интерфейсом ICloneable.

Но вот словарь мы будем использовать свой, потому что мы не хотим, чтобы в нашем словаре было больше 7 записей, потому что дней недели всего 7.

Для этого создадим класс MyDictionary и унаследуем его от универсального интерфейса словарей IDictionary.

Прокси используется в том случае, если мы хоти сохранить старое поведение объекта и хотим добавить какое-либо поведение “сверху”, в нашем случае это будет проверка на количество записей в словаре при добавлении записи в словарь, если записей уже 7, как и дней в неделе, мы будем вызывать ошибку переполнения с пояснением ошибки. Так же на всякий случай укажем в комментарии к данному классу, чем отличается наш класс от стандартного.

И реализуем данный интерфейс в стандартном классе DefaultWeek унаследовав от интерфейса IWeek и предав в интерфейс тип данных double.

3 IP-адресация

3.1 Адресация по протоколу IPv4

Межсетевой протокол IP или IPv4 и является базовым протоколом стека TCP/IP. Первоначальное назначение этого протокола – передача пакетов в гетерогенных (составных) сетях со сложной топологией. Он не гарантирует доставку данных до узла назначения, т.е. является датаграммным протоколом [5].

Пакет протокола IP имеет заголовок длиной до 20 байт и поле данных. Структура заголовка IP-пакета приведена на рис. 3.1.1.



Рисунок 3.1.1 — Формат заголовка IPv4

Заголовок IPv4-пакета состоит из следующих полей:

* версия — для IPv4 значение поля должно быть равно 4 (4 бита);
* длина заголовка — длина заголовка IP-пакета в 32-битных словах (4 бита);
* тип сервиса — байт, содержащий набор критериев, определяющих тип обслуживания IP-пакетов (8 бит);
* полная длина — длина пакета в октетах, включая заголовок и данные, минимальное корректное значение для этого поля равно 20, максимальное 65535;
* идентификатор пакета — значение, назначаемое отправителем пакета и предназначенное для определения корректной последовательности фрагментов при сборке пакета (16 бит);
* флаги — первый бит равен нулю (резервный), второй бит определяет возможность фрагментации пакета и третий бит показывает, является ли этот пакет последним в цепочке пакетов (3 бита);
* смещение фрагмента — значение, определяющее позицию фрагмента в потоке данных;
* время жизни — число маршрутизаторов, которые должен пройти этот пакет. При прохождении маршрутизатора это число уменьшатся на единицу. Если значения этого поля равно нулю то, пакет должен быть отброшен;
* протокол — идентификатор протокола следующего уровня указывает, данные какого протокола содержит пакет;
* контрольная сумма заголовка — вычисляется в соответствии с использованием операций сложения 16-разрядных слов заголовка по модулю;
* поле IP-опции не обязательно присутствует в каждой дейтаграмме, размер поля опции зависит от того, какие опции применены.

Традиционная схема деления IP-адреса на номер сети и номер узла основана на понятии класса, который определяется значениями нескольких первых бит адреса. Именно потому, что первый байт адреса 185.23.44.206 попадает в диапазон 128-191, мы можем сказать, что этот адрес относится к классу В, а значит, номером сети являются первые два байта, дополненные двумя нулевыми байтами - 185.23.0.0, а номером узла - 0.0.44.206.

В качестве признака, с помощью которого можно было, бы более гибко устанавливать границу между номером сети и номером узла сейчас получили широкое распространение маски. Маска – это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Поскольку номер сети является цельной частью адреса, единицы в маске также должны представлять непрерывную последовательность. Для стандартных классов сетей маски имеют следующие значения:

Для стандартных классов сетей маски имеют следующие значения:

класс А - 11111111. 00000000. 00000000. 00000000 (255.0.0.0);

класс В - 11111111. 11111111. 00000000. 00000000 (255.255.0.0);

класс С - 11111111. 11111111.11111111. 00000000 (255.255.255.0).

3.2 Адресация по протоколу IPv6

IPv6 представляет собой новую версию протокола Интернет, являющуюся преемницей версии 4. Изменения IPv6 по отношению к IPv4 можно поделить на следующие группы:

Расширение адресации. В IPv6 длина адреса расширена до 128 бит (против 32 в IPv4), что позволяет обеспечить больше уровней иерархии адресации, увеличить число адресуемых узлов.

Спецификация формата заголовков. Некоторые поля заголовка IPv4 отбрасываются или делаются опционными, уменьшая издержки, связанные с обработкой заголовков пакетов с тем, чтобы уменьшить влияние расширения длины адресов в IPv6.

Улучшенная поддержка расширений и опций. Изменение кодирования опций IP-заголовков позволяет облегчить переадресацию пакетов, ослабляет ограничения на длину опций и делает более доступным введение дополнительных опций в будущем.

Возможность пометки потоков данных. Введена возможность помечать пакеты, принадлежащие определенным транспортным потокам, для которых отправитель запросил определенную процедуру обработки, например, нестандартный тип TOS (вид услуг) или обработка данных в реальном масштабе времени.

Идентификация и защита частных обменов. В IPv6 введена спецификация идентификации сетевых объектов или субъектов, для обеспечения целостности данных и при желании защиты частной информации. На рисунке 3.2.1 представлен формат заголовка IPv4.



Рисунок 3.2.1 – Формат заголовка IPv6

Заголовок IPv6-пакета состоит из следующих полей:

* версия – для IPv6 значение поля должно быть равно 6 (4 бита);
* приоритет – код приоритета (8 бит);
* метка потока – используется отправителем для обозначения последовательности пакетов, которые должны быть подвергнуты определенной обработке маршрутизаторами (20 бит);
* размер поля данных – число, указывающее длину поля данных, идущего за заголовком пакета (с учетом расширенного заголовка), (16 бит);
* следующий заголовок – задает тип расширенного заголовка IPv6, который следует за фиксированным (8 бит);
* предельное число шагов – уменьшается на 1 в каждом узле, через который передается пакет; при значении, равном 0, пакет отбрасывается (8 бит);
* адрес отправителя – 128-битный адрес отправителя пакета;
* адрес получателя – 128-битный адрес получателя пакета.

Версия протокола IPv6 внесла существенные изменения в систему адресации. Прежде всего, это коснулось увеличения разрядности адреса: вместо 4 байт IP-адреса в версии IPv4 в новой версии под адрес отведено 16 байт. Это дает возможность пронумеровать огромное количество узлов:

340 282 366 920 938 463 463 374 607 431 762 211 456.

Главной целью изменения системы адресации было не механическое увеличение адресного пространства, а повышения эффективности работы стека TCP/IP в целом.

Вместо прежних двух уровней иерархии адреса (номер сети и номер узла) в IPv6 имеется 4 уровня, из которых три уровня используются для идентификации сетей, а один – для идентификации узлов сети. В новой версии не поддерживаются классы адресов (А, В, С, D, Е), но широко используется технология CIDR. Благодаря этому, а также усовершенствованной системе групповой адресации и введению адресов нового типа IPv6 позволяет снизить затраты на маршрутизацию. Для написания 16-байтовых адресов была выработана новая нотация. Адреса в IPv6 записываются в виде восьми групп по четыре шестнадцатеричных цифры, разделенных двоеточиями, например:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Поскольку многие адреса будут содержать большое количество нулей, были разрешены три метода сокращенной записи адресов. Во-первых, могут быть опущены ведущие нули в каждой группе, например, 0123 можно записывать как 123. Во-вторых, одна или более групп, полностью состоящих из нулей, могут заменяться парой двоеточий, но только один раз. Таким образом, приведенный выше адрес принимает вид:

8000::123:4567:89AB:CDEF

Для сетей, поддерживающих обе версии протокола (IPv4 и IPv6), разрешается задействовать для младших 4 байтов традиционную для IPv4 десятичную запись: 0:0:0:0:0:FFFF:129.144.52.38. В новой версии IPv6 предусмотрено три основных типа адресов: индивидуальные адреса, групповые адреса и адреса произвольной рассылки. Тип адреса определяется значением нескольких старших битов адреса, которые названы префиксом формата. Индивидуальные адреса делятся на несколько подтипов. Основным подтипом индивидуального адреса является глобальный агрегируемый уникальный адрес. Такие адреса могут агрегироваться для упрощения маршрутизации. В отличие от уникальных адресов узлов версии IPv4, которые состоят из двух полей номера сети и номера узла, – глобальные агрегируемые уникальные адреса IPv6 имеют более сложную структуру, включающую шесть полей. Структура полей изображена на рисунке 3.2.2.

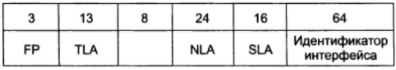


Рисунок 3.2.2 – Структура глобального агрегируемого уникального адреса

Префикс формата (Format Prefix, FP) для этого типа адресов имеет размер 3 бита и значение 001.

Поле TLA (Тор-Level Aggregation, TLA) предназначено для идентификации сетей самых крупных поставщиков услуг. Конкретное значение этого поля представляет собой общую часть адресов, которыми располагает данный поставщик услуг. Сравнительно небольшое количество разрядов, отведенных под это поле (13), выбрано специально для ограничения размера таблиц маршрутизации в магистральных маршрутизаторах самого верхнего уровня Интернета. Это поле позволяет перенумеровать 8196 сетей поставщиков услуг верхнего уровня, а значит, число записей, описывающих маршруты между этими сетями, также будет ограничено значением 8196, что ускорит работу магистральных маршрутизаторов. Следующие 8 разрядов зарезервированы на будущее для расширения при необходимости поля TLA.

Поле NLA (Next-Level Aggregation, NLA) предназначено для нумерации сетей средних и мелких поставщиков услуг. Значительный размер поля NLA позволяет путем агрегирования адресов отразить многоуровневую иерархию поставщиков услуг.

Поле SLA (Site-Level Aggregation, SLA) предназначено для адресации подсетей отдельного абонента, например, подсетей одной корпоративной сети.

Идентификатор интерфейса является аналогом номера узла в IPv4. Отличием версии IPv6 является то, что в общем случае идентификатор интерфейса просто совпадает с его локальным (аппаратным) адресом, а не представляет собой произвольно назначенный администратором номер узла. Идентификатор интерфейса имеет длину 64 бита, что позволяет поместить туда МАС-адрес (48 бит), адрес конечного узла ATM (48 бит) или номер виртуального соединения ATM (до 28 бит), а также, вероятно, даст возможность использовать локальные адреса технологий, которые могут появиться в будущем. Такой подход делает ненужным протокол ARP, поскольку процедура отображения IP-адреса на локальный адрес становится тривиальной – она сводится к простому отбрасыванию старшей части адреса. Кроме того, в большинстве случаев отпадает необходимость ручного конфигурирования конечных узлов, так как младшую часть адреса – идентификатор интерфейса – узел узнает от аппаратуры, а старшую – номер подсети – ему сообщает маршрутизатор.

Протокол IPv6 оставляет в полном распоряжении клиента 2 байта (поле SLA) для нумерации сетей и 8 байт (поле идентификатора интерфейса) для нумерации узлов. Имея такой огромный диапазон номеров подсетей, администратор получает широкие возможности.

Для сравнительно небольшой сети он может выбрать плоскую организацию, назначая каждой имеющейся подсети произвольные неповторяющиеся значения из диапазона в 65 535 адресов, игнорируя оставшиеся. В крупных сетях более эффективным способом (сокращающим размеры таблиц корпоративных маршрутизаторов) может оказаться иерархическая структуризация сети на основе агрегирования адресов. В этом случае используется та же технология CIDR, но уже не поставщиком услуг, а администратором корпоративной сети.

3.3 Решение задач

Задание 1

Определить маску подсети, адрес сети, адрес подсетей, количество хостов в сети, для решения задания данные указаны в таблицах 3.3.1 и 3.3.2.

Таблица 3.3.1 – Данные для задания

|  |  |
| --- | --- |
| Параметр | Предпоследняя цифра номера студенческого билета |
| 1 |
| Класс сети | B |
| Количество подсетей | 48 |

Таблица 3.3.2 – Данные для задания

|  |  |
| --- | --- |
| Параметр | Предпоследняя цифра номера студенческого билета |
| 7 |
| Адреса подсетей | 2,3,4 |

Для 48 подсетей необходимо выделить 6 бит.

Маска: 255.255.0.0

Адрес сети: 140.111.0.0/16

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |  |  |  |  |  | 8 | 4 | 2 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  |  |  |  | 2 | 0 | 0 | 1 | 0 |
| 2: | 0 | 0 | 0 | 0 | 1 | 0 |  |  |  |  |  |  | 3 | 0 | 0 | 1 | 1 |
| 3: | 0 | 0 | 0 | 0 | 1 | 1 |  |  |  |  |  |  | 4 | 0 | 1 | 0 | 0 |
| 4: | 0 | 0 | 0 | 1 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |

Маска подсети: 255.255.252.0

Хостов: 210 – 2 = 1024 – 2 = 1022

Адрес 2 подсети: 140.111.8.0/22

Адрес 3 подсети: 140.111.12.0/22

Адрес 4 подсети: 140.111.16.0/22

Задание 2

Для сети с IP-адресом класса В сформировать N подсетей, указав их адреса в двоичном и десятичном представлениях, и для подсети N перечислить адреса всех узлов в двоичном и десятичном представлениях. Адрес базовой сети подберите самостоятельно. Данные выберите из таблицы 3.3.3.

Таблица 3.3.3 – Данные для задания

|  |  |
| --- | --- |
| Параметр | Последняя цифра номера студенческого билета |
| 7 |
| Nподсетей | 48 |

Для организации 48 подсетей необходимо 6 бит.

Маска: 255.255.0.0

Маска подсети: 255.255.252.0

Адрес сети: 140.111.0.0/16

В таблице 3.3.4 представлены номера подсетей.

Таблица 3.3.4 – Номера подсетей

|  |  |  |
| --- | --- | --- |
| N подсети | IP-адрес в двоичном представлении | IP-адрес в десятичном представлении |
| 1 | 10001100.01101111.**000001**00.00000000 | 140.111.4.0/22 |
| 2 | 10001100.01101111.**000010**00.00000000 | 140.111.8.0/22 |
| 3 | 10001100.01101111.**000011**00.00000000 | 140.111.12.0/22 |
| 4 | 10001100.01101111.**000100**00.00000000 | 140.111.16.0/22 |
| 5 | 10001100.01101111.**000101**00.00000000 | 140.111.20.0/22 |
| … | … | … |
| 46 | 10001100.01101111.**101110**00.00000000 | 140.111.184.0/22 |
| 47 | 10001100.01101111.**101111**00.00000000 | 140.111.188.0/22 |
| 48 | 10001100.01101111.**110000**00.00000000 | 140.111.192.0/22 |

В таблице 3.3.5 представлены номера узлов.

Таблица 3.3.5 – Номера узлов

|  |  |  |
| --- | --- | --- |
| N подсети | IP-адрес в двоичном представлении | IP-адрес в десятичном представлении |
| 1 | 10001100.01101111.**110000**00.00000001 | 140.111.192.1/22 |
| 2 | 10001100.01101111.**110000**00.00000010 | 140.111.192.1/22 |
| 3 | 10001100.01101111.**110000**00.00000011 | 140.111.192.1/22 |
| 4 | 10001100.01101111.**110000**00.00000100 | 140.111.192.1/22 |
| … | … | … |
| 253 | 10001100.01101111.**110000**00.11111101 | 140.111.192.253/22 |
| 254 | 10001100.01101111.**110000**00.11111110 | 140.111.192.254/22 |

Широковещательный адрес: 140.111.192.255/22

Задание 3

Для сети с IP-адресом класса С сформировать N подсетей, указав их адреса в двоичном и десятичном представлениях, и для подсети N перечислить адреса всех узлов в двоичном и десятичном представлениях. Адрес базовой сети подберите самостоятельно. Данные выберите из таблицы 3.3.6.

Таблица 3.3.6 – Данные для задания

|  |  |
| --- | --- |
| Параметр | Последняя цифра номера студенческого билета |
| 7 |
| N подсетей | 11 |

Для организации 11 сетей необходимо 4 бит.

Маска: 255.255.255.0

Маска подсети: 255.255.255.240

Адрес сети: 200.48.23.0/24

В таблице 3.3.7 сформированы номера подсетей.

Таблица 3.3.7 – Номера подсетей

|  |  |  |
| --- | --- | --- |
| N подсети | IP-адрес в двоичном представлении | IP-адрес в десятичном представлении |
| 1 | 11001000.00110000.00010111.**0001**0000 | 200.48.23.16/28 |
| 2 | 10001100.01101111.00001000.**0010**0000 | 200.48.23.32/28 |
| 3 | 10001100.01101111.00001000.**0011**0000 | 200.48.23.48/28 |
| 4 | 10001100.01101111.00001000.**0100**0000 | 200.48.23.64/28 |
| 5 | 10001100.01101111.00001000.**0101**0000 | 200.48.23.80/28 |
| … | … | … |
| 9 | 10001100.01101111.00001000.**1001**0000 | 200.48.23.144/28 |
| 10 | 10001100.01101111.00001000.**1010**0000 | 200.48.23.160/28 |
| 11 | 10001100.01101111.00001000.**1011**0000 | 200.48.23.176/28 |

В таблице 3.3.8 сформированы номера узлов.

Таблица 3.3.8 – Номера узлов

|  |  |  |
| --- | --- | --- |
| N подсети | IP-адрес в двоичном представлении | IP-адрес в десятичном представлении |
| 1 | 10001100.01101111.00001000.**1011**0001 | 200.48.23.177/28 |
| 2 | 10001100.01101111.00001000.**1011**0010 | 200.48.23.178/28 |
| 3 | 10001100.01101111.00001000.**1011**0011 | 200.48.23.179/28 |
| 4 | 10001100.01101111.00001000.**1011**0100 | 200.48.23.180/28 |
| 5 | 10001100.01101111.00001000.**1011**0101 | 200.48.23.181/28 |
| … | … | … |
| 12 | 10001100.01101111.00001000.**1011**1100 | 200.48.23.188/28 |
| 13 | 10001100.01101111.00001000.**1011**1101 | 200.48.23.189/28 |
| 14 | 10001100.01101111.00001000.**1011**1110 | 200.48.23.190/28 |

Задание 4

Даны адреса сетей определить: номер подсети и номер хоста в подсети.

Адрес: 12.63.41.93\19

Сеть класса A

12.63.41.93/19

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 23 | 22 | 21 | 212 | 211 | 210 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| Номер подсети | | | Номер узла подсети | | | | | | | | | | | | |

Номер подсети равен: 1

Номер узла подсети равен: 2048 + 256 + 64 + 16 + 8 + 4 + 1 = 2397

Задание 5

Дан адрес класса C с адресом 200.45.222.0/24

Необходимо разбить сеть на 6 подсетей. С количеством узлов в подсетях:

1) 1,2,3,4 подсетях необходимо создать по 10 узлов в каждой подсети;

2) в 5 подсети количество узлов должно быть равно 36:

3) в 6 подсети количество узлов должно быть равно 62.

На рисунке 3.3.1 изображена схема распределения подсетей.

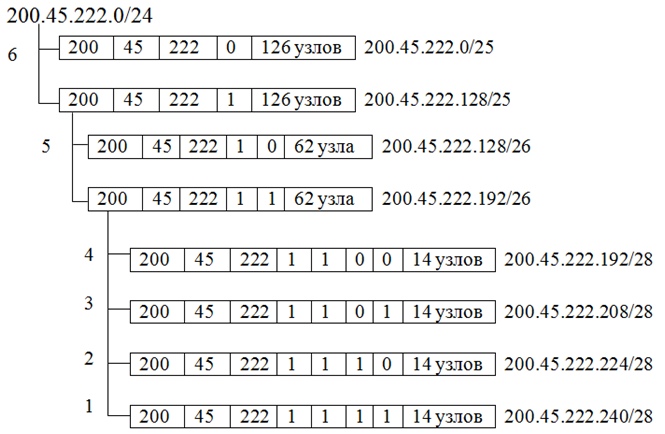


Рисунок 3.3.1 – Схема распределения подсетей

4 Проектирование ЛВС

4.1 Задание на проектирование ЛВС

Зарисовать проект сети и расчет характеристик, определить аппаратные и программные средства для комплектации вычислительной сети, размещение узлов сети и каналов сетевой связи, доступ к ресурсам глобальной сети (web- ftp- mail-серверам), расчет технических характеристик корпоративной сети. Зарисовать структурированную кабельную систему (СКС) и размещение оборудования. Зарисовать телекоммуникационный шкаф и размещение телекоммуникационного оборудования в шкафу. Выбрать коммутационное оборудование, используемое для в структуре сети и рассмотреть технические характеристики. В качестве прототипа зданий организации используются корпуса института. Необходимые расстояния выбираются условно и устанавливаются студентом самостоятельно. Основные исходные данные приведены в таблице 4.1.1.

Таблица 4.1.1 – Исходные данные

|  |  |
| --- | --- |
| Предметная область | Учебная компьютерная сеть |
| Количество серверов | 4 (назначение серверов определяете самостоятельно) плюс сервер печати |
| Условия подключения к провайдеру Интернет | Расстояние до провайдера 12 км. Использовать кабель ВОЛС |
| Кол-во рабочих станций | 120 |
| Модемы | 1 |
| Маршрутизаторы | 2 |
| Коммутаторы | 3 |

Целью проектирования является расчет технических характеристик локальной сети, корпоративной или технологической сети соответственно, определение аппаратных и программных средств комплектации сетей предприятия, размещение узлов сети и каналов сетевой связи, расчет полезной пропускной способности сети, расчет экономических характеристик корпоративной, локальной или технологической сети.

Для проектирования сети необходимо определить:

* перечень необходимых задач и служб, выполняющихся в сети;
* расположение компьютеров – рабочих станций и серверов;
* план помещений, в которых необходимо построить компьютерную сеть;
* дополнительные технические, экономические и эксплуатационные требования.

При проектировании необходимо выполнить следующие задачи:

1. выбор сетевой технологии (технологий);
2. выбор топологии сетевых соединений;
3. разводки и размещения рабочих станций и серверов (структурированная кабельная система);
4. выбор и определение перечня используемого сетевого программного обеспечения и протоколов;
5. расчет и планирование среднего трафика и коэффициента использования сети;
6. расчет полезной пропускной способности сети;
7. расчет стоимости внедрения локальной сети и экономического эффекта ее эксплуатации.

4.2 Планирование проекта ЛВС

Планирование проекта ЛВС начинается с предварительного выбора базовой сетевой технологии для проектируемой локальной сети на основании технических требований, экспертных данных и теоретического материала. Данная учебная компьютерная сеть расположена на небольшой территории, внутри здания так, что кабеля большой длины не требуется. А так как у нас используется кабель UTP, и ЛВС основана на протоколе Fast Ethernet, топология сети будет смешанной [2].

4.3 Определение эффективного трафика

Определяется эффективный [трафик](http://v.ladimir.kiev.ua/kmis/kmis.htm#traffic) на основе формулы (4.1).

, (4.1)

где Пэi – эффективный трафик;

tср.i – среднего времени занятия задачей сети;

tраб – общее время работы сети;

Пн – номинальная пропускная способность сети.

Расчёты производятся на основе таблицы 4.3.1.

Таблица 4.3.1 – Сетевые задачи, используемые в современных локальных сетях

| Задача | Среднее время занятия задачей сети, мин. в сут. | Серверная часть | Клиентская  часть |
| --- | --- | --- | --- |
| обмен файлами | 10-60 на 1 станцию | [Сетевая ОС](http://v.ladimir.kiev.ua/kmis/kmis.htm#sos) | Сетевая ОС |
| [файловый сервер](http://v.ladimir.kiev.ua/kmis/kmis.htm#file_server) | 120-360 | Серверная сетевая ОС | Клиентская сетевая ОС |
| резервирование информации | 5-30 на 1 раб. станцию  10-120 на 1 сервер | Сетевая ОС | Сетевая ОС |
| [сетевая печать](http://v.ladimir.kiev.ua/kmis/kmis.htm#print_server) | 1-20 на 1 станцию | Сетевая ОС | Сетевая ОС |
| [служба терминалов](http://v.ladimir.kiev.ua/kmis/kmis.htm#terminal) | 10-300 на 1 станцию  (трафик 14-100 кбит/с) | Серверная сетевая ОС | Клиентская сетевая ОС |
| [СУБД](http://v.ladimir.kiev.ua/kmis/kmis.htm#db_server) | 5-30 на 1 станцию | Сервер БД | Приложения БД |
| [удаленный доступ](http://v.ladimir.kiev.ua/kmis/kmis.htm#ras) | 60-480 на 1 пару модемов | Сервер удал. доступа | Клиент удал. доступа |
| [Интернет](http://v.ladimir.kiev.ua/kmis/kmis.htm#web) | 10-120 на 1 клиента | Прокси-сервер | Браузер |
| [электронная почта](http://v.ladimir.kiev.ua/kmis/kmis.htm#mail) | 0,5-2 на 1 клиента | Почтовый сервер | Почтовый клиент |
| [Интернет](http://v.ladimir.kiev.ua/kmis/kmis.htm#intranet) | 5-20 на 1 клиента | Веб-сервер | Браузер |
| интерактивные сообщения | 1-5 на 1 станцию | различные | различные |
| голосовая связь  (IP-телефония) | 10-60 на 1 станцию  (трафик 33-64 кбит/с) | различные | различные |
| видеоконференции | 20-40 на 1 станцию  (трафик 0,1-1 Мбит/с) | различные | различные |
| службы сетевой безопасности | 15-20 на 1 сервер  + 2-5 на 1 клиента | Серверная сетевая ОС | Клиентская сетевая ОС |

Общее время работы сети будет равняться суткам, так как сервера работают практически всегда круглосуточно. Номинальная пропускная способность составляет 100Мбит/с, так как используется технология Fast Ethernet. Сеть занята задачами не полностью, потому при расчётах номинальная пропускная способность не учитывается. Используя данные из таблицы 4.3.1, проведём расчёты эффективности сети.

Данные, полученные при расчётах структурированы в таблице 4.3.2.

Таблица 4.3.2 – Результаты расчётов эффективного трафика

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Среднее |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | время |  |  |  |  |  |  |  |  |  |  |
|  | Задача |  |  | занятия |  |  | Серверная |  |  | Клиентская |  |  | Пэ i | |
|  |  |  | задачей |  |  | часть |  |  | часть |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | сети, мин. в |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | сут. |  |  |  |  |  |  |  |  |  |  |
| обмен файлами | | |  | 10-60 на 1 | |  | [Сетевая ОС](http://v.ladimir.kiev.ua/kmis/kmis.htm#sos) | |  | Сетевая ОС | |  | |  |
|  |  |  |  | 60 |
|  |  |  |  | станцию | |  |  |  |  |  |  | 480 | |
|  |  |  |  |  |  |  |  |  |  | = 0,125 | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |  |
| резервирование | | |  | 5-30 на 1 | |  | Сетевая ОС | |  | Сетевая ОС | |  | |  |
|  |  |  |  | 190 |
| информации | | |  | раб. | |  |  |  |  |  |  | 480 | |
|  |  |  |  |  |  |  | = 0,393 | | |
|  |  |  |  | станцию | |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 10-120 на 1 | |  |  |  |  |  |  |  |  |  |
|  |  |  |  | сервер | |  |  |  |  |  |  |  |  |  |
| [сетевая печать](http://v.ladimir.kiev.ua/kmis/kmis.htm#print_server) | | |  | 1-20 на 1 | |  | Сетевая ОС | |  | Сетевая ОС | |  | |  |
|  |  |  |  | 20 |
|  |  |  |  | станцию | |  |  |  |  |  |  | 480 | |
|  |  |  |  |  |  |  |  |  |  | = 0,042 | | |
| [служба терминалов](http://v.ladimir.kiev.ua/kmis/kmis.htm#terminal) | | |  | 10-300 на 1 | |  | Серверная | |  | Клиентская | |  | | ∗ |
|  |  |  |  | 300 |
|  |  |  |  | станцию | |  | сетевая ОС | |  | сетевая ОС | | 480 | |
|  |  |  |  |  |  | = 0,625 | | |
|  |  |  |  | (трафик 14- | |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 100 кбит/с) | |  |  |  |  |  |  |  |  |  |
| [СУБД](http://v.ladimir.kiev.ua/kmis/kmis.htm#db_server) | | |  | 5-30 на 1 | |  | Сервер БД | |  | Приложения | |  | |  |
|  |  |  |  | 30 |
|  |  |  |  | станцию | |  |  |  |  | БД | | 480 | |
|  |  |  |  |  |  |  |  | = 0,063 | | |
| [Интернет](http://v.ladimir.kiev.ua/kmis/kmis.htm#web) | | |  | 10-120 на 1 | |  | Прокси- | |  | Браузер | |  | |  |
|  |  |  |  | 120 |
|  |  |  |  | клиента | |  | сервер | |  |  |  | 480 | |
|  |  |  |  |  |  |  |  | = 0,25 | | |
| [электронная почта](http://v.ladimir.kiev.ua/kmis/kmis.htm#mail) | | |  | 0,5-2 на 1 | |  | Почтовый | |  | Почтовый | |  | |  |
|  |  |  |  | 2 |
|  |  |  |  | клиента | |  | сервер | |  | клиент | | 480 | |
|  |  |  |  |  |  | = 0,004 | | |
| [Интранет](http://v.ladimir.kiev.ua/kmis/kmis.htm#intranet) | | |  | 5-20 на 1 | |  | Веб-сервер | |  | Браузер | |  | |  |
|  |  |  |  | 20 |
|  |  |  |  | клиента | |  |  |  |  |  |  | 480 | |
|  |  |  |  |  |  |  |  |  |  | = 0,042 | | |
| интерактивные | | |  | 1-5 на 1 | |  | различные | |  | различные | |  | |  |
|  |  |  |  | 5 |
| сообщения | | |  | станцию | |  |  |  |  |  |  | 480 | |
|  |  |  |  |  |  |  | = 0.01 | | |
| службы сетевой | | |  | 15-20 на 1 | |  | Серверная | |  | Клиентская | |  | |  |
|  |  |  |  | 25 |
| безопасности | | |  | Сервер | |  | сетевая ОС | |  | сетевая ОС | | 480 | |
|  |  |  | = 0,052 | | |
|  |  |  |  | +2-5на1 | |  |  |  |  |  |  |  |  |  |
|  |  |  |  | клиента | |  |  |  |  |  |  |  |  |  |
| Итого: | | |  | ПΣ=3,942 | |  |  |  |  |  |  |  |  |  |

Значение ПΣ з. умножается на коэффициент служебного, широковещательного и неучтенного трафика kс.т. = (0,05¸0,07)\*n, где n – количество компьютеров в сети, и коэффициент запаса kз = (1,2¸2,0) для учета будущего развития сети(4.2).

где kс.т – коэффициент служебного трафика;

n – количество компьютеров в сети.

Расчёт общего сетевого трафика:

ПΣ 37,843 Мбит/с. (4.2)

По полученному значению ПΣ уточняется выбранная технология ЛВС таким образом, чтобы [коэффициент использования сети](http://v.ladimir.kiev.ua/kmis/kmis.htm#kzs) kисп. был не более (0,3÷0,6). [Коэффициент использования сети](http://v.ladimir.kiev.ua/kmis/kmis.htm#kzs) рассчитывается по формуле (4.3).

(4.3)

где kисп – [коэффициент использования сети](http://v.ladimir.kiev.ua/kmis/kmis.htm#kzs)

ПΣ – общий сетевой трафик;

Пном – номинальная пропускная способность сети.

[Коэффициент использования сети](http://v.ladimir.kiev.ua/kmis/kmis.htm#kzs):

= 0,378

[Коэффициент использования сети](http://v.ladimir.kiev.ua/kmis/kmis.htm#kzs) в пределах нормы. Топология сети выбрана верна.

4.4 Построение схем учебной компьютерной сети

Согласно выбранной нами технологии строим схему локально-вычислительной сети и схему структурированной кабельной системы. Схема ЛВС сети представлена на рисунке 4.4.1, а структурированной кабельной системы представлена на рисунке 4.4.2. Все связующие провода необходимо проложить в кабель-канал под фальшпотолком.

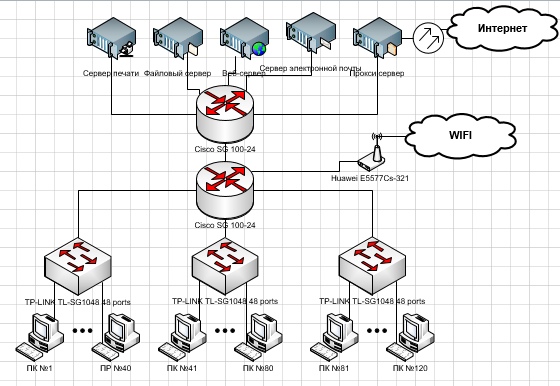


Рисунок 4.4.1 – Схема локально-вычислительная сети

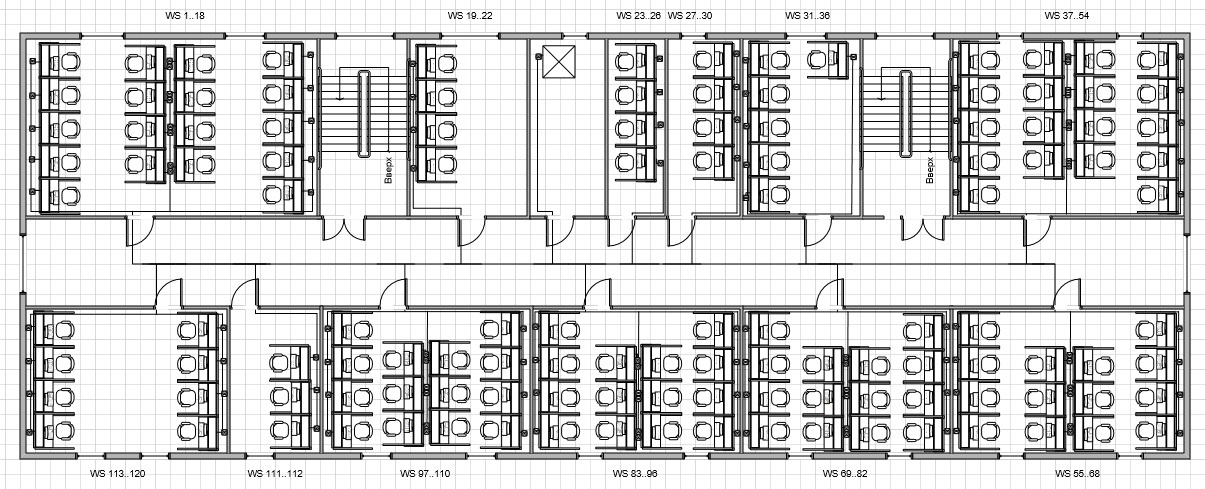


Рисунок 4.4.2 – Структурированная кабельная система 1 этажа

4.5 Коммутационный шкаф

Для хранения сетевого оборудования будем использовать коммутационный шкаф, который находится на втором этаже. Оборудования, которое находится в шкафу представлено в таблице 4.5.1, а сам шкаф представлен на рисунке 4.5.1.

Таблица 4.5.1 – Оборудование коммутационное

|  |  |
| --- | --- |
| Устройство | Наименование |
| Модем | Huawei Е5577Cs-321 |
| Маршрутизатор | Cisco SG100-24 |
| Коммутаторы | TP-Link TL-sg1048 48 ports |
| Сервер управления | DEPO Storm 1360NS |
| Прокси сервер | DEPO Storm 1360NS |
| Сервер печати | HP Jetdirect 695 |
| Сервер базы данных | DEPO Storm 1360NS |
| Веб сервер | DEPO Storm 1360NS |

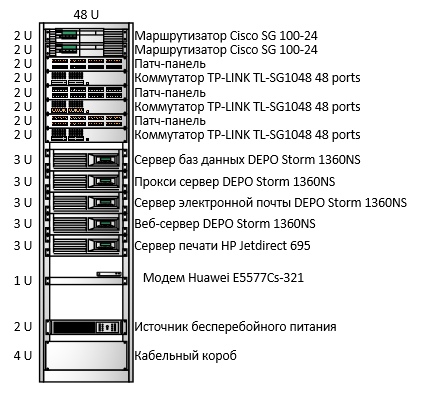


Рисунок 4.5.1 – Коммутационный шкаф

4.6 Расчет полезной пропускной способности сети Ethernet

Различают полезную и полную пропускную способность. Полезная пропускная способность - скорость передачи полезной информации, объем которой всегда меньше полной передаваемой информации, так как каждый передаваемый кадр содержит служебную информацию, гарантирующую его правильную доставку адресату.

Отличие полезной пропускной способности от полной пропускной способности зависит от длины кадра. Так как доля служебной информации всегда одна и та же, то, чем меньше общий размер кадра, тем выше «накладные расходы». Служебная информация в кадрах Ethernet составляет 18 байт (без преамбулы и стартового байта), а размер поля данных кадра меняется от 46 до 1500 байт. Сам размер кадра меняется от 46 + 18 = 64 байт до 1500 + 18 = 1518 байт. Поэтому для кадра минимальной длины полезная информация составляет:

46 / 64 ≈ 0,72,

от общей передаваемой информации, а для кадра максимальной длины:

1500 / 1518 ≈ 0,99,

от общей информации.

Чтобы рассчитать полезную пропускную способность сети для кадров максимального и минимального размера, необходимо учесть различную частоту следования кадров. Чем меньше размер кадров, тем больше таких кадров будет проходить по сети за единицу времени, перенося с собой большее количество служебной информации.

Для передачи кадра минимального размера, который вместе с преамбулой имеет длину 72 байта = 576 бит, потребуется время, равное 576 bt, с учётом межкадрового интервала в 96 bt период следования кадров составит 672 bt. При скорости передачи в 100 Мбит/с это соответствует времени 6,72 мкс. Тогда частота следования кадров, то есть количество кадров, проходящих по сети за 1 секунду, составит 1/6,72 мкс ≈ 148810 кадр/с.

При передаче кадра максимального размера, который вместе с преамбулой имеет длину 1526 байт = 12208 бит, период следования составляет:

12 208 bt + 96 bt = 12 304 bt,

а частота кадров при скорости передачи 100 Мбит/с составит:

1/123,04 мкс = 8127 кадр/с.

Зная частоту следования кадров f и размер полезной информации Vп в байтах, переносимой каждым кадром, по формуле №рассчитывается полезная пропускная способность сети:

Пп (бит/с) = Vп · 8 · f

Для кадра минимальной длины (46 байт) теоретическая полезная пропускная способность равна:

Ппт1 = 148 810 кадр/с = 54,76 Мбит/с,

что составляет лишь немногим больше половины от общей максимальной пропускной способности сети.

Для кадра максимального размера (1500 байт) полезная пропускная способность сети равна:

Ппт2 = 8127 кадр/с = 97,52 Мбит/с

Таким образом, в сети Fast Ethernet полезная пропускная способность может меняться в зависимости от размера передаваемых кадров от 54,76 до 97,52 Мбит/с.

5 Экономическая часть

Стоимость оборудования, которое будет закуплено на учебную компьютерную схему, указана в таблице 5.1.

Таблица 5.1 – Стоимость и количество оборудования

|  |  |  |
| --- | --- | --- |
| Наименование товара | Количество, шт | Цена за ед. наименование, руб |
| Сервер DEPO Storm 1360NT | 5 | 73544 |
| Коммутатор TP-Link TL-sg1048 48 ports | 3 | 18830 |
| Роутер Cisco SG100-24 | 2 | 14959 |
| Модем Huawei E5577Cs-321 | 1 | 8775 |
| LAN Оптический кабель СЛ-ОКМБ-01НУ-4Е2-1,5 | 12 | 47800 |
| Напольный серверный шкаф ЦМО ШТК-СП-48.8.12-48АА-9005 | 1 | 82125 |
| ИБП APC Smart-UPS SMT3000I, 3000ВA | 1 | 92530 |
| Сетевой фильтр MOST EHV, 2м | 1 | 1150 |
| Кабельный органайзер Горизонтальный  ЦМО (ГКО-2-6-9005) односторонний кольца  2U шир.:19" глуб.:85мм | 1 | 580 |
| Патч-панель Lanmaster (TWT-PP48UTP ) 19"  2U 48xRJ45 кат.5e UTP | 3 | 3710 |
| Z83V 4G 64G Windows10 мини ПК Intel Atom x5-Z8350 Quad Core 1000M lan AC dual WiFi HD VGA | 120 | 9508,64 |
| Монитор Iiyama GE2288HS-B1 | 120 | 6000 |
| Клавиатура Genius SlimStar 130, Black | 120 | 683 |
| Мышь Genius NetScroll 120 v2 | 120 | 500 |
| Комплект Dr.Web Универсальный | 8 | 4740 |
| Итого: | 3 264 934руб. | |

По имеющемуся ценам на сетевое оборудование, программного обеспечения и стоимости работ по прокладке кабелей и установке, рассчитывается суммарная стоимость внедрения ЛВС Pвнедр. – необходимые единоразовые капитальные вложения.

По среднему времени производства единицы товара (продукции или услуги) до внедрения t0 и после внедрения ЛВС tЛВС определяется повышение производительности труда организации:

Увеличение ежемесячного дохода от реализации товара при условии сохранения цены на товар и полного объема реализации рассчитывается по формуле (5.1).

ΔД = С × Δn (5.1)

где С – цена за единицу товара.

Срок окупаемости ЛВС в месяцах рассчитывается по формуле (5.2).

(5.2)

где ЗЛВС – ежемесячные затраты на эксплуатацию ЛВС.

Если получившееся значение отрицательно, внедрение ЛВС экономически неэффективно. Если значение срока окупаемости больше 5-7 лет – внедрение ЛВС малоэффективно из-за высокого срока окупаемости.

Срок окупаемости составит 16 месяцев, следовательно внедрение ЛВС имеет смысл.

Рассчитаем экономическую эффективность внедрения ЛВС как изменение ежемесячной прибыли от реализации товара. Цена единицы товара формируется из его себестоимости и плановой прибыли:

Себестоимость единицы товара состоит из отчислений на заработную плату, затрат на сырье и производство, транспортировку и других составляющих, а также амортизации основных средств Ам.

Для учета только амортизации ЛВС примем, что себестоимость единицы товара возрастает только на величину амортизационных отчислений ЛВС. Поскольку мы условились, что цена на товар остается постоянной, величина планируемой прибыли с единицы товара должна быть уменьшена на эту же величину рассчитывается по формуле (5.3).

(5.3)

где Пр0 – прибыль с единицы товара до внедрения ЛВС;

АмЛВС – ежемесячные амортизационные отчисления на ЛВС;‬

nЛВС= n0+ Δn – объем производства после внедрения ЛВС, единиц в месяц.

При пропорциональном способе расчета амортизационных отчисления рассчитываются по формуле (5.4).

(5.4)

где Pостат. – остаточная стоимость ЛВС на конец срока амортизации, принимается равной 10% от стоимости внедрения ЛВС; 1474261,17‬

Ок – законодательно определяемый полный срок амортизации ЛВС в месяцах.

Принимая, что ежемесячные отчисления (аренда, налоги и т.п.) не изменились, ежемесячное изменение прибыли за период амортизации:

Экономическая эффективность рассчитывается как ΔПр / Пр0 и выражается в %.Если получившееся значение отрицательно, внедрение ЛВС нерентабельно.

По результатам вычислений можно сделать вывод, что внедрение ЛВС рентабельно.

Заключение

В рамках данного курсового проекта было проведено изучение сетевых технологий, в частности, была изучена спецификация [1000BASE-CX](https://www.fs.com/ru/products/11774.html), ее назначение, характеристики, достоинства и недостатки.

Была разработана программа – учебник – с помощью языка программирования C#, библиотеки Windows Forms, интегрированной среды разработки Visual Studio.

Были изучены протоколы IPv4, IPv6, а также в ходе выполнения задания были определены: маска подсети, адрес сети, адрес подсетей и количество хостов в сети.

В данной работе было выполнено проектирование информационной сети предприятия, произведен расчет параметров локальной сети, определена стоимость реализации, оценены сроки окупаемости и рентабельности.

В результате выполнения курсовой работы была спроектирована корпоративная сеть предприятия, построена по топологии звезда, в здании располагается один серверный шкаф, к которому подключается 240 рабочих станции. Рабочие станции и сервера получают доступ в интернет через прокси-сервер. Соединение с провайдером осуществляется по ВОЛС кабелю. Также в схему заложена беспроводная сеть Wi-Fi. В сети присутствуют: прокси-сервер, сервер печати, файловый сервер, сервер баз данных и веб сервер.

В работе присутствует экономическое обоснование того, что сеть является рентабельной, срок окупаемости при этом составляет около 22 месяцев.

В процессе выполнения курсовой работы были приобретены навыки построения структурированной кабельной системы и работы с подсетями, использующими протокол IPv4.

Результаты данной работы могут быть применены при изучении технологий локальных сетей, а также при проектировании локально-вычислительной сети предприятия.

Список использованных источников

1. Олифер В., Олифер Н., компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 5-е изд. – СПБ.: Питер, 2016. – 992с.: ил. – («Серия учебник для вузов»).
2. Сети и телекоммуникации: Методические указания к выполнению курсовой работы / Н.В. Будылдина - Екатеринбург: УрТИСИ СибГУТИ, 2017. – 30 с.
3. Протокол IPv4 Ethernet [Электронный ресурс] // just-networks.ru Вычислительные сети, 2014. Режим доступа: http://just-networks.ru/seti-tcpip/protokol-ipv4.
4. Сетевые технологии высокоскоростной передачи данных: Учебное пособие/ Н.В. Будылдина – Екатеринбург: УрТИСИ ФГОБУ ВПО «СибГУТИ», 2014. – 248 с.

Приложение А

Технические характеристика маршрутизатора Cisco SG100-24 представлены в таблице 1.

Таблица 1 – Характеристики маршрутизатора

|  |  |
| --- | --- |
| Процессор | * Atheros AR9344, 600 МГц |
| Интерфейсы | * 10хEthernet 10/100/1000BASE-T * 8x10GBASE-R SFP+/1000BASE-X * 1xConsole (RJ-45) * 2хUSB 2.0 |
| Подключаемые интерфейсы | * USB 3G/4G/LTE модем * E1 TopGate SFP |
| Производительность | * Производительность Firewall/NAT/маршрутизации (фреймы 1518B) - 9,8 Гбит/с; 808 k пакетов/с * Производительность IPsec VPN (фреймы 1456B) - 3,8 Гбит/с; 328 k пакетов/с * Производительность IPS/IDS 10k правил - 493,94 Мбит/с; 92,66 k пакетов/с |
| Системные характеристики | * Количество VPN-туннелей - 500 * Статические маршруты - 11k * Количество конкурентных сессий - 512k * Поддержка VLAN - до 4k активных VLAN в соответствии с 802.1Q * Количество маршрутов BGP - 2,8M * Количество маршрутов OSPF - 500k * Количество маршрутов RIP - 10k * Таблица MAC-адресов - 128k * Размер базы FIB - 1,7M * VRF Lite - 32 |
| Клиенты Remote Access VPN | * PPTP * PPPoE * L2TP * OpenVPN * IPsec XAUTH |
| Сервер Remote Access VPN | * L2TP * PPTP * OpenVPN * IPsec XAUTH |
| Site-to-site VPN | * IPSec: Policy-based и route-based режимы * DMVPN * Алгоритмы шифрования DES, 3DES, AES, Blowfish, Camelia * Аутентификация сообщений IKE MD5, SHA-1, SHA-2 |
| Туннелирование | * IPoGRE, EoGRE * IPIP * L2TPv3 * LT (inter VRF-lite routing) |
| Функции L2 | * Коммутация пакетов (bridging) * Агрегация интерфейсов LAG/LACP (802.3ad) * Поддержка VLAN (802.1Q) * Логические интерфейсы * LLDP, LLDP MED * VLAN на основе MAC |
| Функции L3 (IPv4/IPv6) | * Трансляция адресов NAT, Static NAT, ALG * Статические маршруты * Динамические протоколы маршрутизации RIPv2, OSPFv2/v3, BGP * Фильтрация маршрутов (prefix list) * VRF Lite * Policy Based Routing (PBR) * BFD для BGP, OSPF, статических маршрутов |
| Функции сетевой защиты | * Система обнаружения и предотвращения вторжений (IPS/IDS) * Web-фильтрация по URL, по содержимому (cookies, ActiveX, Javascript) * Zone-based Firewall * Фильтрация фаерволом на базе L2/L3/L4 полей и по приложениям * Поддержка списков контроля доступа на базе L2/L3/L4 полей * Защита от DoS/DDoS атак и оповещение об атаках * Логирование событий атак, событий срабатывания правил |
| Управление IP-адресацией (IPv4/IPv6) | * Статические IP-адреса * DHCP-клиент * DHCP Relay Option 82 * Встроенный сервер DHCP, поддержка опций 43, 60, 61, 150 * DNS resolver * IP unnumbered |

Технические характеристики коммутатора Коммутатор TP-Link TL-SG1048 48 PORTS представлены в таблице 2.

Таблица 2 – Характеристики коммутатора

|  |  |
| --- | --- |
| Порты | * 48 портов 10/100/1000 Мбит/сек |
| Память и процессор | * Двухъядерный процессор Freescale P2020 с частотой 1,2 ГГц * 16 Мбайт флэш-памяти * карта SD на 1 Гбайт * 4 Гбайта памяти DDR3 SODIMM |
| Время задержки | * Задержка при 1000 Мбит/с: < 2,8 мкс * Задержка при 10 Гбит/с: < 1,8 мкс * Задержка при 40 Гбит/с: < 1,5 мкс |
| Пропускная способность | * До 571,4 млн пакетов/с |
| Производительность маршрутизации/коммутации | * 960 Гбит/с |
| Скорость передачи данных коммутатора | * 1015 Гбит/с |

Технические характеристики модема Huawei Е5577Cs-321 представлены в таблице 3.

Таблица 3 – Характеристики модема Huawei Е5577Cs-321

|  |  |
| --- | --- |
| Стандарт беспроводной связи | 802.11 b/g/n/, частота 2.4ГГц, |
| Поддержка MIMO | есть |
| Макс. скорость беспроводного соединения | 300 Мбит/с |
| Защита информации | WEP, WPA, WPA2 |
| Поддержка IPv6 | есть |
| Количество внешних антенн | 1 |
| Протоколы динамической маршрутизации | IGMP v1, IGMP v2 |

Технические характеристики сервера DEPO Storm 1360NS представлены в таблице 4.

Таблица 4 – Характеристики сервера

|  |  |
| --- | --- |
| Процессор | Intel Xeon E3-1200 v3 |
| Оперативная память | 32Гб DDR3-1600/1333 с поддержкой ЕСС. |
| HDD | noHDD (до 8 hdd 2.5″) |
| Модуль удаленного управления | iLO 5 Advanced |
| Сетевая карта | Gigabit Ethernet (10/100/1000Мбит) 1x Intel® i217LM & 1x Intel® i210A |
| RAID controller | HPE Smart Array P408i-a SR (2GB+FBWC) |
| Объем кэша контроллера, Мб | 2000 |
| Форм-фактор | 3U |

Приложение Б

Листинг разработанной и представленной во второй главе проекта программы.

Листинг программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

int theoryFile = 1;

int tcpImage = 1;

int transportTCP = 1;

int countQuest = 1;

int countAswer = 0;

int answerNumber = 0;

List<Boolean> checkTest = new List<Boolean> { false, false, false, false};

public Form1()

{

InitializeComponent();

}

private void openTest()

{

if (checkTest[0] && checkTest[1] && checkTest[2] && checkTest[3])

{

button5.Enabled = true;

}

}

private void button1\_Click(object sender, EventArgs e)

{

button1.Visible = false;

button2.Visible = true;

button3.Visible = true;

button2.Enabled = true;

button12.Visible = false;

button5.Visible = false;

ViewTheoryFiles();

}

private void ViewTheoryFiles()

{

VieweHtmlFile(string.Format("Theory\_page\_{0}.html", theoryFile), webBrowser1);

}

private void viewTest(string fileName)

{

var fileContent = string.Empty;

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "C:\\Users\\taksh\\source\\repos\\WindowsFormsApp1\\WindowsFormsApp1\\";

openFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

openFileDialog.FileName = openFileDialog.InitialDirectory + fileName;

Console.WriteLine(openFileDialog.FileName);

var fileStream = openFileDialog.OpenFile();

using (StreamReader reader = new StreamReader(fileStream))

{

fileContent = reader.ReadToEnd();

}

var flag = true;

while (flag)

{

try

{

var tempStringArray = fileContent.Split('\n');

label1.Text = tempStringArray[0];

radioButton1.Text = tempStringArray[1];

radioButton2.Text = tempStringArray[2];

radioButton3.Text = tempStringArray[3];

answerNumber = int.Parse(tempStringArray[4]);

countQuest += 1;

flag = false;

}

catch

{

System.Threading.Thread.Sleep(500);

}

}

}

}

private void VieweHtmlFile(string fileName, WebBrowser webBrowser)

{

var fileContent = string.Empty;

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "C:\\Users\\taksh\\source\\repos\\WindowsFormsApp1\\WindowsFormsApp1\\";

openFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

openFileDialog.FileName = openFileDialog.InitialDirectory + fileName;

Console.WriteLine(openFileDialog.FileName);

var fileStream = openFileDialog.OpenFile();

using (StreamReader reader = new StreamReader(fileStream))

{

fileContent = reader.ReadToEnd();

}

webBrowser.Visible = true;

var flag = true;

while (flag)

{

try

{

webBrowser.DocumentText = fileContent;

flag = false;

}

catch

{

System.Threading.Thread.Sleep(500);

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (theoryFile < 3)

{

theoryFile += 1;

button3.Enabled = true;

ViewTheoryFiles();

}

if (theoryFile == 3)

{

checkTest[0] = true;

button2.Enabled = false;

button8.Visible = true;

}

}

private void button3\_Click(object sender, EventArgs e)

{

theoryFile -= 1;

if (theoryFile == 1)

{

button3.Enabled = false;

ViewTheoryFiles();

}

if (theoryFile < 3)

{

button2.Enabled = true;

}

ViewTheoryFiles();

}

private void button8\_Click(object sender, EventArgs e)

{

webBrowser1.Visible = false;

button1.Visible = true;

button2.Visible = false;

button3.Visible = false;

button8.Visible = false;

button12.Visible = true;

button5.Visible = true;

openTest();

}

private void button4\_Click(object sender, EventArgs e)

{

button1.Visible = false;

button17.Visible = false;

button9.Visible = true;

button10.Visible = true;

button13.Visible = false;

button4.Visible = false;

button9.Enabled = true;

pictureBox1.Visible = true;

webBrowser2.Visible = true;

var html = "TCP\_page\_{0}.html";

var image = "Установление соединения TCP\_{0}.jpg";

imageBoxWithHtml(pictureBox1, webBrowser2, html, image, tcpImage);

}

private void imageBoxWithHtml(PictureBox pictureBox, WebBrowser webBrowser, string html, string image, int value)

{

pictureBox.Image = Image.FromFile("C:\\Users\\taksh\\source\\repos\\WindowsFormsApp1\\WindowsFormsApp1\\" + string.Format(image, value));

VieweHtmlFile(string.Format(html, value), webBrowser);

}

private void button9\_Click(object sender, EventArgs e)

{

var html = "TCP\_page\_{0}.html";

var image = "Установление соединения TCP\_{0}.jpg";

if (tcpImage < 3)

{

tcpImage += 1;

button10.Enabled = true;

imageBoxWithHtml(pictureBox1, webBrowser2, html, image, tcpImage);

}

if (tcpImage == 3)

{

checkTest[1] = true;

button9.Enabled = false;

button11.Visible = true;

}

}

private void button11\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button1, button12, button5 };

List<Button> button11ListFalse = new List<Button> { button9, button10, button11 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox1.Visible = false;

webBrowser2.Visible = false;

openTest();

}

private void button10\_Click(object sender, EventArgs e)

{

var html = "TCP\_page\_{0}.html";

var image = "Установление соединения TCP\_{0}.jpg";

tcpImage -= 1;

if (tcpImage == 1)

{

button10.Enabled = false;

imageBoxWithHtml(pictureBox1, webBrowser2, html, image, tcpImage);

}

if (tcpImage < 3)

{

button9.Enabled = true;

}

imageBoxWithHtml(pictureBox1, webBrowser2, html, image, tcpImage);

}

private void button12\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button4, button13, button17 };

List<Button> button11ListFalse = new List<Button> { button1, button12, button5 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

}

private void button13\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button14, button15 };

List<Button> button11ListFalse = new List<Button> { button4, button13, button17 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox2.Visible = true;

transportTCP = 1;

var format = "Передача {0}.png";

imageBox(pictureBox2, format, transportTCP);

}

private void imageBox(PictureBox pictureBox, string format, int value)

{

pictureBox.Image = Image.FromFile("C:\\Users\\taksh\\source\\repos\\WindowsFormsApp1\\WindowsFormsApp1\\" + string.Format(format, value));

}

private void button14\_Click(object sender, EventArgs e)

{

var format = "Передача {0}.png";

if (transportTCP < 7)

{

transportTCP += 1;

button15.Enabled = true;

imageBox(pictureBox2, format, transportTCP);

checkTest[2] = true;

}

if (transportTCP == 7)

{

button14.Enabled = false;

button16.Visible = true;

}

}

private void button15\_Click(object sender, EventArgs e)

{

transportTCP -= 1;

var format = "Передача {0}.png";

if (transportTCP == 1)

{

button15.Enabled = false;

imageBox(pictureBox2, format, transportTCP);

}

if (transportTCP < 7)

{

button14.Enabled = true;

}

imageBox(pictureBox2, format, transportTCP);

}

private void button16\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button1, button5, button12 };

List<Button> button11ListFalse = new List<Button> { button14, button15, button16 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox2.Visible = false;

openTest();

}

private void button20\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button14, button15 };

List<Button> button11ListFalse = new List<Button> { button4, button13 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox2.Visible = true;

transportTCP = 1;

var format = "Передача {0}.png";

imageBox(pictureBox2, format, transportTCP);

}

private void button17\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button18 };

List<Button> button11ListFalse = new List<Button> { button4, button12, button13, button17 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox1.Visible = true;

webBrowser2.Visible = true;

var html = "end\_{0}.html";

var image = "end\_{0}.png";

checkTest[3] = true;

imageBoxWithHtml(pictureBox1, webBrowser2, html, image, 1);

}

private void button18\_Click(object sender, EventArgs e)

{

List<Button> button11ListTrue = new List<Button> { button1, button12, button5 };

List<Button> button11ListFalse = new List<Button> { button17, button18 };

foreach (Button button in button11ListFalse)

button.Visible = false;

foreach (Button button in button11ListTrue)

button.Visible = true;

pictureBox1.Visible = false;

webBrowser2.Visible = false;

openTest();

}

private void button5\_Click(object sender, EventArgs e)

{

viewTest(string.Format("question\_{0}.txt", countQuest));

button7.Visible = true;

button1.Visible = false;

button12.Visible = false;

button5.Visible = false;

label1.Visible = true;

radioButton1.Visible = true;

radioButton2.Visible = true;

radioButton3.Visible = true;

}

private void button7\_Click(object sender, EventArgs e)

{

List<RadioButton> radioList = new List<RadioButton> { radioButton1, radioButton2, radioButton3 };

if (radioList[0].Checked || radioList[1].Checked || radioList[2].Checked)

if (radioList[answerNumber - 1].Checked == true)

countAswer += 1;

if (countQuest == 11)

{

label1.Visible = false;

foreach (var radio in radioList)

radio.Visible = false;

label2.Visible = true;

button7.Visible = false;

label2.Text = string.Format("Вы ответили правильно на {0} вопросов из {1}", countAswer, countQuest - 1);

}

else

viewTest(string.Format("question\_{0}.txt", countQuest));

}

private void button6\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}