Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: «Операционные среды и системное программирование»

К защите допустить
И.о. заведующего кафедрой информатики
_____ С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

УТИЛИТА ДЛЯ АВТОМАТИЧЕСКОЙ НАСТРОЙКИ СЕТЕВЫХ ПАРАМЕТРОВ (IP-АДРЕСА, МАСКА ПОДСЕТИ, ШЛЮЗЫ) НА ОСНОВЕ АНАЛИЗА СЕТЕВОЙ ИНФРАСТРУКТУРЫ

БГУИР КП 1-40 04 01 01 001

Студент К.А. Тимофеев
Руководитель Н.Ю. Гриценко
Нормконтроль Н.Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

Введение 6
1 Настраиваемые параметры сетевой инфраструктуры 10
2 Платформа программного обеспечения 14
3 Теоретическое обоснование разработки программного продукта 14
4 Проектирование функциональных возможностей программы 14
5 Выполнение программы в разных сетевых средах 14
Заключение 14
Список использованных источников 21
Приложение А (обязательное) Листинг исходного кода 22

ВВЕДЕНИЕ

В настоящее время сетевые технологии играют ключевую роль в работе организаций и предприятий, обеспечивая связность и эффективность бизнес-процессов. Современные сетевые технологии позволяют быстро и без потерь передавать колоссальные объемы данных, обеспечивая развитие современного интернета.

Не меньшее значение сетевые технологии имеют в сфере бизнеса. Позволяя быстро передавать многочисленные пакеты данных, они позволяют значительно повысить скорость управления на больших предприятиях.

Однако для обеспечения стабильной работы сетей необходимо правильно настраивать сетевые параметры, такие как IP-адреса, маска подсети и шлюзы. Неэффективное использование доступных IP-адресов может привести к конфликтам, перегрузке сети и снижению производительности.

Для оптимизации использования ресурсов и предотвращения конфликтов необходимо регулярно анализировать сетевую инфраструктуру и настраивать сетевые параметры на основе полученных данных. Такой подход позволит оптимально использовать доступные IP-адреса.

Но при регулярном изменении количества устройств в сети регулярный процесс анализа сетевой инфраструктуры будет занимать неоправданно большое количество времени. Использование автоматической утилиты для настройки сетевых параметров не только повысит эффективность работы сети, но и значительно сократит время, затрачиваемое IT-специалистами на рутинные задачи по настройке сетевых параметров. В итоге это позволит сосредоточиться на более важных задачах и повысить общую производительность и надежность сети.

Дополнительным преимуществом использования автоматической утилиты для настройки сетевых параметров является уменьшение вероятности человеческих ошибок при конфигурации сети. Автоматизация процесса настройки позволяет исключить случайные ошибки, которые могут возникнуть при ручной настройке сетевых параметров.

В целом, автоматизация настройки сетевых параметров при помощи специализированных утилит позволяет значительно повысить эффективность работы сети, обеспечить стабильность и надежность ее функционирования, а также уменьшить трудозатраты на администрирование сети.

1 НАСТРАИВАЕМЫЕ ПАРАМЕТРЫ СЕТЕВОЙ ИНФРАСТУКТУРЫ

Сетевая инфраструктура представляет собой комплекс элементов, необходимых для обеспечения надежной и эффективной работы сети. Важными компонентами сетевой инфраструктуры являются IP-адреса, подсети, шлюзы, DNS-сервера, DHCP-серверы, VLANы, маршруты, системы безопасности (включая файерволы и VPN), прокси-серверы, NAT (Network Address Translation) и системы мониторинга и управления сетью. Каждый из этих элементов играет свою роль в обеспечении стабильной работы сети и обеспечении безопасности передаваемых данных. Управление и настройка всех этих компонентов сетевой инфраструктуры являются ключевыми задачами сетевых администраторов для обеспечения эффективной работы организации.

IP-адрес и маска подсети: IP-адрес является уникальным идентификатором устройства в сети, который используется для маршрутизации данных. Каждому устройству в сети присваивается уникальный IP-адрес, который позволяет маршрутизаторам определять путь передачи данных. IP-адреса используются для идентификации устройств в сети, обеспечивая возможность обмена данными между ними.

Стандартный IP-адрес использует 32-битную адресацию, которая обычно записывается как четыре числа от 0 до 255, разделенные точками, например, 192.168.1.1. Всего возможно около 4 миллиардов уникальных IPv4-адресов.

Маска подсети определяет, какая часть IP-адреса относится к адресу сети, а какая - к адресу устройства в этой сети. Маска подсети состоит из последовательности битов, которые определяют длину сетевой части и длину адреса устройства в сети. Для простоты записи маску принято записывать в виде десятичного числа, обозначающего количество бит в маске, которые начиная с первого бита маски имеют значение 1.

Проведя операцию конъюнкции над битами маски и IP-адреса можно получить адрес сети устройства. Оставшиеся биты обозначают адрес хоста в подсети. Этот принцип изображен на рисунке 1.1.

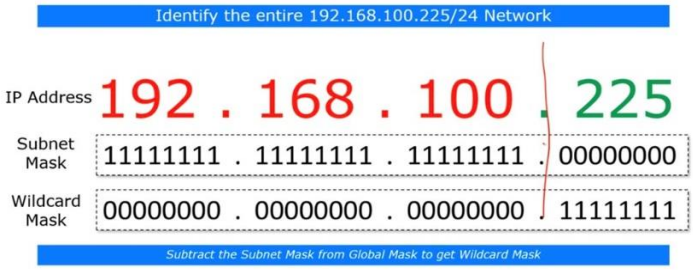

Рисунок 1.1 – Принцип работы маски подсети

На данный момент существует два стандарта IP-адресов: IPv4 и IPv6. Для расширения возможного адресного пространства в IPv6 расширили размер адреса до 128 бит, что позволяет иметь огромное количество уникальных адресов. Исходя из этого запись адреса в IPv6 представляется в виде шестнадцатеричных чисел, разделенных точкам двоеточиями. Примеры записи IPv6-адреса приведены на рисунке 1.2.

Рисунок 1.2 – Пример IPv6-адреса

Однако из-за сложностей перехода и настройки IPv6 основным стандартом все еще считается IPv4.

В IPv4 существуют четыре основных класса IP-адресов: A, B, C и D.

- класс A (от 1.0.0.0 до 126.255.255.255) обычно используется для крупных организаций;
- класс B (от 128.0.0.0 до 191.255.255.255) используется для средних организаций;
- класс C (от 192.0.0.0 до 223.255.255.255) используется для малых организаций и домашних сетей;
- класс D (от 224.0.0.0 до 239.255.255.255) зарезервирован для многоадресных групп.

Сетевые шлюзы: сетевой шлюз (или просто шлюз) - это устройство или программное обеспечение, которое обеспечивает связь между различными

сетями, позволяя устройствам в одной сети общаться с устройствами в другой сети. Шлюзы имеют множество функций:

- маршрутизация: шлюз принимает пакеты данных от устройств в одной сети и передает их в другую сеть, выбирая оптимальный путь для доставки;

- перевод адресов (NAT): шлюз может изменять IP-адрес отправителя или получателя в пакетах данных, обеспечивая прозрачное взаимодействие между локальной сетью и интернетом;

- фильтрация трафика: шлюз может блокировать или разрешать определенные типы трафика на основе правил безопасности;

- проксирование: шлюз может действовать как посредник между устройствами и серверами, управляя запросами и ответами для повышения производительности сети.

Сетевыми шлюзами могут быть маршрутизаторы или специальные прокси-серверы. Так же существует особый тип шлюзов под названием Firewall. Это специальные сервера, обеспечивающие защиту сети от несанкционированного доступа.

Шлюзы играют важную роль в обеспечении связности и безопасности сетей, поэтому их правильная настройка и управление являются ключевыми задачами сетевых администраторов.

DHCP: DHCP (Dynamic Host Configuration Protocol) - это протокол, который позволяет устройствам в компьютерной сети автоматически получать IP-адрес, шлюз по умолчанию, DNS-серверы и другие сетевые настройки от DHCP сервера.

DHCP сервер предоставляет автоматическую настройку сетевых параметров для устройств в сети, что облегчает администрирование сети и уменьшает вероятность ошибок при конфигурации устройств.

Когда устройство подключается к сети, оно отправляет запрос на получение IP-адреса DHCP серверу. DHCP сервер выделяет свободный IP-адрес из своего пула адресов и предоставляет его устройству на определенное время. Общий формат работы DHCP сервера по выдаче адресов продемонстрирован на рисунке 1.3.

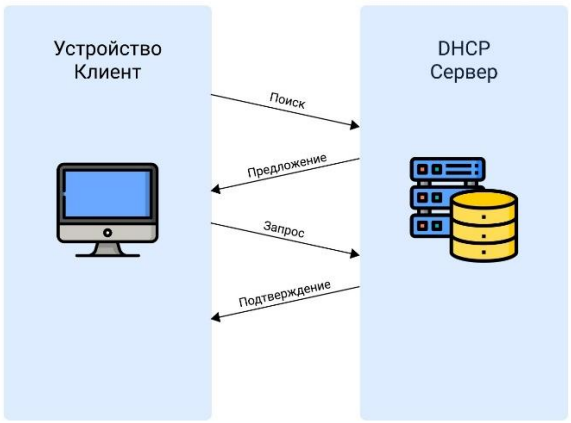

Рисунок 1.3. – Схема выдачи IP-адресов DHCP-сервером

DHCP сервер имеет определенный диапазон IP-адресов, из которого он выделяет адреса клиентам. Этот диапазон должен быть настроен таким образом, чтобы не пересекаться с уже используемыми статическими адресами в сети. Администратор сети может зарезервировать определенные IP-адреса для конкретных устройств (например, для серверов или принтеров), чтобы избежать конфликтов адресов.

DHCP сервер назначает каждому устройству IP-адрес на определенное время, называемое «временем аренды». По истечении этого времени устройство должно обновить свой IP-адрес у DHCP сервера.

Помимо IP-адреса, DHCP сервер также может предоставлять другие параметры сети, такие как шлюз по умолчанию, DNS-серверы, маску подсети т.д.

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Платформа программного обеспечения — это среда, в которой работают приложения. Она включает в себя операционную систему, которая управляет аппаратными ресурсами компьютера, а также программные инструменты и библиотеки, которые используются для разработки приложений. Для разработки данного проекта была выбрана ОС Ubuntu, являющаяся членом семейства операционных систем Linux. В качестве языка для написания программы был выбран язык общего назначения C++.

ОС Ubuntu и семейство Linux: Linux — это семейство Unix-подобных операционных систем, основанных на ядре Linux. Они включают в себя различные утилиты и программы проекта GNU. Ядро Linux является монолитным ис состоит из многих компонентов:

- Linux framebuffer USB core — подсистема для поддержки USB-устройств и контроллеров шины USB;
- Evdev (англ. event device) — компонент для работы с устройствами ввода (клавиатурой, джойстиком, мышью);
- Kvm (англ. kernel-based virtual machine);
- DRM (англ. direct rendering manager);
- Cgroups (англ. control groups);
- Dm (англ. device mapper);
- SELinux (англ. security-enhanced Linux) и многие другие.

Ядро Linux обладает мощными возможностями для взаимодействия с сетью и предоставляет программные интерфейсы для реализации различных сетевых функций. Ядро Linux поддерживает широкий спектр сетевых протоколов, таких как TCP/IP, UDP, ICMP, HTTP, FTP и другие, что позволяет разработчикам создавать сетевые приложения, работающие поверх этих протоколов. Также ядро предоставляет возможности управления сетевыми устройствами такими как сетевые карты, мосты, VLAN и т. д. Это позволяет настраивать и контролировать сетевые соединения на уровне ядра.

В ядре Linux есть API для работы с сокетами, что позволяет приложениям устанавливать и управлять сетевыми соединениями. Кроме того, ядро предоставляет интерфейсы для работы с сетевыми устройствами и настройки сетевых параметров.

В ядре Linux встроены механизмы безопасности для защиты сетевых соединений и данных. Это включает в себя механизмы аутентификации, шифрования данных, контроль доступа и другие функции безопасности.

Ядро Linux также предоставляет поддержку для различных сетевых сервисов, таких как DNS, DHCP, NAT, маршрутизация и др. Это позволяет создавать разнообразные сетевые приложения и сервисы на базе ядра Linux. Схема сетевой подсистемы ядра представлена на рисунке 2.1.

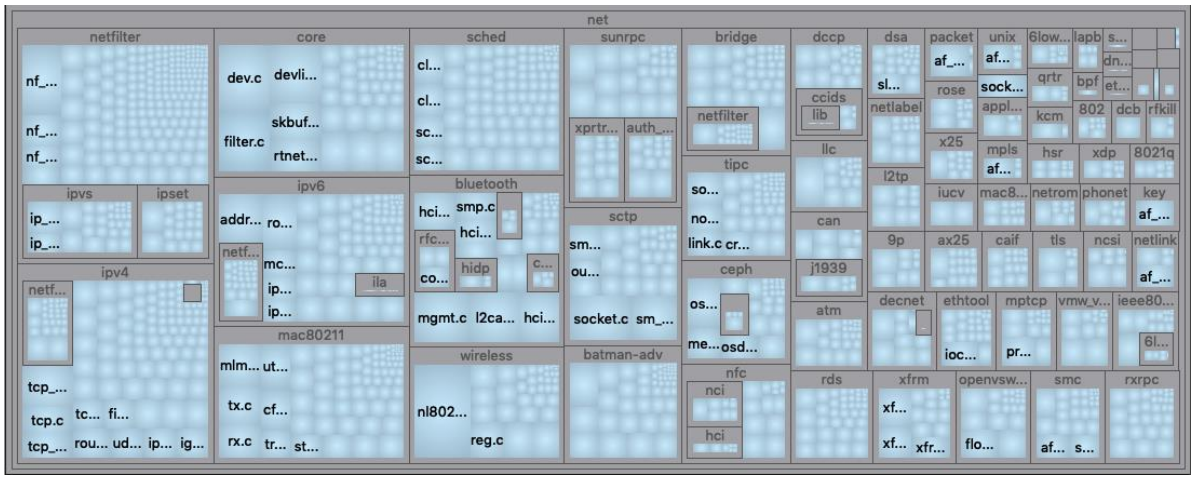

Рисунок 2.1 – Сетевая подсистема ядра Linux

Ubuntu - это один из самых популярных дистрибутивов Linux, который основан на ядре Linux. Исходный код данного дистрибутива свободен, что позволяет пользователям и разработчикам свободно использовать, изменять и распространять его. Популярность данного дистрибутива обеспечили удобный графический интерфейс, минимальные системные требования и поддержка популярных приложений, из-за чего пользоваться Ubuntu может человек, не знакомый с командной строкой.

Программный интерфейс работы с сетью в Ubuntu взят из ядра. Благодаря этому программы могут без лишних затрат быть портированы на другие дистрибутивы Linux.

Язык программирования C++: C++ является компилируемым и статически типизированным языком программирования общего назначения. Одной из его особенностей является мультипарадигмальность, что означает поддержку различных парадигм программирования, включая процедурное, объектно-ориентированное и обобщённое программирование.

Богатая стандартная библиотека C++ предлагает разнообразные контейнеры и алгоритмы, возможности ввода-вывода, поддержку регулярных выражений и многопоточности. Это делает C++ удобным инструментом для разработчиков. C++ также отличается гибкостью, сочетая свойства как высокоуровневых, так и низкоуровневых языков. Это делает его подходящим

для широкого спектра задач, от программирования микроконтроллеров до разработки сложных систем. На рисунке 2.2 изображен список встроенных в язык C++ библиотек.

Header File	Library	man Pages
	C++ STL	
string	STL strings type	std::string
sstream	stringstream, for writing to strings as if they are streams	std::stringstream
iostream	C++ standard stream library	std::ios, std::iostream
memory	C++ memory-related routines	std::bad_alloc, std::auto_ptr
	C Standard Library	
cstring, string.h	Functions for C `char*` strings	string, strcpy, strcmp
cstdlib, stdlib.h	C Standard Library	random, srandom, getenv, setenv
cstdio, stdio.h	Standard input/output	stdin, stdout, printf, scanf
cassert	Assert macros	assert

Рисунок 2.2 – Список стандартных библиотек

Синтаксис C++ унаследован от языка C, что обеспечивает совместимость с ним. Однако, стоит отметить, что C++ не является в строгом смысле надмножеством C. Одной из ключевых особенностей C++ является поддержка объектно-ориентированного программирования, что позволяет создавать модульные и повторно используемые программы.

Наконец, C++ имеет активное сообщество разработчиков и множество ресурсов для изучения, что делает его одним из наиболее мощных и гибких языков программирования.

Для взаимодействия с сетями в каждой операционной системе предусмотрен свой набор программных интерфейсов. Для дистрибутивов Linux предусмотрен заголовочный файл <sys/socket.h>. Он позволяет устанавливать API сокетов для работы с сетью.

В операционной системе Linux существуют различные программные интерфейсы для настройки IP-адресов, шлюзов и масок подсети. Один из наиболее распространенных методов - это использование системного вызова ioctl. Этот метод позволяет получить информацию о сетевом интерфейсе, например, его IP-адрес. Однако, для более сложных задач, таких как настройка IP-адресов, шлюзов и масок подсети, можно использовать инструмент командной строки nmcli.

nmcli - это мощный инструмент, который позволяет управлять сетевыми настройками прямо из командной строки. Например, вы можете использовать nmcli для настройки статического IP-адреса, шлюза и DNS-

сервера. Это делается с помощью одной простой команды, которую можно вызвать из программы на C++, используя функцию system.

Visual Studio Code: в качестве среды разработки был выбран текстовый редактор Visual Studio Code от компании Microsoft. Редактор доступен для Windows, Linux и macOS. Он предлагает множество функций, таких как подсветка синтаксиса, автодополнение кода с помощью IntelliSense, интеграция с Git, инструменты профилирования и маркетплейс с более чем 9000 расширений. Visual Studio Code поддерживает различные языки программирования, фреймворки и инструменты, что делает его идеальным для кроссплатформенной разработки. Установка VS Code проста и возможна на всех популярных операционных системах. Несмотря на свою легкость и быстродействие, VS Code обладает всем необходимым функционалом и подходит для работы даже на не очень мощных компьютерах. Простота освоения делает его доступным для начинающих разработчиков, в то время как его мощные функции и гибкость делают его полезным инструментом для опытных программистов. Интерфейс программы приведен на рисунке 2.3.

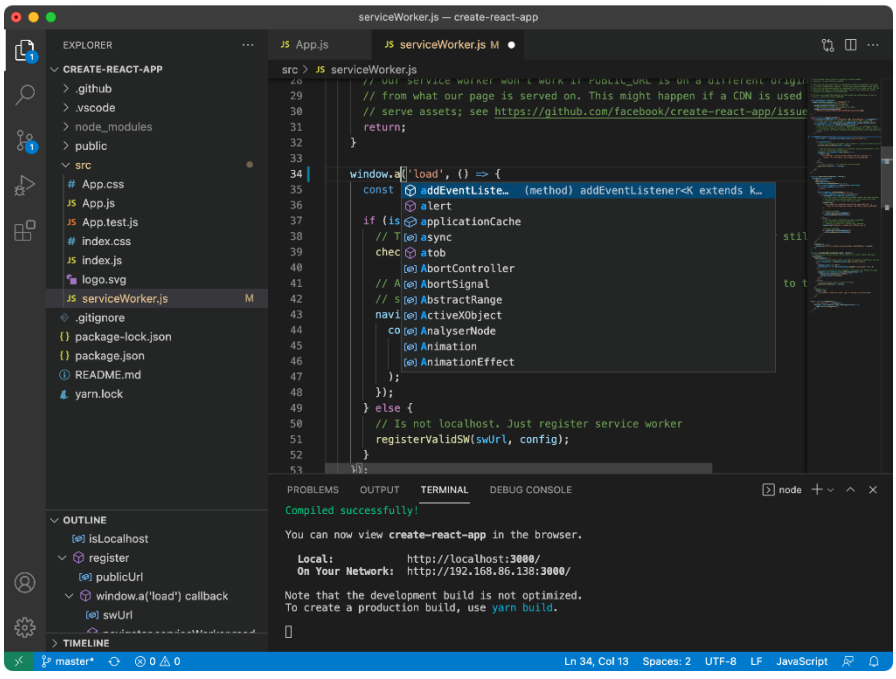

Рисунок 2.3 – Интерфейс редактора Visual Studio Code

Выбранная платформа программного обеспечения использует популярные и проверенные временем решения.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Существующие виды назначения IP-адресов: Существует несколько различных видов назначения IP-адресов в компьютерных сетях. Они могут быть разделены на две основные категории: статическое и динамическое назначение IP-адресов.

Первый вариант это статическое назначение IP-адресов. Оно может быть двух видов:

- вручную: администратор сети назначает IP-адрес каждому устройству вручную. Этот метод требует постоянного вмешательства администратора и может быть неэффективным в больших сетях;

- автоматически: администратор настраивает сервер DHCP (Протокол динамической конфигурации хоста), который автоматически назначает IP-адрес каждому устройству в сети. Этот метод более эффективен, чем ручное назначение IP-адресов, особенно в больших сетях.

Второй вид это динамическое назначение IP-адресов. Их так же существуют два вида:

- DHCP (Протокол динамической конфигурации хоста): DHCP-сервер автоматически назначает IP-адрес устройствам в сети при их подключении. Каждое устройство получает временный IP-адрес из пула доступных адресов. Этот метод упрощает процесс управления IP-адресами и позволяет эффективно использовать доступные адреса.

- APIPA (Автоматическое назначение частных IP-адресов): если DHCP сервер недоступен, устройство может назначить себе временный IP-адрес из диапазона 169.254.0.1 - 169.254.255.254. Это позволяет устройствам в локальной сети общаться друг с другом, даже если нет доступного DHCP сервера.

Выбор маски подсети: выбор маски подсети для IP-адреса зависит от нескольких факторов.

Первым фактором является тип сети. Для локальных сетей выбирается маска /24, а для крупных сетей или для сетей провайдеров выбираются маски /16 или /8.

Вторым фактором является количество устройств в настраиваемой сети. Количество битов, которые маска оставляет для обозначения номера устройства в сети, должно вмещать в себя количество устройств в сети или превышать.

Так же стоит учитывать наличие подсетей.

Особые IP-адреса: перед присвоением адреса стоит проверить, не является ли данный адрес одним из зарезервированных. Если первый октет сети начинается со 127, такой адрес считается адресом машины – источника пакета. В этом случае пакет не выходит в сеть, а возвращается на компьютер-отправитель. Такие адреса называются loopback (петля, замыкание на себя) и используются для проверки функционирования стека TCP/IP.

Если все биты IP-адреса равны нулю, адрес обозначает узел отправитель и используется в некоторых сообщениях ICMP.

Если все биты IP-адреса сети равны 0, а все биты IP-адреса хоста равны 1, то адрес называется ограниченным широковещательным (Limited Broadcast). Пакеты, направленные по такому адресу, рассылаются всем узлам той подсети, в которой находится отправитель пакета. К данному же типу адреса можно относить и IP-адрес, в котором 32 разряда заполнены 1, так как пакет с таким адресом не будет «выпущен» за пределы сети отправителя устройством, связующим сети, например коммутатром или маршрутизатором.

Если все биты IP-адреса хоста равны 1, а биты IP-адреса сети не равны 0, т. е. однозначно идентифицируют адрес подсети, то адрес называется широковещательным (Broadcast); пакеты, имеющие широковещательный адрес, доставляются всем узлам подсети назначения.

Частные и публичные IP-адреса: Поскольку каждый узел сети Интернет должен обладать уникальным IP-адресом, то, безусловно, важной является задача координации распределения адресов отдельным сетям и узлам. Такую координирующую роль выполняет интернет-корпорация по распределению имен и адресов (the Internet Corporation for Assigned Names and Numbers, ICANN).

Служба распределения номеров IANA (Internet Assigned Numbers Authority) зарезервировала для частных сетей следующие три блока адресов:

- 10.0.0.0–10.255.255.255 (1 сеть класса A);
- 172.16.0.0–172.31.255.255 (16 сетей класса B);
- 192.168.0.0–192.168.255.255 (256 сетей класса C).

Любая организация может использовать IP-адреса из этих блоков без согласования с ICANA или Internet-регистраторами. В результате эти адреса используются во множестве организаций. Таким образом, уникальность адресов сохраняется только в масштабе одной или нескольких организаций, которые согласованно используют общий блок адресов. В такой сети каждая рабочая станция может обмениваться информацией с любой другой рабочей станцией частной сети.

Перед распределением адресов из частного и публичного блоков следует определить, какие из рабочих станций сети должны иметь связь с внешними системами на сетевом уровне. Для таких рабочих станций следует использовать публичные адреса, остальным же – можно присваивать адреса из частных блоков, это не помешает им взаимодействовать со всеми рабочими станциями частной сети организации, независимо от того, какие адреса используются (частные или публичные). Однако прямой доступ во внешние сети для рабочих станций с адресами из частного блока невозможен. Для организации их доступа во внешние шлюзы придется использовать прокси-серверы. Перемещение рабочей станции из частной сети в публичную (и обратно) связано со сменой IP-адреса, соответствующих записей DNS и изменением конфигурационных файлов на других рабочих станцях, которые их идентифицируют по IP-адресам. Поскольку частные адреса не имеют глобального значения, маршрутная информация о частных сетях не должна выходить за пределы этих сетей, а пакеты с частными адресами отправителей или получателей не должны передаваться через межсетевые каналы. Предполагается, что маршрутизаторы в публичных сетях (особенно маршрутизаторы провайдеров Internet) будут отбрасывать маршрутную информацию из частных сетей. Если маршрутизатор публичной сети получает такую информацию, ее отбрасывание не должно трактоваться как ошибка протокола маршрутизации.

Шлюзы: Шлюз является наиболее сложной ретрансляционной системой, обеспечивающей взаимодействие сетей с различными наборами протоколов всех семи уровней. В свою очередь, наборы протоколов могут опираться на различные типы физических средств соединения. На рисунке 3.1 изображена структура работы шлюза.

Рисунок 3.1 – Структура работы шлюза

При соединении информационных сетей часть уровней может иметь одни и те же протоколы. В этом случае сети соединяются не при помощи шлюза, а на основе более простых ретрансляционных систем, именуемых маршрутизаторами и мостами.

В качестве шлюза обычно используется выделенный компьютер, на котором запущено программное обеспечение шлюза и производятся преобразования, позволяющие взаимодействовать нескольким системам в сети. Другой функцией шлюзов является преобразование протоколов. При получении сообщения IPX/SPX для клиента TCP/IP шлюз преобразует сообщения в протокол TCP/IP.

Таким образом, для правильной настройки сетевых параметров необходимо знать, является сеть локальной или глобальной. Также нужно знать количество устройств в сети, и их назначение в общем устройстве сети.

4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

Получение информации о сетевой инфраструктуре: функция get_network_info() открывает процесс для выполнения команды ip route show, которая выводит информацию о маршрутах IP. Результат команды сохраняется в строку и возвращается из функции. Это позволяет пользователю получить обзор текущей сетевой конфигурации устройства.

Настройка сетевых параметров: Функция configure_network() принимает IP-адрес, маску подсети и адрес шлюза в качестве параметров и использует команды ifconfig и route для настройки сетевых параметров устройства. Это обеспечивает возможность изменения конфигурации сети без необходимости вручную вводить команды в терминале.

Автоматическое определение параметров: параметры сетевой конфигурации вычисляются исходя из текущего количества хостов в сети и их адресов. Зная текущее количество хостов, можно вычислить оптимальную маску. Далее перебирая доступные адреса в текущей подсети, вычисляется первый доступный адрес. Если этот адрес не является зарезервированным, то он подается на выход функции. Если этот адрес зарезервирован, то поиск свободного адреса продолжается.

Такой алгоритм позволяет динамически изменять параметры исходя из количества подключенных к сети устройств и типа их IP-адресов. При добавлении нового устройства в сеть достаточно запустить утилиту на всех нужных устройствах, и новые IP-адреса будут выданы.

Обработка ошибок: в коде предусмотрены проверка на успешность открытия процесса для выполнения команды. Ошибки могут возникнуть на стороне системы, поэтому результат каждого вызова системного API проверяется на ожидаемое поведение. При ошибках, программа прекратит выполнение и выведет сообщение об ошибке.

Ввод параметров: на вход программы можно подать адрес шлюза виде параметра флага –g. Таким образом, пользователю необходимо ввести корректный адрес шлюза, и шлюз будет применен к устройству.

Возможные улучшения: утилиту можно доработать. Возможны как расширение текущих возможностей, так и повышение удобства использования программы.

Вместо IP-адреса действующего шлюза можно передавать в параметры адрес сервера, отправив TCP запрос на который можно узнать адрес шлюза устройства с данным MAC-адресом. Также на сервере можно хранить файл

настройки в установленном формате, который позволял бы однозначно установить шлюзы всех устройств, использующих эту утилиту. Таким образом, системный администратор может назначить шлюзы, потом запустив программу и получив файл конфигурации автоматически распределить все устройства по шлюзам и выдать исходя из их расположения в сети IP-адреса. В таком случае, устройство, хранящее файл конфигурации, должно иметь постоянный в рамках данной сети IP-адрес, который можно будет указать при вызове программы.

Также вместо ручного вызова программы ее можно сделать фоновым процессом, который при определенных условиях будет автоматически проверять параметры сетевой конфигурации раз в определенное количество времени. Таким образом при добавлении новых устройств в сеть не будет необходимости в ручном запуске программы.

Совместив два вышеописанных подхода, можно получить приложение, работающее в фоновом режиме с файлом конфигурации сети. При изменении файла программа будет динамически переназначать свои сетевые параметры, отчего задачей сетевого администратора станет только составление файла конфигурации сети на конфигурационном сервере.

ВЫВОДЫ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг исходного кода

Листинг 1 – Программный код разработанного приложения

```
#!/bin/bash

function getsize(){
	# local tmpfile="$(($RANDOM+0))_$(($RANDOM+0)).txt"
	local res=$(du "$1")

	if [[ "$res" =~ ([0-9]+) ]];
	then
	return ${BASH_REMATCH[1]}
	else
		echo "unable to parse string $res"
		exit 1
	fi

}
function sizeComparator(){
	local testA=0
	local testB=0

	getsize $1
	local size1=$?
	getsize $2
	local size2=$?
	echo "size1 = $size1 ; size2 = $size2"

	if ((size1 == size2))
	then
		# echo "equal length"
		return 1
	fi

	(( size1 < size2 )) || testA=1
	[ "$3" = "desc" ] || testB=1

	if [ "$testA" -ne "$testB" ]
	then
		return 1
	else
		return 0
	fi
}
function getdate(){
	local res=$(stat "$1")
	if [[ "$res" =~ ([0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}) ]];
	then
		retval=${BASH_REMATCH[1]}
	else
		echo "can't parse date of $1"
	fi
}
function dateComparator(){
	local testA=0
	local testB=0

	getdate $1
	local date1=$retval
	getdate $2
	local date2=$retval

	if [ "$date1" = "$date2" ];
	then
		return 1
	fi

	[[ "$date1" < "$date2" ]] || testA=1
	[ "$3" = "desc" ] || testB=1
	if [ "$testA" -ne "$testB" ]
	then
```

```
                return 1
        else
                return 0
        fi
}
function alphabetComparator(){
        local testA=0
        local testB=0
        [[ "$1" < "$2" ]] || testA=1
        [ "$3" = "desc" ] || testB=1

        if  [ "$testA" -ne "$testB" ]
        then
                return 1
        else
                return 0
        fi
}
function namelengthComparator(){
        local testA=0
        local testB=0
        echo "length of $1 = ${#1} length of $2 = ${#2}"
        if ((${#1} == ${#2}))
        then
                # echo "equal length"
                return 1
        fi

        (( ${#1} < ${#2} )) || testA=1
        [ "$3" = "desc" ] || testB=1

        if [ "$testA" -ne "$testB" ]
        then
                return 1
        else
                return 0
        fi
}
function random(){
        local arraysize=${#filesarr[@]}
        local randcount
        local i
        ((randcount = arraysize * 2 ))
        echo "count = $randcount"
        for ((i=0; i < randcount; ++i))
        do
                local i1
                ((i1=$RANDOM % arraysize))
                local i2
                ((i2=$RANDOM % arraysize))
                # echo "i1= $i1 i2=$i2"
                local tmp=${filesarr[i1]}
                filesarr[$i1]=${filesarr[i2]}
                filesarr[$i2]=$tmp
        done
}
function mysort(){
        local comparator=$1
        local i
        local j
        for ((i=0; i < ${#filesarr[@]}; ++i))
        do
                for ((j=0; j < ${#filesarr[@]} - i - 1; ++j ))
                do
                        # echo "i = $i j= $j"
                        if  $comparator ${filesarr[j]} ${filesarr[j+1]} $2
                        then
                                # echo "swap"
                                local tmp=${filesarr[j]}
                                filesarr[$j]=${filesarr[j+1]}
                                filesarr[((j+1))]=$tmp
                        fi
                done
        done
}
function undoNumbers(){
        for ((i = 0; i < ${#filesarr[@]}; ++i))
        do
```

```
                    local filename=${filesarr[i]}
                    if [[ $filename =~ ^[0-9]{4}\.(.*) ]];
                    then
                              # echo "filename without number ${BASH_REMATCH[1]}"
                              mv "$filename" "${BASH_REMATCH[1]}"
                    fi
          done
}
function getFiles(){
          local tmpfile="$(($RANDOM+0))_$(($RANDOM+0)).txt"
          $(ls -p >> $tmpfile)
          local tmpfile2="$(($RANDOM+0))_$(($RANDOM+0)).txt"
          grep ".*[^/]$" $tmpfile >> $tmpfile2
          rm $tmpfile
          local i=0
          while read line;do
                    if [ $line = $tmpfile ] || [ $line = $tmpfile2 ] || [ $line = $scriptname ]
                    then
                              continue
                    fi
                    filesarr[$i]=$line
                    echo "DEBUG i = $i line = $line filesarr[i] = ${filesarr[i]}"
                    ((i=i+1))
                    # echo ${filesarr[*]}
          done < "$tmpfile2"
          rm "$tmpfile2"
}
function showHelp(){
          echo "-h -- help. Nothing more will be done"
          echo "-d -- sort by datetime of last change"
          echo "-s -- sort by size of file"
          echo "-a -- sort by alphabet order"
          echo "-l -- sort by namelength"
          echo "-u -- undo numbers. Nothing more will be done"
          echo "-p -- path to directory. If there is no path script will executes in current
directory"
          echo "bash script.sh -p 'path' [-d (desc/asc)]"
}
scriptname="$0"
declare -a filesarr
declare -a sortstack
i=0
while getopts ':hrua:d:l:s:p:' opt; do
          case "$opt" in
          a)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort alphabetComparator $OPTARG"
                              ((i=i+1))
                    else
                              exit 1
                    fi
          ;;
          d)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort dateComparator $OPTARG"
                              ((i=i+1))
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          l)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort namelengthComparator $OPTARG"
                              ((i=i+1))
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          s)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort sizeComparator $OPTARG"
                              ((i=i+1))
```

```
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          r)
                    # echo "in random"
                    sortstack[$i]="random"
                    ((i=i+1))
          ;;
          h)
                    showHelp
                    exit 0
          ;;
          u)
                    # echo " in undoNumbers case"
                    getFiles
                    undoNumbers
                    exit 0
          ;;
          p)
                    cd "$OPTARG"
                    if (($? != 0));
                    then
                              echo "Error in path $OPTARG"
                              exit 1
                    fi
          ;;
          :)
                    echo -e "Option requires an argument desc/asc"
                    exit 1
          ;;
          ?)
                    echo -e "There is no such option. Use -h to get help"
                    exit 1
          ;;
  esac
  echo "option"
done
getFiles
undoNumbers
for file in ${filesarr[@]}
do
          echo "file $file"
done
for ((i=i-1; i >= 0; --i))
do
          eval "${sortstack[i]}"
done
for file in ${filesarr[@]}
do
          getdate "$file"
          getsize "$file"
          echo "size=$? date = $retval file $file "
done
for ((i=0; i < ${#filesarr[@]}; ++i))
do
          declare number
          file=${filesarr[i]}
          if ((i <= 9));
          then
                    number=000$i
          elif ((i <= 99))
          then
                    number=00$i
          elif ((i <= 999))
          then
                    number=0$i
          else
                    number=$i
          fi
          mv $file "$number.$file"
done
```