СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Развитие средств вычислительной техники обеспечило возможности для создания и широкого использования систем обработки данных разнообразного назначения.

Бурное развитие глобальных компьютерных сетей и появление новых технологий поиска и обработки информации привлекают все больше внимания к сети Internet со стороны, как частных лиц, так и различных организаций. Многие используют глобальные сети в повседневной работе, объединяя удаленные офисы или даже полностью перемещая свое данные в «облако» (глобальную сеть). Использование сетей в коммерческих целях и при передаче информации, содержащей сведения конфиденциального характера, влечет за собой необходимость построения эффективной системы мониторинга сети, разграничения доступа к глобальной сети, учета трафика. В настоящее время в России глобальные сети применяются для передачи коммерческой информации различного уровня конфиденциальности, например для связи с удаленными офисами из головной штаб квартиры организации, доступа на распределенные ресурсы, файловые сервисы и т.д.

Вряд ли нужно перечислять все преимущества, которые получает современное предприятие, имея доступ к глобальной сети Internet. Но, как и многие другие новые технологии, использование Internet имеет и негативные последствия. Это неограниченный и бесконтрольный доступ сотрудников организации в глобальную сеть. В любом случае трафик контролируется провайдером. Но если одним каналом пользуется несколько человек, а то и весь штат сотрудников, может возникнуть необходимость считать трафик для каждого из них по отдельности.

В корпоративной сети также необходимо следить, какой сотрудник сколько трафика использует. Анализируя трафик, можно также определить, что это за информация, следовательно, и проконтролировать. Что даст возможность сотрудникам внутренней безопасности организации, следить за действиями пользователей локальной сети.

Также учёт трафика часто нужен для диагностики сети. Например, для выявления её «узких» мест. Отслеживая реальную скорость передачи данных между хостами. Неконтролируемый доступ к выделенному каналу может приносить также и косвенные убытки, в связи с перегрузкой корпоративной локальной сети. С этим можно бороться за счёт ограничения сетевого трафика. Это может быть необходимым и в целях контроля расходов или регулирования доступа к какой-то конкретной и зачастую, критической информации.

Информационная инфраструктура современного предприятия представляет собой сложнейший конгломерат разномасштабных и разнородных сетей и систем, поэтому мониторинг сети учет трафика и разграничение доступа в глобальную сеть становится серьезной и сложной задачей.

К сожалению, стандартные средства MS Windows не обладают необходимой функциональностью для решения подобного рода задач. Поэтому, выходом из такой ситуации является разработка специализированного программного обеспечения.

Создание современной программной системы – весьма трудоемкая задача: обычный размер ПО превышает сотни тысяч операторов. Для эффективного создания подобных программных продуктов специалист должен иметь представление о методах анализа, проектирования, реализации и тестирования программных систем; ориентироваться в существующих подходах и технологиях.

Создание программного продукта для работы разграничения доступа в сети учета трафика - это и есть главная задача дипломной работы.

Пояснительная записка состоит из 3 глав.

В главе 1 помимо постановки задачи рассматриваются некоторые теоретические аспекты и особенности данной темы.

В главе 2 описаны применяемые программные сервисы и среды, с помощью которых создавался наш программный продукт, приводится обоснование выбора программного продукта, а так же моделирование стенда, необходимого для разработки и отладки нашего программного обеспечения.

Глава 3 рассматривает логический и подробно физический этапы проектирования программного продукта, а так же тестирование и оценку надежности программного продукта.

В заключении подводится итог нашей работы, рассматриваются результаты.

Итого 3 главы, введение, заключение, список использованных источников и полный программный код в приложении.

1 Теоретическая часть

1.1 Понятия контроль и мониторинг локальных сетей

Непрерывный контроль над работой локальной сети, ныне составляющей основу любой корпоративной сети, необходим для поддержания ее в постоянном работоспособном состоянии. Контроль - это главный этап, который должен выполняться при управлении сетью в первую очередь. Ввиду важности этой функции ее часто отделяют от других функций систем управления и реализуют специализированными средствами. Использование автономных средств контроля помогает администратору сети выявить проблемные участки и устройства сети, а их отключение или реконфигурацию он может выполнять в этом случае вручную. Процесс контроля работы сети обычно делят на два этапа - мониторинг и анализ.

На этапе мониторинга выполняется более простая процедура - процедура сбора первичных данных о работе сети: статистики о количестве циркулирующих в сети пакетов, объема данных, количестве сетевых адресов и так далее.

Далее выполняется этап анализа, под которым понимается более сложный и интеллектуальный процесс осмысления собранной на этапе мониторинга информации, сопоставления ее с данными, полученными ранее, и выработки предположений о возможных причинах замедленной или ненадежной работы сети.

Задачи мониторинга решаются программными и аппаратными средствами, сетевыми анализаторами, встроенными средствами мониторинга коммуникационных устройств, а также специализированными программными продуктами.

Сетевые мониторы собирают данные о статистических показателях трафика, например средней интенсивности общего трафика сети, средней интенсивности потока пакетов с определенным типом.

1.2 Проблемы и методы их решения

1.2.1 Распределение прав доступа к внутренним и внешним ресурсам и сервисам сети

Внутренняя безопасность сети обеспечивается использованием контроллера домена. У каждого сотрудника есть своя учетная запись для входа в систему и свой пароль, который необходимо менять раз, в несколько месяцев. У всех сотрудников четко разграниченные права, например, сотрудники, технической поддержки не смогут получить доступ к базам данных бухгалтера, соответственно бухгалтер, не сможет воспользоваться информацией, с которой работает служба технической поддержки.

Но также, стоит задача разграничение доступа к глобальной сети пользователей корпоративной сети. Сделать это можно на шлюзе. Это благоприятно скажется на работе сотрудников и обезопасит локальную сеть от вредного контента и вредоносных приложений. Так же по усмотрению руководства компании, можно ограничить доступ к некоторым сетевым сервисам, которые могут использоваться вне рабочих целей. Ограничение происходит на главном маршрутизаторе, используемом в локальной сети, а именно для встроенного брандмауэра пишется правило доступа из внутренней сети, к внешней, на определённые порты, и доступ к ним прекращается. Однако, для более гибкого и быстрого изменения настроек ограничения, без использования ресурсов аппаратных маршрутизаторов и будет реализовано программное обеспечение «Мониторинга сети и учета трафика». Для того чтобы однозначно определять конкретного пользователя сети, будет осуществляться проверка, так называемых, MAC адресов. Это аппаратный адрес сетевого интерфейса (сетевой карты). Если IP адрес пользователь может изменить вручную, то MAC адрес – это уникальный идентификатор конкретного сетевого узла, следовательно, и пользователя сети. И не пуская пакеты с определенным MAC адресом дальше в сеть, можно ограничить доступ конкретного пользователя в Internet или удаленный сетевой ресурс.

Тут, первостепенной проблемой является получение входящего трафика, для дальнейшего анализа. Решить ее можно «встав» на сетевой интерфейс и прослушивая его (прослушивая весь трафик, который на него поступает). Существует определенное множество программных библиотек, которые дают возможность считывать сетевой трафик. Выбор этих библиотек будет рассмотрен в следующем разделе.

Второстепенной задачей является разбор входящих пакетов, для изъятия из них IP и MAC адресов, размера пакетов, а также иной информации. Выбор средств для решения этой проблемы, будет описываться в следующем разделе.

1.2.2 Мониторинг активных устройств

Мониторинг активного оборудования это одна из основных задач сетевого монитора. В реализуемой программе необходимо будет получить данные об активных сетевых интерфейсах, которые установлены на локальном ПК, так как именно с них будет проводиться захват и анализ пакетов, а также их дальнейшая отправка.

Составление перечня активного сетевого оборудования можно получить, анализируя MAC адреса поступающих пакетов. Для этого понадобится специализированные инструменты для возможности прослушивания сетевых интерфейсов, а также для «парсинга» (обработки) поступающих пакетов.

1.2.3 Отправка полученных пакетов

Пакеты, которые поступают на принимающий интерфейс, будем его называть клиентским интерфейсом, после их анализа, подлежат дальнейшей отправки через второй интерфейс, будем его называть внешним. Данную задачу можно решить написанием, так называемого, программного маршрутизатора. Но с некоторыми ограниченными возможностями.

Маршрутизатор (роутер) от английского «router» - специализированный сетевой компьютер, имеющий два или более сетевых интерфейсов и пересылающий

пакеты данных между различными сегментами сети. Маршрутизатор может связывать разнородные сети различных архитектур. Для принятия решений о пересылке пакетов используется информация о топологии сети и определённые правила, заданные администратором. Более простыми словами – это сетевое устройство, которое умеет передавать данные между различными сетями, например сетью интернет провайдера и корпоративной локальной сетью.

Маршрутизаторы работают на более высоком «сетевом» (третьем) уровне сетевой модели OSI, нежели коммутатор (или сетевой мост) и концентратор (хаб), которые работают соответственно на втором и первом уровнях модели OSI.

Существует ряд технологий в различных языках программирования, на которых можно реализовать подобного рода роутер. Но конкретная реализация будет зависеть от выбранных средств программирования.

1.2.4 Хранение данных о сетевых устройствах

Для функционирования программного обеспечения и выполнение им надлежащих функций, необходимо хранить информацию о сетевых устройствах. Например, при закрытии программы, необходимо сохранять список разрешенных MAC адресов, прикрепленных к ним IP адресов, а также их объем трафика.

Вариантов для решения этой проблемы два. Первый – это использование полноценной сервер БД. Например, Microsoft SQL Server. Для этого, на локальной машине, где будет функционировать разрабатываемое приложение, необходимо установить и настроить сервер БД. Создать БД, таблицы и в проекте программы реализовать логику работы с этой БД.

Второй вариант – это использование специальных пользовательских файлов, с определенной структурой. Которые располагаются в той же директории, что и исполнительный модуль программы. Простота этого варианта в том, что для работы с ними, нет потребности в установке сервера БД. Работа с такими файлами осуществляется из программы напрямую.

В связи с тем, что необходимо будет хранить только несколько свойств сетевых устройств, потребность в использовании какого-либо сервера БД, отпадает. Но при расширении функций ПО в будущем, использование сервера, все же, может понадобиться. И перейти от файловой структуры хранения данных к полноценной БД, не составит труда.

1.3 Требования к разрабатываемому программному обеспечению

Требования к программному обеспечению - совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации.

Список общих требований к будущему программному продукту:

– программа должна работать в операционных системах семейства Windows, версии Seven или более поздней версии;

– должна иметь дружественный, интуитивный пользовательский интерфейс;

– потреблять минимум системных ресурсов для своего функционирования.

Требования к функционалу программного продукта:

– возможность выбора «клиентского» и «внешнего» сетевых устройств, из общего списка сетевых устройств, установленных на ПК;

– возможность ручного ввода физического адреса «внешнего» сетевого устройства;

– определение и отображение на экране адресов всех сетевых устройств, которые вещают в сети;

– возможность добавления и удаления разрешенных сетевых адресов (устройств);

– учет трафика по каждому сетевому устройству (по каждому MAC адресу);

– возможность запускать и останавливать работу программы;

– в режиме реального времени выводить результаты работы (сканирования и обработки трафика).

Таким образом, данный программный продукт должен работать с неограниченным количеством пользователей (клиентов), трафик от которых, поступает на второй интерфейс локального ПК, перенаправляя трафик на первый интерфейс, за которым может следовать роутер, подключенный к сети Интернет или другая локальная сеть. Общая примерная схема структуры сети для работы данного программного обеспечения, выглядит следующим образом, как показано на рисунке 1.1.

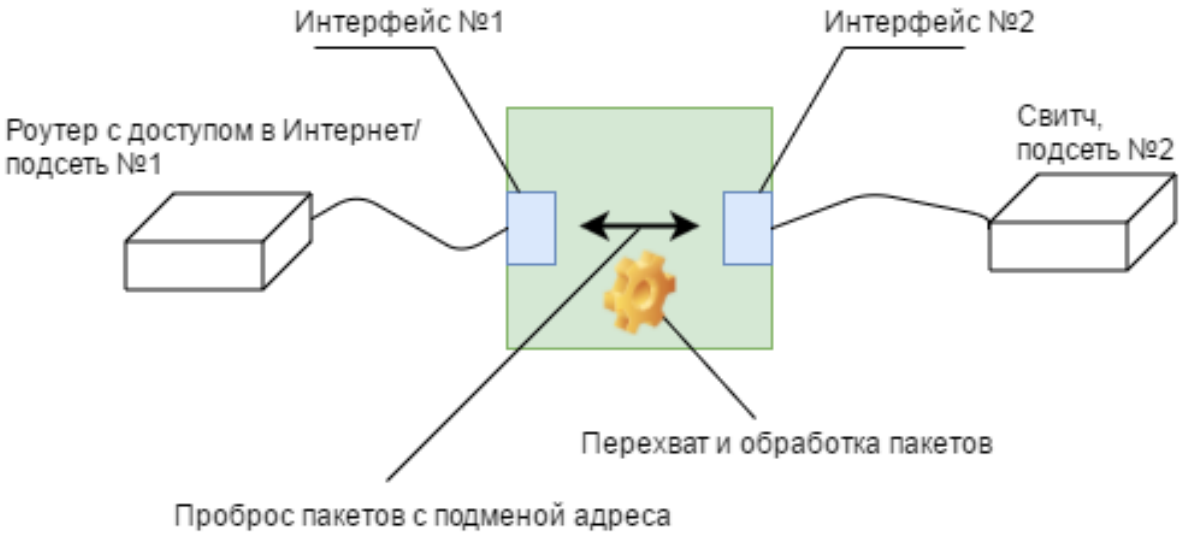

Рисунок 1.1 - Общая схема структуры сети для функционирования ПО

2 Выбор средств и инструментов реализации

2.1 Выбор языка программирования

Ниже будет рассмотрено несколько языков программирования, которые были в роли кандидатов для написания практической части работы.

На данный момент существует большой выбор языков программирования. Под каждую задачу можно выбрать оптимальный язык, на котором будет проще и удобней реализовывать ту или иную задачу. В качестве кандидатов для выполнения данной задачи были выбраны следующие нижеперечисленные языки программирования.

Microsoft Visual C++. Язык программирования Visual C++ из среды разработки Visual Studio компании Microsoft наиболее мощное средство разработки системных программ. Однако, разработка программ на этом языке - трудоемкий и сложный процесс.

Microsoft Visual Basic. Visual Basic от компании Microsoft представляет собой систему быстрой разработки приложений RAD (Rapid Application Development), похожую на среду Delphi. Скорее всего, это объясняется конкуренцией между компаниями Microsoft и Borland International. Однако, не смотря на схожесть интерфейсов, разница в языках существенна.

Visual Basic - объектно-ориентированный язык программирования, как правило, встроенный во многие приложения Microsoft Office. К достоинствам можно отнести простоту создания не сложных приложений, а также возможность редактирования и создания компонент Microsoft Office.

Delphi. Основанная на Паскале, объектно-ориентированная среда разработки компании Borland и имеющая удобные средства для разработки оконных приложений. Помимо этого есть стандартные компоненты для работы и с базами данных и с веб-сервисами.

C#.NET. Для управляющих конструкций, блоков кода, описания сигнатуры методов и многого другого C# имеет стиль языка C. Присутствует много общего с

Java. Это отсутствие множественного наследования и шаблонов, наличие сборщика мусора. Ориентированность на создание компонент – черта схожая с Делфи.

В основе языка - легкость использования, преобладающая над мощностью и скоростью выполнения. Сборщик мусора, управляя объектными ссылками, автоматически освобождает память. Дает безопасность работы с типами – один из важнейших, по мнению многих, фактор избегания ошибок.

C# объектно-ориентированный язык, как и вся платформа .NET. Более того, это язык, ориентированный на написание компонент. C# создан для программирования в управляемой среде с присутствующим сборщиком мусора, но позволяет писать и неуправляемый (unmanaged) код. Переняв многое от своих предшественников - языков C++, Java, Delphi, Модула и Smalltalk - C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# не поддерживает множественное наследование классов (в отличие от C++).

C# дает возможность реализации графически приложений. Windows. Forms - это набор различных управляемых библиотек, с помощью которых можно выполнить все необходимые для оконного приложения действия, начиная от обмена сообщениями с операционной системой для отслеживания любых событий клиентского окна, заканчивая диалоговыми системами, связью с другими компьютерами по сети и многими другими возможностями.

Windows Forms представляет собой одну из двух технологий, используемую в Visual C# для создания интеллектуальных клиентских приложения на основе Windows, выполняемых в среде .NET Framework. Технология Windows Forms специально создана для быстрой разработки приложений, в которых обширный графический пользовательский интерфейс не является приоритетом. Для создания пользовательского интерфейса используется конструктор Windows Forms.

В языке Java используется технология объектно-ориентированного программирования, которая позволяет сократить общее время разработки и писать повторно используемый код. Java-приложения являются независимыми от платформы. Это достигается путем совмещения в языке свойств компилятора и

интерпретатора. Платформонезависимость байт-кода обеспечивается наличием виртуальных java-машин для всех основных платформ. В комплект поставки Java входят стандартные классы, которые обладают достаточной функциональностью для быстрой разработки приложений, Развитые средства безопасности позволяют использовать Java для разработки приложений, работающих в Интернете.

К примеру, Ext JS предназначен для создания интернет-приложений, которые могут иметь весьма насыщенный интерфейс.

Ext JS помогает организовывать и управлять элементами на веб-странице и проводить более точную и эффективную их настройку. Причем данный фреймворк позволяет не только эффективно работать с элементами управления, но и с некоторыми графическими моментами, например, с диаграммами.

Возможности, предоставляемые данным фреймворком, позволяют применять к веб-приложениям шаблон MVC, который позволяет отделить логику приложения, и его данные от визуальной части.

Единственным недостатком Java является - медленная скорость работы, обусловленная использованием ЛТ - компиляторов - цена кроссплатформенности.

После анализа, сделан вывод, что самый подходящий язык программирования для реализации нашего программного продукта – именно C#. Это обосновано тем, что C# имеет широкий инструментарий для работы с сетью, сетевыми устройствами, работой с сетевым трафиком. К нему написано множество библиотек, которые дают возможность работать с сетевыми устройствами и трафиком, который через них проходит. Кроме этого, язык имеет богатый инструментарий для построения экранных форм.

2.2 Выбор среды разработки

В ходе поиска среды разработки, были выделены следующие:

– Sharp Developer;

– MonoDevelop IDE;

– Microsoft Visual Studio.

Далее приведена информация по каждой из них:

Sharp Developer - свободная среда разработки для C#, Visual Basic .NET, Boo, Iron Python, Iron Ruby, F#, C++. SharpDevelop предоставляет интегрированный отладчик, у него есть собственные библиотеки и он взаимодействует с исполняющей средой .NET через COM Interop.

SharpDevelop хотя и использует файлы проекта в формате MSBuild, он по-прежнему использует компиляторы от .NET Framework 1.0 и 1.1, а также от Mono.

Возможности и особенности:

– написана полностью на C#;

– подсветка синтаксиса для C#, IronPython, HTML, ASP, ASP.NET, VBScript, VB.NET, XML, XAML;

– визуальный редактор для WPF и форм Windows Forms (COM-компоненты не поддерживаются);

– интегрированная поддержка NUnit, MbUnit и NCover;

– интегрированная поддержка анализатора сборок FxCop;

– интегрированный отладчик;

– интегрированный профайлер;

– интегрированная поддержка SVN, Mercurial и Git;

– конвертор кода между языками C#, VB.NET, IronPython и Boo;

– расширяемость внешними инструментами;

– расширяемость на основе механизма Add-Ins.

MonoDevelop IDE - свободная мультиплатформенная среда разработки, предназначенная для создания приложений на языках C#, C, C++, Java, Visual Basic.NET, CIL, Nemerle, Boo. На данный момент среда является частью проекта Mono и является одной из самых лучших IDE для разработки проектов на базе Mono.

Microsoft Visual Studio - линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Интегрированная среда разработки (Integrated Development Environment, IDE) Microsoft Visual Studio .NET является последней по

времени выпуска версией популярной и широко используемой среды разработки профессионального программного обеспечения (ПО) производства компании Microsoft.

Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Объединяя в своем составе все положительные стороны предыдущих версий, данная обеспечивает возможность использования всех преимуществ современной технологии Microsoft .NET. В числе основных достоинств MS VS .NET, по достоинству оцененных сообществом профессиональных программистов, можно отметить следующие моменты:

Повышение производительности труда разработчиков - Среда разработки Visual Studio продолжает традиции корпорации Microsoft в области предоставления эффективных инструментальных средств для разработчиков сложного ПО. Обеспечивая среду разработки для всех языков программирования, дополненную набором окон с интуитивно понятными инструментальными средствами, контекстной справкой и автоматизированными механизмами выполнения разнообразных задач разработки, Visual Studio позволяет в сжатые сроки проводить профессиональную разработку программ различного назначения;

Поддержка нескольких языков программирования - В большинстве профессиональных групп разработчиков, как правило, используется несколько языков программирования - для поддержки такой практики в Visual Studio впервые была обеспечена возможность использования сразу нескольких языков в рамках одной и той же среды. Более того, Visual Studio позволяет программистам многократно использовать уже имеющиеся у них наработки, а также навыки разработчиков, создающих свои программы на разных языках программирования;

Единая модель программирования для всех приложений - При создании приложений ранее разработчикам приходилось использовать различные приемы программирования, которые существенным образом зависели от типа приложения - технологии разработки клиентского программного обеспечения, общедоступных веб-приложений, программного обеспечения для мобильных устройств и бизнес-логики промежуточного уровня значительно различались между собой. Среда разработки Visual Studio решает данную проблему, предоставляя в распоряжение разработчиков единую модель создания приложений всех категорий. Эта интегрированная модель обладает привычным и одновременно интуитивно понятным интерфейсом, позволяя разработчикам использовать свои навыки и знания для эффективного создания широкого спектра приложений,

Таким образом, безусловное превосходство среды разработки Microsoft Visual Studio над другими средами разработки стало фактором выбора именно этого средства для разработки программного обеспечения.

2.3 Описание библиотек для работы с сетевыми устройствами и трафиком

2.3.1 Библиотека Pcap

Библиотека Pcap (Packet Capture) позволяет создавать программы анализа сетевых данных, поступающих на сетевую карту компьютера. Примером программного обеспечения, использующего библиотеку Pcap, служит программа Wireshark. Разнообразные программы мониторинга и тестирования сети, снифферы используют эту библиотеку. Она предназначена для использования совместно с языками C/C++/C#, а для работы с библиотекой на других языках, таких как Java, .NET, используют обёртки. Для Unix-подобных систем это библиотека libpcap, а для Microsoft Windows — WinPcap. Программное обеспечение сетевого мониторинга может использовать libpcap или WinPcap, чтобы захватить пакеты, путешествующие по сети, и (в более новых версиях) для передачи пакетов в сети. Libpcap и WinPcap также поддерживают сохранение захваченных пакетов в файл и чтение файлов, содержащих сохранённые пакеты.

Программы, написанные на основе libpcap или WinPcap, могут захватить сетевой трафик, анализировать его. Файл захваченного трафика сохраняется в формате, понятном для приложений, использующих Pcap.

Другими словами, для того чтобы перехватывать пакеты, сетевая карта должна работать в так называемом режиме promiscuous-mode. Для работы в таком режиме нужна поддержка на уровне драйверов, которую и обеспечивает специальная библиотеке WinPcap

Состав WinPcap:

1) фильтр пакетов на уровне ядра

2) низкоуровневая DLL (packet.dll);

3) высокоуровневая системно-независимая библиотека (wpcap.dll).

Единственный недостаток данной библиотеки в том, что она работает не со всеми нестандартными адаптерами (Wi-Fi-карточками, VPN и т. д.).

2.3.2 Библиотеки SharpPcap и Packet.Net

SharpPcap - библиотека для .NET, которая позволяет перехватывать пакеты. По сути, это обертка над библиотекой Pcap, которая используется во многих популярных продуктах. Например, сниффер Wireshark, IDS Snort.

С SharpPcap также поставляется замечательная библиотека для парсинга пакетов - Packet.Net.

Packet.Net поддерживает следующий протоколы:

Ethernet, LinuxSLL, Ip (IPv4 and IPv6), Tcp, Udp, ARP, ICMPv4 и ICMPv6, IGMPv2, PPPoE, PTP, Link Layer Discovery Protocol (LLDP), Wake-On-LAN (WOL).

2.4 Вспомогательные программы

2.4.1 Виртуальная машина VMware

VMware Workstation - программное обеспечение виртуализации, предназначенное для компьютеров x86-64 операционных систем Microsoft Windows и Linux. Позволяет пользователю установить одну или более виртуальных машин на один физический компьютер и запускать их параллельно с ним. Каждая виртуальная машина может выполнять свою операционную систему, включая Microsoft Windows, Linux, BSD, и MS-DOS. VMware Workstation разработана и продается компанией VMware, подразделением EMC Corporation.

VMware Workstation поддерживает мосты с сетевым адаптером реального компьютера, а также создание общих папок с виртуальной машиной. Программа может монтировать реальные CD или DVD диски или ISO образы в виртуальные оптические приводы, при этом виртуальная машина будет считать, что приводы настоящие. Виртуальные жесткие диски хранятся в файлах .vmdk.

VMware Workstation в любой момент может сохранить текущее состояние виртуальной машины (снимок). Данные снимки позже могут быть восстановлены, что возвращает виртуальную машину в сохраненное состояние.

VMware Workstation включает в себя возможность объединять несколько виртуальных машин в группу, которую можно включать, выключать, приостанавливать или возобновлять как единый объект, что является полезным для тестирования технологий клиент-сервер.

Виртуальные машины на платформе VMware позволяют пользователям создавать различные комбинации виртуальных систем, работающих по различным принципам сетевого взаимодействия. Основой сети VMware являются следующие компоненты:

– виртуальные коммутаторы (Virtual Switches);

– виртуальные сетевые интерфейсы (Virtual Ethernet Adapters);

– виртуальный мост (Virtual Bridge);

– встроенный DHCP-сервер;

– устройство трансляции сетевых адресов (NAT, Network Address Translation).

Фундаментальным элементом сетевого взаимодействия в VMware Workstation является виртуальный коммутатор. Он обеспечивает сетевое взаимодействие виртуальных машин на манер физического устройства: на виртуальном коммутаторе есть порты, к которым могут быть привязаны виртуальные сетевые интерфейсы виртуальных машин, а также другие компоненты виртуальной инфраструктуры в пределах хоста. Несколько виртуальных машин, подключенных к одному виртуальному коммутатору, принадлежат одной подсети. Виртуальный мост представляет собой механизм, посредством которого происходит привязка физического сетевого адаптера компьютера к виртуальным сетевым интерфейсам. Встроенный DHCP-сервер VMware позволяет виртуальным машинам автоматически получать IP-адрес в своей подсети, а виртуальное NAT-устройство обеспечивает трансляцию сетевых адресов при общении виртуальных машин с внешней сетью.

Продукты VMware Workstation предоставляют пользователям возможность назначить виртуальной машине один из трех базовых типов сетевого взаимодействия для каждого из виртуальных сетевых адаптеров:

– Bridged;

– Host-only;

– NAT.

Bridged Networking - тип сетевого взаимодействия, который позволяет привязать сетевой адаптер виртуальной машины к физическому сетевому интерфейсу компьютера, что дает возможность разделять ресурсы сетевой карты между хостовой и виртуальной системой. Виртуальная машина с таким типом сетевого взаимодействия будет вести себя по отношению к внешней сети хостовой системы как независимый компьютер. Структура Bridged Networking приведена на рисунке 2.1.

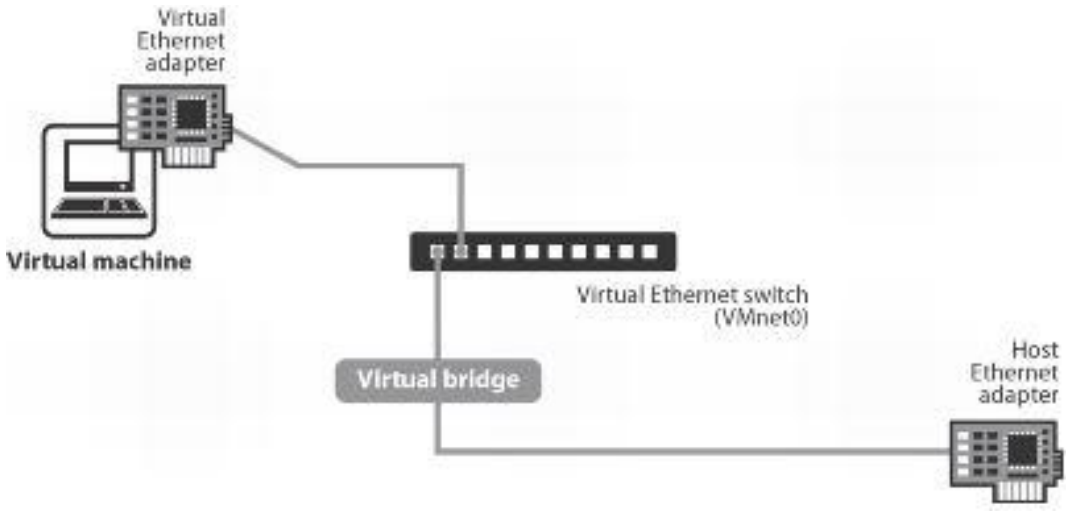

Рисунок 2.1 - Структура сетевого взаимодействия в режиме Bridged Networking

Host-Only Networking - тип сетевого взаимодействия, оптимален для целей тестирования программного обеспечения, когда требуется организовать виртуальную сеть в пределах хоста, а виртуальным машинам не требуется выход во внешнюю сеть. Структура Host-Only Networking приведена на рисунке 2.2.

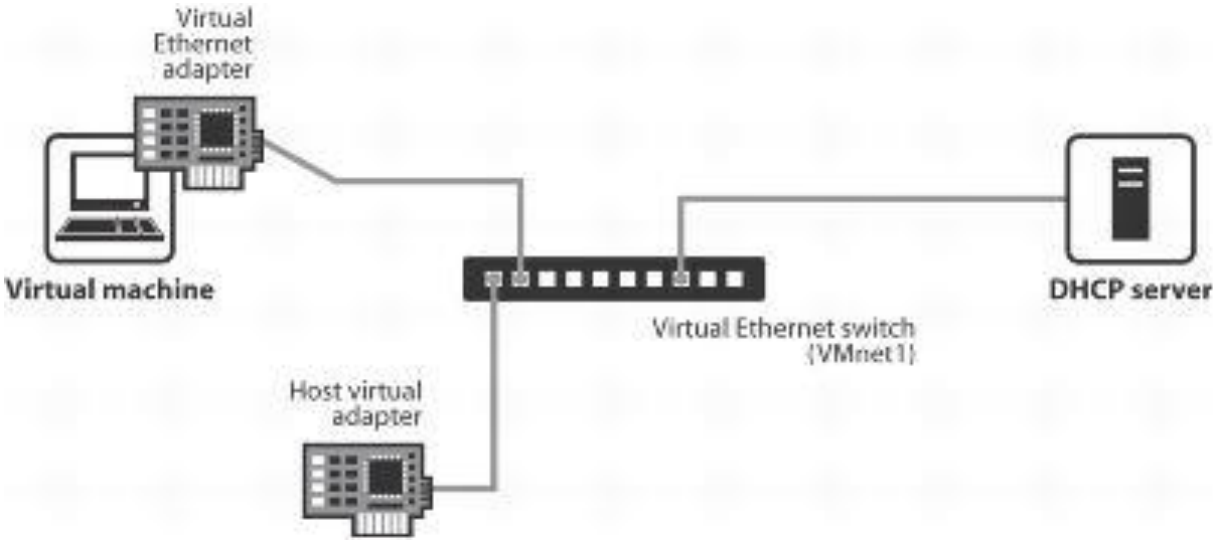

Рисунок 2.2 – Структура Host-Only Networking

NAT Networking. Этот тип сетевого взаимодействия очень похож на Host-Only, за одним исключением: к виртуальному коммутатору VMnet8 подключается устройство трансляции IP-адресов (NAT). К этому коммутатору также подключается DHCP-сервер, раздающий виртуальным машинам адреса из заданного диапазона (по умолчанию 192.168.89.128 - 192.168.89.254) и, непосредственно, сами виртуальные машины. NAT-устройство позволяет осуществлять трансляцию IP-адресов, что позволяет виртуальным машинам инициировать соединения во внешнюю сеть, не предоставляя при этом механизма доступа к виртуальным машинам извне. Структура NAT Networking приведена на рисунке 2.3.

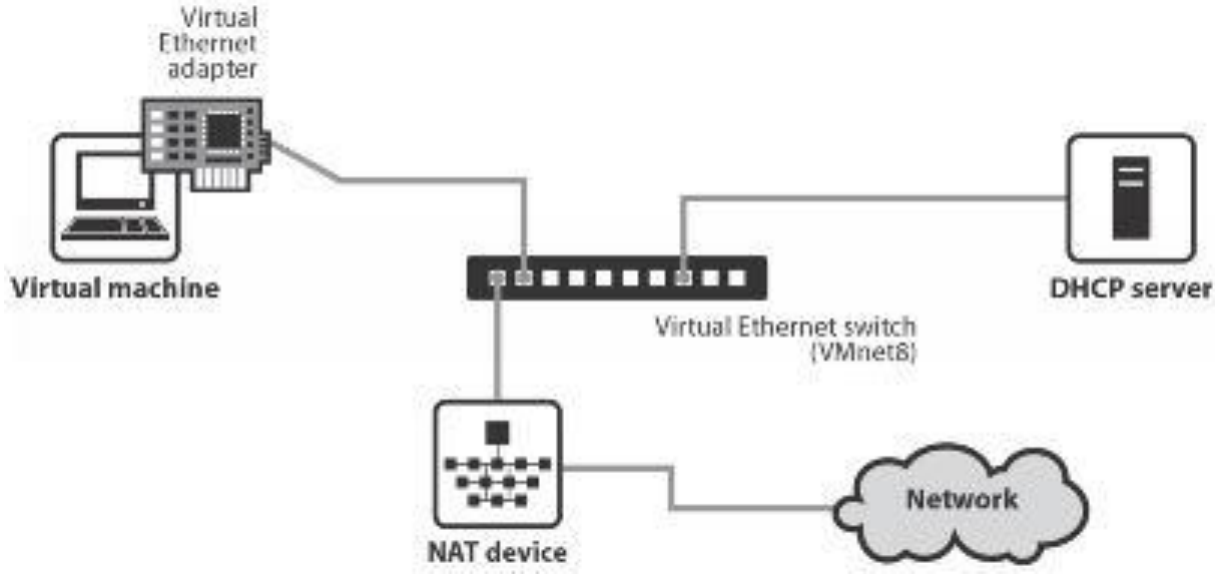

Рисунок 2.3 - Структура NAT Networking

В данном случае, при создании и настройке стенда, будем использовать подключение Host-Only.

2.4.2 Анализатора трафика Wireshark

Wireshark - анализатор компьютерных сетей Ethernet и трафика внутри них. Wireshark предоставляет функционал схожий с приложением tcpdump, однако в отличие от неё имеет графический интерфейс и предоставляет больше возможностей по фильтрации и последующей сортировке данных. Программа просматривает весь проходящий по сети трафик в режиме реального времени. Достигается это переводом сетевых карт в неразборчивый режим (promiscuous mode).

Wireshark может разобрать любой сетевой пакет и отобразить значения всех его полей. Достигается это благодаря знанию структур самых разных сетевых протоколов.

Использование библиотеки pcap, дает возможность захватить данные только из тех сетей, которые поддерживаются этой библиотекой. Но Wireshark умеет работать и со множеством других форматов входных данных. Это дает возможность расширить возможности захвата, открытием файлов данных, захваченных другими программами.

2.5 Создание и настройка стенда

Для отладки и тестирования разрабатываемого программного продукта, необходимо было создать и настроить стенд. И как уже стало ясно из предыдущего подраздела, для создания локальной сети из трех персональных компьютеров, использован продукт VMware workstation.

В роли двух клиентских машин, принято решения использовать машины на базе операционных систем Windows XP. Центральная машина, на которой будет

разрабатываться программное обеспечение, получила ОС Windows 7. Кроме этого, на нее были установлены:

– Visual Studio 2012;

– анализатор пакетов Wireshark, с библиотекой WinPcap;

– библиотеки SharpPcap.dll и Pcap.NET.dll.

Первой клиентской машине был назначен IP:192.168.115.10, шлюз (Gateway):192.168.115.20, MAC адрес: 00:0C:29:62:89:AB. Второй машине назначены IP:192.168.116.10, шлюз:192.168.116.20, MAC адрес: 00:0C:29:B3:66:CA. Основная машина (с двумя интерфейсами) так же получила сетевые адреса.

Для первого сетевого адаптера:

– IP: 192.168.115.10;

– MAC:00:0C:29:17:5E:3C ;

Для второго сетевого адаптера:

– IP: 192.168.116.10;

– MAC: 00:0C:29:17:5E:46.

Подключив первую машину к первому сетевому интерфейсу главной машины, а вторую машину, ко второму интерфейсу, получили стенд с минимальным количеством средств, необходимых для разработки и тестирования разрабатываемого программного обеспечения. Схема подключения представлена на рисунке 2.4.

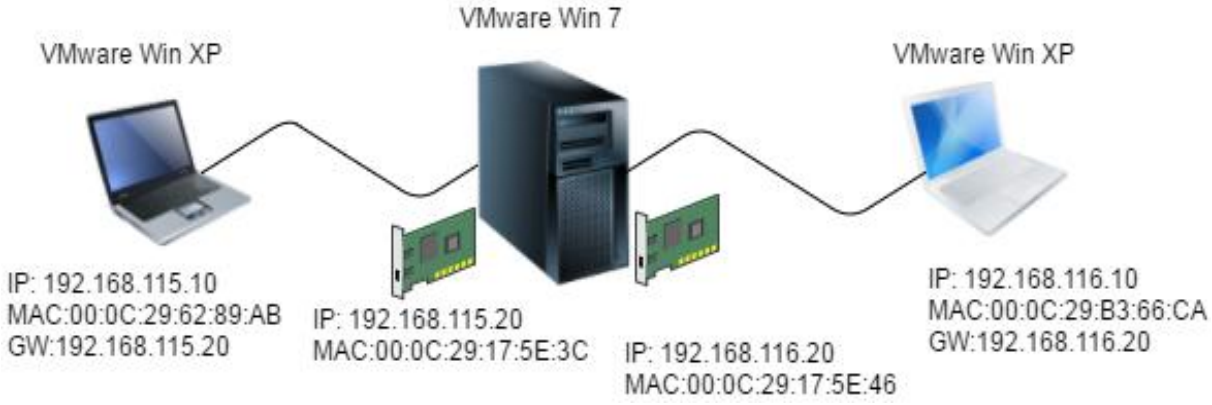

Рисунок 2.4 - Схема стенда для разработки ПО

После подключения, связь между машинами была проверена командой Ping. На этом моменте подготовка к реализации программного обеспечения выполнена.

3 Разработка программного обеспечения

На этапе проектирования определяется подробные спецификации разрабатываемого ПО.

Процесс проектирование ПО обычно включает:

- проектирование общей структуры – определение основных частей (компонентов) и их взаимосвязей по управлению и данным;
- декомпозицию компонентов и построение структурных иерархий в соответствии с рекомендациями блочно-иерархического подхода;
- проектирование компонентов.

Результатом проектирования является детальная модель разрабатываемого ПО вместе со спецификациями его компонентов всех уровней. Тип модели зависит от выбранного или заданного подхода (структурный, объектно-ориентированный или компонентный) и конкретной технологии проектирования. Однако в любом случае процесс проектирования охватывает как проектирование обрабатывающих программ (подпрограмм) и определение взаимосвязей между ними, так и проектирование данных, с которыми взаимодействуют эти программы или подпрограммы.

3.1 Реализация программного интерфейса

Первым делом, в среде разработки был создан проект WinForms - проект с графическим интерфейсом. Консольный вариант программы не подходит для выполнения поставленной задачи, по причине отсутствия инструментов для управления программным обеспечением.

Структурно, программный будет состоять из трех главных составляющих:

– интерфейсная часть;

– логическая часть;

– вспомогательные библиотеки.

1) Интерфейс программы будет реализован в виде графического окна, на котором размещены элементы управления программы, а также поля для ввода и вывода информации.

2) Логическая часть программы - это программный код, который отвечает за логику работы программы. Другими словами - это все составные функции программы, которые отвечают за ее функционал и работу.

3) Вспомогательные библиотеки, необходимы для того чтобы не изобретать велосипед, для доступа к сетевым интерфейсам и для перехвата трафика (пакетов), использованы вспомогательные библиотеки (Pcap.dll, SharpPcap.dll, Pcap.NET.dll). Для этого во вкладке "References" добавлены ссылки на вышеуказанные библиотеки. Для того чтобы добавить ссылки, необходимо нажать правой кнопкой мыши на пункт " References ", и в меню на пункт "Add Reference...", как показано на рисунке 3.1.

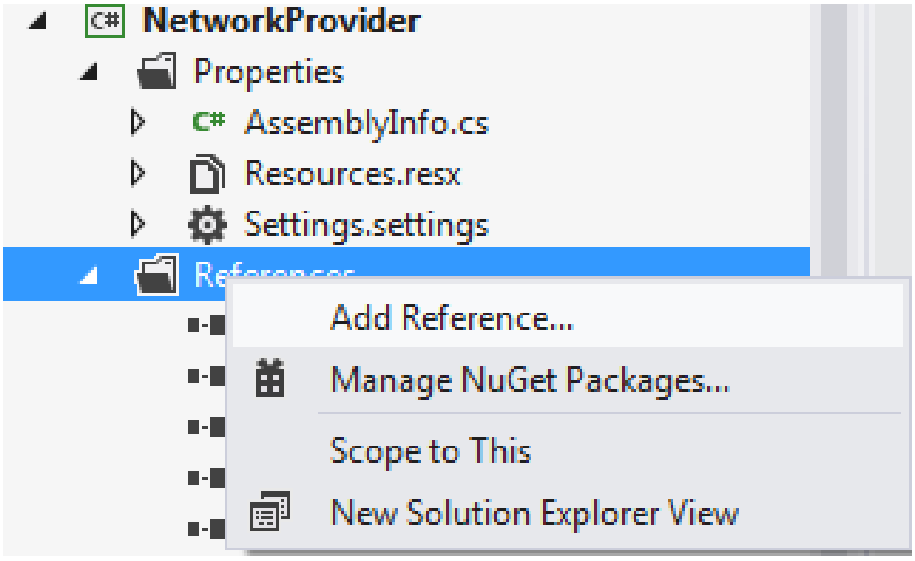

Рисунок 3.1 - Демонстрация шагов, для добавления ссылок

После чего, на экране появится менеджер поиска и выбора ссылок как показано на рисунке 3.2.

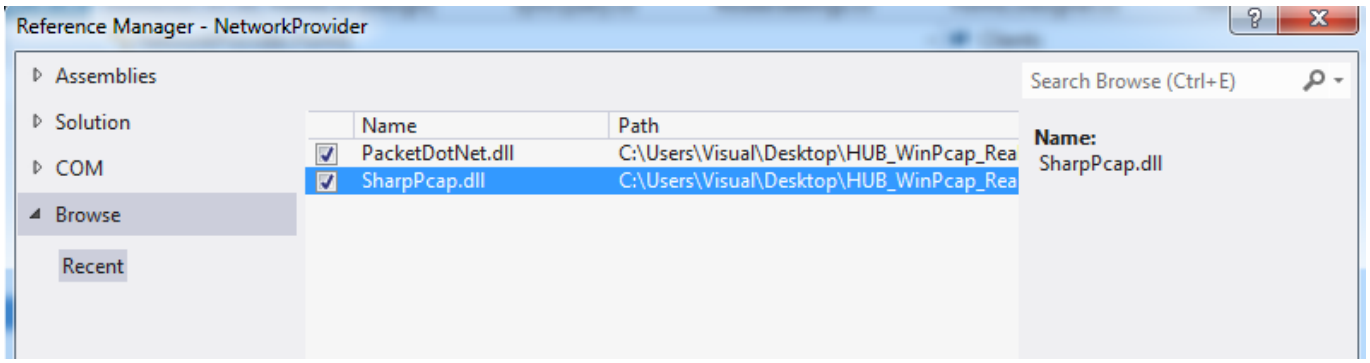

Рисунок 3.2 - Выбор и добавление ссылок

На форму были добавлены следующие компоненты (порядок в списке соответствует порядку добавления на форму):

– cbFirstDev - компонент comboBox, для выбора сетевого подключения со стороны клиентов;

– cbLastDev - компонент comboBox, для выбора сетевого подключения со стороны второй подсети (роутера с Интернет);

– tbDest - компонент textBox, для ввода MAC адреса роутера подключенного к Интернет или другой подсети;

– dgOther - компонент dataGridView, для вывода списка всех сетевых устройств, от которых идет вещание в сети (через второй интерфейс основной машины);

– dgClients - компонент dataGridView, для вывода списка всех разрешенных сетевых устройств(клиентов). Срока состоит из: MAC адреса, IP адреса, объема трафика (в байтах);

– btRefresh - компонент button, для обновления списков dgOther и dgClients;

– AddClient - компонент button, для добавления выбранного клиента из списка dgOther;

– button2 - компонент button, для удаления выбранного клиента из списка dgClients;

– btstart - компонент button, для запуска процесса сканирования сети, перехвата пакетов и других процессов;

– btstop - компонент button, для остановки всех процессов.

Так же, для наглядности и описания всех компонент были добавлены компоненты label. Для периодического обновления списков dgOther и dgClients в проект добавлен компонент timer. После добавления всех необходимых компонент и их обозначении с помощью компонентов label, форма принимает вид как показано на рисунке 3.3.

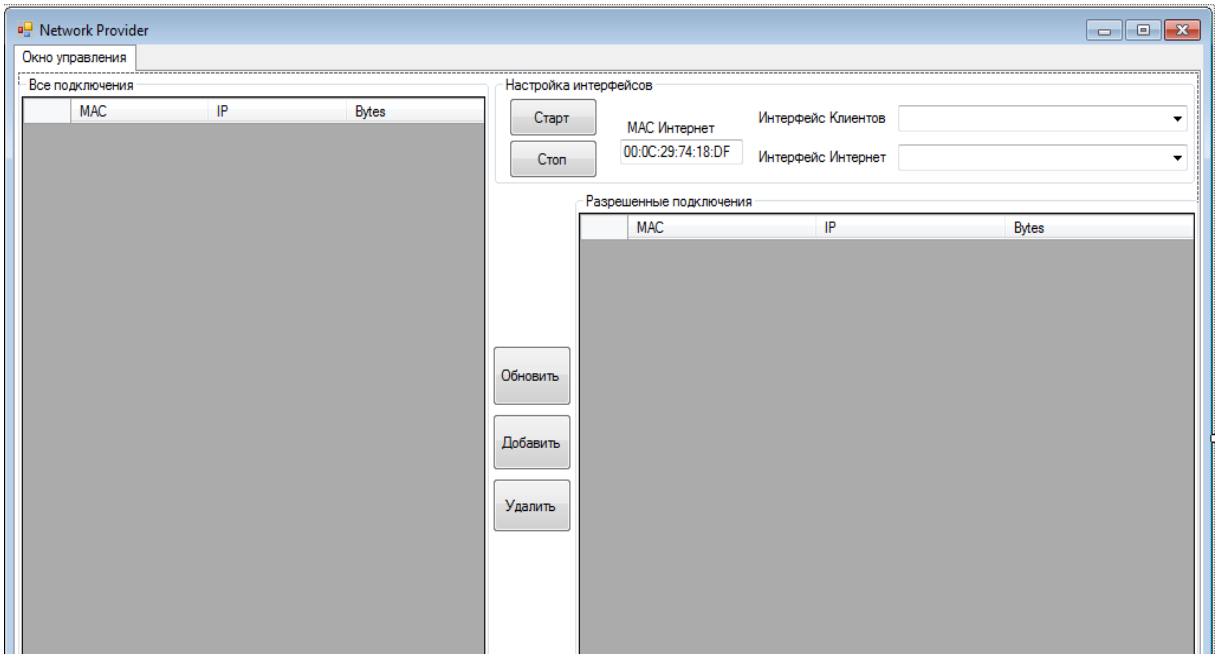

Рисунок 3.3 - Интерфейс программы, после добавления компонентов в форму

Интерфейс программы реализован, следующий этап - реализация логической части программного продукта (реализация классов, функций, логики работы программы).

3.2 Реализация классов, функций и логики работы программы

Для того чтобы иметь доступ к текущим сетевым подключениям, устройствам, иметь возможность перехватывать сетевой трафик и обрабатывать (вносить изменения) пакеты, командой "using", подключены библиотеки "SharpPcap.dll" и "Pcap.NET.dll". Библиотека "Pcap" устанавливается с пакетом "WinPcap", поэтому ее подключение не требуется. Также, была подключена библиотека System.Net.NetworkInformation, показанная на рисунке 3.2.

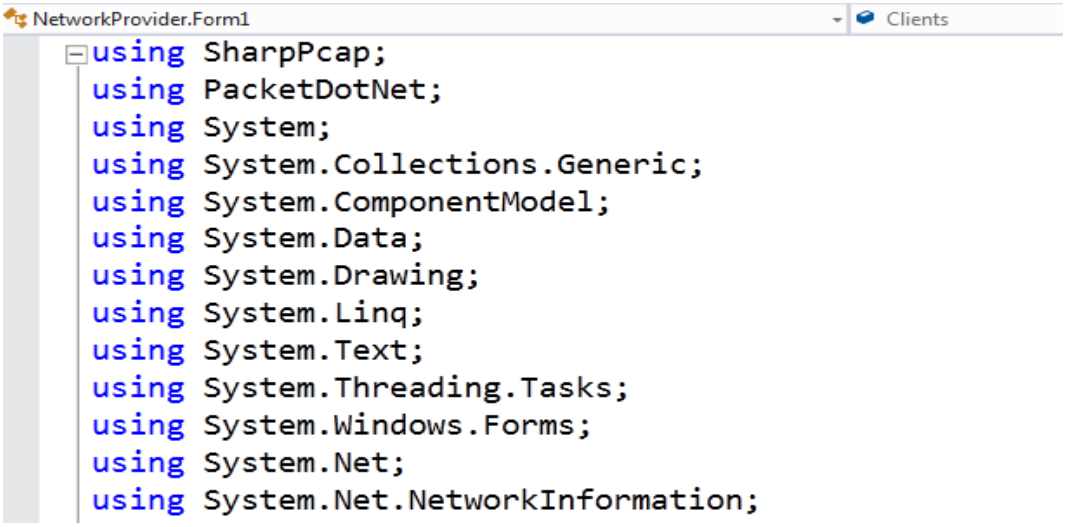

Рисунок 3.2 - Подключаемые библиотеки

Для хранения данных о разрешенных сетевых интерфейсах (клиентах) для проброски пакетов, используется специальный текстовый файл "Clients.txt". Формат записей файла: [MAC адрес]:[IP адрес]:[объем трафика].

Первой, была реализована функция обработчик события загрузки формы (функция, которая отрабатывается при первой загрузке формы (запуске приложения)):

```
private void Form1_Load(object sender, EventArgs e)
        {
            if (!File.Exists("Clients.txt")) {
                using (File.Create("Clients.txt")) ; }
            LoadInterface();
            btrefresh.Enabled = false;
            btstop.Enabled = false;
            LoadClients();
            ViewClient();
        }
```

В самом начале функции идет проверка на существование файла "Clients.txt". Если такой файл отсутствует, то он создается в директории, где находится бинарный (exe) файл. Если такой файл существует, выполняется функция LoadInterface(), затем функция LoadClients() и в итоге, функция ViewClient(). Ниже будут расписаны эти три функции.

```
private void LoadInterface()
        {
            var deviceList = CaptureDeviceList.Instance;
            foreach (var item in deviceList)
```

```
            {
                cbFirstDev.Items.Add(item.ToString().Split('\n')[1]);
                cbLastDev.Items.Add(item.ToString().Split('\n')[1]);
            }
            cbFirstDev.Text = "Выберите устройство";
            cbLastDev.Text = "Выберите устройство";
        }
```

С помощью метода CaptureDeviceList.Instance, получаем список сетевых подключений (устройств). Записываем списки устройств в компоненты cbFirstDev и cbLastDev. Теперь, в компонентах находятся списки сетевых устройств. Затем, выполняется функция LoadClients().

```
private void LoadClients()
        {
            using (StreamReader read = new StreamReader("Clients.txt"))
            {
                while (!read.EndOfStream)
                {
                    var str = read.ReadLine();
                    PhysicalAddress mac = PhysicalAddress.Parse(str.Split(':')[0]);
                    IPAddress ip = IPAddress.Parse(str.Split(':')[1]);
                    int Bytes = int.Parse(str.Split(':')[2]);
                    Client client = new Client(ip, mac, Bytes);
                    lock (Clients)
                    {
                        Clients.Add(mac, client);
                    }}}}
```

Сначала, с файла "Clients.txt" считываются данные о клиентах и записываются в объект класса " Client". Реализация данного класса приведена ниже:

```
public class Client
    {
        public IPAddress IPAdress { get; private set; }
        public PhysicalAddress MACAddress { get; private set; }
        public int CountBytes { get; set; }
        public Client(IPAddress ip, PhysicalAddress mac, int bytes)
        {
            IPAdress = ip;
            MACAddress = mac;
            CountBytes = bytes;
        }}
```

После того как все записи из файла "Clients.txt" были считаны и записаны в список Clients, вызывается функция вывода списка клиентов в форму, в компонент dgClients:

```
private void ViewClient()
        {
```

```
            lock (Clients)
            {
                dgClients.SuspendLayout();
                dgClients.Rows.Clear();
                foreach (var item in Clients)
                    dgClients.Rows.Add(item.Value.MACAddress,    item.Value.IPAdress,
item.Value.CountBytes);
                dgClients.ResumeLayout();
            }}
```

При выполнении данной функции, содержимое списка Clients, записывается в компонент dgClients.

Далее рассмотрим функцию обработчик события нажатия на кнопку "Старт":

```
private void button3_Click(object sender, EventArgs e)
        {ICaptureDevice first = null;
            ICaptureDevice sec = null;
            var deviceList = CaptureDeviceList.Instance;
            foreach (var item in deviceList)
            {if ((item.ToString().Split('\n')[1]) == cbFirstDev.Text)
                    first = item;
                if ((item.ToString().Split('\n')[1]) == cbLastDev.Text)
                    sec = item;}
            if (first == null || sec == null)
            {MessageBox.Show("Выберите устройсто");
                return;}
            IPAddress ipfirst = null;
            IPAddress ipsec = null;
            PhysicalAddress ipfirstmac = null;
            PhysicalAddress ipsecmac = null;
            bool fined = false;
            int index1 = 0;
            int index2 = 0;
            int index3 = 0;
            int index4 = 0;
            btrefresh.Enabled = true;
            while (index1 < 20)
            {try
                {ipfirst                                                          =
IPAddress.Parse(first.ToString().Split('\n')[index1].Split(':')[1].Replace(" ", ""));
                    break;}catch
                {index1++;
                    continue;}}
            while (index3 < 20)
            {try{
                    ipfirstmac                                                    =
PhysicalAddress.Parse(first.ToString().Split('\n')[index3].Split(':')[2].Replace(" ",
""));
                    break; }
                catch
                {  index3++;
                    continue;}}
            while (index4 < 20)
            {try{ipsecmac                                                         =
PhysicalAddress.Parse(sec.ToString().Split('\n')[index4].Split(':')[2].Replace("   ",
""));
                    break; }
                catch
```

```
                {   index4++;
                    continue;  }}
            while (index2 < 20)
            {try
                {ipsec                                                              =
IPAddress.Parse(sec.ToString().Split('\n')[index2].Split(':')[1].Replace(" ", ""));
                    break;}
                catch
                {index2++;
                    continue;
                } }
            PhysicalAddress   add   =   PhysicalAddress.Parse(tbDest.Text.Replace(":",
"").ToString().ToUpper());
            router = new Router(ipfirst, ipfirstmac, ipsec, ipsecmac, add,this);
            router.Capture += Router_Capture;
            router.Start(first, sec);
            timer1.Tick += Timer1_Tick;
            timer1.Start();
            btstart.Enabled = false;
            btstop.Enabled = true;
        }
```

После запуска, функция, подключившись, к выбранным сетевым интерфейсам. Запускает процесс перехвата и обработки пакетов. По таймеру осуществляется обновление списка клиентов.

```
private void ViewsOther()
        {lock (Other)
            {dgOther.SuspendLayout();
                dgOther.Rows.Clear();
                foreach (var item in Other)
                { lock (Clients)
                    {if (!Clients.ContainsKey(item.Key))
                            dgOther.Rows.Add(item.Value.MACAddress,
item.Value.IPAdress, item.Value.CountBytes);
                    }}
                dgOther.ResumeLayout();
                dgOther.Invoke(new   delegateRefreshGrid((o)   =>   o.Refresh()),   new
object[] { dgOther });} }
```

Функция ViewsOther() осуществляет запись в компонент dgOther списка всех клиентов, от которых поступает трафик за время, которое работает процесс перехвата. Данная функция срабатывает циклически по таймеру. Таймер установлен на 20 секунд.

Далее будет описана функция AddClient_Click():

```
private void AddClient_Click(object sender, EventArgs e)
        {var Select = dgOther.SelectedRows;
            PhysicalAddress                              val                          =
PhysicalAddress.Parse(Select[0].Cells[0].Value.ToString().ToUpper());
```

```
            lock (Clients)
            {if (!Clients.ContainsKey(val))
                {Client client = Other[val];
                    Clients.Add(val, client);
                    using (StreamWriter wr = new StreamWriter("Clients.txt"))
                    {wr.WriteLine(client.MACAddress + ":" + client.IPAdress + ":" +
client.CountBytes);
                        wr.Flush();}}
                ViewClient();}
            ViewsOther();}
```

Данная функция (обработчик события нажатия на кнопку "Добавить"), осуществляет, копирование выбранной строки из компонента dgOther, в файл "Clients.txt", а оттуда в компонент dgClients.

Функция button4_Click():

```
private void button4_Click(object sender, EventArgs e)
        {ViewsOther();
            ViewClient();
            RefreshClients();}
```

Данная функция, при нажатии кнопки button4, осуществляет обновление списков dgClients, dgOther и списка Clients, путем запуска функции RefreshClients():

```
private void RefreshClients()
        {lock (Clients)
            {File.Delete("Clients.txt");
                using (StreamWriter wr = new StreamWriter("Clients.txt"))
                {foreach (var item in Clients.Values)
                    {  PhysicalAddress mac = item.MACAddress;
                        IPAddress ip = item.IPAdress;
                        int Bytes = item.CountBytes;
                        wr.WriteLine(mac + ":" + ip + ":" + Bytes);}}}}
```

Обновляет список клиентов в файле "Clients.txt".

При закрытии формы, выполняется функция this.Dispose(). Для того, чтобы завершить (прекратить) процесс перехвата, необходимо нажать на кнопку "Стоп". Для нее реализована функция обработки нажатия:

```
private void btstop_Click(object sender, EventArgs e)
        {router.Stop(router);
            btstop.Enabled = false;
            btstart.Enabled = true;}
```

Функция останавливает процесс перехвата.

В программе (в файле SyncQuery.cs) реализованы специализированные стек и очереди, для простоты реализации процесса перехвата и обработки пакетов.

В отдельном файле "Router.cs", реализован класс "class Router". Для которого написан пользовательский конструктор и следующие методы:

– public void Start (ICaptureDevice first, ICaptureDevice second) - принимающий на вход два сетевых устройства, с которых будет считываться и отправляться сетевой трафик. Этот метод осуществляет подмену MAC адресов и перенаправляет пакеты (осуществляет функцию роутера), именно благодаря ему, имеется возможность отправлять пакеты из одной подсети, в другую.

– public void Stop (Router router) - принимает на вход объект типа Router. Осуществляет остановку захвата пакетов.

– void Program_OnPacketArrival10(object sender, CaptureEventArgs e) - метод, обрабатывает события получение нового пакета интерфейсом. При получении нового пакета, обрабатывает его.

– private void ProcessPacket() - осуществляет обработку пакетов уровня IP.

3.3 Диаграмма UML

Унифицированный язык моделирования UML является языком, который выступает основой определения, проектирования, представления, и документирования программных систем, технических систем и иных систем различной направленности. UML включает стандартный набор диаграмм и нотаций различных видов.

UML предоставляет в средства для создания визуальных моделей, которые единообразно понимаются всеми разработчиками, которые вовлечены в проект и выступают как средство коммуникации в рамках проекта.

Унифицированный Язык Моделирования UML:

– не зависит от объектно-ориентированных языков программирования;

– не зависит от используемой методологии разработки проекта;

– может поддерживать любой объектно-ориентированный язык программирования.

UML является открытым и обладает средствами расширения базового ядра. На UML можно содержательно описывать классы, объекты и компоненты в различных предметных областях, часто сильно отличающихся друг от друга.

На диаграмме классов, приведенной на рисунке 3.4 представлены выделенные классы, используемые для реализации ПО.

В ходе работы выделены следующие классы:

– Form1 - класс экранной формы;

– Client - класс сведений о клиентах;

– Router - класс отвечающий за захват и обработку пакетов;

– SyncDelay, SyncQueue - классы работы с очередями.

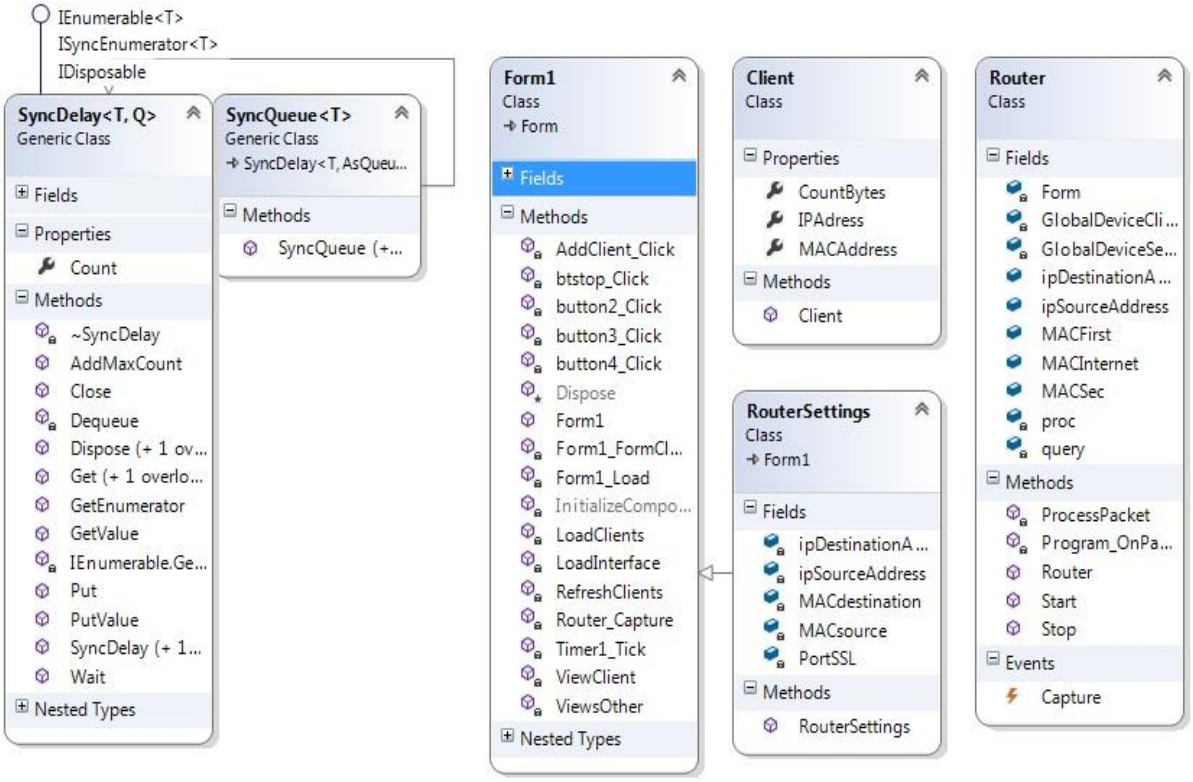

Рисунок 3.4 - Диаграммы классов UML

3.4 Руководство пользователя

Перед тем как запускать программу, необходимо убедиться, что на локальной машине существуют два сетевых устройства,что они подключены и работают

правильно. Необходимо проверить подключение к этим устройствам внешних клиентов. С одной стороны клиентов, инормация о которых будет фильтроваться и для которых будет осуществлена политика разграничения доступа во внешнюю сеть. Внешней сетью может быть как локальная сеть, адрес подсети которой, отличается от первой, как показано на рисунке 3.5 б, так и роутер, имеющий доступ к сети интернет, представленный на рисунке 3.5 а.

Рисунок 3.5 - Схема вариантов структуры сети

Следующим шагом, является проверка связи подсети №1 с соответственным интерфейсом центральной машины, а подсети №2 с сетевым устройством, в которое онаподключена на центральной машине. Это осуществляется командой "ping". После того как все узлы связаны, можно приступать к запуску программы.

Для запуска программы есть определенные рекомендации, а также обязательные условия. К рекомендациям относятся минимальные требования к конфигурации ПК(центрального ПК):

– процессор: Intel Core 2 Duo, AMD Sempron ли выше;

– объем ОП: не менее 256-512 Мб свободной памяти;

– объем жесткого диска: не менее 128 Мб свободного места;

– минимум два сетевых интерфейса, с поддержкой режима прослушивания сети;

– ОС не ниже Windows 7.

Обязательными требованиями есть:

– версия NET.FrameWork не ниже 4.5.2;

– версия WinPcap не ниже 4.1.3;

– наличие библиотек SharpPcap.dll, Pcap.NET.dll в директории с исполняемым модулем программы;

– сетевые экраны на на центральной машине, должны быть отключены в обязательном порядке. Возможен вариант добавления адресов со стороны клиентских машин дух подсете, в список разрешенных, но в таком случае, корректная работа программы - не гарантирована.

Перед запуском программы, директория с исполняемым модулем должна выглядеть следующим образом, как показано на рисунке 3.6.

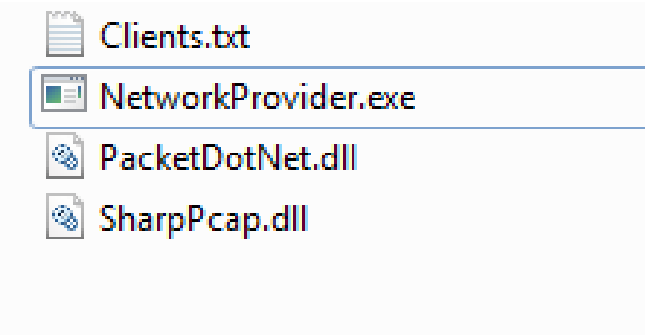

Рисунок 3.6 - Содержимое папки перед запуском программы

Убедившись, что все необходимые библиотеки на месте, можно запускать программу.

После запуска, первым действием является запись MAC адреса устройства на которое будет осуществляться перенаправление трафика (подсеть №1). Это необходимо сделать в поле "3". Затем, в списках "1" и "2", необходимо выбрать соответсвующие сетевые устройства(их названия в списках будут соответсвовать названиям, которые они имеют в ОС). Для запуска процесса мониторинга и

фильтрации, необходимо нажать на кнопку "6". После запуска, ожидаем появления первых адресов сетевых устройств, пакеты от которых проходят через сетевой интерфейс со стороны подсети №2. Информация о таких сетевых устройствах будет появляться в списке "5". А сама запись об устройстве будет иметь вид "8".

Для того, чтобы добавить сетевой интерфейс из общего списка в список разрешенных "4", необходимо его выбрать из "5" и нажать кнопку "10". После чего, сетевое устройство попадает в список "4", это наглядно видно из записи "9". Так же, после того как устройство попадает в список "4", для него осуществляется подсчет трафика.

Для того, чтобы не дожидаться обновления информации в списка по таймеру, можно воспользоваться кнопкой "12", нажав на нее, списки мгновенно обновятся.

Для удаления устройства из списка разрешенных "4", гнеобходимо нажать кнопку "13". И выбранное сетевое устройство снова попадет в список "5". Для остановки процесса мониторинга и фильтрации, необходимо нажать на кнопку "7".

Для повторного запуска, достаточно будет снова нажать на кнопку "6". Если закрыть программу и снова ее запустить, то список разрешенных клиентов загрузиться из файла. Программа, при закрытии, запоминает списки разрешенных клиентов.

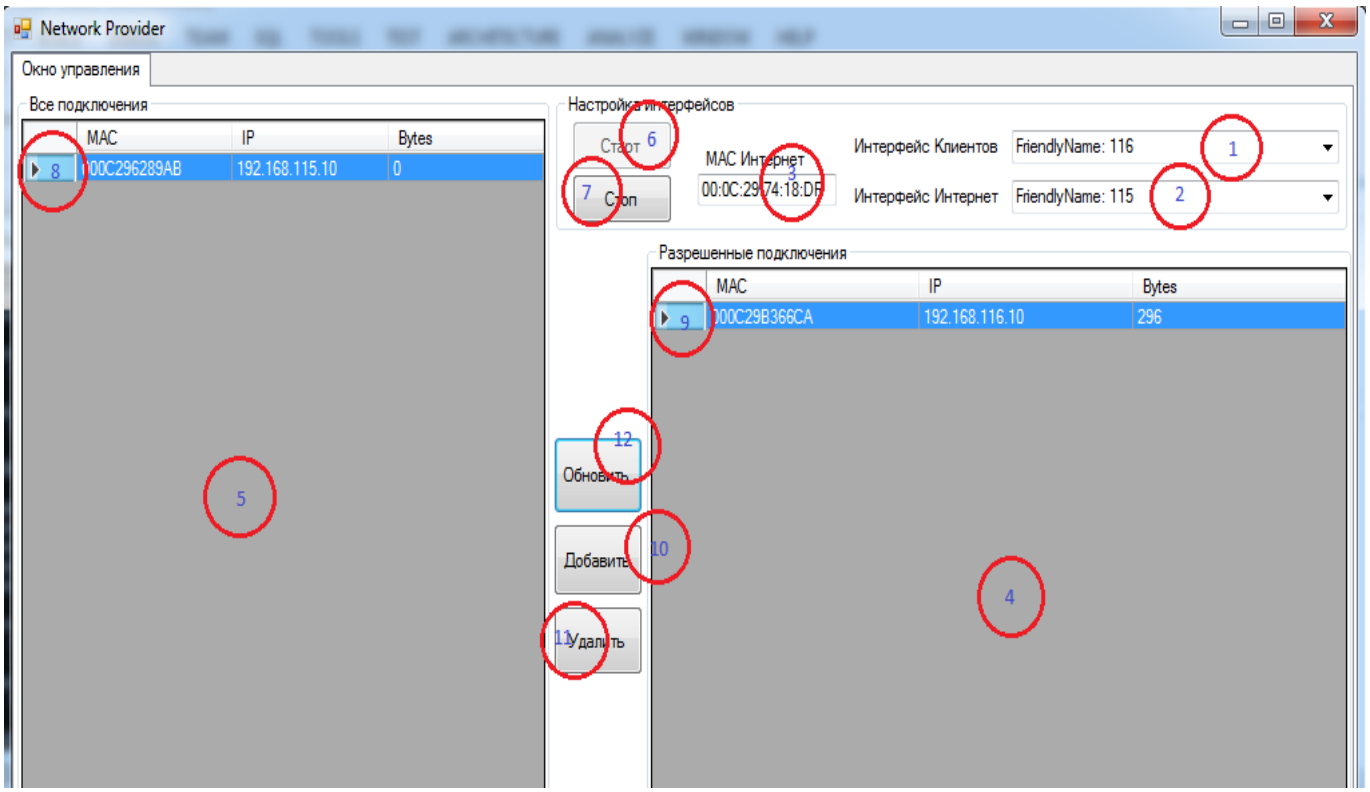

Рисунок 3.7 - Окно программы во время работы

3.5 Тестирование программы

Тестирование – важный этап жизненного цикла программного обеспечения. Тестирование позволяет выявить ошибки в программе для их дальнейшего исправления. Исходные данные для отладки являются результатом тестирования.

Отладка – процесс, позволяющий запустить программу с выполнением кода построчно, используя точки для остановки в необходимых областях тестирования.

Системное тестирование - это тестирование программы в целом. Для небольших проектов это, как правило, ручное тестирование.

Объектом испытаний является наш программный продукт. Перед запуском ПО на центральной машине с ОС Windows 7, проводилась попытка командой "Ping" достучаться из одной подсети, в другую. Для этого, с машины 192.168.116.10, отослали команду "Ping" на узел 192.168.115.10. Но ответа от этого узла не последовало и это верно. Так как данные машины находятся в разных подсетях. После чего, была запущена программа. Введен MAC адрес внешнего узла и выбраны два сетевых интерфейса, клиентский и внешний. Затем, снова послали

команду "Ping". Теперь, это нужно было для того, чтобы наш клиент появился в общем списке сетевых устройств. Как только он появился, выбрав его и нажав на кнопку "Добавить", пакеты "Ping" стали доходить до узла 192.168.115.10, а ответы, поступать обратно к 192.168.116.10.

Тестирование всех программных модулей проводилось на этапе отладки с устранением выявленных ошибок.

Для тестирования пользовательского интерфейса и реализованных функций (логики), был разработан набор возможных штатных и нештатных действий пользователя:

– попытка подключения внешнего сетевого узла с несуществующим MAC адресом;

– попытка запуска программы с невыбранными сетевыми интерфейсами на локальной машине;

– запуск программы без файла клиентов;

– указание неверного формата данных в файле с клиентами;

– проверка работы всех кнопок;

– проверка корректности отображения списков;

– проверка корректности работы таймера;

– проверка корректности перехвата и проброса пакетов;

– проверка корректности подсчета и отображения объема трафика;

– проверка выхода из ПО;

– прохождение полного цикла пользования ПО (от создания пользователя и внесения новых автомобилей в БД, до вывода статистики и результатов работы автомастерской).

В результате тестирования, были устранены найденные ошибки. В результате повторного тестирования дополнительных ошибок обнаружено не было. Но все же, есть вероятность, что на иных ОС системах, сетевых устройствах, возможны погрешности в работе. Это, так же, может зависеть от версий библиотек, типов сетевых устройств и некоторых других факторов.

В целом, программное обеспечение в результате тестирования показало стабильную работу, и готово к эксплуатации.

3.6 Оценка надежности програмного продукта

Одной из основных характеристик качества ПО является надежность.

Надежность - это свойство ПО сохранять работоспособность в течение определенного периода времени, в определенных условиях эксплуатации с учетом последствий для пользователя каждого отказа.

Работоспособным называется такое состояние ПО, при котором оно способно выполнять заданные функции с параметрами, установленными требованиями технического задания. С переходом программного средства в неработоспособное состояние связано событие отказа.

Основным средством оценки и прогнозирования надежности являются модели надежности. Модель надежности программного средства – это математическая модель, построенная для оценки зависимости надежности ПС от заранее известных или оцененных в ходе тестирования параметров.

Существует два разных типа моделей надежности:

– эмпирические;

– аналитические.

Эмпирические модели в основном базируются на анализе структурных особенностей программного средства. Эти модели часто не дают конечных результатов показателей надежности, однако их использование на этапе проектирования ПС полезно для прогнозирования требующихся ресурсов тестирования, уточнения плановых сроков завершения проекта и так далее.

Модель сложности может быть описана такими характеристиками, как размер ПС (количество программных модулей), количество и сложность межмодульных интерфейсов.

При проведении тестирования известна структура программы, имитирующей действия основной, но не известен конкретный путь, который будет выполняться

при вводе определенного тестового входа. Кроме того, выбор очередного тестового набора из множества тест-входов случаен, т.е. в процессе тестирования не обосновывается выбор очередного тестового входа. Эти условия вполне соответствуют реальным условиям тестирования больших программ.

Полученные данные анализируются, проводится расчет показателей надежности, и считается, что реальное ПС, выполняющее аналогичные функции, с подобными характеристиками и в реальных условиях должно вести себя аналогичным или похожим образом.

Аналитические модели надежности дают возможность исследовать закономерности проявления ошибок в программах, а также прогнозировать надежность при разработке и эксплуатации. Модели надежности программ строятся на предположении, что проявление ошибки является случайным событием и поэтому имеет вероятностный характер. Такие модели предназначены для оценки показателей надежности программ и программных комплексов в процессе тестирования: числа ошибок, оставшихся не выявленными; времени, необходимого для выявления очередной ошибки в процессе эксплуатации программы; времени, необходимого для выявления всех ошибок с заданной вероятностью и т. д. Модели дают возможность принять обоснованное решение о времени прекращения отладочных работ.

В настоящее время основными типами применяемых моделей надежности программ являются модели, основанные на предположении о дискретном изменении характеристик надежности программ в моменты устранения ошибок, и модели, основанные на экспоненциальном характере изменения числа ошибок в зависимости от времени тестирования и функционирования программы.

Исходные данные для модели Шумана собираются в процессе тестирования ПО в течение фиксированных или случайных временных интервалов. Модель Шумана предполагает проведение тестирования в несколько этапов. Каждый, из которых проводится на полном комплексе разработанных тестовых данных. При этом зарегистрированные ошибки не исправляются. Количественные показатели надежности могут быть рассчитаны на основе полученных результатов. И уже

только после этого исправляются ошибки, обнаруженные на предыдущем этапе. Корректируются тестовые наборы, и затем проводится новый этап тестирования. При использовании модели Шумана предполагается, что исходное количество ошибок в программе постоянно, но, по мере того, как ошибки выявляются и исправляются, в процессе тестирования может уменьшаться.

Таблица 3.1 - Исходные данные

Номер прогона	1	2	3A	4	5	6	7	8B	9
Количество ошибок	0	5	2	2	1	0	2	1	1
Время прогона (мин)	2	3	3	4	7	6	7	2	1

В программе имеется $I_t = 1953$ оператора.

За точку А возьмём 3-ый прогон, за точку В – 8-ый прогон.

$\tau_{\text{A}} = 8$ – Суммарное время прогона для точки А.

$\tau_B = 26$– Суммарное время прогона для точки В.

$\varepsilon\ (\tau_{\text{A}}) = 7$ – Суммарное число ошибок для точки А.

$\varepsilon\ (\tau_B) = 6$ – Суммарное число ошибок для точки В.

Предполагается, что до начала тестирования в ПС имеется E_t ошибок. В течение времени тестирования τ обнаруживается ε_c ошибок в расчете на команду в машинном языке.

Таким образом, удельное число ошибок на одну машинную команду, оставшихся в системе после t времени тестирования, равно:

$$\varepsilon_{\text{c}}(\tau_{\text{A}}) = \frac{7}{1953} = 0{,}00358 \quad (3.1)$$

$$\varepsilon_{\text{c}}(\tau_{\text{B}}) = \frac{6}{1953} = 0{,}00307 \quad (3.2)$$

Тогда, если время работы ПС без отказа отсчитывается от точки t = 0, а τ остается фиксированным, функция надежности, или вероятность безотказной работы на интервале времени от 0 до t, равна:

$$R(t,\tau) = \exp\{-C[\frac{E_T}{I_T} - \varepsilon_c(\tau)]t\} \quad (3.3)$$

Среднее время безотказной работы соответственно:

$$t_{cp} = \frac{1}{C[\frac{E_t}{I_T} - \varepsilon_c(\tau)]} \quad (3.4)$$

Из величин, входящих в формулы (3.3) и (3.4), не известны начальное значение ошибок в ПС (E_T) и коэффициент пропорциональности - С.

Предполагая, что интенсивность появления ошибок постоянна и равна λ, можно вычислить ее как число ошибок в единицу времени:

$$\lambda = \frac{\sum_{i=1}^{k} A_i}{\tau} \quad (3.5)$$

где A_i – суммарный прогон;

τ – суммарное время прогона.

$$\lambda_A = \frac{7}{8} = 0{,}88$$

$$\lambda_B = \frac{6}{26} = 0{,}23$$

$$t_{\text{ср}} = \frac{\tau}{\sum_{i=1}^{k} A_i} \quad (3.6)$$

Имея данные для двух различных моментов тестирования τ_A и τ_B, которые выбираются произвольно с учетом требования, чтобы $\varepsilon_c(\tau_B) < \varepsilon_c(\tau_A)$ можно сопоставить уравнения (3.3) и (3.4) при:

$$\frac{1}{\lambda_A} = \frac{1}{C[\frac{E_T}{I_T} - \varepsilon_c(\tau_A)]} \quad (3.7)$$

$$\frac{1}{\lambda_B} = \frac{1}{C[\frac{E_T}{I_T} - \varepsilon_c(\tau_B)]} \quad (3.8)$$

Вычисляя отношения (3.7) и (3.8), получим:

$$E_T = \frac{I_T[\frac{\lambda\tau_B}{\lambda\tau_A}\varepsilon_c(\tau_A) - \varepsilon_c(\tau_B)]}{\left(\frac{\lambda\tau_B}{\lambda\tau_A}\right) - 1} \quad (3.9)$$

$$E_T = \frac{1953 * (\frac{0{,}23}{0{,}88} * 0{,}00358 - 0{,}00307)}{\left(\frac{0{,}23}{0{,}88}\right) - 1} = 3{,}28 \approx 3 \text{ ошибки}$$

Подставив полученную оценку параметров E_T, в выражение (3.9), получим оценку для второго неизвестного параметра:

$$C = \frac{\lambda\tau_A}{[\frac{E_T}{I_T} - \varepsilon_c(\tau_A)]} \quad (3.10)$$

$$C = \frac{0{,}23}{\frac{3{,}28}{1953} - 0{,}00358} = -121{,}05$$

Получив неизвестные Et и C, можно рассчитать надежность программы по формуле (3.3). Возьмем время безотказной работы t=5 мин.

$$\exp\left(121{,}05 * \left(\frac{3{,}28}{1953} - 0{,}00665\right) * 5)\right) \approx 0{,}91$$

Надежность безотказной работы велика и вероятность сбоев и возникновения ошибок минимальна.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного проекта было разработано программное обеспечение, позволяющее осуществлять мониторинг сети, вести учет трафика, проводить политику разграничения доступа сетевых устройств одной подсети, к устройствам другой, или их доступа в сеть Интернет. Программа осуществляет подмену адресов, реализован алгоритм ответа на ARP пакеты. Таким образом, программы выполняет функции роутера.

При разработке проекта были подробно рассмотрены теоретические аспекты проектирования программы, а также проблемы, с которыми необходимо было столкнуться, а также варианты решения этих проблем. Во втором разделе, сделан обзор языков программирования. На основании этого обзора, выбран один из них. После чего выбрана среда разработки, необходимые библиотеки для реализации самых тонких задач (работа с сетевыми устройствами и сетевым трафиком). Для реализации понадобились библиотеки SharpPcap.dll, Pcap.NET.dll и Pcap.dll с пакета WinPcap. Среда разработки приложения стала Visual Studio 2012. В рамках поставленной задачи так же был создан и натроен стенд для тестирования и отладки ПО. Развернут на базе виртульной машины VMware Workstation 9.0.

Главным преимуществом использования данного программного средства является осуществление контроля над сетевым трафиком, который поступает от пользователей конкретной подсети. Ограничением их доступа во внешнюю сеть, а также возможностью проводить учет трафика от каждого сетевого устройства. Поэтому, хоть программа реализовывалась в учебных целях, практически, ее можно применять на реальных сетевых объектах, например, сетевыми администраторами в частных фирмах, организациях, для контроля сетевой деятельности их сотрудников.

В заключении можно сказать, что реализованное программное средство полностью удовлетворяет требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Культин Н.Б. Microsoft Visual C# в задачах и примерах – 2-е издание. БВХ-Петербург,2014. – 320 стр.
2. Шилдт Герберт. C# 4.0 полное руководство – 2011. 427 стр.
3. В.И. Кияев, О.Н. Граничин. Безопасность информационных систем. Электронная книга. НОУ «Интуит»., 2016 – 192 с.
4. Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. - ДМК Пресс, 2012. – 593 с.
5. Вильям Столлингс. Основы защиты сетей. Приложения и стандарты. - Вильямс – 2-е издание, 2012 – 432 с.
6. http://programming-lang.com [Электронный ресурс]
7. http://www.ixbt.com [Электронный ресурс]