

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление

09.03.01 «Информатика и вычислительная техника»

подготовки:

Профиль подготовки:

«Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

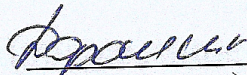
на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ
ПРЕОБРАЗОВАНИЯ СЕТЕВЫХ АДРЕСОВ

Студент

 31.05.22 Колобов К.А.
(подпись, дата)

Руководитель

 Дорошенко И.Н.
(подпись, дата) 31.05.22.

Нормоконтролёр

 Бычков А.С.
(подпись, дата) 31.05.22

Работа допущена к защите (протокол заседания кафедры от

02.06.2022 № 13)

Заведующий кафедрой

(подпись)

М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК

от 15.06.2022 № 9)

Секретарь ГЭК

(подпись)

Е.И. Гурин

Пенза, 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«26» 04 2022 г.

ЗАДАНИЕ

НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Колобов Кирилл Андреевич группы 18BB1 факультета вычислительной техники.
2. Тема работы: «Программное обеспечение для преобразования сетевых адресов».
3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Дорошенко И.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать программное обеспечение для преобразования сетевых адресов, осуществляющее перехват, анализ и модификацию сетевого трафика, проходящего через устройство маршрутизации, а также спроектировать для ПО пользовательский интерфейс.
5. Перечень подлежащих разработке вопросов
Требуется провести анализ предметных областей: изучить основы сетей на основе стека протоколов *TCP/IP*, изучить основы разработки

программного обеспечения на системном уровне, определить архитектуру разрабатываемого программного обеспечения. Также необходимо изучить функционал программных платформ *Windows API*, *.net Framework* и *Windows Forms*.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объём работы, ч	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	75	26.04.2022 — 3.05.2022	<i>Дорошенко И.Н.</i>
2.	Анализ требований	10	4.05.2022 — 4.05.2022	<i>Дорошенко И.Н.</i>
3.	Выбор технологий, применяемых при проектировании и разработке	20	5.05.2022 — 6.05.2022	<i>Дорошенко И.Н.</i>
4.	Описание алгоритмов	45	7.05.2022 — 10.05.2022	<i>Дорошенко И.Н.</i>
5.	Разработка программного обеспечения для преобразования сетевых адресов	150	10.05.2022 — 29.05.2022	<i>Дорошенко И.Н.</i>
6.	Оформление пояснительной записки	24	30.05.2022 — 31.05.2022	<i>Дорошенко И.Н.</i>

Дата выдачи задания «26» апреля 2022 г.

Руководитель бакалаврской работы

Дорошенко И.Н.

Дорошенко И.Н.

(подпись, *26.04.2022*)

Задание к исполнению принял студент

Колобов К.А.

Колобов К.А.

(подпись, дата)

Работу к защите допустить

Декан факультета

Фионова Л.Р.

Фионова Л.Р.

(подпись, дата)

Реферат

Бакалаврская работа содержит 170 листов, 45 рисунков, 3 таблицы, 3 источника и 3 приложения.

ДРАЙВЕР, СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ, ПРЕОБРАЗОВАНИЕ СЕТЕВЫХ АДРЕСОВ, ИНТЕРНЕТ, СЕТЬ.

Цель данной работы – разработать программное обеспечение, осуществляющее перехват, анализ и модификацию сетевого трафика, проходящего через устройство маршрутизации, а также спроектировать для ПО пользовательский интерфейс.

Объектом разработки является программное обеспечение, осуществляющее перехват, анализ и модификацию сетевого трафика, проходящего через устройство маршрутизации. ПО состоит из сетевого драйвера-фильтра *NDIS*, пользовательского интерфейса для управления драйвером, а также библиотеки для обеспечения взаимодействия пользовательского интерфейса с драйвером.

В результате проведённой работы реализован проект, удовлетворяющий поставленной задаче. А именно: создано программное обеспечение, осуществляющее перехват, анализ и модификацию сетевого трафика, проходящего через устройство маршрутизации. Данное программное обеспечение было внедрено в защищенную операционную систему *QР ОС*, разработанную отечественной ИТ компанией ООО НТП «Криптософт».

					ПГУ 1.090301.17.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программное обеспечение для преобразования сетевых адресов Пояснительная записка	Лит.	Лист	Листов
Разраб.		Колобов К.А.					2	170
Провер.		Дорошенко И.Н.						
Н. Контр.		Бычков А.С.						
Утверд.		Митрохин М.А.				ФВТ, каф. ВТ, гр. 18ВВ1		

Содержание

Перечень сокращений и обозначений.....	6
Введение.....	8
1 Программное обеспечение для преобразования сетевых адресов.....	10
1.1 Анализ предметной области	10
1.1.1 Принцип работы NAT	10
1.1.2 Типы ПО для преобразования сетевых адресов	26
1.1.3 Анализ аналогов ПО	27
1.1.4 Постановка задачи	28
1.1.5 Выбор инструментов разработки	29
1.2 Реализация программного обеспечения	32
1.2.1 Описание архитектуры программного обеспечения для преобразования сетевых адресов	32
1.2.2 Разработка программного обеспечения для преобразования сетевых адресов.....	33
1.3 Анализ результата	49
1.3.1 Руководство пользователя	49
1.3.2 Тестирование и отладка	53
2 Безопасность жизнедеятельности.....	72
2.1 Введение в безопасность жизнедеятельности при работе с ЭВМ	72
2.2 Характеристика потенциальных опасностей и вредностей.....	72
2.3 Электробезопасность	73
2.4 Пожарная безопасность	74
2.5 Санитарно-гигиеническая характеристика производства	75
2.6 Расчётно-аналитическая часть.....	76

Заключение	78
Список используемых источников	79
Приложение А – <i>UML</i> -диаграммы	80
Приложение Б – Листинг программы	84
Приложение В - Презентация	165

Перечень сокращений и обозначений

API (Application Programming Interfaces) – интерфейс программирования приложений

ACK (Acknowledgement) – флаг *Acknowledgement*

ARP (Address Resolution Protocol) – протокол определения адреса

DLL (Dynamic Link Library) – библиотека динамической компоновки

GUI (Graphical User Interface) – Графический пользовательский интерфейс

IOCTL (Input/Output Control) – интерфейс, позволяющий обеспечить связь пользовательского приложения с драйвером

IP (Internet Protocol) – межсетевой протокол

IPv4 (Internet Protocol version 4) – межсетевой протокол 4-й версии

IPv6 (Internet Protocol version 6) – межсетевой протокол 6-й версии

NAT (Network Address Translation) – преобразование сетевых адресов

NCP (Network Control Protocol) – протокол управления сетью

NDIS (Network Driver Interface Specification) – спецификация интерфейса сетевого драйвера

OSI (Open Systems Interconnection) – модель взаимодействия открытых систем

PAT (Port Address Translation) – преобразование сетевых портов

SYN (Synchronize Sequence Number) – синхронизирующий номер последовательности

TCP (Transmission Control Protocol) – протокол управления передачей

TCP/IP (Transmission Control Protocol/ Internet Protocol) – стек протоколов *TCP/IP*

TFS (Team Foundation Server) – продукт компании *Microsoft Team Foundation Server*

UDP (User Datagram Protocol) – протокол пользовательских датаграмм

UI (User Interface) – пользовательский интерфейс

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

WDK (*Windows Driver Kit*) – инструментарий для разработки драйверов платформы *Windows*

НТП – научно-техническое предприятие

ООО – общество с ограниченной ответственностью

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

ЭВМ – электронно-вычислительная машина

					ПГУ 1.090301.17.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Интернет (англ. *Internet*) – глобальная система, которая в своё время произвела революцию в области коммуникаций, позволив объединить в одну сеть устройства, находящиеся в разных частях планеты.

Изначально Интернет был разработан для объединения вычислительных комплексов под контролем специализированных учреждений. Но в 1980 – 1990-х годах началось его развитие на массовом рынке в связи с распространением персональных ЭВМ – они же ПК – в быту.

По данным на 2020-й год количество пользователей сети Интернет примерно составляет 4.5 миллиарда человек – больше половины численности населения Земли. Что не удивительно, ведь Интернет предоставляет обширные возможности, позволяющие использовать его практически в любых видах деятельности, взаимодействующих с информацией в том или ином виде. Кроме того, в современных реалиях доступ в Интернет можно получить чуть ли не из любого места на планете в любой момент времени.

При создании Интернета его разработчики и не предполагали, что сеть получит широкое распространение. В итоге столь высокая популярность Интернета привела к явлению, называемому исчерпанием *IPv4* адресов (англ. *IPv4 address exhaustion*).

Для долгосрочного решения проблемы исчерпания *IPv4* адресов была разработана новая версия протокола *IP* – *IPv6*. Но высокая стоимость миграции, отсутствие обратной совместимости и технические сложности при внедрении *IP* 6-й версии сильно замедляют введение данного протокола в широкое применение. В качестве же временного и вместе с тем более оперативного выхода из положения был представлено программное обеспечение *NAT*.

NAT – это программное обеспечение, позволяющее преобразовывать *IP*-адреса транзитных пакетов. Оно позволяет транслировать несколько

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

внутренних *IP*-адресов в один внешний *IP*-адрес, что даёт возможность использовать лишь один публичный *IP*-адрес целой сетью компьютеров. Именно по данной системе построена большая часть сетей в мире. *NAT* используется повсеместно самыми разными устройствами – от маршрутизаторов, используемых интернет-провайдерами для обеспечения *IP*-адресами целых жилых районов, до смартфонов, служащих в роли мобильной точки доступа.

Также дополнительной функцией *NAT* является повышение конфиденциальности локальной сети путём скрывания внутренних адресов её устройств от внешней сети.

Целью данной работы является разработка программного обеспечения *NAT* с типом трансляции *PAT*, так же известной как *NAT Overload*, и интегрирование его в отечественную защищенную операционную систему *QP ОС*.

1 Программное обеспечение для преобразования сетевых адресов

1.1 Анализ предметной области

1.1.1 Принцип работы NAT

1.1.1.1 Общее понятие сети Интернет

В первую очередь, необходимо уточнить принцип работы ПО для преобразования сетевых адресов. Начать стоит с самой базовой информации – устройства сети Интернет.

Сеть Интернет, как следует из названия, представляет собой глобальную сеть. Данная сеть состоит из сетей поменьше, каждая из которых соединена с другой прямо или косвенно.

Сеть сама по себе представляет набор подключенных друг к другу компьютеров, которые имеют возможность передавать данные между собой.

Интернет не имеет какого-то центрального сервера, который предоставляет устройствам доступ к сети. Это значит, что он не зависит от какой-либо определенной машины. Напротив, каждый компьютер, который способен отправлять и принимать данные корректным образом, является частью Интернета.

Функционирование сети Интернет базируется, по сути, на двух аспектах: пакетах данных и сетевых протоколах.

1.1.1.2 Пакеты

Когда сегмент данных отправляется через Интернет, он в первую очередь разбивается на пакеты, которые затем преобразуются в биты. Пакеты передаются устройству назначения, а принимающее устройство затем собирает пакеты в нужном порядке и объединяет их для воссоздания сегмента данных в первоначальном виде. Получившийся сегмент данных далее может быть использован устройством по назначению.

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

Как уже было сказано, в сети пакеты представляют собой маленькие части поделённого сегмента данных. Каждый пакет содержит информацию сегмента данных (англ. *Data*) и заголовок (англ. *Header*). Заголовком называется информация о содержании пакета и располагается, в соответствии с названием, в самом начале пакета. Заголовок нужен для того, чтобы принимающее пакет устройство могло определить, как с данным пакетом нужно взаимодействовать.

Стоит отметить, что каждый сетевой протокол имеет индивидуальную структуру заголовка. Так как *NAT* взаимодействует непосредственно с заголовками пакетов, далее будут представлена информация об основных сетевых протоколах.

1.1.1.3 Сетевые протоколы

Сетевой протокол представляет собой стандартизированный способ передачи данных в сети и решает такую задачу, как обеспечение корректного взаимодействия двух устройств, каждое из которых может иметь отличную от другого конфигурацию как в плане оборудования, так и в плане программного обеспечения.

Существует множество сетевых протоколов, но в рамках данной темы достаточно понимания лишь некоторых из них, входящих в состав модели *TCP/IP*.

1.1.1.4 Модель *TCP/IP*

Интернет построен на основе сетевой модели передачи данных *TCP/IP*. *TCP/IP* – это сетевая модель для передачи данных в цифровом виде, которая была разработана на основе протокола *NCP* и описывает способ передачи данных от источника информации к получателю в сети.

Как может быть понятно из названия, основу модели *TCP/IP* составляют протоколы *TCP* и *IP*. Тем не менее, они не являются единственными протоколами, составляющими данную сетевую модель. В

общем, *TCP/IP* представляет собой уровень абстракции между интернет-приложениями и структурой маршрутизации и коммутации.

Основной принцип работы *TCP/IP* заключается в разделении решения задачи связи на слои или уровни. При отправке пакет данных проходит через все уровни модели в прямом порядке – сверху вниз – для преобразования информации в пакеты данных, а при получении – в обратном – снизу вверх – для преобразования пакетов данных в первоначальный вид.

Далее на рисунке 1 представлена схема модели *TCP/IP*.

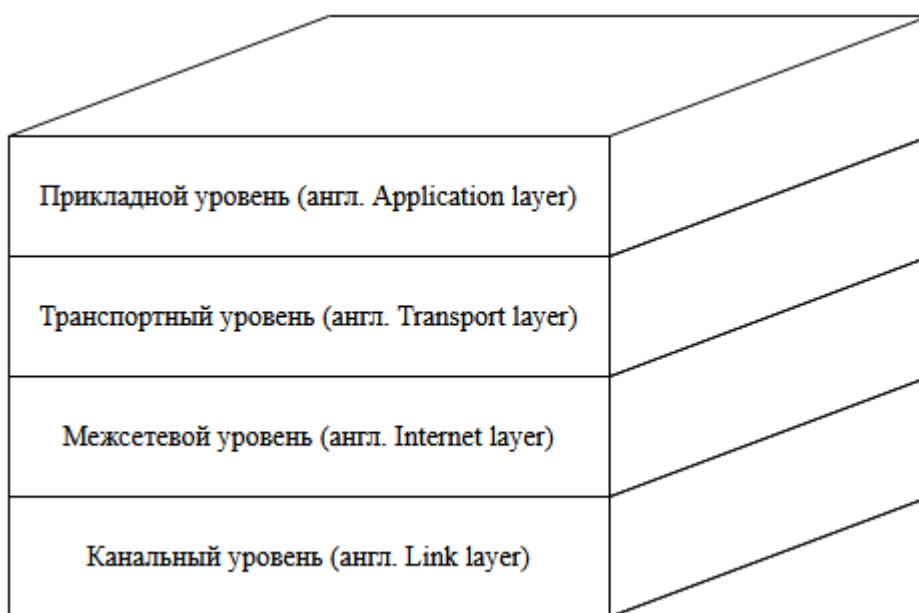


Рисунок 1 - Схема модели *TCP/IP*

На схеме можно увидеть, что модель *TCP/IP* состоит из четырёх слоёв:

- прикладной уровень (англ. *Application layer*) – на прикладном уровне работает большинство сетевых приложений, которые предоставляют какой-либо сервис пользователю. Они взаимодействуют с функционалом нижних уровней по уже установленному соединению. Для таких приложений, как правило, созданы свои протоколы – например, *HTTP*;

- транспортный уровень (англ. *Transport layer*) – протоколы транспортного уровня отвечают за контроль доставки пакетов данных и

правильную последовательность их обработки. На данном уровне функционируют такие протоколы как *TCP* и *UDP*;

- межсетевой уровень (англ. *Internet layer*) – Межсетевой уровень позволяет объединить локальные сети в глобальную, что, по сути, и образует Интернет. Данный уровень регламентирует передачу информации между отдельными сетями, с помощью чего они могут взаимодействовать между собой. Основными протоколами межсетевого уровня являются *IP* и *ICMP*;

- канальный уровень (англ. *Link layer*) – Канальный уровень осуществляет описание, каким образом происходит обмен информацией с точки зрения физических устройств. Примеры протоколов – *Ethernet*, *WLAN*, *IEEE 802.11*.

Протоколы, которые представляют каждый из слоёв модели *TCP/IP*, полностью реализуют функциональные возможности абстрактной модели *OSI*.

Модель *OSI* – это эталонная абстрактная модель, описывающая семь слоёв или уровней, которые компьютерные системы используют для взаимодействия по сети. *OSI* является первой стандартной моделью сетевых коммуникаций, которая была принята в качестве международного стандарта в начале восьмидесятых годов двадцатого века.

Далее на рисунке 2 представлена схема модели *OSI*.

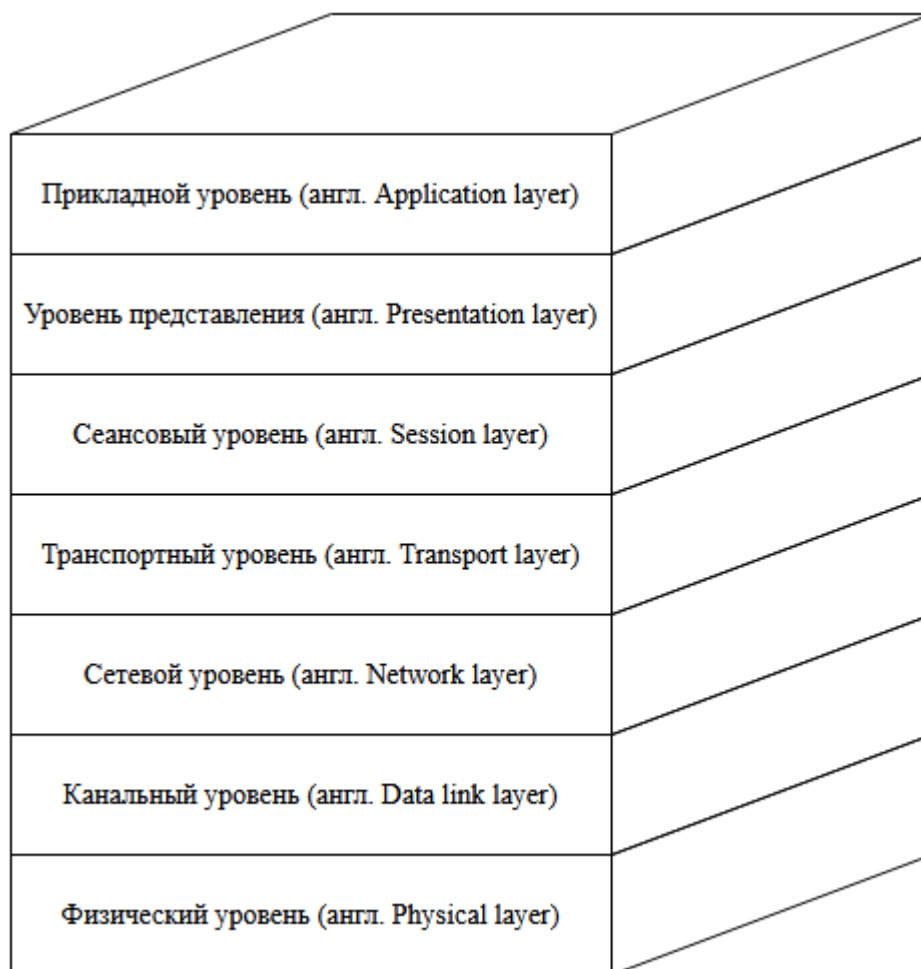


Рисунок 2 - Схема модели *OSI*

Различия между *TCP/IP* и *OSI*:

- в *TCP/IP* сеансовый, прикладной и уровень представления модели *OSI* рассматриваются как единственный прикладной уровень. Аналогичная ситуация и с канальным уровнем *TCP/IP* – в модели *OSI* он разделяется на 2 части – канальный и физический уровень;
- модель *OSI* представляет собой лишь концептуальную модель, тогда как *TCP/IP* является материальной структурой и представляет модель клиент/сервер;
- *TCP/IP* является стандартной моделью работы сети Интернет, тогда как модель *OSI* является эталонной моделью, разработанной для понимания архитектуры системы.

1.1.1.5 Протокол транспортного уровня - *TCP*

Сетевой протокол *TCP* – это один из основных протоколов транспортного уровня в сети Интернет, обеспечивающий надёжность передачи данных. Протокол осуществляет повторный запрос данных в случае их потери и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым целостность передаваемой информации и уведомление отправителя о результатах передачи. Механизм *TCP* предоставляет поток для обмена данными с предварительной установкой соединения, которое происходит путём выполнения так называемого “тройного рукопожатия” (англ. *Three-way handshake*).

“Тройное рукопожатие” – это процесс, используемый протоколом *TCP* для установки соединения между клиентом и сервером. Процесс, как можно понять из названия, состоит из 3-х этапов, каждый из которых требует взаимодействия клиента и сервера путём обмена пакетами с флагами *SYN* и *ACK*, до начала обмена реальными данными.

Флаги *SYN* и *ACK* можно представить как запрос и ответ соответственно. Флагу *SYN* сопутствует порядковый номер, генерирующийся случайным образом, а флагу *ACK* принадлежит порядковый номер предшествующего ему флага *SYN*, значение которого было увеличено на 1 единицу. Например, если флаг *SYN* отправлялся с порядковым номером 1111, флаг *ACK* должен будет иметь порядковый номер $1111 + 1 = 1112$. Стоит уточнить, что порядковые номера флагов используются как механизм синхронизации между устройствами.

Далее механизм тройного рукопожатия рассмотрен пошагово (рисунок 3).

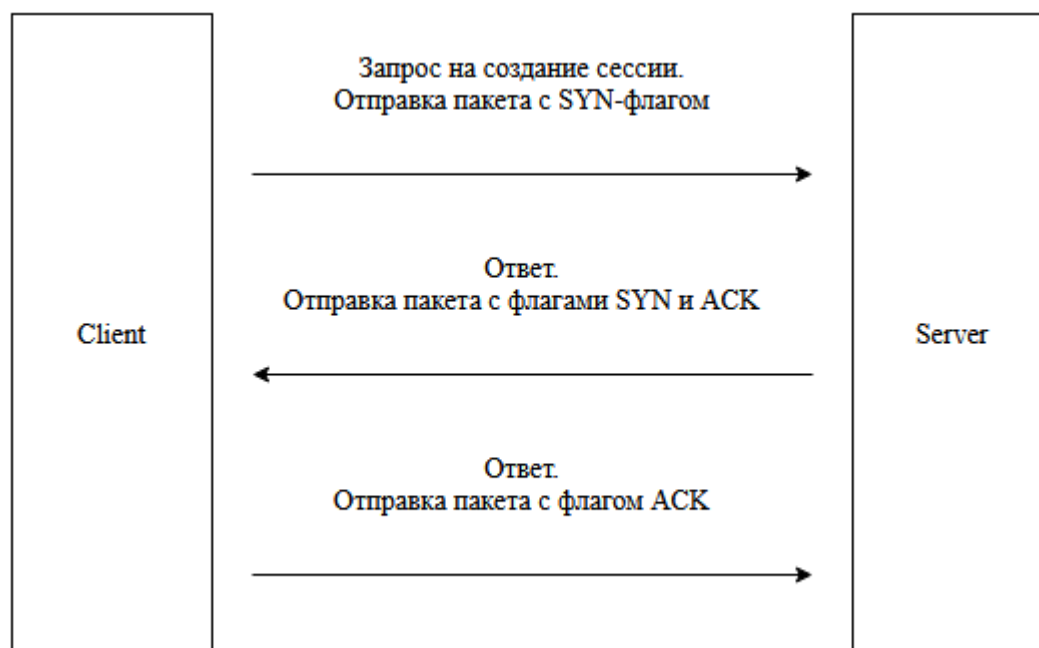


Рисунок 3 - Механизм "тройного рукопожатия" протокола *TCP*

Шаг 1 (флаг *SYN*). На первом этапе клиент устанавливает соединение с сервером. Клиент отправляет пакет, который содержит флаг *SYN* и номер последовательности, серверу. Данный пакет информирует сервер о факте попытки установления связи клиентом, а также сообщает номер последовательности, требуемый для формирования дальнейшего ответа.

Шаг 2 (флаги *SYN + ACK*). Далее сервер отвечает на запрос клиента. Сервер отправляет пакет, содержащий комбинацию флагов *SYN* и *ACK*, а также их порядковые номера.

Шаг 3 (флаг *ACK*). На заключительном этапе установки соединения клиент отправляет пакет, содержащий флаг *ACK*, в ответ на ранее отправленный сервером пакет с флагом *SYN*.

После выполнения данных этапов процесс “тройного рукопожатия” окончен, порядковые номера синхронизированы и соединение считается установленным – это означает, что устройства с данного момента имеют возможность обмениваться информацией. В дополнение, в дальнейшем отправка пакета с флагом *SYN* будет сигнализировать о возникшей ошибке в процессе передачи данных.

На рисунке 4 представлен конкретный пример механизма “тройного рукопожатия”.

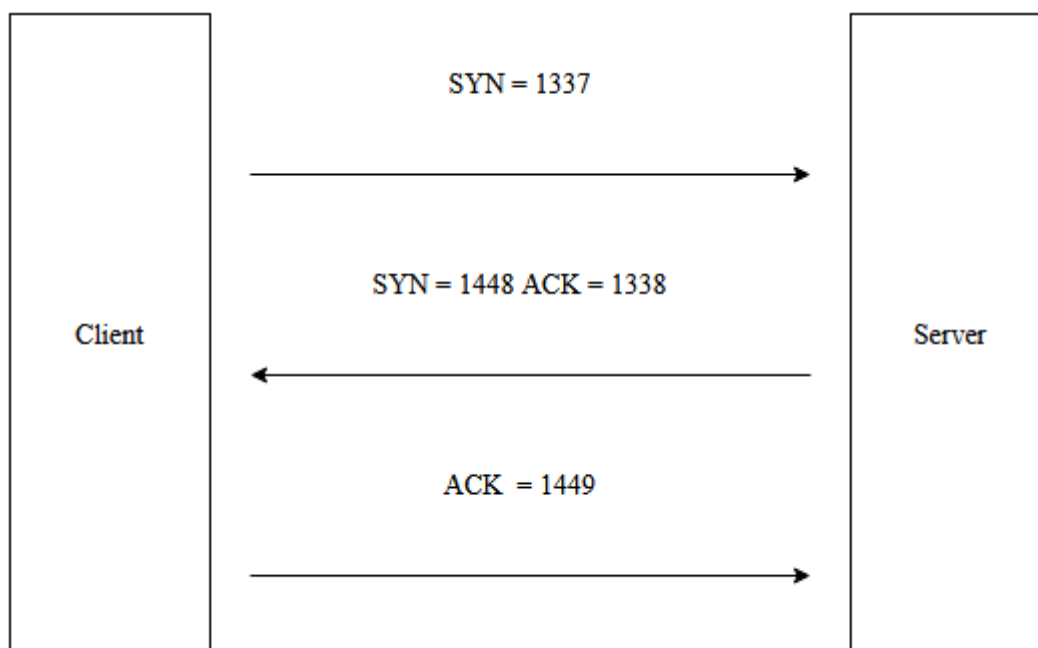


Рисунок 4 - Механизм "тройного рукопожатия" протокола *TCP* на примере

Альтернативой протоколу *TCP* служит *UDP* – он отличается большей оперативностью, жертвуя при этом надёжностью передачи данных.

1.1.1.6 Протокол транспортного уровня – *UDP*

Сетевой протокол *UDP*, как и *TCP*, является одним из ключевых протоколов передачи данных в сети Интернет. Данный протокол позволяет передавать данные по сети без необходимости предварительной установки соединения. Процесс передачи данных при помощи протокола *UDP* не включает в себя никаких “рукопожатий” для синхронизации между устройствами, как это происходит в том же *TCP*. Всё это создаёт преимущество *UDP* перед *TCP* с точки зрения скорости обмена информацией. Тем не менее, данные при передаче могут приходить не по порядку, дублироваться или вообще не прийти. При использовании *UDP* подразумевается, что пакеты либо не должны проверяться на наличие ошибок, либо проверяются уже на прикладном уровне модели *TCP/IP*.

Далее на рисунке 5 представлена схема, демонстрирующая условный вариант обмена данными при использовании протокола *UDP*.

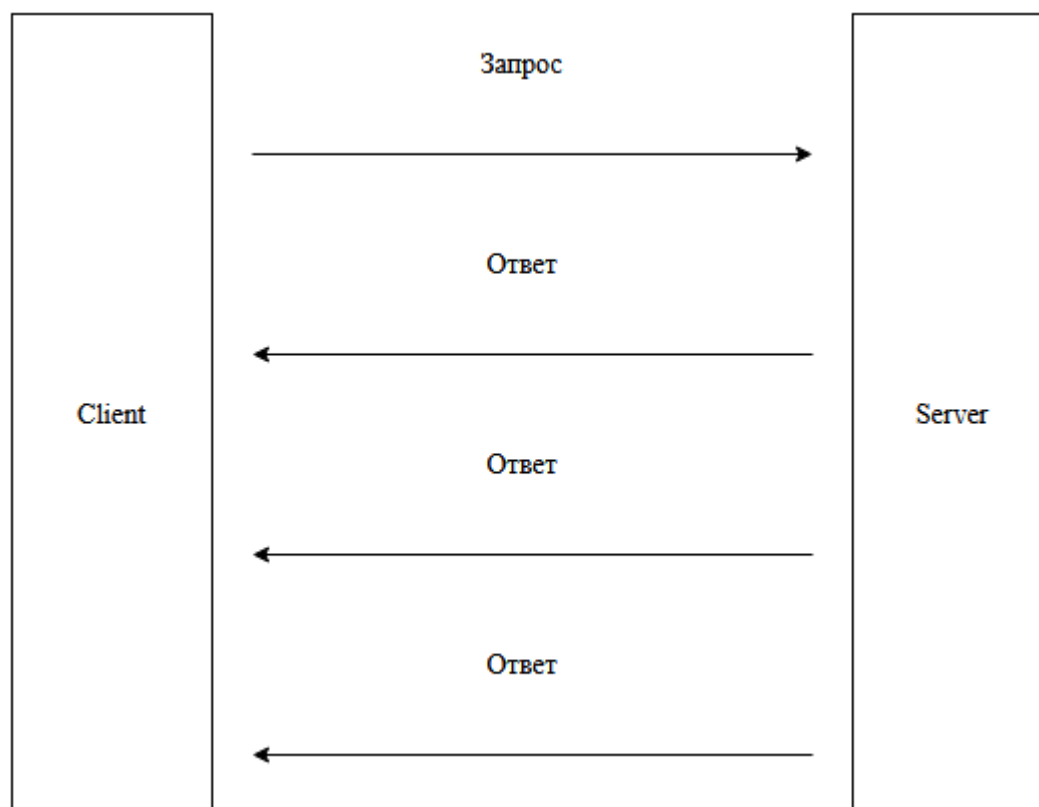


Рисунок 5 – Условный вариант передачи данных при использовании протокола *UDP*

На данном этапе были представлены протоколы транспортного уровня модели *TCP/IP*. Далее будут рассмотрены протоколы сетевого уровня модели *TCP/IP*, требуемые для понимания механизма работы ПО для преобразования сетевых адресов.

1.1.1.7 Протокол сетевого уровня – *ICMP*

Сетевой протокол *ICMP* используется сетевыми устройствами для определения проблем при передаче данных по сети. *ICMP* в основном используется для проверки того, доставляются ли данные по назначению и, если доставляются, в какой срок это происходит.

Далее на рисунке 6 и рисунке 7 представлены схемы, демонстрирующие типичные варианты работы утилиты *ping*, использующей протокол *ICMP*.

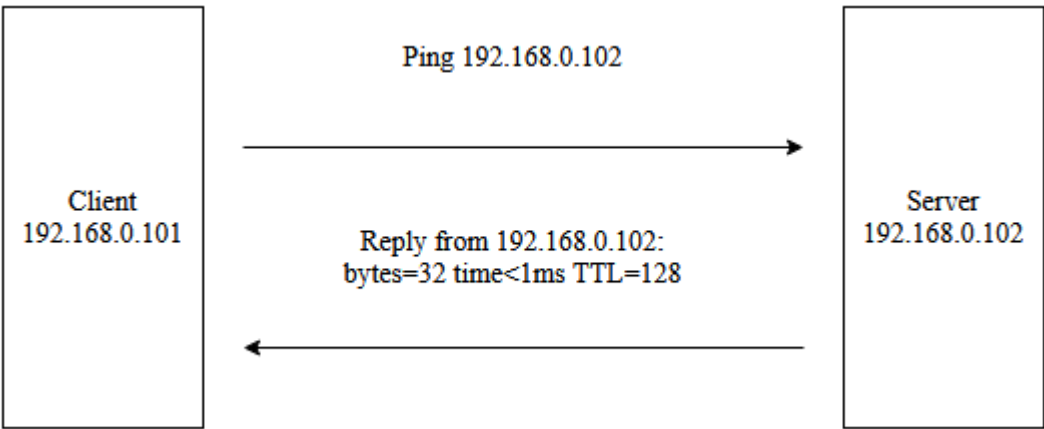


Рисунок 6 - Вариант выполнения команды *ping* при успешном соединении клиента с сервером

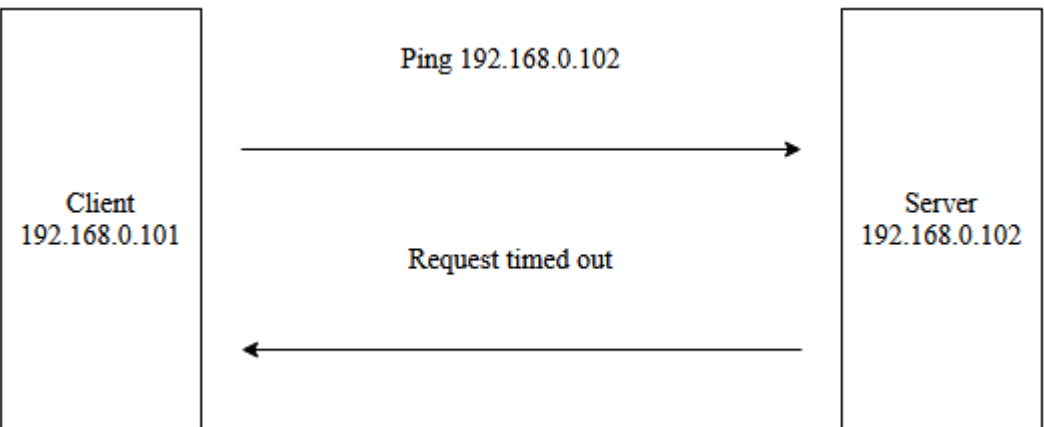


Рисунок 7 - Вариант выполнения команды *ping* при неудачном соединении клиента с сервером

Протокол не используется пользовательским ПО, кроме таких утилит, как *ping* и *tracert*. Так как *ICMP* не передаёт никаких данных, он не попадает в список протоколов транспортного уровня модели *TCP/IP*. Также стоит отметить, что *ICMP* работает только в паре с *IPv4*, так как для взаимодействия с *IPv6* используется *ICMPv6*.

1.1.1.8 Протокол сетевого уровня – ARP

Сетевой протокол *ARP* – протокол, позволяющий определить фиксированный физический адрес устройства – *MAC*-адрес с помощью знакомого *IP*-адреса.

Следует уточнить, что *MAC*-адресом называется уникальный идентификатор, который присваивается устройству как непосредственно физической единице. *MAC*-адрес имеет длину в 6 байт и состоит из 6 пар чисел шестнадцатеричной системы счисления. Данный адрес, в отличие от *IP*-адреса, постоянен и присвоен, как уже было сказано, к конкретной единице оборудования.

Ровно как и *ICMP*, протокол *ARP* существует только в паре с *IPv4*, так как в семействе *IPv6* его функции выполняет *ICMPv6*.

Далее на рисунке 8 представлена упрощённая схема обмена *ARP* пакетами для определения *MAC*-адреса устройства в сети.

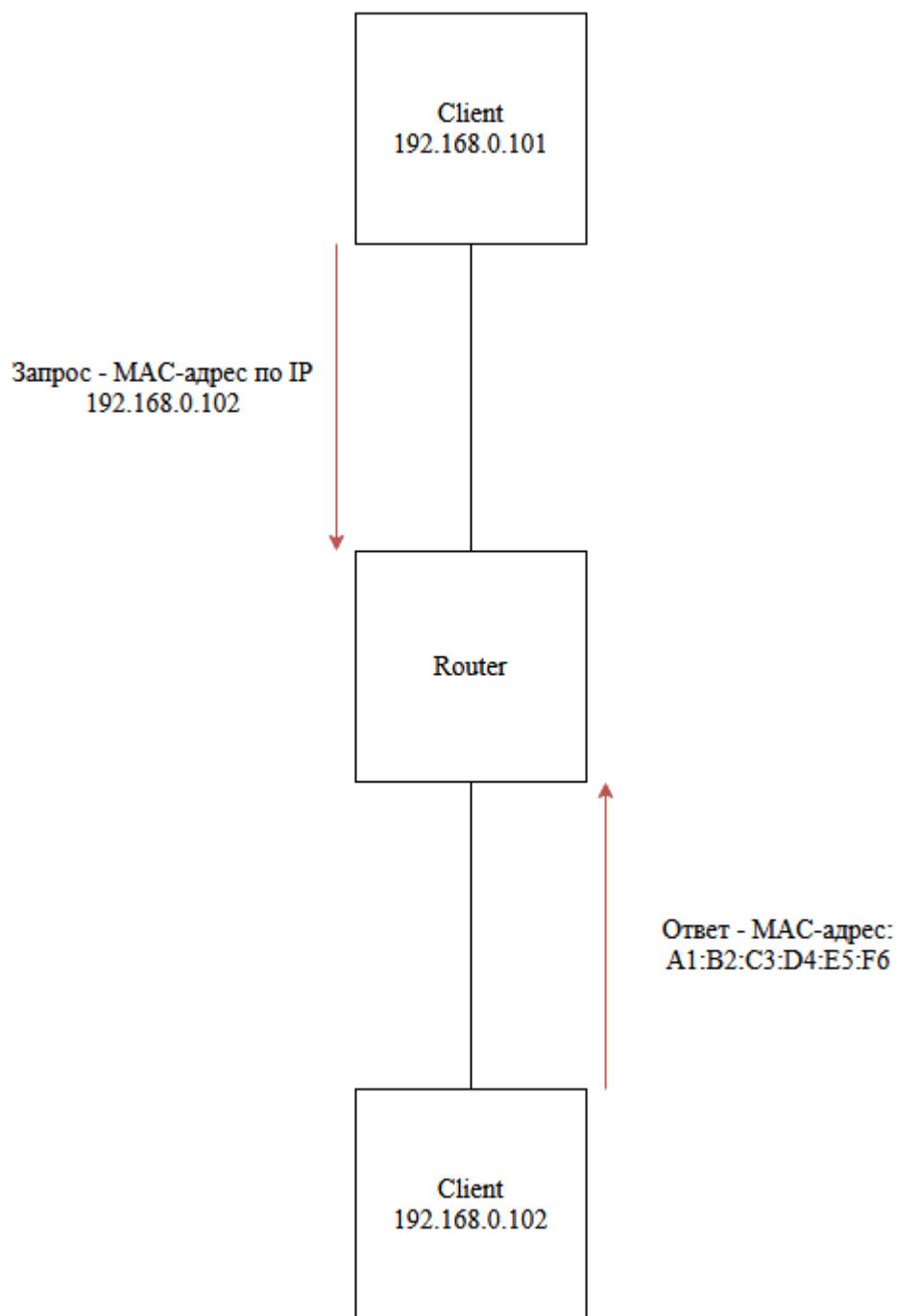


Рисунок 8 - Упрощённая схема обмена *ARP* пакетами для определения *MAC*-адреса

1.1.1.9 Протокол сетевого уровня - *IP*

Сетевой протокол *IP* - протокол для маршрутизации и адресации данных для их передачи по сети и успешной доставки конечному устройству. *IP* представляет собой набор правил для ретрансляции пакетов

данных за границы сети. Он определяет маршрут доставки и уточнение типа сетевого интерфейса устройства-приёмника. Именно данный протокол обеспечивает межсетевое взаимодействие и, по сути, образует сеть Интернет. *IP* выполняет задачу передачи информации от клиента к серверу исключительно за счёт *IP*-адресов, содержащихся в заголовках пакетов.

Также стоит отметить, что *IP*-адрес представляет собой уникальный числовой идентификатор компьютера в сети, работающей по протоколу *IP*. *IP*-адрес необходим для маршрутизации и обмена пакетами данных между источником и получателем в сети Интернет.

Существует две версии протокола *IP*: *IPv4* и *IPv6*.

1.1.1.10 Протокол сетевого уровня – *IPv4*

IPv4 является первой и всё ещё доминирующей на данный момент версией протокола *IP*. Протокол *IP* четвёртой версии предоставляет узлу сети адрес длиной 4 байта. Каждый байт содержит десятичное число от 0 до 255. В итоге *IPv4* позволяет обеспечить уникальными адресами 4.22 миллиарда устройств. Именно нехватка количества адресов 4-й версии протокола *IP* и привела к явлению, называемому исчерпанием *IPv4* адресов.

Необходимо внести ясность, что исчерпанием *IPv4* адресов называется явление, при котором стало невозможно выделять уникальные *IP*-адреса четвёртой версии каждому устройству в Интернете. Данное событие привело к созданию *IPv6* и *NAT*.

1.1.1.11 Протокол сетевого уровня – *IPv6*

IPv6 является обновленной версией протокола *IP*, которая была разработана для разрешения проблемы исчерпания *IPv4* адресов и позволяет адресовать значительно большее количество узлов сети, чем его предшественник. Адрес шестой версии протокола *IP* имеет длину в 16 байт и представляет собой восемь четырехзначных чисел в шестнадцатеричной системе счисления, что позволяет обеспечить более 300 миллионов *IP*-адресов на каждого жителя земли.

Тем не менее, внедрение *IPv6* хоть и является единственным долгосрочным решением проблемы исчерпания *IPv4*-адресов, представляет собой весьма масштабную задачу. Несмотря на то, что *IPv6* вводится в эксплуатацию начиная с 1996-го года, согласно данным *Google* на 2020-й год, доля адресов *IP* шестой версии составляет около 30% относительно суммарного количества адресов обеих версий протокола.

1.1.1.12 *Network Address Translation*

Для скорейшего выхода из ситуации был разработан *NAT*. *NAT* представляет собой инструмент в сетях *TCP/IP*, позволяющий преобразовывать *IP*-адреса пакетов, проходящих через устройство маршрутизации. Данная технология позволяет нескольким устройствам, объединенным в сеть, использовать один внешний *IP*-адрес. Суть механизма состоит в замене адреса источника при прохождении пакета данных через маршрутизатор в одну сторону и обратной замене адреса назначения в ответном пакете данных. Наряду с адресами могут также заменяться номера портов источника.

К тому же, снижение темпов исчерпания *IPv4* адресов путём экономии свободных идентификаторов не является единственной функцией *NAT*. Данная технология также повышает безопасность локальных сетей путем ограничения их от глобальной сети.

Далее рассмотрен общий принцип работы *NAT*.

На данной схеме (рисунок 9) представлено:

- клиент, находящийся в локальной сети, с *IP*-адресом 192.168.0.101;
- маршрутизатор, осуществляющий преобразование сетевых адресов;
- клиент, находящийся в сети Интернет, с *IP*-адресом 105.31.10.55.

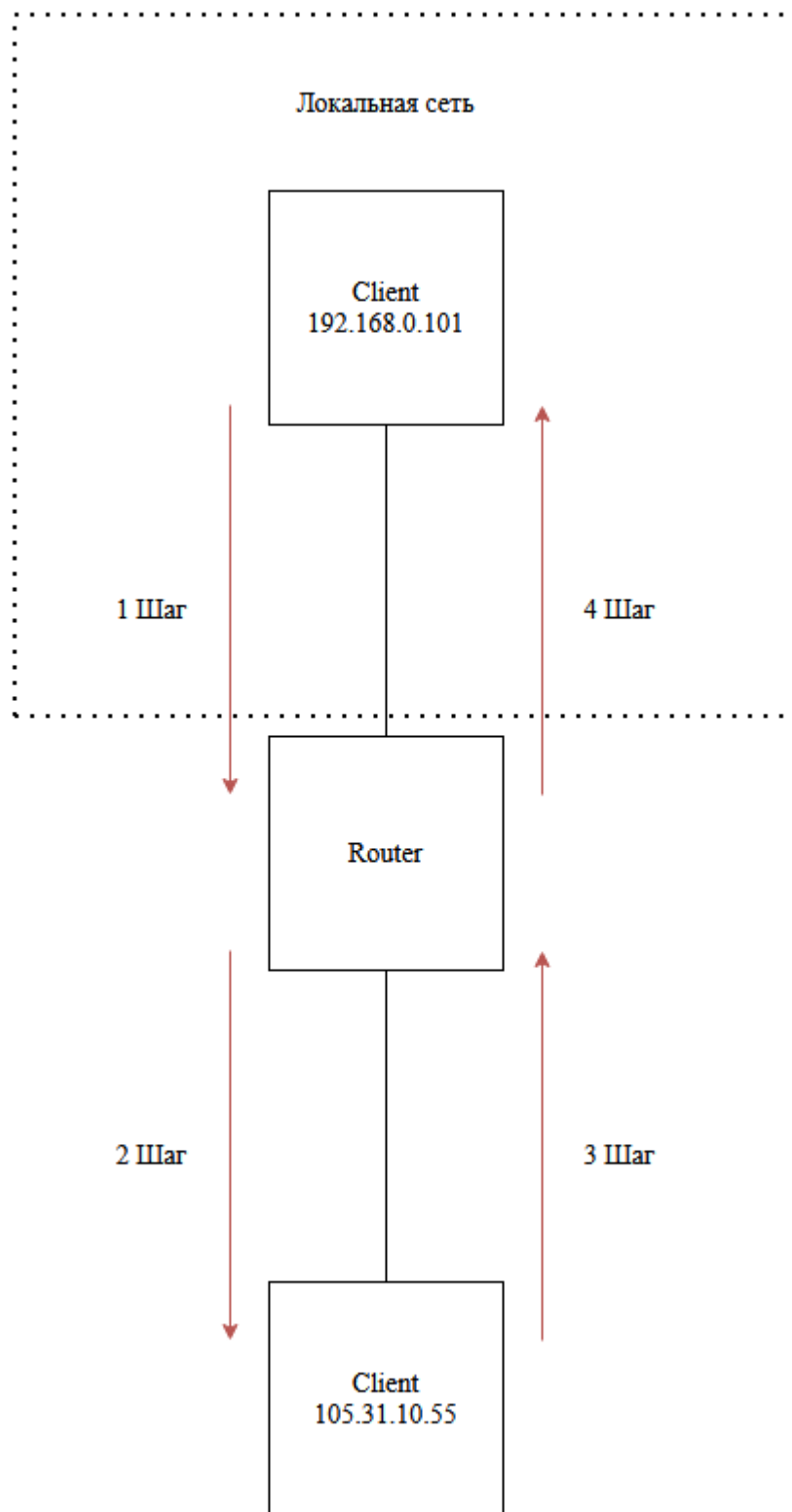


Рисунок 9 - Общая схема работы NAT

Далее описан алгоритм работы NAT при отправке пакета клиентом 192.168.0.1 из локальной сети в сеть Интернет клиенту 105.31.10.55.

Шаг 1. Клиент 192.168.0.1 отправляет пакет на устройство маршрутизации.

Шаг 2. Устройство маршрутизации получает пакет и меняет локальный IP-адрес источника 192.168.0.101, указанный в пакете, на глобальный IP-адрес. Допустим, этим глобальным IP-адресом будет 155.1.1.10. Далее маршрутизатор отправляет модифицированный пакет данных по адресу назначения – 105.31.10.55.

Шаг 3. Устройство маршрутизации получает ответный пакет с адреса 105.31.10.55. Данный пакет имеет адрес назначения 155.1.1.10, который был введен условно в предыдущем пункте. Далее устройство маршрутизации определяет, что адрес назначения данного пакета должен быть модифицирован для отправки в локальную сеть. Происходит замена адреса 155.1.1.10 на первоначальный 192.168.0.1 и отправка пакета клиенту в локальную сеть.

Шаг 4. Клиент 192.168.0.1 получает пакет данных, являющийся ответом на отправленный в шаге пакет.

На данном этапе цикл обмена сообщениями между двумя клиентами, один из которых находится в локальной сети, с помощью NAT завершён.

Стоит отметить, что программное обеспечение для преобразования сетевых адресов в основном взаимодействует с IP-адресами, но на этом список модифицируемых NAT данных не заканчивается. Например, NAT *Overload*, описание которого будет представлено далее, также модифицирует значения сетевых портов.

Также нужно подчеркнуть, что в заголовках каждого из ранее представленных протоколов имеется такое значение, как контрольная сумма.

Контрольная сумма представляет собой числовое значение, используемое для проверки целостности данных пакета. Данный элемент тоже модифицируется NAT при изменении любых других значений в пакете. Пример расчёта контрольной суммы будет представлен далее.

1.1.2 Типы ПО для преобразования сетевых адресов

1.1.2.1 Введение в типы ПО для преобразования сетевых адресов

Как было отмечено в предыдущем пункте, работа NAT не заканчивается на одной лишь модификации значений сетевых адресов. В зависимости от типа трансляции разнится и набор преобразуемых данных, и алгоритм их преобразования.

Существуют и широко используются следующие типы ПО для преобразования сетевых адресов:

- статичный NAT (англ. *Static NAT*);
- динамический NAT (англ. *Dynamic NAT*);
- перегруженный NAT (англ. *NAT Overload*).

Далее представлена более подробная характеристика каждого из типов NAT.

1.1.2.2 Статичный NAT

При использовании статичного типа NAT один локальный IP-адрес сопоставляется с одним глобальным IP-адресом, т.е. локальные адреса сопоставляются с глобальными один к одному.

Статичный NAT обычно используется для веб-хостинга и не подходит, например, для использования в организациях, имеющих большое количество устройств, которым требуется доступ в Интернет. Причина проста – для предоставления выхода в Интернет сотне условных устройств потребуется сто зарегистрированных публичных IP-адресов.

1.1.2.3 Динамический NAT

Данный тип NAT преобразует локальный IP-адрес в один из глобальных IP-адресов из пула свободных на данный момент глобальных

адресов. Если все *IP*-адреса из пула глобальных адресов заняты, пакет не будет преобразован.

Это значит, что в данном примере локальные и глобальные адреса сопоставляются не один к одному, как это было со статичным *NAT*, а один ко многим.

Тем не менее, аналогично примеру статичного *NAT*, каждому устройству, которое требует доступ в Интернет, потребуется свой зарегистрированный *IP*-адрес.

1.1.2.4 Перегруженный *NAT*

Перегруженный *NAT* заметно превосходит статичный и динамический типы ПО для преобразования сетевых адресов, так как при его использовании множество локальных *IP*-адресов может использовать один-единственный глобальный зарегистрированный *IP*-адрес. Количество локальных *IP*-адресов ограничено лишь количеством портов маршрутизатора – 65535 единиц.

Перегруженный тип ПО для преобразования сетевых адресов, следуя своему назначению, модифицирует не только значение *IP*-адреса источника данных, но и значение порта источника данных в пакете.

Данный тип *NAT* экономически выгоден, так как при его использовании через один *IP*-адрес могут быть подключены тысячи устройств в соотношении многие к одному. Также следует заметить, что именно данный тип *NAT* помог существенно замедлить скорость исчерпания *IPv4* адресов.

1.1.3 Анализ аналогов ПО

Так как проекты, реализующие *NAT* в качестве самостоятельного продукта, отсутствуют в широком потреблении, далее представлен список программного обеспечения, который затрагивает функционал преобразования сетевых адресов в том или ином виде:

- *Netstat-nat* представляет собой утилиту операционной системы *Linux* и позволяет отслеживать все сетевые соединения, модифицированные NAT;
- *Cisco's IOS «show ip nat translation»* является инструментом просмотра таблицы NAT, построенной устройствами *Cisco*;
- *FortiView for FortiGate Firewalls* – брандмауэр, предоставляющий функции по фильтрации трафика, проходящего через программное обеспечение для преобразования сетевых адресов.

1.1.4 Постановка задачи

Необходимо разработать и интегрировать в защищенную операционную систему *QP ОС* оригинальную версию программного обеспечения NAT.

В рамках данной выпускной квалификационной работы требуется реализовать совместимое с ОС *Windows* программное обеспечение для преобразования сетевых адресов с типом преобразования *NAT Overload*, функционал которого соответствует следующим требованиям:

- перехват сетевого трафика, проходящего через устройство маршрутизации;
- анализ сетевого трафика, проходящего через устройство маршрутизации;
- модификация сетевого трафика, проходящего через устройство маршрутизации, в соответствии с общепринятыми принципами работы *NAT Overload*. А точнее: модификация значения IP-адреса в сетевых пакетах данных; модификация значения сетевого порта в сетевых пакетах данных; модификация значения контрольных сумм в сетевых пакетах данных.

Также программное обеспечение должно обладать пользовательским интерфейсом, которое должно соответствовать следующим требованиям:

- предоставление пользователю возможности создать правило, в соответствии с которым будет происходить модификация данных. Данное

правило включает в себя: локальный IP-адрес; глобальный IP-адрес; локальный сетевой порт; глобальный сетевой порт;

- предоставление пользователю возможности удалить правило, в соответствии с которым происходит модификация данных;

- предоставление пользователю возможности видеть список актуальных правил, в соответствии с которыми происходит модификация данных, и взаимодействовать с каждым из элементов списка.

При реализации программного обеспечения должны быть решены следующие задачи:

- проектирование программного обеспечения, осуществляющего алгоритмы перехвата, анализа и модификации сетевого трафика, проходящего через устройство маршрутизации;

- реализация программного обеспечения, осуществляющего алгоритмы перехвата, анализа и модификации сетевого трафика, проходящего через устройство маршрутизации;

- проектирование интуитивно понятного и соответствующего ранее заданными требованиям пользовательского интерфейса;

- реализация интуитивно понятного и соответствующего ранее заданным требованиям пользовательского интерфейса;

- тестирование разработанного программного обеспечения;

- отладка разработанного программного обеспечения.

1.1.5 Выбор инструментов разработки

1.1.5.1 Выбор среды разработки

Для реализации данного проекта выбор пал на среду разработки *Microsoft Visual Studio 2019*. *Visual Studio* – популярная интегрированная среда разработки, созданная компанией *Microsoft*. Данная *IDE* используется в разработке самых разных продуктов: веб-сайтов, веб-приложений, веб-сервисов, мобильных приложений, компьютерных программ, юнит-тестов и даже драйверов. *VS* позволяет использовать такие платформы для

разработки ПО, как *Windows API*, *Windows Forms* и т.д. Также данная среда разработки поддерживает систему контроля версий *TFS* и подключение сторонних плагинов для разработки.

1.1.5.2 Выбор основного языка программирования

Языком программирования для создания драйвера и *DLL* библиотеки был выбран язык Си.

Си является процедурным императивным языком программирования общего назначения, который был разработан в 1972 году Деннисом Ритчи для разработки операционной системы *UNIX*.

Основными преимуществами языка Си являются низкоуровневый доступ к памяти и языковым конструкциям, которые преобразуются в машинный код с минимальной задержкой. Данные факторы делают язык Си одним из самых быстрых и лёгких для обработки компьютером.

Си в основном используется при разработке операционных систем, драйверов и других проектов системного уровня. В меньшей степени – при разработке прикладного ПО.

Многие языки программирования в дальнейшем прямо или косвенно заимствовали синтаксис и функционал из языка Си, что лишь подтверждает его популярность, ведь несмотря на то, что Си был разработан в семидесятых годах двадцатого века, он и на данный момент остаётся одним из самых используемых языков программирования.

1.1.5.3 Инструментарий для разработки драйвера

В качестве инструментария для создания драйвера используется *WDK*. *WDK* – это набор инструментов, использующийся для разработки, анализа, сборки, установки и тестирования драйверов под платформу *Windows*. *WDK* позволяет обнаруживать, анализировать и исправлять ошибки в коде драйвера в процессе разработки.

Данный инструментарий включает в себя документацию, примеры проектов, среду разработки и прочие инструменты для разработки драйверов.

1.1.5.4 Выбор языка программирования пользовательского интерфейса

В качестве языка программирования пользовательского интерфейса был выбран *C#*.

C# является современным объектно-ориентированным языком программирования общего назначения. Он разработан компанией *Microsoft* в 2000 году под руководством Андерса Хейлсберга в рамках инициативы *.Net*.

C# используется при создании веб-приложений, приложений для *Windows*, в разработке мобильных приложений, облачных сервисов и даже компиляторов.

C# является производной от таких языков программирования, как *Си* и *C++*, но был создан с нуля.

1.1.5.5 Инструментарий для разработки пользовательского интерфейса

Для разработки пользовательского интерфейса была выбрана платформа *Windows Forms*.

Windows Forms является фреймворком (англ. *Framework*) *.Net*, который позволяет создавать *GUI* для приложений с помощью графического дизайнера. Данная платформа имеет заранее созданный набор элементов для реализации интерфейса, но также предоставляет возможность создавать свои собственные элементы.

Стоит заметить, что *Windows Forms* используется только для разработки приложений под ОС *Windows*. ПО, реализуемое в рамках данной ВКР, как уже уточнялось ранее, изначально разрабатывалось под операционную систему *QP ОС*, где для реализации графического

интерфейса была использована иная платформа, разработанная ООО НТП «Криптософт».

1.2 Реализация программного обеспечения

1.2.1 Описание архитектуры программного обеспечения для преобразования сетевых адресов

В качестве общей архитектуры проекта был выбран шаблон, называемый многоуровневой архитектурой.

Данный шаблон подразумевает, что программное обеспечение будет разбито на модули, которые схематично можно расположить один над или под другим. Каждый такой модуль может вызывать только соседние по отношению к нему модули.

Абстрактный пример шаблона многоуровневой архитектуры приводится на рисунке 10:

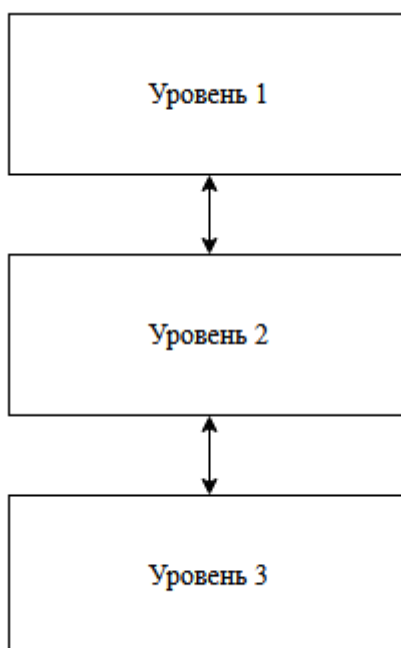


Рисунок 10 - Абстрактный пример шаблона многоуровневой архитектуры

На данном абстрактном примере показано 3 модуля ПО. Каждый из которых взаимодействует с другими следующим образом:

- уровень 1 может взаимодействовать с уровнем 2 и не может взаимодействовать с уровнем 3;
- уровень 2 может взаимодействовать как с уровнем 1, так и с уровнем 3;
- уровень 3, по аналогии с уровнем 1, может взаимодействовать только с уровнем 2, но не может взаимодействовать непосредственно с уровнем 1.

Используя данный шаблон, программное обеспечение, реализуемое в рамках данной выпускной квалификационной работы, схематично можно представить в следующем виде (рисунок 11):

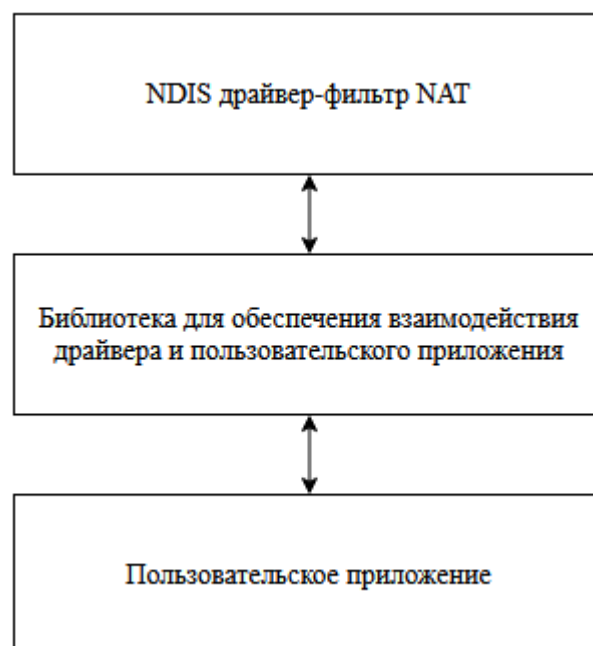


Рисунок 11 - Схематичное представление архитектуры проекта при использовании многоуровневого шаблона

1.2.2 Разработка программного обеспечения для преобразования сетевых адресов

1.2.2.1 Выбор модели реализации основного модуля NAT

Так как в качестве основного модуля программного обеспечения невозможно использовать пользовательское приложение, было решено разработать драйвер-фильтр *NDIS*.

Причиной, по которой ПО для преобразования сетевых адресов не предоставляется возможным реализовать как пользовательское приложение, служит тот факт, что *NAT* лишь взаимодействует с уже существующими пакетами данных, проходящими через устройство маршрутизации. Как было отмечено в пункте 1.1.1.4 Модель *TCP/IP*, пакет проходит следующий путь при отправке (рисунок 12):

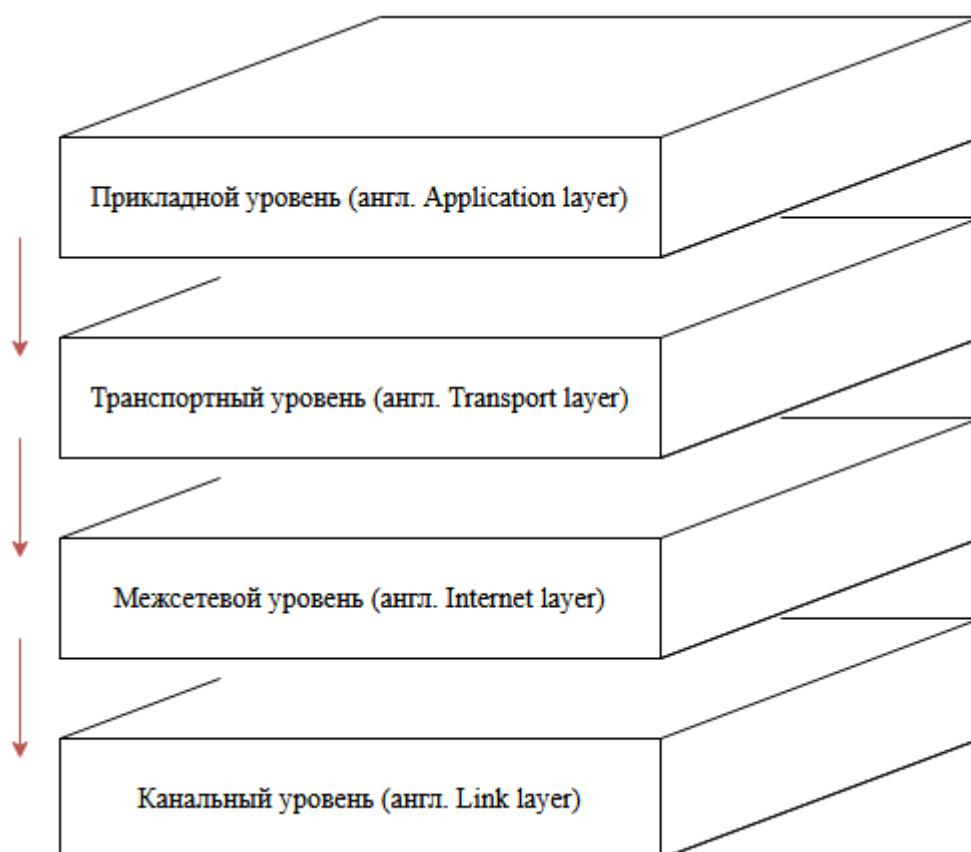


Рисунок 12 - Схема отправки пакета по модели *TCP/IP*

Ранее отмечалось, что на прикладном уровне, который и используют пользовательские приложения, пакет только создаётся, после чего передаётся на нижние уровни модели *TCP/IP*. Также стоит отметить, что протокол транспортного уровня *ICMP* тоже имеет возможность создания пакетов данных. Тем не менее, преобразование сетевых адресов происходит после формирования и “обволакивания” пакета всеми необходимыми

заголовками сетевых протоколов. Встраивание *NAT* в данную модель можно представить следующим образом (рисунок 13):



Рисунок 13 - Схема отправки пакета по модели *TCP/IP* после включения в неё *NAT*

Всё это подтверждает факт, что основной модуль программного обеспечения, разрабатываемого в рамках данной работы, не может быть реализован на прикладном уровне. Однако, *NAT* может быть реализован на системном уровне с помощью драйвера.

Именно драйверы являются той фундаментальной единицей, позволяющей взаимодействовать операционной системе с физическими устройствами. Стоит уточнить, что не все драйверы взаимодействуют с

устройствами напрямую. Драйверы также могут быть предназначены для взаимодействия друг с другом, осуществляя какой-либо функционал в стеке драйверов.

Алгоритм взаимодействия ОС с сетевым адаптером в случае установки *NAT* условно можно представить следующим образом (рисунок 14):



Рисунок 14 – Условная схема взаимодействия ОС и сетевого адаптера после установки *NAT*

Как подчёркивалось ранее, существует 2 типа драйверов: функциональный драйвер и драйвер-фильтр.

Функциональным называется драйвер, который производит взаимодействие непосредственно с каким-либо устройством.

Драйвер-фильтр же, напротив, не производит никакого взаимодействия с устройством, а выполняет лишь вспомогательную обработку передаваемой через него информации.

Условно схему взаимодействия операционной системы с физическим устройством при помощи обоих типов драйверов можно представить следующим образом (рисунок 15):

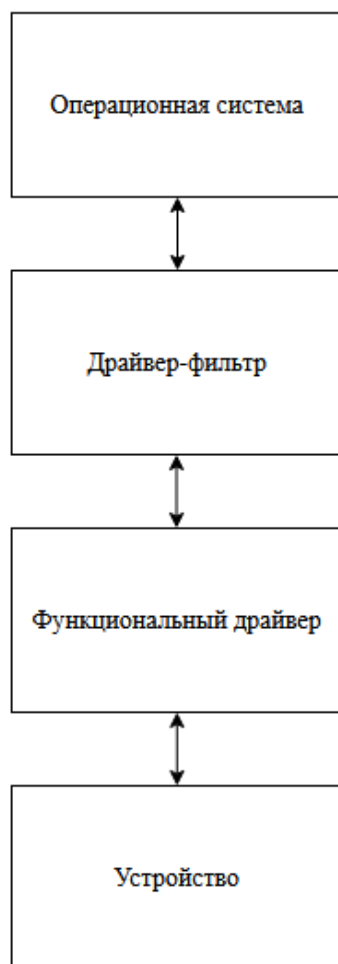


Рисунок 15 – Условная схема взаимодействия операционной системы и физического устройства при помощи функционального драйвера и драйвера-фильтра

Но для разработки сетевых драйверов используется спецификация *NDIS*. *NDIS* представляет собой набор инструментов для поддержки сетевых интерфейсов и их стеков и служит для координации ОС и сетевых адаптеров.

В рамках *NDIS* существуют следующие типы сетевых драйверов:

- драйвер минипорта (англ. *Miniport driver*) – драйвер, который управляет адаптерами минипорта и предоставляет интерфейс к адаптерам, которые впоследствии используются драйверами более высоких уровней. Адаптером минипорта называется абстрактная единица, которая представляет физическое или виртуальное устройство – например, сетевую карту;

- драйвер протокола (англ. *Protocol driver*) – данный драйвер привязывается к адаптерам минипорта и реализует интерфейс на своей верхней границе для предоставления каких-либо услуг прикладному уровню модели *TCP/IP*. На нижней же своей границе драйвер протокола предоставляет интерфейс для взаимодействия с иным драйвером, находящимся ниже в стеке;

- драйвер-фильтр (англ. *Filter driver*) – фильтрующий драйвер. Как можно понять из названия, фильтрует информацию, транслируемую между драйверами минипорта и драйверами протокола. Драйвер-фильтр имеет возможность просмотра и изменения проходящей через него информации.

Исходя из всей предоставленной информации, можно сделать вывод, что для реализации основного модуля *NAT*, который будет осуществлять перехват, анализ и модификацию сетевых данных, идеально подойдёт модель *NDIS* драйвера-фильтра.

1.2.2.2 Разработка драйвера-фильтра *NAT*

Основными модулями, без которых невозможно функционирование драйвера, являются непосредственно модуль драйвера-фильтра и модуль

объекта устройства. В данном проекте они соответствуют модулям *Nat.c* и *Device.c*.

Модуль *Device.c* отвечает за операции, проводимые с объектом устройства. К таким операциям относятся:

- регистрация объекта устройства в системе при вызове функции *RegisterDevice()*;
- deregистрация объекта устройства в системе при вызове функции *DeregisterDevice()*;
- обработка *IOCTL* запросов при вызове функции *DeviceIoControl()*.

Сам же объект устройства представляет собой абстракцию и предназначен для реализации функционала по обеспечению взаимодействия программного обеспечения системного уровня и приложений пользовательского уровня при помощи *IOCTL* запросов.

IOCTL запросы, в свою очередь, представляют собой интерфейсы, через которые обеспечивается связь между пользовательским приложением и объектом устройства, который впоследствии взаимодействует непосредственно с драйвером.

Модуль *Nat.c* представляет непосредственно основной функционал сетевого драйвера. Данный модуль содержит такие элементы, как:

- точка входа в драйвер, функция *DriverEntry()*. Данная функция служит исходной точкой инициализации драйвера в системе и выполняется сразу при начале загрузки драйвера системой;
- функции по регистрации и deregистрации драйвера в *NDIS*;
- функции по взаимодействию с объектом устройства;
- *callback*-функции, которые вызываются при возникновении определённых событий, затрагивающих драйвер, в системе. Примером таких событий являются отправка системой сетевого пакета и приём системой сетевого пакета.

Общая структура событий, отслеживаемых и обрабатываемых драйвером, довольно объёмна по своей сути, но подвергается модификации в зависимости от назначения драйвера. В рамках данной работы будут описаны только основные функции, использующиеся драйвером-фильтром для осуществления преобразования сетевых адресов.

Функциями драйвера-фильтра, за счёт которых осуществляется преобразование сетевых адресов, являются:

- *FilterSendNetBufferLists()* – функция, выполняющая обработку пакетов, переданных системой на отправку;
- *FilterSendNetBufferListsComplete()* – функция, освобождающая память пакета после его передачи системой на канальный уровень;
- *FilterReceiveNetBufferLists()* – функция, выполняющая обработку пакетов, переданных системой на приём;
- *FilterReturnNetBufferLists()* – функция, осуществляющая передачу принятого и уже обработанного драйвером пакета в систему.

Стоит подчеркнуть, что алгоритмы функций *FilterSendNetBufferLists()* и *FilterReceiveNetBufferLists()*, ровно как *FilterSendNetBufferListsComplete()* и *FilterReturnNetBufferLists()* очень похожи. Их отличия состоят в том, что *FilterSendNetBufferLists()* и *FilterSendNetBufferListsComplete()* осуществляют манипуляции с пакетами, направленными системой на канальный уровень для дальнейшей передачи в сеть, а *FilterReceiveNetBufferLists()* и *FilterReturnNetBufferLists()* – манипуляции с принятыми из сети пакетами для их дальнейшей передачи на прикладной уровень.

Исходя из данного факта, далее будут представлены алгоритмы работы функций *FilterSendNetBufferLists()* и *FilterSendNetBufferListsComplete()* и их описание.

На рисунке 16 представлен алгоритм работы функции *FilterSendNetBufferLists()*.

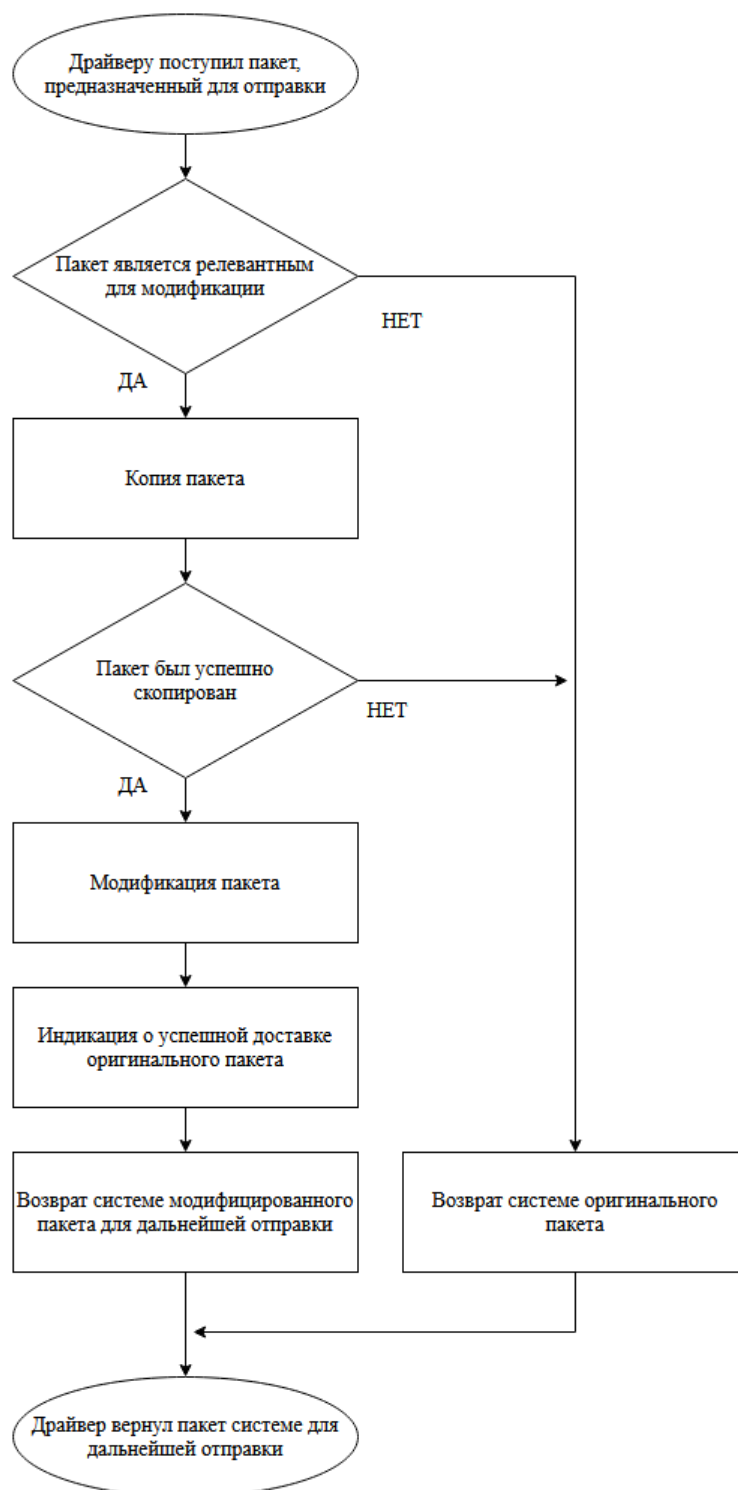


Рисунок 16 - Алгоритм работы *FilterSendNetBufferLists()*

Первым делом, когда пакет данных попадает в функцию, он проверяется на релевантность для дальнейшей модификации. В случае NAT, релевантным является пакет, принадлежащий семействам сетевых

протоколов *ARP* или *IPv4*. Также *IP*-адрес и порт источника данного пакета должны быть занесены в таблицу преобразования сетевых адресов, иначе пакет не будет модифицирован.

Далее происходит полная копия пакета данных. Делается это по той причине, что модификация пакета, созданного системой, приведёт к ошибке безопасности, вызванной *NDIS*. По факту успешной копии пакета, происходит дальнейшая его модификация.

Модификация пакета представляет собой замену значений *IP*-адреса, сетевого порта в случае работы с протоколами семейств *TCP* или *UDP*, а также значений контрольной суммы.

Как было сказано ранее, контрольной суммой называется значение, выполняющее роль по обнаружению ошибок в заголовках протоколов. На примере протокола *IPv4*, контрольная сумма представляет собой 16-битное значение, являющееся результатом сложения информации заголовка протокола пословно. “Пословно” в данном контексте обозначает, что одно слагаемое представляет собой значение размером в 2 байта, которое также называется словом (англ. *Word*).

Несмотря на то, что пакет, подвергнутый модификации, является скопированным и никак не затрагивает оригинальный пакет, системный пакет всё равно требует обработки. На данном этапе выполняется индикация оригинального пакета системе и освобождение его памяти. Это необходимо, во-первых, для контроля за памятью, во-вторых, для сигнализации *NDIS*, о том, что пакет, созданный системой, был успешно обработан. Исключение данного шага неизбежно приведёт к краху системы.

Последним этапом служит передача модифицированного пакета в систему для его дальнейшей отправки в сеть. Данный шаг по обработке отправляемого пакета является последним для драйвера.

На рисунке 17 представлен алгоритм работы функции *FilterSendNetBufferListsComplete()*.

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

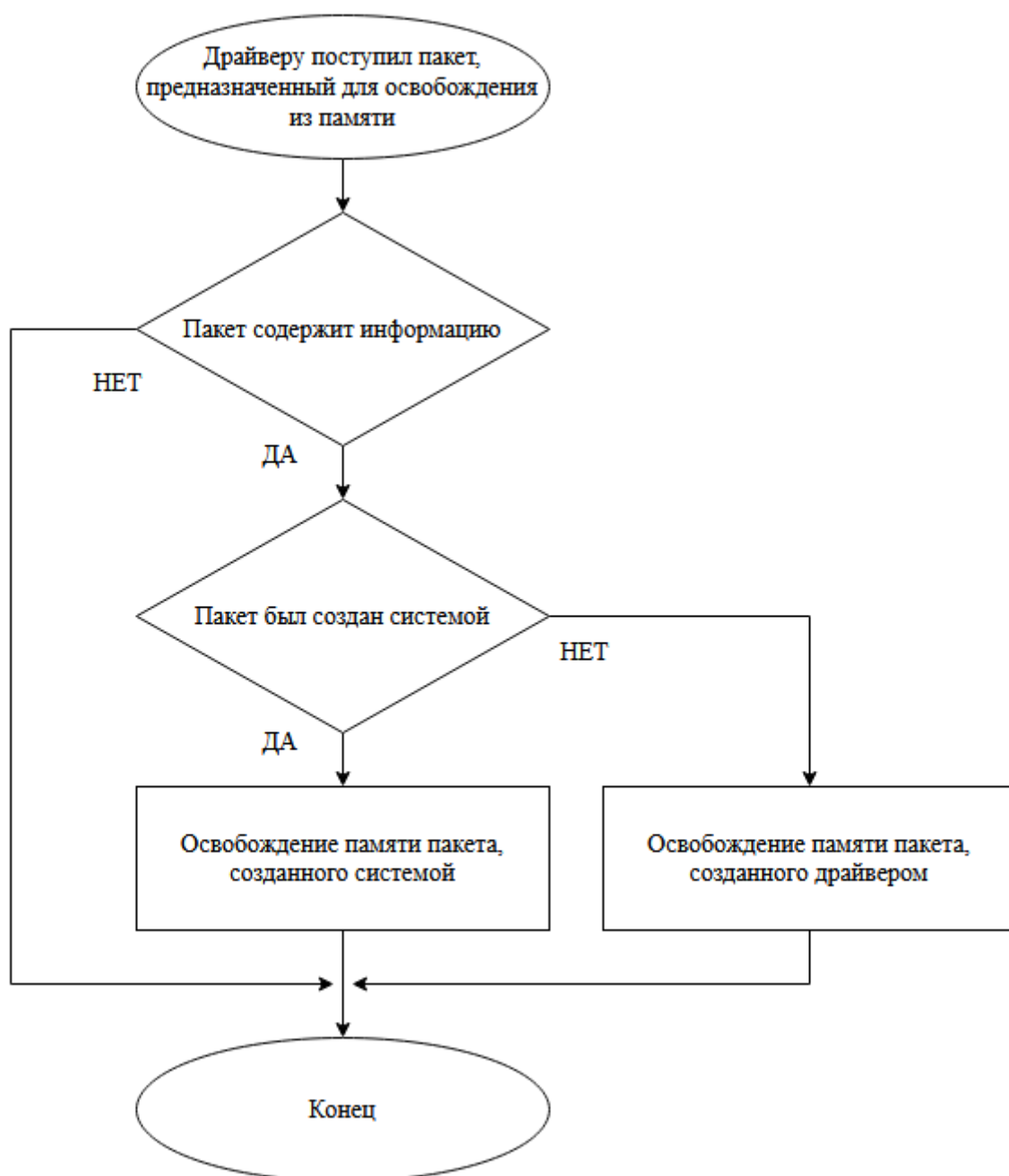


Рисунок 17 - Алгоритм работы *FilterSendNetBufferListsComplete()*

Как уже было указано, функция *FilterSendNetBufferListsComplete()* осуществляет уведомление системы об успешной обработке пакета данных и дальнейшее освобождение памяти пакета.

Когда пакет попадает в данную функцию, он проверяется на наличие каких-либо данных. Затем происходит его обработка.

На данном этапе возможны 2 варианта: пакет либо создан системой, либо является созданным драйвером для модификации. Модифицированный пакет отслеживается путём просмотра дескриптора памяти, по которому был создан данный пакет.

Стоит отметить, что пакеты, созданные драйвером, необходимо обрабатывать самому драйверу вплоть до момента освобождения их памяти, тогда как пакеты, выделенные системой, обрабатываются с помощью функционала *NDIS*.

Драйвер-фильтр также имеет второстепенные модули, каждый из которых каким-либо образом участвует в процессе преобразования сетевых адресов:

- *NatTable.c* осуществляет функционал таблицы *NAT*. Данный модуль отвечает за инициализацию таблицы, различную модификацию её элементов, а также предоставляет функционал для поиска существующих в ней элементов. С *NatTable.c* взаимодействует модуль *Device.c* для обеспечения синхронизации таблицы, представленной на пользовательском уровне, и данной таблице, располагающейся на системном уровне;

- *PacketCopying.c* отвечает за копирование пакетов данных. Данный модуль предоставляет такие функции как копия пакета данных, проверка пакета данных на релевантность и освобождение копированного пакета;

- *PacketModification.c* предоставляет функционал по модификации пакета данных, осуществляя поиск единицы в таблице *NAT* в соответствии с данными в самом пакете и дальнейшее преобразование информации в пакете данных.

Теперь, когда произведено описание всех модулей драйвера-фильтра, можно составить схему взаимодействия данных модулей (рисунок 18).

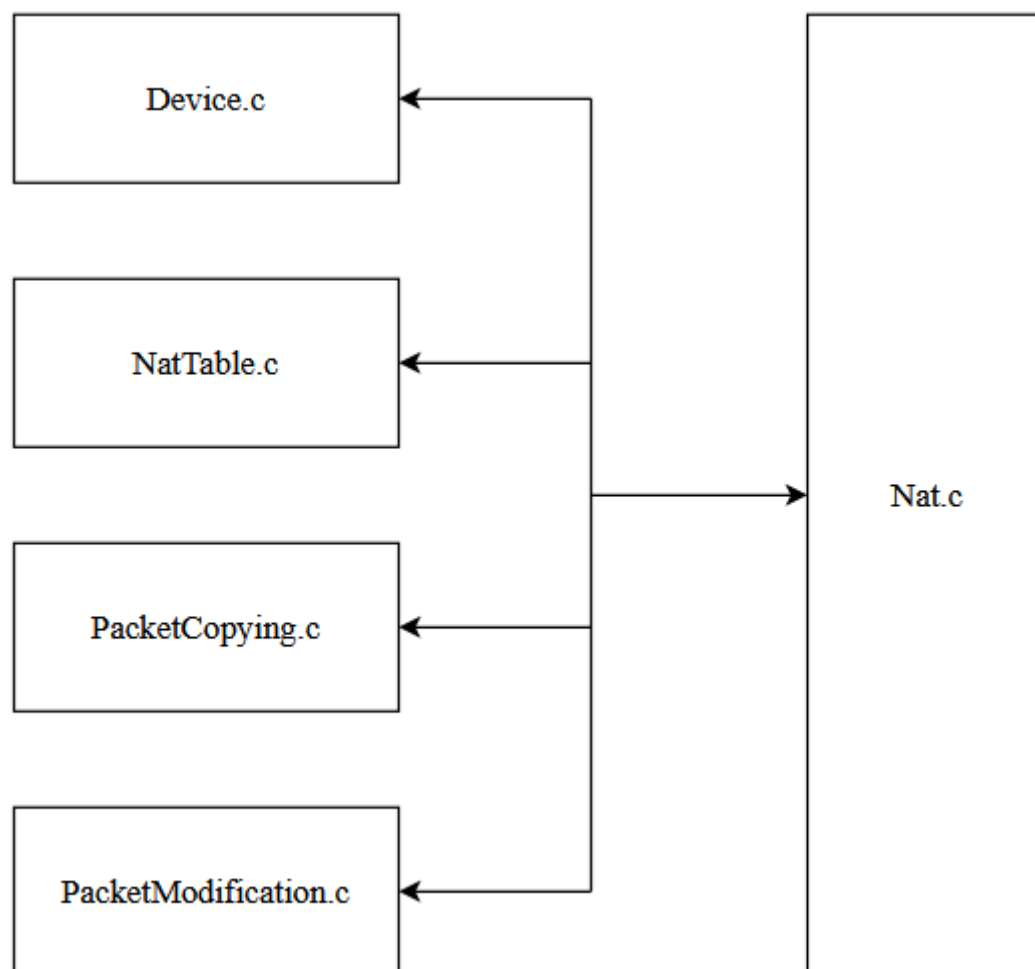


Рисунок 18 - Схема взаимодействия модулей драйвера-фильтра NAT

Несмотря на то, что *Nat.c* является главным модулем, который имеет доступ ко всем остальным модулям драйвера, *NatTable.c* также играет одну из ключевых ролей в преобразовании сетевых адресов, поскольку все второстепенные модули драйвера так или иначе используют возможности *NatTable.c* для реализации своего функционала.

Так, *Device.c* взаимодействует с *NatTable.c* при обработке *IOCTL* запросов. Например, когда таблица NAT модифицируется на пользовательском уровне, приложение отправляет *IOCTL* запрос объекту устройства, функционал которого и обеспечивает *Device.c*. Далее, в зависимости от запроса, *Device.c* вызывает ту или иную функцию *NatTable.c*.

Модуль *PacketCopying.c* при копировании пакета данных может обратиться к *NatTable.c*, чтобы уточнить, является ли пакет релевантным для модификации.

В заключение, *PacketModification.c* обращается к *NatTable.c*, чтобы получить информацию о соответствующей пакету NAT-единице для дальнейшей модификации пакета.

1.2.2.3 Разработка пользовательского интерфейса

Для разработки пользовательского графического интерфейса использовалась платформа для создания приложений на *Windows*, называемая *Windows Forms*.

При разработке учитывалось, что интерфейс должен быть максимально простым, понятным и приятным пользователю.

На рисунке 19 представлен графический интерфейс пользовательского приложения NAT.

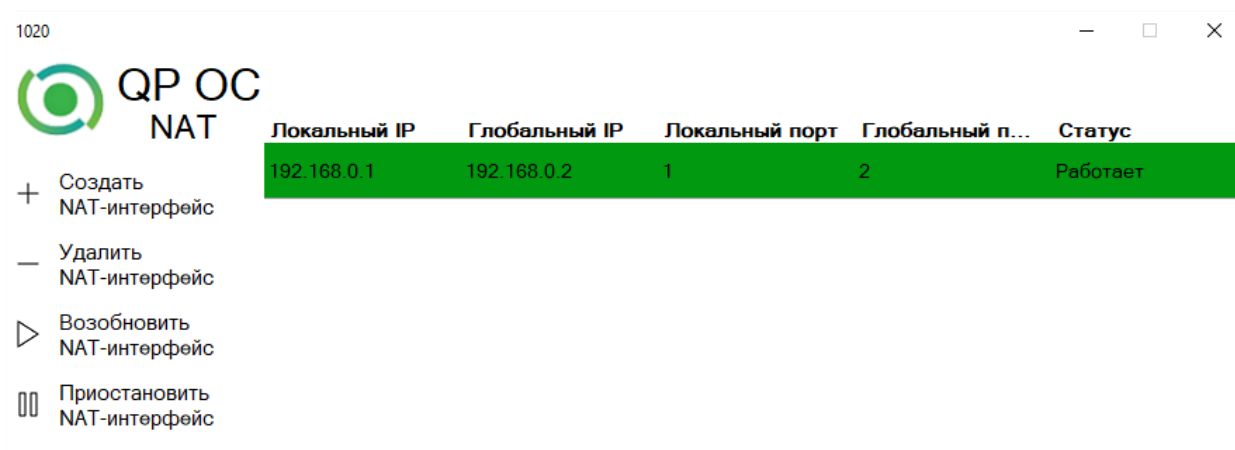


Рисунок 19 - Графический интерфейс пользовательского приложения NAT

Основным элементом интерфейса служит таблица, являющаяся представлением таблицы NAT драйвера. Манипуляции таблицей производятся с помощью представленных пользователю кнопок.

При создании нового NAT-интерфейса будет вызвана второстепенная форма, на которой пользователю будет предложено задать значения, в соответствии с которыми будет производиться модификация сетевого

трафика. На рисунке 20 представлен интерфейс формы для создания элемента таблицы NAT.

The image shows a graphical user interface for creating a new NAT element. It consists of a light gray rectangular container. Inside, there are four text input fields arranged vertically. The first field is labeled 'Локальный IP' (Local IP), the second 'Глобальный IP' (Global IP), the third 'Локальный порт' (Local port), and the fourth 'Глобальный порт' (Global port). Below these fields is a wide, light gray button with the text 'Создать' (Create) in the center.

Рисунок 20 - Графический интерфейс формы добавления нового элемента в таблицу NAT

1.2.2.4 Разработка библиотеки для взаимодействия драйвера-фильтра NAT и пользовательского приложения

На данном этапе имеется разработанный драйвер-фильтр NAT и пользовательское приложение NAT. Для обеспечения взаимодействия программного обеспечения сетевого и пользовательского уровня используется функционал *Windows API*.

Windows API представляет собой набор базовых функций ОС *Windows*. Данный инструментарий позволяет обеспечить взаимодействие ПО пользовательского уровня с операционной системой *Windows*.

Функционал данной библиотеки необходимо было выделить в отдельный проект, так как *Windows API* спроектирован для использования языком программирования Си, в то время как пользовательское приложение NAT создано на языке C#.

Библиотека интегрируется в пользовательское приложение NAT путём импорта *DLL* файла, а с драйвером она взаимодействует при помощи *IOCTL* запросов.

Для взаимодействия с драйвером необходимо иметь его дескриптор. Чтобы получить дескриптор драйвера, находящегося в системе, используется функция *CreateFile()*.

Код функции *CreateFile()*, применённой в рамках данной работы:

```
HANDLE hDriver = CreateFile  
(  
    NAT_DRIVER_NAME_RING3,  
    GENERIC_WRITE | GENERIC_READ,  
    0,  
    NULL,  
    OPEN_EXISTING,  
    FILE_ATTRIBUTE_NORMAL,  
    NULL  
);
```

После получения дескриптора появляется возможность взаимодействия с драйвером при помощи функции *DeviceIoControl()*. Данная функция обеспечивает взаимодействие с драйвером при помощи *IOCTL* запросов.

Код функции *DeviceIoControl()*, применённой в рамках данной работы при запросе на добавление элемента в таблицу *NAT*:

```
BOOLEAN bStatus = DeviceIoControl  
(  
    hDriver,  
    IOCTL_NAT_CREATE_UNIT,  
    pcDataBuffer,  
    iDataBufferSize,  
    NULL,  
    0,  
    NULL,  
    NULL  
);
```

Полный список функций, которые предоставляет библиотека при взаимодействии с драйвером выглядит следующим образом:

- *DllNatOpen()* позволяет получить дескриптор драйвера в системе для дальнейшего взаимодействия с ним;
- *DllNatClose()* закрывает дескриптор при завершении работы с драйвером;
- *DllNatCreateUnit()* отправляет запрос на добавление элемента в таблицу NAT;
- *DllNatRemoveUnit()* отправляет запрос на удаление элемента из таблицы NAT;
- *DllNatWipeUnitList()* отправляет запрос на удаление всех элементов таблицы NAT;
- *DllNatIsUnitExists()* отправляет запрос на уточнение, существует ли данный элемент в таблице NAT;
- *DllNatIsUnitRunning()* отправляет запрос на уточнение, производится ли модификация данных в соответствии с заданным элементом таблицы NAT;
- *DllNatSuspendUnit()* отправляет запрос на прекращение модификации данных в соответствии с заданным элементом таблицы NAT;
- *DllNatResumeUnit()* отправляет запрос на возобновление модификации данных в соответствии с заданным элементом таблицы NAT.

1.3 Анализ результата

1.3.1 Руководство пользователя

В данном разделе описано руководство по корректному использованию разработанного программного обеспечения.

При запуске пользователю открывается вид на пока ещё пустую таблицу NAT, в соответствии с элементами которой будет осуществляться преобразование сетевых адресов, а также элементы для взаимодействия с

таблицей. Вид приложения при начале работы с ним представлен на рисунке 21.

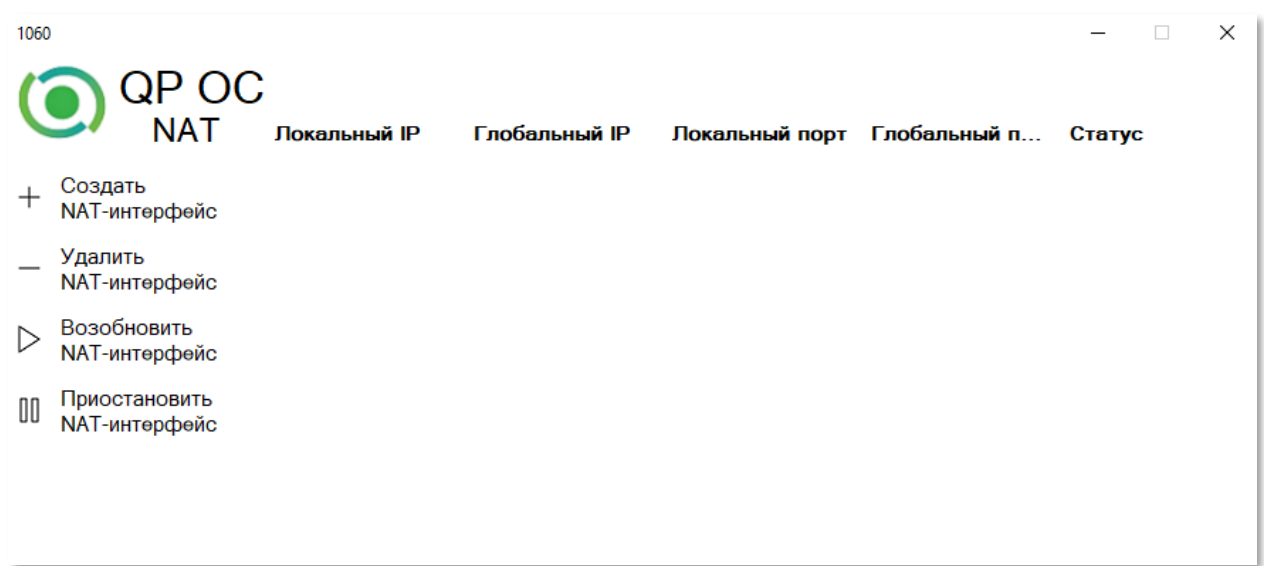


Рисунок 21 - Интерфейс главного окна приложения при запуске

Для преобразования сетевых адресов необходимо создать элемент таблицы NAT, в соответствии с которым будет производиться модификация. Чтобы создать элемент таблицы NAT, нужно нажать кнопку «Создать NAT-интерфейс» и заполнить поля данных. Окно создания элемента таблицы NAT с незаполненными полями данных представлено на рисунке 22.

Рисунок 22 - Интерфейс окна создания элемента таблицы NAT

Корректным значением *IP* адреса считается значение, состоящее из четырёх чисел от 0 до 255 включительно, каждое из которых разделено точками между собой. Например, «192.168.0.1».

Корректным значением сетевого порта является число от 0 до 65535.

Окно создания элемента таблицы *NAT* с корректно заполненными полями данных представлено на рисунке 23.

Локальный IP	192.168.0.10
Глобальный IP	192.168.0.101
Локальный порт	1025
Глобальный порт	10250
Создать	

Рисунок 23 - Интерфейс окна создания элемента таблицы *NAT* с заданными значениями

После нажатия на кнопку «Создать» окно создания элемента таблицы *NAT* закроется, и созданный элемент появится в таблице *NAT*. Вид таблицы *NAT* после добавления элемента представлен на рисунке 24.

1060 QP OS NAT		Локальный IP	Глобальный IP	Локальный порт	Глобальный п...	Статус
+	Создать NAT-интерфейс	192.168.0.10	192.168.0.101	1025	10250	Работает
-	Удалить NAT-интерфейс					
▷	Возобновить NAT-интерфейс					
⏏	Приостановить NAT-интерфейс					

Рисунок 24 - Интерфейс главного окна приложения после создания элемента таблицы *NAT*

Для приостановки модификации сетевого трафика в соответствии с выбранным элементом таблицы NAT требуется нажать кнопку «Приостановить NAT-интерфейс», для возобновления – «Возобновить NAT-интерфейс». Вид таблицы NAT после приостановки модификации сетевого трафика в соответствии с выбранным элементом таблицы NAT представлен на рисунке 25.

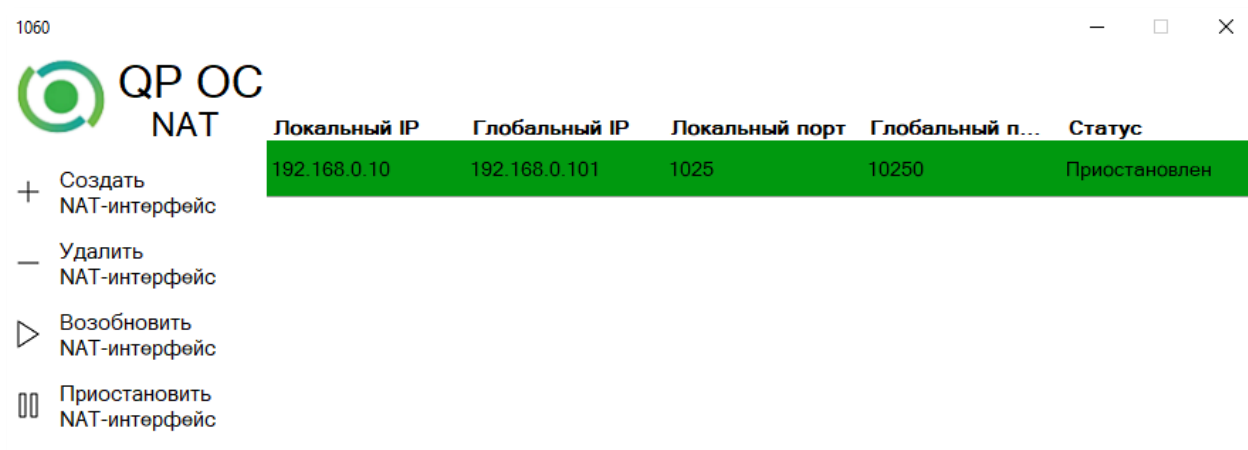


Рисунок 25 - Приостановка модификации данных в соответствии с выбранным элементом таблицы NAT

На рисунке 26 и рисунке 27 представлено создание дополнительного элемента таблицы NAT и удаление выбранного элемента таблицы NAT с помощью нажатия на кнопку «Удалить NAT-интерфейс» соответственно.

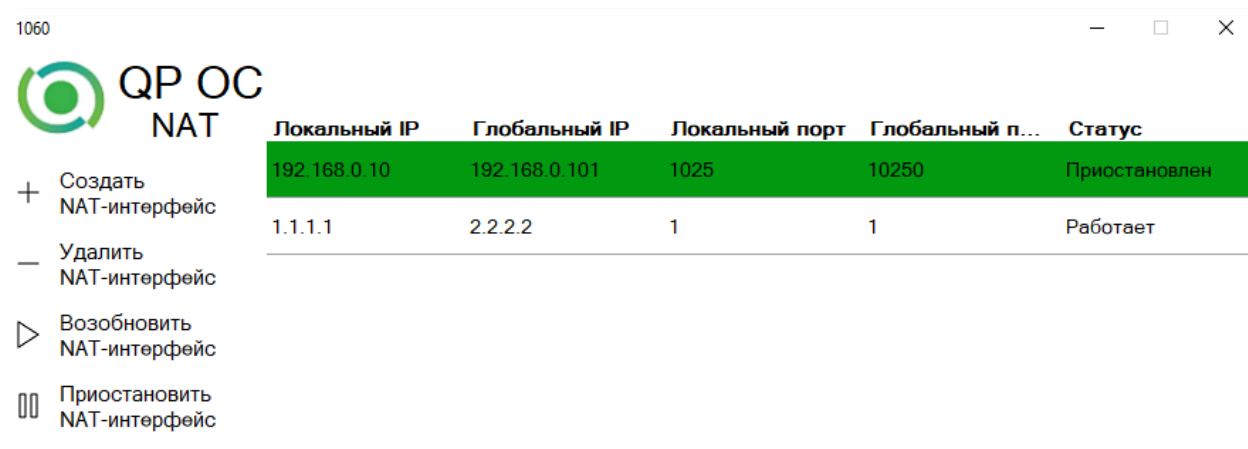


Рисунок 26 - Создание дополнительного элемента таблицы NAT

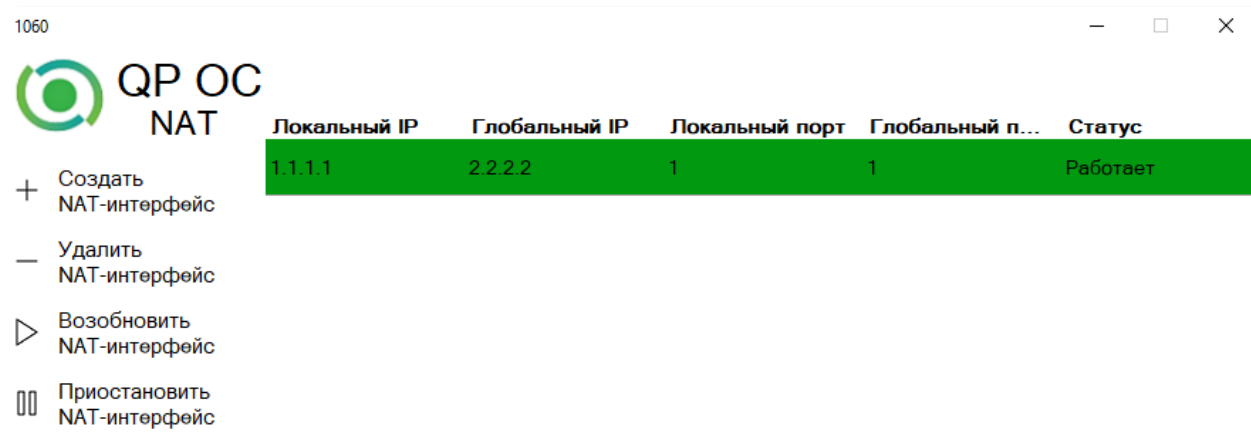


Рисунок 27 - Удаление элемента таблицы NAT

1.3.2 Тестирование и отладка

В ходе разработки программного обеспечения проведено тестирование драйвера-фильтра NAT и пользовательского приложения NAT, а также тестирование их взаимодействия с помощью библиотеки.

Перечень проверенного функционала пользовательского приложения NAT:

- создание элемента таблицы NAT пользовательским приложением;
- проверка вводимых пользователем данных при создании нового элемента таблицы NAT пользовательским приложением - проверка значения локального IP-адреса;
- проверка вводимых пользователем данных при создании нового элемента таблицы NAT пользовательским приложением - проверка значения глобального IP-адреса;
- проверка вводимых пользователем данных при создании нового элемента таблицы NAT пользовательским приложением - проверка значения локального сетевого порта;
- проверка вводимых пользователем данных при создании нового элемента таблицы NAT пользовательским приложением - проверка значения глобального сетевого порта;

- приостановка модификации данных в соответствии с выбранным элементом таблицы *NAT* пользовательским приложением;
- возобновление модификации данных в соответствии с выбранным элементом таблицы *NAT* пользовательским приложением;
- удаление элемента таблицы *NAT* пользовательским приложением.

Перечень проверенного функционала по взаимодействию пользовательского приложения и драйвера-фильтра с помощью библиотеки:

- создание элемента таблицы *NAT* пользовательским приложением и корректная обработка данного *IOCTL* запроса драйвером;
- приостановка модификации данных в соответствии с выбранным элементом таблицы *NAT* пользовательским приложением и корректная обработка данного *IOCTL* запроса драйвером;
- возобновление модификации данных в соответствии с выбранным элементом таблицы *NAT* пользовательским приложением и корректная обработка данного *IOCTL* запроса драйвером;
- удаление элемента таблицы *NAT* пользовательским приложением и корректная обработка данного *IOCTL* запроса драйвером.

Перечень проверенного функционала драйвера-фильтра *NAT*:

- создание элемента таблицы *NAT* драйвером-фильтром;
- приостановка модификации данных в соответствии с определённым элементом таблицы *NAT* драйвером-фильтром;
- возобновление модификации данных в соответствии с определённым элементом таблицы *NAT* драйвером-фильтром;
- удаление элемента таблицы *NAT* драйвером-фильтром.

1.3.1.1 Тестирование пользовательского приложения *NAT*

В приведённой далее таблице 1 описан блок тестов, проведённых с пользовательским приложением *NAT*. В таблице описано: цель теста,

заданная информация, способ выполнения теста, ожидаемый и фактический результат тестирования.

Таблица 1 - Тестирование пользовательского приложения NAT

Проверяется: Функционал пользовательского приложения NAT					
Документ: ВКР, глава 1, пункт 3.1.1					
№ тест а	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактический результат
ПФ 1.1	Проверить корректность работы функции создания элемента таблицы NAT пользовательским приложением	Значения полей: «Локальный IP-адрес» = «1.1.1.1»; «Глобальный IP-адрес» = «2.2.2.2»; «Локальный порт» = «1»; «Глобальный порт» = «2»;	Нажать кнопку «Создать NAT-интерфейс»; Задать указанные значения полей; Нажать кнопку «Создать»	В таблице элементов NAT должен появиться элемент со значениями полей: «Локальный IP-адрес» = «1.1.1.1»; «Глобальный IP-адрес» = «2.2.2.2»; «Локальный порт» = «1»; «Глобальный порт» = «2»;	Совпадает с ожидаемым
ПФ 1.2	Проверить корректность обработки корректно заданной информации в поле	Значение поля «Локальный IP-адрес» = «1.1.1.1»	Задать значение поля «Локальный IP-адрес» = «1.1.1.1»;	Программа не должна сообщать о некорректно введенных данных в поле	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT

Документ: ВКР, глава 1, пункт 3.1.1

№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	«Локальный IP-адрес» при создании элемента таблицы NAT пользовательским приложением		Нажать кнопку «Создать»	«Локальный IP-адрес»	
ПФ 1.3	Проверить корректность обработки некорректно заданной информации в поле «Локальный IP-адрес» при создании элемента таблицы NAT пользовательским приложением	Значение поля «Локальный IP-адрес» = «АБВГД»	Задать значение поля «Локальный IP-адрес» = «АБВГД»; Нажать кнопку «Создать»	Программа должна сообщить о некорректно введенных данных в поле «Локальный IP-адрес»	Совпадает с ожидаемым
ПФ 1.4	Проверить корректность обработки корректно заданной информации в	Значение поля «Глобальный IP-адрес» = «1.1.1.1»	Задать значение поля «Глобальный IP-адрес» = «1.1.1.1»;	Программа не должна сообщать о некорректно введенных данных в поле	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT

Документ: ВКР, глава 1, пункт 3.1.1

№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	поле «Глобальный IP-адрес» при создании элемента таблицы NAT пользовательск им приложением		Нажать кнопку «Создать»	«Глобальный IP-адрес»	
ПФ 1.5	Проверить корректность обработки некорректно заданной информации в поле «Глобальный IP-адрес» при создании элемента таблицы NAT пользовательск им приложением	Значение поля «Глобальн ый IP- адрес» = «АБВГД»	Задать значение поля «Глобальный IP-адрес» = «АБВГД»; Нажать кнопку «Создать»	Программа должна сообщить о некорректно введённых данных в поле «Глобальный IP-адрес»	Совпадает с ожидаемым
ПФ 1.6	Проверить корректность обработки корректно заданной	Значение поля «Локальны й порт» = «123»	Задать значение поля «Локальный порт» = «123»;	Программа не должна сообщать о некорректно введённых	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT

Документ: ВКР, глава 1, пункт 3.1.1

№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	информации в поле «Локальный порт» при создании элемента таблицы NAT пользовательск им приложением		Нажать кнопку «Создать»	данных в поле «Локальный порт»	
ПФ 1.7	Проверить корректность обработки некорректно заданной информации в поле «Локальный порт» при создании элемента таблицы NAT пользовательск им приложением	Значение поля «Локальны й порт» = «АБВГД»	Задать значение поля «Локальный порт» = «АБВГД»; Нажать кнопку «Создать»	Программа должна сообщить о некорректно введённых данных в поле «Локальный порт»	Совпадает с ожидаемым
ПФ 1.8	Проверить корректность обработки корректно	Значение поля «Глобальн	Задать значение поля «Глобальный порт» = «123»;	Программа не должна сообщать о некорректно	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT

Документ: ВКР, глава 1, пункт 3.1.1

№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	заданной информации в поле «Глобальный порт» при создании элемента таблицы NAT пользовательским приложением	ый порт» = «123»	Нажать кнопку «Создать»	введённых данных в поле «Глобальный порт»	
ПФ 1.9	Проверить корректность обработки некорректно заданной информации в поле «Глобальный порт» при создании элемента таблицы NAT пользовательским приложением	Значение поля «Глобальный порт» = «АБВГД»	Задать значение поля «Глобальный порт» = «АБВГД»; Нажать кнопку «Создать»	Программа должна сообщить о некорректно введённых данных в поле «Глобальный порт»	Совпадает с ожидаемым
ПФ 1.10	Проверить корректность работы	Выборанный в таблице элемент	Выбрать в таблице элемент;	Программа должна перевести или	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT

Документ: ВКР, глава 1, пункт 3.1.1

№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	функции приостановки модификации данных в соответствии с выбранным элементом таблицы NAT пользовательск им приложением		Нажать кнопку «Приостанови ть NAT- интерфейс»	оставить статус выбранного элемента в «Приостановле н»	
ПФ 1.11	Проверить корректность работы функции возобновления модификации данных в соответствии с выбранным элементом таблицы NAT пользовательск им приложением	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Возобновить NAT- интерфейс»	Программа должна перевести или оставить статус выбранного элемента в «Работает»	Совпадает с ожидаемым
ПФ 1.12	Проверить корректность работы функции	Выбранный в таблице элемент	Выбрать в таблице элемент;	Программа должна удалить выбранный	Совпадает с ожидаемым

Проверяется: Функционал пользовательского приложения NAT					
Документ: ВКР, глава 1, пункт 3.1.1					
№ тест а	Цель теста	Заданная информаци я	Способ выполнения	Ожидаемый результат	Фактическ ий результат
	удаления элемента таблицы NAT пользовательск им приложением		Нажать кнопку «Удалить NAT- интерфейс»	элемент из таблицы	

Исходя из полученных результатов, можно сделать вывод о том, что пользовательское приложение работает корректно и отвечает указанным требованиям. Далее на рисунке 28 и рисунке 29 представлены снимки экрана, подтверждающие корректное выполнение теста ПФ 1.1.

Локальный IP: 1.1.1.1

Глобальный IP: 2.2.2.2

Локальный порт: 1

Глобальный порт: 2

Создать

Рисунок 28 - ПФ 1.1 - Указание входных данных



Рисунок 29 - ПФ 1.1 - Результат выполнения

На рисунке 30 и рисунке 31 представлены снимки экрана, подтверждающие корректное выполнение теста ПФ 1.3.

The screenshot shows a configuration form with four input fields and a button. The first field, labeled "Локальный IP", contains the text "АБВГД". The other fields are labeled "Глобальный IP", "Локальный порт", and "Глобальный порт". A "Создать" (Create) button is at the bottom.

Рисунок 30 - ПФ 1.3 - Указание входных данных

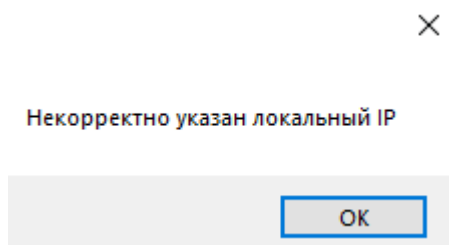


Рисунок 31 - ПФ 1.3 - Результат выполнения

На рисунке 32 и рисунке 33 представлены снимки экрана, подтверждающие корректное выполнение теста ПФ 1.5.

Рисунок 32 - ПФ 1.5 - Указание входных данных

Рисунок 33 - ПФ 1.5 - Результат выполнения

На рисунке 34 и рисунке 35 представлены снимки экрана, подтверждающие корректное выполнение теста ПФ 1.7.

Рисунок 34 - ПФ 1.7 - Указание входных данных

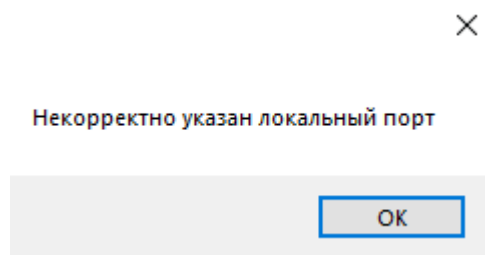


Рисунок 35 - ПФ 1.7 - Результат выполнения

На рисунке 36 и рисунке 37 представлены снимки экрана, подтверждающие корректное выполнение теста ПФ 1.9.

Локальный IP	1.1.1.1
Глобальный IP	1.1.1.1
Локальный порт	1
Глобальный порт	АБВГД
Создать	

Рисунок 36 - ПФ 1.9 - Указание входных данных

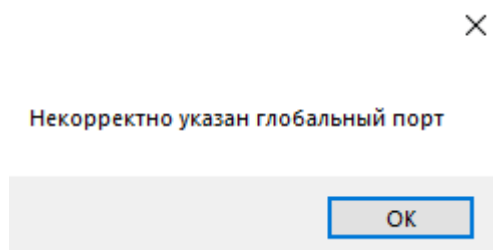


Рисунок 37 - ПФ 1.9 - Результат выполнения

1.3.1.2 Тестирование библиотеки для взаимодействия пользовательского приложения и драйвера-фильтра

В приведённой далее таблице 2 описан блок тестов, проведённых с библиотекой для взаимодействия пользовательского приложения и драйвера-фильтра. В таблице описано: цель теста, заданная информация, способ выполнения теста, ожидаемый и фактический результат тестирования.

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

Таблица 2 - Тестирование библиотеки для взаимодействия
пользовательского приложения и драйвера-фильтра

Проверяется: Функционал библиотеки для взаимодействия пользовательского приложения и драйвера-фильтра NAT Документ: ВКР, глава 1, пункт 3.1.2					
№ теста	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактический результат
ПФ 2.1	Проверит реакцию драйвера на создание элемента таблицы NAT пользовательским приложением	Значения полей: «Локальный IP-адрес» = «1.1.1.1»; «Глобальный IP-адрес» = «2.2.2.2»; «Локальный порт» = «1»; «Глобальный порт» = «2»;	Нажать кнопку «Создать NAT-интерфейс»; Задать указанные значения полей; Нажать кнопку «Создать»	В WinDbg должно появиться сообщение о успешном выполнении IOCTL запроса по созданию элемента	Совпадает с ожидаемым
ПФ 2.2	Проверит реакцию драйвера на приостановление модификации данных в соответствии с выбранным элементом таблицы NAT пользовательским приложением	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Приостановить NAT-интерфейс»	В WinDbg должно появиться сообщение о успешном выполнении IOCTL запроса по приостановке модификации	Совпадает с ожидаемым

Проверяется: Функционал библиотеки для взаимодействия пользовательского приложения и драйвера-фильтра NAT Документ: ВКР, глава 1, пункт 3.1.2					
№ теста	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактический результат
	им приложением				
ПФ 2.3	Проверить реакцию драйвера на возобновление модификации данных в соответствии с выбранным элементом таблицы NAT пользовательским приложением	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Возобновить NAT-интерфейс»	В WinDbg должно появиться сообщение о успешном выполнении IOCTL запроса по возобновлению модификации	Совпадает с ожидаемым
ПФ 2.4	Проверить реакцию драйвера на удаление выбранного элемента таблицы NAT пользовательским приложением	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Удалить NAT-интерфейс»	В WinDbg должно появиться сообщение о выполнении IOCTL запроса по удалению элемента	Совпадает с ожидаемым

Исходя из полученных результатов, можно сделать вывод о том, что библиотека для взаимодействия пользовательского приложения и драйвера-фильтра работает корректно и отвечает указанным требованиям. Далее на

рисунке 38 представлен снимок экрана, подтверждающий корректное выполнение теста ПФ 2.1.

```
DeviceControl: Beginning
IoAddNatTableUnit: Beginning
IoAddNatTableUnit: Status = 0
DeviceControl: Status = 0
```

Рисунок 38 - ПФ 2.1 - Результат выполнения

На рисунке 39 представлен снимок экрана, подтверждающий корректное выполнение теста ПФ 2.2.

```
DeviceControl: Beginning
IoSuspendNatTableUnitMasking: Beginning
IoSuspendNatTableUnitMasking: Status = 0
DeviceControl: Status = 0
```

Рисунок 39 - ПФ 2.2 - Результат выполнения

На рисунке 40 представлен снимок экрана, подтверждающий корректное выполнение теста ПФ 2.3.

```
DeviceControl: Beginning
IoResumeNatTableUnitMasking: Beginning
IoResumeNatTableUnitMasking: Status = 0
DeviceControl: Status = 0
```

Рисунок 40 - ПФ 2.3 - Результат выполнения

На рисунке 41 представлен снимок экрана, подтверждающий корректное выполнение теста ПФ 2.4.

```
DeviceControl: Beginning
IoRemoveNatTableUnit: Beginning
IoRemoveNatTableUnit: Status = 0
DeviceControl: Status = 0
```

Рисунок 41 - ПФ 2.4 - Результат выполнения

1.3.1.3 Тестирование драйвера-фильтра NAT

В приведённой далее таблице 3 описан блок тестов, проведённых с драйвером-фильтром NAT. В таблице описано: цель теста, заданная информация, способ выполнения теста, ожидаемый и фактический результат тестирования.

Таблица 3 - Тестирование драйвера-фильтра NAT

Проверяется: Функционал драйвера-фильтра NAT					
Документ: ВКР, глава 1, пункт 3.1.3					
№ теста	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактически й результат
ПФ 3.1	Проверить корректность работы функции создания элемента таблицы NAT драйвером-фильтром	Значения полей: «Локальный IP-адрес» = «192.168.11.13»; «Глобальный IP-адрес» = «1.3.3.7»; «Локальный порт» = «1025»; «Глобальный порт» = «1337»;	Нажать кнопку «Создать NAT-интерфейс»; Задать указанные значения полей; Нажать кнопку «Создать»	В WinDbg должно появиться сообщение о успешном выполнении IOCTL запроса по созданию элемента; Должна начаться модификация трафика	Совпадает с ожидаемым
ПФ 3.2	Проверить корректность работы функции приостановки и модификации данных в соответствии с выбранным элементом таблицы	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Приостановить NAT-интерфейс»	В WinDbg должно появиться сообщение о успешном выполнении IOCTL запроса по приостановке модификации; Должна прекратиться	Совпадает с ожидаемым

Проверяется: Функционал драйвера-фильтра *NAT*

Документ: ВКР, глава 1, пункт 3.1.3

№ теста	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактический результат
	<i>NAT</i> драйвером-фильтром			модификация трафика	
ПФ 3.3	Проверить корректность работы функции возобновления модификации данных в соответствии с выбранным элементом таблицы <i>NAT</i> драйвером-фильтром	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Возобновить <i>NAT</i> -интерфейс»	В <i>WinDbg</i> должно появиться сообщение о успешном выполнении <i>IOCTL</i> запроса по возобновлению модификации; Должна возобновиться модификация трафика	Совпадает с ожидаемым
ПФ 3.4	Проверить корректность работы функции удаления элемента таблицы <i>NAT</i> драйвером-фильтром	Выбранный в таблице элемент	Выбрать в таблице элемент; Нажать кнопку «Удалить <i>NAT</i> -интерфейс»	В <i>WinDbg</i> должно появиться сообщение о выполнении <i>IOCTL</i> запроса по удалению элемента; Должна начаться	Совпадает с ожидаемым

Проверяется: Функционал драйвера-фильтра NAT					
Документ: ВКР, глава 1, пункт 3.1.3					
№ теста	Цель теста	Заданная информация	Способ выполнения	Ожидаемый результат	Фактически й результат
				прекратиться трафика	

Исходя из полученных результатов, можно сделать вывод о том, что драйвер-фильтр работает корректно и отвечает указанным требованиям. Далее на рисунке 42, рисунке 43, рисунке 44 и рисунке 45 представлены снимки экрана, которые подтверждают корректное выполнение проведённых тестов.

No.	Time	Source	Destination	Protocol	Length	Info
103	137.620723	fe80::cc3e:241b:385...	ff02::1:ffc0:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
104	138.619994	fe80::cc3e:241b:385...	ff02::1:ffc0:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
105	140.702955	192.168.11.13	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=3328/13, ttl=128 (r
106	140.703379	Vmware_f4:b2:6a	Broadcast	ARP	42	Who has 192.168.11.13? Tell 192.168.11.10
107	140.703551	Vmware_07:16:7b	Vmware_f4:b2:6a	ARP	42	192.168.11.13 is at 00:0c:29:07:16:7b
108	140.703561	3.1.43.9	192.168.11.13	ICMP	74	Echo (ping) reply id=0x0000, seq=3328/13, ttl=128 (r
109	141.695203	192.168.11.13	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=3584/14, ttl=128 (r
110	141.695773	3.1.43.9	192.168.11.13	ICMP	74	Echo (ping) reply id=0x0000, seq=3584/14, ttl=128 (r
111	142.696186	192.168.11.13	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=3840/15, ttl=128 (r
112	142.696651	3.1.43.9	192.168.11.13	ICMP	74	Echo (ping) reply id=0x0000, seq=3840/15, ttl=128 (r
113	143.698898	192.168.11.13	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=4096/16, ttl=128 (r
114	143.699674	3.1.43.9	192.168.11.13	ICMP	74	Echo (ping) reply id=0x0000, seq=4096/16, ttl=128 (r
115	148.625929	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
116	149.619707	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
117	150.008765	fe80::250:56ff:fec0...	ff02::1	ICMPv6	102	Router Advertisement
118	150.619196	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1

Рисунок 42 – ICMP трафик до активации модификации трафика драйвером

No.	Time	Source	Destination	Protocol	Length	Info
117	150.008765	fe80::250:56ff:fec0...	ff02::1	ICMPv6	102	Router Advertisement
118	150.619196	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
119	163.626440	fe80::cc3e:241b:385...	ff02::1:ffc0:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
120	164.620734	fe80::cc3e:241b:385...	ff02::1:ffc0:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
121	165.621700	fe80::cc3e:241b:385...	ff02::1:ffc0:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
122	170.256874	1.3.3.7	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=4352/17, ttl=128 (r
123	170.257278	3.1.43.9	1.3.3.7	ICMP	74	Echo (ping) reply id=0x0000, seq=4352/17, ttl=128 (r
124	171.257330	1.3.3.7	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=4608/18, ttl=128 (r
125	171.258332	3.1.43.9	1.3.3.7	ICMP	74	Echo (ping) reply id=0x0000, seq=4608/18, ttl=128 (r
126	172.257938	1.3.3.7	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=4864/19, ttl=128 (r
127	172.258339	3.1.43.9	1.3.3.7	ICMP	74	Echo (ping) reply id=0x0000, seq=4864/19, ttl=128 (r
128	173.267714	1.3.3.7	3.1.43.9	ICMP	74	Echo (ping) request id=0x0000, seq=5120/20, ttl=128 (r
129	173.268107	3.1.43.9	1.3.3.7	ICMP	74	Echo (ping) reply id=0x0000, seq=5120/20, ttl=128 (r
130	178.626835	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
131	179.619574	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
132	180.010546	fe80::250:56ff:fec0...	ff02::1	ICMPv6	102	Router Advertisement

Рисунок 43 - Модификация драйвером IP-адреса на примере ICMP трафика

Capturing from VMware Network Adapter VMnet8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
452	370.987951	3.1.43.9	1.3.3.7	SMB	93	Tree Disconnect Response
453	370.988181	1.3.3.7	3.1.43.9	TCP	54	1025 → 445 [ACK] Seq=870 Ack=1215 Win=1023 Len=0
454	370.988186	1.3.3.7	3.1.43.9	TCP	54	[TCP Dup ACK 453#1] 1025 → 445 [ACK] Seq=870 Ack=1215 Win=1023 Len=0
455	370.988249	1.3.3.7	3.1.43.9	TCP	54	[TCP Window Update] 1025 → 445 [ACK] Seq=870 Ack=1215 Win=1023 Len=0
456	370.988355	1.3.3.7	3.1.43.9	SMB	97	Logoff AndX Request
457	370.988397	1.3.3.7	1.3.3.7	TCP	54	445 → 1025 [ACK] Seq=1215 Ack=913 Win=64240 Len=0
458	370.988687	3.1.43.9	1.3.3.7	SMB	97	Logoff AndX Response
459	370.988860	1.3.3.7	3.1.43.9	TCP	54	1025 → 445 [ACK] Seq=913 Ack=1258 Win=1023 Len=0
460	370.988866	1.3.3.7	3.1.43.9	TCP	54	[TCP Dup ACK 459#1] 1025 → 445 [ACK] Seq=913 Ack=1258 Win=1023 Len=0
461	370.988905	1.3.3.7	3.1.43.9	TCP	54	[TCP Window Update] 1025 → 445 [ACK] Seq=913 Ack=1258 Win=1023 Len=0
462	370.988909	1.3.3.7	3.1.43.9	TCP	54	1025 → 445 [FIN, ACK] Seq=913 Ack=1258 Win=1024 Len=0
463	370.988950	3.1.43.9	1.3.3.7	TCP	54	445 → 1025 [ACK] Seq=1258 Ack=914 Win=64239 Len=0
464	370.989357	3.1.43.9	1.3.3.7	TCP	54	445 → 1025 [RST, ACK] Seq=1258 Ack=914 Win=64239 Len=0
465	373.624840	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
466	374.621377	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1
467	375.619127	Vmware_c0:00:08	Broadcast	ARP	42	Who has 3.1.34.10? Tell 192.168.11.1

Рисунок 44 - Модификация драйвером IP-адреса на примере TCP трафика

Capturing from VMware Network Adapter VMnet8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
605	467.443871	3.1.43.9	1.3.3.7	TCP	54	445 → 1337 [ACK] Seq=782 Ack=482 Win=64240 Len=0
606	467.444139	3.1.43.9	1.3.3.7	SMB	93	Tree Connect AndX Response, Error: STATUS_ACCESS_DENIED
607	467.444325	1.3.3.7	3.1.43.9	TCP	54	1337 → 445 [ACK] Seq=482 Ack=821 Win=1023 Len=0
608	467.444329	1.3.3.7	3.1.43.9	TCP	54	[TCP Dup ACK 607#1] 1337 → 445 [ACK] Seq=482 Ack=821 Win=1023 Len=0
609	467.444408	1.3.3.7	3.1.43.9	TCP	54	[TCP Window Update] 1337 → 445 [ACK] Seq=482 Ack=821 Win=1023 Len=0
610	467.444549	1.3.3.7	3.1.43.9	SMB	97	Logoff AndX Request
611	467.444586	3.1.43.9	1.3.3.7	TCP	54	445 → 1337 [ACK] Seq=821 Ack=525 Win=64240 Len=0
612	467.444937	3.1.43.9	1.3.3.7	SMB	97	Logoff AndX Response
613	467.445107	1.3.3.7	3.1.43.9	TCP	54	1337 → 445 [ACK] Seq=525 Ack=864 Win=1023 Len=0
614	467.445112	1.3.3.7	3.1.43.9	TCP	54	[TCP Dup ACK 613#1] 1337 → 445 [ACK] Seq=525 Ack=864 Win=1023 Len=0
615	467.445179	1.3.3.7	3.1.43.9	TCP	54	[TCP Window Update] 1337 → 445 [ACK] Seq=525 Ack=864 Win=1023 Len=0
616	467.445282	1.3.3.7	3.1.43.9	TCP	54	1337 → 445 [FIN, ACK] Seq=525 Ack=864 Win=1024 Len=0
617	467.445342	3.1.43.9	1.3.3.7	TCP	54	445 → 1337 [ACK] Seq=864 Ack=526 Win=64239 Len=0
618	467.445590	3.1.43.9	1.3.3.7	TCP	54	445 → 1337 [RST, ACK] Seq=864 Ack=526 Win=64239 Len=0
619	467.620515	fe80::cc3e:241b:385...	ff02::1:ff00:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from
620	468.620735	fe80::cc3e:241b:385...	ff02::1:ff00:2222	ICMPv6	86	Neighbor Solicitation for fe80::250:56ff:fec0:2222 from

Рисунок 45 - Модификация драйвером IP-адреса и порта на примере TCP трафика

2 Безопасность жизнедеятельности

2.1 Введение в безопасность жизнедеятельности при работе с ЭВМ

Вычислительная техника всё стремительнее внедряется во все сферы нашей жизни. Экономические или научные достижения не представляются возможными без грамотно подобранного оборудования и специально обученного персонала - главной причиной этому является существенное ускорение оперирования информацией и повышение общей производительности при помощи разного рода устройств в различных сферах деятельности.

Образ жизни, который приходится принять каждому профессионалу, так или иначе задействованному в работе с ЭВМ, существенно влияет на здоровье человека – и это влияние далеко не безобидно. Чтобы минимизировать вред, пользователю ПК необходимо соблюдать соответствующие меры безопасности, такие как грамотно оборудованное рабочее место, грамотно спланированный и соответствующий медицинским рекомендациям режим работы, а также организация соответствующего всем нормам рабочего места.

В данной главе будут рассмотрены негативные факторы, которые возникают при несоблюдении норм безопасности работы с ПК, требования по организации рабочего места для пользователей вычислительных машин, а также способы минимизации воздействия различных вредных факторов на здоровье человека.

2.2 Характеристика потенциальных опасностей и вредностей

Затрагивая тему характеристики потенциальных опасностей и вредностей, сперва стоит отметить, что идентификация потенциальных опасностей и вредностей производится экспертом организации,

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

осуществляющей оценку условий труда. Далее результаты идентификации должны быть утверждены комиссией.

Итак, потенциальными опасностями и вредностями считается негативное воздействие на человека, потенциально способное привести к травме или летальному исходу. Под данное определение попадают следующие факторы:

- использование неисправного оборудования, повышенный уровень напряжения в электрической сети;
- повышенная/пониженная влажность воздуха рабочей зоны/подвижность воздуха рабочей зоны;
- длительное пребывание в одном и том же положении или пребывание в некорректной позе;
- монотонность и психофизиологическая напряжённость труда;
- большой объём перерабатываемой информации и нервно-психические нагрузки;
- недостаток естественного освещения или освещённости рабочей зоны.

2.3 Электробезопасность

По классификации уровней электробезопасности помещения делятся на безопасные, помещения с повышенной опасностью и особо опасные.

Для определения условий, которые влияют на вероятность поражения человека электрическим током, нужно определить критерии, которые выражают степень опасности помещения.

В помещениях, которые попадают в категорию безопасных, вероятность поражения человека током должна быть сведена к минимуму. Данная категория соответствует следующим критериям:

- отсутствие токопроводящей пыли;
- средняя температура 18°C до 20°C;

- влажность воздуха не превышает 60%;
- площадь помещения заполнена металлическими элементами не более чем 20%;
- полы помещения не должны быть металлическими.

В помещениях, которые попадают в категорию повышенной опасности, вероятность поражения человека током выше, чем в безопасных помещениях. Данная категория соответствует следующим критериям:

- средняя температура воздуха превышает 30°C;
- наличие токопроводящей пыли;
- площадь заполнена большим количеством металлических элементов;
- полы помещения являются металлическими;
- повышенная влажность воздуха.

В помещениях, которые попадают в категорию особо опасных, вероятность поражения человека током высокая. Данная категория соответствует следующим критериям:

- средняя температура воздуха превышает 35°C;
- наличие в помещении легковоспламеняющихся веществ или материалов;
- присутствие едких паров или газов в помещении;
- влажность воздуха 100%.

2.4 Пожарная безопасность

Работа с электрическим оборудованием всегда сопровождается повышенной пожарной опасностью.

По классификации уровней пожарной безопасности помещения делятся на 5 категорий:

- группа А – самая высокая степень пожарной опасности. Для помещений, попадающих в данную категорию допустимо хранение

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

горючих газов и материалов, возгорающихся при контакте с водой, кислородом или под высоким давлением;

- группа Б – высокая степень пожарной опасности. Для данных помещений допустимо хранение горючих волокон, материалов с выделением горючей пыли;

- группа В – стандартная степень пожарной опасности. Допустимо хранение твёрдых горючих материалов, пыли и волокон, а также горючих и трудногорючих жидкостей;

- группа Г – умеренная степень пожарной опасности. Допустимо хранение негорючих веществ и материалов и горючих газов, жидкостей и твёрдых материалов, используемых в качестве топлива;

- группа Д – степень условной пожарной опасности. Здесь хранятся негорючие вещества и материалы в холодном состоянии.

2.5 Санитарно-гигиеническая характеристика производства

В настоящее время при работе на вычислительных устройствах следует руководствоваться СанПиН 2.2.4.335916 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах».

Исходя из данного документа, руководство персонала должно соблюдать следующие нормы:

- норма площади на одно рабочее место, оснащенное плоским экраном – 4.5 квадратных метра;

- оконные проёмы должны быть оборудованы регулируемыми устройствами типа жалюзи, занавесей и т.д;

- оптимальная температура должна составлять от 21 до 25°C, а относительная влажность воздуха – от 40 до 60 % при его движении в 0,1-0,2 м/с;

- уровень шума на рабочем месте не должен превышать 50 ДБА;

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

- помещения, где размещены ПК, должны быть оборудованы заземлением;
- осветительная аппаратура не должна создавать блики на поверхности экрана;
- между столами, на которых располагаются мониторы, должно быть расстояние не меньше двух метров;
- конструкция рабочего кресла должна обеспечивать поддержание рациональной рабочей позы при работе на ПК;
- рабочее кресло должно быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки.

2.6 Расчётно-аналитическая часть

Максимальная работоспособность возникает у человека только в том случае, если условия его труда полностью комфортны для его организма. В случае плохой освещённости возникает головная боль и усталость в глазах.

Так как санитарные и гигиенические нормы разрабатывались годами и не без причины, лучшее, что может сделать работодатель – соблюдать рекомендации СанПин и СНиП.

Далее приводится расчёт освещённости рабочего места в соответствии с рекомендациями санитарной и гигиенической безопасности.

Условия расчёта: комната длиной 5 метров, шириной 3 метра, высотой 3 метра и высотой подвеса в 2.92 метра. На потолок и стены нанесено матовое белое покрытие, обладающее коэффициентом отражения в 80%. Пол, имеющий серое матовое покрытие, имеет индекс отражения в 30%. Используется светильник с лампами ЛБ40-1.

Рассчитаем световой поток по формуле, представленной ниже:

$$F = \frac{E * K * S * Z}{n}, \quad (1)$$

где F – световой поток в лм;

E – минимальная освещённость для работы на ПК, которая в данном случае равна 300 лк;

S – площадь комнаты – 15 метра квадратных;

Z – коэффициент отклонения – 1.1;

K – коэффициент запаса на уменьшение светового потока, условно равный 1.5.

Далее необходимо определить индекс помещения. Расчёт индекса помещения происходит по следующей формуле:

$$i = \frac{A * B}{Hc * (A + B)}, \quad (2)$$

Подставив значения, получаем:

$$i = \frac{3 * 5}{2.92 * (3 + 5)} = 0.64$$

В соответствии с результатом, уточняем, что коэффициент светового потока n равен 0.53. Вычисление с подстановкой получившихся значений по формуле для расчёта светового потока:

$$F = \frac{300 * 1.5 * 15 * 1.1}{0.53} = 14009$$

Так как световой поток одной лампы ЛБ40-1 равен 4320лм, для помещения с площадью 15 квадратных метров требуется $14009/4320 = 4$ лампы.

Заключение

В рамках данной выпускной квалификационной работы было разработано программное обеспечения для преобразования сетевых адресов с типом преобразования *PAT*, также известное как *NAT Overload*.

В процессе разработки был создан драйвер-фильтр *NDIS* и библиотека, осуществляющее взаимодействие между драйвером-фильтром и пользовательским интерфейсов, с помощью языка программирования Си, а также пользовательское приложение, разработанное на языке *C#* на платформе *Windows Forms*, предоставляющее интуитивно понятный интерфейс для управления драйвером.

Таким образом задание ВКР выполнено в полном объеме.

В ходе выполнения данной работы был получен опыт разработки системного ПО, такое как драйверы для ОС *MS Windows* и *QP ОС*, а также опыт создания пользовательских приложений на платформе *Windows Forms*. Была изучена документация о сети Интернет и азах устройства сетей *TCP/IP*.

В дальнейшем в разработанное ПО можно добавить функции хранения таблицы *NAT* в памяти для её многократного использования и реализовать поддержку задания диапазонов *IP*-адресов и сетевых портов

Список используемых источников

1. Общая информация о сети Интернет [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Main_Page.
2. Overview of NDIS driver types [Электронный ресурс]. URL: <https://docs.microsoft.com/en-us/windowshardware/drivers/network/ndis-drivers>.
3. Требования к освещению помещений и рабочих мест [Электронный ресурс]. URL: <https://lampaexpert.ru/osveschenie/osveschennost-rabochikh-mest-normy-osveschennosti>.

Приложение А

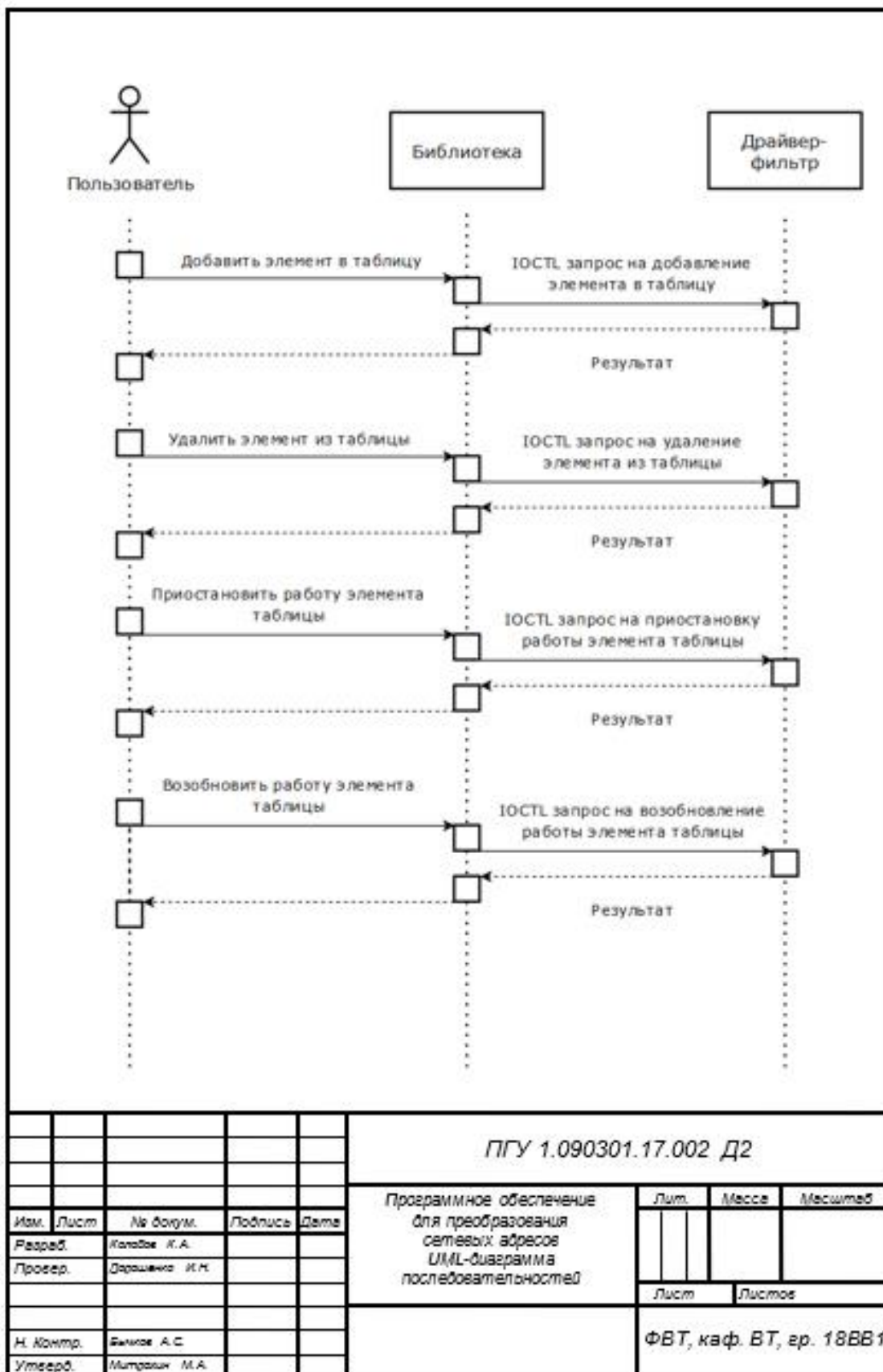
(Обязательное)

Программное обеспечение для преобразования сетевых адресов

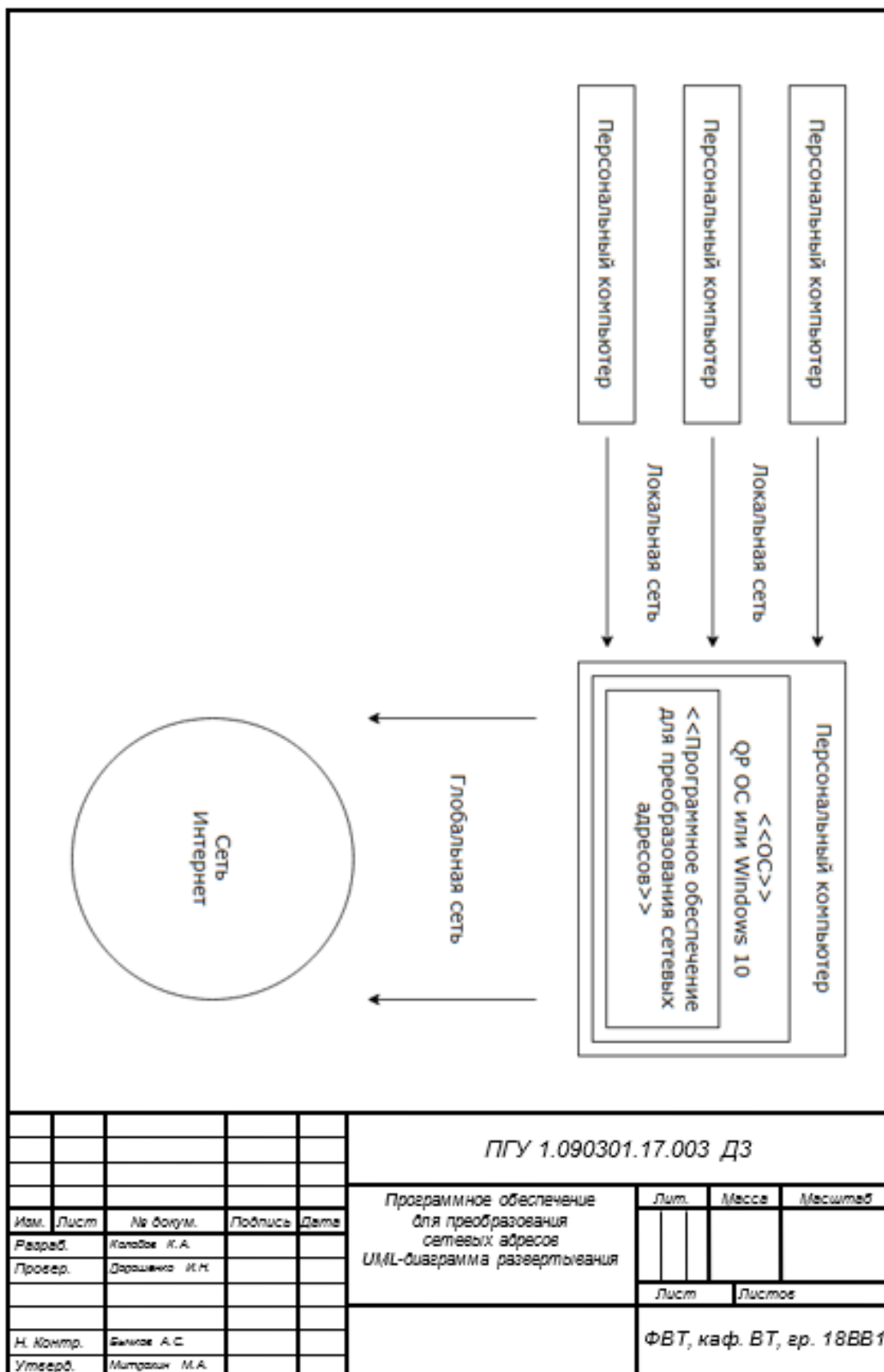
UML-диаграммы

					ПГУ 1.090301.17.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

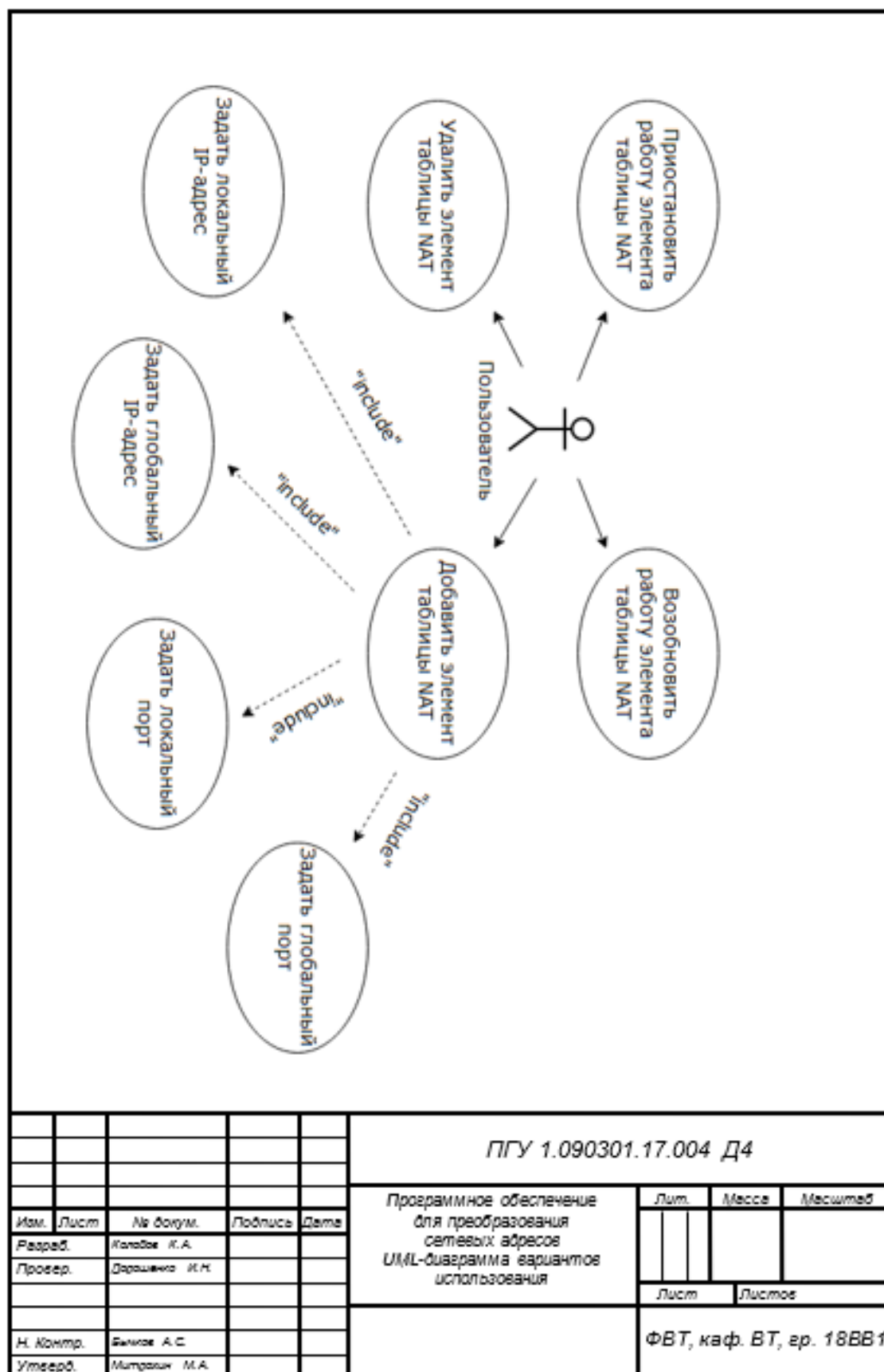
Приложение А.1 – Диаграмма последовательностей



Приложение А.2 – Диаграмма развёртывания



Приложение А.3 – Диаграмма вариантов использования



Приложение Б

(Обязательное)

Программное обеспечение для преобразования сетевых адресов

Листинг программы

					ПГУ 1.090301.17.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б.1 – Драйвер -фильтр NDIS

Файл Device.c

```
#include "StdAfx.c"

/*
 * Создание объекта устройства
 *
 * @param    Нет заданных параметров
 * @return    Статус
 */
NDIS_STATUS
RegisterDevice
(
    VOID
)
{
    DbgPrintEx(0, 0, "RegisterDevice: Beginning\n");

    PSFilterDeviceExtension
pstDeviceExtension;
    PDRIVER_OBJECT                pDriverObject;
    NDIS_DEVICE_OBJECT_ATTRIBUTES DeviceAttribute;
    UNICODE_STRING                sDeviceName;
    UNICODE_STRING                sDeviceLinkUnicodeString;

    NDIS_STATUS                    Status =
NDIS_STATUS_SUCCESS;

    /*Инициализация имён устройства*/
    NdisInitUnicodeString(&sDeviceName, NAT_DRIVER_NAME);
    NdisInitUnicodeString(&sDeviceLinkUnicodeString,
DOS_NAT_DRIVER_NAME);

    /*Таблица обработки IRP*/
    PDRIVER_DISPATCH
pDispatchTable[IRP_MJ_MAXIMUM_FUNCTION + 1];

    /*Инициализация и заполнение таблицы обработки IRP*/
    NdisZeroMemory(pDispatchTable, (IRP_MJ_MAXIMUM_FUNCTION +
1) * sizeof(PDRIVER_DISPATCH));

    pDispatchTable[IRP_MJ_CREATE]                = DeviceDispatch;
    pDispatchTable[IRP_MJ_CLOSE]                  = DeviceDispatch;
    pDispatchTable[IRP_MJ_CLEANUP]                = DeviceDispatch;
    pDispatchTable[IRP_MJ_DEVICE_CONTROL]        = DeviceIoControl;

    /*Инициализация и заполнение структуры атрибутов
устройства*/
```

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

```

    NdisZeroMemory(&DeviceAttribute,
sizeof(NDIS_DEVICE_OBJECT_ATTRIBUTES));

    DeviceAttribute.Header.Type                =
NDIS_OBJECT_TYPE_DEVICE_OBJECT_ATTRIBUTES;
    DeviceAttribute.Header.Revision           =
NDIS_DEVICE_OBJECT_ATTRIBUTES_REVISION_1;
    DeviceAttribute.Header.Size                =
sizeof(NDIS_DEVICE_OBJECT_ATTRIBUTES);

    DeviceAttribute.DeviceName                 = &sDeviceName;
    DeviceAttribute.SymbolicName               =
&sDeviceLinkUnicodeString;
    DeviceAttribute.MajorFunctions             =
&pDispatchTable[0];
    DeviceAttribute.ExtensionSize              =
sizeof(SFilterDeviceExtension);

    /*Создание объекта устройства*/
    Status = NdisRegisterDeviceEx
    (
        g_FilterDriverHandle,
        &DeviceAttribute,
        &g_pNdisDeviceObject,
        &g_NdisFilterDeviceHandle
    );

    if (Status == NDIS_STATUS_SUCCESS)
    {
        pstDeviceExtension =
(PSFilterDeviceExtension)NdisGetDeviceReservedExtension(g_pNdi
sDeviceObject);

        pstDeviceExtension->m_ulSignature      = 'FTDR';
        pstDeviceExtension->m_Handle          =
g_FilterDriverHandle;

        pDriverObject =
(PDRIVER_OBJECT)g_FilterDriverObject;
    }

    DbgPrintEx(0, 0, "RegisterDevice: Status = %8x\n",
Status);

    return Status;
}

/*
* Удаление объекта устройства
*
* @param    Нет заданных параметров
* @return    Нет возвращаемых значений

```

```

*/
VOID
DeregisterDevice
(
    VOID
)
{
    DbgPrintEx(0, 0, "DeregisterDevice: Beginning\n");

    if (g_NdisFilterDeviceHandle != NULL)
    {
        NdisDeregisterDeviceEx(g_NdisFilterDeviceHandle);
        DbgPrintEx(0, 0, "DeregisterDevice: Device
Deregistered\n");
    }

    g_NdisFilterDeviceHandle = NULL;

    DbgPrintEx(0, 0, "DeregisterDevice: Success\n");
}

/*
* Обработка запросов к устройству
*
* @param[in]    pDeviceObject    Структура с информацией об
устройстве
* @param[in]    pIrp             Заголовок IRP запроса
* @return       Статус
*/
NTSTATUS
DeviceDispatch
(
    PDEVICE_OBJECT    pDeviceObject,
    PIRP              pIrp
)
{
    PIO_STACK_LOCATION    pIrpStack;
    NTSTATUS               Status = STATUS_SUCCESS;

    UNREFERENCED_PARAMETER(pDeviceObject);

    pIrpStack = IoGetCurrentIrpStackLocation(pIrp);

    switch (pIrpStack->MajorFunction)
    {
    case IRP_MJ_CREATE:
        break;

    case IRP_MJ_CLEANUP:
        break;

    case IRP_MJ_CLOSE:

```

```

        break;

default:
    break;
}

pIrp->IoStatus.Status = Status;
IoCompleteRequest(pIrp, IO_NO_INCREMENT);

return Status;
}

/*
 * Обработка IOCTL запросов к устройству
 *
 * @param[in]    pDeviceObject    Структура с информацией об
устройстве
 * @param[in]    pIrp            Заголовок IRP запроса
 * @return                               Статус
 */
NTSTATUS
DeviceIoControl
(
    PDEVICE_OBJECT    pDeviceObject,
    PIRP              pIrp
)
{
    DbgPrintEx(0, 0, "DeviceIoControl: Beginning\n");

    PIO_STACK_LOCATION    pIrpSp;
    PSFilterDeviceExtension    pstDeviceExtension;
    NTSTATUS                Status = STATUS_SUCCESS;

    do
    {
        pIrpSp = IoGetCurrentIrpStackLocation(pIrp);
        if (pIrpSp->FileObject == NULL)
        {
            Status = STATUS_UNSUCCESSFUL;
            break;
        }

        pstDeviceExtension =
        (PSFilterDeviceExtension)NdisGetDeviceReservedExtension(pDeviceObject);
        ASSERT(pstDeviceExtension->m_ulSignature == 'FTDR');

        switch (pIrpSp->Parameters.DeviceIoControl.IoControlCode)
        {

            case IOCTL_NAT_CREATE_UNIT:

```

```

        Status = IoAddNatTableUnit(pIrp, pIrpSp);
        break;

case IOCTL_NAT_REMOVE_UNIT:

    Status = IoRemoveNatTableUnit(pIrp, pIrpSp);
    break;

case IOCTL_NAT_WIPE_UNIT_LIST:

    Status = IoWipeNatTableUnitList(pIrp, pIrpSp);
    break;

case IOCTL_NAT_IS_UNIT_EXISTS:

    Status = IoIsNatTableUnitExists(pIrp, pIrpSp);
    break;

case IOCTL_NAT_IS_UNIT_RUNNING:

    Status = IoIsNatTableUnitMasking(pIrp, pIrpSp);
    break;

case IOCTL_NAT_SUSPEND_UNIT:

    Status = IoSuspendNatTableUnitMasking(pIrp,
pIrpSp);
    break;

case IOCTL_NAT_RESUME_UNIT:

    Status = IoResumeNatTableUnitMasking(pIrp,
pIrpSp);
    break;

case IOCTL_NAT_IS_DRIVER_MASKING:

    Status = IoIsNatTableMasking();
    break;

case IOCTL_NAT_DRIVER_START_MASKING:

    Status = IoSuspendNatTableMasking();
    break;

case IOCTL_NAT_DRIVER_STOP_MASKING:

    Status = IoResumeNatTableMasking();
    break;

default:

```

```

        break;
    }

    pIrp->IoStatus.Status      = Status;
    pIrp->IoStatus.Information = 0;

    IoCompleteRequest(pIrp, IO_NO_INCREMENT);

} while (FALSE);

DbgPrintEx(0, 0, "DeviceIoControl: Status = %8x\n",
Status);

return Status;
}

/*
* Добавление нового интерфейса в список NAT
*
* @param[in]    pIrp          Заголовок IRP запроса
* @param[in]    pIrpSp       I/O stack location IRP запроса
* @return       Статус
*/
NTSTATUS
IoAddNatTableUnit
(
    PIRP                pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoAddNatTableUnit: Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS    Status      = STATUS_UNSUCCESSFUL;
    PBYTE       pcInBuffer = pIrp->
>AssociatedIrp.SystemBuffer;

    if
    (
        AddNatTableUnit
        (
            *(PDWORD) (pcInBuffer),          //Local IP      -
DWORD
            *(PDWORD) (pcInBuffer + 4),      //Global IP      -
DWORD
            *(PWORD) (pcInBuffer + 8),        //Local port     -
WORD
            *(PWORD) (pcInBuffer + 10)       //Global port     -
WORD
        )
    )

```

```

    )
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoAddNatTableUnit: Status = %8x\n",
Status);

    return Status;
}

/*
* Удаление интерфейса из списка NAT
*
* @param[in]    pIrp            Заголовок IRP запроса
* @param[in]    pIrpSp         I/O stack location IRP запроса
* @return       Статус
*/
NTSTATUS
IoRemoveNatTableUnit
(
    PIRP                pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoRemoveNatTableUnit: Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS Status = STATUS_UNSUCCESSFUL;
    PBYTE pcInBuffer = pIrp->AssociatedIrp.SystemBuffer;

    if
    (
        RemoveNatTableUnit
        (
            *(PDWORD) (pcInBuffer),          //Local IP      -
            *(PDWORD) (pcInBuffer + 4),      //Global IP     -
            *(PWORD) (pcInBuffer + 8),       //Local port    -
            *(PWORD) (pcInBuffer + 10)      //Global port   -
        )
    )
    {
        Status = STATUS_SUCCESS;
    }
}

```

```

        DbgPrintEx(0, 0, "IoRemoveNatTableUnit: Status = %8x\n",
Status);

        return Status;
    }

/*
* Удаление списка интерфейсов полностью
*
* @param[in]      pIrp          Заголовок IRP запроса
* @param[in]      pIrpSp        I/O stack location IRP запроса
* @return          Статус
*/
NTSTATUS
IoWipeNatTableUnitList
(
    PIRP          pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoWipeNatTableUnitList: Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS Status = STATUS_UNSUCCESSFUL;

    if (WipeNatTableUnitList())
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoWipeNatTableUnitList: Status = %8x\n",
Status);

    return Status;
}

/*
* Проверка, существует ли интерфейс
*
* @param[in]      pIrp          Заголовок IRP запроса
* @param[in]      pIrpSp        I/O stack location IRP запроса
* @return          Статус
*/
NTSTATUS
IoIsNatTableUnitExists
(
    PIRP          pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{

```

```

    DbgPrintEx(0, 0, "IoIsNatTableUnitExists: Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS      Status      = STATUS_UNSUCCESSFUL;
    PBYTE         pcInBuffer  = pIrp->AssociatedIrp.SystemBuffer;

    if
    (
        GetNatTableUnit
        (
            *(PDWORD) (pcInBuffer),          //Local IP      -
        DWORD
            *(PDWORD) (pcInBuffer + 4),      //Global IP     -
        DWORD
            *(PWORD) (pcInBuffer + 8),       //Local port    -
        WORD
            *(PWORD) (pcInBuffer + 10)      //Global port   -
        WORD
        )
    )
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoIsNatTableUnitExists: Status = %8x\n",
    Status);

    return Status;
}

/*
* Проверка, активен ли интерфейс
*
* @param[in]    pIrp          Заголовок IRP запроса
* @param[in]    pIrpSp       I/O stack location IRP запроса
* @return       Статус
*/
NTSTATUS
IoIsNatTableUnitMasking
(
    PIRP          pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoIsNatTableUnitMasking: Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

```

```

NTSTATUS      Status      = STATUS_UNSUCCESSFUL;
PBYTE         pcInBuffer = pIrp->AssociatedIrp.SystemBuffer;

if
(
    IsNatTableUnitMaskingByData
    (
        *(PDWORD) (pcInBuffer),          //Local IP      -
DWORD      *(PDWORD) (pcInBuffer + 4),  //Global IP     -
DWORD      *(PWORD) (pcInBuffer + 8),   //Local port    -
WORD       *(PWORD) (pcInBuffer + 10)   //Global port   -
WORD
    )
)
{
    Status = STATUS_SUCCESS;
}

DbgPrintEx(0, 0, "IoIsNatTableUnitMasking: Status =
%8x\n", Status);

return Status;
}

/*
* Приостановка работы интерфейса
*
* @param[in]    pIrp          Заголовок IRP запроса
* @param[in]    pIrpSp       I/O stack location IRP запроса
* @return       Статус
*/
NTSTATUS
IoSuspendNatTableUnitMasking
(
    PIRP          pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoSuspendNatTableUnitMasking:
Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS      Status      = STATUS_UNSUCCESSFUL;
    PBYTE         pcInBuffer = pIrp->AssociatedIrp.SystemBuffer;

    if
    (

```

```

        SuspendNatTableUnitMaskingByData
        (
            *(PDWORD) (pcInBuffer),          //Local IP      -
DWORD      *(PDWORD) (pcInBuffer + 4),      //Global IP      -
DWORD      *(PWORD) (pcInBuffer + 8),        //Local port     -
WORD       *(PWORD) (pcInBuffer + 10)       //Global port     -
WORD
        )
    )
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoSuspendNatTableUnitMasking: Status =
    %8x\n", Status);

    return Status;
}

/*
* Возобновление работы интерфейса
*
* @param[in]    pIrp          Заголовок IRP запроса
* @param[in]    pIrpSp        I/O stack location IRP запроса
* @return       Статус
*/
NTSTATUS
IoResumeNatTableUnitMasking
(
    PIRP          pIrp,
    PIO_STACK_LOCATION pIrpSp
)
{
    DbgPrintEx(0, 0, "IoResumeNatTableUnitMasking:
    Beginning\n");

    UNREFERENCED_PARAMETER(pIrp);
    UNREFERENCED_PARAMETER(pIrpSp);

    NTSTATUS      Status      = STATUS_UNSUCCESSFUL;
    PBYTE         pcInBuffer = pIrp->AssociatedIrp.SystemBuffer;

    if
    (
        ResumeNatTableUnitMaskingByData
        (
            *(PDWORD) (pcInBuffer),          //Local IP      -
DWORD

```

```

        *(PDWORD) (pcInBuffer + 4),    //Global IP      -
DWORD
        *(PWORD) (pcInBuffer + 8),    //Local port     -
WORD
        *(PWORD) (pcInBuffer + 10)    //Global port    -
WORD
    )
    )
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoResumeNatTableUnitMasking: Status =
%8x\n", Status);

    return Status;
}

/*
* Проверка, изменяет ли драйвер пакеты
*
* @param Нет заданных параметров
* @return Статус
*/
NTSTATUS
IoIsNatTableMasking
(
    VOID
)
{
    DbgPrintEx(0, 0, "IoIsNatTableMasking: Beginning\n");

    NTSTATUS Status = STATUS_UNSUCCESSFUL;

    if(IsNatTableMasking())
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoIsNatTableMasking: Status = %8x\n",
Status);

    return Status;
}

/*
* Остановка изменения пакетов
*
* @param Нет заданных параметров
* @return Статус
*/
NTSTATUS

```

```

IoSuspendNatTableMasking
(
    VOID
)
{
    DbgPrintEx(0, 0, "IoSuspendNatTableMasking: Beginning\n");

    NTSTATUS Status = STATUS_UNSUCCESSFUL;

    if (SuspendNatTableMasking())
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoSuspendNatTableMasking: Status =
%8x\n", Status);

    return Status;
}

/*
* Старт изменения пакетов
*
* @param Нет заданных параметров
* @return Статус
*/
NTSTATUS
IoResumeNatTableMasking
(
    VOID
)
{
    DbgPrintEx(0, 0, "IoResumeNatTableMasking: Beginning\n");

    NTSTATUS Status = STATUS_UNSUCCESSFUL;

    if (ResumeNatTableMasking())
    {
        Status = STATUS_SUCCESS;
    }

    DbgPrintEx(0, 0, "IoResumeNatTableMasking: Status =
%8x\n", Status);

    return Status;
}

```

Файл Nat.c

```
#include "StdAfx.c"

/*
 * Точка входа драйвера
 *
 * @param[in]   pDriverObject  Структура с информацией о
 драйвере
 * @param[in]   pRegistryPath  Путь регистрации драйвера в
 реестре
 * @return                               Статус
 */
NTSTATUS
DriverEntry
(
    PDRIVER_OBJECT pDriverObject,
    PUNICODE_STRING pszRegistryPath
)
{
    DbgPrintEx(0, 0, "DriverEntry: Beginning\n");

    NDIS_FILTER_DRIVER_CHARACTERISTICS FCharacteristics;
    NDIS_STRING
    sFriendlyName;
    NDIS_STRING                               sUniqueName;
    NDIS_STRING                               sServiceName;
    NDIS_STATUS                               Status;

    UNREFERENCED_PARAMETER(pszRegistryPath);

    NdisInitUnicodeString(&sFriendlyName,
        FILTER_FRIENDLY_NAME);
    NdisInitUnicodeString(&sUniqueName,
        FILTER_UNIQUE_NAME);
    NdisInitUnicodeString(&sServiceName,
        FILTER_SERVICE_NAME);

    NdisZeroMemory(&FCharacteristics,
        sizeof(NDIS_FILTER_DRIVER_CHARACTERISTICS));

    FCharacteristics.Header.Type
    = NDIS_OBJECT_TYPE_FILTER_DRIVER_CHARACTERISTICS;
    FCharacteristics.Header.Size
    = sizeof(NDIS_FILTER_DRIVER_CHARACTERISTICS);

    #if NDIS_SUPPORT_NDIS61
        FCharacteristics.Header.Revision =
        NDIS_FILTER_CHARACTERISTICS_REVISION_2;
    #else
        FCharacteristics.Header.Revision =
        NDIS_FILTER_CHARACTERISTICS_REVISION_1;
    #endif
}
```

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		97

```

#endif

FCharacteristics.MajorNdisVersion
= NDIS_FILTER_MAJOR_VERSION;
FCharacteristics.MinorNdisVersion
= NDIS_FILTER_MINOR_VERSION;

FCharacteristics.MajorDriverVersion
    = 1;
FCharacteristics.MinorDriverVersion
    = 0;

FCharacteristics.Flags
    = 0;

FCharacteristics.FriendlyName
= sFriendlyName;
FCharacteristics.UniqueName
    = sUniqueName;
FCharacteristics.ServiceName
= sServiceName;

FCharacteristics.AttachHandler
    = FilterAttach;
FCharacteristics.DetachHandler
    = FilterDetach;
FCharacteristics.RestartHandler
    = FilterRestart;
FCharacteristics.PauseHandler
    = FilterPause;

FCharacteristics.SendNetBufferListsHandler
= FilterSendNetBufferLists;
FCharacteristics.ReturnNetBufferListsHandler      =
FilterReturnNetBufferLists;
FCharacteristics.SendNetBufferListsCompleteHandler
= FilterSendNetBufferListsComplete;
FCharacteristics.ReceiveNetBufferListsHandler
= FilterReceiveNetBufferLists;
FCharacteristics.CancelSendNetBufferListsHandler  =
FilterCancelSendNetBufferLists;

FCharacteristics.DevicePnPEventNotifyHandler      =
FilterDevicePnPEventNotify;
FCharacteristics.NetPnPEventHandler
    = FilterNetPnPEvent;
FCharacteristics.StatusHandler
    = FilterStatus;

FCharacteristics.OidRequestHandler
= NULL;

```

```

FCharacteristics.OidRequestCompleteHandler
= NULL;
FCharacteristics.CancelOidRequestHandler
= NULL;

FCharacteristics.SetOptionsHandler
= NULL;
FCharacteristics.SetFilterModuleOptionsHandler
= NULL;

pDriverObject->DriverUnload
    = DriverUnload;

g_FilterDriverObject
    = pDriverObject;
g_FilterDriverHandle
    = NULL;

do
{
    NdisAllocateSpinLock(&g_FilterSpinLock);
    InitializeListHead(&g_FilterList);

    Status = NdisFRegisterFilterDriver
    (
        pDriverObject,
        (NDIS_HANDLE)g_FilterDriverObject,
        &FCharacteristics,
        &g_FilterDriverHandle
    );

    if (Status != NDIS_STATUS_SUCCESS)
    {
        DbgPrintEx(0, 0, "DriverEntry:
NdisFRegisterFilterDriver failed. Status=0x%X\n", Status);

        NdisFreeSpinLock(&g_FilterSpinLock);

        break;
    }

    Status = RegisterDevice();

    if (Status != NDIS_STATUS_SUCCESS)
    {
        DbgPrintEx(0, 0, "DriverEntry: NatRegisterDevice
failed. Status=0x%X\n", Status);

        NdisFDeregisterFilterDriver(g_FilterDriverHandle);
        NdisFreeSpinLock(&g_FilterSpinLock);
    }
}

```

```

        break;
    }

    Status = AllocateNatTableAndPools();

    if (Status != NDIS_STATUS_SUCCESS)
    {
        DbgPrintEx(0, 0, "DriverEntry:
AllocateNatTableAndPools failed. Status=0x%X\n", Status);

        DeregisterDevice();

        NdisFDeregisterFilterDriver(g_FilterDriverHandle);
        NdisFreeSpinLock(&g_FilterSpinLock);

        break;
    }

    DbgPrintEx(0, 0, "DriverEntry: Successful\n");

} while (FALSE);

return Status;
}

/*
* Точка выгрузки драйвера
*
* @param[in]   pDriverObject  Структура с информацией о
драйвере
* @return                               Нет возвращаемых значений
*/
VOID
DriverUnload
(
    PDRIVER_OBJECT pDriverObject
)
{
    UNREFERENCED_PARAMETER(pDriverObject);

    do
    {
        DbgPrintEx(0, 0, "DriverUnload: Beginning\n");

        FreeNatTableAndPools();
        DeregisterDevice();
        NdisFDeregisterFilterDriver(g_FilterDriverHandle);
        NdisFreeSpinLock(&g_FilterSpinLock);

        DbgPrintEx(0, 0, "DriverUnload: Successful\n");

    } while (FALSE);
}

```

```

}

/*
* Функция подключения модуля к минипорту
*
* @param[in] hNdisFilterHandle          Идентификатор модуля
* @param[in] hFilterDriverContext      Контекст драйвера фильтра
* @param[in] pAttachParameters        Параметры подключения
* @return                               Статус
*/
NDIS_STATUS
FilterAttach
(
    NDIS_HANDLE                      hNdisFilterHandle,
    NDIS_HANDLE                      hFilterDriverContext,
    PNDIS_FILTER_ATTACH_PARAMETERS  pAttachParameters
)
{
    PSNATContext                    pstContext;
    ULONG                          ulContextSize;
    NDIS_FILTER_ATTRIBUTES          stFilterAttributes;
    NDIS_STATUS                    Status;

    do
    {
        ASSERT(hFilterDriverContext ==
        (NDIS_HANDLE)g_FilterDriverObject);
        if (hFilterDriverContext !=
        (NDIS_HANDLE)g_FilterDriverObject)
        {
            Status = NDIS_STATUS_INVALID_PARAMETER;
            break;
        }

        if (( pAttachParameters->MiniportMediaType !=
        NdisMedium802_3)
            && (pAttachParameters->MiniportMediaType !=
        NdisMediumWan)
            && (pAttachParameters->MiniportMediaType !=
        NdisMediumWirelessWan))
        {
            Status = NDIS_STATUS_INVALID_PARAMETER;
            break;
        }

        ulContextSize = sizeof(SNATContext) +
        pAttachParameters->FilterModuleGuidName->Length;

        pstContext =
        (PSNATContext)NdisAllocateMemoryWithTagPriority(hNdisFilterHandle,
        ulContextSize, NAT_CONTEXT_ALLOC_TAG, LowPoolPriority);
    }
    while (Status != NDIS_STATUS_SUCCESS);
}

```

```

        if (pstContext == NULL)
        {
            Status = NDIS_STATUS_RESOURCES;
            break;
        }

        NdisZeroMemory(pstContext, sizeof(SNatContext));

        pstContext->m_FilterHandle =
hNdisFilterHandle;

        NdisZeroMemory(&stFilterAttributes,
sizeof(NDIS_FILTER_ATTRIBUTES));

        stFilterAttributes.Header.Size =
sizeof(NDIS_FILTER_ATTRIBUTES);
        stFilterAttributes.Header.Revision =
NDIS_FILTER_ATTRIBUTES_REVISION_1;
        stFilterAttributes.Header.Type =
NDIS_OBJECT_TYPE_FILTER_ATTRIBUTES;
        stFilterAttributes.Flags = 0;

        Status = NdisFSetAttributes(hNdisFilterHandle,
pstContext, &stFilterAttributes);

        if (Status != NDIS_STATUS_SUCCESS)
        {
            break;
        }

        /*Filter starts work in paused state*/
        pstContext->m_State = FilterPaused;

        NdisAcquireSpinLock(&g_FilterSpinLock);
        InsertHeadList(&g_FilterList, &pstContext->
m_FilterModuleLink);
        NdisReleaseSpinLock(&g_FilterSpinLock);

        if (Status != NDIS_STATUS_SUCCESS)
        {
            if (pstContext != NULL)
            {
                NdisFreeMemoryWithTag(pstContext,
NAT_CONTEXT_ALLOC_TAG);
            }
        }

    } while (FALSE);

    if (Status == NDIS_STATUS_SUCCESS)
    {
        DbgPrintEx(0, 0, "FilterAttach: Successful\n");
    }

```

```

    }
    else
    {
        DbgPrintEx(0, 0, "FilterAttach: Failed\n");
    }

    return Status;
}

/*
 * Функция отключения модуля от минипорта
 *
 * @pragma[in]  hFilterModuleContext    Идентификатор модуля
 * @return      Нет возвращаемых
 значений
 */
VOID
FilterDetach
(
    NDIS_HANDLE          hFilterModuleContext
)
{
    PSNatContext  pstContext =
    (PSNatContext)hFilterModuleContext;

    /*Filter must be in paused state*/
    ASSERT(pstContext->m_State == FilterPaused);

    NdisAcquireSpinLock(&g_FilterSpinLock);
    RemoveEntryList(&pstContext->m_FilterModuleLink);
    NdisReleaseSpinLock(&g_FilterSpinLock);

    ASSERT(pstContext);

    if (pstContext->m_FilterModuleName.Buffer)
    {
        NdisFreeMemoryWithTag(pstContext->m_FilterModuleName.Buffer, NULL);
    }

    NdisFreeMemoryWithTag(pstContext, NULL);

    DbgPrintEx(0, 0, "FilterDetach: Successful\n");
}

/*
 * Выделение памяти для таблицы NAT, pool'a NBL и pool'a NB &
 MDL & Data
 *
 * @param      Нет заданных параметров
 * @return      Статус
 */

```

```

NDIS_STATUS
AllocateNatTableAndPools
(
    VOID
)
{
    NDIS_STATUS Status = NDIS_STATUS_SUCCESS;

    do
    {
        /*Выделение памяти для pool'a NBL*/
        NdisZeroMemory(&g_NblParameters,
            sizeof(NET_BUFFER_LIST_POOL_PARAMETERS));
        NdisZeroMemory(&g_NblPoolHandle,
            sizeof(NDIS_HANDLE));

        g_NblParameters.Header.Type =
            NDIS_OBJECT_TYPE_DEFAULT;
        g_NblParameters.Header.Revision =
            NET_BUFFER_LIST_POOL_PARAMETERS_REVISION_1;
        g_NblParameters.Header.Size =
            NDIS_SIZEOF_NET_BUFFER_LIST_POOL_PARAMETERS_REVISION_1;

        g_NblParameters.PoolTag = NAT_NBL_POOL_ALLOC_TAG;
        g_NblParameters.ProtocolId = NDIS_PROTOCOL_ID_TCP_IP;

        g_NblParameters.ContextSize = 0;

        /*NB и MDL выделяются вручную*/
        g_NblParameters.fAllocateNetBuffer = FALSE;
        g_NblParameters.DataSize = 0;

        g_NblPoolHandle =
            NdisAllocateNetBufferListPool(g_FilterDriverHandle,
                &g_NblParameters);
        if (!g_NblPoolHandle)
        {
            DbgPrintEx(0, 0, "AllocateNatTableAndPools: NBL
pool allocating is failed\n");

            Status = NDIS_STATUS_INVALID_PARAMETER;
        }

        /*Выделение памяти для pool'a NB & MDL & Data*/
        NdisZeroMemory(&g_NbMdlParameters,
            sizeof(NET_BUFFER_POOL_PARAMETERS));
        NdisZeroMemory(&g_NbMdlPoolHandle,
            sizeof(NDIS_HANDLE));

        g_NbMdlParameters.Header.Type =
            NDIS_OBJECT_TYPE_DEFAULT;

```

```

        g_NbMdlParameters.Header.Revision =
NET_BUFFER_POOL_PARAMETERS_REVISION_1;
        g_NbMdlParameters.Header.Size =
NDIS_SIZEOF_NET_BUFFER_POOL_PARAMETERS_REVISION_1;

        g_NbMdlParameters.PoolTag =
NAT_NB_MDL_POOL_ALLOC_TAG;
        g_NbMdlParameters.DataSize = 0;

        g_NbMdlPoolHandle =
NdisAllocateNetBufferPool(g_FilterDriverHandle,
&g_NbMdlParameters);
        if (!g_NbMdlPoolHandle)
        {
            DbgPrintEx(0, 0, "AllocateNatTableAndPools: NB
pool allocating is failed\n");

            /*Если выделение pool'a неудачно, память pool'a
NBL будет освобождена*/
            NdisFreeNetBufferListPool(g_NblPoolHandle);

            Status = NDIS_STATUS_INVALID_PARAMETER;
        }

        /*Выделение памяти для таблицы NAT*/
        g_pstNatTable =
NdisAllocateMemoryWithTagPriority(g_FilterDriverHandle,
sizeof(SNatTable), NAT_TABLE_ALLOC_TAG, LowPoolPriority);
        if (!g_pstNatTable)
        {
            DbgPrintEx(0, 0, "AllocateNatTableAndPools: NAT
table allocating is failed\n");

            /*Если выделение таблицы неудачно, память уже
выделенных pool'ов будет освобождена*/
            NdisFreeNetBufferPool(g_NbMdlPoolHandle);
            NdisFreeNetBufferListPool(g_NblPoolHandle);

            Status = NDIS_STATUS_INVALID_PARAMETER;
        }

        /*Обнуляются указатели на первый и последний
интерфейс, т.к. по умолчанию интерфейсов нет*/
        g_pstNatTable->m_pstTableHead = g_pstNatTable-
>m_pstTableTail = NULL;

        /*Модификация пакетов включена по умолчанию*/
        g_pstNatTable->m_bMasking = TRUE;

        /*ID первого модифицированного пакета*/
        g_pstNatTable->m_wNextPackageId = 0x0001;

```

```

        /*Дескриптор таблицы*/
        g_pstNatTable->m_NatTableHandle =
g_FilterDriverHandle;

        NatTableInitCriticalSection();

        DbgPrintEx(0, 0, "AllocateNatTableAndPools:
Successful\n");

        } while (FALSE);

        return Status;
    }

/*
* Освобождение памяти таблицы NAT, pool'a NBL и pool'a NB &
MDL & Data
*
* @param      Нет заданных параметров
* @return      Нет возвращаемых значений
*/
VOID
FreeNatTableAndPools
(
    VOID
)
{
    do
    {
        NatTableFreeCriticalSection();
        WipeNatTableUnitList();
        NdisFreeNetBufferPool(g_NbMdlPoolHandle);
        NdisFreeNetBufferListPool(g_NblPoolHandle);
        NdisFreeMemoryWithTag(g_pstNatTable,
NAT_TABLE_ALLOC_TAG);

        DbgPrintEx(0, 0, "FreeNatTableAndPools:
Successful\n");

        } while (FALSE);
    }

/*
* Функция приостановки модуля
*
* @param[in]    hFilterModuleContext      Контекст модуля
* @param[in]    pPauseParameters          Параметры приостановки
* @return                          Статус
*/
NDIS_STATUS
FilterPause
(

```

```

    NDIS_HANDLE
    hFilterModuleContext,
    PNDIS_FILTER_PAUSE_PARAMETERS pPauseParameters
)
{
    UNREFERENCED_PARAMETER(pPauseParameters);

    PSNatContext pstContext =
(PSNatContext) (hFilterModuleContext);

    ASSERT(pstContext->m_State == FilterRunning);

    NdisAcquireSpinLock(&pstContext->m_Lock);
    pstContext->m_State = FilterPausing;
    NdisReleaseSpinLock(&pstContext->m_Lock);

    pstContext->m_State = FilterPaused;

    DbgPrintEx(0, 0, "FilterPause: Successful\n");

    return NDIS_STATUS_SUCCESS;
}

/*
* Функция перезапуска модуля
*
* @param[in]    hFilterModuleContext        Контекст модуля
* @param[in]    pFilterRestartParameters    Параметры перезапуска
* @return                               Статус
*/
NDIS_STATUS
FilterRestart
(
    NDIS_HANDLE
    hFilterModuleContext,
    PNDIS_FILTER_RESTART_PARAMETERS
    pFilterRestartParameters
)
{
    UNREFERENCED_PARAMETER(pFilterRestartParameters);

    PSNatContext pstContext =
(PSNatContext) hFilterModuleContext;

    ASSERT(pstContext->m_State == FilterPaused);

    NdisAcquireSpinLock(&pstContext->m_Lock);
    pstContext->m_State = FilterRunning;
    NdisReleaseSpinLock(&pstContext->m_Lock);

    DbgPrintEx(0, 0, "FilterRestart: Successful\n");
}

```

```

        return NDIS_STATUS_SUCCESS;
    }

    /*
    * Оповещение о произошедшем PnP событии от оборудования
    *
    * @param[in]   hFilterModuleContext      Контекст модуля фильтра
    * @param[in]   pNetDevicePnPEvent       PnP событие от
    оборудования
    * @return                               Нет возвращаемых
    значений
    */
    VOID
    FilterDevicePnPEventNotify
    (
        NDIS_HANDLE                hFilterModuleContext,
        PNET_DEVICE_PNP_EVENT      pNetDevicePnPEvent
    )
    {
        PSNATContext                pstContext          =
        (PSNATContext)hFilterModuleContext;
        NDIS_DEVICE_PNP_EVENT      DevicePnPEvent =
        pNetDevicePnPEvent->DevicePnPEvent;

        switch (DevicePnPEvent)
        {
            case NdisDevicePnPEventQueryRemoved:
            case NdisDevicePnPEventRemoved:
            case NdisDevicePnPEventSurpriseRemoved:
            case NdisDevicePnPEventQueryStopped:
            case NdisDevicePnPEventStopped:
            case NdisDevicePnPEventPowerProfileChanged:
            case NdisDevicePnPEventFilterListChanged:
                break;

            default:
                break;
        }

        NdisFDevicePnPEventNotify(pstContext->m_FilterHandle,
        pNetDevicePnPEvent);

        DbgPrintEx(0, 0, "FilterDevicePnPEventNotify: Device PnP
        event\n");
    }

    /*
    * Оповещение о произошедшем сетевом PnP событии
    *
    * @param[in]   hFilterModuleContext      Контекст модуля
    фильтра
    * @param[in]   pNetPnPEventNotification Сетевое PnP событие
    */

```

```

* @return                                     Статус
*/
NDIS_STATUS
FilterNetPnPEvent
(
    NDIS_HANDLE
    hFilterModuleContext,
    PNET_PNP_EVENT_NOTIFICATION
    pNetPnPEventNotification
)
{
    PSNatContext    pstContext                =
(PSNatContext)hFilterModuleContext;
    NDIS_STATUS     Status                    =
NDIS_STATUS_SUCCESS;

    Status = NdisFNetPnPEvent(pstContext->m_FilterHandle,
pNetPnPEventNotification);

    DbgPrintEx(0, 0, "FilterNetPnPEvent: PnP event\n");

    return Status;
}

/*
* Оповещение об изменении состояния нижележащего драйвера
*
* @param[in]    hFilterModuleContext    Контекст модуля фильтра
* @param[in]    pStatusIndication      Статусная информация
* @return                               Нет возвращаемых
значений
*/
VOID
FilterStatus
(
    NDIS_HANDLE                                hFilterModuleContext,
    PNDIS_STATUS_INDICATION                    pStatusIndication
)
{
    PSNatContext pstContext =
(PSNatContext)hFilterModuleContext;

    NdisFIndicateStatus(pstContext->m_FilterHandle,
pStatusIndication);

    DbgPrintEx(0, 0, "FilterStatus: Status indication\n");
}

/*
* Отправка пакета данных
*
* @param[in]    hFilterModuleContext    Контекст модуля фильтра

```

```

* @param[in]   pNetBufferLists           Указатель на
первый NBL в NBL list'e
* @param[in]   ulPortNumber              Номер порта адаптера
минипорта
* @param[in]   ulSendFlags               Флаг отправки
* @return      Нет возвращаемых
значений
*/
VOID
FilterSendNetBufferLists
(
    NDIS_HANDLE                hFilterModuleContext,
    PNET_BUFFER_LIST            pNetBufferLists,
    NDIS_PORT_NUMBER            ulPortNumber,
    ULONG                       ulSendFlags
)
{
    PSNATContext                pstContext =
(PSNATContext)hFilterModuleContext;
    PNET_BUFFER_LIST            pCopiedNbl = NULL;

    do
    {
        if (IsNatTableMasking())
        {
            if (IsPacketRelevant(pNetBufferLists, TRUE))
            {
                DbgPrintEx(0, 0, "FilterSendNetBufferLists:
Packet is relevant\n");

                NdisAcquireSpinLock(&pstContext->m_Lock);
                pCopiedNbl = CopyPacket(pNetBufferLists);
                NdisReleaseSpinLock(&pstContext->m_Lock);

                DbgPrintEx(0, 0, "FilterSendNetBufferLists:
Packet copied\n");

                if (pCopiedNbl)
                {
                    ModificatePacket(pCopiedNbl, TRUE);

                    NdisFSendNetBufferLists(pstContext->
m_FilterHandle, pCopiedNbl, ulPortNumber, ulSendFlags);

                    NdisFSendNetBufferListsComplete(pstContext->
m_FilterHandle, pNetBufferLists, ulSendFlags);

                    DbgPrintEx(0, 0,
"FilterSendNetBufferLists: Packet modified and sended\n");
                    break;
                }
            }
        }
    }
}

```

```

    }

    NdisFSendNetBufferLists(pstContext->m_FilterHandle,
pNetBufferLists, ulPortNumber, ulSendFlags);

    } while (FALSE);
}

/*
* Завершение отправки пакета данных
*
* @param[in]    hFilterModuleContext    Контекст модуля фильтра
* @param[in]    pNetBufferLists        Указатель на
первый NBL в NBL list'e
* @param[in]    ulSendCompleteFlags    Флаги завершения
отправки
* @return                                Нет возвращаемых
значений
*/
VOID
FilterSendNetBufferListsComplete
(
    NDIS_HANDLE                hFilterModuleContext,
    PNET_BUFFER_LIST            pNetBufferLists,
    ULONG                       ulSendCompleteFlags
)
{
    PSNATContext                pstContext =
(PSNATContext)hFilterModuleContext;

    if (pNetBufferLists != NULL)
    {
        if (IsPacketCopied(pNetBufferLists))
        {
            FreePacket(pNetBufferLists);

            DbgPrintEx(0, 0,
"FilterSendNetBufferListsComplete: Modified packet freed\n");
        }
        else
        {
            NdisFSendNetBufferListsComplete(pstContext->m_FilterHandle, pNetBufferLists, ulSendCompleteFlags);
        }
    }
}

/*
* Отмена отправки списка NET_BUFFER_LIST'ов
*
* @param[in]    hFilterModuleContext    Контекст модуля фильтра

```

```

* @param[in]  pCancelId          ID отменяемого
NET_BUFFER_LIST'a
* @return          Функция не
возвращает значений
*/
VOID
FilterCancelSendNetBufferLists
(
    NDIS_HANDLE          hFilterModuleContext,
    PVOID                pCancelId
)
{
    PSNAT_CONTEXT  pstContext =
(PSNAT_CONTEXT)hFilterModuleContext;

    NdisCancelSendNetBufferLists(pstContext->m_FilterHandle,
pCancelId);

    DbgPrintEx(0, 0, "FilterCancelSendNetBufferLists: Packet
canceled\n");
}

/*
* Приём пакета данных
*
* @param[in]  hFilterModuleContext      Контекст модуля
фильтра
* @param[in]  pNetBufferLists          Указатель на
первый NBL в NBL list'e
* @param[in]  ulPortNumber            Номер порта
адаптера минипорта
* @param[in]  ulNumberOfNetBufferLists  Количество принимаемых
NBL
* @param[in]  ulReceiveFlags          Флаги приёма
* @return          Нет
возвращаемых значений
*/
VOID
FilterReceiveNetBufferLists
(
    NDIS_HANDLE          hFilterModuleContext,
    PNET_BUFFER_LIST     pNetBufferLists,
    NDIS_PORT_NUMBER     ulPortNumber,
    ULONG                ulNumberOfNetBufferLists,
    ULONG                ulReceiveFlags
)
{
    PSNAT_CONTEXT  pstContext =
(PSNAT_CONTEXT)hFilterModuleContext;
    PNET_BUFFER_LIST  pCopiedNbl = NULL;

    do

```

```

{
    if (IsNatTableMasking())
    {

        if (IsPacketRelevant(pNetBufferLists, FALSE))
        {
            DbgPrintEx(0, 0,
"FilterReceiveNetBufferLists: Packet is relevant\n");

            NdisAcquireSpinLock(&pstContext->m_Lock);
            pCopiedNbl = CopyPacket(pNetBufferLists);
            NdisReleaseSpinLock(&pstContext->m_Lock);

            DbgPrintEx(0, 0,
"FilterReceiveNetBufferLists: Packet copied\n");
            if (pCopiedNbl)
            {
                ModifyPacket(pCopiedNbl, FALSE);

                NdisFIndicateReceiveNetBufferLists
                (
                    pstContext->m_FilterHandle,
                    pCopiedNbl,
                    ulPortNumber,
                    ulNumberOfNetBufferLists,
                    ulReceiveFlags
                );

                DbgPrintEx(0, 0,
"FilterReceiveNetBufferLists: Modified packet indicated\n");

                if (!(NDIS_RECEIVE_FLAGS_RESOURCES &
ulReceiveFlags))
                {
                    ULONG ulReturnFlags = 0;
                    if
(NDIS_RECEIVE_FLAGS_DISPATCH_LEVEL & ulReceiveFlags)
                    {
                        ulReturnFlags |=
NDIS_RETURN_FLAGS_DISPATCH_LEVEL;
                    }

                    NdisFReturnNetBufferLists(pstContext->m_FilterHandle,
pNetBufferLists, ulReturnFlags);
                    DbgPrintEx(0, 0,
"FilterReceiveNetBufferLists: Original packet returned\n");
                }

                break;
            }
        }
    }
}

```

```

    }
}

NdisFIndicateReceiveNetBufferLists
(
    pstContext->m_FilterHandle,
    pNetBufferLists,
    ulPortNumber,
    ulNumberOfNetBufferLists,
    ulReceiveFlags
);

} while (FALSE);
}

/*
* Освобождение памяти NBL list'a
*
* @param[in]   hFilterModuleContext   Контекст модуля фильтра
* @param[in]   pNetBufferLists        Указатель на
первый NBL в NBL list'e
* @param[in]   ulReturnFlags          Флаги освобождения
памяти
* @return                               Нет возвращаемых
значений
*/
VOID
FilterReturnNetBufferLists
(
    NDIS_HANDLE           hFilterModuleContext,
    PNET_BUFFER_LIST      pNetBufferLists,
    ULONG                 ulReturnFlags
)
{
    PSNATContext           pstContext =
(PSNATContext)hFilterModuleContext;

    if (pNetBufferLists != NULL)
    {
        if (IsPacketCopied(pNetBufferLists))
        {
            FreePacket(pNetBufferLists);

            DbgPrintEx(0, 0, "FilterReturnNetBufferLists:
Modified NBLs freed\n");
        }
        else
        {
            NdisFReturnNetBufferLists(pstContext->m_FilterHandle, pNetBufferLists, ulReturnFlags);
        }
    }
}

```

Файл NatTable.c

```
#include "StdAfx.c"

/*
 * Инициализация критической секции таблицы NAT
 *
 * @param Нет заданных параметров
 * @return Нет возвращаемых значений
 */
VOID
NatTableInitCriticalSection
(
    VOID
)
{
    DbgPrintEx(0, 0, "NatTableInitCriticalSection:
Beginning\n");

    NdisAllocateSpinLock(&g_pstNatTable->m_NatTableSpinLock);

    DbgPrintEx(0, 0, "NatTableInitCriticalSection:
Success\n");
}

/*
 * Освобождение критической секции таблицы NAT
 *
 * @param Нет заданных параметров
 * @return Нет возвращаемых значений
 */
VOID
NatTableFreeCriticalSection
(
    VOID
)
{
    DbgPrintEx(0, 0, "NatTableFreeCriticalSection:
Beginning\n");

    NdisFreeSpinLock(&g_pstNatTable->m_NatTableSpinLock);

    DbgPrintEx(0, 0, "NatTableFreeCriticalSection:
Success\n");
}

/*
 * Вход в критическую секцию таблицы NAT
 *
 * @param Нет заданных параметров
 * @return Нет возвращаемых значений
 */
```

```

VOID
NatTableEnterCriticalSection
(
    VOID
)
{
    NdisAcquireSpinLock(&g_pstNatTable->m_NatTableSpinLock);
}

/*
 * Выход из критической секции таблицы NAT
 *
 * @param Нет заданных параметров
 * @return Нет возвращаемых значений
 */
VOID
NatTableLeaveCriticalSection
(
    VOID
)
{
    NdisReleaseSpinLock(&g_pstNatTable->m_NatTableSpinLock);
}

/*
 * Проверка, включена ли модификация пакетов таблицей NAT
 *
 * @param Нет заданных параметров
 * @return TRUE, если модификация пакетов включена
 */
BOOLEAN
IsNatTableMasking
(
    VOID
)
{
    BOOLEAN bIsMasking = FALSE;

    NatTableEnterCriticalSection();
    bIsMasking = g_pstNatTable->m_bMasking;
    NatTableLeaveCriticalSection();

    return bIsMasking;
}

/*
 * Приостановка модификации пакетов таблицей NAT
 *
 * @param Нет заданных параметров
 * @return TRUE, если модификация пакетов приостановлена
 */
BOOLEAN

```

```

SuspendNatTableMasking
(
    VOID
)
{
    BOOLEAN bIsSuspended = FALSE;

    NatTableEnterCriticalSection();
    if (!(g_pstNatTable->m_bMasking = FALSE))
    {
        bIsSuspended = TRUE;
    }
    NatTableLeaveCriticalSection();

    return bIsSuspended;
}

/*
* Возобновление модификации пакетов таблицей NAT
*
* @param Нет заданных параметров
* @return TRUE, если модификация пакетов возобновлена
*/
BOOLEAN
ResumeNatTableMasking
(
    VOID
)
{
    BOOLEAN bIsResumed = FALSE;

    NatTableEnterCriticalSection();
    if ((g_pstNatTable->m_bMasking = TRUE))
    {
        bIsResumed = TRUE;
    }
    NatTableLeaveCriticalSection();

    return bIsResumed;
}

/*
* Проверка, включена ли модификация пакетов NAT-единицей
таблицы
*
* @param[in]    dwLocalIp        Локальный адрес интерфейса
* @param[in]    dwGlobalIp       Глобальный адрес интерфейса
* @param[in]    wLocalPort       Локальный порт интерфейса
* @param[in]    wGlobalPort      Глобальный порт интерфейса
* @return       TRUE, если модификация
пакетов включена
*/

```

```

BOOLEAN
IsNatTableUnitMaskingByData
(
    DWORD      dwLocalIp,
    DWORD      dwGlobalIp,
    WORD  wLocalPort,
    WORD  wGlobalPort
)
{
    PSNatTableUnit pstUnit = GetNatTableUnit(dwLocalIp,
dwGlobalIp, wLocalPort, wGlobalPort);
    BOOLEAN bIsMasking = FALSE;

    NatTableEnterCriticalSection();
    bIsMasking = pstUnit->m_bMasking;
    NatTableLeaveCriticalSection();

    return bIsMasking;
}

/*
* Проверка, включена ли модификация пакетов NAT-единицей
таблицы
*
* @param[in]    pstCurrentUnit NAT-единица
* @return                      TRUE, если модификация
пакетов включена
*/
BOOLEAN
IsNatTableUnitMasking
(
    PSNatTableUnit pstUnit
)
{
    BOOLEAN bIsMasking = FALSE;

    NatTableEnterCriticalSection();
    if (pstUnit)
    {
        bIsMasking = pstUnit->m_bMasking;
    }
    NatTableLeaveCriticalSection();

    return bIsMasking;
}

/*
* Приостановка модификации пакетов NAT-единицей таблицы
*
* @param[in]    dwLocalIp      Локальный адрес интерфейса
* @param[in]    dwGlobalIp     Глобальный адрес интерфейса

```

```

* @param[in]    wLocalPort        Локальный порт интерфейса
* @param[in]    wGlobalPort       Глобальный порт интерфейса
* @return       TRUE, если модификация
пакетов приостановлена
*/
BOOLEAN
SuspendNatTableUnitMaskingByData
(
    DWORD        dwLocalIp,
    DWORD        dwGlobalIp,
    WORD wLocalPort,
    WORD wGlobalPort
)
{
    PSNatTableUnit pstUnit = GetNatTableUnit(dwLocalIp,
dwGlobalIp, wLocalPort, wGlobalPort);
    BOOLEAN bIsSuspended = FALSE;

    NatTableEnterCriticalSection();
    if (!(pstUnit->m_bMasking = FALSE))
    {
        bIsSuspended = TRUE;
    }
    NatTableLeaveCriticalSection();

    return bIsSuspended;
}

/*
* Приостановка модификации пакетов NAT-единицей таблицы
*
* @param[in]    pstCurrentUnit NAT-единица
* @return       TRUE, если модификация
пакетов приостановлена
*/
BOOLEAN
SuspendNatTableUnitMasking
(
    PSNatTableUnit pstUnit
)
{
    BOOLEAN bIsSuspended = FALSE;

    NatTableEnterCriticalSection();
    if (pstUnit)
    {
        if (!(pstUnit->m_bMasking = FALSE))
        {
            bIsSuspended = TRUE;
        }
    }
    NatTableLeaveCriticalSection();
}

```

```

        return bIsSuspended;
    }

    /*
    * Возобновление модификации пакетов NAT-единицей таблицы
    *
    * @param[in]    dwLocalIp        Локальный адрес интерфейса
    * @param[in]    dwGlobalIp      Глобальный адрес интерфейса
    * @param[in]    wLocalPort      Локальный порт интерфейса
    * @param[in]    wGlobalPort     Глобальный порт интерфейса
    * @return      TRUE, если модификация
    пакетов возобновлена
    */
    BOOLEAN
    ResumeNatTableUnitMaskingByData
    (
        DWORD        dwLocalIp,
        DWORD        dwGlobalIp,
        WORD wLocalPort,
        WORD wGlobalPort
    )
    {
        PSNatTableUnit pstUnit = GetNatTableUnit(dwLocalIp,
dwGlobalIp, wLocalPort, wGlobalPort);
        BOOLEAN bIsResumed = FALSE;

        NatTableEnterCriticalSection();
        if ((pstUnit->m_bMasking = TRUE))
        {
            bIsResumed = TRUE;
        }
        NatTableLeaveCriticalSection();

        return bIsResumed;
    }

    /*
    * Возобновление модификации пакетов NAT-единицей таблицы
    *
    * @param[in]    pstCurrentUnit NAT-единица
    * @return      TRUE, если модификация
    пакетов возобновлена
    */
    BOOLEAN
    ResumeNatTableUnitMasking
    (
        PSNatTableUnit pstUnit
    )
    {
        BOOLEAN bIsResumed = FALSE;

```

```

    NatTableEnterCriticalSection();
    if (pstUnit)
    {
        if ((pstUnit->m_bMasking = TRUE))
        {
            bIsResumed = TRUE;
        }
    }
    NatTableLeaveCriticalSection();

    return bIsResumed;
}

/*
* Попытка добавить NAT-единицу в таблицу
*
* @param[in]    dwLocalIp        Локальный адрес интерфейса
* @param[in]    dwGlobalIp       Глобальный адрес интерфейса
* @param[in]    wLocalPort       Локальный порт интерфейса
* @param[in]    wGlobalPort      Глобальный порт интерфейса
* @return       TRUE, если был добавлен
новый интерфейс
*/
BOOLEAN
AddNatTableUnit
(
    DWORD        dwLocalIp,
    DWORD        dwGlobalIp,
    WORD wLocalPort,
    WORD wGlobalPort
)
{
    BOOLEAN bSuccess = FALSE;

    /*Если аналогичной NAT-единицы ещё нет в таблице*/
    if (!GetNatTableUnit(dwLocalIp, dwGlobalIp, wLocalPort,
wGlobalPort))
    {
        bSuccess = CreateNatTableUnit(dwLocalIp, dwGlobalIp,
wLocalPort, wGlobalPort);
    }

    return bSuccess;
}

/*
* Запрос NAT-единицы из таблицы по заданным параметрам
*
* @param[in]    dwLocalIp        Локальный адрес интерфейса
* @param[in]    dwGlobalIp       Глобальный адрес интерфейса
* @param[in]    wLocalPort       Локальный порт интерфейса
* @param[in]    wGlobalPort      Глобальный порт интерфейса

```

```

* @return                                Указатель на NAT-единицу,
если была найдена
*/
PSNatTableUnit
GetNatTableUnit
(
    DWORD      dwLocalIp,
    DWORD      dwGlobalIp,
    WORD  wLocalPort,
    WORD  wGlobalPort
)
{
    PSNatTableUnit pstCurrentUnit = NULL;
    PSNatTableUnit pstRelevantUnit = NULL;

    NatTableEnterCriticalSection();

    for
    (
        pstCurrentUnit = g_pstNatTable->m_pstTableHead;
        pstCurrentUnit;
        pstCurrentUnit = pstCurrentUnit->m_pstNext
    )
    {
        if
        (
            pstCurrentUnit->m_dwLocalIp      == dwLocalIp
            &&
            pstCurrentUnit->m_dwGlobalIp == dwGlobalIp &&
            pstCurrentUnit->m_wLocalPort == wLocalPort &&
            pstCurrentUnit->m_wGlobalPort == wGlobalPort
        )
        {
            pstRelevantUnit = pstCurrentUnit;
            break;
        }
    }

    NatTableLeaveCriticalSection();

    return pstRelevantUnit;
}

/*
* Запрос NAT-единицы из таблицы по параметрам из буфера
данных
*
* @param[in]  pcDataBuffer  Указатель на буфер данных
* @param[in]  bPacketRoute  Флаг, является ли пакет
исходящим. TRUE - исходящий пакет, FALSE - входящий пакет
* @return                                Указатель на NAT-единицу,
если была найдена

```

```

*/
PSNatTableUnit
GetNatTableUnitByDataBuffer
(
    PBYTE      pcDataBuffer,
    BOOLEAN bPacketRoute
)
{
    PSNatTableUnit pstCurrentUnit = NULL;
    PSNatTableUnit pstRelevantUnit = NULL;
    DWORD          dwIpAddress      = 0;
    WORD           wPort             = 0;
    WORD           wEtherType        = 0;
    BYTE           cProtocol         = 0;

    wEtherType = GetReversedWord(*(PWORD) (pcDataBuffer +
ETH_TYPE_INDEX));
    if (wEtherType == IPV4_ETH_TYPE_CODE)
    {
        cProtocol = *(PBYTE) (pcDataBuffer +
IPV4_PROTOCOL_INDEX);

        if (bPacketRoute)
        {
            dwIpAddress = *(PDWORD) (pcDataBuffer +
IPV4_IP_SOURCE_INDEX);

            if (cProtocol == TCP_PROTOCOL_CODE || cProtocol
== UDP_PROTOCOL_CODE)
            {
                wPort = *(PWORD) (pcDataBuffer +
IPV4_PORT_SOURCE_INDEX);
            }
        }
        else
        {
            dwIpAddress = *(PDWORD) (pcDataBuffer +
IPV4_IP_DESTINATION_INDEX);

            if (cProtocol == TCP_PROTOCOL_CODE || cProtocol
== UDP_PROTOCOL_CODE)
            {
                wPort = *(PWORD) (pcDataBuffer +
IPV4_PORT_DESTINATION_INDEX);
            }
        }
    }
    else if (wEtherType == ARP_ETH_TYPE_CODE)
    {
        if (bPacketRoute)
        {

```

```

        dwIpAddress = *(PDWORD) (pcDataBuffer +
ARP_IP_SOURCE_INDEX);
    }
    else
    {
        dwIpAddress = *(PDWORD) (pcDataBuffer +
ARP_IP_DESTINATION_INDEX);
    }
}

NatTableEnterCriticalSection();

for
(
    pstCurrentUnit = g_pstNatTable->m_pstTableHead;
    pstCurrentUnit;
    pstCurrentUnit = pstCurrentUnit->m_pstNext
)
{
    if (bPacketRoute)
    {
        if (cProtocol == TCP_PROTOCOL_CODE || cProtocol
== UDP_PROTOCOL_CODE)
        {
            if (dwIpAddress == pstCurrentUnit-
>m_dwLocalIp && wPort == pstCurrentUnit->m_wLocalPort)
            {
                pstRelevantUnit = pstCurrentUnit;
                break;
            }
        }
        else
        {
            if (dwIpAddress == pstCurrentUnit-
>m_dwLocalIp)
            {
                pstRelevantUnit = pstCurrentUnit;
                break;
            }
        }
    }
    else
    {
        if (cProtocol == TCP_PROTOCOL_CODE || cProtocol
== UDP_PROTOCOL_CODE)
        {
            if (dwIpAddress == pstCurrentUnit-
>m_dwGlobalIp && wPort == pstCurrentUnit->m_wGlobalPort)
            {
                pstRelevantUnit = pstCurrentUnit;
                break;
            }
        }
    }
}

```

```

        }
        else
        {
            if (dwIpAddress == pstCurrentUnit->m_dwGlobalIp)
            {
                pstRelevantUnit = pstCurrentUnit;
                break;
            }
        }
    }

    NatTableLeaveCriticalSection();

    return pstRelevantUnit;
}

/*
 * Создание NAT-единицы
 *
 * @param[in]    dwLocalIp        Локальный адрес интерфейса
 * @param[in]    dwGlobalIp      Глобальный адрес интерфейса
 * @param[in]    wLocalPort      Локальный порт интерфейса
 * @param[in]    wGlobalPort     Глобальный порт интерфейса
 * @return       Указатель на успешно
                созданную NAT-единицу
 */
PSNatTableUnit
CreateNatTableUnit
(
    DWORD        dwLocalIp,
    DWORD        dwGlobalIp,
    WORD wLocalPort,
    WORD wGlobalPort
)
{
    PSNatTableUnit pstUnit = NULL;

    do
    {
        pstUnit = NdisAllocateMemoryWithTagPriority
        (
            g_pstNatTable->m_NatTableHandle,
            sizeof(SNatTableUnit),
            NAT_TABLE_UNIT_ALLOC_TAG,
            LowPoolPriority
        );

        if (!pstUnit)
        {
            break;
        }
    }
    while (1);
}

```

```

    }

    /*Основная информация NAT-единицы*/
    pstUnit->m_dwLocalIp      = dwLocalIp;
    pstUnit->m_dwGlobalIp     = dwGlobalIp;
    pstUnit->m_wLocalPort     = wLocalPort;
    pstUnit->m_wGlobalPort    = wGlobalPort;

    /*Модификация пакетов данной NAT-единицей по
    умолчанию включена*/
    pstUnit->m_bMasking       = TRUE;

    /*Далее происходит интеграция NAT-единицы в таблицу*/

    NatTableEnterCriticalSection();

    /*Если в таблице уже есть элементы*/
    if (g_pstNatTable->m_pstTableHead && g_pstNatTable-
    >m_pstTableTail)
    {
        g_pstNatTable->m_pstTableTail->m_pstNext      =
pstUnit;
        pstUnit->m_pstPrevious
    = g_pstNatTable->m_pstTableTail;
        g_pstNatTable->m_pstTableTail
    =
pstUnit;
        pstUnit->m_pstNext
    = NULL;
    }
    else
    {
        g_pstNatTable->m_pstTableHead
    =
pstUnit;
        g_pstNatTable->m_pstTableTail
    =
pstUnit;
        pstUnit->m_pstNext
    = NULL;
        pstUnit->m_pstPrevious
    = NULL;
    }

    NatTableLeaveCriticalSection();

    } while (FALSE);

    return pstUnit;
}

/*
* Удаление NAT-единицы
*
* @param[in]    dwLocalIp        Локальный адрес интерфейса

```

```

* @param[in]   dwGlobalIp           Глобальный адрес интерфейса
* @param[in]   wLocalPort          Локальный порт интерфейса
* @param[in]   wGlobalPort         Глобальный порт интерфейса
* @return      TRUE, если NAT-единица
успешно удалена
*/
BOOLEAN
RemoveNatTableUnit
(
    DWORD      dwLocalIp,
    DWORD      dwGlobalIp,
    WORD wLocalPort,
    WORD wGlobalPort
)
{
    PSNatTableUnit pstUnit      = NULL;
    BOOLEAN          bSuccess    = FALSE;

    pstUnit = GetNatTableUnit(dwLocalIp, dwGlobalIp,
wLocalPort, wGlobalPort);
    if (pstUnit)
    {
        NatTableEnterCriticalSection();

        /*Если элемент является первым в таблице*/
        if (pstUnit == g_pstNatTable->m_pstTableHead)
        {
            g_pstNatTable->m_pstTableHead = g_pstNatTable->m_pstTableTail = NULL;
        }
        /*Если элемент является последним в таблице*/
        else if (pstUnit == g_pstNatTable->m_pstTableTail)
        {
            pstUnit->m_pstPrevious->m_pstNext = NULL;
            g_pstNatTable->m_pstTableTail = pstUnit->m_pstPrevious;
        }
        /*Если элемент является промежуточным в таблице*/
        else if (pstUnit != g_pstNatTable->m_pstTableHead &&
pstUnit != g_pstNatTable->m_pstTableTail)
        {
            pstUnit->m_pstPrevious->m_pstNext = pstUnit->m_pstNext;
            pstUnit->m_pstNext->m_pstPrevious = pstUnit->m_pstPrevious;
        }

        NdisFreeMemoryWithTag(pstUnit,
NAT_TABLE_UNIT_ALLOC_TAG);

        NatTableLeaveCriticalSection();
    }
}

```

```

        bSuccess = TRUE;
    }

    return bSuccess;
}

/*
 * Удаление всех NAT-единиц из таблицы
 *
 * @param Нет заданных параметров
 * @return TRUE, если таблица очищена
 */
BOOLEAN
WipeNatTableUnitList
(
    VOID
)
{
    PSNatTableUnit pstCurrentUnit = NULL;
    PSNatTableUnit pstNextUnit    = NULL;
    BOOLEAN          bSuccess      = FALSE;

    NatTableEnterCriticalSection();

    for
    (
        pstCurrentUnit = g_pstNatTable->m_pstTableHead;
        pstCurrentUnit;
        pstCurrentUnit = pstNextUnit
    )
    {
        pstNextUnit = pstCurrentUnit->m_pstNext;
        RemoveNatTableUnit(pstCurrentUnit->m_dwLocalIp,
        pstCurrentUnit->m_dwGlobalIp, pstCurrentUnit->m_wLocalPort,
        pstCurrentUnit->m_wGlobalPort);
    }

    NatTableLeaveCriticalSection();

    return (bSuccess = TRUE);
}

/*
 * Обновление ID для следующего отправляемого модифицируемого
 пакета
 *
 * @param Нет заданных параметров
 * @return Нет возвращаемых значений
 */
VOID
UpdateNatTableNextPacketId
(

```

```

        VOID
    )
    {
        NatTableEnterCriticalSection();

        /*Если ID не превышает разрядность WORD*/
        if (g_pstNatTable->m_wNextPackageId < 0xffff)
        {
            g_pstNatTable->m_wNextPackageId++;
        }
        else
        {
            g_pstNatTable->m_wNextPackageId = 0x0001;
        }

        NatTableLeaveCriticalSection();
    }

```

Файл PacketCopying.c

```

#include "StdAfx.c"

/*
 * Полная копия NBL
 *
 * @param[in]    pSourceNbl    Копируемый pNBL
 * @return       Скопированный pNBL
 */
PNET_BUFFER_LIST
CopyPacket
(
    PNET_BUFFER_LIST pNetBufferLists
)
{
    PNET_BUFFER_LIST    pSourceNbl                = NULL;
    PNET_BUFFER_LIST    pDestinationNbl           = NULL;
    PNET_BUFFER_LIST    pFirstDestinationNbl      = NULL;
    PNET_BUFFER         pSourceNb                 = NULL;
    PNET_BUFFER         pDestinationNb            = NULL;
    PMDL                pSourceMdl                = NULL;
    PMDL                pDestinationMdl           = NULL;
    PMDL                pFirstDestinationMdl      = NULL;
    PVOID               pMdlGag                   =
    NULL;

    if ((pFirstDestinationNbl = pDestinationNbl =
        NdisAllocateNetBufferList(g_NblPoolHandle, NULL, NULL)))
    {
        for
        (
            pSourceNbl = pNetBufferLists;
            pSourceNbl;

```

```

        pSourceNbl =
NET_BUFFER_LIST_NEXT_NBL(pSourceNbl)
    )
    {
        /*Дескриптор нужен для дальнейшего отслеживания
пакета*/
        pDestinationNbl->SourceHandle = g_NblPoolHandle;

        if ((pDestinationNb =
NET_BUFFER_LIST_FIRST_NB(pDestinationNbl) =
NdisAllocateNetBuffer(g_NbMdlPoolHandle, NULL, 0, 0)))
        {
            for
            (
                pSourceNb =
NET_BUFFER_LIST_FIRST_NB(pSourceNbl);
                pSourceNb;
                pSourceNb =
NET_BUFFER_NEXT_NB(pSourceNb)
            )
            {
                if
                (
                    pDestinationMdl =
NET_BUFFER_CURRENT_MDL(pDestinationNb) = NdisAllocateMdl
                    (
                        g_NbMdlPoolHandle,

                        NdisAllocateMemoryWithTagPriority(g_NbMdlPoolHandle,
NET_BUFFER_CURRENT_MDL(pSourceNb)->ByteCount,
NAT_NB_MDL_POOL_ALLOC_TAG, LowPoolPriority),

                        NET_BUFFER_CURRENT_MDL(pSourceNb)->ByteCount
                    )
                )
                {
                    for
                    (
                        pSourceMdl =
NET_BUFFER_CURRENT_MDL(pSourceNb);
                        pSourceMdl;
                        pSourceMdl = pSourceMdl-
>Next
                    )
                    {

                        NdisMoveMemory(pDestinationMdl->MappedSystemVa,
pSourceMdl->MappedSystemVa, pSourceMdl->ByteCount);

                        if (pSourceMdl->Next)
                        {

```

```

pDestinationMdl->Next = NdisAllocateMdl
(
    g_NbMdlPoolHandle,

    NdisAllocateMemoryWithTagPriority(g_NbMdlPoolHandle,
pSourceMdl->Next->ByteCount, NAT_NB_MDL_POOL_ALLOC_TAG,
LowPoolPriority),

    pSourceMdl->Next->ByteCount

);
}
else
{
    pDestinationMdl->Next =
NULL;
}

/*Сохранение указателя на Current
MDL Destination NetBuffer'a перед копированием данных из
Source NetBuffer'a*/
pDestinationMdl =
NET_BUFFER_CURRENT_MDL(pDestinationNb);

ULONG ulBytesCopied = 0;
NdisCopyFromNetBufferToNetBuffer
(
    pDestinationNb,
    pDestinationNb->DataOffset,
    pSourceNb->DataLength,
    pSourceNb,
    pSourceNb->DataOffset,
    &ulBytesCopied
);

/*Восстановление указателей на
CurrentMdl и MdlChain Destination NetBuffer'a*/
pDestinationNb->CurrentMdl =
pDestinationNb->MdlChain = pDestinationMdl;

for
(
    pSourceMdl =
NET_BUFFER_CURRENT_MDL(pSourceNb), pDestinationMdl =
NET_BUFFER_CURRENT_MDL(pDestinationNb);
    pSourceMdl;
    pSourceMdl = pSourceMdl->Next, pDestinationMdl = pDestinationMdl->Next
)
{

```

```

        NdisMoveMemory(pDestinationMdl->MappedSystemVa,
pSourceMdl->MappedSystemVa, pSourceMdl->ByteCount);
    }
}

    NET_BUFFER_DATA_LENGTH(pDestinationNb)
= NET_BUFFER_DATA_LENGTH(pSourceNb);
    NET_BUFFER_DATA_OFFSET(pDestinationNb)
= NET_BUFFER_DATA_OFFSET(pSourceNb);

    NET_BUFFER_CURRENT_MDL_OFFSET(pDestinationNb) =
NET_BUFFER_CURRENT_MDL_OFFSET(pSourceNb);

    if (NET_BUFFER_NEXT_NB(pSourceNb))
    {
        pDestinationNb =
NET_BUFFER_NEXT_NB(pDestinationNb) =
NdisAllocateNetBuffer(g_NbMdlPoolHandle, NULL, 0, 0);
    }
    else
    {
        NET_BUFFER_NEXT_NB(pDestinationNb) = NULL;
    }
}

    if (NET_BUFFER_LIST_NEXT_NBL(pSourceNbl))
    {
        pDestinationNbl =
NET_BUFFER_LIST_NEXT_NBL(pDestinationNbl) =
NdisAllocateNetBufferList(g_NblPoolHandle, 0, 0);
    }
    else
    {
        NET_BUFFER_LIST_NEXT_NBL(pDestinationNbl) =
NULL;
    }
}

    return pFirstDestinationNbl;
}

/*
* Освобождение скопированного NBL
*
* @param[in]    pSourceNbl        Освобождаемый pNBL
* @return       Нет возвращаемых значений
*/
VOID

```

```

FreePacket
(
    PNET_BUFFER_LIST pNetBufferLists
)
{
    PNET_BUFFER_LIST    pCurrentNbl    = NULL;
    PNET_BUFFER_LIST    pNextNbl    = NULL;
    PNET_BUFFER          pCurrentNb     = NULL;
    PNET_BUFFER          pNextNb       = NULL;
    PMDL                pCurrentMdl    = NULL;
    PMDL                pNextMdl      = NULL;

    for
    (
        pCurrentNbl = pNetBufferLists;
        pCurrentNbl;
        pCurrentNbl = pNextNbl
    )
    {
        pNextNbl = NET_BUFFER_LIST_NEXT_NBL(pCurrentNbl);

        for
        (
            pCurrentNb =
NET_BUFFER_LIST_FIRST_NB(pCurrentNbl);
            pCurrentNb;
            pCurrentNb = pNextNb
        )
        {
            pNextNb = NET_BUFFER_NEXT_NB(pCurrentNb);

            for
            (
                pCurrentMdl =
NET_BUFFER_CURRENT_MDL(pCurrentNb);
                pCurrentMdl;
                pCurrentMdl = pNextMdl
            )
            {
                pNextMdl = pCurrentMdl->Next;
                NdisFreeMdl(pCurrentMdl);
            }

            NdisFreeNetBuffer(pCurrentNb);
        }

        NdisFreeNetBufferList(pCurrentNbl);
    }
}

/*

```

* Проверка, является ли список NBL подходящим для модификации. Подходящим считается IPv4/ARP пакет, для которого была найдена соответствующая NAT-единица

```

*
* @param[in]      pSourceNbl      Проверяемый pNBL
* @param[in]      bPacketRoute    Флаг, является ли пакет
исходящим. TRUE - исходящий пакет, FALSE - входящий пакет
* @return          TRUE, если pNBL
подходящий
*/
BOOLEAN
IsPacketRelevant
(
    PNET_BUFFER_LIST    pNetBufferLists,
    BOOLEAN              bPacketRoute
)
{
    PNET_BUFFER_LIST    pCurrentNbl          = NULL;
    PNET_BUFFER          pCurrentNb           = NULL;
    PBYTE               pcDataBuffer          = NULL;
    ULONG               ulDataBufferLength    = 0;
    WORD                wEtherType           = 0;
    BOOLEAN              bIsRelevant          = FALSE;

    for
    (
        pCurrentNbl = pNetBufferLists;
        pCurrentNbl;
        pCurrentNbl = NET_BUFFER_LIST_NEXT_NBL(pCurrentNbl)
    )
    {
        for
        (
            pCurrentNb =
NET_BUFFER_LIST_FIRST_NB(pCurrentNbl);
            pCurrentNb;
            pCurrentNb = NET_BUFFER_NEXT_NB(pCurrentNb)
        )
        {
            do
            {
                ulDataBufferLength =
NET_BUFFER_DATA_LENGTH(pCurrentNb);
                if (ulDataBufferLength == 0)
                {
                    break;
                }

                pcDataBuffer =
NdisGetDataBuffer(pCurrentNb, ulDataBufferLength, NULL, 1, 0);
                if (!pcDataBuffer)
                {

```

```

        break;
    }

    wEtherType =
    GetReversedWord(*(PWORD) (pcDataBuffer + 12));
    if (wEtherType == IPV4_ETH_TYPE_CODE ||
wEtherType == ARP_ETH_TYPE_CODE)
    {
        if
        (IsNatTableUnitMasking(GetNatTableUnitByDataBuffer(pcDataBuffer, bPacketRoute)))
        {
            bIsRelevant = TRUE;
        }
    }

    } while (FALSE);
}

return bIsRelevant;
}

/*
* Проверка, является ли NBL скопированным
*
* @param[in]   pSourceNbl           Проверяемый pNBL
* @return      TRUE, если pNBL
скопированный
*/
BOOLEAN
IsPacketCopied
(
    PNET_BUFFER_LIST pNetBufferLists
)
{
    PNET_BUFFER_LIST    pCurrentNbl    = NULL;
    BOOLEAN              bIsCopied = FALSE;

    for
    (
        pCurrentNbl = pNetBufferLists;
        pCurrentNbl;
        pCurrentNbl = NET_BUFFER_LIST_NEXT_NBL(pCurrentNbl)
    )
    {
        if (pCurrentNbl->SourceHandle == g_NblPoolHandle)
        {
            bIsCopied = TRUE;
            break;
        }
    }
}

```

```

        return bIsCopied;
    }

```

Файл PacketModification.c

```

#include "StdAfx.c"

/*
 * Модификация пакета данных
 *
 * @param[in]      pSourceNbl      Модифицируемый pNBL
 * @param[in]      bPacketRoute   Флаг, является ли пакет
исходящим. TRUE - исходящий пакет, FALSE - входящий пакет
 * @return        Нет возвращаемых
значений
 */
VOID
ModificatePacket
(
    PNET_BUFFER_LIST    pNetBufferLists,
    BOOLEAN              bPacketRoute
)
{
    PSNATTableUnit      pstRelevantUnit      = NULL;
    PNET_BUFFER_LIST    pSourceNbl          = NULL;
    PNET_BUFFER         pSourceNb           = NULL;
    PBYTE               pcDataBuffer        = NULL;
    ULONG               ulDataBufferLength  = 0;
    WORD                wEtherType          = 0;
    BYTE                cProtocol           = 0;

    for
    (
        pSourceNbl = pNetBufferLists;
        pSourceNbl;
        pSourceNbl = NET_BUFFER_LIST_NEXT_NBL(pSourceNbl)
    )
    {
        for
        (
            pSourceNb =
NET_BUFFER_LIST_FIRST_NB(pSourceNbl);
            pSourceNb;
            pSourceNb = NET_BUFFER_NEXT_NB(pSourceNb)
        )
        {
            do
            {
                ulDataBufferLength =
NET_BUFFER_DATA_LENGTH(pSourceNb);

```

					ПГУ 1.090301.17.001 ПЗ	Лист
						136
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (ulDataBufferLength == 0)
        {
            break;
        }

        pcDataBuffer = NdisGetDataBuffer(pSourceNb,
ulDataBufferLength, NULL, 1, 0);
        if (!pcDataBuffer)
        {
            break;
        }

        pstRelevantUnit =
GetNatTableUnitByDataBuffer(pcDataBuffer, bPacketRoute);
        if (!pstRelevantUnit ||
!IsNatTableUnitMasking(pstRelevantUnit))
        {
            break;
        }

        wEtherType =
GetReversedWord(*(PWORD)(pcDataBuffer + ETH_TYPE_INDEX));
        if (wEtherType == IPV4_ETH_TYPE_CODE)
        {
            cProtocol = *(PBYTE)(pcDataBuffer +
IPV4_PROTOCOL_INDEX);

            if (bPacketRoute)
            {
                UpdateNatTableNextPacketId();

                if (cProtocol ==
TCP_PROTOCOL_CODE || cProtocol == UDP_PROTOCOL_CODE)
                {

                    UpdateTransportProtocolChecksum(pstRelevantUnit,
pcDataBuffer, bPacketRoute);

                    NatTableEnterCriticalSection();

                    /*Замена портов*/
                    if ((* (PWORD)(pcDataBuffer +
IPV4_PORT_SOURCE_INDEX)) == pstRelevantUnit->m_wLocalPort)
                    {
                        * (PWORD)(pcDataBuffer +
IPV4_PORT_SOURCE_INDEX) = (pstRelevantUnit->m_wGlobalPort);
                    }

                    NatTableLeaveCriticalSection();
                }
            }
        }
    }

```

```

UpdateIpv4HeaderChecksum(pstRelevantUnit, pcDataBuffer,
bPacketRoute);

NatTableEnterCriticalSection();

/*Замена ID*/
*(PWORD) (pcDataBuffer +
IPV4_PACKET_ID_INDEX) = GetReversedWord(g_pstNatTable-
>m_wNextPackageId);

/*Замена IP-адреса*/
if ((* (PDWORD) (pcDataBuffer +
IPV4_IP_SOURCE_INDEX)) == pstRelevantUnit->m_dwLocalIp)
{
    *(PDWORD) (pcDataBuffer +
IPV4_IP_SOURCE_INDEX) = (pstRelevantUnit->m_dwGlobalIp);
}

NatTableLeaveCriticalSection();
}
else
{
    if (cProtocol ==
TCP_PROTOCOL_CODE || cProtocol == UDP_PROTOCOL_CODE)
    {

UpdateTransportProtocolChecksum(pstRelevantUnit,
pcDataBuffer, bPacketRoute);

NatTableEnterCriticalSection();

/*Замена портов*/
if ((* (PWORD) (pcDataBuffer +
IPV4_PORT_DESTINATION_INDEX)) == pstRelevantUnit-
>m_dwGlobalIp)
{
    *(PWORD) (pcDataBuffer +
IPV4_PORT_DESTINATION_INDEX) = (pstRelevantUnit-
>m_wLocalPort);
}

NatTableLeaveCriticalSection();
}

UpdateIpv4HeaderChecksum(pstRelevantUnit, pcDataBuffer,
bPacketRoute);

```

```

        NatTableEnterCriticalSection();

        /*Замена IP-адреса*/
        if ((* (PDWORD) (pcDataBuffer +
IPV4_IP_DESTINATION_INDEX)) == pstRelevantUnit->m_dwGlobalIp)
        {
            * (PDWORD) (pcDataBuffer +
IPV4_IP_DESTINATION_INDEX) = (pstRelevantUnit->m_dwLocalIp);
        }

        NatTableLeaveCriticalSection();
    }
}
else if (wEtherType == ARP_ETH_TYPE_CODE)
{
    if (bPacketRoute)
    {
        NatTableEnterCriticalSection();

        /*Замена IP-адреса*/
        if ((* (PDWORD) (pcDataBuffer +
ARP_IP_SOURCE_INDEX)) == pstRelevantUnit->m_dwLocalIp)
        {
            * (PDWORD) (pcDataBuffer +
ARP_IP_SOURCE_INDEX) = (pstRelevantUnit->m_dwGlobalIp);
        }

        NatTableLeaveCriticalSection();
    }
    else
    {
        NatTableEnterCriticalSection();

        /*Замена IP-адреса*/
        if ((* (PDWORD) (pcDataBuffer +
ARP_IP_DESTINATION_INDEX)) == pstRelevantUnit->m_dwGlobalIp)
        {
            * (PDWORD) (pcDataBuffer +
ARP_IP_DESTINATION_INDEX) = (pstRelevantUnit->m_dwLocalIp);
        }

        NatTableLeaveCriticalSection();
    }
}

    } while (FALSE);
}

}

/*
* Замена слагаемого Checksum

```

```

*
* @param[in]    dwChecksum                Значение Checksum
* @param[in]    dwValueToSubtract        Вычитаемый параметр
* @param[in]    dwValueToAdd             Прибавляемый параметр
* @return                               Новое значение Checksum
*/
DWORD
ReplaceChecksumParameter
(
    DWORD    dwChecksum,
    DWORD    dwValueToSubtract,
    DWORD    dwValueToAdd
)
{
    /*Пока вычитаемый параметр больше значения Checksum*/
    while (dwValueToSubtract > dwChecksum)
    {
        /*Создаётся переполнение*/
        dwChecksum = CreateWordOverflow(dwChecksum);
    }

    dwChecksum -= dwValueToSubtract;
    dwChecksum += dwValueToAdd;

    return dwChecksum;
}

/*
* Обновление значения контрольной суммы в заголовке IPv4 в
соответствии с измененными данными в пакете
*/
* @param[in]    pstUnit                  Указатель на NAT-
единицу
* @param[in]    pcDataBuffer             Указатель на DataBuffer
* @param[in]    bPacketRoute            Флаг исходящего пакета. TRUE
- исходящий пакет, FALSE - входящий пакет
* @return                               Нет возвращаемых
значений
*/
VOID
UpdateIpv4HeaderChecksum
(
    PSNatTableUnit pstUnit,
    PBYTE           pcDataBuffer,
    BOOLEAN         bPacketRoute
)
{
    DWORD dwHeaderChecksum =
(WORD) (~GetReversedWord(*(PWORD) (pcDataBuffer +
IPV4_HEADER_CHECKSUM_INDEX)));

    if (bPacketRoute)

```

```

{
    NatTableEnterCriticalSection();

    /*Т.к. в исходящем пакете обновляется ID*/
    dwHeaderChecksum = ReplaceChecksumParameter
    (
        dwHeaderChecksum,
        GetReversedWord(*(PWORD) (pcDataBuffer +
IPV4_PACKET_ID_INDEX)),          // Дефолтный ID
        g_pstNatTable->m_wNextPackageId
                                   // Новый ID
    );

    /*Замена Source IP: Local -> Global*/
    dwHeaderChecksum = ReplaceChecksumParameter
    (
        dwHeaderChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_IP_SOURCE_INDEX)),          // Дефолтный Source IP
        pstUnit->m_dwGlobalIp
                                   // Новый Source IP
    );

    NatTableLeaveCriticalSection();
}
else
{
    NatTableEnterCriticalSection();

    /*Замена Destination IP: Global -> Local*/
    dwHeaderChecksum = ReplaceChecksumParameter
    (
        dwHeaderChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_IP_DESTINATION_INDEX)),    // Дефолтный Destination IP
        pstUnit->m_dwLocalIp
                                   // Новый Destination IP
    );

    NatTableLeaveCriticalSection();
}

/*Преобразование результата для записи в пакет*/
dwHeaderChecksum = NeglateWordOverflow(dwHeaderChecksum);
dwHeaderChecksum = (WORD) (~dwHeaderChecksum);

/*Запись нового значения Header Checksum в пакет*/
*(PWORD) (pcDataBuffer + IPV4_HEADER_CHECKSUM_INDEX) =
GetReversedWord( (WORD) dwHeaderChecksum);
}

/*

```

```

* Запрос индекса, по которому находится контрольная сумма в
* протоколах TCP и UDP
*
* @param[in]   cProtocol Протокол пакета данных
* @return      Индекс, по которому находится
* Protocol Checksum в DataBuffer
*/
BYTE
GetTransportProtocolChecksumIndex
(
    BYTE cProtocol
)
{
    BYTE bChecksumIndex = 0;

    if (cProtocol == TCP_PROTOCOL_CODE)
    {
        bChecksumIndex = TCP_CHECKSUM_INDEX;
    }
    else if (cProtocol == UDP_PROTOCOL_CODE)
    {
        bChecksumIndex = UDP_CHECKSUM_INDEX;
    }

    return bChecksumIndex;
}

/*
* Обновление значения контрольной суммы в заголовке TCP или
* UDP в соответствии с измененными данными в пакете
*
* @param[in]   pstUnit          Указатель на NAT-
*                               единицу
* @param[in]   pcDataBuffer     Указатель на DataBuffer
* @param[in]   bPacketRoute     Флаг исходящего пакета. TRUE
*                               - исходящий пакет, FALSE - входящий пакет
* @return      Нет возвращаемых
* значений
*/
VOID
UpdateTransportProtocolChecksum
(
    PSNatTableUnit pstUnit,
    PBYTE          pcDataBuffer,
    BOOLEAN        bPacketRoute
)
{
    BYTE cProtocol = *(PBYTE) (pcDataBuffer +
    IPV4_PROTOCOL_INDEX);
    DWORD dwProtocolChecksum =
    (WORD) (~GetReversedWord(*(PWORD) (pcDataBuffer +
    GetTransportProtocolChecksumIndex(cProtocol))));
}

```

```

if (bPacketRoute)
{
    NatTableEnterCriticalSection();

    /*Замена Source IP: Local -> Global*/
    dwProtocolChecksum = ReplaceChecksumParameter
    (
        dwProtocolChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_IP_SOURCE_INDEX)),          // Дефолтный Source IP
        pstUnit->m_dwGlobalIp
                                   // Новый Source IP
    );

    /*Замена Source port: Local -> Global*/
    dwProtocolChecksum = ReplaceChecksumParameter
    (
        dwProtocolChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_PORT_SOURCE_INDEX)),          // Дефолтный Source
port
        pstUnit->m_wGlobalPort
                                   // Новый Source
port
    );

    NatTableLeaveCriticalSection();
}
else
{
    NatTableEnterCriticalSection();

    /*Замена Destination IP: Global -> Local*/
    dwProtocolChecksum = ReplaceChecksumParameter
    (
        dwProtocolChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_IP_DESTINATION_INDEX)),      // Дефолтный Destination IP
        pstUnit->m_dwLocalIp
                                   // Новый
Destination IP
    );

    /*Замена Destination port: Global -> Local*/
    dwProtocolChecksum = ReplaceChecksumParameter
    (
        dwProtocolChecksum,
        GetReversedDword(*(PDWORD) (pcDataBuffer +
IPV4_PORT_DESTINATION_INDEX)),    // Дефолтный Destination
port

```

```

        pstUnit->m_wLocalPort
// Новый
Destination port
    );

    NatTableLeaveCriticalSection();
}

/*Преобразование результата для записи в пакет*/
dwProtocolChecksum =
NeglateWordOverflow(dwProtocolChecksum);
dwProtocolChecksum = (WORD) (~dwProtocolChecksum);

/*Запись нового значения Protocol Checksum в пакет*/
*(PWORD) (pcDataBuffer +
GetTransportProtocolChecksumIndex(cProtocol)) =
GetReversedWord( (WORD) dwProtocolChecksum);
}

/*
* Создание переполнения в WORD
*
* @param[in]    dwSource    Исходное значение
* @return              Преобразованное значение
*/
DWORD
CreateWordOverflow
(
    DWORD dwSource
)
{
    /*Младший разряд WORD (1-й байт) уменьшается на 1, а
    разряд переполнения (5-й байт) увеличивается на 1*/
    DWORD dwResult = 1 << 16;
    dwResult += dwSource - 1;

    return dwResult;
}

/*
* Невелирование переполнения в WORD
*
* @param[in]    dwSource    Исходное значение
* @return              Преобразованное значение
*/
WORD
NeglateWordOverflow
(
    DWORD dwSource
)
{

```

```

        /*Разряд переполнения WORD (5-й байт) складывается с
        младшим разрядом WORD (1-й байт)*/
        WORD wResult = (WORD)dwSource;
        wResult += dwSource >> 16;

        return wResult;
    }

    /*
    * Запись WORD в обратном порядке
    *
    * @param[in]    wSource        Исходное значение
    * @return        Преобразованное значение
    */
    WORD
    GetReversedWord
    (
        WORD wSource
    )
    {
        WORD wBuffer = 0;

        wBuffer += wSource << 8;
        wBuffer += wSource >> 8;

        return wBuffer;
    }

    /*
    * Запись DWORD в обратном порядке
    *
    * @param[in]    dwSource    Исходное значение
    * @return        Преобразованное значение
    */
    DWORD
    GetReversedDword
    (
        DWORD dwSource
    )
    {
        DWORD dwBuffer = 0;

        dwBuffer += GetReversedWord((WORD)dwSource);
        dwBuffer = dwBuffer << 16;
        dwBuffer += GetReversedWord((WORD)(dwSource >> 16));

        return dwBuffer;
    }

```

Приложение Б.2 – DLL для взаимодействия драйвера-фильтра и пользовательского приложения

Файл NatDll.cpp

```
#include "StdAfx.h"

/*
 * Создание дескриптора для взаимодействия с драйвером
 *
 * @param Нет заданных параметров
 * @return Дескриптор драйвера
 */
NAT_API
HANDLE
DllNatOpen
(
    VOID
)
{
    HANDLE hDriver = CreateFile
    (
        NAT_DRIVER_NAME_RING3,
        GENERIC_WRITE | GENERIC_READ,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL
    );

    if (hDriver == INVALID_HANDLE_VALUE)
    {
        return (int*)GetLastError();
    }

    return hDriver;
}

/*
 * Закрытие дескриптора драйвера
 *
 * @param[in] hDriver Дескриптор драйвера
 * @return TRUE, если дескриптор драйвера
    закрыт
 */
NAT_API
BOOLEAN
DllNatClose
(
```

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		146

```

        HANDLE hDriver
    )
    {
        BOOLEAN bStatus = FALSE;

        if (hDriver != 0)
        {
            bStatus = CloseHandle(hDriver);
        }

        return bStatus;
    }

/*
* Запрос на создание NAT-единицы
*
* @param[in]    hDriver                Дескриптор драйвера
* @param[in]    pcDataBuffer            Буффер данных (IP, порты)
* @param[in]    iDataBufferSize        Размер буффера данных
* @return       TRUE, если NAT-единица
создана
*/
NAT_API
BOOLEAN
DllNatCreateUnit
(
    HANDLE        hDriver,
    PBYTE         pcDataBuffer,
    int            iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_CREATE_UNIT,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

/*
* Запрос на удаление NAT-единицы
*
* @param[in]    hDriver                Дескриптор драйвера
* @param[in]    pcDataBuffer            Буффер данных (IP, порты)
* @param[in]    iDataBufferSize        Размер буффера данных

```

```

* @return                                     TRUE, если NAT-единица
удалена
*/
NAT_API
BOOLEAN
DllNatRemoveUnit
(
    HANDLE      hDriver,
    PBYTE       pcDataBuffer,
    int         iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_REMOVE_UNIT,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

/*
* Запрос на очистку всего списка NAT-единиц
*
* @param[in]  hDriver          Дескриптор драйвера
* @param[in]  pcDataBuffer     Буффер данных (IP, порты)
* @param[in]  iDataBufferSize  Размер буффера данных
* @return     TRUE, если список
очищен
*/
NAT_API
BOOLEAN
DllNatWipeUnitList
(
    HANDLE hDriver
)
{
    BOOLEAN bStatus;
    bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_WIPE_UNIT_LIST,
        NULL,
        NULL,
        NULL,
        0,
        NULL,
        NULL
    );
}

```

```

        NULL,
        NULL
    );

    return bStatus;
}

/*
 * Запрос на проверку существования NAT-единицы
 *
 * @param[in]   hDriver           Дескриптор драйвера
 * @param[in]   pcDataBuffer      Буффер данных (IP, порты)
 * @param[in]   iDataBufferSize  Размер буффера данных
 * @return      TRUE, если NAT-единица
                существует
 */
NAT_API
BOOLEAN
DllNatIsUnitExists
(
    HANDLE      hDriver,
    PBYTE       pcDataBuffer,
    int         iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_IS_UNIT_EXISTS,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

/*
 * Запрос на проверку модификации пакетов NAT-единицей
 *
 * @param[in]   hDriver           Дескриптор драйвера
 * @param[in]   pcDataBuffer      Буффер данных (IP, порты)
 * @param[in]   iDataBufferSize  Размер буффера данных
 * @return      TRUE, если NAT-единица
                модифицирует данные
 */
NAT_API
BOOLEAN
DllNatIsUnitRunning

```

```

(
    HANDLE    hDriver,
    PBYTE     pcDataBuffer,
    int       iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_IS_UNIT_RUNNING,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

/*
* Запрос на приостановку работы NAT-единицы
*
* @param[in]    hDriver            Дескриптор драйвера
* @param[in]    pcDataBuffer        Буффер данных (IP, порты)
* @param[in]    iDataBufferSize    Размер буффера данных
* @return       TRUE, если работа NAT-
единицы приостановлена
*/
NAT_API
BOOLEAN
DllNatSuspendUnit
(
    HANDLE    hDriver,
    PBYTE     pcDataBuffer,
    int       iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_SUSPEND_UNIT,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

```

```

}

/*
* Запрос на возобновление работы NAT-единицы
*
* @param[in]   hDriver           Дескриптор драйвера
* @param[in]   pcDataBuffer      Буффер данных (IP, порты)
* @param[in]   iDataBufferSize  Размер буффера данных
* @return      TRUE, если работа NAT-
единицы возобновлена
*/
NAT_API
BOOLEAN
DllNatResumeUnit
(
    HANDLE      hDriver,
    PBYTE       pcDataBuffer,
    int         iDataBufferSize
)
{
    BOOLEAN bStatus = DeviceIoControl
    (
        hDriver,
        IOCTL_NAT_RESUME_UNIT,
        pcDataBuffer,
        iDataBufferSize,
        NULL,
        0,
        NULL,
        NULL
    );

    return bStatus;
}

```

Приложение Б.3 – Пользовательское приложение NAT

Файл CreationForm.cs

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

namespace NatGui
{
    public partial class CreationForm : Form
    {
        private MainForm m_Parent_Control;

        public CreationForm(MainForm Parent_Control)
        {
            InitializeComponent();
            StyleGui();

            m_Parent_Control = Parent_Control;
        }

        void StyleGui()
        {
            _LocalIpTitle_Label.Font = Common.m_Font_Cell;

            _LocalIpContent_TextBox.Font = Common.m_Font_Cell;

            _GlobalIpTitle_Label.Font = Common.m_Font_Cell;

            _GlobalIpContent_TextBox.Font =
Common.m_Font_Cell;

            _LocalPortTitle_Label.Font = Common.m_Font_Cell;

            _LocalPortContent_TextBox.Font =
Common.m_Font_Cell;

            _GlobalPortTitle_Label.Font = Common.m_Font_Cell;

            _GlobalPortContent_TextBox.Font =
Common.m_Font_Cell;

            _Create_Button.Font = Common.m_Font_Buttons;
            _Create_Button.Click += CreateButtonClick;
        }
    }
}
```

```

    }

    void CreateButtonClick(object sender, EventArgs e)
    {
        do
        {
            if
            (!DataProcess.IsStringIsIpv4Address(_LocalIpContent_TextBox.Te
            xt))
            {
                MessageBox.Show("Некорректно указан
                локальный IP");
                break;
            }

            if
            (!DataProcess.IsStringIsIpv4Address(_GlobalIpContent_TextBox.T
            ext))
            {
                MessageBox.Show("Некорректно указан
                глобальный IP");
                break;
            }

            if
            (!DataProcess.IsStringIsPort(_LocalPortContent_TextBox.Text))
            {
                MessageBox.Show("Некорректно указан
                локальный порт");
                break;
            }

            if
            (!DataProcess.IsStringIsPort(_GlobalPortContent_TextBox.Text))
            {
                MessageBox.Show("Некорректно указан
                глобальный порт");
                break;
            }

            m_Parent_Control.CreateUnitRequest(_LocalIpContent_TextBox.Tex
            t, _GlobalIpContent_TextBox.Text,
            _LocalPortContent_TextBox.Text,
            _GlobalPortContent_TextBox.Text);

            this.Close();

        } while (false);
    }
}

```

Файл MainForm.cs

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

namespace NatGui
{
    public partial class MainForm : Form
    {
        DriverApi _DriverApi;

        public MainForm()
        {
            InitializeComponent();
            StyleGui();

            _DriverApi = new DriverApi();
            this.Text =
            _DriverApi.m_hDriver.ToInt32().ToString();
        }

        public void StyleGui()
        {
            SuspendLayout();

            Size = MinimumSize = MaximumSize =
            Common.m_MainForm_Size;
            Margin = new Padding(0);
            Padding = new Padding(0);

            Info_Panel.SuspendLayout();

            Info_Panel.Location = new Point(3, 0);
            Info_Panel.Size = new Size(180, 75);
            Info_Panel.BackColor = Common.m_Color_Primary;
            Info_Panel.Margin = new Padding(0);
            Info_Panel.Padding = new Padding(0);

            Icon_PictureBox.Location = new Point(5, 5);
            Icon_PictureBox.Size = new Size(65, 65);
            Icon_PictureBox.Image =
            Image.FromFile("qpos.png");

            Os_Label.Text = "QP OC";
        }
    }
}
```

```

        Os_Label.Location = new Point(70, 5);
        Os_Label.Font = Common.m_Font_Os;

        App_Label.Text = "NAT";
        App_Label.Location = new Point(95, 38);
        App_Label.Font = Common.m_Font_App;

        Info_Panel.ResumeLayout(false);

        NatTableButtons_Panel.Location = new Point(0, 75);
        NatTableButtons_Panel.Size = new Size(180,
Height);
        NatTableButtons_Panel.BackColor =
Common.m_Color_Primary;
        NatTableButtons_Panel.Margin = new Padding(0);
        NatTableButtons_Panel.Padding = new Padding(0);

        ButtonStyle(CreateUnit_Button);
        CreateUnit_Button.Location = new Point(0, 0);
        CreateUnit_Button.Text = "          Создать\n
NAT-интерфейс";
        CreateUnit_Button.Image =
Image.FromFile("plus.png");
        CreateUnit_Button.Visible = true;
        CreateUnit_Button.Click += CreateUnitButtonClick;

        ButtonStyle(RemoveUnit_Button);
        RemoveUnit_Button.Location = new
Point(CreateUnit_Button.Location.X,
CreateUnit_Button.Location.Y + CreateUnit_Button.Height);
        RemoveUnit_Button.Text = "          Удалить\n
NAT-интерфейс";
        RemoveUnit_Button.Image =
Image.FromFile("minus.png");
        RemoveUnit_Button.Visible = true;
        RemoveUnit_Button.Click += RemoveUnitButtonClick;

        ButtonStyle(UnitStartMasking_Button);
        UnitStartMasking_Button.Location = new
Point(RemoveUnit_Button.Location.X,
RemoveUnit_Button.Location.Y + RemoveUnit_Button.Height);
        UnitStartMasking_Button.Text = "
Возобновить\n          NAT-интерфейс";
        UnitStartMasking_Button.Image =
Image.FromFile("play.png");
        UnitStartMasking_Button.Visible = true;
        UnitStartMasking_Button.Click +=
ResumeUnitButtonClick;

        ButtonStyle(UnitStopMasking_Button);
        UnitStopMasking_Button.Location = new
Point(UnitStartMasking_Button.Location.X,

```

```

UnitStartMasking_Button.Location.Y +
UnitStartMasking_Button.Height);
        UnitStopMasking_Button.Text = "
Приостановить\н        NAT-интерфейс";
        UnitStopMasking_Button.Image =
Image.FromFile("pause.png");
        UnitStopMasking_Button.Visible = true;
        UnitStopMasking_Button.Click +=
SuspendUnitButtonClick;

        NatTableButtons_Panel.ResumeLayout(false);

        /*DataGridView*/
        DataGridViewStyle(NatTable_DataGridView);
        NatTable_DataGridView.Location = new
Point(NatTableButtons_Panel.Location.X +
NatTableButtons_Panel.Width + 5, 45);
        NatTable_DataGridView.Size = new Size(Width -
NatTableButtons_Panel.Width - 5 - Common.m_Size_Borders.Width,
NatTableButtons_Panel.Height + NatTableButtons_Panel.Height);

        ResumeLayout(false);

void ButtonStyle(Button _Button)
{
    _Button.Size = new Size(180, 50);
    _Button.BackColor = Color.Transparent;
    _Button.FlatStyle = FlatStyle.Flat;
    _Button.FlatAppearance.BorderSize = 0;
    _Button.Margin = new Padding(0);
    _Button.Padding = new Padding(0);
    _Button.Font = Common.m_Font_Buttons;
    _Button.TextAlign =
ContentAlignment.MiddleLeft;
    _Button.ImageAlign =
ContentAlignment.MiddleLeft;
}

void DataGridViewStyle(DataGridView _DataGridView)
{
    _DataGridView.AllowUserToAddRows = false;
    _DataGridView.AllowUserToDeleteRows = false;
    _DataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
    _DataGridView.BackgroundColor = Color.White;
    _DataGridView.BorderStyle = BorderStyle.None;
    _DataGridView.CellBorderStyle =
DataGridViewCellBorderStyle.None;
    _DataGridView.ColumnHeadersBorderStyle =
DataGridViewHeaderBorderStyle.None;
    _DataGridView.ReadOnly = true;
}

```

```

        _DataGridView.RowHeadersBorderStyle =
DataGridViewHeaderBorderStyle.None;
        _DataGridView.RowHeadersVisible = false;
        _DataGridView.RowTemplate.DividerHeight = 1;
        _DataGridView.RowTemplate.Height = 40;
        _DataGridView.RowTemplate.ReadOnly = true;
        _DataGridView.RowTemplate.Resizable =
DataGridViewTriState.False;
        _DataGridView.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

        DataGridViewCellStyle ColumnHeaderCellStyle =
new DataGridViewCellStyle();
        DataGridViewCellStyle CellStyle = new
DataGridViewCellStyle();

        CellStyle.Font = Common.m_Font_Cell;
        CellStyle.BackColor = Common.m_Color_Primary;
        CellStyle.ForeColor = Common.m_Color_Text;
        CellStyle.SelectionBackColor =
Common.m_Color_Secondary;
        CellStyle.SelectionForeColor =
Common.m_Color_Text;
        CellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;

        _DataGridView.DefaultCellStyle = CellStyle;

        ColumnHeaderCellStyle.Font =
Common.m_Font_ColumnHeader;
        ColumnHeaderCellStyle.BackColor =
Common.m_Color_Primary;
        ColumnHeaderCellStyle.ForeColor =
Common.m_Color_Text;
        ColumnHeaderCellStyle.SelectionBackColor =
Common.m_Color_Primary;
        ColumnHeaderCellStyle.SelectionForeColor =
Common.m_Color_Text;
        ColumnHeaderCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;

        _DataGridView.ColumnHeadersDefaultCellStyle =
ColumnHeaderCellStyle;
        _DataGridView.ColumnHeadersHeight = 50;

        _DataGridView.EnableHeadersVisualStyles =
false;
    }

    void CreateUnitButtonClick(object sender,
EventArgs e)
    {

```

```

        Thread CreationThread = new Thread(delegate ()
{ Application.Run(new CreationForm(this)); });
        CreationThread.Start();
    }

    void RemoveUnitButtonClick(object sender,
EventArgs e)
    {
        RemoveUnitRequest();
    }

    void SuspendUnitButtonClick(object sender,
EventArgs e)
    {
        SuspendUnitRequest();
    }

    void ResumeUnitButtonClick(object sender,
EventArgs e)
    {
        ResumeUnitRequest();
    }
}

    public void CreateUnitRequest(string strLocalIp,
string strGlobalIp, string strLocalPort, string strGlobalPort)
    {
        byte[] bdataArray =
DataProcess.ConvertUnitInfoToByteArray(strLocalIp,
strGlobalIp, strLocalPort, strGlobalPort);

        if (_DriverApi.CreateUnit(bdataArray))
        {
            string strUnitStatus = "";

            if(_DriverApi.IsUnitRunning(bdataArray))
            {
                strUnitStatus = "Работает";
            }
            else
            {
                strUnitStatus = "Приостановлен";
            }

            NatTable_DataGridView.Invoke((MethodInvoker)delegate
            {
                NatTable_DataGridView.Rows.Add(strLocalIp,
strGlobalIp, strLocalPort, strGlobalPort, strUnitStatus);
            });
        }
    }
}

```

```

        public void RemoveUnitRequest()
        {
            DataGridViewRow SelectedRow =
NatTable_DataGridView.SelectedRows[0];

            byte[] bdataArray =
DataProcess.ConvertUnitInfoToByteArray
            (
                SelectedRow.Cells[0].Value.ToString(),
                SelectedRow.Cells[1].Value.ToString(),
                SelectedRow.Cells[2].Value.ToString(),
                SelectedRow.Cells[3].Value.ToString()
            );

            if (_DriverApi.IsUnitExists(bdataArray))
            {
                if(_DriverApi.RemoveUnit(bdataArray))
                {

NatTable_DataGridView.Rows.Remove(SelectedRow);
                }
            }

            public void SuspendUnitRequest()
            {
                DataGridViewRow SelectedRow =
NatTable_DataGridView.SelectedRows[0];

                byte[] bdataArray =
DataProcess.ConvertUnitInfoToByteArray
                (
                    SelectedRow.Cells[0].Value.ToString(),
                    SelectedRow.Cells[1].Value.ToString(),
                    SelectedRow.Cells[2].Value.ToString(),
                    SelectedRow.Cells[3].Value.ToString()
                );

                if (_DriverApi.IsUnitRunning(bdataArray))
                {
                    _DriverApi.SuspendUnit(bdataArray);
                    SelectedRow.Cells[4].Value = "Приостановлен";
                }
            }

            public void ResumeUnitRequest()
            {
                DataGridViewRow SelectedRow =
NatTable_DataGridView.SelectedRows[0];

```

```

        byte[] bdataArray =
DataProcess.ConvertUnitInfoToByteArray
    (
        SelectedRow.Cells[0].Value.ToString(),
        SelectedRow.Cells[1].Value.ToString(),
        SelectedRow.Cells[2].Value.ToString(),
        SelectedRow.Cells[3].Value.ToString()
    );

    if (!_DriverApi.IsUnitRunning(bdataArray))
    {
        _DriverApi.ResumeUnit(bdataArray);
        SelectedRow.Cells[4].Value = "Работает";
    }
}
}
}

```

Файл DriverApi.cs

```

using System;
using System.Runtime.InteropServices;

namespace NatGui
{
    public class DriverApi
    {
        /*Дескриптор драйвера*/
        public IntPtr m_hDriver;

        [DllImport("NatDll.dll")]
        public static extern
        IntPtr DllNatOpen();

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatClose(IntPtr hDriver);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatCreateUnit(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatRemoveUnit(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatWipeUnitList(IntPtr hDriver);
    }
}

```

```

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatIsUnitExists(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatIsUnitRunning(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatSuspendUnit(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        [DllImport("NatDll.dll")]
        public static extern
        bool DllNatResumeUnit(IntPtr hDriver, byte[]
dataArray, int dataArrayLength);

        /*
        * Дефолтный конструктор класса NatDriver
        *
        * @param Нет заданных параметров
        */
        public DriverApi()
        {
            m_hDriver = DllNatOpen();
        }

        /*
        * Дефолтный деструктор класса NatDriver
        *
        * @param Нет заданных параметров
        */
        ~DriverApi()
        {
            DllNatClose(m_hDriver);
        }

        /*
        * Создание нового интерфейса на стороне драйвера
        *
        * @param[in]      dataArray      Массив данных о
интерфейсе
        * @return          TRUE, если интерфейс
был создан
        */
        public bool CreateUnit(byte[] dataArray)
        {
            return DllNatCreateUnit(m_hDriver, dataArray,
dataArray.Length);

```

```

    }

    /*
    * Удаление интерфейса на стороне драйвера
    *
    * @param[in]    DataArray    Массив данных о
интерфейсе
    * @return                                TRUE, если интерфейс
был удалён
    */
    public bool RemoveUnit(byte[] DataArray)
    {
        return DllNatRemoveUnit(m_hDriver, DataArray,
DataArray.Length);
    }

    /*
    * Проверка, существует ли интерфейс на стороне
драйвера
    *
    * @param[in]    DataArray    Массив данных о
интерфейсе
    * @return                                TRUE, если интерфейс
существует
    */
    public bool IsUnitExists(byte[] DataArray)
    {
        return DllNatIsUnitExists(m_hDriver, DataArray,
DataArray.Length);
    }

    /*
    * Проверка, активен ли интерфейс
    *
    * @param[in]    DataArray    Массив данных о
интерфейсе
    * @return                                TRUE, если интерфейс
активен
    */
    public bool IsUnitRunning(byte[] DataArray)
    {
        return DllNatIsUnitRunning(m_hDriver, DataArray,
DataArray.Length);
    }

    /*
    * Приостановка работы интерфейса
    *
    * @param[in]    DataArray    Массив данных о
интерфейсе
    * @return                                TRUE, если работа
интерфейса приостановлена

```

```

        */
        public bool SuspendUnit(byte[] DataArray)
        {
            return DllNatSuspendUnit(m_hDriver, DataArray,
DataArray.Length);
        }

        /*
        * Возобновление работы интерфейса
        *
        * @param[in]      DataArray      Массив данных о
интерфейсе
        * @return          TRUE, если работа
интерфейса возобновлена
        */
        public bool ResumeUnit(byte[] DataArray)
        {
            return DllNatResumeUnit(m_hDriver, DataArray,
DataArray.Length);
        }
    }
}

```

Приложение В

(Обязательное)

Программное обеспечение для преобразования сетевых адресов

Презентация

					ПГУ 1.090301.17.001 ПЗ	Лист
						164
Изм.	Лист	№ докум.	Подпись	Дата		

Программное обеспечение для преобразования сетевых адресов

Автор работы:

студент группы 18ВВ1 Колобов Кирилл Андреевич

Научный руководитель:

кандидат точных наук, доцент Дорошенко Иван Николаевич

Пенза, 2022

Постановка задачи

Цель работы: Разработка программного обеспечения для преобразования сетевых адресов с типом преобразования РАТ.

Задачи, решённые в процессе выполнения работы:

- изучение основ устройства сетей TCP/IP;
- изучение основ разработки ПО на системном уровне;
- разработка сетевого драйвера-фильтра NDIS;
- разработка пользовательского приложения;
- обеспечение взаимодействия между модулями ПО на системном и пользовательском уровне.

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		165

Актуальность работы

Израсходование IPv4 адресов считается проблемой с 1980-го года 20 века.

Для разрешения проблемы была разработана новая версия протокола IP - IPv6.

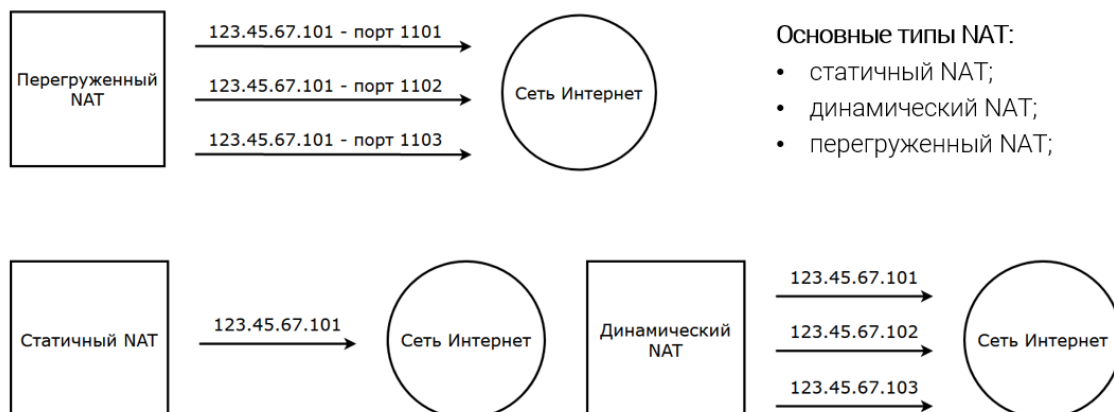
Из-за несовместимости между протоколами и прочих технических сложностей, внедрение IPv6 и по сей день продвигается не самыми быстрыми темпами.

В качестве временного, но вместе с тем более оперативного выхода из положения было представлено программное обеспечение для преобразования сетевых адресов.

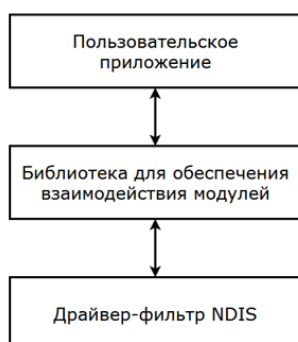
Что такое NAT?



Разновидности NAT



Архитектура программного обеспечения



Для проектирования общей архитектуры проекта был выбран шаблон, называемый многоуровневой архитектурой.

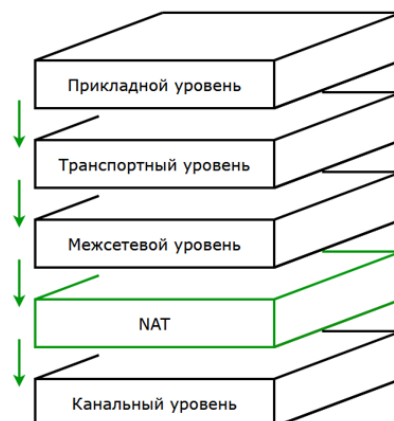
Данный шаблон подразумевает разбиение ПО на модули, каждый из которых выполняет определенную функцию и взаимодействует только с «соседними» модулями.

Разработка драйвера-фильтра NDIS



Информация в пакете протокола ARP

```
60 32 b1 16 a7 05 48 e7 da 57 9a 55 08 06 00 01
08 00 06 04 00 02 48 e7 da 57 9a 55 c0 a8 00 6f
60 32 b1 16 a7 05 c0 a8 00 01
```



Разработка пользовательского приложения

QP OC NAT

	Локальный IP	Глобальный IP	Локальный порт	Глобальный п...	Статус
+ Создать NAT-интерфейс	192.168.0.1	192.168.0.2	1	2	Работает
- Удалить NAT-интерфейс					
▷ Возобновить NAT-интерфейс					
⏏ Приостановить NAT-интерфейс					

При создании элемента таблицы используется дополнительная форма.

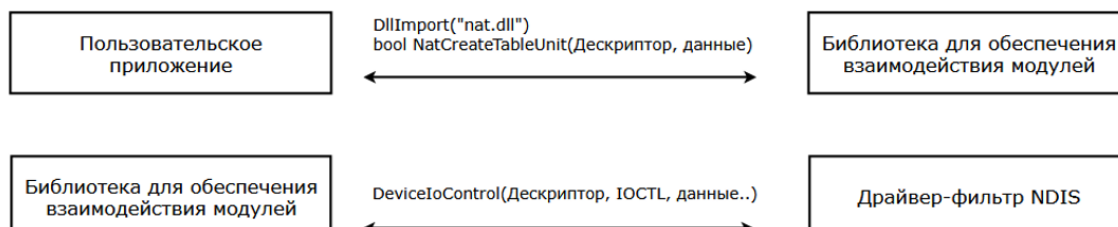
NAT-таблица представляет собой список элементов, в соответствии с которыми происходит модификация данных.

Локальный IP	<input type="text"/>
Глобальный IP	<input type="text"/>
Локальный порт	<input type="text"/>
Глобальный порт	<input type="text"/>
<input type="button" value="Создать"/>	

Разработка библиотеки

Взаимодействие с драйвером происходит с помощью IOCTL запросов, применение которых возможно за счёт использования функционала Windows API.

Библиотека служит для обеспечения взаимодействия между пользовательским приложением и драйвером-фильтром, так как они созданы с помощью языков программирования C# и Си соответственно.



Результаты тестирования

867	511.793468	192.168.11.13	3.1.43.9	ICMP	74 Echo (ping) request
868	511.793891	3.1.43.9	192.168.11.13	ICMP	74 Echo (ping) reply
869	512.796368	192.168.11.13	3.1.43.9	ICMP	74 Echo (ping) request
870	512.796815	3.1.43.9	192.168.11.13	ICMP	74 Echo (ping) reply

Сетевой трафик до преобразования сетевых адресов

```
0000 00 50 56 f4 b2 6a 00 0c 29 07 16 7b 08 00 45 00
0010 00 3c 01 3e 00 00 80 01 3f c4 c0 a8 0b 0d 03 01
0020 2b 09 08 00 df ee 00 00 17 00 01 02 03 04 05 06
0030 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16
0040 17 18 19 1a 1b 1c 1d 1e 1f 20
```

891	547.153755	1.3.3.7	3.1.43.9	ICMP	74 Echo (ping) request
892	547.154174	3.1.43.9	1.3.3.7	ICMP	74 Echo (ping) reply
893	548.156126	1.3.3.7	3.1.43.9	ICMP	74 Echo (ping) request
894	548.156557	3.1.43.9	1.3.3.7	ICMP	74 Echo (ping) reply

Сетевой трафик после преобразования сетевых адресов

```
0000 00 50 56 f4 b2 6a 00 0c 29 07 16 7b 08 00 45 00
0010 00 3c 01 37 00 00 80 01 07 77 01 03 03 07 03 01
0020 2b 09 08 00 da ee 00 00 1c 00 01 02 03 04 05 06
0030 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16
0040 17 18 19 1a 1b 1c 1d 1e 1f 20
```

Итоги работы

В результате выполнения выпускной квалификационной работы было разработано программное обеспечение для преобразования сетевых адресов с типом преобразования PAT для двух операционных систем: MS Windows и QP ОС.

В настоящее время разработанная система внедрена в защищённую операционную систему QP ОС, разработанную отечественной IT компанией ООО НТП «Криптосoft».

В дальнейшем в разработанное ПО можно добавить поддержку задания диапазонов IP-адресов и сетевых портов, поддержку адресов IPv6, а также функции хранения таблицы NAT в памяти для её многократного использования.



Спасибо за внимание!

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		170