Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Реализация межсетевого экрана

Студент ИУ7-72Б _____ _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Н.Ю. Рязанова
(Подпись, дата) (И.О.Фамилия)

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ7____
(Индекс)
_____ __И.В. Рудаков_
(И.О.Фамилия)
« _____ » _____ 2021 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____Операционные системы_____

Студент группы ___ИУ7-72Б_____

_____Брянская Екатерина Вадимовна_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____Реализация межсетевого экрана_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
__учебный_____
Источник тематики (кафедра, предприятие, НИР) ___кафедра_____

График выполнения проекта: 25% к _4_ нед., 50% к _7_ нед., 75% к 11 нед., 100% к _14_ нед.

Задание__Разработать межсетевой экран, осуществляющий контроль проходящего через него_ сетевого трафика, в виде загружаемого модуля, который использует для обработки пакетов___ символьное устройство. Предоставить пользователю возможность редактирования списка____ правил фильтрации и изменение видимости модуля в системе. _____

Оформление курсового проекта:

Расчетно-пояснительная записка на _20-30__ листах формата А4.
Расчетно-пояснительная записка должна содержать постановку задачи, введение,_____ аналитическую, конструкторскую, технологическую части, заключение, список литературы.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):_____ на защиту работы должна быть предоставлена презентация, состоящая из 15-20 слайдов._____ На слайдах должны быть отражены: постановка задачи, использованные методы и_____ алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс._____

Дата выдачи задания «27» __сентября___ 2021 г.

Руководитель курсового проекта _____ __Н.Ю. Рязанова____
(Подпись, дата) (И.О.Фамилия)

Студент _____ __Е.В. Брянская_____
(Подпись, дата) (И.О.Фамилия)

Содержание

ВВЕДЕНИЕ

Информация – это один из важнейших ресурсов, который представляет из себя движущую силу развития человечества. Потребность в ней – одна из основных для современного человека. Б**о**льший процент знаний приходится на приобретённые либо вследствие непосредственного получения информации, либо в результате анализа уже существующих данных.

Объём информационных ресурсов в любой области растёт огромными темпами, это связано, прежде всего, с усложнением всех сфер жизнедеятельности общества. Кроме того, непрерывно увеличиваются массивы передаваемой информации, речь идёт не только о бытовых разговорах, но и всего инфопотока в Интернете в целом.

Технический прогресс не стоит на месте, и сейчас практически каждый компьютер подключается к сети для обмена какими-либо данными. Компьютерная сеть изначально является незащищённой и уязвимой для внешних атак системой. И для того, чтобы предотвратить несанкционированный доступ к устройству, подключенному к глобальной или частной сети, необходимо использовать специальные программные средства, называемые межсетевыми экранами (также известные, как сетевые фильтры, брандмауэры, Firewall-ы).

Цель данной работы - разработать межсетевой экран, осуществляющий контроль проходящего через него сетевого трафика, в виде загружаемого модуля.

Необходимо предоставить пользователю возможность задания правил фильтрации и изменения видимости модуля в системе.

Для достижения поставленной цели необходимо решить следующие задачи:

1) изучить основные принципы работы сети и межсетевых экранов;
2) ознакомиться со способом перехвата пакетов сети;
3) проанализировать особенность misc драйвера и основные принципы ра-

боты с ним;

4) изучить методы передачи информации из пространства пользователя в пространство ядра и наоборот;
5) реализовать межсетевой экран.

1 Аналитическая часть

1.1 Постановка задачи

Необходимо разработать межсетевой экран для операционной системы Linux, задача которого – анализировать проходящий через него сетевой трафик.

Программное обеспечение должно быть реализовано в виде драйвера и позволять пользователю добавлять/удалять правила фильтрации и изменять видимость самого модуля в системе.

Правила описывают те пакеты, которые считаются «запрещёнными» по той или иной причине, и представляют из себя перечисление параметров с принимаемыми значениями, по которым будет в дальнейшем осуществляться анализ пакетов.

1.2 Общие принципы работы сети

Компьютерная сеть – совокупность компьютеров и других устройств, соединённых линиями связи и обменивающихся информацией между собой в соответствии с определёнными правилами – **протоколами**. [1]

Информация преобразуется в пакеты и передаётся от одного компьютера к другому, и для этого используются протоколы. Каждый пакет проходит несколько стадий, которые определены в модели OSI (наглядно представлена в виде таблицы 1).

Таблица 1: Модель OSI

№	Название
7	Прикладной уровень
6	Уровень представления
5	Сеансовый уровень
4	Транспортный уровень
3	Сетевой уровень
2	Канальный уровень
1	Физический уровень

Из пакета можно получить различную информацию, такую как:

- IP source – IP-адрес источника;
- IP destination – IP-адрес назначения;
- source port - порт источника;
- destination port - порт назначения (для известных сервисов порты зарезервированы и известны заранее);
- TCP/UDP - протоколы транспортного уровня;
- другое.

Для каждого приложения ведутся две системные очереди: очередь данных, поступающих к приложению из сети, и очередь данных, отправляемых этим приложением в сеть. Такие очереди называются **портами**, причём входная и выходная очереди одного приложения рассматриваются как один порт. Для идентификации портов им присваивают номера.

1.3 Межсетевой экран

Межсетевой экран – это программное средство, предназначенное для фильтрации входящего и исходящего трафика, в соответствии с некоторыми

заранее заданными критериями, правилами, тем самым, осуществляя защиту компьютера от сетевых угроз.

В общих чертах его работа изображена на Рис. 1.1.

Рисунок 1.1 – Принцип работы межсетевого экрана

Межсетевой экран в основном работает на пакетном уровне (уровни 3 и 4 модели OSI, которая представлена в таблице 1). [2]

Пакеты могут анализироваться в соответствии со следующими критериями:

- формальная корректность пакета;
- направление (входящий или исходящий);
- тип протокола;
- порт (источника, назначения);
- и т.д.

Для того, чтобы отфильтровать пакеты по тому или иному признаку, необходимо задать соответствующие правила, которые создаются пользователем. Когда межсетевой экран перехватывает пакет, он просматривает все его поля и сравнивает их с правилами из таблицы, причём в том порядке, как они были заданы. Если совпадение было найдено, то пакет дальше никуда не передаётся.

1.4 Загружаемый модуль ядра

Для того, чтобы добавить новый функционал в ядро Linux, нужно либо перекомпилировать его (что небезопасно), либо воспользоваться загружаемым

модулем ядра. [3]

После загрузки модуля он становится частью операционной системы и ему доступны все структуры и функции ядра. Когда функциональность, предоставляемая модулем, больше не требуется, то он может быть выгружен.

В Linux все модули обычно хранятся в каталоге /lib/modules и имеют расширение .ko. Модули загружаются и выгружаются с помощью специальных команд, приведённых в Листинге 1.

Листинг 1: Команды для загрузки и выгрузки загружаемого модуля ядра

```
// загрузить модуль в ядро
insmod имя< модуля.ko>

// выгрузить модуль из ядра
rmmod имя< модуля>
```

Загружаемый модуль ядра должен иметь определённую структуру. Обязательной частью любого загружаемого модуля являются.

- **Функция загрузки (инициализации) модуля.** В заголовке используется макрос __init__. Её задачей является подготовка модуля для дальнейшего функционирования как части ядра ОС.
- **Функция выгрузки модуля.** В заголовке используется макрос __exit__. Его задачей является освобождение ресурсов, занимаемых модулем в конце его работы.
- **Макросы module_init(init_func), module_exit(exit_func)** нужны для формирования кода, который будет выполняться при загрузке/выгрузке ядра (в частности внутри этого кода содержится вызов функций __init__ и __exit__, переданных в макросы).
- **Макрос MODULE_LICENSE(char* license)** сообщает под какой лицензией находится модуль. Обычно указывается ”GPL”.

Приведённые первые две функции являются **точками входа**. Также в модуле

может быть указано следующее.

- **MODULE_AUTHOR(”Name”), MODULE_DESCRIPTION(”LAB”)** – макросы, определяющие информацию о модуле (имя автора, краткое описание).
- **Дополнительные точки входа.** Это можно сделать, например, с помощью структуры file_operations, где можно указать функции модуля в качестве точек входа в него при различных действиях с файлом.

1.5 Управление внешними устройствами

Одна из самых сложных задач системы – управление внешними устройствами. Их достаточно много (мыши, клавиатуры, сканеры, принтеры и т.д.), все разных типов и выполняют разные задачи.

Структура **struct file_operations** (Листинг 2) используется для определения собственных функций для работы с конкретным устройством.

Листинг 2: struct file_operations

```
1  struct file_operations {
2      struct module *owner;
3      ...
4      ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
5      ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
6      ...
7      int (*open) (struct inode *, struct file *);
8      ...
9      int (*release) (struct inode *, struct file *);
10     ...
11 }
```

Файл устройства – это специальный файл, который обеспечивает связь между файловой системой и драйверами устройств. В отличие от обычных файлов они являются только указателями на соответствующие драйверы в ядре.

Файлы устройств имеют три дополнительных атрибута, которые характе-

ризуют устройство, соответствующее данному файлу:

1) **класс устройства**:
    - блок-ориентированные (блочные) (пример: жёсткий диск) передают данные блоками, взаимодействие только через буферную память;
    - байт-ориентированные (символьные) (пример: принтер) передают данные посимвольно, как непрерывный поток байт, для взаимодействия буфер не требуется;
2) **старший номер** устройства, обозначающий тип устройства (посмотреть их можно в /proc/devices);
3) **младший номер** устройства применяется для нумерации устройств одного типа, т. е. устройств с одинаковыми старшими номерами.

Для того, чтобы идентифицировать устройство, используются старший и младший номера. [3]

1.6 Драйвер в ОС Linux

Драйвер – особая программа, является частным случаем загружаемого модуля ядра. Есть отличительная особенность – множество точек входа (более 2), и все определены в соответствующей структуре. Основная задача драйвера – управление внешними устройствами. [3]

В Linux драйверы делятся на 3 типа:

1) **встроенные в ядро**
    - инициализируются при запуске системы;
    - позволяют автоматически находить соответствующие устройства при обращении к ним;
2) **реализованные как загружаемые модуля ядра**;
3) **код модулей поделён между ядром и специальной утилитой**
    - например, у драйвера принтера ядро отвечает за взаимодействие с параллельным портом, а формирование управляющих сигналов выполняет демон печати.

Межсетевой экран – драйвер второго типа, поскольку может быть свободно загружен и выгружен пользователем.

1.6.1 Misc Device Driver

Misc (от слова ”miscellaneous”) **драйвером** называется простой символьный драйвер. Его используют в случаях, когда возможно использовать упрощение: не задавать самостоятельно старший номер устройства (major), поскольку он определён заранее и равен 10. Но разработчик должен задать младший номер в пределах от 1 до 255. [4]

Такой подход позволяет сэкономить оперативную память, особенно, если необходимо создать несколько символьных драйверов для нескольких простых устройств.

На misc драйверах могут быть определены операции open, read, write, close и т.д. И так же создаётся файл /dev/{misc_file}.

По аналогии с символьными драйверами, которые описываются структурой struct cdev, misc драйверы описываются с помощью структуры struct miscdevice, поля которой приведены в Листинге 3.

Листинг 3: struct miscdevice

```
1 struct miscdevice {
2     int minor;
3     const char *name;
4     struct file_operations *fops;
5     umode_t i_mode;
6     struct miscdevice *next, *prev;
7 };
```

Поле minor может принимать значение MISC_DYNAMIC_MINOR, означающее, что он будет назначен автоматически.

Для работы с таким драйвером используются функции, представленные в Листинге 4.

Листинг 4: Функции для регистрации и удаления misc драйвера

```
// регистрация драйвера
int misc_register(struct miscdevice *misc);

// удаление драйвера
int misc_deregister(struct miscdevice *misc);
```

1.7 netfilter

netfilter – это механизм фильтрации сетевых пакетов, который позволяет отслеживать их перемещение и при необходимости можно перехватить, блокировать. [5]

1.7.1 Точки перехвата

Путь сетевого пакета в ядре изображен на Рис. 1.2.

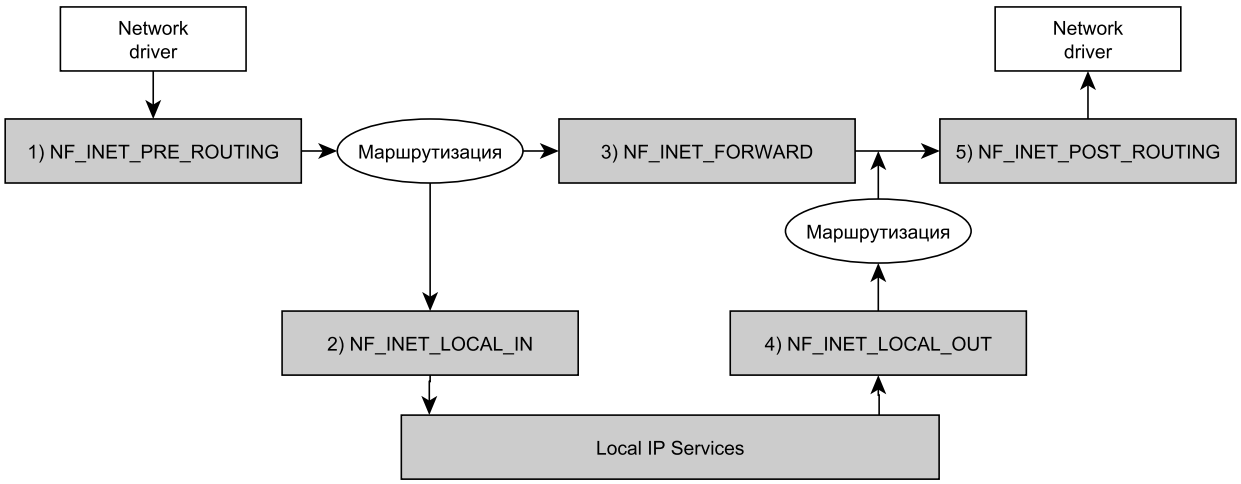

Рисунок 1.2 – Путь сетевого пакета

Предоставляется 5 точек, на которых могут быть определены функции перехвата, которые называются **хук-функциями**:

1) NF_INET_PRE_ROUTING – для всех входных пакетов;
2) NF_INET_LOCAL_IN – используется, чтобы перехватить пакеты, предназначенные для локального процесса;

3) NF_INET_FORWARD – используется для пакетов, предназначенных для другого интерфейса;
4) NF_INET_LOCAL_OUT – для пакетов, которые создают локальные процессы;
5) NF_INET_POST_ROUTING – для пакетов, которые уже настроены для дальнейшего прохождения по сети к своему адресату и готовы покинуть текущий сетевой стек.

1.7.2 Хук-функции

Для того, чтобы использовать хук-функцию, необходимо сначала заполнить структуру **struct nf_hook_ops**. Структура с основными полями приведена в Листинге 5.

Листинг 5: struct nf_hook_ops

```
1 struct nf_hook_ops {
2     nf_hookfn           *hook;
3     ...
4     u_int8_t            pf;
5     unsigned int        hooknum;
6     int                 priority;
7 };
```

В структуре находятся следующие поля:

- **hook** – функция, которая будет вызвана для обработки пакета, принимается решение отбросить или принять пакет;
- **pf** – семейство протоколов;
- **hooknum** – точка перехвата;
- **priority** – приоритет.

Регистрация и удаление хуков осуществляется посредством вызова функций, которые представлены в Листинге 6.

Листинг 6: Функции для регистрации и удаления хук-функций

```
1  // регистрация
2  int nf_register_net_hook(struct net *net, const struct nf_hook_ops *ops);
3
4  // удаление
5  void nf_unregister_net_hook(struct net *net, const struct nf_hook_ops *
   ops);
```

1.8 Передача данных из адресного пространства пользователя в адресное пространство ядра и наоборот

Поскольку правила фильтрации пакетов задаются пользователем, то необходимо передавать данные из адресного пространства пользователя в адресное пространство ядра. И так как межсетевой экран сохраняет их у себя для дальнейшей работы, действующие правила можно посмотреть, вызвав соответствующую команду, то есть, необходимо также обеспечивать передачу данных и в обратную сторону.

Для осуществления таких задач используются специальные функции (Листинг 7).

Листинг 7: Специальные функции

```
1  // копирование буфера из пользовательского пространства в пространство
   ядра
2  unsigned long copy_from_user(void * to, const void __user * from,
   unsigned long n);
3
4  // копирует данные из ядра в пространство пользователя
5  unsigned long copy_to_user(void __user * to, const void * from, unsigned
   long n);
```

1.9 Выводы из аналитического раздела

В этом разделе была формализована поставленная задача и рассмотрены основные моменты для её реализации, также подробно изучены принципы работы межсетевого экрана.

Для достижения поставленной цели было принято решение использовать простой misc драйвер, поскольку он ориентирован на выполнение небольших задач и имеет упрощённую схему создания.

2 Конструкторская часть

2.1 Требования к программе

Межсетевой экран должен быть реализован в виде загружаемого модуля, и предоставлять следующие возможности:

- добавление нового правила;
- удаление правила;
- просмотр всех правил;
- сокрытие модуля в системе;
- обнаружение модуля в системе.

Для непосредственного взаимодействия с пользователем необходимо разработать отдельную программу, представляющую из себя интерфейс для работы с загружаемым модулем. Программа выполняет не только функцию связующего звена между пользователем и межсетевым экраном, но и проверку входных данных: корректность вводимых команд и правил.

2.2 Инициализация модуля

На рисунке 2.1 представлена подробная схема алгоритма инициализации модуля.

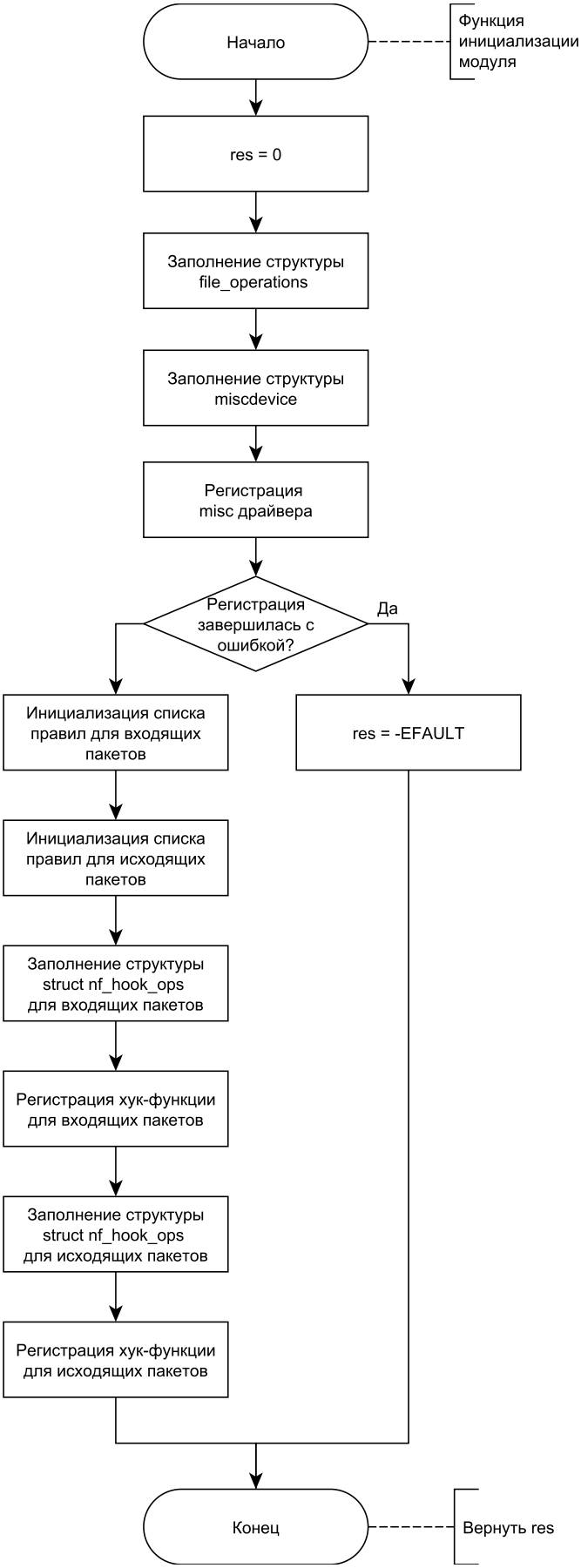

Рисунок 2.1 – Схема инициализации модуля

2.3 Завершение работы модуля

При выгрузке необходимо освободить все ресурсы, которые были зарезервированы в процессе работы модуля.

Детально работа этой функции изложена на рисунке 2.2.

Рисунок 2.2 – Схема выгрузки модуля

2.4 Основные функции, определённые в struct file_operations

При инициализации модуля одним из первых действий является заполнение структуры **struct file_operations**, в которой определяются функции для работы с файлами. В рамках поставленной задачи необходимо указать свои функции read (для того, чтобы получить список всех правил), write (используется для обработки нового правила).

На рисунках 2.3 – 2.4 приведены схемы для функций чтения и записи.

Рисунок 2.3 – Схема работы функции чтения

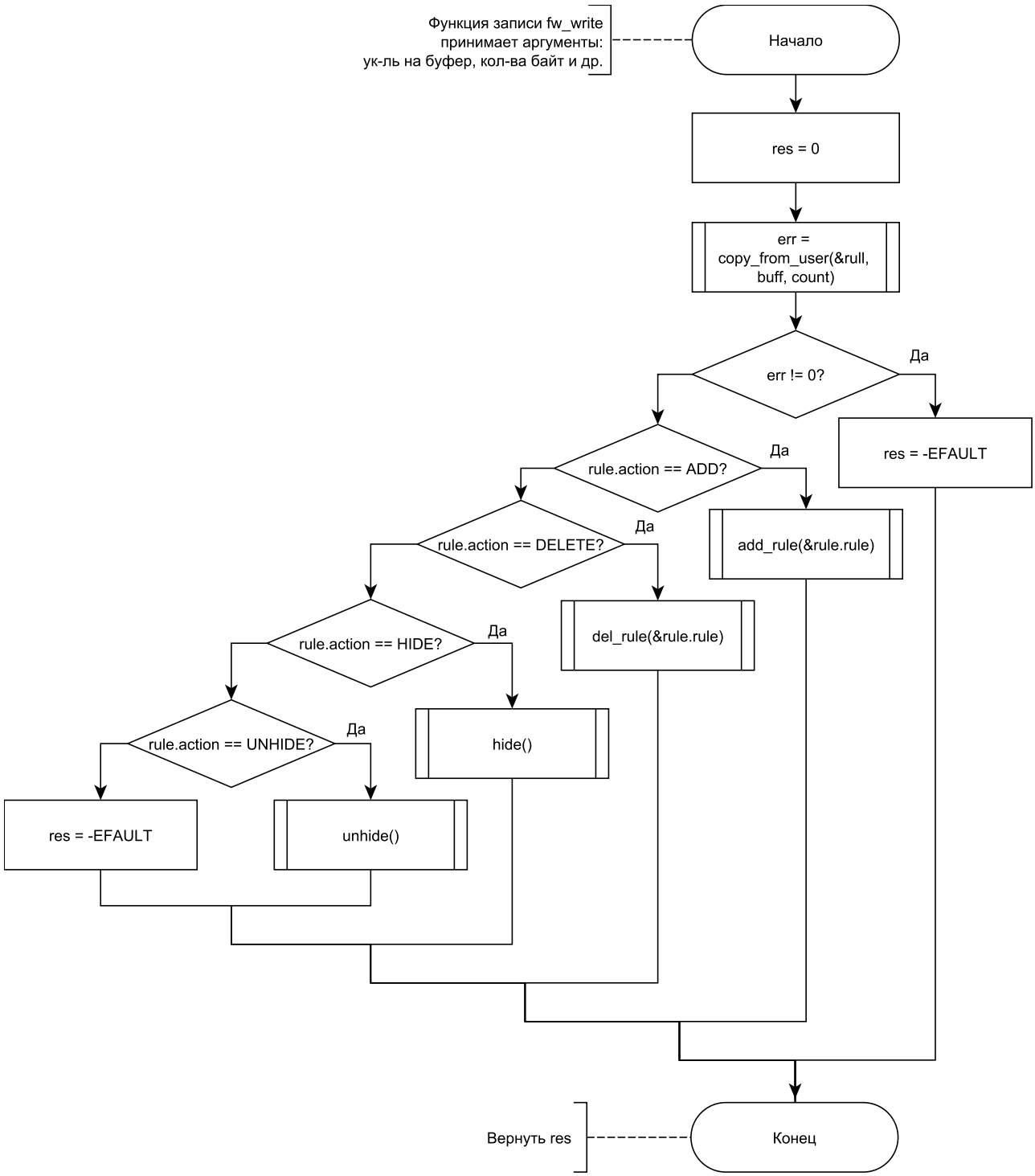

Рисунок 2.4 – Схема работы функции записи

2.5 Функция фильтрации пакетов

В процессе инициализации модуля также происходит регистрация хук-функций, заданных в структуре **struct nf_hook_ops** и необходимых для работы межсетевого экрана.

В рамках поставленной задачи регистрируются две функции: для обработки входящих и исходящих пакетов. Поскольку главное их отличие – направление анализируемых единиц, то рекомендуется реализовать одну функцию, в которую подаётся соответствующий список правил. Детали работы этой функции представлены на рисунке 2.5 – 2.6.

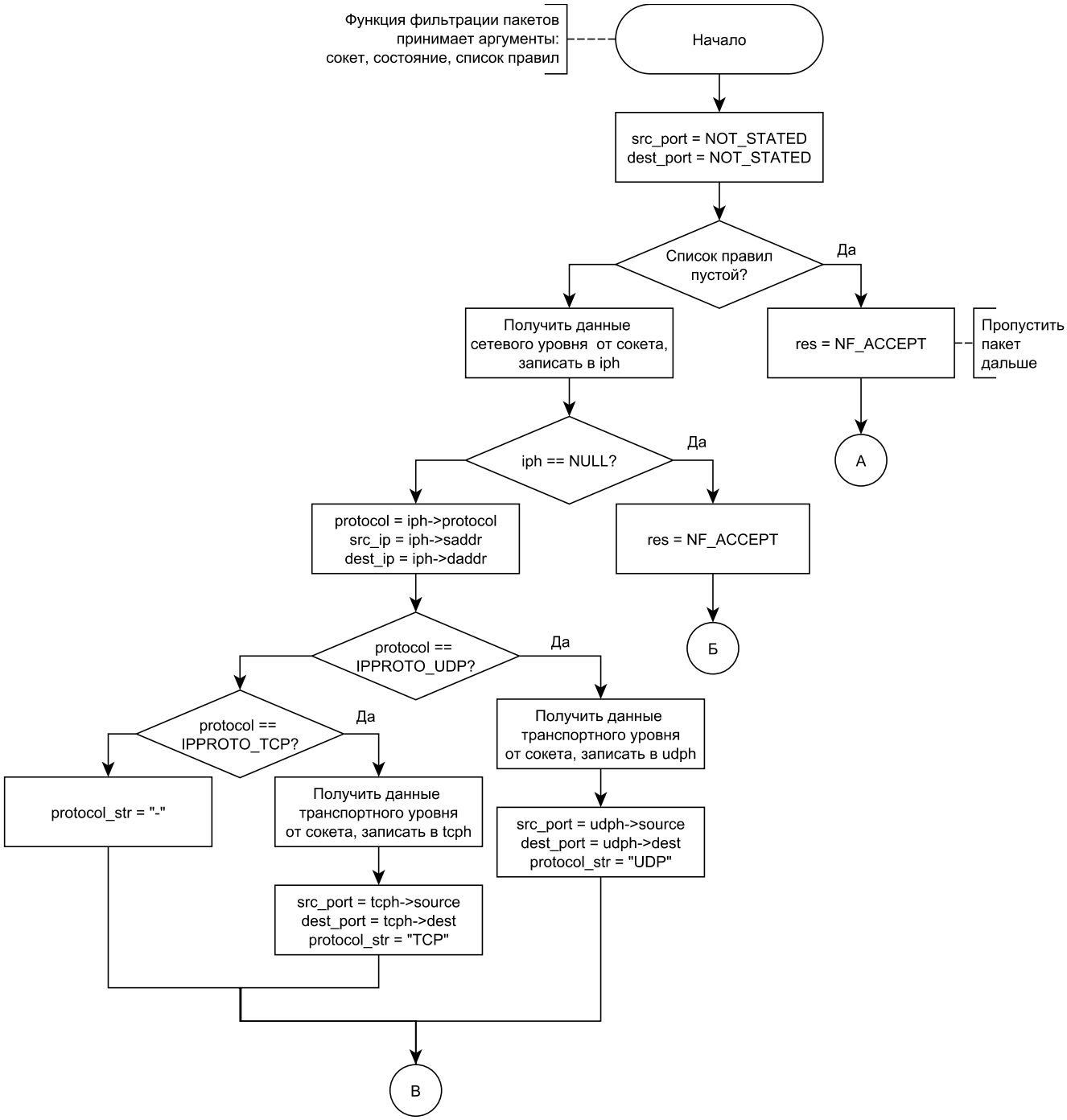

Рисунок 2.5 – Схема работы функции фильтрации пакетов

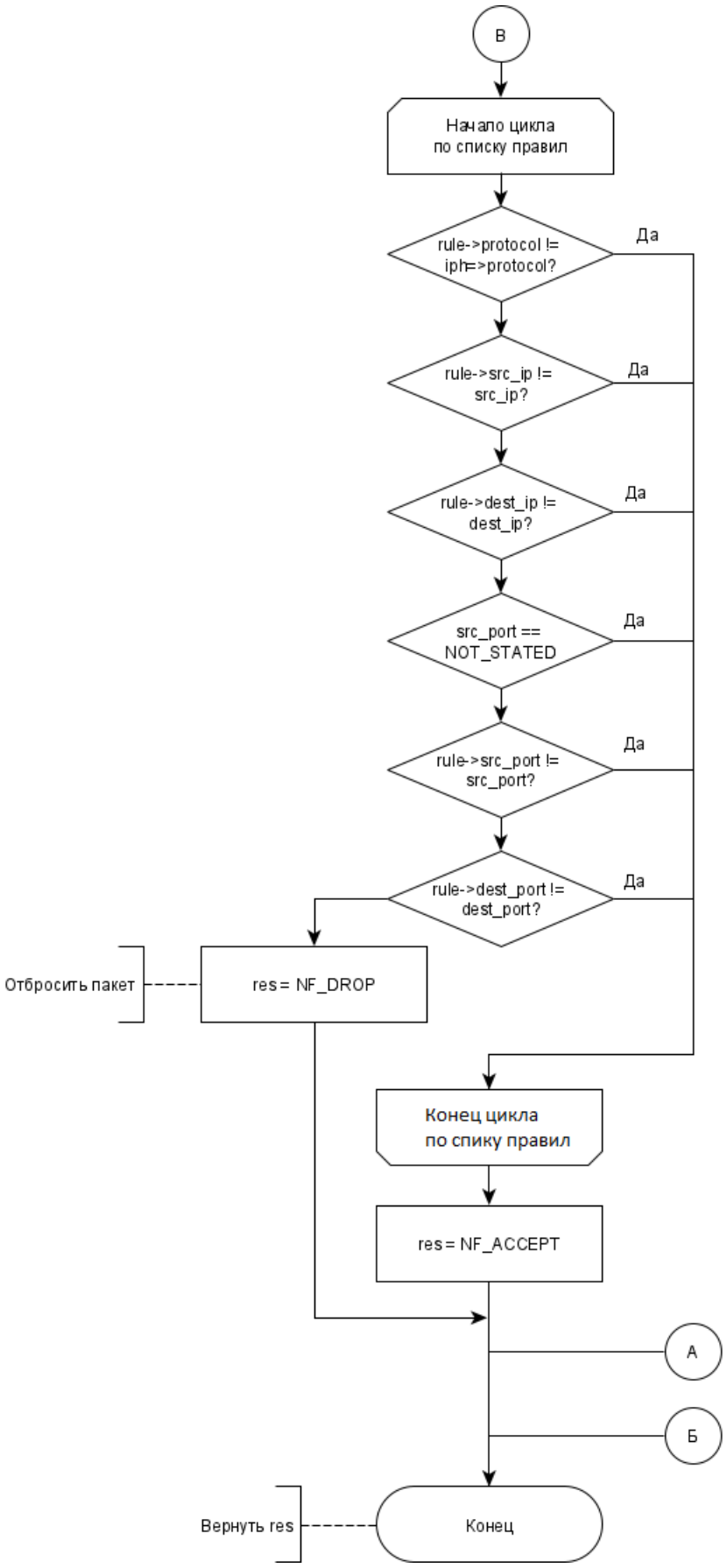

Рисунок 2.6 – Схема работы функции фильтрации пакетов (продолжение)

2.6 Что-то про видимость модуля в системе

2.7 Выводы

3 Технологическая часть

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. Рогозин Н.О., Курс лекций по дисциплине «Компьютерные сети» [Текст]
2. Программные межсетевые экраны [Электронный ресурс]. – Режим доступа: https://www.securitylab.ru/analytics/240197.php (дата обращения 01.10.2021).
3. Рязанова Н.Ю., Курс лекций по дисциплине «Операционные системы» [Текст]
4. Misc Device Driver – Linux Device Driver Tutorial Part 32 [Электронный ресурс]. – Режим доступа: https://embetronicx.com/tutorials/linux/device-drivers/misc-device-driver/#Misc_Device_Driver (дата обращения 05.10.2021).
5. NETFILTER [Электронный ресурс]. – Режим доступа: http://samag.ru/archive/article/169 (дата обращения 07.10.2021).