МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Забайкальский государственный университет»

(ФГБОУ ВПО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

**КУРСОВОЙ ПРОЕКТ**

По дисциплине: Сети и телекоммуникации

На тему: Программа для генерации и приёма LACPDU трафика

Выполнил студент группы ИВТ–18–2, Пичуев Дмитрий Андреевич

Руководитель работы: доцент кафедры ИВТ и ПМ, Машкин Владимир Анатольевич

Чита

2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Забайкальский государственный университет»

(ФГБОУ ВПО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

**ЗАДАНИЕ**

на курсовой проект

По дисциплине: Сети и телекоммуникации

Студенту: Пичуеву Дмитрию Андреевичу

Специальности (направления подготовки): Информатика и вычислительная техника

1 Тема курсовой работы: Программа для генерации и приёма LACPDU трафика

2 Срок подачи студентом законченной работы: 10.06.2021

3 Исходные данные к работе: описание предметной области

Дата выдачи задания: 18.03.2021

Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Машкин В.А /

(подпись, расшифровка подписи)

Задание принял к исполнению

«18» марта 2021 г.

Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Пичуев Д.А. /

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Забайкальский государственный университет»

(ФГБОУ ВПО «ЗабГУ»)

Факультет: Энергетический

Кафедра: Информатики, вычислительной техники и прикладной математики

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

По дисциплине: Сети и телекоммуникации

На тему: Программа для генерации и приёма LACPDU трафика

Выполнил студент группы ИВТ–18–2, Пичуев Дмитрий Андреевич

Руководитель работы: доцент кафедры ИВТ и ПМ, Машкин Владимир Анатольевич

Чита

2021

УТВЕРЖДАЮ

Зав.кафедрой\_\_\_\_\_\_\_\_

« » 20\_\_ г.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование раздела  курсового проекта | Месяц, неделя | | | | | | | | | | | | | | | |
| Март | | | | Апрель | | | | Май | | | | Июнь | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. Утверждение темы |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. Защита темы в предметной области |  |  |  |  | + |  |  |  |  |  |  |  |  |  |  |  |
| 3. Утверждение функционала |  |  |  |  |  |  | + |  |  |  |  |  |  |  |  |  |
| 4. Защита ТЗ |  |  |  |  |  |  |  | + |  |  |  |  |  |  |  |  |
| 5. Анализ данных |  |  |  |  |  |  |  |  | + |  |  |  |  |  |  |  |
| 6. Разработка алгоритмов |  |  |  |  |  |  |  |  |  | + |  |  |  |  |  |  |
| 7. Защита программной части |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  |
| 8. Защита реферативной части |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |
| 9. Публичная защита курсового проекта |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |

План выполнен: руководитель

(подпись, расшифровка подписи)

« » 20 г.

# РЕФЕРАТ

Пояснительная записка – 24с., рисунки – 8, источники – 5 таблицы – 1.

LACP, LACPDU, АГРЕГИРОВАНИЕ, SCAPY, QT DESIGNER, PYQT5, PANDAS, РЕГУЛЯРНЫЕ ВЫРОЖЕНИЯ.

В данной работе рассматривается процесс создания программ, позволяющих посредством LACP отправлять и отлавливать пакеты, от одного компьютера к другому. В работе определены методы разработки и описан процесс их применения при создании программ.

**СОДЕРЖАНИЕ**

Оглавление

[РЕФЕРАТ 5](#_Toc74225164)

[ВВЕДЕНИЕ 7](#_Toc74225165)

[1. Анализ поставленной задачи 8](#_Toc74225166)

[1.1 Описание предметной области 8](#_Toc74225167)

[1.2 Постановка задачи 8](#_Toc74225168)

[2. Теоретическая часть 10](#_Toc74225169)

[2.1 Основные определения 10](#_Toc74225170)

[2.2 Средства реализации 14](#_Toc74225171)

[3. Анализ данных 15](#_Toc74225172)

[3.1 Входные данные 15](#_Toc74225173)

[3.2 Промежуточные данные 15](#_Toc74225174)

[3.3 Выходные данные 15](#_Toc74225175)

[4. Программная реализация 16](#_Toc74225176)

[4.1 Структура приложения 16](#_Toc74225177)

[4.2 Алгоритмы 16](#_Toc74225178)

[5. Документирование 18](#_Toc74225179)

[5.1 Руководство пользователя 18](#_Toc74225180)

[ЗАКЛЮЧЕНИЕ 19](#_Toc74225181)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc74225182)

# ВВЕДЕНИЕ

Цель работы ­­­– сделать программу для генерации и приема трафика LACPDU. Изучить предметную область LACPDU. Используя знания, полученные в ходе лекций, практических работ и самообучения реализовать следующие возможности программы:

1. Программу для генерации и отправки трафика LACPDU;

2. Программу для приема трафика LACPDU

3. Вывод данных в таблицу для удобного прочтения;

4. Задание времени в программе приема трафика;

5. Удобный интерфейс программ;

6. Возможность отчистить таблицу;

В данной работе для генерации протокола используется язык программирования Python, вместе с Scapy. Для создания дизайна используется Qt Designer и PyQt5.

# 1. Анализ поставленной задачи

## 1.1 Описание предметной области

В работе будет показана генерация, отлов и отправка трафика LACPDU. Работа программы представлена в двух экземплярах, одна для отлова, вторая для генерации и отправки трафика. В программе для генерации нужно ввести некоторые данные для генерации и отправки. В программе для отлова трафика нужно вести время. В течение введённого времени программа будет отлавливать пакеты и записывает данные в файл и в таблицу.

## 1.2 Постановка задачи

Задачей проекта является разработка приложения для отправки LACPDU пакета и его приема. Задача будет реализована следующим образом:

1. Программа для отлова пакетов;
2. Программа для генерации и отправки пакетов;

Программа для генрации пакетов включает в себя: таблицу, данные пакета и необходимые элементы на форме.

Таблица включает себя: dst, src, data, partner\_state, actor\_state, subtype, type.

Данные для генерации трафика: dst, src, data, partner\_state, actor\_state.

Элементы на форме: таблица, кнопка отправки, кнопки для переключения типа partner\_state и actor\_state, кнопка для отчистки таблицы.

Программа для отливки пакетов включает в себя: таблицу, запись времени, необходимые элементы на форме и запись в файл.

Таблица включает себя: dst, src, data, subtype, type.

Запись времени происходит в секундах.

Элементы на форме: таблица, кнопка поиска, кнопка для отчистки таблицы, кнопка для вывода информации из файла.

Запись в файл происходи после отлова пакета. Она служит для сохранения информации о предыдущих пактах, после закрытия программы.

# 2. Теоретическая **часть**

## 2.1 Основные определения

Итак, есть такой класс протоколов, которые призваны контролировать различные аспекты Ethernet, иначе говоря, Flow Control. Они являются частью стандарта 802.3 и делятся на два вида:

* **Быстрые**. Они должны отрабатывать моментально для предотвращения снижения производительности и перерыва в предоставлении сервисов вообще. Как правило они реализуются аппаратно. Собственно, представитель этого класса — *механизм PAUSE* – когда порт устройства получает трафика больше, чем может обработать, он отсылает PAUSE Frame противоположному узлу с просьбой понизить скорость отправки. Мехнизм «Паузы» — это технология для полнодуплексных ликнов. Для Half-Duplex существует *Back Pressure*. Он весьма строгий и не просто заставляет притормозить, а блокирует передачу данных, потому что фактически сообщает передающему коммутатору, что имеет место коллизия и нужно прекратить свои тщетные попытки. Но кого это сейчас волнует?
* **Медленные** – те самые Slow Protocols. У них не такие большие аппетиты на частоту отправки и задержки. Они реализуются программно. OAM и LACP относятся именно к этому виду.  
  Для них используется специальный MAC-адрес: – 0180-с200-0002, и EtherType 8809.

**Ограничения**

Во-первых, на такого вида протоколы накладываются следующие ограничения:

1) Передавать не более 10 кадров в секунду

2) Максимальное число протоколов с EtherType 8809 — десять. Теоретически их может быть больше, но для них уже будет указан другой тип в заголовке Ethernet.

3) Размер кадра «медленных» протоколов ограничен 128 байтами. \*Не сказать, что выполняется честно — OAMPDU, например, спокойно может превышать этот размер.

Вышеуказанные ограничения преследуют простую цель — уменьшить объём служебного трафика в сети.  
Как для протоколов более высокого уровня существуют специально выделенные мультикастовые IP-адреса (224.0.0.5 для OSPF, например), так и «медленным» протоколам назначили отдельный мультикастовый MAC-адрес: 01-80-C2-00-00-02.

Данный МАС-адрес принадлежит диапазону выделенному ISO/IEC 15802-3 для протоколов, ограниченных одним линком. Фактически это означает, что кадры, передающиеся на данный адрес не могут быть перенаправлены за пределы данного конкретного линка.

Вполне возможно, что могут существовать «медленные» протоколы, которые требуют юникастовой пересылки кадра — это не воспрещается. То есть адрес 01-80-C2-00-00-02 — это не требование — это пожелание.  
«Медленный» протокол может использовать другой МАС-адрес, но никакой другой протокол, кроме «медленного», не может использовать данный (01-80-C2-00-00-02).

Как уже я заметил выше, тип протоколов группы «Slow Protocols» (EtherType) — 8809. Далее определяются подтипы — subtypes для конкретных представителей сего славного класса:

**Подтипы**

|  |  |
| --- | --- |
| **Подтип** | **Предназначение** |
| 0 | Неиспользуемый запрещённый подтип |
| 1 | Link Aggregation Control Protocol (LACP) |
| 2 | Link Aggregation—Marker Protocol |
| 3 | Operations, Administration, and Maintenance (OAM) |
| 4-9 | Зарезервировано |
| 10 | Organization Specific Slow Protcol (OSSP) |
| 11–255 | Неиспользуемый запрещённый подтип |

Таблица 1 – Подтипы

Как видите, выделено 8 бит под подтип. О типе 10 поговорим чуть ниже, а с остальными всё понятно:  
1 — обычный знакомый нам LACP  
2 — также часть механизма LAG для балансировки нагрузки, упорядоченного получения кадров и оптимизации управления линками LAG.  
3 — протоколы Ethernet OAM.

По сути, два известных нам сейчас протокола, которые используют этот стандарт — это LACP и OAM, но вообще-то есть возможность создать собственный «медленный» протокол по своим нуждам. Это и есть тот самый пунктдесять — OSSP — Organization Specific Slow Protocol.

**Статическое агрегирование**

Статическое агрегирование производится путем ручного конфигурирования пользователем и не требует использования протокола **LACP**. При конфигурировании статического агрегирования используется режим “**on**” для добавления порта в Port-Group.

**Динамическое агрегирование LACP**

LACP (Link Aggregation Control Protocol) - протокол агрегирования каналов, описанный в стандарте **IEEE 802.3ad. LACP** использует**LACPDU** сообщения для обмена информацией с соседней стороной.

После включения LACP порт посылает **LACPDU**, уведомляя ответную сторону о приоритете и MAC адресе системы, приоритете и адресе порта и ключе операции. Когда ответный порт получает эту информацию, он сравнивает её с информацией о своих портах, настроенных на агрегацию. Таким образом обе стороны достигают соглашения о включении или исключении порта из динамической группы агрегации.

В динамической группе агрегации порты имеют 2 статуса - выбранный (**selected**) и в ожидании (**standby**). Порты могут посылать и принимать **LACPDU** находясь в любом статусе, но в статусе **standby** порт не может передавать данные.  
Поскольку существует ограничение на количество портов в группе, если текущее число членов агрегации превышает это ограничение, коммутатор согласовывает статус порта с другой стороной на основании**port ID.** Согласование происходит следующим образом:

1. Сравнение ID устройств (приоритет системы + MAC адресе системы). Если приоритет устройств одинаков - сравниваются MAC адреса устройств. Наименьший номер будет иметь наивысший приоритет;
2. Сравнение ID портов (приоритет порта + идентификатор порта). Для каждого порта на стороне устройства с наивысшим приоритетом системы сравниваются приоритеты портов. Если приоритеты одинаковые - сравниваются ID портов. Порт с наименьшим идентификатором порта становится выбранным **(selected),** а остальные - в режим ожидания **(standby).**
3. В данной **Port-Group** порт с наименьшим идентификатором и статусом **standby**становится мастер-портом. Другие порты со статусом **selected**становятся членами группы.

## 2.2 Средства реализации

Язык программирования Python будет использован как основа для написания программы.

Scapy ­­как инструмент, для создания и работы с сетевыми пакетами. Программа написана на языке Python, автор Philippe Biondi

Qt Designer и PyQt5 для создания дизайна программы.

Qt Designer — кроссплатформенная свободная среда для разработки графических интерфейсов (GUI) программ использующих библиотеку Qt. Входит в состав Qt framework.

PyQt5— является одним из наиболее часто используемых модулей для создания **GUI приложений** в Python.

Pandas - программная библиотека на языке Python для обработки и анализа данных.

# 3. Анализ данных

## 3.1 Входные данные

К входным данным генерации пакета относиться: dst(destination), src(source), data, partner\_state, actor\_state. Эти данные служат для генерации протокола и запрашиваются в специальных полях в программе.

К входным данным поиска пакета относиться time. Оно служит для задания промежутка времени поиска пакетов.

## 3.2 Промежуточные данные

К промежуточным данным генерации пакета относиться таблица с данными о пакете. Эта таблица состоит из: dst, src, data, partner\_state, actor\_state, subtype, type.

К промежуточным данным поиска пакета относиться таблица с данными о пакете. Данные следующие: dst, src, data, subtype, type.

## 3.3 Выходные данные

К выходным данным относиться, хранение полученных пакетов в текстовом файле.

# 4. Программная реализация

## 4.1 Структура приложения

В результате работы с программой была придумана структура, по которой велась разработка.

Модули приложения выглядят следующим образом:

1. Модуль (Lacp\_main.py), точка вхождения в программу генерации . К ней подключается GUI приложения.
2. Модуль (GUI.py), дизайн программы генерации пакета. К модулю подключается система отправки.
3. Модуль (Lacp.py), производит отправку протокола.
4. Модуль (Client\_lacp/Client\_main.py), точка вхождения в программу отправки. К ней подключается GUI приложения.
5. Модуль (Client\_lacp/GUI2.py), дизайн программы генерации пакета. К нему подключается поиск пакета.
6. Модуль (Client\_lacp/lacp\_client.py), поиск отправленных пакетов с пункта 3.
7. Модуль (Client\_lacp/data\_lacp.txt), хранит данные отловленных пакетов.

## 4.2 Алгоритмы

В данной работе реализованы следующие алгоритмы:

1. **Алгоритм отправки и генерации пакета.** Удобная программа считывает введение данные, генерирует пакет LACP и отправляет. После данные появляются в таблице программы.
2. **Алгоритм отлова пакета**. Записанное время в программе считывается и в течение этого времени происходит поиск пакетов LACP. Все найденные пакеты записываются в текстовый файл, по истечению поиска можно вывести их в таблицу.
3. **Алгоритм записи в текстовый файл**. С помощью библиотеки Pandas, в удобном формате, записываются пакеты в текстовый файл. После считываются в таблицу из файла по нажатию кнопки.
4. **Алгоритм неправильно введённых данных.** В обеих программах, если неправильно ведены данные, программа выведет предупреждение. Для этого используются регулярные выражения.

# 5. Документирование

## 5.1 Руководство пользователя

В работе используются две программы: одна для отлова, вторая для генерации трафика.

Программа для генерации трафика, включается по модулю (Lacp\_main.py). Как программа включилась нужно ввести данные о пакете. Такие как: dst, src, data, partner\_state, actor\_state. Partner\_state, actor\_state не могут быть оба passive, иначе программа не сможет отправить пакет по технологии LACP. В dst и src вводится MAC-адрес, желательно в dst вводить отдельный мультикастовый MAC-адрес LACP. В data запиываются сообщения для отправки на другой компьютер, например: «Привет с другого компьютера», «Привет» и т.д. В программе есть две кнопки: одна для отправки пакета введённых данных, вторая для отчистки таблицы.

Программа для отлова пакетов, включается по модулю (Client\_lacp/Client\_main.py). После открытия программы нужно ввести время и нажать кнопку «Искать». Пакеты автоматически запишутся в текстовый файл. Время не может быть больше 10000 секунд. По истечению отлова появится окно, для удобного использования и понимания когда закончился отлов пакетов. По нажатию кнопки «Читать из файла» данные в таблицу программы. По нажатию кнопки «Отчистить», таблица программы отчищается.

Примеры для всех представленных действий смотреть в приложении A.

# ЗАКЛЮЧЕНИЕ

В результате была разработано две программы: для генерации и для отлова трафика LACP.

Проработан удобный дизайн программ, все данные удобно выводятся в таблицу. Разработана система отправки и отлова пакетов, с помощью Scapy.

Сделан модуль для удобной записи в файл, с помощью библиотеки Pandas.

Использованы регулярные выражения, для отлова неправильно введённых данных.

В результате работы изучена программная область, LACPDU, Slow Protocols и Scapy.

В связи с нехваткой времени не все функции, и алгоритмы были реализованы. Например: Использование базы данных для грамотного хранения пакетов, более полный вывод информации о пакетах LACP и алгоритмы для более полной проверки LACPDU.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы компьютерных сетей Тема №8. Протокол агрегирования каналов: Etherchannel [Электронный ресурс] – URL <https://habr.com/ru/post/334778/>
2. LACP и агрегация портов. [Электронный ресурс] – URL <https://nag.wiki/pages/viewpage.action?pageId=1214130>
3. АГРЕГИРОВАНИЕ ПОРТОВ. LACP. [Электронный ресурс] – URL <http://netwild.ru/aggregation/>
4. Что такое Slow Protocols [Электронный ресурс] – URL <https://habr.com/ru/post/180761/>
5. scapy.contrib.lacp [Электронный ресурс] – URL <https://scapy.readthedocs.io/en/latest/api/scapy.contrib.lacp.html>

**ПРИЛОЖЕНИЯ A**

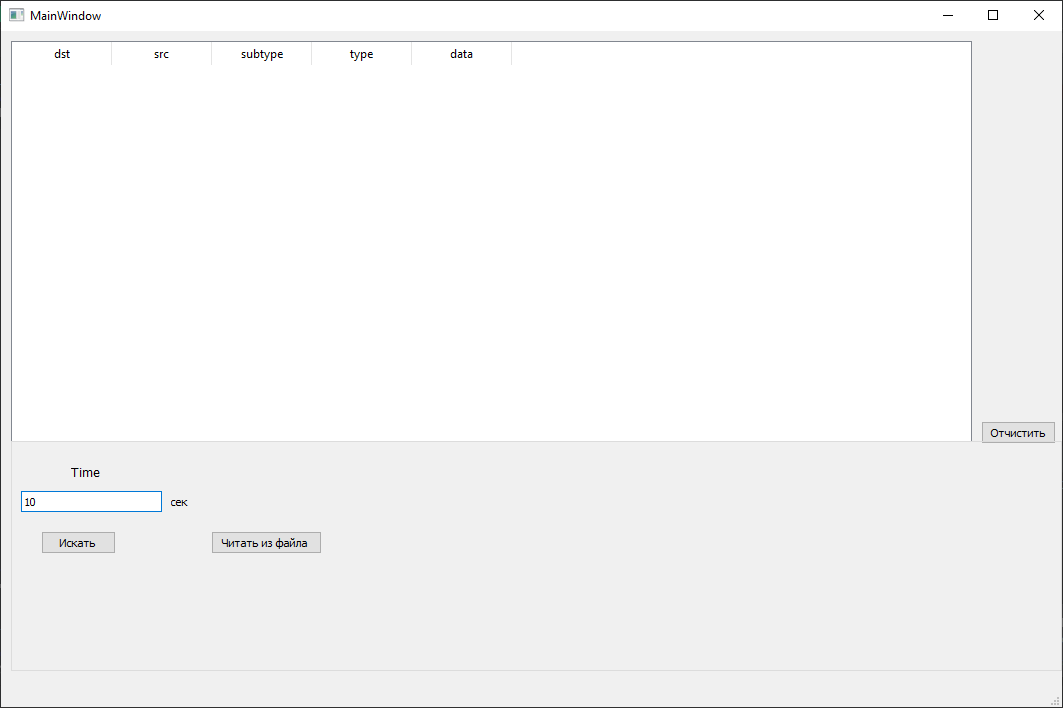


Рисунок 1 – Задаем время

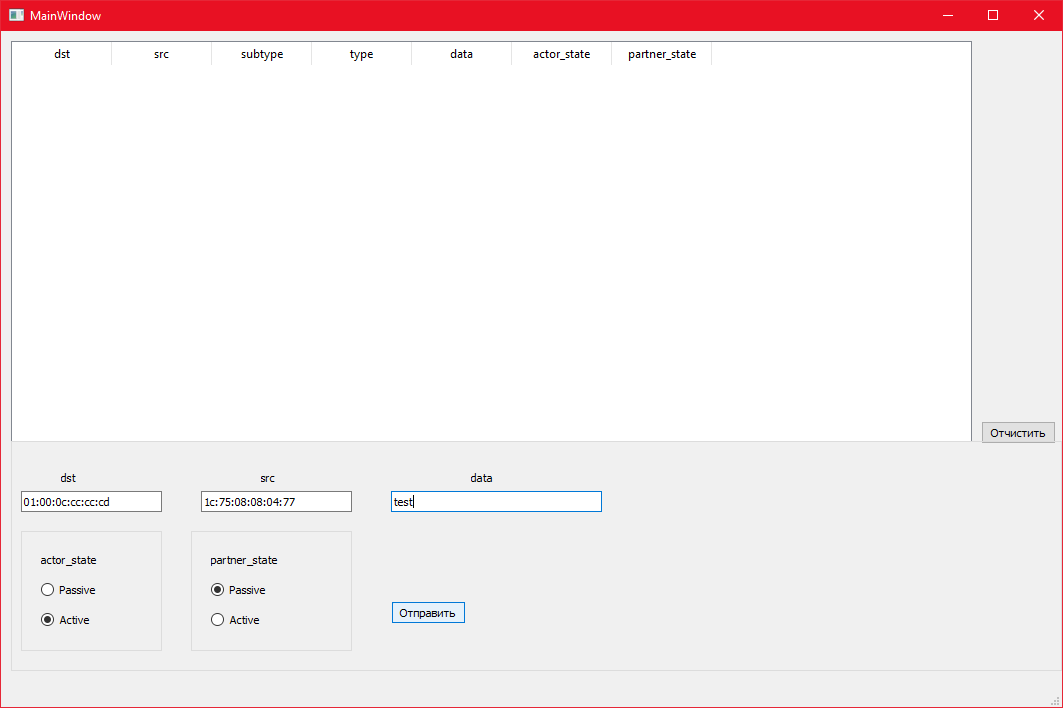


Рисунок 2 – Отправляем пакет

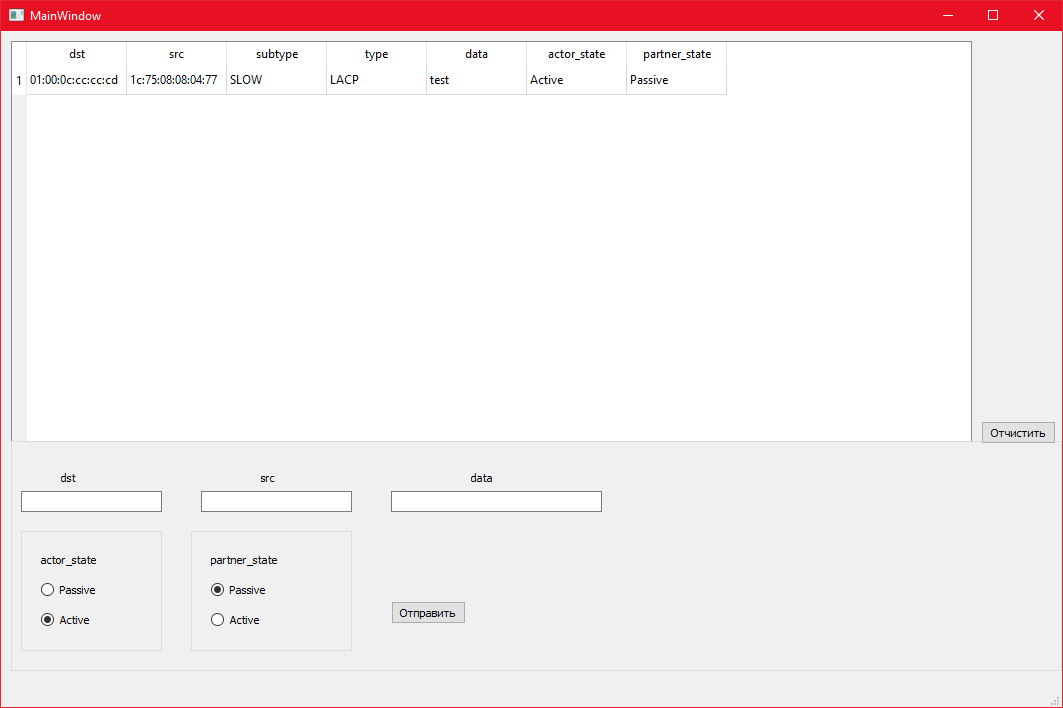


Рисунок 3 – Данные в таблице

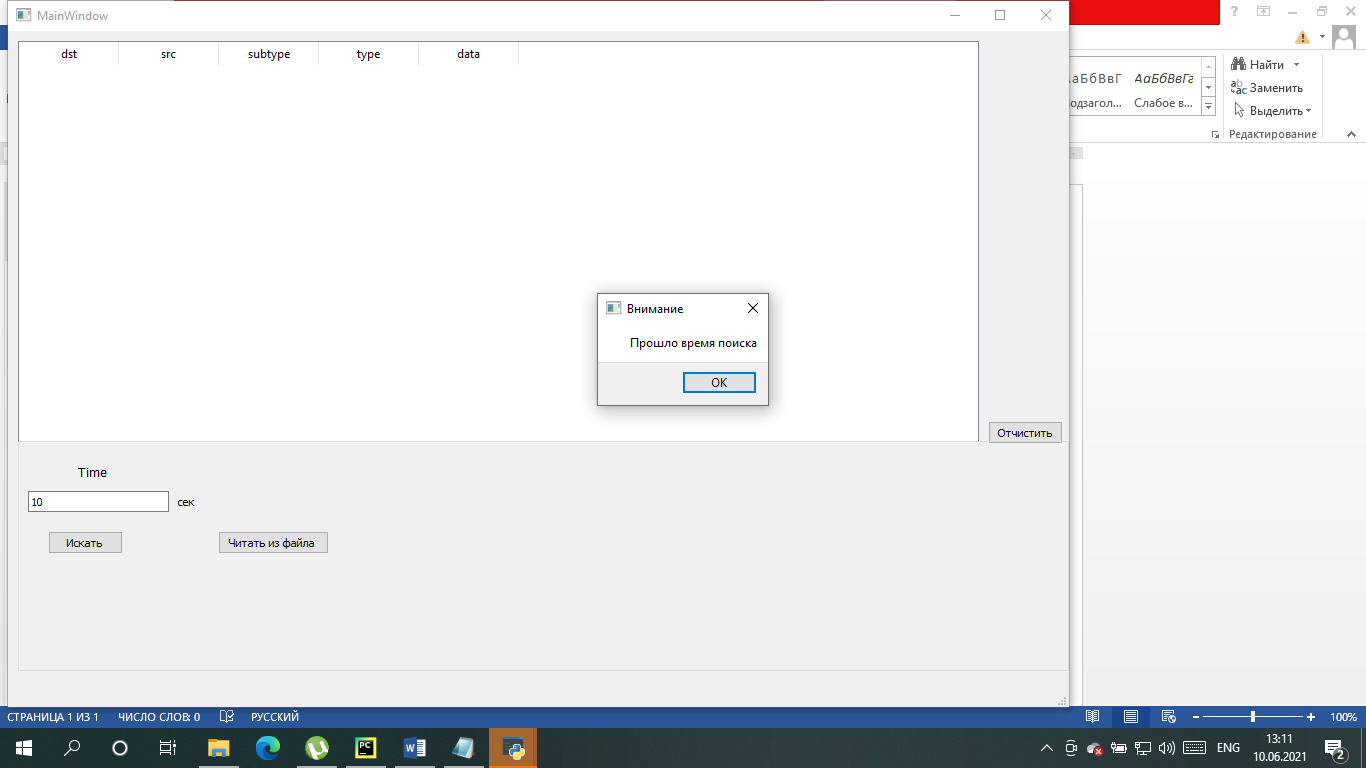


Рисунок 4 – Закончилось время поиска

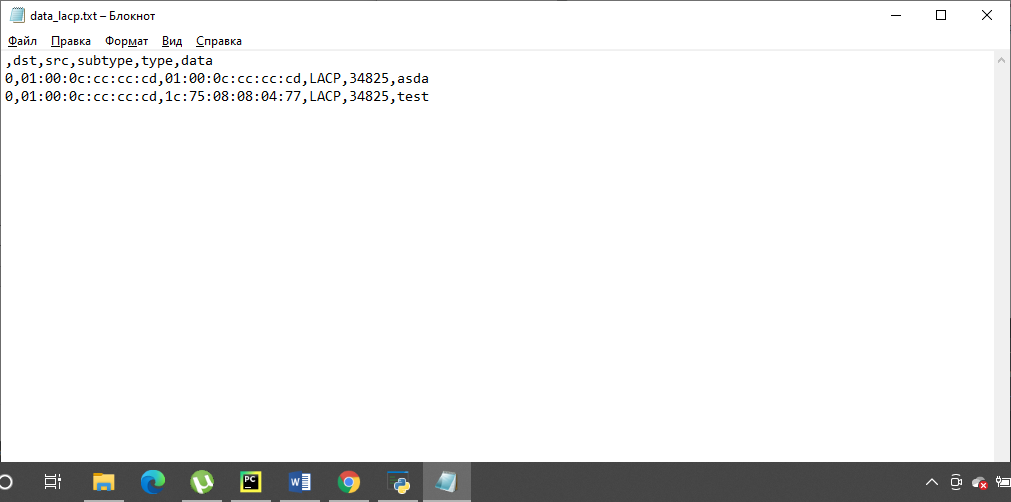


Рисунок 5 – Данные в текстовом файле

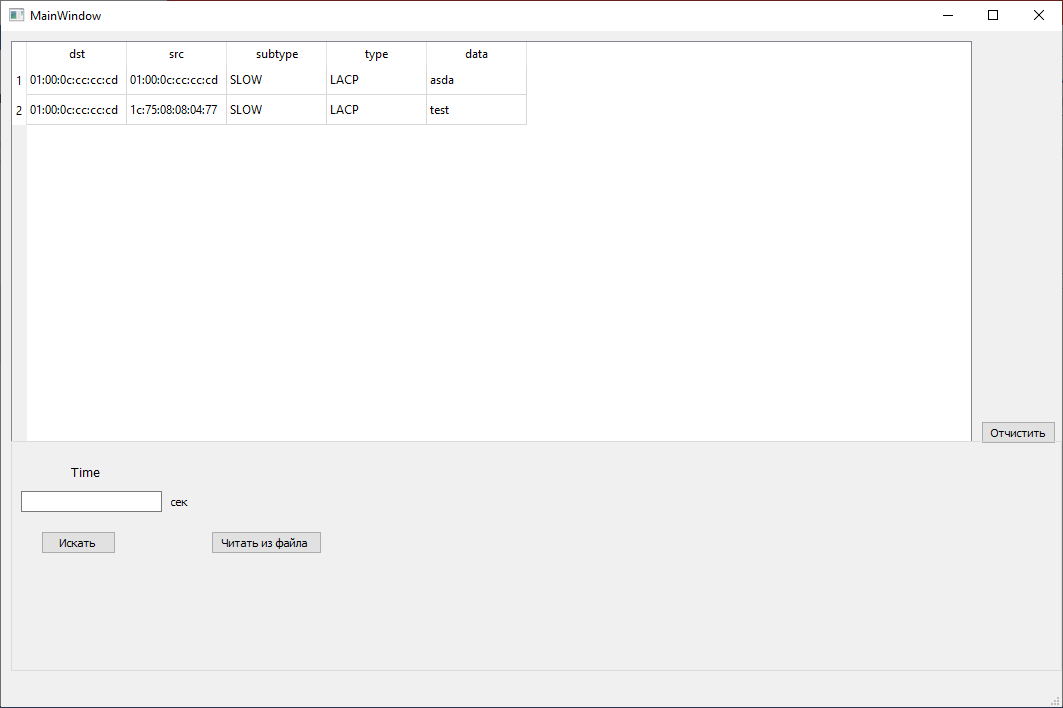


Рисунок 6 – Данные в программе отлова

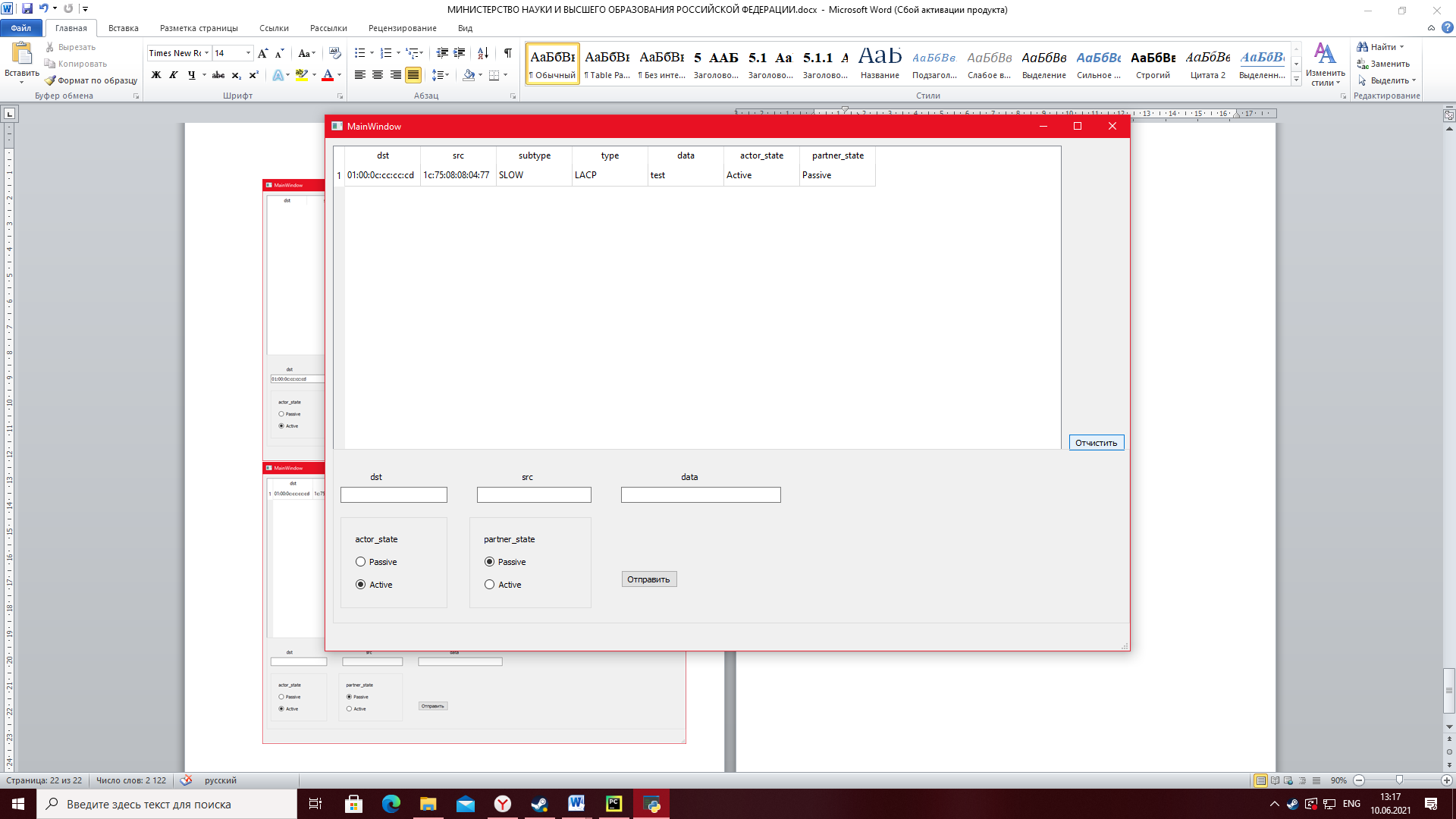


Рисунок 7 – Отчистка таблицы

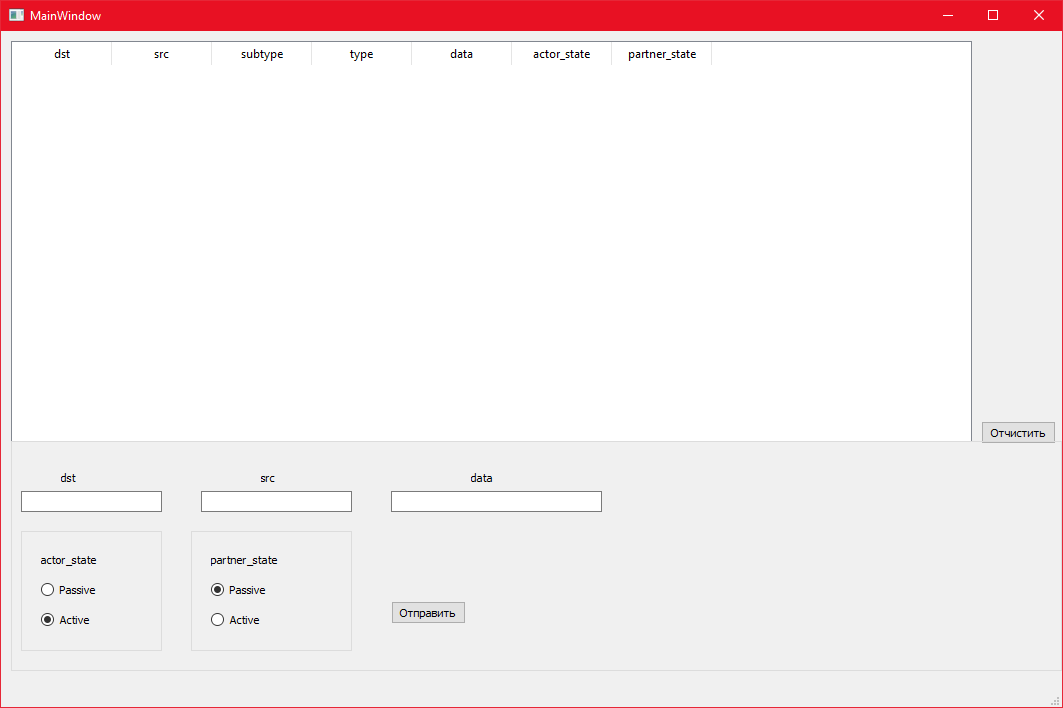


Рисунок 8 – Отчистка таблицы произошла