МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ | Факультет прикладной математики и информатики

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

ЛАБОРАТОРНАЯ РАБОТА № 4

ПО ДИСЦИПЛИНЕ «ОПЕРАЦИОННЫЕ СИСТЕМЫ И КОМПЬЮТЕРНЫЕ СЕТИ»

АНАЛИЗ СТРУКТУРЫ КАДРА ТЕХНОЛОГИИ ETHERNET

Бригада	ВЕСЕЛЫЙ ДЕНИС
№1	ВОРОНЧУК ИЛЬЯ
Группа	ПМИ-32
Преподаватели	КОБЫЛЯНСКИЙ В.Г.
	СИВАК М.А.

Новосибирск, 2025

Цель работы

Спроектировать и реализовать программу, выполняющую анализ структуры кадра/фрейма технологии Ethernet.

Ход выполнения работы

Для выполнения задания была разработана программа-анализатор на языке C++, способная читать двоичные файлы с дампами сетевого трафика и разбирать содержащиеся в них Ethernet-кадры. Программа реализует алгоритм автоматического распознавания четырех основных форматов кадров: Ethernet II (DIX), IEEE 802.3 LLC, IEEE 802.3 Novell RAW и IEEE 802.3 SNAP.

1. Результаты работы программы-анализатора

Программа была последовательно запущена для анализа файлов `ethers01.bin`, `ethers06.bin` и `ethers07.bin`. Ниже представлены скриншоты протокола работы программы для каждого файла.

```
Enter the name of the .bin file to analyze: ../data/ethers01.bin
--------------------------------------
Frame #1
  Source MAC:      00:90:27:a1:36:d0
  Destination MAC: 00:02:16:09:fa:40
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       195.62.2.11
  Destination IP:  92.125.134.243
  Frame Size:      108 bytes
--------------------------------------
Frame #2
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       81.181.78.206
  Destination IP:  195.62.2.11
  Frame Size:      210 bytes
--------------------------------------
Frame #3
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       92.125.134.243
  Destination IP:  195.62.2.11
  Frame Size:      81 bytes
--------------------------------------
Frame #4
```

Рис. 1: Результат анализа файла ethers01.bin

```
Enter the name of the .bin file to analyze: ../data/ethers06.bin
--------------------------------------
Frame #1
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       81.181.78.206
  Destination IP:  195.62.2.11
  Frame Size:      218 bytes
--------------------------------------
Frame #2
  Source MAC:      00:90:27:a1:36:d0
  Destination MAC: 00:02:16:09:fa:40
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       195.62.2.11
  Destination IP:  81.181.78.206
  Frame Size:      66 bytes
--------------------------------------
Frame #3
  Source MAC:      00:04:4d:8a:b0:d5
  Destination MAC: 01:80:c2:00:00:00
  Frame Length:    38 bytes
  Frame Type:      IEEE 802.3 LLC
  Frame Size:      52 bytes
--------------------------------------
Frame #4
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
```

Рис. 2: Результат анализа файла ethers06.bin

```
Enter the name of the .bin file to analyze: ../data/ethers07.bin
--------------------------------------
Frame #1
  Source MAC:      00:90:27:a1:36:d0
  Destination MAC: 00:02:16:09:fa:40
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       195.62.2.11
  Destination IP:  62.167.64.216
  Frame Size:      153 bytes
--------------------------------------
Frame #2
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       81.181.78.206
  Destination IP:  195.62.2.11
  Frame Size:      66 bytes
--------------------------------------
Frame #3
  Source MAC:      00:02:16:09:fa:40
  Destination MAC: 00:90:27:a1:36:d0
  Frame Type:      Ethernet II (DIX)
  Upper Protocol:  IP (0x0800)
  Source IP:       81.181.78.206
  Destination IP:  195.62.2.11
  Frame Size:      86 bytes
--------------------------------------
Frame #4
```

Рис. 3: Результат анализа файла ethers07.bin

2. Ручной анализ кадра №1

Для детального анализа был выбран **первый кадр** из файла `ethers01.bin`, в соответствии с номером бригады (№1). Анализ проводился с использованием 16-ричного редактора.

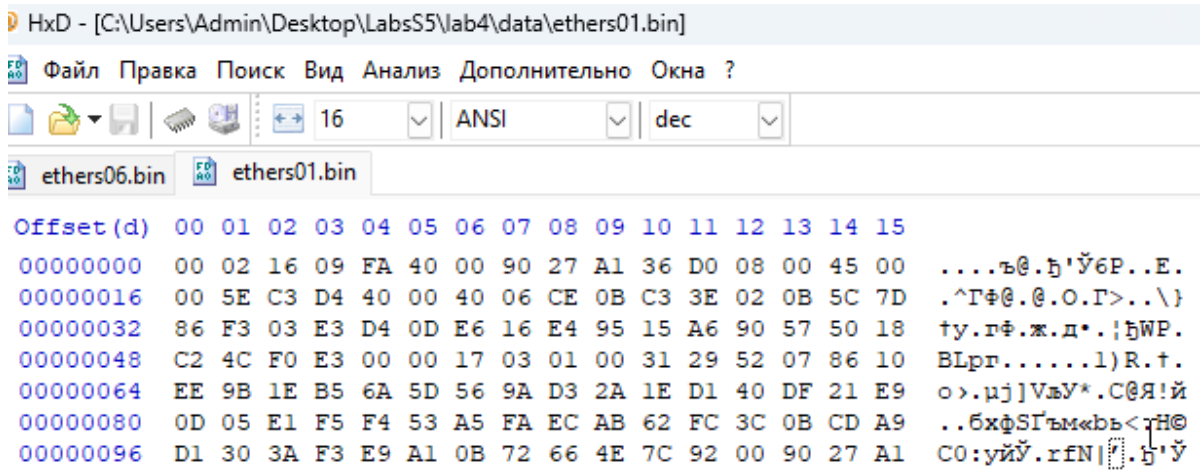

Рис. 4: Дамп первого кадра из файла ethers01.bin в 16-ричном редакторе

Побайтовый разбор кадра:

- **Размер кадра:** 108 байт (согласно выводу программы).
- **Байты 0-5** (`00 02 16 09 FA 40`)**:**

  MAC-адрес получателя (Destination MAC) — `00:02:16:09:FA:40`.
- **Байты 6-11** (`00 90 27 A1 36 D0`)**:**

  MAC-адрес отправителя (Source MAC) — `00:90:27:A1:36:D0`.
- **Байты 12-13** (`08 00`)**:** Поле "Тип/Длина". Значение `0x0800` (2048) больше 1500, следовательно, это кадр типа **Ethernet II**, а значение `0x0800` является кодом инкапсулированного протокола — **IP**.
- **Байты 14-33 (следующие 20 байт):** Заголовок IP-пакета.
  - **Байт 14** (`45`)**:** Версия IPv4 (4) и длина заголовка (5), т.е. 5 * 4 = 20 байт.
  - **Байты 16-17** (`00 5C`)**:** Общая длина IP-пакета — 92 байта.
  - **Байт 22** (`3C`)**:** TTL (Время жизни пакета) — 60.
  - **Байт 23** (`06`)**:** Протокол транспортного уровня — TCP.
  - **Байты 26-29** (`C3 3E 02 0B`)**:** IP-адрес отправителя (`195.62.2.11`).
  - **Байты 30-33** (`5C 7D 86 F3`)**:** IP-адрес получателя (`92.125.134.243`).
- **Оставшиеся байты (с 34 по 107):** Данные (заголовок TCP и полезная нагрузка).

Результаты ручного анализа полностью совпадают с данными, выведенными программой для первого кадра.

Вывод

В ходе выполнения лабораторной работы были изучены и применены на практике основные принципы анализа низкоуровневого сетевого трафика. Была разработана программа, корректно реализующая алгоритм определения типа Ethernet-кадра на основе значения поля "Тип/Длина".

Работа позволила получить глубокое понимание структуры данных на канальном уровне и закрепить навыки системного программирования.

Приложение

main.cpp

```
#include <fstream>
#include <iomanip>
#include <iostream>
#include <vector>

// Для кроссплатформенной работы с сетевыми функциями
#ifdef _WIN32
#include <winsock2.h>
#pragma comment(lib, "Ws2_32.lib")
#else
#include <arpa/inet.h>  // Для ntohs
#endif

// Выравнивание по 1 байту критически важно, чтобы компилятор не добавил
"пустышки" между полями
#pragma pack(push, 1)

// Структура, описывающая заголовок Ethernet (первые 14 байт кадра)
struct EthernetHeader {
    unsigned char dest_mac[6];
    unsigned char source_mac[6];
    unsigned short type_len;
};

// Структура, описывающая IP-заголовок (минимальный размер 20 байт)
// Нам нужны только адреса, поэтому опишем поля до них
struct IpHeader {
    unsigned char ihl_version;      // Длина заголовка и версия
    unsigned char tos;              // Тип сервиса
    unsigned short total_len;       // Общая длина пакета
    unsigned short identification;  // Идентификатор
    unsigned short flags_offset;    // Флаги и смещение
    unsigned char ttl;              // Время жизни
    unsigned char protocol;         // Протокол
    unsigned short checksum;        // Контрольная сумма заголовка
    unsigned char source_ip[4];     // IP-адрес отправителя
    unsigned char dest_ip[4];       // IP-адрес получателя
};
```

```
#pragma pack(pop)

/**
 * @brief Выводит MAC-адрес в стандартном формате.
 * @param mac Указатель на 6-байтовый массив MAC-адреса.
 */
void print_mac_address(const unsigned char* mac) {
    for (int i = 0; i < 6; ++i) {
        std::cout << std::hex << std::setw(2) << std::setfill('0') <<
(int)mac[i];
        if (i < 5) std::cout << ":";
    }
    std::cout << std::dec;  // Возвращаем поток вывода в десятичный режим
}

/**
 * @brief Выводит IP-адрес в стандартном формате.
 * @param ip Указатель на 4-байтовый массив IP-адреса.
 */
void print_ip_address(const unsigned char* ip) {
    for (int i = 0; i < 4; ++i) {
        std::cout << (int)ip[i];
        if (i < 3) std::cout << ".";
    }
}

int main() {
    std::string filename;
    std::cout << "Enter the name of the .bin file to analyze: ";
    std::cin >> filename;

    std::ifstream file(filename, std::ios::binary);
    if (!file) {
        std::cerr << "Error: Could not open file " << filename << std::endl;
        return 1;
    }

    // Статистика
    int frame_counter = 0;
    int ip_frames = 0;
    int arp_frames = 0;
    int novell_raw_frames = 0;
    int llc_frames = 0;
    int snap_frames = 0;
```

```
    while (!file.eof()) {
        EthernetHeader eth_header;

        // Читаем заголовок Ethernet (14 байт)
        file.read((char*)&eth_header, sizeof(EthernetHeader));
        if (file.gcount() < sizeof(EthernetHeader)) {
            // Если прочитали меньше 14 байт, значит, достигли конца файла
            // или в файле "мусор" в конце
            break;
        }

        frame_counter++;
        std::cout << "---------------------------------------\n";
        std::cout << "Frame #" << frame_counter << std::endl;

        // Преобразуем поле Type/Length из сетевого порядка байт (Big-Endian)
в хостовый
        unsigned short type_len = ntohs(eth_header.type_len);

        std::cout << "  Source MAC:      ";
        print_mac_address(eth_header.source_mac);
        std::cout << std::endl;
        std::cout << "  Destination MAC: ";
        print_mac_address(eth_header.dest_mac);
        std::cout << std::endl;

        long long frame_data_len = 0;

        // ------ Логика определения типа кадра ------

        if (type_len > 1500) {  // Это кадр Ethernet II (DIX)
            std::cout << "  Frame Type:      Ethernet II (DIX)" << std::endl;
            if (type_len == 0x0800) {  // IP-пакет
                ip_frames++;
                std::cout << "  Upper Protocol:  IP (0x0800)" << std::endl;

                // Читаем IP заголовок
                IpHeader ip_header;
                file.read((char*)&ip_header, sizeof(IpHeader));
                frame_data_len = file.gcount();

                std::cout << "  Source IP:       ";
                print_ip_address(ip_header.source_ip);
```

```
                std::cout << std::endl;
                std::cout << "  Destination IP:  ";
                print_ip_address(ip_header.dest_ip);
                std::cout << std::endl;

                // Пропускаем оставшуюся часть IP-пакета, чтобы перейти к
следующему кадру
                // Общая длина IP-пакета хранится в ip_header.total_len
                long remaining_bytes = ntohs(ip_header.total_len) -
sizeof(IpHeader);
                if (remaining_bytes > 0) {
                    file.seekg(remaining_bytes, std::ios::cur);
                    frame_data_len += remaining_bytes;
                }

            } else if (type_len == 0x0806) {  // ARP-пакет
                arp_frames++;
                std::cout << "  Upper Protocol:  ARP (0x0806)" << std::endl;
                // Размер ARP-пакета стандартный - 28 байт
                file.seekg(28, std::ios::cur);
                frame_data_len = 28;
            } else {
                std::cout << "  Upper Protocol:  Unknown (0x" << std::hex <<
type_len << std::dec << ")" << std::endl;
                // Неизвестный тип Ethernet II. Пропускаем остаток кадра.
                // В реальном анализаторе нужно было бы смотреть на размер
пакета,
                // но здесь для простоты считаем его минимальным.
                // Длина поля данных должна быть минимум 46 байт
                file.seekg(46, std::ios::cur);
                frame_data_len = 46;
            }

        } else {  // Это кадр IEEE 802.3
            std::cout << "  Frame Length:    " << type_len << " bytes" <<
std::endl;

            // Читаем первые два байта поля данных, чтобы определить подтип
            unsigned short data_header;
            file.read((char*)&data_header, sizeof(data_header));

            if (data_header == 0xFFFF) {
                novell_raw_frames++;
```

```
                std::cout << "  Frame Type:      IEEE 802.3 Novell RAW" <<
std::endl;
            } else if (data_header == 0xAAAA) {
                snap_frames++;
                std::cout << "  Frame Type:      IEEE 802.3 SNAP" <<
std::endl;
            } else {
                llc_frames++;
                std::cout << "  Frame Type:      IEEE 802.3 LLC" <<
std::endl;
            }

            // Пропускаем оставшуюся часть поля данных
            // Мы уже прочитали 2 байта из type_len, так что пропускаем
остаток
            if (type_len > 2) {
                file.seekg(type_len - 2, std::ios::cur);
            }
            frame_data_len = type_len;
        }

        std::cout << "  Frame Size:      " << sizeof(EthernetHeader) +
frame_data_len << " bytes" << std::endl;
    }

    // Вывод итоговой статистики
    std::cout << "\n=====================================\n";
    std::cout << "           Analysis Summary\n";
    std::cout << "=====================================\n";
    std::cout << "Total frames processed: " << frame_counter << std::endl;
    std::cout << "\n--- Ethernet II (DIX) Frames ---\n";
    std::cout << "  IP Frames:            " << ip_frames << std::endl;
    std::cout << "  ARP Frames:           " << arp_frames << std::endl;
    std::cout << "\n--- IEEE 802.3 Frames ---\n";
    std::cout << "  LLC Frames:           " << llc_frames << std::endl;
    std::cout << "  Novell RAW Frames:    " << novell_raw_frames <<
std::endl;
    std::cout << "  SNAP Frames:          " << snap_frames << std::endl;
    std::cout << "=====================================\n";

    file.close();

    return 0;
}
```