|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 4 | | |
| по дисциплине «Компьютерные сети» | | |
| **Анализ структуры кадра/фрейма технологии Ethernet** | | |
|  | | |
|  | Бригада 4 | буров евгений |
| Группа ПМ-23 | Гайченко максим |
|  |  |
| Преподаватели | Кобылянский валерий григорьевич |
|  | сивак мария алексеевна |
|  |  |
|  |  |
|  |  |
| Новосибирск, 2025 | | |

1. Цель работы

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

1. Задание

Разработать и отладить программу, выполняющую анализ потока кадров. Потокикадров представлены в виде файлов двоичного формата, место нахождения которых уточняется у преподавателя. В кадрах отсутствует преамбула и контрольная сумма, для исходящего кадра длина может быть меньше минимальной. Каждая бригада выполняет обработку одного файла с именем, соответствующим номеру бригады.

1. Текст программы

#include <iostream>

#include <string>

#include <windows.h>

#include <filesystem>

#include <vector>

#include <fstream>

#include <iomanip>

void process\_file(std::string& filename) {

std::ifstream file(filename, std::ios::binary);

if (!file) {

std::cerr << "Failed to open file." << std::endl;

return;

}

char header[14];

file.read(header, 14);

int count\_ipv4 = 0, count\_arp = 0, count\_eth\_dix = 0, count\_eth\_snap = 0, count\_novell\_802\_2 = 0, count\_novell\_802\_3 = 0;

for (int i = 1; file && !file.eof(); i++, file.read(header, 14)) {

int size = 14;

std::cout << "Frame #" << std::dec << i << std::endl;

std::cout << " MAC addresses: ";

for (int j = 0; j < 6; j++) {

std::cout << std::hex << std::setw(2) << std::setfill('0') << (0xFF & static\_cast<unsigned int>(header[j]));

if (j != 5) std::cout << ":";

}

std::cout << " <= ";

for (int j = 0; j < 6; j++) {

std::cout << std::hex << std::setw(2) << std::setfill('0') << (0xFF & static\_cast<unsigned int>(header[6 + j]));

if (j != 5) std::cout << ":";

}

std::cout << std::endl;

unsigned short type = ((0xFF & header[12]) << 8) | (0xFF & header[13]);

if (type > 0x0600) {

if (type == 0x800) {

std::cout << " Type: IPv4" << std::endl;

char ip\_header[20];

file.read(ip\_header, 20);

size += 20;

std::cout << " IP addresses: ";

for (int j = 0; j < 4; j++) {

std::cout << std::dec << (0xFF & static\_cast<unsigned int>(ip\_header[12 + j]));

if (j != 3) std::cout << ".";

}

std::cout << " => ";

for (int j = 0; j < 4; j++) {

std::cout << std::dec << (0xFF & static\_cast<unsigned int>(ip\_header[16 + j]));

if (j != 3) std::cout << ".";

}

std::cout << std::endl;

unsigned short total\_length = ((0xFF & ip\_header[2]) << 8) | (0xFF & ip\_header[3]);

file.seekg(total\_length - 20, std::ios::cur);

size += total\_length - 20;

count\_ipv4++;

}

else if (type == 0x806) {

std::cout << " Type: ARP" << std::endl;

file.seekg(28, std::ios::cur);

size += 28;

count\_arp++;

}

else {

std::cout << " Type: Ethernet DIX (Ethernet II)" << std::endl;

file.seekg(type, std::ios::cur);

size += type;

count\_eth\_dix++;

}

}

else {

char dsap\_ssap[2];

file.read(dsap\_ssap, 2);

size += 2;

if (dsap\_ssap[0] == 0xAA && dsap\_ssap[1] == 0xAA) {

std::cout << " Type: Ethernet SNAP" << std::endl;

file.seekg(type - 2, std::ios::cur);

size += type - 2;

count\_eth\_snap++;

}

else if (type <= 0x05DC) {

std::cout << " Type: Novell 802.2" << std::endl;

file.seekg(type - 2, std::ios::cur);

size += type - 2;

count\_novell\_802\_2++;

}

else if (type == 0xFFFF) {

std::cout << " Type: Novell 802.3" << std::endl;

file.seekg(type - 2, std::ios::cur);

size += type - 2;

count\_novell\_802\_3++;

}

}

std::cout << " Size: " << std::dec << size << std::endl;

}

std::cout << std::endl << "Processed frames: " << std::dec << count\_ipv4 + count\_arp + count\_eth\_dix + count\_eth\_snap + count\_novell\_802\_2 + count\_novell\_802\_3 << std::endl;

std::cout << " IPv4: " << count\_ipv4 << std::endl;

std::cout << " ARP: " << count\_arp << std::endl;

std::cout << " Ethernet DIX (Ethernet II): " << count\_eth\_dix << std::endl;

std::cout << " Ethernet SNAP: " << count\_eth\_snap << std::endl;

std::cout << " Novell 802.2: " << count\_novell\_802\_2 << std::endl;

std::cout << " Novell 802.3: " << count\_novell\_802\_3 << std::endl;

file.close();

}

int main() {

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string filename;

std::cout << "Filename: ";

std::getline(std::cin, filename);

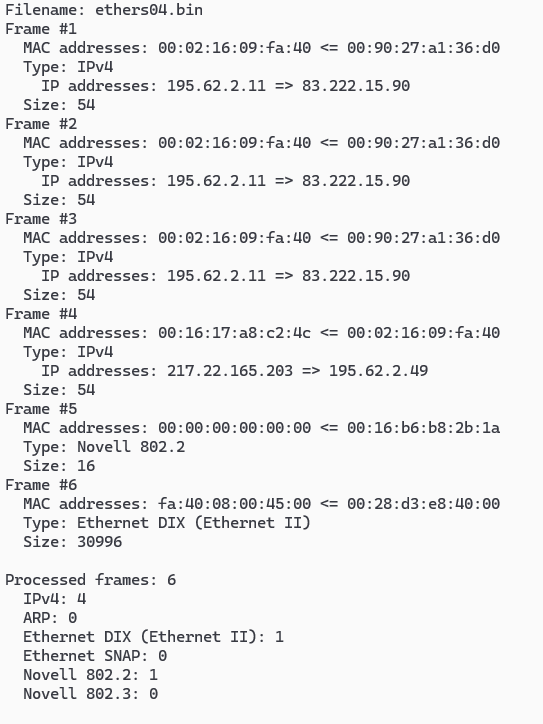
process\_file(filename);

return 0;

}

1. Ход работы

*Обработка потока кадров ethers04.bin:*

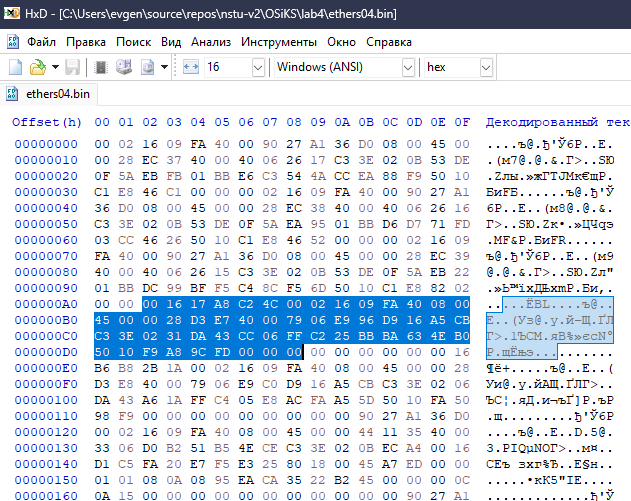
**

*Полный анализ кадра с номером 4:*

Найдём начало кадра №4. Так как адресация начинается с нуля, то адресом его начала будет сумма размеров предыдущих кадров.

*.*

Нужный адрес: 0x000000A2. Конец фрейма: 162+54=216=D8, 0x000000D8.



|  |  |  |
| --- | --- | --- |
| **Ethernet-заголовок** | | |
| **Байты** | **Поле** | **Значение/Описание** |
| 00 16 17 A8 C2 4C | MAC назначения | 00:16:17:A8:C2:4C (получатель) |
| 00 02 16 09 FA 40 | MAC источника | 00:02:16:09:FA:40 (отправитель) |
| 08 00 | EtherType | 0x0800 = IPv4 |

|  |  |  |
| --- | --- | --- |
| **IPv4-заголовок** | | |
| **Байты** | **Поле** | **Значение/Описание** |
| 45 | Версия/IHL | 4 (IPv4), 5 (длина заголовка = 20 байт) |
| 00 | DSCP/ECN | Без приоритета |
| 00 28 | Общая длина | 40 байт (IPv4 + TCP заголовки) |
| D3 E7 | Идентификатор | 0xD3E7 (для фрагментации) |
| 40 00 | Флаги/Смещение | 0x4000 = Не фрагментировать (DF) |
| 79 | Время жизни | 121 (осталось прыжков) |
| 06 | Протокол | 6 = TCP |
| E9 96 | Контрольная сумма | Проверка целостности заголовка |
| D9 16 A5 CB | IP источника | 217.22.165.203 |
| C3 3E 02 31 | IP назначения | 195.62.2.49 |

|  |  |  |
| --- | --- | --- |
| **TCP-заголовок** | | |
| **Байты** | **Поле** | **Значение/Описание** |
| DA 43 | Порт источника | 55875 (клиентский порт) |
| CC 06 | Порт назначения | 52230 (серверный порт) |
| FF C2 25 BB | Seq Number | 0xFFC225BB (порядковый номер) |
| BA 63 4E B0 | Ack Number | 0xBA634EB0 (подтверждение) |
| 50 10 | Флаги/Окно | 0x50 (длина заголовка: 20 байт), 0x10 = ACK |
| F9 A8 | Размер окна | 63912 байт (буфер получателя) |
| 9C FD | Контрольная сумма | Проверка TCP + псевдо-заголовка |
| 00 00 | Urgent Pointer | 0 (не используется) |