Министерство науки и высшего образования

Российской федерации

Федеральное государственное бюджетное

Образовательное учреждение высшего образования

«Новосибирский Государственный Технический Университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №4

«Анализ структуры кадра/фрейма технологии Ethernet»

Факультет: прикладной математики и информатики

Группа: ПМИ-12

Бригада: 1

Студенты: Михайловский М.А.

Швадченко А.В.

Преподаватели: Кобылянский В.Г.

Филиппова Е.В.

Новосибирск, 2023

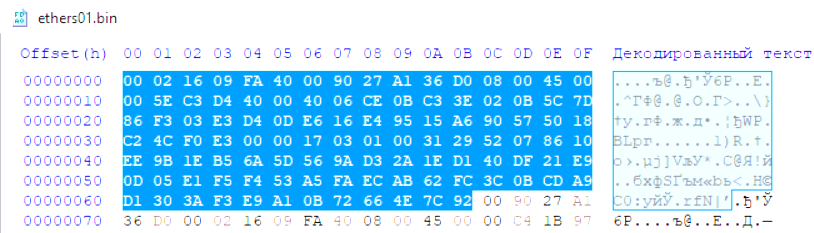
1. Цель работы:

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

1. Содержание работы

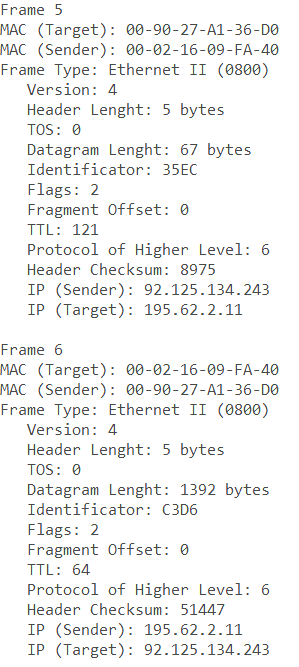
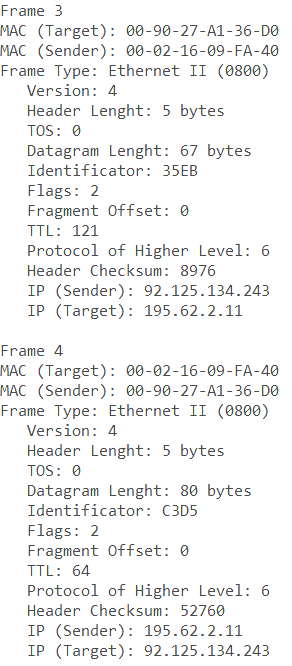
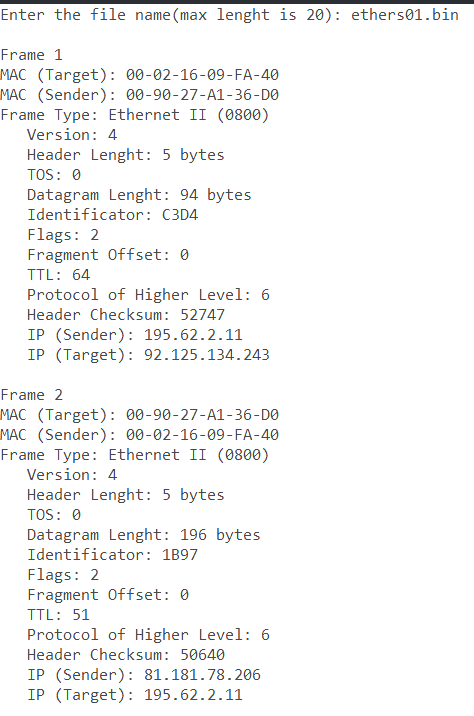
Разработать и отладить программу, выполняющую анализ потока кадров. Согласно варианту, выполняется обработка файла: ethers01.bin

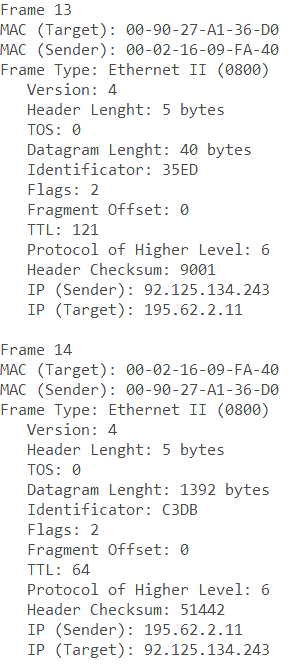
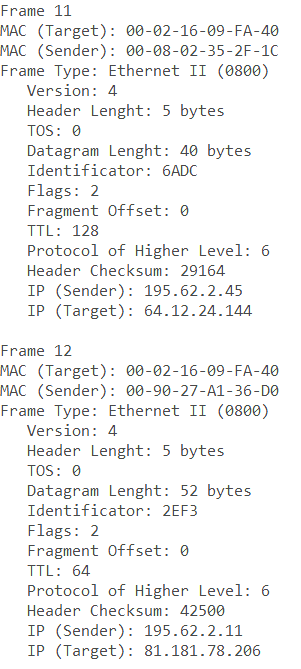
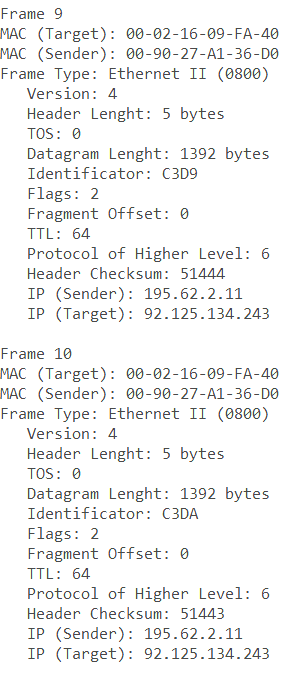
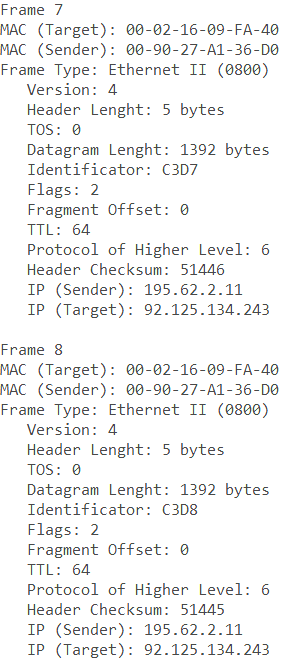
1. Выполнение работы:
   1. Анализ кадра 1

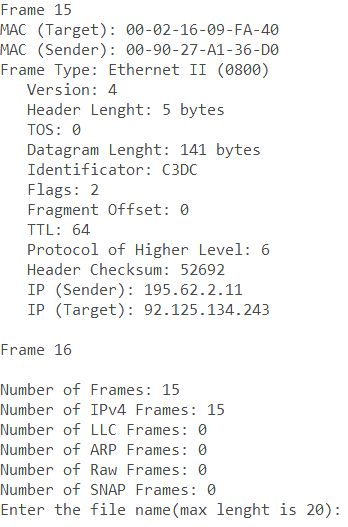


|  |  |  |
| --- | --- | --- |
| HEX | Имя поля | Описание |
| 00 02 16 09 FA 40 | MAC-адрес получателя (6 байт) | Адрес доступа 00-02-16-FA-40 |
| 00 90 27 A1 36 D0 | MAC-адрес отправителя (6 байт) | Адрес доступа 00-09-27-A1-36-D0 |
| 08 00 | Тип протокола(2 байта) | IPv4 |
| 4 | Версия протокола (4 бита) | Для IPv4 версия 4 |
| 5 | Размер заголовка(4 бита) | Размер заголовка 20 байт |
| 00 | Тип службы(1 байт) | Определяется в RFC 2474 |
| 00 5E | Полный размер пакета(2 байта) | Размер всего пакета 94 байта |
| C3 D4 | Идентификатор(2 байта) | Используется для фрагментов пакета при фрагментации |
| 0102 | Флаги (3 бита) | Флаг 2 означает, что у пакета ещё есть фрагменты. |
| 0 0000 00002 | Смещение (13 бит) | Нулевое смещение в фрагментации означает, что этот  пакет первый. |
| *40* | Время жизни (1 байт) | Определяет максимальное количество маршрутизаторов на  пути следования пакета.  В данном случае 64. |
| *06* | Протокол верхнего уровня(1 байт) | Номер 6 означает, что  используется протокол TCP. |
| *CE 0B* | Контрольная сумма заголовка  (2 байта) | 16-битная контрольная сумма, используемая для проверки  целостности заголовка. |
| *C3 E2 02 0B* | IP-адрес источника (4 байта) | 195.62.2.11 |
| *5C 7D 86 F3* | IP-адрес назначения (4 байта) | 92.125.134.243 |
| *03 E3* | Порт источника (2 байта) | Порт источника идентифицирует приложение клиента, с которого отправлены пакеты. В данном случае порт – 995. |
| *D4 0D* | Порт назначения (2 байта) | Порт назначения идентифицирует порт, на который отправлен пакет.  В данном случае 54285. |
| *E6 16 E4 95* | Порядковый номер (4 байта) | Порядковый номер измеряется в байтах, и каждый переданный байт полезных данных (payload) увеличивает это значение на 1. |
| *15 A6 90 57* | Номер подтверждения (4 байта) | Так как установлен флаг ACK, то это поле содержит порядковый номер октета, который  отправитель данного сегмента желает получить. |
| *5* | Длина заголовка (4 бита) | Указывает значение длины заголовка в 32-битовых словах. В  данном случае 20 байт. |
| *0000 002* | Зарезервировано (6 бит) | 6 битов для будущего использования. |
| *01 10002* | Флаги (6 бит) | 6 битовых флагов. В данном случае работают:  ACK — поле «Номер  подтверждения» задействовано. PSH - инструктирует получателя протолкнуть данные,  накопившиеся в приёмном буфере, в приложение  пользователя. |
| *C2 4C* | Размер окна (2 байта) | Определяет количество байт данных (payload), после передачи которых отправитель ожидает  подтверждения от получателя, что  данные получены. |
| *F0 E3* | Контрольная сумма (2 байта) | Это 16-битное дополнение к  сумме всех 16-битных слов заголовка и данных. |
| *00 00* | Указатель важности (2 байта) | Это поле указывает порядковый номер октета, которым  заканчиваются важные данные. Поле принимается во внимание только для пакетов с  установленным флагом URG. |
|  | Данные | Сами данные, которые мы и хотели передать(payload) |

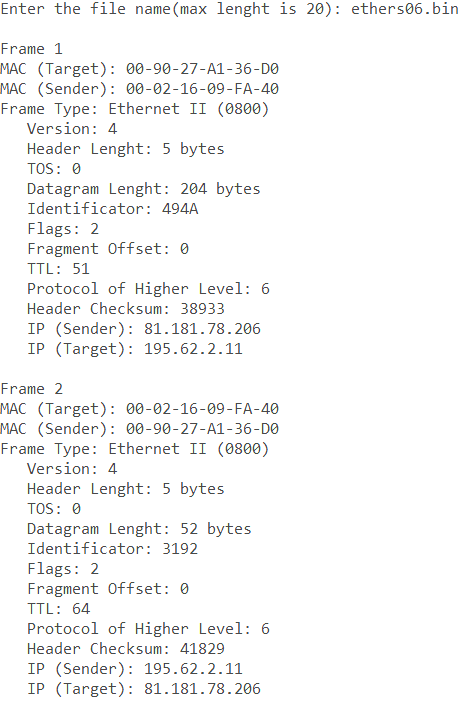
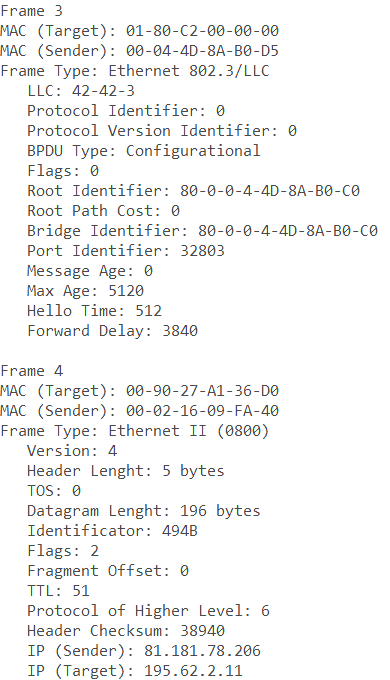
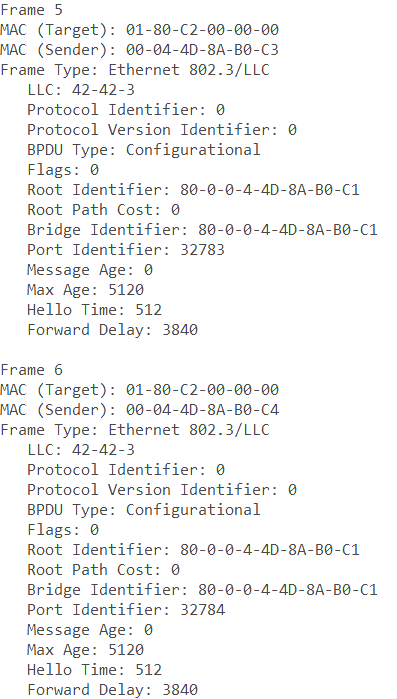
* 1. Анализ файла ethers01.bin

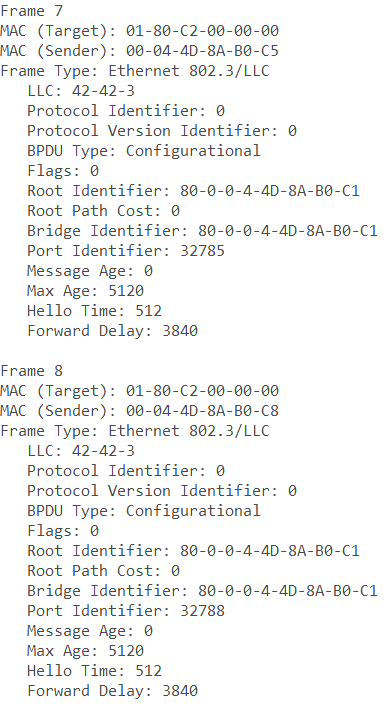
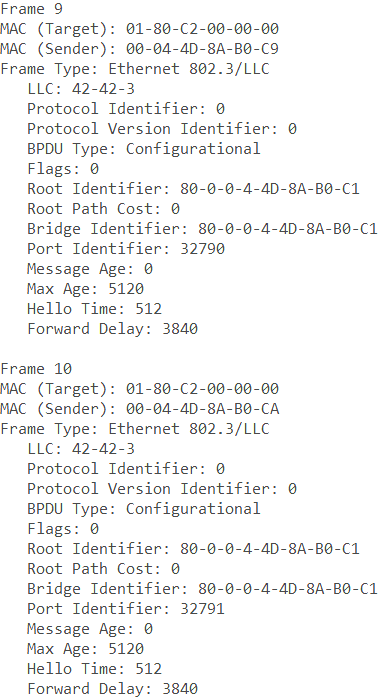
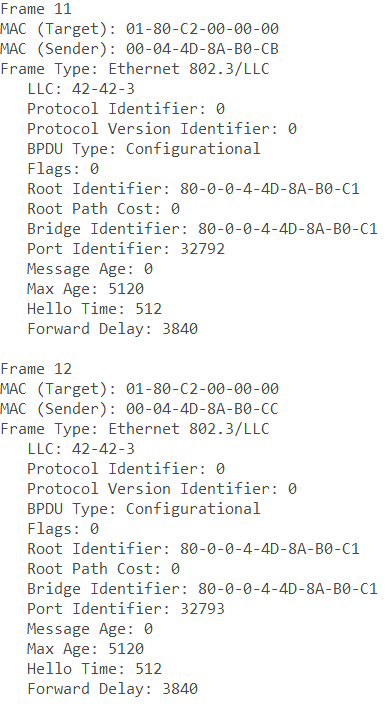
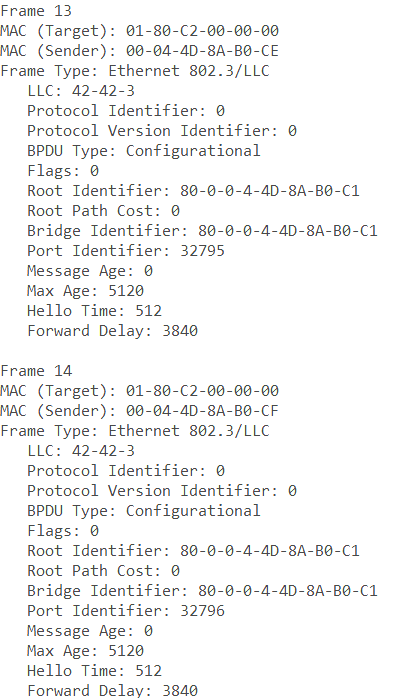
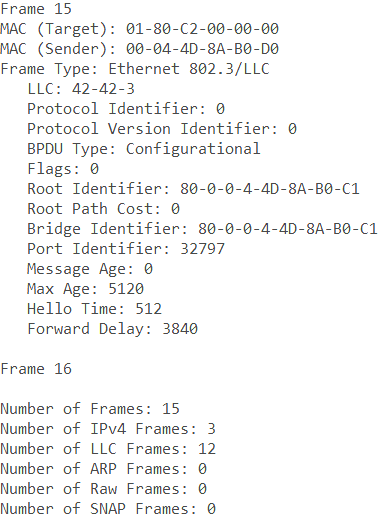




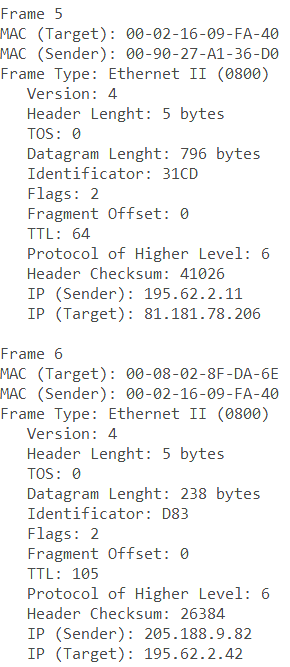
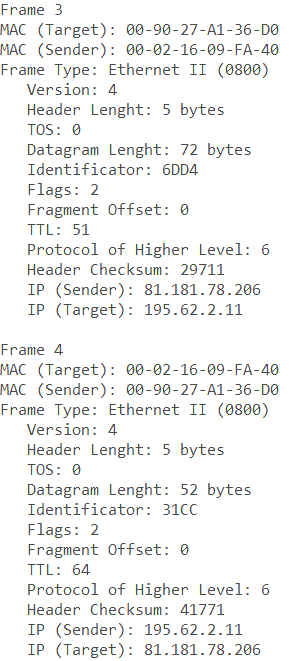
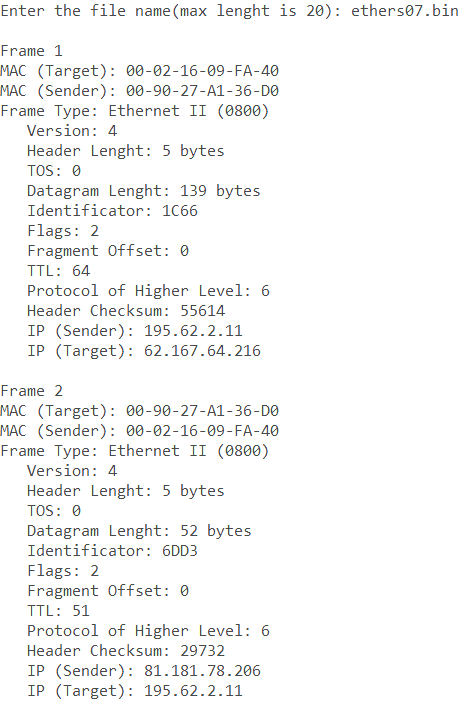


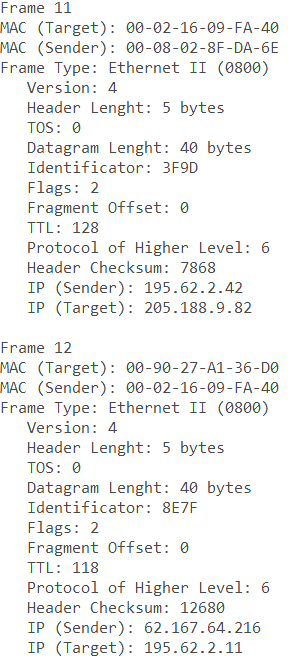
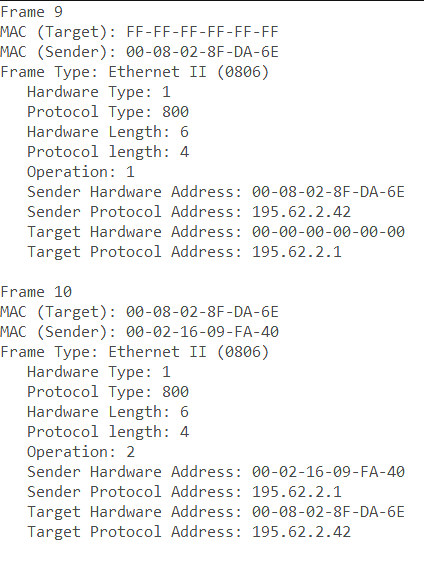
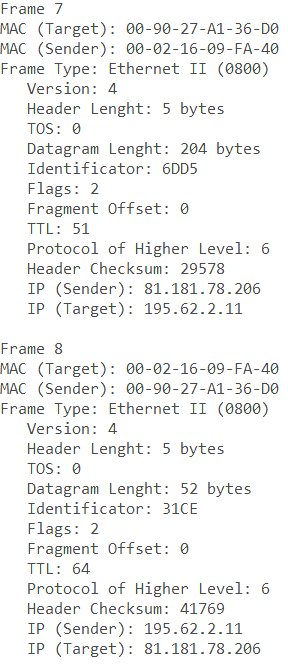
* 1. Анализ файла ethers06.bin

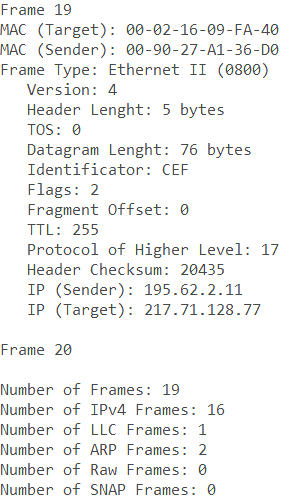
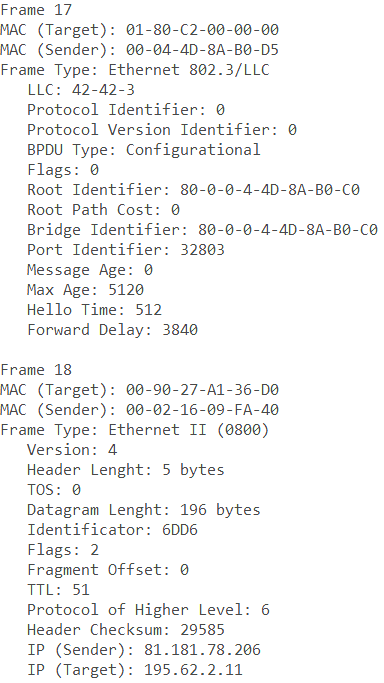
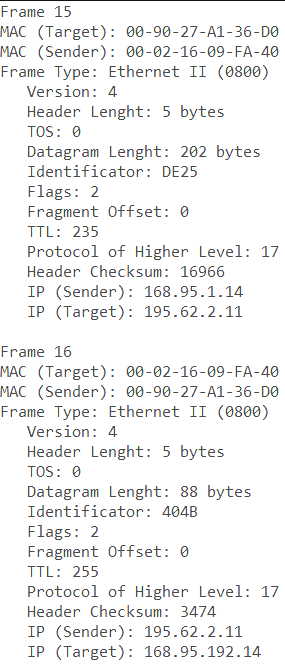
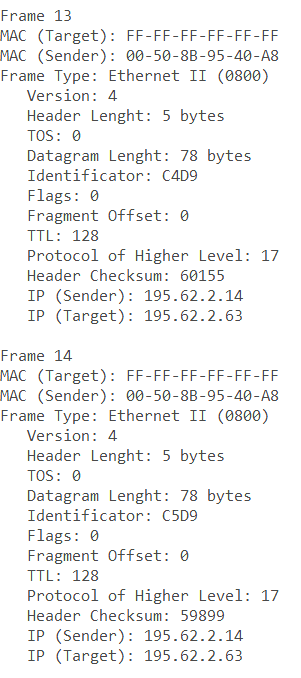
  

* 1. Анализ файла ethers07.bin







* 1. Код программы

Модуль main.cpp

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <iostream>

#include "FrameLib.h"

int main()

{

int numRaw;

int numSNAP;

int numLLC;

int numIPv4;

int numARP;

while (true) {

numRaw = 0;

numSNAP = 0;

numLLC = 0;

numIPv4 = 0;

numARP = 0;

char fileName[20];

int i = 1;

printf("Enter the file name(max lenght is 20): ");

gets\_s(fileName);

FILE\* in;

int check = fopen\_s(&in, fileName, "rb");

if (check == 0) {

do

{

printf("\nFrame %d\n", i);

i++;

} while (!readFrame(in, numRaw, numSNAP, numLLC, numIPv4, numARP));

fclose(in);

printStat(i, numRaw, numSNAP, numLLC, numIPv4, numARP);

}

else {

std::cout << "file reading Err" << std::endl;

break;

}

}

return 0;

}

Модуль FrameLib.h

#pragma once

#include <stdio.h>

#include <conio.h>

#include <string.h>

const int maxFrameLength = 0x05FE;

const int Ethernet802\_3 = 0xFFFF;

const int EthernetSNAP = 0xAA;

const int numMAC = 6;

int readFrame(FILE\* in, int& numRaw, int& numSNAP, int& numLLC, int& numIPv4, int& numARP);

int readDatagram(FILE\* in);

int BPDU(FILE\* in);

int ARP(FILE\* in);

void printStat(int i, int numRaw, int numSNAP, int numLLC, int numIPv4, int numARP);

Модуль FrameLib.cpp

#include "FrameLib.h"

int readFrame(FILE\* in, int& numRaw, int& numSNAP, int& numLLC, int& numIPv4, int& numARP)

{

int i;

bool flag = false;

unsigned char MAC[numMAC], BUF[2], OUI[3], LLC3;

unsigned short int orgType, firstTwoBytes, typeLength;

while (!flag)

{

if (fread(MAC, 1, numMAC, in) != numMAC)

return 1;

for (i = 0; i < numMAC; i++)

flag = flag || MAC[i];

}

printf("MAC (Target): ");

printf("%02X", MAC[0]);

for (i = 1; i < numMAC; i++)

printf("-%02X", MAC[i]);

fread(MAC, 1, numMAC, in);

printf("\nMAC (Sender): ");

printf("%02X", MAC[0]);

for (i = 1; i < numMAC; i++)

printf("-%02X", MAC[i]);

fread(BUF, 1, 2, in);

typeLength = (BUF[0] << 8) | BUF[1];

if (typeLength > maxFrameLength)

{

printf("\nFrame Type: Ethernet II (%04X)\n", typeLength);

if (typeLength == 0x0800)

{

readDatagram(in);

numIPv4++;

}

if (typeLength == 0x0806)

{

ARP(in);

numARP++;

}

}

else

{

fread(BUF, 1, 2, in);

firstTwoBytes = (BUF[0] << 8) | BUF[1];

if (firstTwoBytes == Ethernet802\_3)

{

printf("\nFrame Type: Ethernet 802.3 Raw\n");

numRaw++;

printf(" Length: %d", firstTwoBytes);

readDatagram(in);

}

else if (BUF[0] == EthernetSNAP && BUF[1] == EthernetSNAP)

{

printf("\nFrame Type: EthernetSNAP\n");

fread(&LLC3, 1, 1, in);

printf(" LLC: %2X-%2X-%2X\n", BUF[0], BUF[1], LLC3);

fseek(in, 1, SEEK\_CUR);

fread(OUI, 1, 3, in);

printf(" Organization by Standart:");

for (i = 0; i < 2; i++)

printf("%02X-", OUI[i]);

printf("%02X\n", OUI[2]);

fread(BUF, 1, 2, in);

orgType = (BUF[0] << 8) | BUF[1];

printf(" Frame Type by Organization: (%04X)\n", orgType);

readDatagram(in);

numSNAP++;

}

else

{

printf("\nFrame Type: Ethernet 802.3/LLC\n");

fread(&LLC3, 1, 1, in);

printf(" LLC: %0X-%0X-%0X\n", BUF[0], BUF[1], LLC3);

if (BUF[0] == 0x6 && BUF[1] == 0x6)

readDatagram(in);

if (BUF[0] == 0x42 && BUF[1] == 0x42)

BPDU(in);

numLLC++;

}

}

return 0;

}

int readDatagram(FILE\* in)

{

int i;

unsigned short int datagramLength, ident, BUF2, checksum;

unsigned char BUF, TOS, TTL, protocol, bufArr[2], ipSender[4], ipTarget[4];

fread(&BUF, 1, 1, in);

unsigned char version = (BUF & 0xF0) >> 4;

unsigned char headerLength = BUF & 0x0F;

unsigned char flags;

unsigned short offset;

printf(" Version: %X\n", version);

printf(" Header Lenght: %d bytes\n", headerLength);

fread(&TOS, 1, 1, in);

printf(" TOS: %X\n", TOS);

fread(&bufArr, 2, 1, in);

datagramLength = (bufArr[0] << 8) | bufArr[1];

printf(" Datagram Lenght: %d bytes\n", datagramLength);

fread(&bufArr, 2, 1, in);

ident = (bufArr[0] << 8) | bufArr[1];

printf(" Identificator: %X\n", ident);

fread(&bufArr, 2, 1, in);

BUF2 = (bufArr[0] << 8) | bufArr[1];

flags = (BUF2 & 0xE000) >> 13;

printf(" Flags: %X\n", flags);

offset = (BUF2 & 0x1FFF);

printf(" Fragment Offset: %d\n", offset);

fread(&TTL, 1, 1, in);

printf(" TTL: %d\n", TTL);

fread(&protocol, 1, 1, in);

printf(" Protocol of Higher Level: %d\n", protocol);

fread(&bufArr, 2, 1, in);

checksum = (bufArr[0] << 8) | bufArr[1];

printf(" Header Checksum: %d\n", checksum);

fread(&ipSender, 1, 4, in);

printf(" IP (Sender): ");

for (i = 0; i < 3; i++)

printf("%d.", ipSender[i]);

printf("%d\n", ipSender[3]);

printf(" IP (Target): ");

fread(&ipTarget, 1, 4, in);

for (i = 0; i < 3; i++)

printf("%d.", ipTarget[i]);

printf("%d\n", ipTarget[3]);

fseek(in, datagramLength - 20, SEEK\_CUR);

return 0;

}

int BPDU(FILE\* in)

{

int i;

unsigned int rootPathCost;

unsigned char BUF, BUF2[2], BUF4[4], BUF8[8];

short unsigned protID, portIdent, msgAge, maxAge, helloTime, forwardDelay;

fread(&protID, 2, 1, in);

printf(" Protocol Identifier: %X\n", protID);

fread(&BUF, 1, 1, in);

printf(" Protocol Version Identifier: %X\n", BUF);

fread(&BUF, 1, 1, in);

if (BUF == 0x00)

printf(" BPDU Type: Configurational\n");

else

printf(" BPDU Type: Topology Changed\n");

fread(&BUF, 1, 1, in);

printf(" Flags: %X\n", BUF);

fread(&BUF8, 1, 8, in);

printf(" Root Identifier: ");

for (i = 0; i < 7; i++)

printf("%X-", BUF8[i]);

printf("%X\n", BUF8[7]);

fread(&BUF4, 1, 4, in);

rootPathCost = (BUF4[0] << 32) | (BUF4[1] << 16) | (BUF4[2] << 8) | BUF4[3];

printf(" Root Path Cost: %d\n", rootPathCost);

fread(&BUF8, 1, 8, in);

printf(" Bridge Identifier: ");

for (i = 0; i < 7; i++)

printf("%X-", BUF8[i]);

printf("%X\n", BUF8[7]);

fread(&BUF2, 1, 2, in);

portIdent = (BUF2[0] << 8) | BUF2[1];

printf(" Port Identifier: %d\n", portIdent);

fread(&BUF2, 1, 2, in);

msgAge = (BUF2[0] << 8) | BUF2[1];

printf(" Message Age: %d\n", msgAge);

fread(&BUF2, 1, 2, in);

maxAge = (BUF2[0] << 8) | BUF2[1];

printf(" Max Age: %d\n", maxAge);

fread(&BUF2, 1, 2, in);

helloTime = (BUF2[0] << 8) | BUF2[1];

printf(" Hello Time: %d\n", helloTime);

fread(&BUF2, 1, 2, in);

forwardDelay = (BUF2[0] << 8) | BUF2[1];

printf(" Forward Delay: %d\n", forwardDelay);

return 0;

}

int ARP(FILE\* in)

{

int i;

unsigned char hlen, plen, BUF2[2], BUF4[4];

unsigned char\* bufnh, \* bufnp;

short unsigned hwType, protocolType, operation;

fread(&BUF2, 1, 2, in);

hwType = (BUF2[0] << 8) | BUF2[1];

printf(" Hardware Type: %X\n", hwType);

fread(&BUF2, 1, 2, in);

protocolType = (BUF2[0] << 8) | BUF2[1];

printf(" Protocol Type: %X\n", protocolType);

fread(&hlen, 1, 1, in);

printf(" Hardware Length: %d\n", hlen);

fread(&plen, 1, 1, in);

printf(" Protocol length: %d\n", plen);

bufnh = new unsigned char[hlen];

bufnp = new unsigned char[plen];

fread(&BUF2, 1, 2, in);

operation = (BUF2[0] << 8) | BUF2[1];

printf(" Operation: %X\n", operation);

fread(bufnh, 1, hlen, in);

printf(" Sender Hardware Address: ");

for (i = 0; i < hlen - 1; i++)

printf("%02X-", bufnh[i]);

printf("%02X\n", bufnh[i]);

fread(bufnp, 1, plen, in);

printf(" Sender Protocol Address: ");

for (i = 0; i < plen - 1; i++)

printf("%d.", bufnp[i]);

printf("%d\n", bufnp[i]);

fread(bufnh, 1, hlen, in);

printf(" Target Hardware Address: ");

for (i = 0; i < hlen - 1; i++)

printf("%02X-", bufnh[i]);

printf("%02X\n", bufnh[i]);

fread(bufnp, 1, plen, in);

printf(" Target Protocol Address: ");

for (i = 0; i < plen - 1; i++)

printf("%d.", bufnp[i]);

printf("%d\n", bufnp[i]);

return 0;

}

void printStat(int i, int numRaw, int numSNAP, int numLLC, int numIPv4, int numARP)

{

printf("\n");

printf("Number of Frames: %d\n", i - 2);

printf("Number of IPv4 Frames: %d\n", numIPv4);

printf("Number of LLC Frames: %d\n", numLLC);

printf("Number of ARP Frames: %d\n", numARP);

printf("Number of Raw Frames: %d\n", numRaw);

printf("Number of SNAP Frames: %d\n", numSNAP);

}