Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

Кафедра теоретической прикладной информатики

Лабораторная работа № 4
по дисциплине «Операционные системы, среды и оболочки»

Анализ структуры кадра/фрейма технологии Ethernet

Факультет:	ПМИ
Группа:	ПМ-92
Бригада:	8
Студенты:	Иванов В., Кутузов И.
Преподаватель:	Кобылянский В. Г.

Новосибирск

2021

Цель работы

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

Задание

Разработать программу, выполняющую анализ потока кадров (вывод на экран информации по каждому кадру и итоговой статистики). Потоки кадров представлены в виде файла двоичного формата ethers08.bin. В кадрах отсутствует преамбула и контрольная сумма, для исходящего кадра длина может быть меньше минимальной. Предусмотреть возможность ввода имени файла с клавиатуры. Выполнить полный анализ кадра с номером 8.

Код программы

```
#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 numRaw = 0;
int numSNAP = 0;
int numLLC = 0;
int numIPv4 = 0;
int numARP = 0;
int readFrame(FILE* in);
int readDatagram(FILE* in);
int BPDU(FILE* in);
int ARP(FILE* in);
void printStat(int);

int main()
{
    char fileName[20];
    int i = 1;

    printf("Enter the file name: ");
    gets_s(fileName); // читаем название файла
    FILE* in;
    fopen_s(&in, fileName, "rb+"); // открываем файл

    do
    {
        printf("\nFrame %d\n", i); // выводим номер кадра
        i++;
    } while (!readFrame(in)); // перебираем все кадры

    printStat(i); // выводим общую статистику

    return 0;
```

```
}

int readFrame(FILE* in)
{
    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); // записываем в MAC 6 байтов из in
    printf("\nMAC (Sender): ");
    printf("%02X", MAC[0]); // выводим первый байт MAC-адреса
    for (i = 1; i < numMAC; i++) // через дефис выводим остальные 5
        printf("-%02X", MAC[i]);

    fread(BUF, 1, 2, in); // записываем в буфер длину кадра (2 байта)
    typeLength = (BUF[0] << 8) | BUF[1]; // побитовое ‘или’ между вторым
байтом и сдвигом влево на 8 бит первого байта вторым байтом

    if (typeLength > maxFrameLength) // если больше 0x05FE - это DIX
    {
        printf("\nFrame Type: Ethernet II (%04X)\n", typeLength);
        if (typeLength == 0x0800) // если 0800 - это IPv4
        {
            readDatagram(in); // переходим к обработке IPv4-дейтаграммы
            numIPv4++;
        }
        if (typeLength == 0x0806) // если 0806 - это ARP
        {
            ARP(in); // переходим к обработке ARP
            numARP++;
        }
    }
    else // иначе продолжаем проверку: если первые 2 байта равны 0xFFFF
    {
        fread(BUF, 1, 2, in);
        firstTwoBytes = (BUF[0] << 8) | BUF[1];
        if (firstTwoBytes == Ethernet802_3)
        {
            printf("\nFrame Type: Ethernet 802.3 Raw\n");
            numRaw++; // то это RAW
            printf("   Length: %d", firstTwoBytes);
            readDatagram(in); // переходим к обработке IPv4-дейтаграммы
        } // в противном случае если оба байта равны 0xAA - это SNAP
```

```
        else if (BUF[0] == EthernetSNAP && BUF[1] == EthernetSNAP)
        {
            printf("\nFrame Type: EthernetSNAP\n");

            fread(&LLC3, 1, 1, in); // выводим его LLC-заголовок
            printf("   LLC: %2X-%2X-%2X\n", BUF[0], BUF[1], LLC3);
            fseek(in, 1, SEEK_CUR);

            // и SNAP-заголовок (OUI и T)
            fread(OUI, 1, 3, in); // организационно уникальный идентификатор
            printf("   Organization by Standart:");
            for (i = 0; i < 2; i++)
                printf("%02X-", OUI[i]);
            printf("%02X\n", OUI[2]);

            fread(BUF, 1, 2, in); // тип, задаваемый организацией, его
уникальность обеспечивается OUI
            orgType = (BUF[0] << 8) | BUF[1];
            printf("   Frame Type by Organization: (%04X)\n", orgType);

            readDatagram(in); // переходим к обработке IPv4-дейтаграммы
            numSNAP++;
        }
        else // если ничего не подходит - это обычный Ethernet_802.3/802.2
        {
            printf("\nFrame Type: Ethernet 802.3/LLC\n");

            fread(&LLC3, 1, 1, in); // выводим его LLC-заголовок
            printf("   LLC: %0X-%0X-%0X\n", BUF[0], BUF[1], LLC3);

            // проверяем тип пакета
            if (BUF[0] == 0x6 && BUF[1] == 0x6)
                readDatagram(in); // переходим к обработке IPv4-дейтаграммы
            if (BUF[0] == 0x42 && BUF[1] == 0x42)
                BPDU(in); // переходим к обработке BPDU
            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; // версия протокола IP
    unsigned char headerLength = BUF & 0x0F; // длина заголовка для
определения его конца и начала данных
    unsigned char flags;
    unsigned short offset;

    printf("   Version: %X\n", version);
    printf("   Header Lenght: %d bytes\n", headerLength);
```

```
    fread(&TOS, 1, 1, in); // тип службы для возможности разделять
IP-дейтаграммы на типы
    printf("   TOS: %X\n", TOS);

    fread(&bufArr, 2, 1, in); // полная длина дейтаграммы
    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); // время жизни для предотвращения вечной
циркуляции в сети
    printf("   TTL: %d\n", TTL);

    fread(&protocol, 1, 1, in); // какому протоколу транспортного уровня
передать данные из дейтаграммы (6 - TCP, 17 - UDP)
    printf("   Protocol of Higher Level: %d\n", protocol);

    fread(&bufArr, 2, 1, in); // контрольная сумма для проверки целостности
    checksum = (bufArr[0] << 8) | bufArr[1];
    printf("   Header Checksum: %d\n", checksum);

    fread(&ipSender, 1, 4, in); // IP отправителя
    printf("   IP (Sender): ");
    for (i = 0; i < 3; i++)
        printf("%d.", ipSender[i]);
    printf("%d\n", ipSender[3]);

    printf("   IP (Target): "); // IP получателя
    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); // идентификатор версии протокола STP
    printf("   Protocol Identifier: %X\n", protID);

    fread(&BUF, 1, 1, in); // версия STP
    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); // идентификатор корневого коммутатора
    printf("   Root Identifier: ");
    for (i = 0; i < 7; i++)
        printf("%X-", BUF8[i]);
    printf("%X\n", BUF8[7]);

    fread(&BUF4, 1, 4, in); // расстояние до корневого коммутатора
    rootPathCost = (BUF4[0] << 32) | (BUF4[1] << 16) | (BUF4[2] << 8) | BUF4[3];
    printf("   Root Path Cost: %d\n", rootPathCost);

    fread(&BUF8, 1, 8, in); // идентификатор моста
    printf("   Bridge Identifier: ");
    for (i = 0; i < 7; i++)
        printf("%X-", BUF8[i]);
    printf("%X\n", BUF8[7]);

    fread(&BUF2, 1, 2, in); // идентификатор порта
    portIdent = (BUF2[0] << 8) | BUF2[1];
    printf("   Port Identifier: %d\n", portIdent);

    fread(&BUF2, 1, 2, in); // TTL
    msgAge = (BUF2[0] << 8) | BUF2[1];
    printf("   Message Age: %d\n", msgAge);

    fread(&BUF2, 1, 2, in); // max TTL
    maxAge = (BUF2[0] << 8) | BUF2[1];
    printf("   Max Age: %d\n", maxAge);

    fread(&BUF2, 1, 2, in); // интервал, через который посылаются пакеты BPDU
    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;

    // номер канального протокола (для Ethernet - 0x0001)
    fread(&BUF2, 1, 2, in);
    hwType = (BUF2[0] << 8) | BUF2[1];
    printf("   Hardware Type: %X\n", hwType);

    // код сетевого протокола
    fread(&BUF2, 1, 2, in);
    protocolType = (BUF2[0] << 8) | BUF2[1];
    printf("   Protocol Type: %X\n", protocolType);

    // длина физического адреса
    fread(&hlen, 1, 1, in);
    printf("   Hardware Length: %d\n", hlen);

    // длина логического адреса
    fread(&plen, 1, 1, in);
    printf("   Protocol Length: %d\n", plen);

    bufnh = new unsigned char[hlen];
    bufnp = new unsigned char[plen];

    // код операции отправителя (0x0001 - запрос, 0x0002 - ответ)
    fread(&BUF2, 1, 2, in);
    operation = (BUF2[0] << 8) | BUF2[1];
    printf("   Operation: %X\n", operation);

    fread(bufnh, 1, hlen, in); // физический адрес отправителя
    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); // логический адрес отправителя
    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); // физический адрес получателя
    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); // физический адрес получателя
    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)
{
    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);
}
```

Результаты

Результат анализа программой файла ethers08.bin

```
Frame 1
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 59 bytes
   Identificator: 1FBB
   Flags: 2
   Fragment Offset: 0
   TTL: 255
   Protocol of Higher Level: 17
   Header Checksum: 53805
   IP (Sender): 195.62.2.11
   IP (Target): 195.62.1.65
```

```
Frame 2
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-16-17-A8-C2-4C
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 1117 bytes
   Identificator: 546A
   Flags: 2
   Fragment Offset: 0
   TTL: 128
   Protocol of Higher Level: 6
   Header Checksum: 6008
   IP (Sender): 195.62.2.49
   IP (Target): 195.62.2.11
```

```
Frame 3
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 1064 bytes
   Identificator: EC84
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 8650
   IP (Sender): 195.62.2.11
   IP (Target): 83.222.15.90
```

```
Frame 4
MAC (Target): 00-16-17-A8-C2-4C
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 40 bytes
   Identificator: 939
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 42718
   IP (Sender): 195.62.2.11
   IP (Target): 195.62.2.49
```

```
Frame 5
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-02-16-09-FA-40
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 136 bytes
   Identificator: 4FA
   Flags: 2
   Fragment Offset: 0
   TTL: 51
   Protocol of Higher Level: 6
   Header Checksum: 56489
   IP (Sender): 81.181.78.206
   IP (Target): 195.62.2.11
```

```
Frame 6
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 136 bytes
   Identificator: 31EB
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 41656
   IP (Sender): 195.62.2.11
   IP (Target): 81.181.78.206
```

```
Frame 7
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-02-16-09-FA-40
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 125 bytes
   Identificator: 5E10
   Flags: 0
   Fragment Offset: 0
   TTL: 61
   Protocol of Higher Level: 17
   Header Checksum: 38295
   IP (Sender): 195.62.1.65
   IP (Target): 195.62.2.11
```

```
Frame 8
MAC (Target): 01-80-C2-00-00-00
MAC (Sender): 00-04-4D-8A-B0-D5
Frame Type: Ethernet 802.3/LLC
   LLC: 42-42-3
   Protocol Identifier: 0
   Protocol Version Identifier: 0
   BPDU Type: Configurational
   Rags: 0
   Root Identifier: 80-0-0-4-4D-8A-B0-C0
   Root Path Cost: 0
   Bridge Identifier: 80-0-0-4-4D-8A-B0-C0
   Port Identifier: 32803
   Message Age: 0
   Max Age: 5120
   Hello Time: 512
   Forward Delay: 3840
```

```
Frame 9
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 70 bytes
   Identificator: 4FF5
   Flags: 2
   Fragment Offset: 0
   TTL: 255
   Protocol of Higher Level: 17
   Header Checksum: 54608
   IP (Sender): 195.62.2.11
   IP (Target): 208.11.193.11
```

```
Frame 10
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-02-16-09-FA-40
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 104 bytes
   Identificator: 4FB
   Flags: 2
   Fragment Offset: 0
   TTL: 51
   Protocol of Higher Level: 6
   Header Checksum: 56520
   IP (Sender): 81.181.78.206
   IP (Target): 195.62.2.11
```

```
Frame 11
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-02-16-09-FA-40
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 52 bytes
   Identificator: 4FC
   Flags: 2
   Fragment Offset: 0
   TTL: 51
   Protocol of Higher Level: 6
   Header Checksum: 56571
   IP (Sender): 81.181.78.206
   IP (Target): 195.62.2.11
```

```
Frame 12
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 52 bytes
   Identificator: 31EC
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 41739
   IP (Sender): 195.62.2.11
   IP (Target): 81.181.78.206
```

```
Frame 13
MAC (Target): 00-90-27-A1-36-D0
MAC (Sender): 00-02-16-09-FA-40
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 60 bytes
   Identificator: E514
   Flags: 2
   Fragment Offset: 0
   TTL: 51
   Protocol of Higher Level: 6
   Header Checksum: 64730
   IP (Sender): 81.181.78.206
   IP (Target): 195.62.2.11
```

```
Frame 14
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 64 bytes
   Identificator: 31ED
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 41726
   IP (Sender): 195.62.2.11
   IP (Target): 81.181.78.206
```

```
Frame 15
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 52 bytes
   Identificator: 31EE
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 41737
   IP (Sender): 195.62.2.11
   IP (Target): 81.181.78.206
```

```
Frame 16
MAC (Target): 00-02-16-09-FA-40
MAC (Sender): 00-90-27-A1-36-D0
Frame Type: Ethernet II (0800)
   Version: 4
   Header Lenght: 5 bytes
   TOS: 0
   Datagram Lenght: 139 bytes
   Identificator: 4B16
   Flags: 2
   Fragment Offset: 0
   TTL: 64
   Protocol of Higher Level: 6
   Header Checksum: 61199
   IP (Sender): 195.62.2.11
   IP (Target): 88.247.226.6
```

```
Frame 17
MAC (Target): 01-80-C2-00-00-00
MAC (Sender): 00-04-4D-8A-B0-C3
Frame Type: Ethernet 802.3/LLC
   LLC: 42-42-3
   Protocol Identifier: 0
   Protocol Version Identifier: 0
   BPDU Type: Configurational
   Rags: 0
   Root Identifier: 80-0-0-4-4D-8A-B0-C1
   Root Path Cost: 0
   Bridge Identifier: 80-0-0-4-4D-8A-B0-C1
   Port Identifier: 32783
   Message Age: 0
   Max Age: 5120
   Hello Time: 512
   Forward Delay: 3840
```

```
Frame 18
MAC (Target): 01-80-C2-00-00-00
MAC (Sender): 00-04-4D-8A-B0-C4
Frame Type: Ethernet 802.3/LLC
   LLC: 42-42-3
   Protocol Identifier: 0
   Protocol Version Identifier: 0
   BPDU Type: Configurational
   Rags: 0
   Root Identifier: 80-0-0-4-4D-8A-B0-C1
   Root Path Cost: 0
   Bridge Identifier: 80-0-0-4-4D-8A-B0-C1
   Port Identifier: 32784
   Message Age: 0
   Max Age: 5120
   Hello Time: 512
   Forward Delay: 3840
```

```
Frame 19
MAC (Target): 01-80-C2-00-00-00
MAC (Sender): 00-04-4D-8A-B0-C5
Frame Type: Ethernet 802.3/LLC
   LLC: 42-42-3
   Protocol Identifier: 0
   Protocol Version Identifier: 0
   BPDU Type: Configurational
   Rags: 0
   Root Identifier: 80-0-0-4-4D-8A-B0-C1
   Root Path Cost: 0
   Bridge Identifier: 80-0-0-4-4D-8A-B0-C1
   Port Identifier: 32785
   Message Age: 0
   Max Age: 5120
   Hello Time: 512
   Forward Delay: 3840
```

```
Number of Frames: 19
Number of IPv4 Frames: 15
Number of LLC Frames: 4
Number of ARP Frames: 0
Number of Raw Frames: 0
Number of SNAP Frames: 0
```

Кадр 802.3/LLC

6	6	2	1	1	1(2)	46–1497 (1496)	4
DA	SA	L	DSAP	SSAP	Control	Data	FCS
			Заголовок LLC				

Кадр Raw 802.3/Novell 802.3

6	6	2	46–1500	4
DA	SA	L	Data	FCS

Кадр Ethernet DIX (II)

6	6	2	46–1500	4
DA	SA	T	Data	FCS

Кадр Ethernet SNAP

6	6	2	1	1	1	3	2	46–1492	4
DA	SA	L	DSAP	SSAP	Control	OUI	T	Data	FCS
			AA	AA	03	000000			
			Заголовок LLC			Заголовок SNAP			

В потоке кадров были обнаружены лишь фреймы типов Ethernet 802.3/LLC и Ethernet DIX (II).

Теперь проведем анализ кадра с номером 8, информация о котором представлена справа.

Для получения hex-представления файла был использован сервис **hexed.it**.

```
MAC (Target): 01-80-C2-00-00-00
MAC (Sender): 00-04-4D-8A-B0-D5
Frame Type: Ethernet 802.3/LLC
   LLC: 42-42-3
   Protocol Identifier: 0
   Protocol Version Identifier: 0
   BPDU Type: Configurational
   Rags: 0
   Root Identifier: 80-0-0-4-4D-8A-B0-C0
   Root Path Cost: 0
   Bridge Identifier: 80-0-0-4-4D-8A-B0-C0
   Port Identifier: 32803
   Message Age: 0
   Max Age: 5120
   Hello Time: 512
   Forward Delay: 3840
```

Перейдем к участку файла с кадром 8, в первых 6-ти байтах заголовка которого видим MAC-адрес получателя (**Target**):

```
00000AD0  01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0  8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0  04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00  C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Заметим, что первый бит старшего байта указывает на тип адреса multicast. Это значит, что данный адрес предназначается для группы узлов. Второй бит определяет способ назначения адреса - в нашем случае он назначен централизованно (комитетом IEEE).

За ним следует MAC-адрес отправителя (**Sender**):

```
00000AD0  01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0  8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0  04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00  C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Заметим, что первый бит адреса отправителя всегда равен 0.

Следующие 2 байта - длина кадра, что в переводе из шестнадцатеричной системы счисления равно **38**:

```
00000AD0  01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0  8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0  04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00  C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Далее находится **LLC-заголовок**, в котором первые 2 байта указывают на то, что кадр имеет тип Ethernet 802.3/LLC, причем для передачи этого кадра используется протокол **STP**:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Цель полей **SSAP** и **DSAP** - идентифицировать вышестоящий протокол.

И, как правило, протоколы STP используют пакеты типа **BPDU**, первые 3 байта (идентификатор протокола и его версия) которых всегда равны 0:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Четвертый байт со значением 0 здесь указывает на тип BPDU - **конфигурационный**.

Следующий байт содержит **флаги**, говорящие об изменении конфигурации:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Далее 8 байт отведено под ID корневого коммутатора (**Root Identifier**) - некого аналога IP-адреса в IP-пакете:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

После чего следуют 2 байта для значения **расстояния** от корневого коммутатора до текущего узла (моста):

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Далее находится ID текущего узла (**Bridge Identifier**):

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Поскольку он совпадает с ID корневого коммутатора, становится понятно, почему значение расстояния нулевое.

Далее имеем идентификатор порта (**Port Identifier**):

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Время жизни сообщения (**TTL / Message Age**), равное нулю, поскольку ID моста и корневого коммутатора совпадают:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Максимальная продолжительность жизни (**Max Age**):

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Когда пакет BPDU имеет TTL больше максимального, коммутаторы начинают его игнорировать.

Следующие 2 байта содержат значение времени приветствия (**Hello Time**) - времени, через которое посылаются пакеты BPDU корневым коммутатором:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```

Наконец, последние 2 байта пакета отведены под задержку смены состояний (**Forward Delay**) - значения минимального времени перехода портов коммутатора в активное состояние:

```
00000AD0   01 51 80 00 00 02 58 01 80 C2 00 00 00 00 04 4D
00000AE0   8A B0 D5 00 26 42 42 03 00 00 00 00 00 80 00 00
00000AF0   04 4D 8A B0 C0 00 00 00 00 80 00 00 04 4D 8A B0
00000B00   C0 80 23 00 00 14 00 02 00 0F 00 00 02 16 09 FA
```