广州大学学生实验报告

开课学院及实验室：计算机科学与工程实验室 518　　2019 年 12 月 5 日

学院	计算机科学与教育软件学院	年级/专业/班	软件 171	姓名	谢金宏	学号	1706300001
实验课程名称	计算机网络实验					成绩	
实验项目名称	网络程序设计					指导老师	唐琳

一、实验目的

1 、编写一个程序模拟网桥功能。
2 、编写一个程序计算文件的 16 位 RFC1071 校验和。

二、实验环境

安装有 C++编程软件的计算机。

三、实验内容

1、 编写一个程序模拟网桥的功能。

模拟实现网桥的转发功能，以从文件中读取帧模拟网桥从网络中收到一帧，即从两个文件中读入一系列帧，从第一个文件中读入一帧然后从第二个文件中再读入一帧，如此下去。对每一帧，显示网桥是否会转发，及显示转发表内容。

要求：Windows 或 Linux 环境下运行，程序应在单机上运行。

分析：用程序模拟网桥功能，可以假定用两个文件分别代表两个网段上的网络帧数据。而两个文件中的数据应具有帧的特征，即有目的地址，源地址和帧内数据程序交替读入帧的数据，就相当于网桥从网段中得到帧数据。对于网桥来说，能否转发帧在于把接收到的帧与网桥中的转发表相比较。判断目的地址后才决定是否转发。由此可见转发的关键在于构造转发表。这里转发表可通过动态生成。

2、 编写一个程序计算文件的 16 位 RFC1071 校验和。

要求：以命令行的方式运行程序：checksum filename，输出文件的校验和。

四、实验步骤、记录和结果

1、 模拟网桥功能。

首先定义数据结构。由于网桥工作在数据链路层，网桥转发的都是数据链路层帧，因此需要定义链路层帧数据结构。链路层帧包含目的地址、源地址和数据。由于程

序的重点在模拟网桥动态生成转发表的算法，因此这里忽略帧的数据部分。定义数据结构如下（下方的代码片段省略了对输入和输出流的处理）：

```
// 模拟MAC地址。
class MacAddress {
public:
    string address; // 6字节MAC地址，使用长度为12的十六进制数字表示。

    // 随机产生6字节的MAC地址。
    MacAddress static getRandomAddress() {
        stringstream ss;
        for (int i = 0; i < 6; ++i) {
            ss << hex << setw(2) << setfill('0') << int(uint8_t(rand()));
        }
        return MacAddress(ss.str());
    }
};

// 模拟帧
// 这里省略了帧的数据部分，因为程序意在模拟网桥。
class Frame {
public:
    MacAddress dstMac; // 目的MAC地址
    MacAddress srcMac; // 源MAC地址

    // 初始化一个空的帧。
    Frame() {
        this->dstMac = MacAddress();
        this->srcMac = MacAddress();
    }

    // 初始化一个带有源主机地址和目的主机地址的帧。
    Frame(MacAddress dstMac, MacAddress srcMac) {
        this->dstMac = dstMac;
        this->srcMac = srcMac;
    }

    // 判断帧是否为空。
    bool isEmpty() {
        Frame emptyFrame;
        return *this == emptyFrame;
    }
};
```

本次模拟的网桥设有两个端口 A、B。当网桥从某个端口上收到链路层帧时，网桥从帧中读取源主机的 MAC 地址，并将 MAC 地址与收到该帧的端口号成对记录到转发表中。接下来，网桥从帧中读取目的主机的 MAC 地址并查找转发表。若转发表中没有目的主机的 MAC 地址的条目，则网桥在所有接口上转发此帧。若转发表中存在目的主机的 MAC 地址的条目，则判断目的主机所在的接口与网桥收到此帧的接口是否为同一个接口。若为同一个接口，则不进行转发；若接口不同，则将此帧转发到目的主机所在的接口。

转发表自学习的关键代码如下：

```
// 从端口中读取数据。
if (in) {
    // 读取一帧并跳过空帧。
    Frame frame; in >> frame;
    if (frame.isEmpty()) {
        continue;
    }

    cout << "从" << (port == Port::A ? "A" : "B") << "端口收到来自"
        << frame.srcMac << "发往" << frame.dstMac << "的帧。"
        << endl;

    // 自学习源主机的端口。
    transferTable[frame.srcMac] = port;

    // 检查 Mac 地址。
    if (transferTable.find(frame.dstMac) != transferTable.end()) {
        if (transferTable[frame.dstMac] == transferTable[frame.srcMac]) {
            // 若源主机与目的主机所在端口相同，则丢弃该帧不进行转发。
            drop(frame);
        } else {
            // 若源主机与目的主机所在端口不同，则向目的主机所在端口转发该帧。
            transfer(frame, transferTable[frame.dstMac]);
        }
    } else {
        // 若转发表中没有目的主机的信息，则广播该帧。
        broadcast(frame);
    }

    // 打印修改后的转发表。
    showTransferTable();
}
```

运行程序，结果如下：

模拟网桥有两个端口 A、B。

端口 A 连接了主机 1、主机 2；端口 2 连接了主机 3、主机 4。

主机的 MAC 地址随机生成。

发送的数据保存在 bridge-test1.bin 和 bridge-test2.bin 中，也是随机生成的。

模拟网桥开始运行。

从 B 端口收到来自 db3c870c3e99 发往 2923be84e16c 的帧。

在所有端口上广播此帧。

转发表:

```
+-----------------------+
|     Mac Address|  Port|
+-----------------------+
|    db3c870c3e99|     B|
+-----------------------+
```

从 A 端口收到来自 2923be84e16c 发往 db3c870c3e99 的帧。

在 B 上发送此帧。

转发表:

```
+-----------------------+
|     Mac Address|  Port|
+-----------------------+
|    2923be84e16c|     A|
|    db3c870c3e99|     B|
+-----------------------+
```

从 B 端口收到来自 db3c870c3e99 发往 f1bbe9ebb3a6 的帧。

在所有端口上广播此帧。

……（省略中间的输出，完整输出参见 bridge-output.txt。）

从 A 端口收到来自 d6ae529049f1 发往 2923be84e16c 的帧。

不进行转发，因为源主机与目的主机在网桥同一端口上。

转发表:

```
+-----------------------+
|     Mac Address|  Port|
+-----------------------+
|    2923be84e16c|     A|
|    d6ae529049f1|     A|
|    db3c870c3e99|     B|
|    f1bbe9ebb3a6|     B|
+-----------------------+
```

模拟网桥运行完毕。

2、 计算校验和。

校验和计算的原理是：把要发送的数据看成 16 比特的二进制整数序列，并计算他们的和。若数据字节长度为奇数，则在数据尾部补一个字节的 0 以凑成偶数。任意一步计算过程中，如果效验和大于 16 位，那么把进位一起加到效验和中。

完整代码如下：

```cpp
int main(int argc, const char* argv[]) {
    // 检查参数。
    if (argc < 2) {
        cout << "参数不足，需要给出一个文件名。" << endl;
        return 1;
    } else if (argc > 2) {
        cout << "给出参数过多，只需要给出一个文件名即可。" << endl;
        return 1;
    }

    string filename(argv[1]); // 取得文件名。
    ifstream fin(filename, ios::binary); // 以二进制模式打开文件。

    // 检查文件是否可以打开。
    if (!fin.is_open()) {
        cout << "文件无法打开，请检查文件名是否正确。" << endl;
        return 1;
    }

    uint32_t checksum = 0; // 待计算的校验和。

    // 从文件中读取数据，一次读入两个字节。
    uint8_t buffer[2]; // 若使用 char 类型，必须使用 u_char，否则类型转换时按补码规则补 1 导致计算。
    for (int count = 0; count = fin.readsome((char *)buffer, 2); ) {
        uint16_t in = 0;

        // 若数据长度为奇数，则在数据末尾补充一个字节的 0 以凑成偶数。
        if (count == 1) {
            in = buffer[0];
            in <<= 8;
        } else {
            in = (uint16_t(buffer[0]) << 8) + buffer[1];
        }

        // 计算校验和。注意处理进位。
        checksum += in;
        while (checksum >= (1<<16)) {
            uint32_t carry = checksum >> 16;
            checksum = checksum % (1<<16) + carry;
        }
    }

    // 输出 16 位校验和
    cout << hex << setw(4) << setfill('0') << checksum << endl;
}
```

下面使用两组输入来测试程序：

第一组输入和程序输出

checksum-test1.bin

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  对应文本
00000000   48 65 6C 6C 6F 20 77 6F 72 6C 64 2E              Hello world.
```

Figure 1 第一组程序输入

```
PS C:\Users\light\OneDrive - e.gzhu.edu.cn\.stack\计算机网络实验\实验5> ./checksum .\checksum-test1.bin
71fc
```

Figure 2 第一组程序输出

第二组输入和程序输出

checksum-test2.bin

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  对应文本
00000000   00 01 F2 03 F4 F5 F6 F7                          .ò.ôõö÷
```

Figure 3 第二组输入

```
PS C:\Users\light\OneDrive - e.gzhu.edu.cn\.stack\计算机网络实验\实验5> ./checksum .\checksum-test2.bin
ddf2
```

Figure 4 第二组输出

程序的输出与理论计算是相符的。

请查阅附件获取实验中所使用的完整代码：

实验5源代码和输入数据以及输出.zip

五、实验分析

1、 模拟网桥实验中，最关键的部分是转发表的自学习。为简单起见，本次实验中没有模拟网络拓扑发生变化时的情况。本次模拟的网桥设有两个端口 A、B。当网桥从某个端口上收到链路层帧时，网桥从帧中读取源主机的 MAC 地址，并将 MAC 地址与收到该帧的端口号成对记录到转发表中。接下来，网桥从帧中读取目的主机的 MAC 地址并查找转发表。若转发表中没有目的主机的 MAC 地址的条目，则网桥在所有接口上转发此帧。若转发表中存在目的主机的 MAC 地址的条目，则判断目的主机所在的接口与网桥收到此帧的接口是否为同一个接口。若为同一个接口，则不进行转发；若接口不同，则将此帧转发到目的主机所在的接口。

2、 计算校验和实验中，最关键的部分是文件的读取以及进位的处理。使用 C++编程时，文件应该用二进制模式打开，每次读取两个字节。任意一步计算过程中，若效验和大于 16 位，那么要把进位一起加到效验和中。

六、实验建议

无。