**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电子楼518室 2018年12月 17 日

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 计算机科学与教育软件 | | 年级、专业、班 | 软件162 | 姓名 | 吴广城 | 学号 | 1606100138 |
| 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 | 綦科 |

**（1）实验目的**

通过程序网络数据包的监听与分析

的工作原理以及检验和的计算。

**（2）实验内容**

**2、编写一个计算机程序用来计算一个文件的16位效验和。最快速的方法是用一个32位的整数来存放这个和。记住要处理进位（例如，超过16位的那些位），把它们加到效验和中。**

**要求**：1）以命令行形式运行：check\_sum infile

其中check\_sum为程序名，infile为输入数据文件名。

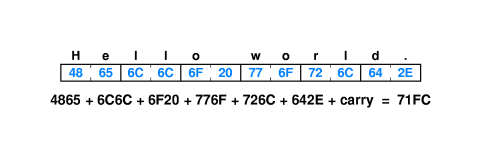
2）输出：数据文件的效验和

附：效验和（checksum)

**参见RFC1071 - Computing the Internet checksum**

* 原理：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。
* 例子：16位效验和计算，下图表明一个小的字符串的16位效验和的计算。

为了计算效验和，发送计算机把每对字符当成16位整数处理并计算效验和。如果效验和大于16位，那么把进位一起加到最后的效验和中。



**3**、**网络数据包的监听与分析**

要求使用Wincap编制一个简单的网络数据包监听与捕获程序，同时，将捕获的数据包进行分析并将分析结果显示在屏幕上。

1. **程序设计思路及实现**

**2.计算校验和**

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

int main()

{

int data = 0, c=0,H = 0, L = 0;//data=H+L.L是低位，H是高位，若数据字节长度为奇数，则data=H

int sum = 0;

int carry = 0;

fstream fin("infile.txt", ios::in);//读取文件

if (!fin)

{

cout << "读取不到该文件" << endl;

exit(1);

}

while (true)

{

c = fin.get();

if (c == EOF)

break;

H = c << 8;

c = fin.get();

L = c;

if (c == EOF)

{

data = H;

cout <<hex<< data <<" ";

sum += data;

}

data = H + L;

cout <<hex <<data << " ";

sum += data;

}

carry = sum >> 16;

sum = sum - carry \* (0X10000);

cout <<'\n'<< "校验和为：" << sum+carry << endl;

fin.close();

system("pause");

}

**实现：**

**首先目录应该存有**infile.txt**该文件，然后通过每次读取该文件中的一个字符，第一次读取存入一个int类型高8位，第二次读取则存入低8位，若不存在低8位补0，通过把每次读取组成的16位相加，把溢出的部分算出，再加入16位的校验和当中，得出最后的校验和。**

**3.网络数据包的监听与分析**

#pragma warning(disable : 4996)

#pragma comment(lib,"ws2\_32.lib")

#define WIN32

#include "pcap.h"

#include"stdio.h"

#include <iostream>

using namespace std;

struct ether\_header

{

u\_int8\_t ether\_dhost[6];

u\_int8\_t ether\_shost[6];

u\_int16\_t ether\_type;

};

/\* packet handler 函数原型\*/

void packet\_analysis(u\_char \*param, const struct pcap\_pkthdr \*header, const u\_char \*pkt\_data);

int main()

{

pcap\_if\_t \*alldevs;

pcap\_if\_t \*d;

int inum;

int i = 0;

pcap\_t \*adhandle;

char errbuf[PCAP\_ERRBUF\_SIZE];

/\* 获取本机设备列表\*/

if (pcap\_findalldevs\_ex(PCAP\_SRC\_IF\_STRING, NULL, &alldevs, errbuf) == -1)

{

fprintf(stderr, "pcap\_findalldevs错误: %sn", errbuf);

exit(1);

}

/\* 打印列表\*/

for (d = alldevs; d; d = d->next)

{

printf("设备号：%d.名称：%s\n", ++i, d->name);

if (d->description)

printf("设备描述:%s)\n", d->description);

else

printf("无可用描述！！！\n");

}

if (i == 0)

{

printf("无法找到设备，请确保winpcap正确安装！！！");

return -1;

}

printf("请输入接口编号(1-%d):", i);

scanf\_s("%d", &inum);

if (inum < 1 || inum > i)

{

printf("输入错误！！超出范围！！");

/\* 释放设备列表\*/

pcap\_freealldevs(alldevs);

return -1;

}

/\* 跳转到选中的适配器\*/

for (d = alldevs, i = 0; i< inum - 1; d = d->next, i++);

/\* 打开设备\*/

if ((adhandle = pcap\_open(d->name, // 设备名

65536, // 65535保证能捕获到不同数据链路层上的每个数据包的全部内容

PCAP\_OPENFLAG\_PROMISCUOUS, // 混杂模式

1000, // 读取超时时间

NULL, // 远程机器验证

errbuf // 错误缓冲池

)) == NULL)

{

fprintf(stderr, "\nnUnable to open the adapter. %s is not supported by WinPcapn\n", d->name);

/\* 释放设备列表\*/

pcap\_freealldevs(alldevs);

return -1;

}

printf("\n正在监听%s...n\n", d->description);

/\* 释放设备列表\*/

pcap\_freealldevs(alldevs);

/\* 开始捕获\*/

pcap\_loop(adhandle, 0, packet\_analysis, NULL);

return 0;

}

/\* 每次捕获到数据包时，libpcap都会自动调用这个回调函数\*/

void packet\_analysis(u\_char \*argument\_wentaoliu, const struct pcap\_pkthdr \*packet\_header, const u\_char \*packet\_content)

{

struct tm \*ltime;

char time[16];

time\_t local\_tv\_sec;

/\* 将时间戳转换成可识别的格式\*/

local\_tv\_sec = packet\_header->ts.tv\_sec;

ltime = localtime(&local\_tv\_sec);

strftime(time, sizeof time, "%H:%M:%S", ltime);

u\_short type;

struct ether\_header \*ethernet\_protocol;

u\_char \*mac\_string;

static int Num = 1;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("捕获第%d个数据包\n", Num);

printf("数据包长度:");

printf("%d\n", packet\_header->len);

printf("捕获时间：");

printf("%s\n", time);

printf("上层协议的协议类型是：");

ethernet\_protocol = (struct ether\_header \*)packet\_content;

type = ntohs(ethernet\_protocol->ether\_type);

{if (type == 0x0800)

printf("IP协议\n");

else

if (type == 0x0806)

printf("ARP协议\n");

else

if (type == 0x0835)

printf("RARP协议\n");

else

printf("未知协议\n");

}

printf("源以太网地址：");

mac\_string = ethernet\_protocol->ether\_shost;

printf("%02x:%02x:%02x:%02x:%02x:%02x\n", mac\_string, (mac\_string + 1), (mac\_string + 2), (mac\_string + 3), (mac\_string + 4), (mac\_string + 5));

//获得源以太网地址

printf("目的以太网地址:");

mac\_string = ethernet\_protocol->ether\_dhost;

printf("%02x:%02x:%02x:%02x:%02x:%02x\n", mac\_string, (mac\_string + 1), (mac\_string + 2), (mac\_string + 3), (mac\_string + 4), (mac\_string + 5));

//获得目的以太网地址

Num++;

}

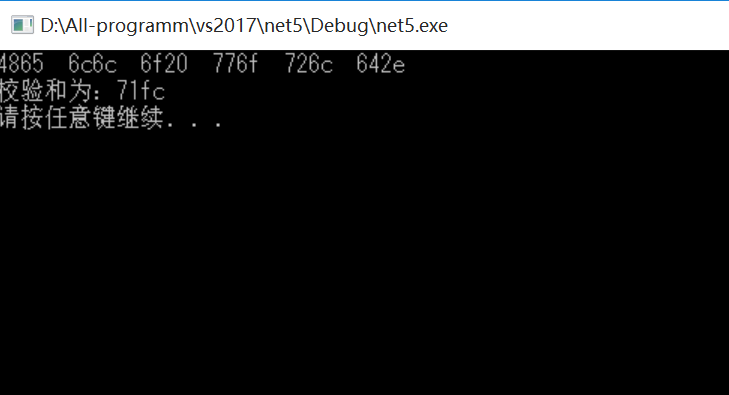
**实现:**

**使用pcap\_findalldevs\_ex函数获取已连接的网络适配器列表。这个函数返回一个 pcap\_if 结构的链表， 每个这样的结构都包含了一个适配器的详细信息。其中数据域name表示适配器名称，但难以理解，如rpcap://\Device\NPF\_{7CD080C7-2B67-4839-B6F6-1D90273D960A}，另一个数据域description表示一个可以让人们理解的描述，如(Network adapter 'Realtek PCIe FE Family Controller' on local host)。**

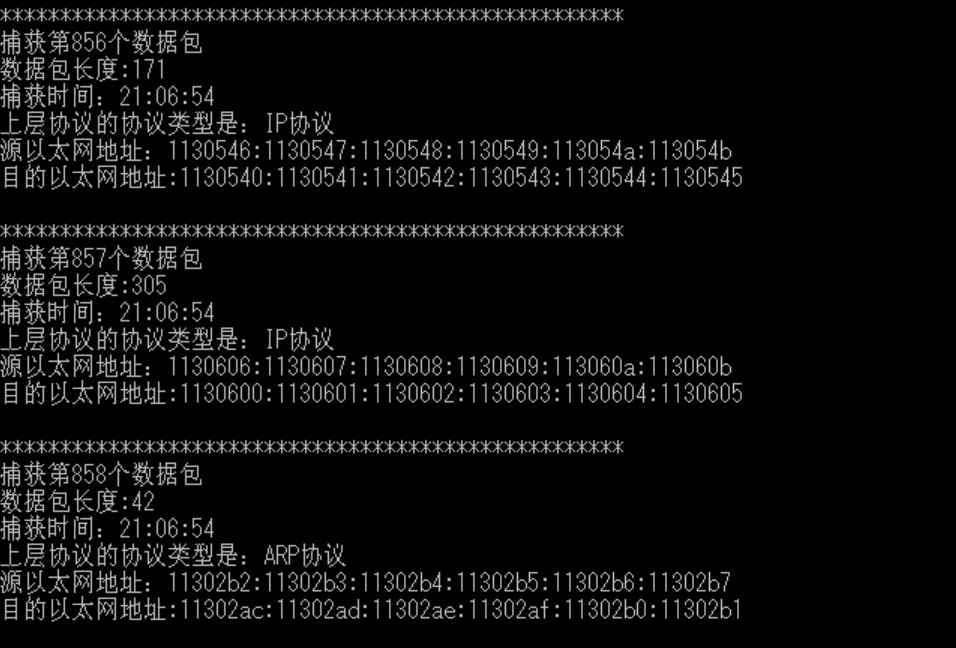
**使用pcap\_open()函数打开适配器，将这个函数的snaplen参数设为65535，这比能遇到的最大的MTU还要大，以保证能收到完整的数据包。另一个参数flag设为PCAP\_OPENFLAG\_PROMISCUOUS，意为混杂模式，不管这个数据包是不是发给本机的，都会去捕获。参数to\_ms 指定读取数据的超时时间，这里设置为1000ms，在适配器上进行读取操作都会在 to\_ms 毫秒时间内响应。使用pcap\_loop函数开始捕捉数据包**。

1. **实验结果记录**

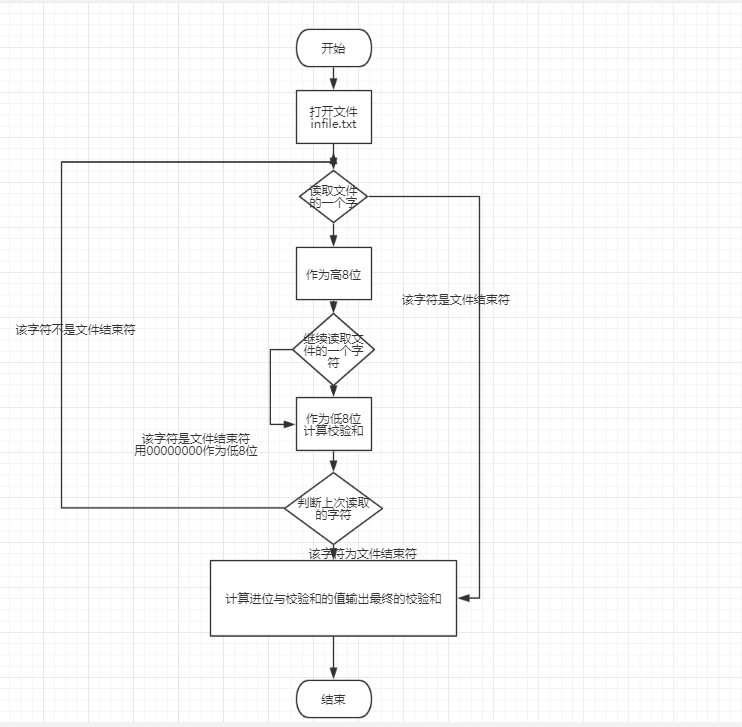
**2.计算校验和**



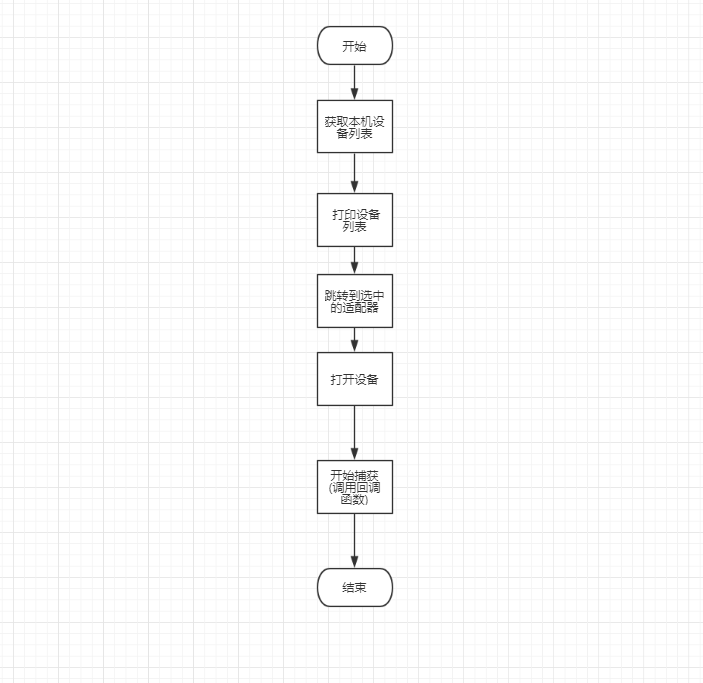
**3.网络数据包的监听与分析**



1. **流程图**
2. **计算校验和**



1. **网络数据包的监听与分析**



**（6）开发中遇到的问题及解决办法**

实验中遇到的主要问题有两个，一是在vs环境下对winpcap的环境配置。二是对winpcap相关库函数的使用规则。主要通过网上查找配置方法，在官网文档查看相关函数的使用，如pcap\_findalldevs\_ex()函数,pcap\_open()函数等。