

**计算机网络原理**

**课程设计文档**

**题目：**协议分析软件开发及网络协议分析

**学 院 计算机科学学院**

**专 业 软件工程**

**班 级 软件工程1701班**

**小组成员 桑展 41709050127**

**赵侍书 41703070125**

**任豆豆 41702130120**

**强倩雯 41709050126**

**指导教师 薛 文 生**

**2019年 12 月 27日**

目录

[一、任务概述 1](#_Toc1630)

[1.1 设计目的 1](#_Toc1603)

[1.2 项目任务和要求 1](#_Toc22978)

[1.3 参考资料 1](#_Toc26989)

[二、开发环境 1](#_Toc20706)

[三、项目需求分析 2](#_Toc30800)

[四、项目设计和实现 2](#_Toc17682)

[4.1 总体设计 2](#_Toc16475)

[4.2 功能设计 3](#_Toc27076)

4.2.1 通信协议的定义------------------------------------------------------------------3

4.2.1.1 IP协议的定义----------------------------------------------------------------3

4.2.1.2 ICMP协议的定义------------------------------------------------------------5

4.2.1.3 TCP协议的定义--------------------------------------------------------------7

4.2.1.4 UDP协议的定义-------------------------------------------------------------9

4.2.2 功能设计-------------------------------------------------------------------------10

[4.3 系统实现 10](#_Toc7154)

[五、程序的运行和测试结果 12](#_Toc14748)

[六、 设计中的问题 12](#_Toc17840)

[1.抓数据包问题- 14](#_Toc11564)

[2.客户端问题- 14](#_Toc25274)

[3.页面报500错误 14](#_Toc4219)

[七、附录 14](#_Toc9385)

[7.1 程序清单 15](#_Toc13756)

[7.2 其他需要说明的内容 28](#_Toc26586)

**摘要：**网络数据包捕获是进行网络分析的基础，通过对Linux操作系统平台下网络数据包捕获模型的论述，实现了如何在Mac分析客户端下进行编程进而对网络数据包捕获和分析的方法，并通过serverlet连接，返回给页面一个存储有每个协议分析后结果的list，遍历list中的数据包，最终以网页界面的形式呈现。

关键字：数据包捕获 数据包 协议分析 界面

**Abstract：**Network packet capture is the basis of network analysis. Through the discussion of the network packet capture model under the Linux operating system platform, the method of how to program and then capture and analyze the network packet under the Mac analysis client is realized, and through the serverlet connection, it returns to the page a list that stores the result of each protocol analysis.

**Key words：**packet capture，data packet，protocol analysis，interface

# 一、任务概述

## 1.1 设计目的

使学生深入理解和掌握计算机网络的基本理论及工作原理，熟悉计算机网络和互联网的组成，运用计算机网络知识设计协议分析软件，并进行网络协议分析，最后通过网页呈现，加深对计算机网络系统的认知、设计与应用开发能力，同时培养学生运用计算机网络技术解决实际问题的能力。

## 1.2 项目任务和要求

要求学生掌握计算机网络协议原理、协议分析方法及协议分析软件的开发。理解网络协议的层次结构及网络协议的工作过程，掌握网络协议的分析方法及网络协议的设计与应用。

## 1.3 参考资料

·《计算机网络》，谢希仁，电子工业出版社，2017年1月第7版。

· 通过因特网查阅资料。

# 二、开发环境

运行的操作系统：Linux抓包服务器；Mac分析客户端

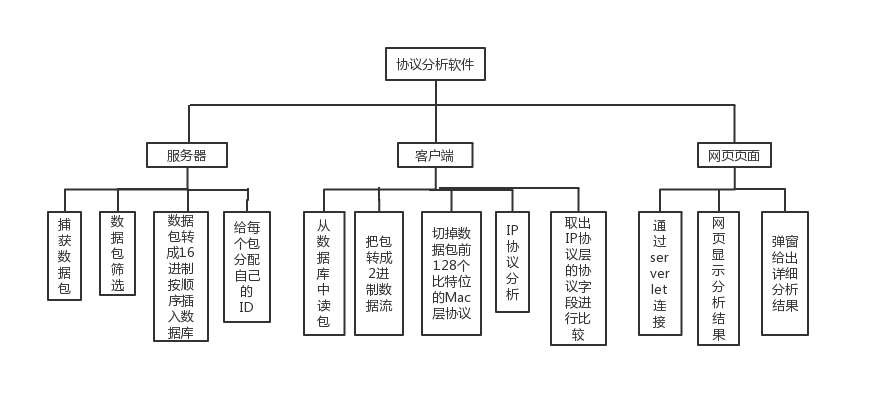
编程工具:Linux下的vim、idea

集成开发环境：idea；Linux下的vim

Linux:服务器IP：192.168.43.173

Mac客户端IP：192.168.43.172

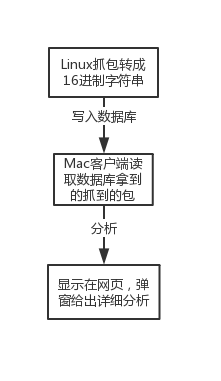
# 三、项目需求分析



界面管理模块的主要功能是方便用户对网络协议分析的管理，网络数据包捕获模块是系统的开始部分，它的重要性在于数据包捕获的准确性和及时性，不可丢包，否则将影响整个系统的性能。协议解析模块主要是完成对捕获的数据包进行详细的分析，在此基础上由协议分析模块进一。步分析网络协议。存储模块的主要功能是存储捕获来的、并进行协议分析后的网络信息，以便事后分析网络性能。

# 四、项目设计和实现

## 4.1 总体设计

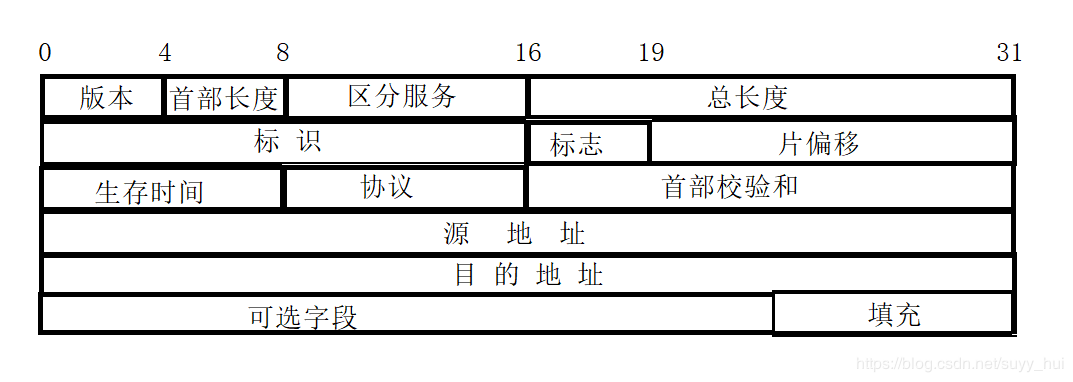
关键类：libcap函数库；serverlet；jsp的页面设计

## 4.2 功能设计

4.2.1 通信协议的定义：

**4.2.1.1** IP协议：

IP地址表现由一组32位2进制数组成，每8位为一个段，共分为4段，以‘.’间隔。含义由网络地址(NetID)和主机地址(HostID)两部分组成。网络地址表示其属于互联网中的哪一个网络，而主机地址则表示其属于该网络中的哪一台主机，两者之间是主从关系。IP地址对应应用于网络层，基于IP协议的网络地址就是IP地址。



·IP协议分析：

package agreement;  
  
import java.io.UnsupportedEncodingException;  
import java.util.Arrays;  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
public class IP{  
public static String versions;  
public static int headLength;  
public static int totalLength;  
public static int identificationNumber;  
public static int signnumber;  
public static int SliceOffsetNUmber;  
public static String agreement;  
public static int headCheckSum;  
public static String destinationaddress;  
public static String sourceaddress;  
  
public static void ip(byte[]bytes)throws UnsupportedEncodingException{  
  
byte[]versionsarr=Arrays.copyOfRange(bytes,0,4);  
int versionsnum=binaryToDecimalish(versionsarr);  
if(versionsnum==4){  
versions="IPV4";  
}  
if(versionsnum==6){  
versions="IPV6";  
}  
//System.out.println("版本："+versions);  
  
byte[]head=Arrays.copyOfRange(bytes,4,8);  
headLength=binaryToDecimalish(head);  
//System.out.println("首部长度："+headLength);  
  
byte[]Length=Arrays.copyOfRange(bytes,16,32);  
totalLength=binaryToDecimalish(Length);  
//System.out.println("总长度："+totalLength);  
  
byte[]identification=Arrays.copyOfRange(bytes,32,48);  
identificationNumber=binaryToDecimalish(identification);  
//System.out.println("标识："+identificationNumber);  
  
byte[]sign=Arrays.copyOfRange(bytes,48,51);  
signnumber=binaryToDecimalish(sign);  
//System.out.println("标志："+signnumber);  
  
byte[]SliceOffset=Arrays.copyOfRange(bytes,51,64);  
SliceOffsetNUmber=binaryToDecimalish(SliceOffset);  
//System.out.println("片偏移："+SliceOffsetNUmber);  
  
byte[]Agreement=Arrays.copyOfRange(bytes,72,80);  
int agreementnum=binaryToDecimalish(Agreement);  
if(agreementnum==1){  
agreement="ICMP";  
}else if(agreementnum==6){  
agreement="TCP";  
}else if(agreementnum==17){  
agreement="UDP";  
}else{  
agreement="IP";  
}  
  
byte[]headChecknum=Arrays.copyOfRange(bytes,80,96);  
headCheckSum=binaryToDecimalish(headChecknum);  
//System.out.println("首部校验和："+headCheckSum);  
  
int[]sourceAddress=new int[4];  
sourceAddress[0]=binaryToDecimalish(Arrays.copyOfRange(bytes,96,104));  
sourceAddress[1]=binaryToDecimalish(Arrays.copyOfRange(bytes,104,112));  
sourceAddress[2]=binaryToDecimalish(Arrays.copyOfRange(bytes,112,120));  
sourceAddress[3]=binaryToDecimalish(Arrays.copyOfRange(bytes,120,128));  
StringBuffer sb=new StringBuffer();  
for(int i=0;i<4;i++){  
if(i==3){  
sb.append(sourceAddress[i]);  
break;  
}  
sb.append(sourceAddress[i]);  
sb.append(":");  
}  
sourceaddress=sb.toString();  
  
int[]destinationAddress=new int[4];  
destinationAddress[0]=binaryToDecimalish(Arrays.copyOfRange(bytes,128,136));  
destinationAddress[1]=binaryToDecimalish(Arrays.copyOfRange(bytes,136,144));  
destinationAddress[2]=binaryToDecimalish(Arrays.copyOfRange(bytes,144,152));  
destinationAddress[3]=binaryToDecimalish(Arrays.copyOfRange(bytes,152,160));  
StringBuffer sb1=new StringBuffer();  
for(int i=0;i<4;i++){  
if(i==3){  
sb1.append(destinationAddress[i]);  
break;  
}  
sb1.append(destinationAddress[i]);  
sb1.append(":");  
}  
destinationaddress=sb1.toString();  
  
}  
}

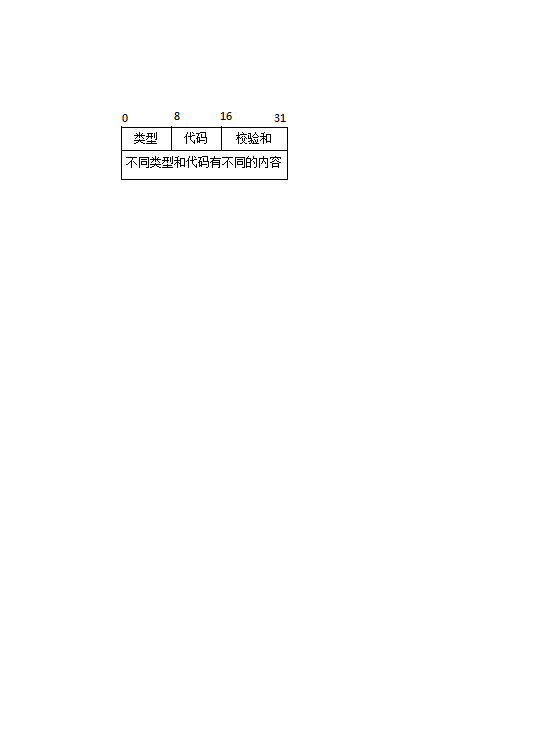
**4.2.1.2** ICMP协议：

是一个网络层协议。 一个新搭建好的网络，往往需要先进行一个简单的测试，来验证网络是否畅通；但是IP协议并不提供可靠传输。如果丢包了，IP协议并不能通知传输层是否丢包以及丢包的原因。 所以我们就需要一种协议来完成这样的功能——ICMP协议。

ICMP的报文格式：

ICMP报文包含在IP数据报中，IP报头在ICMP报文的最前面。一个ICMP报文包括IP报头（至少20字节）、ICMP报头（至少八字节）和ICMP报文（属于ICMP报文的数据部分）。当IP报头中的协议字段值为1时，就说明这是一个ICMP报文。

ICMP报头如下图所示：



字段说明：

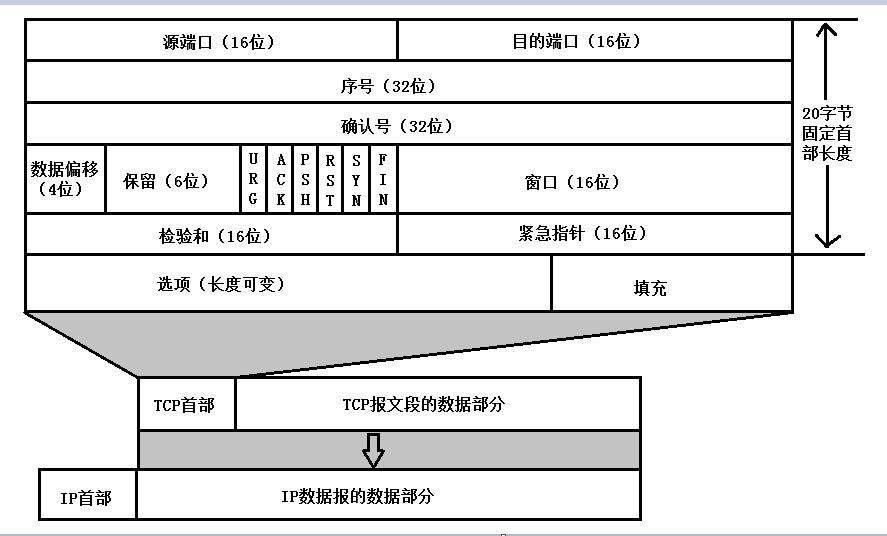
|  |  |
| --- | --- |
| 类型 | 说明 |
| 类型 | 占一字节，标识ICMP报文的类型，从类型值来看ICMP报文可以分为两大类。第一类是取值为1~127的差错报文，第2类是取值128以上的信息报文 |
| 代码 | 占一字节，标识对应ICMP报文的代码。它与类型字段一起共同标识了ICMP报文的详细类型 |
| 校验和 | 这是对包括ICMP报文数据部分在内的整个ICMP数据报的校验和，以检验报文在传输过程中是否出现了差错（其计算方法与在我们介绍IP报头中的校验和计算方法是一样的） |

·ICMP协议分析：

package agreement;  
  
import java.io.UnsupportedEncodingException;  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class ICMP {  
    public static String Type;  
    public static String code;  
    public static int Checksum;  
    public static void icmp(byte[]icmpArr) throws UnsupportedEncodingException {  
        byte[]type= Arrays.copyOfRange(icmpArr,0,8);  
        int Typenum=binaryToDecimalish(type);  
        if(Typenum==3){  
           Type="终点不可达";  
        }  
        if(Typenum==11){  
            Type="时间超过";  
        }  
        if(Typenum==12){  
           Type="参数问题";  
        }  
        if(Typenum==5){  
            Type="改变路由";  
        }  
        if(Typenum==8||Typenum==0){  
            Type="回送请求或回答";  
        }  
        if(Typenum==13||Typenum==14){  
            Type="时间戳请求或回答";  
        }  
  
        code=new String(icmpArr,8,16-8,"UTF-8");  
        //System.out.println(code);  
  
        byte[]checksum=Arrays.copyOfRange(icmpArr,16,32);  
        Checksum=binaryToDecimalish(checksum);  
        //System.out.println("校验和："+Checksum);  
  
    }  
}

**4.2.1.3** TCP协议**：**

TCP协议是TCP/IP协议体系中非常复杂的一个协议。它是面向连接的运输层协议，每一条TCP连接只能有两个端点(endpoint)，每一条TCP连接只能是点对点的(一对一)。TCP提供可靠交付的服务，通过TCP连接传送的数据，无差错、不丢失、不重复，并且按序到达。TCP提供全双工通信，面向字节流，TCP中的“流”(stream)指的是流入到进程或从进程流出的字节序列。TCP连接是一条虚连接(也就是逻辑链接)，而不是一条真正的物理连接。TCP并不关心应用进程一次把多长的报文发送到TCP的缓存中，而是根据对方给出的窗口值和当前网络拥塞的程度来决定一个报文段应包含多少个字节。

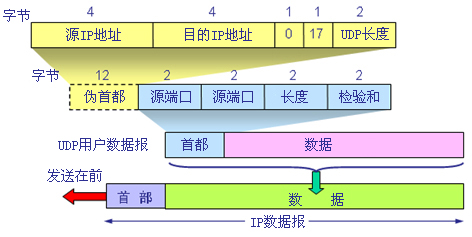
****

·TCP协议分析：

package agreement;  
  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class TCP {  
    public static int sourcePort;  
    public static int destinationPort;  
    public static int serialNumber;  
    public static int confirmationNumber;  
    public static int windowSize;  
    public static void tcp(byte[]bytes) {  
        byte[]source= Arrays.copyOfRange(bytes,0,16);  
        sourcePort=binaryToDecimalish(source);  
        //System.out.println("源端口："+sourcePort);  
  
        byte[]destination=Arrays.copyOfRange(bytes,16,32);  
        destinationPort=binaryToDecimalish(destination);  
        //System.out.println("目的端口："+destinationPort);  
  
        byte[] serial=Arrays.copyOfRange(bytes,32,64);  
        serialNumber=binaryToDecimalish(serial);  
        //System.out.println("序号："+serialNumber);  
  
        byte[]confirmation=Arrays.copyOfRange(bytes,64,96);  
        confirmationNumber=binaryToDecimalish(confirmation);  
        //System.out.println("确认号："+confirmationNumber);  
  
        byte[]window=Arrays.copyOfRange(bytes,112,128);  
        windowSize=binaryToDecimalish(window);  
        //System.out.println("窗口大小："+windowSize);  
  
    }  
}

**4.2.1.4** UDP协议**：**

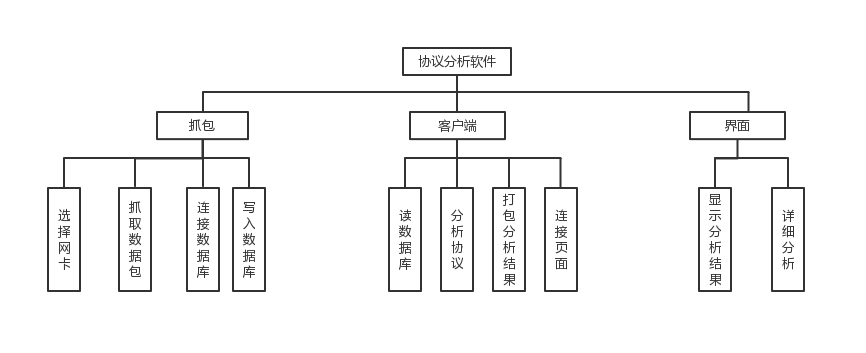
UDP 是OSI参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，UDP在IP报文的协议号是17。UDP协议全称是用户数据报协议，在网络中它与TCP协议一样用于处理数据包，是一种无连接的协议。在OSI模型中，在第四层—传输层，处于IP协议的上一层。面向报文，UDP对应用程序交付的报文，添加UDP首部后直接交给IP层。UDP首部开销较小，8字节（TCP为20字节、IP为20字节）。



·UDP协议分析：

package agreement;  
  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class UDP{  
public static int sourcePort;  
public static int destinationPort;  
public static int Length;  
public static int Checksum;  
public static void udp(byte[]udparr){  
byte[]sourceport=Arrays.copyOfRange(udparr,0,16);  
sourcePort=binaryToDecimalish(sourceport);  
//System.out.println("源端口："+udpsourcePort);  
  
byte[]destinationport=Arrays.copyOfRange(udparr,16,32);  
destinationPort=binaryToDecimalish(destinationport);  
//System.out.println("目的端口："+udpdestinationPort);  
  
byte[]length=Arrays.copyOfRange(udparr,32,48);  
Length=binaryToDecimalish(length);  
//System.out.println("长度："+udpLength);  
  
byte[]checksum=Arrays.copyOfRange(udparr,48,64);  
Checksum=binaryToDecimalish(checksum);  
//System.out.println("校验和："+udpChecksum);  
}  
}

4.2.2 功能设计



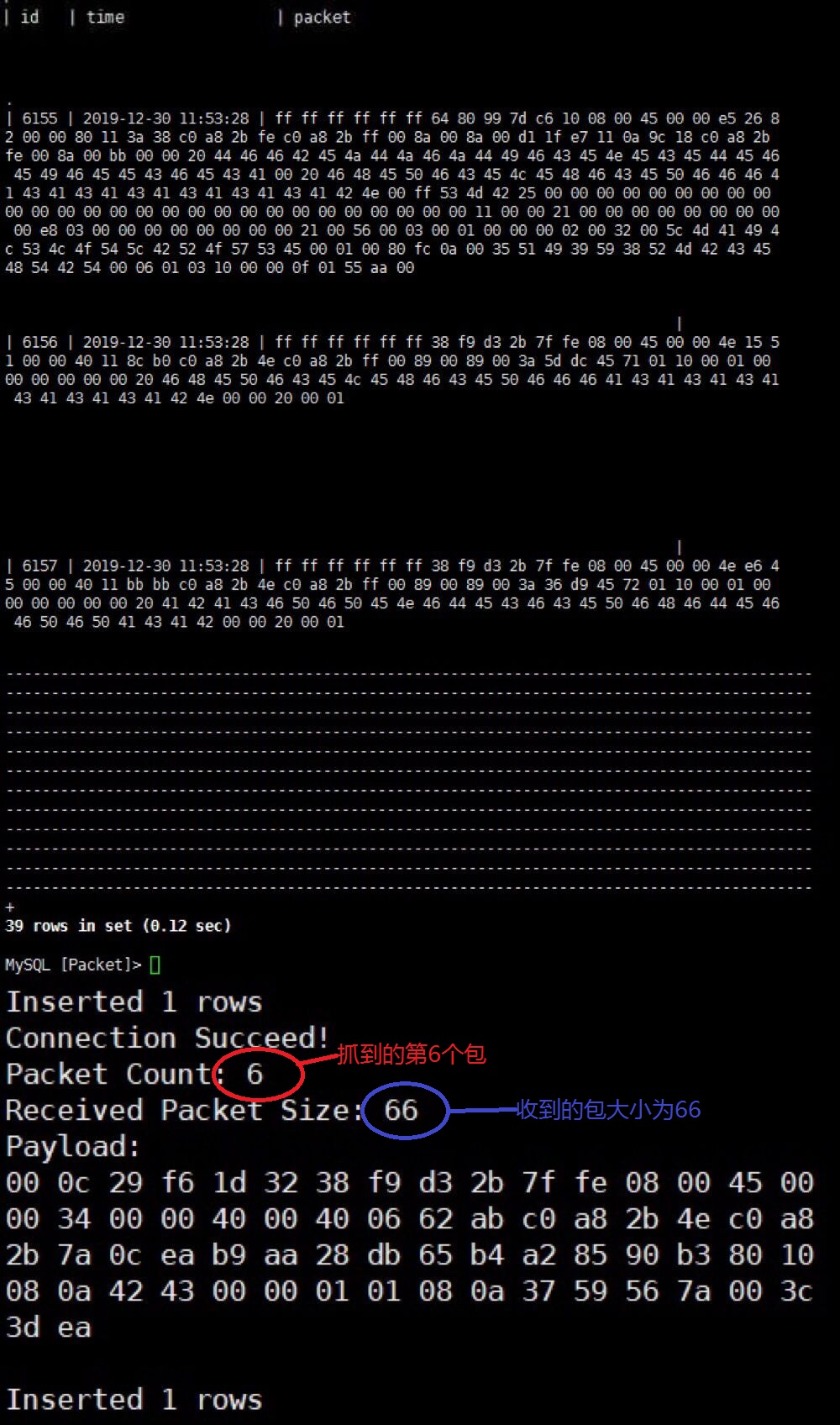
可读取多台服务器的数据，不局限于本机，使软件功能更广泛。

4.3 系统实现

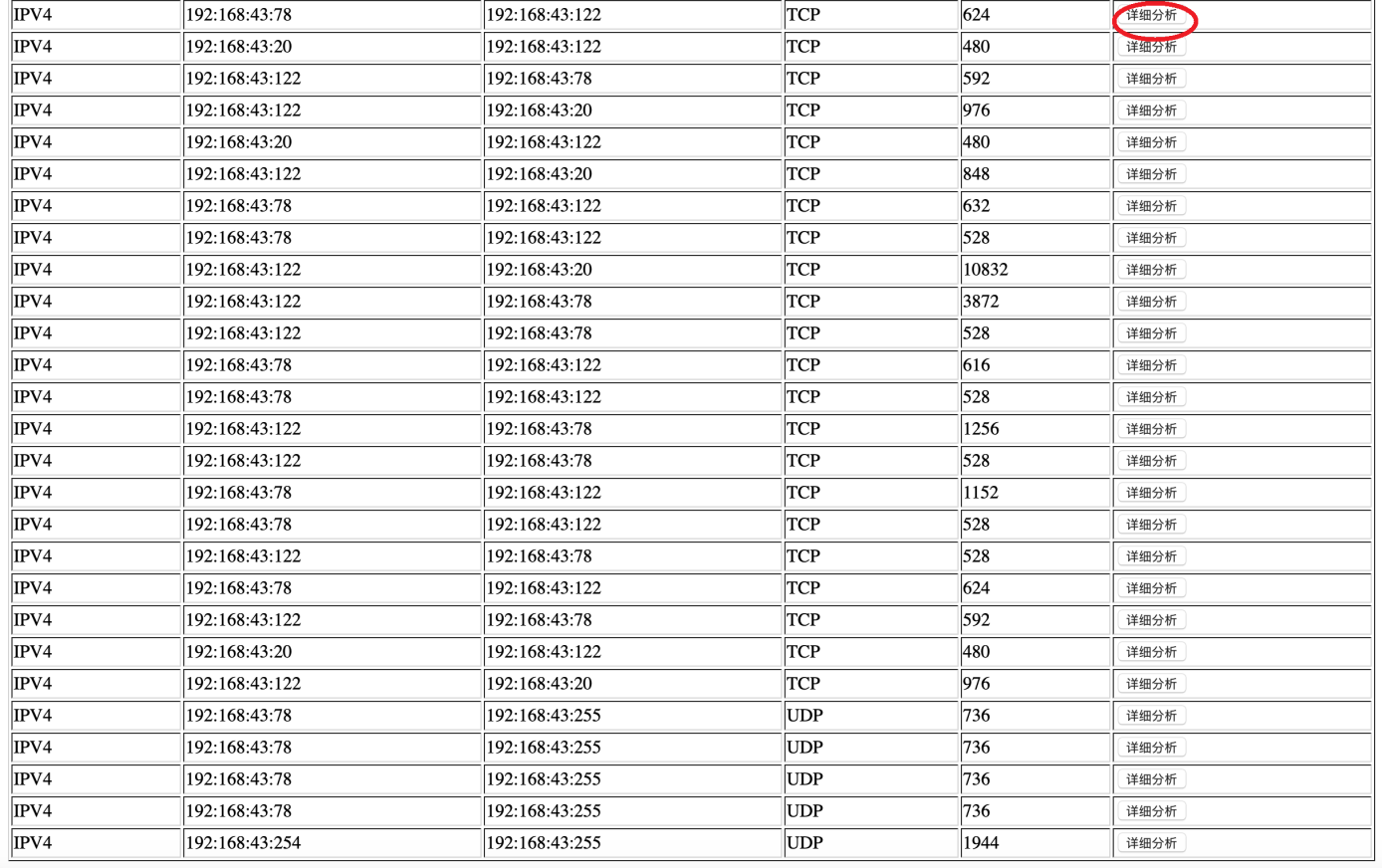
### qt_temp

# 程序的运行和测试结果

抓包：

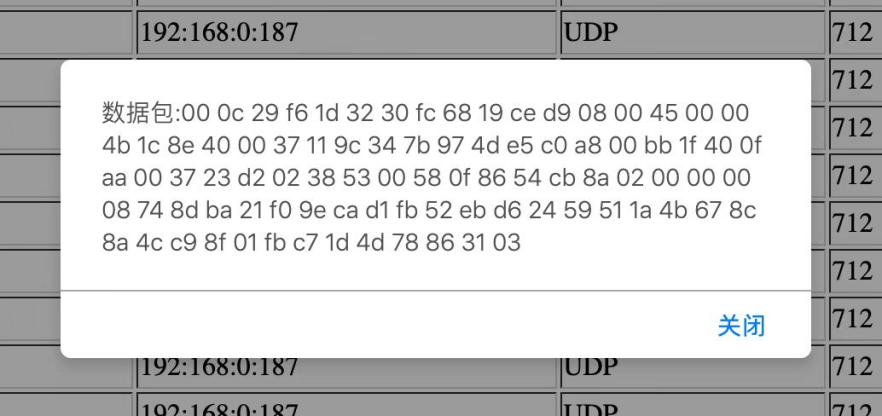


最终分析结果以网页界面形式显示：



分析后的结果通过serverlet连接后，以网页界面显示，点击表格中【详细分析】按钮跳转页面，弹出分析弹窗对数据包进行详细的分析。

跳转页面：



# 设计中的问题

试验中遇到的问题及解决：

1.抓数据包问题：

|  |  |
| --- | --- |
| 问题 | 解决 |
| （1）抓不出数据包 | 发现监测的网卡有问题，是一个虚拟网卡； |
| （2）后来发送给客户端的数据包都一样 | Linux服务器和Mac客户端连接不上 |
| （3）Linux服务器上的双进程难以控制 | 换成数据库链接Linux服务器和Mac客户端。 |

2.客户端问题：

|  |  |
| --- | --- |
| 问题 | 解决 |
| 客户端页面设计为动态但一直为静态 | 转用serverlet来连接页面与Java代码的联系 |

3.页面报500错误

|  |  |
| --- | --- |
| 问题 | 解决 |
| 页面报500错误 | 配置问题，已解决 |

七、附录

## 7.1 程序清单

·抓包：

#define\_CRT\_SECURE\_NO\_WARNINGS 1

#include<sys/time.h>

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<pcap.h>

#include<mysql/mysql.h>

unsigned int timeout=7;//超时时间7秒

#define PORT 3333

#define BACKLOG 1

#define MAXRECVLEN 65535

char trans[MAXRECVLEN];

char sql\_insert[MAXRECVLEN];

void change(int i,char tmp[]){

int low=i%16;

int high=i/16;

if(low<=9){

tmp[1]='0'+low;

}

else{

tmp[1]='a'+low-10;

}

if(high<=9){

tmp[0]='0'+high;

}

else{

tmp[0]='a'+high-10;

}

}

void processPacket(u\_char\*arg,const struct pcap\_pkthdr\*pkthdr,const u\_char\*packet)

{

MYSQL\*conn\_ptr;

//连接数据库

int ret=0;

conn\_ptr=mysql\_init(NULL);//初始化

if(!conn\_ptr)

{

printf("mysql\_init failed!\n");

return;

}

ret=mysql\_options(conn\_ptr,MYSQL\_OPT\_CONNECT\_TIMEOUT,(const char\*)&timeout);//设置超时>选项

if(ret)

{

printf("Options Set ERRO!\n");

}

conn\_ptr=mysql\_real\_connect(conn\_ptr,"192.168.0.194","root","root","packet",0,NULL,0);//连接MySQ L testdb数据库

if(conn\_ptr)

{

printf("Connection Succeed!\n");

}

int\*count=(int\*)arg;

printf("Packet Count:%d\n",++(\*count));

printf("Received Packet Size:%d\n",pkthdr->len);

printf("Payload:\n");

int i=0;

int j=0;

char tmp[2];

memset(trans,0,MAXRECVLEN);

for(i=0;i<pkthdr->len;++i){

change((int)packet[i],tmp);

trans[j++]=tmp[0];

trans[j++]=tmp[1];

trans[j++]='';

printf("%02x",packet[i]);

if((i+1)%16==0)

printf("\n");

}

printf("\n\n");

memset(sql\_insert,0,MAXRECVLEN);

sprintf(sql\_insert,"insert into Info(time,packet)VALUES(now(),'%s');",trans);

ret=mysql\_query(conn\_ptr,sql\_insert);

if(!ret)

{

printf("Inserted%lu rows\n",(unsigned long)mysql\_affected\_rows(conn\_ptr));//返回上次UPDATE更改行数

}

else

{

printf("Connect Erro:%d%s\n",mysql\_errno(conn\_ptr),mysql\_error(conn\_ptr));//返回错误代码、错误消息

}

mysql\_close(conn\_ptr);

}

int Get()

{

char errBuf[PCAP\_ERRBUF\_SIZE],\*devStr;

//devStr=pcap\_lookupdev(errBuf);

pcap\_if\_t\*alldevs;

pcap\_if\_t\*d;

int i=0;

char errbuf[PCAP\_ERRBUF\_SIZE];

/\*获取本地机器设备列表\*/

if(pcap\_findalldevs(&alldevs,errbuf)==-1)

{

fprintf(stderr,"Error in pcap\_findalldevs\_ex:%s\n",errbuf);

exit(1);

}

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

{

if(strcmp(d->name,"ens33")==0)

{

printf("%d.%s",++i,d->name);

break;

}

}

devStr=d->name;

if(devStr)

printf("success:device:%s\n",devStr);

else

{

printf("error:%s\n",errBuf);

exit(1);

}

pcap\_t\*device=pcap\_open\_live(devStr,65535,1,0,errBuf);

if(!device)

{

printf("error:pcap\_open\_live():%s\n",errBuf);

exit(1);

}

struct bpf\_program filter;

pcap\_compile(device,&filter,"ip",1,0);

pcap\_setfilter(device,&filter);

int count=0;

pcap\_loop(device,-1,processPacket,(u\_char\*)&count);

printf("transInGet=%s\n\n",trans);

pcap\_close(device);

return 0;

}

int main(int argc,char\*argv[])

{

Get();

return 0;

}

·客户端主程序：

package agreement;  
import binaryToDecimalism.BinaryToDecimalish;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.sql.\*;  
import java.util.Arrays;  
import java.util.LinkedList;  
import java.util.List;  
  
import static agreement.UDP.Length;  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
import static binaryToDecimalism.BinaryToDecimalish.hexadecimalbinary;  
@WebServlet("/协议分析")  
public class ServletTest extends javax.servlet.http.HttpServlet {  
    public static String packet=null;  
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
        this.doGet(request,response);  
    }  
  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
        response.setContentType("text/html;charset=UTF-8");  
        List<Packages> list = new LinkedList<>();  
        int i = 0;  
        while (i < 42) {  
            try {  
                int id = -1;  
                Class.forName("com.mysql.jdbc.Driver");  
                Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/Packet?useSSL=false", "root", "root");  
                Statement statement = connection.createStatement();  
                ResultSet s = statement.executeQuery("select \* from Info where id in(select max(id) from Info) ");  
                while (s.next()) {  
                    packet = s.getString("packet");  
                    id = s.getInt("id");  
                }  
                statement.close();  
                Statement statement1 = connection.createStatement();  
                int row = statement1.executeUpdate("delete from Info where id = " + id + "");  
                statement.close();  
                connection.close();  
            } catch (ClassNotFoundException e) {  
                e.printStackTrace();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
            byte[] bytes = hexadecimalbinary(packet);  
            byte[] mac = Arrays.copyOfRange(bytes, 0, 112);  
            //ip层协议分析  
            int agreementnumber=binaryToDecimalish(Arrays.copyOfRange(bytes,112,116));  
            if(agreementnumber!=4){  
                continue;  
            }  
            int iplength = binaryToDecimalish(Arrays.copyOfRange(bytes, 116, 120))\*32;  
            int ipbegin = 112;  
            int ipend = 112+ iplength;  
            byte[] ipArr = Arrays.copyOfRange(bytes, ipbegin, ipend);  
            int upperLayerProtocol = binaryToDecimalish(Arrays.copyOfRange(bytes, ipbegin + 72, ipbegin + 80));  
            IP.ip(ipArr);  
            //TCP协议分析  
            if (upperLayerProtocol == 6) {  
                int tcplength = binaryToDecimalish(Arrays.copyOfRange(bytes, ipend + 96, ipend + 96 + 4)) \* 32;  
                int tcpbegin = ipend;  
                int tcpend = ipend + tcplength;  
                byte[] tcpArr = Arrays.copyOfRange(bytes, tcpbegin, tcpend);  
                TCP.tcp(tcpArr);  
            } else if (upperLayerProtocol == 1) {  
                //ICMP  
                byte[] icmpArr = Arrays.copyOfRange(bytes, ipend, ipend + 32);  
                ICMP.icmp(icmpArr);  
            } else if (upperLayerProtocol == 17) {  
                //UDP  
                byte[] udpArr = Arrays.copyOfRange(bytes, ipend, ipend + 64);  
                UDP.udp(udpArr);  
            }  
  
            Packages p = new Packages( BinaryToDecimalish.length, IP.versions,IP.headLength,  
                    IP.totalLength, IP.identificationNumber, IP.signnumber,  
                    IP.SliceOffsetNUmber,IP.agreement, IP.headCheckSum,  
                    IP.destinationaddress, IP.sourceaddress, ICMP.Type,  
                    ICMP.code, ICMP.Checksum, TCP.sourcePort,  
                    TCP.destinationPort, TCP.serialNumber, TCP.confirmationNumber,  
                    TCP.windowSize, UDP.sourcePort, UDP.destinationPort,  
                    UDP.Length,UDP.Checksum,packet);  
            list.add(p);  
            i++;  
        }  
        for (Packages z:list) {  
            System.out.println(z.Length);  
        }  
        request.setAttribute("list",list);  
        request.getRequestDispatcher("index.jsp").forward(request,response);  
    }  
}

·IP协议分析：

package agreement;  
  
import java.io.UnsupportedEncodingException;  
import java.util.Arrays;  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
public class IP{  
public static String versions;  
public static int headLength;  
public static int totalLength;  
public static int identificationNumber;  
public static int signnumber;  
public static int SliceOffsetNUmber;  
public static String agreement;  
public static int headCheckSum;  
public static String destinationaddress;  
public static String sourceaddress;  
  
public static void ip(byte[]bytes)throws UnsupportedEncodingException{  
  
byte[]versionsarr=Arrays.copyOfRange(bytes,0,4);  
int versionsnum=binaryToDecimalish(versionsarr);  
if(versionsnum==4){  
versions="IPV4";  
}  
if(versionsnum==6){  
versions="IPV6";  
}  
//System.out.println("版本："+versions);  
  
byte[]head=Arrays.copyOfRange(bytes,4,8);  
headLength=binaryToDecimalish(head);  
//System.out.println("首部长度："+headLength);  
  
byte[]Length=Arrays.copyOfRange(bytes,16,32);  
totalLength=binaryToDecimalish(Length);  
//System.out.println("总长度："+totalLength);  
  
byte[]identification=Arrays.copyOfRange(bytes,32,48);  
identificationNumber=binaryToDecimalish(identification);  
//System.out.println("标识："+identificationNumber);  
  
byte[]sign=Arrays.copyOfRange(bytes,48,51);  
signnumber=binaryToDecimalish(sign);  
//System.out.println("标志："+signnumber);  
  
byte[]SliceOffset=Arrays.copyOfRange(bytes,51,64);  
SliceOffsetNUmber=binaryToDecimalish(SliceOffset);  
//System.out.println("片偏移："+SliceOffsetNUmber);  
  
byte[]Agreement=Arrays.copyOfRange(bytes,72,80);  
int agreementnum=binaryToDecimalish(Agreement);  
if(agreementnum==1){  
agreement="ICMP";  
}else if(agreementnum==6){  
agreement="TCP";  
}else if(agreementnum==17){  
agreement="UDP";  
}else{  
agreement="IP";  
}  
  
byte[]headChecknum=Arrays.copyOfRange(bytes,80,96);  
headCheckSum=binaryToDecimalish(headChecknum);  
//System.out.println("首部校验和："+headCheckSum);  
  
int[]sourceAddress=new int[4];  
sourceAddress[0]=binaryToDecimalish(Arrays.copyOfRange(bytes,96,104));  
sourceAddress[1]=binaryToDecimalish(Arrays.copyOfRange(bytes,104,112));  
sourceAddress[2]=binaryToDecimalish(Arrays.copyOfRange(bytes,112,120));  
sourceAddress[3]=binaryToDecimalish(Arrays.copyOfRange(bytes,120,128));  
StringBuffer sb=new StringBuffer();  
for(int i=0;i<4;i++){  
if(i==3){  
sb.append(sourceAddress[i]);  
break;  
}  
sb.append(sourceAddress[i]);  
sb.append(":");  
}  
sourceaddress=sb.toString();  
  
int[]destinationAddress=new int[4];  
destinationAddress[0]=binaryToDecimalish(Arrays.copyOfRange(bytes,128,136));  
destinationAddress[1]=binaryToDecimalish(Arrays.copyOfRange(bytes,136,144));  
destinationAddress[2]=binaryToDecimalish(Arrays.copyOfRange(bytes,144,152));  
destinationAddress[3]=binaryToDecimalish(Arrays.copyOfRange(bytes,152,160));  
StringBuffer sb1=new StringBuffer();  
for(int i=0;i<4;i++){  
if(i==3){  
sb1.append(destinationAddress[i]);  
break;  
}  
sb1.append(destinationAddress[i]);  
sb1.append(":");  
}  
destinationaddress=sb1.toString();  
  
}  
}

·packages类：

package agreement;  
  
import binaryToDecimalism.BinaryToDecimalish;  
  
public class Packages{  
public String response;//拿到的包  
public int Length;  
public String ipversions;  
public int ipheadLength;  
public int iptotalLength;  
public int ipidentificationNumber;  
public int ipsignnumber;  
public int ipSliceOffsetNUmber;  
public String ipagreement;  
public int ipheadCheckSum;  
public String ipdestinationaddress;  
public String ipsourceaddress;  
public String icmpType;  
public String icmpcode;  
public int icmpChecksum;  
public int tcpsourcePort;  
public int tcpdestinationPort;  
public int tcpserialNumber;  
public int tcpconfirmationNumber;  
public int tcpwindowSize;  
public int udpsourcePort;  
public int udpdestinationPort;  
public int udpLength;  
public int udpChecksum;  
  
  
public Packages(int length,String versions,int headLength,int totalLength,int identificationNumber,int signnumber,  
int SliceOffsetNUmber,String agreement,int headCheckSum,String destinationaddress,String sourceaddress,  
String Type,String code,int Checksum,int sourcePort,int destinationPort,int serialNumber,int confirmationNumber,  
int windowSize,int udsourcePort,int uddestinationPort,int udLength,int udChecksum,String Response){  
response=ServletTest.packet;  
Length=BinaryToDecimalish.length;  
ipversions=IP.versions;  
ipheadLength=IP.headLength;  
iptotalLength=IP.totalLength;  
ipidentificationNumber=IP.identificationNumber;  
ipsignnumber=IP.signnumber;  
ipSliceOffsetNUmber=IP.SliceOffsetNUmber;  
ipagreement=IP.agreement;  
ipheadCheckSum=IP.headCheckSum;  
ipdestinationaddress=IP.destinationaddress;  
ipsourceaddress=IP.sourceaddress;  
icmpType=ICMP.Type;  
icmpcode=ICMP.code;  
icmpChecksum=ICMP.Checksum;  
tcpsourcePort=TCP.sourcePort;  
tcpdestinationPort=TCP.destinationPort;  
tcpserialNumber=TCP.serialNumber;  
tcpconfirmationNumber=TCP.confirmationNumber;  
tcpwindowSize=TCP.windowSize;  
udpsourcePort=UDP.sourcePort;  
udpdestinationPort=UDP.destinationPort;  
udpLength=UDP.Length;  
udpChecksum=UDP.Checksum;  
  
  
}  
}

·ICMP协议分析：

package agreement;  
  
import java.io.UnsupportedEncodingException;  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class ICMP {  
    public static String Type;  
    public static String code;  
    public static int Checksum;  
    public static void icmp(byte[]icmpArr) throws UnsupportedEncodingException {  
        byte[]type= Arrays.copyOfRange(icmpArr,0,8);  
        int Typenum=binaryToDecimalish(type);  
        if(Typenum==3){  
           Type="终点不可达";  
        }  
        if(Typenum==11){  
            Type="时间超过";  
        }  
        if(Typenum==12){  
           Type="参数问题";  
        }  
        if(Typenum==5){  
            Type="改变路由";  
        }  
        if(Typenum==8||Typenum==0){  
            Type="回送请求或回答";  
        }  
        if(Typenum==13||Typenum==14){  
            Type="时间戳请求或回答";  
        }  
  
        code=new String(icmpArr,8,16-8,"UTF-8");  
        //System.out.println(code);  
  
        byte[]checksum=Arrays.copyOfRange(icmpArr,16,32);  
        Checksum=binaryToDecimalish(checksum);  
        //System.out.println("校验和："+Checksum);  
  
    }  
}

·TCP协议分析：

package agreement;  
  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class TCP {  
    public static int sourcePort;  
    public static int destinationPort;  
    public static int serialNumber;  
    public static int confirmationNumber;  
    public static int windowSize;  
    public static void tcp(byte[]bytes) {  
        byte[]source= Arrays.copyOfRange(bytes,0,16);  
        sourcePort=binaryToDecimalish(source);  
        //System.out.println("源端口："+sourcePort);  
  
        byte[]destination=Arrays.copyOfRange(bytes,16,32);  
        destinationPort=binaryToDecimalish(destination);  
        //System.out.println("目的端口："+destinationPort);  
  
        byte[] serial=Arrays.copyOfRange(bytes,32,64);  
        serialNumber=binaryToDecimalish(serial);  
        //System.out.println("序号："+serialNumber);  
  
        byte[]confirmation=Arrays.copyOfRange(bytes,64,96);  
        confirmationNumber=binaryToDecimalish(confirmation);  
        //System.out.println("确认号："+confirmationNumber);  
  
        byte[]window=Arrays.copyOfRange(bytes,112,128);  
        windowSize=binaryToDecimalish(window);  
        //System.out.println("窗口大小："+windowSize);  
  
    }  
}

·UDP协议分析：

package agreement;  
  
import java.util.Arrays;  
  
import static binaryToDecimalism.BinaryToDecimalish.binaryToDecimalish;  
  
public class UDP{  
public static int sourcePort;  
public static int destinationPort;  
public static int Length;  
public static int Checksum;  
public static void udp(byte[]udparr){  
byte[]sourceport=Arrays.copyOfRange(udparr,0,16);  
sourcePort=binaryToDecimalish(sourceport);  
//System.out.println("源端口："+udpsourcePort);  
  
byte[]destinationport=Arrays.copyOfRange(udparr,16,32);  
destinationPort=binaryToDecimalish(destinationport);  
//System.out.println("目的端口："+udpdestinationPort);  
  
byte[]length=Arrays.copyOfRange(udparr,32,48);  
Length=binaryToDecimalish(length);  
//System.out.println("长度："+udpLength);  
  
byte[]checksum=Arrays.copyOfRange(udparr,48,64);  
Checksum=binaryToDecimalish(checksum);  
//System.out.println("校验和："+udpChecksum);  
}  
}

·网页界面设计：

<%@page import="agreement.Packages"%>  
<%@page import="java.util.List"%><%--  
Created by IntelliJ IDEA.  
User:sangzhan  
Date:2019-12-25  
Time:16:36  
To change this template use File|Settings|File Templates.  
--%>  
<%@page contentType="text/html;charset=UTF-8"language="java"%>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title>欢迎使用KingShark</title>  
</head>  
  
<body>  
<center>  
<h1>WireShark Pro</h1>  
</center>  
  
<table width="100%"border="1">  
<tr>  
<th>versions</th>  
<th>Source</th>  
<th>destination</th>  
<th>Protocol</th>  
<th>Length</th>  
<th>start analyze</th>  
</tr>  
<%List<Packages>list=(List<Packages>)request.getAttribute("list");%>  
<%for(Packages items:list){%>  
<tr>  
<td><%=items.ipversions%></td>  
<td><%=items.ipsourceaddress%></td>  
<td><%=items.ipdestinationaddress%></td>  
<td><%=items.ipagreement%></td>  
<td><%=items.Length%></td>  
<td><button onclick="particularAnalyst()">详细分析</button></td>  
<script>  
function particularAnalyst(){  
alert("IP协议分析:\n\n版本:<%=items.ipversions%>\n首部长度:<%=items.ipheadLength%>\n总长度："  
+"<%=items.iptotalLength%>\n标识:<%=items.ipidentificationNumber%>\n标志:<%=items.ipsignnumber%>"  
+"\n片偏移:<%=items.ipSliceOffsetNUmber%>\n协议:<%=items.ipagreement%>\n首部校验和:<%=items.ipheadCheckSum%>"  
+"\n源地址:<%=items.ipsourceaddress%>\n目的地址:<%=items.ipdestinationaddress%>");  
  
<%if(items.ipagreement=="TCP"){%>  
  
alert("TCP协议分析:\n源端口:<%=items.tcpsourcePort%>\n目的端口:<%=items.tcpdestinationPort%>\n序号:"  
+"<%=items.tcpserialNumber%>\n确认号:<%=items.tcpconfirmationNumber%>\n窗口大小:<%=items.tcpwindowSize%>");  
<%}%>  
<%if(items.ipagreement=="ICMP"){%>  
  
alert("ICMP协议分析:\n\n类型:<%=items.icmpType%>\n代码:<%=items.icmpcode%>\n校验和:<%=items.icmpChecksum%>");  
<%}%>  
<%if(items.ipagreement=="UDP"){%>  
  
alert("UDP协议分析:\n\n源端口:<%=items.udpsourcePort%>\n目的端口:<%=items.udpdestinationPort%>"  
+"\n长度:<%=items.udpLength%>\n校验和:<%=items.udpChecksum%>");  
<%}%>  
alert("数据包:<%=items.response%>");  
  
  
}  
</script>  
<%}%>  
</tr>  
  
</table>  
</body>  
</html>

## 7.2 其他需要说明的内容

虽然这个网络数据捕获和协议分析软件在实际应用中还有些局限性，但通过研究这方面的课题，可以更深刻地理解网络各层通讯协议的机理，也加强了网络编程的技巧。我们后期也会写双进程服务器来弥补目前的局限性。