摘 要

为了完成数据包的获取与分析，使用winPcap的Java实现：Jpcap对原始数据包的获取并利用JAVA语言定义以太网，IP协议、UDP协议、TCP、APR协议数据结构，利用继承关系对原始数据包头部分离之后进行包头的按字节逐个分析获取不同协议的头部信息最后展示在命令行窗口。

**关键词：** 网络抓包，协议定义，数据包分析，Java， Jpcap

目 录

[第一章 绪 论 1](#_Toc43056955)

[1.1 研究工作的背景与意义 1](#_Toc43056956)

[1.2 抓包软件介绍 1](#_Toc43056957)

[1.3 本论文的结构安排 1](#_Toc43056958)

[第二章 常见网络协议基础 2](#_Toc43056959)

[2.1 数据链路层协议 2](#_Toc43056960)

[2.1.1以太网协议 2](#_Toc43056961)

[2.2 网络层协议 3](#_Toc43056962)

[2.2.1 IP（Internet Protocol）协议 3](#_Toc43056963)

[2.2.2 ARP（Address Resolution Protocol）协议 4](#_Toc43056964)

[2.3 传输层 5](#_Toc43056965)

[2.3.1 TCP（Transmission Control Protocol）协议 5](#_Toc43056966)

[2.3.2 UDP（User Datagram Protocol）协议 6](#_Toc43056967)

[第三章 6](#_Toc43056968)

[3.1 定义工具类 6](#_Toc43056969)

[3.2 定义协议的数据结构 8](#_Toc43056970)

[3.2.1 以太网数据结构 8](#_Toc43056971)

[3.2.2 ARP数据结构 9](#_Toc43056972)

[3.2.3 IP数据结构 9](#_Toc43056973)

[3.2.4 TCP数据结构 10](#_Toc43056974)

[3.2.5 UDP数据结构 11](#_Toc43056975)

[3.3 获取网卡列表 11](#_Toc43056976)

[3.4 捕获数据包并输出解析信息。 12](#_Toc43056977)

[第四章 全文总结与展望 15](#_Toc43056978)

[4.1 全文总结 15](#_Toc43056979)

[4.2 后续工作展望 15](#_Toc43056980)

[致 谢 16](#_Toc43056981)

[参考文献 17](#_Toc43056982)

第一章 绪 论

1.1 研究工作的背景与意义

抓包是程序调试的重要过程之一，在程序运行的数据交互中，传输的数据一般都是以数据包的形式传输。在这个发送和接收的过程中，可能发出的数据包中有错误的数据，也可能接收的包中有错误的数据，从而导致后期的程序处理出错。

因此我们直接抓取传输中的数据包，通过其他工具或方法解析数据包中的原始数据，就可以定位是否是因为数据出错而导致程序出错，如果出错又是那一部分的数据出错等。一般抓包测试多用于通信行业和网络行业的测试。

1.2 抓包软件介绍

winPcap是Windows平台下一个免费的，公共网络访问系统，开发winPcap这个项目的目的在于为win32应用程序提供访问网络底层的能力。它用于windows系统下的直接的网络编程。winPcap是使用C语言编写，而本文使用的是其Java实现的Jpcap，Jpcap是一个中间件它调用winPcap给Java提供公共的接口

1.3 本论文的结构安排

本文的章节结构安排如下：

1. 绪论
2. 介绍常见网络协议即本程序定义的协议结构
3. 协议定义的详细介绍
4. 全文的总结与展望

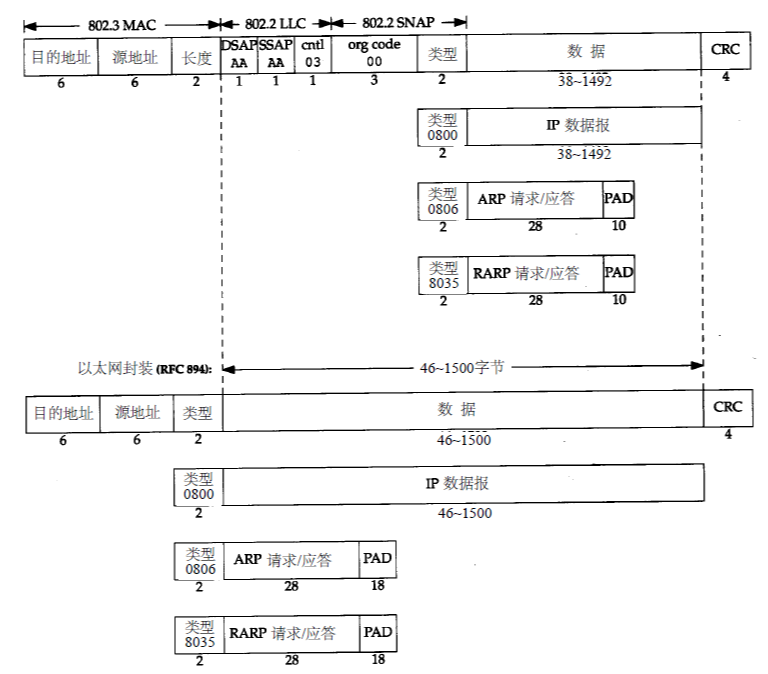
第二章 常见网络协议基础

网络协议是计算机通信双方必须遵守的约定，计算机网络就是构建在一层一层协议的叠加之中，从而保证了对数据顺利传递。

2.1 数据链路层协议

2.1.1以太网协议

网络链路数据分为两种：IEEE 802.2/802.3( RFC1042)和以太网的封装格式（RFC 894）。两种协议的格式如下：



图表 1 RFC1042和RFC 894格式

本文只介绍Ethernet II，它的头部由三个部分组成：

目的MAC（6）：源MAC（6）：类型（2）

类型字段由多个取值，0800H表示IP数据类型，0806表示ARP数据类型，8035表示RARP协议。注意到数据部分最大长度1500字节。

2.2 网络层协议

网络层的目的是实现两个端系统之间的数据透明传送，为传输层提供服务。

协议主要有：IP协议、ARP协议、ICMP协议、IGMP协议。

2.2.1 IP（Internet Protocol）协议

IP协议在网络层的功能主要是寻址和分段，协议可以首部的地址信息将数据包传送到目的地址，在寻址的过程中使用选路路由算法。如果网络内MTU（Maximum Transmission Unit）小于数据包大小，IP协议还会对数据进行切片并在首部标出。



图表 2 IP结构

第一行：

0~3字节：版本信息，标识IP协议的版本号，0100标识IPv4，1010表示IPv6。

4~7字节：首部长度，用来标识IP报文首部的长度单位是4字节，最大长度为(-1)=60个字节，最小为固定部分20字节，首部长度大多为20字节。

8~15字节：用于标识IP包的优先程度，但现在大多未启用。

16~31字节：标识IP报文总长度（首部+数据部分）单位字节，最长字节。

第二行：

0~15：标识字段，如果IP报文的最大长度超过MTU（以太网MTU=1500字节）则对报文进行切片，每一片的标识相同。

16~18：3位标志位，但目前只有2位有意义；最低位为MF，MF=1代表后面还有分片的数据报，MF=0代表当前数据报已是最后的数据报。次低位为DF，DF=1代表不能分片，DF=0代表可以分片。

19~31：相对片位移，标志了该报文在原始报文中的相对位置。

第三行：

0~7：标识IP数据报可以经过的路由器数。

8~15：标识上层协议类型：17代表UDP、6代表TCP、1代表ICMP、2代表IGMP。

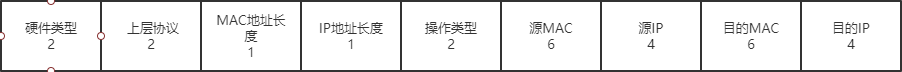
16~31：校验和，用于校验数据是否发生差错。

第四行：源32位IP地址。

第五行：目的32位IP地址。

2.2.2 ARP（Address Resolution Protocol）协议

ARP协议即地址解析协议用于实现从 IP 地址到 MAC 地址的映射，即询问目标IP对应的MAC地址。ARP报文又分为ARP请求和ARP应答报文，它们都具有统一的格式：



图表 3 ARP结构

硬件类型：2字节，表示ARP报文可以在哪种类型的网络上传输，值为1时表示为以太网地址。

上层协议类型：占两字节，表示硬件地址要映射的协议地址类型，映射IP地址时的值为0x0800。

MAC地址长度：占一字节，标识MAC地址长度，以字节为单位，此处为6。

IP协议地址长度：占一字节，标识IP得知长度，以字节为单位，此处为4。

操作类型：占2字节，指定本次ARP报文类型。1标识ARP请求报文，2标识ARP应答报文。

源MAC地址：占6字节，标识发送设备的硬件地址。

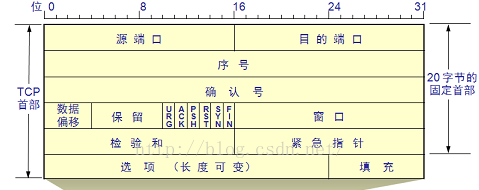
源IP地址：占4字节，标识发送方设备的IP地址。

目的MAC地址：占6字节，表示接收方设备的硬件地址，在请求报文中该字段值全为0，即00-00-00-00-00-00，表示任意地址，因为现在不知道这个MAC地址。

目的IP地址：占4字节，表示接受方的IP地址。

2.3 传输层

2.3.1 TCP（Transmission Control Protocol）协议

TCP协议是一种传输层协议，提供应用程序间的通信。其功能包括：一、格式化信息流；二、提供可靠传输。为实现后者，传输层协议规定接收端必须发回确认，并且假如分组丢失，必须重新发送。首部结构为：

图表 4 TCP首部结构

第一行：

0~15：源进程端口号。

16~32：目的进程端口号。

第二行：代表当前TCP数据段的第一个字节占整个字节流的相对位置。

第三行：代表接收端希望接收的数据序号，为上次接收到数据报的序号+1， 当ACK标志位为1时才生效。

第四行：

0~3：代表报文首部长度。单位4字节。

10~15: 6个标志位，SYN，为同步标志，用于数据同步。ACK，为确认序号 ACK=1时确认号才有效。

FIN，为结束序号，用于发送端提出断开连接。

URG，为紧急序号，URG=1是紧急指针有效。

PSH，指示接收方立即将数据提交给应用层，而不是等待缓冲区满。

RST，重置连接。

16~31：窗口大小。

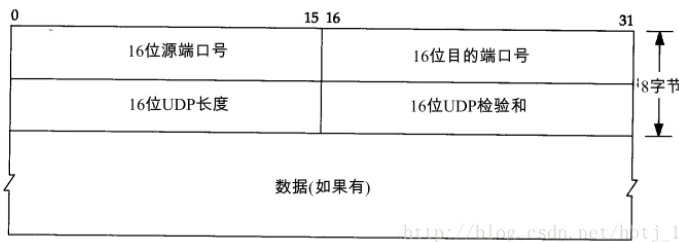
第五行：

0~15：校验和，用于校验TCP数据是否出错。

16~31：只有当URG标识位为1时，紧急指针才有效。紧急指针的值与序号 的相加值为紧急数据的最后一个字节位置。用于发送紧急数据。

2.3.2 UDP（User Datagram Protocol）协议

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，IETF RFC 768是UDP的正式规范。UDP提供了无连接通信，且不对传送数据包进行可靠性保证，适合于一次传输少量数据，UDP传输的可靠性由应用层负责。其头部结构为：



图表 5 UDP结构

第一行：

0~15：源程序端口号。

16~31：目的端口号。

第二行：

0~15：UDP报文长度，最小8字节。发送一份0字节的UDP数据报是 O K。

16~31：校验和。

第三章

利用Jpcap提供的接口对数据包进行处理，解析出包头信息。

3.1 定义工具类

在正式对数据包解析之前先定义一个工具类来实现辅助功能。

由于Java的byte数据类型范围为-128~127而首部信息均非负所以我们需要将抓到的数据包首部进行处理。在输出地址时我们习惯以16进制表示需要整数转化为16进制字符串。在输出标志位时需要整数向二进制转化。

* 将byte转化为256模系统下的int类型。

实现：public static int regular(byte b) {

//转化为256下的模系统

int res = 0;

res = (256 + b)%256;

return res;

}

* 将字节或者整型转化为十六进制

实现：

public static String bytesToHex(byte... bytes) {

StringBuffer sb = new StringBuffer();

for(int i = 0; i < bytes.length; i++) {

String hex = Integer.toHexString(bytes[i] & 0xFF);

if(hex.length() < 2){

sb.append(0);

}

sb.append(hex);

}

return sb.toString();

}

重载函数的自变量为int这里不再重复。

* 将整数转化为二进数

public static int[] intToBin(int b) {

int[] res = new int[8];

res[0] = (b>>7) & 0x1;

res[1] = (b>>6) & 0x1;

res[2] = (b>>5) & 0x1;

res[3] = (b>>4) & 0x1;

res[4] = (b>>3) & 0x1;

res[5] = (b>>2) & 0x1;

res[6] = (b>>1) & 0x1;

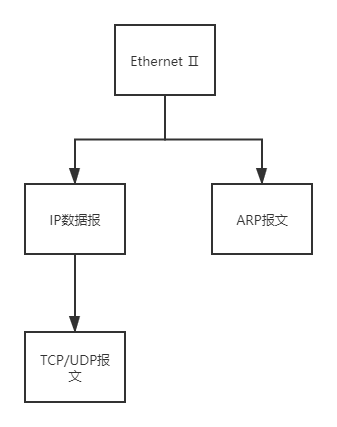
res[7] = (b) & 0x1;

return res;

}

3.2 定义协议的数据结构

采用自底向上的线性继承结构：



图表 6 协议类继承结构

从链路层开始，逐层拆包对包内信息进行解读继承结构使得数据部分得到保留而不同层次的子类对头部信息解析起点也逐渐靠后。

3.2.1 以太网数据结构

Class MyEthernetPacket

属性：

|  |  |
| --- | --- |
| Int[] dst\_mac | 目的MAC地址 |
| Int[] src\_mac | 源MAC地址 |
| String type | 类型字段：arp 、ip、 rarp |
| Byte[] data | 数据部分 |

|  |  |
| --- | --- |
| byte[] head | 头部信息 |
| Int[] con\_head | 转化之后的头部 |

方法：

|  |  |
| --- | --- |
| MyEthernetPacket(Packet p ) | 接受一个原始数据包，初始化类。 |
| String toString() | 重写toString方法，展示以太网信息，格式：  源MAC🡪目的MAC. |

构造方法逻辑：

直接将底层Packet类作为参数，获得首部字节数组浅拷贝给head，然后循环调用tool工具类中的regular函数进行转化赋给con\_head。根据协议的结构信息可知，前12字节为源端和目的端的MAC信息，第13、14字节为类型信息。通过switch语句来判断类型字段并赋值给属性。

3.2.2 ARP数据结构

Class MyARPPacket extends MyEthernetPacket

属性：

|  |  |
| --- | --- |
| Int hardware\_type | 硬件类型字段 |
| String upper\_protocol | 上层协议 |
| String type | 类型字段：arp 、ip、 rarp |
| Int mac\_len 、ip\_len、  op\_type | 分别代表：MAC地址长度、IP地址长度、操作类型。 |
| Int[] src\_ip、dst\_ip | 源、目的端IP地址 |

方法：

|  |  |
| --- | --- |
| MyARPPacket (Packet p ) | 接受一个原始数据包，初始化类。 |
| Void display() | 展示所有ARP字段信息。 |

构造方法逻辑：

将底层Packet类作为参数，调用父类构造函数进行第一步初始化然后从第十五个字节开始解析，于ARP结构对应。

3.2.3 IP数据结构

Class MyARPPacket extends MyEthernetPacket

属性：

|  |  |
| --- | --- |
| Int version | 版本信息： IPv4、IPv6 |
| Int head\_len | 首部长度 |
| Int total\_len | 总长度 |
| Int identification | 标识 |
| Int flag | 标志 |
| Int offset | 偏移 |
| Int ttl | 生存时间 |
| String protocol | 上层协议：ICMP 、IGMP 、TCP、 UDP |
| Int[] src、dst | 源、目的端IP地址 |
| String check\_sum | 校验和 |

方法：

|  |  |
| --- | --- |
| MyIPPacket(Packet p) | 构造方法，初始化类 |
| String getAddress() | 返回报文的源、目的端地址。 |
| void display() | 展示IP报文所有头部信息。 |

构造方法逻辑：

将底层Packet类作为参数，调用父类构造函数进行第一步初始化然后判断type字段是否为“ip”，如果不是则输出"This is not an IP packet"停止后续初始化返回否则继续后续初始化。从第15个字节开始进行解析switch对第25个字节进行协议类型判断。校验和需要调用工具类中的bytesToHex方法转化为字符串。

3.2.4 TCP数据结构

Class MyTCPPacket extends MyIPPacket

属性：

|  |  |
| --- | --- |
| Int src\_port、dst\_port | 源、目的端口号 |
| Long seq\_num、ack\_num | 序号、确认号 |
| Int tcp\_head\_len | Tcp首部长度 |
| Int urg、ack、psh、rst  syn、fin | 六个标志位 |
| Int win\_size | 窗口大小 |
| String tcp\_check\_sum | TCP 校验和 |
| Int urgent\_pointer | 紧急指针 |
| Int[] options | 选项 |

方法：

|  |  |
| --- | --- |
| MyTCPPacket(Packet p) | 构造方法，初始化类。 |
| void display( ) | 展示TCP所有字段信息。 |

构造方法逻辑：

将底层Packet类作为参数，调用父类构造函数进行第一步初始化然后判断protocol字段是否为“tcp”，如果不是则输出"This is not an TCP packet"停止后续初始化返回否则继续后续初始化。从第35个字节开始解析。

3.2.5 UDP数据结构

Class MyUDPPacket extends MyIPPacket

属性：

|  |  |
| --- | --- |
| MyUDPPacket(Packet p) | 构造方法初始化类 |
| void display( ) | 展示UDP首部所有字段信息。 |

构造方法逻辑：

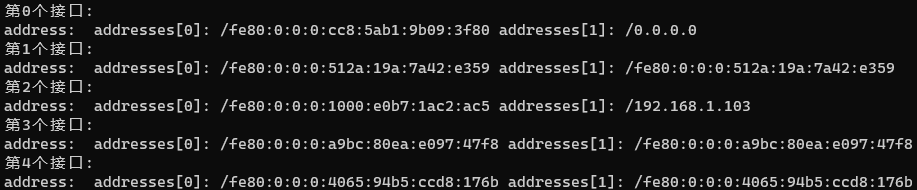
将底层Packet类作为参数，调用父类构造函数进行第一步初始化然后判断protocol字段是否为“udp”，如果不是则输出"This is not an UDP packet"停止后续初始化返回否则继续后续初始化。从第35个字节开始解析。

3.3 获取网卡列表

Class Name

该类的主要目的是获取并输出所有网卡列表，又用户选择一个网卡进行抓包。

效果如图：



图表 7 网卡列表

从图中可以看出我的2号网卡是在使用当中，IPV4地址为192.168.1.103

我们抓包就从从二号网卡接口开始。

3.4 捕获数据包并输出解析信息。

首先实现一个Jpcap接口PacketReceiver该接口是专门处理被捕获的包，需要实现void receivePacket(Packet p)方法。

逻辑：首先利用p实例化一个MyEthernetPacket类 epacket然后根据type字段进行分支：

* “ip”：利用p实例化一个MyIPPacket类ipacket，调用display方法展示

信息然后根据protocol字段进行分支：

* + “tcp”：实例化一个MyTCPPacket类t，调用display方法进行展示。
  + “udp”：实例化一个MyUDPPacket类u，调用display方法进行展示。
  + 其他协议：输出"This is a/an "+ipacket.protocol+" packet but there is no more information "。
* “arp”：实例化一个MyARPPacket类a，调用display方法进行展示。
* 其他字段：输出"This is a/an "+epacket.type+" type but there is no more information "

最后输出数据部分，调用bytesToHex以16进制字符串输出。

定义抓包程序启动类Class Capture

属性：

|  |  |
| --- | --- |
| JpcapCaptor captor | Jpcap主要功能类，负责数据包的抓取 |
| Int len = 1600; | 最大包的的长度，略大于以太网MTU |
| NetworkInterface[] devices | 网卡接口列表 |
| Packet p ; | 原始数据包 |

方法：

|  |  |
| --- | --- |
| Capture(int dev\_id,int cnt, String filter) | 实现对数据包的抓取。 |
| static void main(String args[]) | 启动类入口，负责接收参数传送给启动类构造方法。 |

Main方法逻辑：

实例化一个key\_inScanner类接受用户输出的参数用户按照：

网卡号‘空格’抓包数量‘空格’抓取包的类型

进行输入。抓取包的类型可以不指定即抓取所有的包。抓包数量为-1时一直抓取直到用户在终端终止程序。获得参数之后判断是否合法，非法时提示用户重新输入然后再次判断。得到合法输入之后传递给Capture构造方法开始抓包。

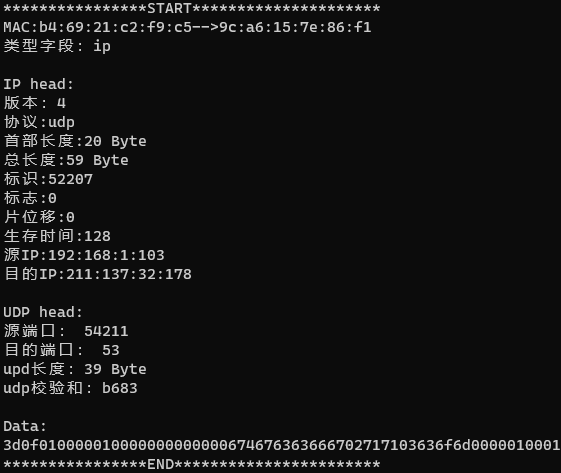
Capture逻辑：

接受三个参数：dev\_id网卡号、cnt包数量、filter过滤器

* 调用JpcapCaptor.getDeviceList()方法得到设备列表然后打开指定网卡同时设置网卡的工作模式为混杂模式。
* 调用captor.setFilter(filter, true)方法设置过滤器。若filter为空则接受所有包。
* 实例化MyPacketHandler接口为handler
* 调用captor.loopPacket(cnt, handler)开始抓包。



图表 8 TCP数据包



图表 9 UDP报文



图表 10 ARP报文

第四章 全文总结与展望

4.1 全文总结

本文以winPcap的Java实现Jpcap为基础，完成了原始数据包的抓取，定义各层协议的数据结构完成了原始数据包的解析实现了一个简单的演示程序。

用户需要先运行name.jar获得网卡信息确定主要抓包的网卡编号。然后运行Capture抓包入口开始抓包按照指定格式输出参数就可以获得每一个数据包从链路层到传输层的所有信息。

4.2 后续工作展望

* 本程序的健壮性还有待提高，在不同版本的java编译之后可能会出现无法运行的情况。
* 功能过于单一，只有抓包和展示信息的功能。未来可能发送数据包的功能。
* 解析的协议较少，只加入了几个主流协议的定义，还有许多协议没有定义

会出现无法解读的情况。未来需要加入更多协议的定义丰富程序。

* 命令行操作不够方便，可以考虑通过GUI来实现参数的传递和输出展示，使得程序实用起来更加方便。

致 谢

本论文的工作是在我的导师王勇老师悉心指导下完成的 ，感谢王勇老师的耐心指导。

参考文献

1. 陈鸣（译）计算机网络：自顶向下方法[M].北京: 机械工业出版社 2009