

本科生毕业设计

|  |
| --- |
| 基于QT的网络数据包分析软件 |

|  |  |
| --- | --- |
| 院 系 | **计算机科学与技术** |
| 专业班级 | **计科201408班** |
| 姓 名 | **陈延良** |
| 学 号 | **U201414782** |
| 指导教师 | **周正勇** |
| 2018年06月10日 | |

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘 要

当今随着信息化发展，网络安全问题日渐突出，网络攻击日益泛滥，因此网络信息安全显得尤为重要。网络主机必须有足够的安全措施，否则网络的价值就会贬值。网络数据包分析，已不是一个新鲜课题，网络管理员能够通过对网络流量采样分析找到网络中存在的问题，而一个优秀的网络数据包分析软件就是网络管理员的强力工具，本毕业设计为开发一款基于Qt的网络数据包分析软件，设计一款能够读取pcap文件，将其中的网络数据包内容显示出来，并进行统计分析，以图表的形式展示给用户。从而能够在一定程度上帮助网络管理人员及时地发现和排除网络异常，提高网络可靠性的监控和保障能力。本次设计借助了libpcap函数库，在Qt的框架下进行编程。设计最终实现了对TCP、UDP、ICMP、ARP、HTTP、DNS、DHCP等协议的支持。

网络数据包分析软件在以后日益复杂的网络中还将持续发挥着其不可替代的作用，也将随着网络的发展不断完善，能够更加精确地分析并定义数据包，让网络管理员能够轻易地区别哪些为正常数据包，哪些为异常数据包。

**关键词**：网络数据包分析；网络协议解析；pcap文件格式；Qt界面开发；libpcap函数库

Abstract

Analysis of network packets.

This graduation design is a network packet analysis software based on Qt framework. Design can read pcap file, and display the content of network packets. The data packets are then statistically analyzed and presented to users in the form of charts.

**Keywords:** Network packet analysis, Network protocol, Pcap file format, Qt interface development, libpcap library function

目 录

[摘 要 I](#_Toc513803305)

[Abstract II](#_Toc513803306)

[1 绪 论 1](#_Toc513803307)

[1.1 课题研究的背景和意义 1](#_Toc513803308)

[1.2 国内外研究现状 2](#_Toc513803309)

[1.3 课题要求 3](#_Toc513803310)

[1.4 研究方案 4](#_Toc513803311)

[1.5 论文结构 4](#_Toc513803312)

[2 网络数据包介绍 6](#_Toc513803313)

[2.1 pcap文件类型 6](#_Toc513803314)

[2.2 网络数据包 9](#_Toc513803315)

[2.3 本章小结 24](#_Toc513803316)

[3 系统设计与实现 25](#_Toc513803317)

[3.1 系统结构 25](#_Toc513803318)

[3.2 数据结构 25](#_Toc513803319)

[3.3 主要函数 27](#_Toc513803320)

[3.4 实现细节 28](#_Toc513803321)

[3.5 本章小结 30](#_Toc513803322)

[4 性能测试与分析 33](#_Toc513803323)

[4.1 测试环境 33](#_Toc513803324)

[4.2 运行测试 33](#_Toc513803325)

[4.3 本章小结 34](#_Toc513803326)

[5 总结与展望 35](#_Toc513803327)

[致 谢 36](#_Toc513803328)

[参考文献 37](#_Toc513803329)

[附录：英文缩写 40](#_Toc513803330)

# 绪 论

## 课题研究的背景和意义

随着计算机网络的持续发展，全球信息化已经逐渐成为当今社会发展的必然趋势。但是由于计算机网络自身所特具有的特点，比如联结形式的多样性和网络的开放性、互连性等特征，所以导致网络易受诸如黑客和一些不法分子还有一些病毒的攻击。所以对于网络信息的安全和保密一直是一个至关重要的问题。不管是对于军用的自动化指挥网络还是银行等传输敏感数据的计算机网络而言，其网上信息的安全和保密尤为重要。因此，网络必须要有足够强的安全措施，否则该网络将是个无用的、甚至危及国家安全的网络。在计算机网络的世界里，存在着许多潜在的威胁，因此网络的安全措施应能够全方位地应对各种不同的威胁，

计算机网络所面临的威胁大体可分为两种：一是对网络中信息的威胁；二是对网络中设备的威胁。影响计算机网络的因素有很多，有些因素可能是有意的，又有可能是无意的；有些可能是人为的，也可能是非人为的；有些可能是外来黑客对网络系统的资源非法占有，归结起来，针对网络安全的威胁主要有三种：

1. 人为的无意失误：如操作员安全配置不当所造成的安全漏洞，用户安全意识不强，用户口令选择的不慎，用户将自己的账号随意转借他人或与别人共享等都会对网络安全带来威胁。
2. 人为的恶意攻击：这是计算机网络所面临的最大威胁，敌手的攻击和计算机犯罪就属于这一类。此类攻击又可分为以下两种：一种是主动攻击，它以各种方式有选择地破坏信息的有效性和完整性；另一类是被动攻击，它是在不影响网络正常工作的情况下，进行截获、窃取、破译以获得重要机密信息。这两种攻击均可对计算机网络造成极大的危害，并导致机密数据的泄漏。
3. 网络软件的漏洞和“后门”：网络软件不可能是百分之百的无缺陷和无漏洞的，然而这些漏洞和缺陷恰恰是黑客进行攻击的首选目标，曾经出现过黑客攻入网络内部的事件，这些事件的大部分就是因为安全措施不完善所招致的苦果。另外软件的“后门”都是软件公司的设计编程人员为了自便而设置的，一般不为外人所知，但一旦“后门”洞开，其造成的后果不堪设想。

为了及早发现并制止网络上的各种攻击，我们需要通过对网路上的数据进行分析来发现并找出问题，提前预防。这也是本此课题的一个重要目的。网络安全管理员运用网络封包截获技术，抓取网络中有用的数据包，然后通过对数据包内容进行分析，确定哪些是有害的或者含有攻击企图的包，以此来达到对网络攻击的预防。同时许多防火墙也是基于包过滤技术的，本文将介绍网络数据包协议分析程序的工作原理以及它的实现。

网络数据包协议分析程序是一种用于收集网络中有用数据程序，这些数据可以是用户的账号密码，也可以是一些商用机密数据等。它是利用计算机的网络接口截获目的地为其他计算机的数据报文的一种工具。

网络数据包协议分析程序的正当用处主要是分析网络的流量，以便找出所关心的网络中潜在的问题。例如，假设网络的某一段运行得不是很好，报文的发送比较慢，而我们又不知道问题出在什么地方，此时就可以用网络数据包协议分析程序来作出精确的问题判断。在合理的网络中，网络数据包协议分析程序的存在对系统管理员是至关重要的，系统管理员通过网络数据包协议分析程序可以诊断出大量不可见模糊问题，这些问题涉及两台乃至多台计算机之间的异常通讯、有些甚至牵涉到各种协议，借助网络数据包协议分析程序系统管理员可以方便确定出多少的通讯量属于哪个网络协议、占主要通讯协议的主机是哪一台、大多数通讯的目的地是哪台主机、报文发送占用多少时间、或者相互主机的报文传送间隔时间等待，这些信息为管理员判断网络问题、管理网络区域提供了非常宝贵的信息。

## 国内外研究现状

现在国内外已经有相当多的成熟且功能强大的网络数据包协议分析软件。比较著名的网络数据包分析软件有：wireshark、TcpDump、EtherPeek等。

Wireshark：Wireshark是一个开放源码的网络分析系统，也是目前最好的使用较多的开放源码的网络协议分析器，支持linux和windows平台。Wireshark起初由Gerald Combs开发，随后由一个松散的Wireshark团队组织进行维护开发。它目前所提供的强大的协议分析功能完全可以媲美商业的网络分析系统。自从1998年发布最早的0.2版本至今，大量的志愿者为Wireshark添加新的协议解析器，如今Wireshark已经支持五百多种协议解析。很难想象如此多的人开发的代码可以很好的融入系统中；并且在系统中加入一个新的协议解析器很简单，一个不了解系统的结构的新手也可以根据留出的接口进行自己的协议开发。这些都归功于Wireshark良好的设计结构。事实上由于网络上各种协议种类繁多，各种新的协议层出不穷，一个好的协议分析器必需有很好的可扩展性和结构。这样才能适应网络发展的需要不断加入新的协议解析器。

TcpDump：顾名思义，TcpDump可以将网络中传送的数据包的“头”完全截获下来提供分析，它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来帮助你去掉无用的信息。用尽量简单的话来定义TcpDump就是：dump the trafiic on a network，根据使用者的定义对网络上数据包进行截获的包分析工具，TcpDump以其强大的功能，灵活的截取策略，成为每个高级的系统管理员分析网络，排查问题等所必备的工具之一。

EtherPeek：这个工具软件开始只是一个网络分析器型的数据包监测软件，经过这些年的发展已经成为一个真正的网络管理工具并具有网络监视和分析等新的功能，被美国联邦调查局用来追踪逃犯、贩卖毒品的人、电脑黑客和一些被怀疑为外国间谍的人。是一个直观，功能强大的以太网网络和协议分析器。支持Macintosh和Windows平台。EtherPeek把查找和修复多平台上的复杂网络任务变得简单化。EtherPeek采用工业标准，非常容易使用，提供解码、过滤和诊断网络的功能。以友好图形界面出名，EtherPeek提供非常详细且多样化的网络使用信息，网络结点的会话和数据包内容。在有问题的局域网络中使用EtherPeek执行一个自定的诊断测试，监控网络的通信和事件，跟踪非法的网络活动，测试和调试网络软硬件。

## 课题要求

使用libpcap库+Qt编程设计一个能够解析pcap文件类型的网络数据包分析软件，能够显示出数据包内容并对其进行分析统计。具体要求如下：

1. 能够读取pcapa文件，并能将其中数据包解析出来。
2. 能够支持几种常见的协议类型，包括网络层、运输层和应用层。
3. 具有数据包内容查看功能，能够将文件中所包含的数据包展示出来。
4. 具有统计分析功能，能够将文件中不同类型的数据包分类，并解析。
5. 具有良好的用户界面，便于用户直观的使用网络数据包分析软件。

## 研究方案

在本次课题的研究中，网络数据包分析软件实现对pcap文件格式类型的网络数据包解析和统计功能，完成一些常用协议的识别和分析：要求至少实现TCP/IP协议簇几个基本协议的分析——ARP、TCP、UDP、ICMP以及一些应用层的常用协议如DHCP、DNS、HTTP等。

## 论文结构

本文共分为五个章节：

第一章，首先介绍了一下网络数据包分析的研究背景，研究现状，介绍了当下比较主流的几个网络分析软件，然后根据需求设计了研究方案，以及要最终达到的预期目标。

第二章，根据课题需要，我们首先介绍了一下网络数据包的分类，以及不同类型数据包的结构，以及协议的一些详细实现细节。

第三章，在整个系统的构建介绍完成之后，开始介绍一些系统操作的设计与实现，比如如何对协议进行判断，如何对协议中不同长度的数据包进行解析，如何对文件中的不同类型数据包进行统计等。

第四章，我们将会进行实验测试结果的展示，主要包括软件的完整性和正确性，对预期支持协议类型的测试，对pcap数据包的实际解析测试，以及用户界面的良好性测试。

第五章，我们将对整个项目进行总结，总结其已经实现的所有功能和达到的性能以及展望项目未来的发展前景和研究方向。

# 网络数据包介绍

在这一章本文将介绍各种不同网络数据包的结构以及保存网络数据包的pcap文件类型。

## pcap文件类型

pcap文件格式是一种常用的网络数据包存储格式，或者说是一种数据流格式，包括wireshark、tcpdump在内的很多软件可以直接把网络数据流转换为这种格式。在Linux里面，pcap可以说时一种通用的数据流格式，很多开源的项目都需要用到这种格式的文件。下面对pcap格式进行详细解析。

Pcap格式结构如下：

文件头 24字节

数据包头 + 数据包（数据包头为16字节，后面紧跟数据报）

数据包头 + 数据包

… …

下面对各个部分进行详细介绍：

* Global Header（pcap文件头 共24byte）

整个数据流文件，只会有一个global header，它定义了本文件的读取规则、最大存储长度限制等内容；

* + Magic（4byte）：标记文件的开始，并用来识别文件自己的字节顺序。0xa1b2c3d4用来表示按照原来的顺序读取，0xd4c3b2a1表示下面的字节都要交换顺序读取。考虑到计算机内存的存储结构，一般会采用0xd4c3b2a1，即所有字节都需要交换顺序读取。
  + Major（2byte）：当前文件主要的版本号，一般为0x0200
  + Minor（2byte）：当前文件次要的版本号，一般为0x0400
  + ThisZone（4byte）：当地的标准时间，如果同的是GMT则全是0，一般都直接写0000 0000
  + SigFigs（4byte）：时间戳的精度，设置为全零即可
  + SnapLen（4byte）：最大的存储长度，如果想把整个包抓下来，设置为0xffff0000，但一般来说0xff7f0000就足够了
  + LinkType（4byte）：链路类型，常用类型有以下几种：

0 BSD loopback devices,except for later OpenBSD

1 Ethernet,and Linux loopback devices

6 802.5 Token Ring

7 ARCnet

8 SLIP

9 PPP

10 FDDI

100 LLC/SNAP-encapsulated ATM

101 “raw IP”,with no link

102 BSD/OS SLIP

103 BSD/OS PPP

104 Cisco HDLC

105 802.11

108 later OpenBSD loopback devices(with the AF\_value in network byte order)

113 special Linux “cooker” capture

114 LocalTalk

* Packet Header（数据包头 共16byte）

Packet Header可以有很多，每个packet header后面会紧跟着一串packet data，packet header定义了packet data的长度、时间戳等信息。

* + Timestamp：被捕获时间戳高位，单位是seconds
  + Timestamp：被捕获时间戳地位，单位是microseconds
  + Caplen：当前数据区的长度，即抓取到的数据帧长度，不包括packet header本身的长度，单位是byte，由此可以得到下一个数据帧的位置
  + Len：离线数据长度：网络中实际数据帧的长度，一般不大于caplen，多数情况下和caplen数值相等。
* Pcaket Data（共Caplen Byte）

包头之后就是数据包的数据了，数据长度就是Caplen个Byte，在这之后就是一个新的Packet Header，定义一个新的Packet Data属性，再接一个新的Packet Data，如此循环。

也就是说pcap文件中没有规定捕获packet数据包之间的间隔字符，下一组的数据包起始是由上一个包的packet header中caplen决定的。以下是对pcap文件的详细结构图：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | | 2 | 3 | 4 | 5 | 6 | | | 7 | | 8 | 9 | A | | B | C | D | | E | F |
| **Pcap文件头（24字节）** | | | | | | | | | | | | | | | | | | | | | |
| **Pcap文件头** | | | | | | | | | | | **pcap数据包头（16字节）** | | | | | | | | | | |
| **Pcap数据包头** | | | | | | | | | | | **MAC以太网帧头（14字节）** | | | | | | | | | | |
| **MAC头** | | | | | | | | | **IP数据报头（一般20字节）** | | | | | | | | | | | | |
| **IP头** | | | | | | | | | | | | | | | **TCP头** | | | | | | |
| **TCP报文段头（一般20字节）** | | | | | | | | | | | | | | | | | | | **数据域** | | |
| **数据域** | | | | | | | | | | | | | | | | | | | | | |
| **Pcap数据包头（16字节** | | | | | | | | | | | | | | | | | | | | | |
| **MAC以太网帧头（14字节）** | | | | | | | | | | | | | | | | | | | **IP头** | | |
| **IP数据报头（一般20字节）** | | | | | | | | | | | | | | | | | | | | | |
| **IP头** | | **TCP报文段头（一般20字节）** | | | | | | | | | | | | | | | | | | | |
| **TCP头** | | | | | | | | **数据域** | | | | | | | | | | | | | |
| **数据域……** | | | | | | | | | | | | | | | | | | | | | |

图3-1 pcap文件格式

## 网络数据包

### 以太网帧

在链路类型中，通常为以太网类型，即LinkType = 1;

下面介绍一些以太网链路层的帧头，即MAC头：

* 目的MAC地址（共6byte）
* 源MAC地址（共6byte）
* 上层协议类型（16bit）
  + IPv4协议类型，代码为0x0800
  + ARP协议类型，代码为0x0806
  + IPv6协议类型，代码为0x86DD
  + PPPoE协议类型，代码为0x8864
  + 802.1Q tag协议类型，代码为0x8100

### IP数据报

IP协议是TCP／IP协议族中最为核心的协议。它提供不可靠，无连接的服务，也即依赖其他层的协议进行差错控制。在局域网环境，IP协议往往被封装在以太网帧中传送。而所有的TCP，UDP，ICMP，IGMP数据都被封装在IP数据报中传送。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| IP报头（20字节） | | | | | | | | | | | | | | | | |
|  | | | IP数据报 | | | | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | |

IP数据报头一般为20字节，如下：

* 版本信息（4 bit）；

用来表明IP协议实现的版本号，当前一般为IPv4，即0100

* 头长度（4 bit）；

是头部占32比特的数字，包括可选项。普通IP数据报，没有任何选项的，该字段的值为5，即160比特=20字节。此字段最大值为60字节。

* 服务类型（8 bit）；

其中前3比特为优先权字段（Precedence，现已被忽略）。第8比特保留未用。第4至7比特分别代表延迟，吞吐量，可靠性和花费。他们取值为1时分别代表要求最小延时，最大吞吐量，最高可靠性和最小费用。这4比特的服务类型中只能置其中1比特为1。可以全为0。服务类型字段声明了数据报被网络系统传输时可以被怎样处理。例如：TELNET协议可能要求最小的延迟，FTP协议可能要求有最大吞吐量，SNMP协议可能要求有最高可靠性，NNTP可能要求最小费用，而ICMP协议可能无特殊要求。大部分主机会忽略这个字段，但一些动态路由如OSPF，IS-IS可以根据这些字段进行理由决策。

* 数据包长度（16 bit）；

指明整个数据报的长度（以字节为单位）。最大长度为65535字节。

* 数据包标识（16 bit）；

用来唯一标识主机发送的每一份数据报。通常每发一份报文，他的值会加1。

* 分片使用，标志（3 bit）+片偏移（13 bit）；

前3比特标志一份数据是否要求分段。后13比特指明该段偏移距原始数据报开始的位置。

* 存活时间（8 bit）；

用来设置数据报最多可以经过的路由器数。由发送数据的源主机设置，通常为32，64，128等，每经过一个路由器，其值减1，直到0时该数据报会被丢弃。

* 协议类型（8 bit）；

指明IP层所封装的上层协议类型，如ICMP（1），IGMP（2），TCP（6），UDP（17）等。

* 首部校验和（16 bit）；

内容是根据IP头计算得到的校验和码。计算方法是，对头部中两个16比特进行二进制反码求和。（和ICMP，IGMP，TCP，UDP不同，IP不对头部后的数据进行校验）。

* 源IP地址（32 bit）；
* 目的IP地址（32 bit）；
* 选项（如果有的话，32 bit）；

用来定义一些任选项：如记录路径，时间戳等，这些选项很少被使用，同时并不是所有主机和路由器都支持这些选项。可选字段的长度必须是32比特的整数倍。如果不足必须填充0以达到此长度要求。

### ARP报文

前14字节为以太网首部，对于以太网首部来说，如果是请求报文的话，以太网目的地址为ff:ff:ff:ff:ff:ff;即广播报，帧类型的话ARP对应的值为0x0806

硬件类型表明ARP协议实现在那种情况类型网络上，值为1即表示以太网地址，协议类型表示解析协议（上层协议），一般为0800，即IP，硬件地址长度，也就是MAC地址长度，即6字节，协议地址长度，也就是IP地址长度，即4字节，操作类型（OP），表示ARP协议数据报类型，1表示ARP请求包，2表示ARP应答包，3表示RARP请求包，4表示RAPR应答包，然后发送以太网地址就是源MAC地址，发送端IP地址也就是源IP地址，目的以太网地址就是目的MAC地址，如果是请求报文的话为00:00:00:00:00:00，目的端IP地址就是目的端IP地址。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| ARP报文头（20字节） | | | | | | | | | | | | | | | | |
|  | | | ARP报文 | | | | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | |

ARP（Address Resolution Protocol）即地址解析协议，是根据IP地址获取物理地址的一个TCP／IP协议。在网络中ARP请求报文通常是以广播的形式发送到局域网中，请求一个IP地址对应的物理地址，其数据包格式如下

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | | 4 | | 5 | 6 | 7 | 8 | 9 | a | | b | c | | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | | | | | HType | |
| PType | | | Hl | | Pl | | Op | | 发送端MAC地址 | | | | | | | | 发送端IP地址 | | | |
| 目的MAC地址 | | | | | | | | | 目的IP地址 | | | | |

* 硬件类型

硬件地址不只以太网一种，如果是以太网类型，此时值为1

* 协议类型

表示要映射的协议地址的类型，要对IPv4地址进行映射，此时值为0x0800

* 硬件地址长度

以太网类型地址长度对应6字节

* 协议地址长度

IPv4地址类型对应长度4字节

* 操作类型

操作类型字段，值为1，表示进行ARP请求；值为2，表示进行ARP应答；值为3，表示进行RARP请求；值为4，表示进行RARP应答。

* 发送端以太网地址

发送ARP请求或应答的硬件地址，这里是以太网地址，和MAC头的源地址相同。

* 发送端IP地址

发送ARP请求或应答的IP地址

* 目的以太网地址

目的以太网地址，请求时通常为广播，即为00:00:00:00:00:00。

* 目的IP地址

目的IP地址，请求中为请求MAC的IP地址，应答时为请求的源IP地址

### TCP报文段

TCP报文段头一般为20字节，如下：

* 源端口号（16 bit）；
* 目的端口号（16 bit）；
* 序列号（32 bit）；
* 确认号（32 bit）；
* TCP头长度（4 bit）；
* 保留部分（6 bit）；
* 标志位（6 bit）；
* 窗口大小（16 bit）；
* 校验和（16 bit）；
* 紧急指针（16 bit）；
* 选项（如果有）；

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | |
|  | | | TCP报文段头 | | | | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | |

### UDP报文段

UDP报文段头为8字节，如下：

* 源端口号（16 bit）；
* 目的端口号（16 bit）；
* 数据包长度（16 bit）；
* 校验和（16 bit）；

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | |
|  | | | UDP报文段头 | | | | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | |

### ICMP报文

ICMP（Internet Control Message Protocol）即网络控制信息协议，协议是IP协议的补充，用于IP层的差错报告，拥塞控制，路径控制以及路由器或主机信息的获取。ICMP与IP协议位于同一层（IP层），而ICMP报文是封装在IP数据包的数据部分进行传输的，不同于TCP和UDP，他并不传输用户数据，但对于用户数据的传递起者至关重要的作用。如下位ICMP的报文结构

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | | |
|  | | | ICMP固定报文头（8字节） | | | | | | | | | ICMP数据部分 | | | | | |
| …… | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| IP报头（20字节） | | | | | | | | | | | | | | | | |
|  | | | ICMP报文 | | | | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | |

ICMP分很多类，总的来看分为两类1）通知错误原因的差错报文 2）请求应答的查询报文

ICMP报文格式及类型，通用格式，前8字节固定

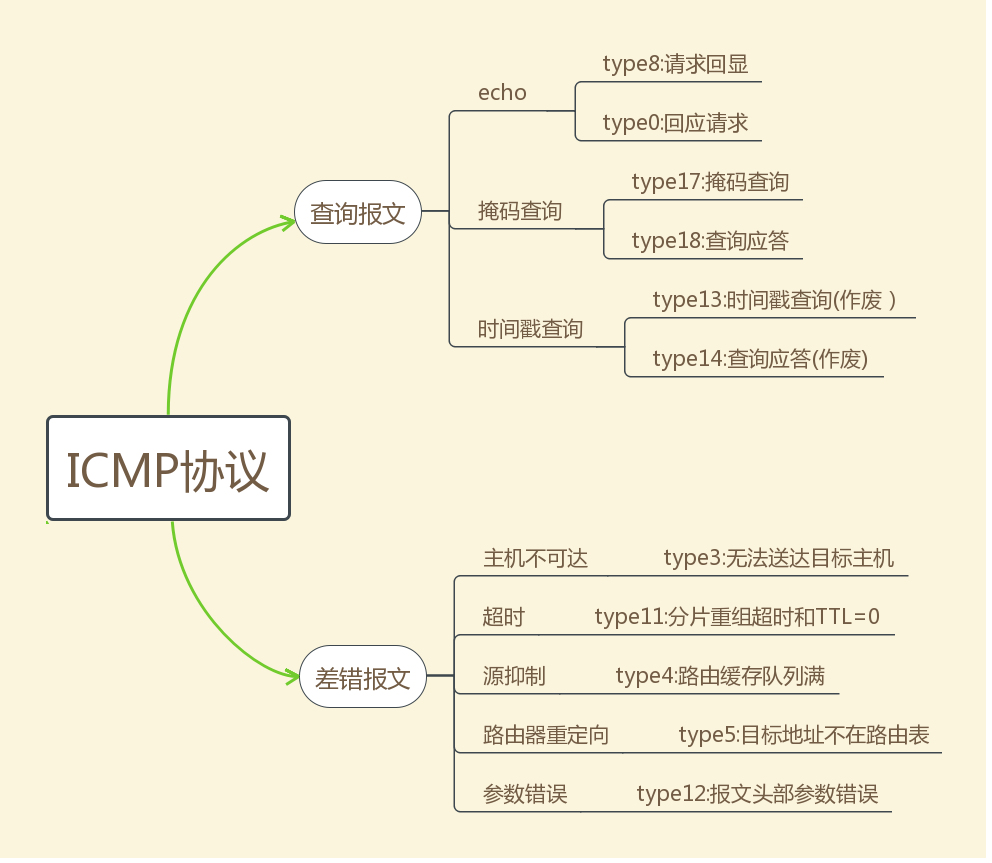
|  |  |  |
| --- | --- | --- |
| 类型 | 代码 | 校验和 |
| （这4字节取决于ICMP报文类型） | | |
| ICMP数据部分 | | |

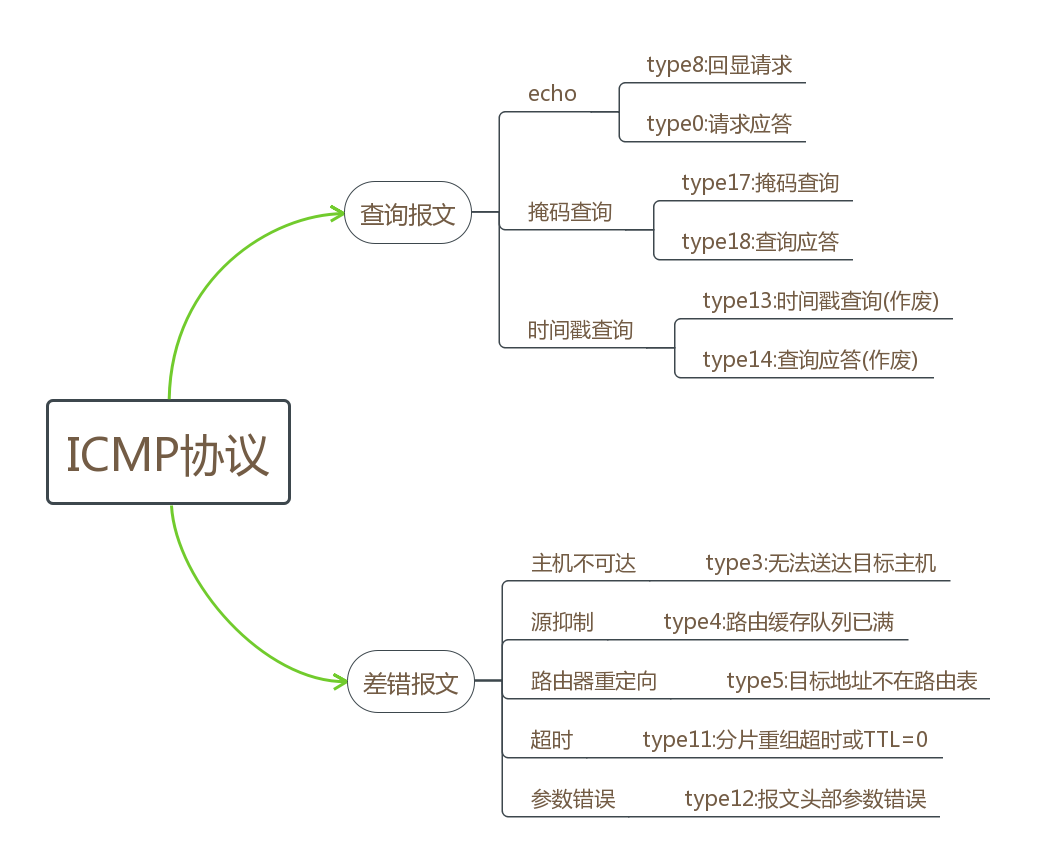
对于类型目的不可达（3），超时（11），参数错误报文（12），源抑制（4），重定向（5）：

|  |  |  |
| --- | --- | --- |
| 类型 | 代码 | 校验和 |
| 未用（全为0） | | |
| IP数据报头（包括选项）+IP数据报原始数据字段前8字节 | | |

对于类型Echo(8、10)、信息请求或信息响应，：

|  |  |  |
| --- | --- | --- |
| 类型 | 代码 | 校验和 |
| 标识符 | | 序列号 |
| ICMP数据部分 | | |





常见类型：

* 回显请求（0）：ping工具通过发送ICMP回显消息检查特定节点的IPv4连接已排查网络问题。
* 回显应答（8）：节点发送回显答复消息响应ICMP回显消息。
* 重定向（5）：路由器发送“重定向”消息，告诉发送主机到目标IPv4地址更好的路由。
* 源抑制（4）：路由器发送“源结束”消息，告诉发送主机以较低的频率发送数据报，以免因为路由器拥塞而导致数据报被丢弃。
* 超时（11）：当超过IP生存期时和分段IP数据报没有在期限内重新组合都会发送这种消息。
* 无法达到目标（3）：路由器和目标主机通知发送主机数据无法传送。

类型大致分为差错报文，查询报文。

查询： 0：回送应答

8：回送请求

9：路由器公告

10：路由器请求

17：地址掩码

18：地址掩码应答

差错： 3：目标不可达

4：源抑制

5：重定向

11：超时

差错报文：

当发送一份差错报文时，报文始终包含IP的首部和产生ICMP差错报文的IP数据报的前8字节，这样接受iCMP差错报文的模块就会吧它与某个特定协议和用户进程联系起来（分别是IP头中协议字段和IP数据报前8字节中的端口号来判断）

### DNS报文

DNS报文一般包含在UDP报文段中。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | | |
|  | | | UDP报文段头（8字节） | | | | | | | | | DNS报文 | | | | | |
| …… | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | |  | |
| IP数据报报头（20字节） | | | | | | | | | | | | | | | | | |
|  | | | UDP报文段头（8字节） | | | | | | | | | DNS报文 | | | | | |
| …… | | | | | | | | | | | | | | | | | |

××DNS报文格式及类型

|  |  |
| --- | --- |
| Transaction ID（会话标识） | Flags(标识) |
| Questions(问题数) | Answer RRs(回答 资源记录数) |
| Authority RRs(授权 资源记录数) | Additional RRs(附加资源记录数) |
| Name(查询名，长度不固定) | |
| Type(查询类型) | Class（查询类） |
| Name(查询名，长度不固定) | |
| Type(查询类型) | Class（查询类） |
| Time to live(生存时间) | |
| Data lenth(资源数据长度) | Data |
| Data（资源数据，长度不固定） | |
| Name(查询名，长度不固定) | |
| Type(查询类型) | Class（查询类） |
| Time to live(生存时间) | |
| Data lenth(资源数据长度) | Data |
| Data（资源数据，长度不固定） | |
| Name(查询名，长度不固定) | |
| Type(查询类型) | Class（查询类） |
| Time to live(生存时间) | |
| Data lenth(资源数据长度) | Data |
| Data（资源数据，长度不固定） | |

* 头部(蓝色区域)
  + - 回话标识（2字节）：DNS报文的ID标识，对于请求报文和其对应的应答报文，这个字段是相同的，通过它可以区分DNS应答报文是哪个请求的响应。
    - 标志（2字节）：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| QR | Opcode | AA | TC | RD | RA | (Zero) | rcode |

QR：查询/响应标志，0为查询，1为响应

Opcode：0表示标准查询，1表示反向查询，2表服务器状态请求。

AA：表示授权回答

TC：表示可截断的

RD：表示期望递归

RA：表示可用递归

Rcode：表返回码，0表没差错，3表名字差错，2表服务器错误。

* + - 数量字段

Questions、Answer RRs、Authority RRs、Additional RRs各自表示后面的四个区域的数目。Questions表示查询问题区域节的数量，Answer RRs表示回答区域的数量，Authority RRs表示授权区域的数量，Additional RRs表示附加区域的数量。

* 查询问题区域（绿色区域）
  + - 查询名：长度不固定，且不使用填充字节，一般该字段表示的就是需要查询的域名，一般格式如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | h | u | b | 4 | h | u | s | t | 3 | e | d | u | 2 | c | n | 0 |

数字表明后面的字节数，hub前为数字3,hust前为4,edu前为3，cn前为2，最后结尾必须是0,所以域名为hub.hust.edu.cn。

* + - 查询类型：

表3-1 DNS查询类型

|  |  |  |
| --- | --- | --- |
| 类型 | 助记符 | 说明 |
| 1 | A | 由域名获得IPv4地址 |
| 2 | NS | 查询域名服务器 |
| 5 | CNAME | 查询规范名称 |
| 6 | SOA | 开始授权 |
| 11 | WKS | 熟知服务 |
| 12 | PTR | 把IP地址转换为域名 |
| 13 | HINFO | 主机信息 |
| 15 | MX | 邮件交换 |
| 28 | AAAA | 由域名获得IPv6地址 |
| 252 | AXFR | 传送整个区的请求 |
| 255 | ANY | 对所有记录的请求 |

* + - 查询类

通常为1，表明是internet数据

* 回答区域（紫色区域）
  + - 域名

格式与Queries区域查询名字字段一样，有一点不同的是，当域名重复出现时，该字段使用2字节的便宜指针表示，具体格式时，前面两个最高位为11.用于识别指针，其余14位表示从DNS报文开始出计数，指出字段在报文中相应偏移字节数。典型的一个列子就是c00c(11000000 00001100)12正好时头部长度，正好指向Queries区域的查询名字字段。

* + - 查询类型

同上，见表格

* + - 查询类

对于internet信息，总是IN

* + - 生存时间

以秒为单位，表示的是资源记录的生命周期，一般用于当地地址解析程序取出资源记录后决定保存及使用缓存数据的时间，它同时也可表示该资源的稳定程度，极为稳定的信息会被分配一个很大的指，如84600，这是一天的秒数。

* + - 资源数据

该字段也是一个变长字段，表示按照查询段的要求返回的相关资源记录，例如如果查询报文想要的回应是一个IP地址，这里就对应一个IP地址。

* 授权区域（黄色区域）

同上回答区域

* 附加区域（橙色区域）

同上回答区域

### DHCP报文

DHCP报文通常是通过UDP报文段传输的。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | | |
|  | | | UDP报文段头（8字节） | | | | | | | | | DHCP报文 | | | | | |
| …… | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | |  | |
| IP报头（20字节） | | | | | | | | | | | | | | | | |
|  | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

DHCP协议

|  |  |  |  |
| --- | --- | --- | --- |
| 报文类型（1B） | 硬件类型（1B） | 硬件地址长（1） | 跳数（1B） |
| 事务ID（4B） | | | |
| 秒数（2B） | | 标志位（2B） | |
| 客户端IP地址（4B） | | | |
| 你的IP地址（4B） | | | |
| 服务器IP地址（4B） | | | |
| 网关IP地址（4B） | | | |
| 客户机硬件地址（16B） | | | |
| 服务器主机名（64B） | | | |
| 启动文件名（128B） | | | |
| 选项（变长） | | | |

* 报文类型

报文类型码，只能取值1和2 ，请求包request为1，应答包reply为2

* 硬件类型

硬件类型代码，1代表千兆以太网

* 硬件地址长度

单位为字节，表明硬件地址长度，以太网对应值为6

* 跳数

DHCP跨网段使用时由reply agent使用，DHCP client发出的包此字段清零。

* 事务ID

值为DHCP client发送数据包时填充的一个随机数，用来在client和server之间表示一组request和reply。

* 秒数

由客户端填充，表示客户端开始请求分配IP或请求更新IP租期所经历的时间。

* 标志

标志字段，目前只使用左边最高位的bit，其余bit必须由client发包时清零，用来表示自己想从server或是reply agent那里得到的回复是unicase类型还是braodcast类型。

* 客户端IP地址

只有在client拥有IP，能够响应ARP请求时，并且client端处于BOUND，RENEW，或者RWBINDING状态时才会被client填充

* 你的IP地址

DHCP server分配给client的IP地址。

* 客户端IP地址

DHCP流程的下一个服务器地址，由DHCP server在DHCPOFFER和DHCPACK保证返回。

* 网关IP地址

DHCP的中继IP地址，当跨网段使用DHCP时，该字段由一个relay agent填充，并且每个agent都将hops加一，通常hops值有一个限制，超过此限制，该包将会被丢弃。

* 客户端硬件地址

占十六个字节

* 服务器名称

占64字节，可选的DHCP server的Hostname,是一个以null结尾的字符串。

* 文件名

启动文件名，128字节是一个以null结尾的字符串，当主机从网络启动时会使用。

* 选项

可选参数列表，长度不固定。

### HTTP报文

HTTP报文通常通过TCP报文段来进行传输。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| MAC帧头（14字节） | | | | | | | | | | | | | | | |  | |
| IP数据报头（20字节） | | | | | | | | | | | | | | | | | |
|  | | | TCP报文段头（一般20字节） | | | | | | | | | | | | | | |
|  | | | | | | | HTTP报文 | | | | | | | | | | |
| …… | | | | | | | | | | | | | | | | | |

## 本章小结

这一章主要是对网络中常见数据包类型进行详细介绍，这些协议类型也是本次网络数据包分析系统设计的基础。协议主要包括TCP/IP协议簇几个基本协议的分析——ARP、TCP、UDP、ICMP以及一些应用层的常用协议如DHCP、DNS、HTTP等。本章对各个协议有详细的协议实现细节，而在实现过程中需要对各个协议的数据结构，数据包长度以及高层协议的类型等都要有比较深的理解

# 系统设计与实现

## 系统结构

整个网络数据包分析工具采用模块化的设计思想，原因是许多程序太长或太复杂，很难写在单一单元中。如果把代码分为较小的功能单元，这样将大大简化编程过程。模块化程序一般比单独程序容易编写、调试和修改。所以本系统采用模块化的设计思想。该网络数据包分析工具主要分三个模块：界面显示模块，数据包解析存储模块，数据包分析统计模块。接下来我们将对这三个模块的关系和各自的功能进行详细介绍。

### 界面显示模块

界面模块主要负责将数据显示给用户，并提供数据输入功能。在整个程序中，界面模块与其他两个模块密切合作。原则是尽力合理的显示数据包的有用信息。

### 数据包解析存储模块

数据包分离模块主要负责读取pcap文件后将其中的数据包解析分离出来，并保存在对应的数据结构中。该模块主要是程序一开始对文件读取成功后进行调用该模块。主要是根据数据包的协议包头进行数据信息的解析，并作出适当的解释，最后根据类型存储到相应数据结构中，以便其他模块读取。

### 数据包分析统计模块

数据包分析统计模块主要负责对每一个数据包的解析以及对整个pcap文件中的数据包进行统计。本模块就是对捕获的数据包按照数据链路层（MAC）、网络层（IP，ARP/RARP）、传输层（TCP，UDP，ICMP）和应用层（HTTP，DNS，DHCP等）的层次结构自底向上进行解析，并将解析结果显示输出，在进行协议解析时要注意各个协议报头多字节整形字段从网络字节序到主机字节序的转换。对于多字节的数据（如short、int、long等），不同CPU在其内部表示一个数时会使用不同字节顺序方案，将低序字节存储在起始地址的称为小段存储，否则称为大端存储。网络中有不同类型的计算机，在网络数据的传输中要统一数字的表示方式，即统一采用大端存储，所以数据包头中表示长度或类型等的多字节数据字段应按需转换为本地主机的表示形式。可以用ntohs()或ntohl()将双字节或四字节整型由网络字节序转换为主机字节序。

## 数据结构

主要包括一些头部数据结构，如PCAP文件头、PCAP数据包头、MAC头，IP头，TCP头，UDP头，DHCP头，DNS头，ICMP头，ARP头等。下面逐一介绍各个头部的结构体。

* 首先是一个pcap文件中只存在一个的PCAP文件头结构体，共24字节

typedef struct pcapFileHeader{

uchar8\_t magic[4]; //读取顺序(4字节)

uint16\_t version\_major; //主要版本号(2字节)

uint16\_t version\_minor; //次要版本号(2字节)

int32\_t thiszone; //时区修正(4字节)

uint32\_t sigfigs; //精准时间戳(4字节)

uint32\_t snaplen; //抓包最大长度(4字节)

uint32\_t linktype; //链路类型(4字节)

}pcapFileHeader\_t;

* 每个数据包都包含的pcap数据包头，共16字节，主要包含数据包的一些基本信息

typedef struct ts{

uint32\_t ts\_sec;//秒数(4字节)

uint32\_t ts\_usec;//毫秒数(4字节)

}ts\_t;

typedef struct pcapPktHeader{

ts\_t ts;

uint32\_t caplen; //数据包长度(4字节)

uint32\_t len; //文件数据包长度(4字节)

}pcapPktHeader\_t;

* 每个以太网帧中都包含的MAC头

typedef struct EtherHeader{

uchar8\_t DstMAC[6]; //目的MAC地址(6字节)

uchar8\_t SrcMAC[6]; //源MAC地址(6字节)

uint16\_t FrameType; //帧类型(2字节)

}EtherHeader\_t;

typedef struct EtherTail{

uint32\_t Checksum; //数据帧尾校验和(4字节)

}EtherTail\_t;

* IP数据包中都包含的IP头，本文主要介绍IPv4数据包，共20字节

typedef struct IPHeader{

uchar8\_t Ver\_Hlen; //版本+报头长度(1字节)

uchar8\_t TOS; //服务类型(1字节)

uint16\_t TotalLen; //总长度(2字节)

uint16\_t ID; //标识(2字节)

uint16\_t Flag\_Segment; //标志(3bit)+片偏移(13bit)(2字节)

uchar8\_t TTL; //生存周期(1字节)

uchar8\_t Protocol; //协议类型(1字节)

uint16\_t Checksum; //头部校验和(2字节)

uint32\_t SrcIP; //源IP地址(4字节)

uint32\_t DstIP; //目的IP地址(4字节)

}IPHeader\_t;

* TCP报文段头(20字节)，还包括TCP可选部分

typedef struct TCPHeader{

uint16\_t SrcPort; //源端口(2字节)

uint16\_t DstPort; //目的端口(2字节)

uint32\_t SeqNum; //序号(4字节)

uint32\_t AckNum; //确认号(4字节)

uchar8\_t HeaderLen; //数据报头的长度(4bit)+保留(4bit)(1字节)

uchar8\_t Flags; //保留(2bit)+(6bit)标识TCP不同的控制信息(1字节)

uint16\_t Window; //窗口大小(2字节)

uint16\_t Checksum; //校验和(2字节)

uint16\_t UrgentPoint; //紧急指针(2字节)

}TCPHeader\_t;

typedef struct TCPOptions{

char8\_t m\_ckind;

char8\_t m\_cLength;

char8\_t m\_cContext[];

}TCPOption\_t;

* UDP报文段头(8字节）

typedef struct UDPHeader{

uint16\_t SrcPort; //源端口号(2字节)

uint16\_t DstPort; //目的端口号(2字节)

uint16\_t SegLen; //报文段长度(2字节)

uint16\_t Checksum; //校验和(2字节)

}UDPHeader\_t;

* ICMP头部(4字节）

typedef struct ICMPHeader{

uchar8\_t ICMPType; //类型(1字节)

uchar8\_t ICMPCode; //代码(1字节)

uint16\_t Checksum; //校验和(2字节)

}ICMPHeader\_t;

* ICMP报文(应答包)

typedef struct ICMPType0{

uint16\_t ICMPID; //2字节

uint16\_t ICMPSeq; //2字节

uchar8\_t ICMPData[];

}ICMPType0\_t;

* ICMP报文(目的地不可达)

typedef struct ICMPType3{

uint16\_t ICMPPmvoid; //2字节

uint16\_t ICMPNextmtu; //2字节

IPHeader\_t IPH;

uchar8\_t IPData[8];

uchar8\_t AddData[];

}ICMPType3\_t;

* ICMP报文(请求包)

typedef struct ICMPType8{

uint16\_t ICMPID; //2字节

uint16\_t ICMPSeq; //2字节

uchar8\_t ICMPData[];

}ICMPType8\_t;

* ICMP报文(超时)

typedef struct ICMPType11{

uint32\_t ICMPVoid; //4字节

IPHeader\_t IPH;

uchar8\_t IPData[8];

uchar8\_t AddData[];

}ICMPType11\_t;

* ARP头

typedef struct ARPHeader{

uint16\_t ARPHrd;

uint16\_t ARPPro;

uchar8\_t ARPHLn;

uchar8\_t ARPPln;

uint16\_t ARPOP;

}ARPHeader\_t;

* DNS报文头部(长度固定部分)

typedef struct DNSHeader{

uint16\_t ID; //标识

uint16\_t Flags; //标志，逐bit分析

uint16\_t Qst\_num; //问题数，通常为1

uint16\_t Rsc\_num; //资源记录数，查询报文中以下三个都为0

uint16\_t Aut\_num; //授权资料记录数

uint16\_t Adt\_num; //额外资源记录数

uchar8\_t data[];

}DNSHeader\_t;

* DHCP报文头部(长度固定部分)

typedef struct DHCPHeader{

uchar8\_t OP; //报文类型，requset:1 reply:2

uchar8\_t HrdType; //硬件类型代码 以太网:1

uchar8\_t HrdLen; //硬件地址长度 以太网:6

uchar8\_t Hops; //跳数

uint32\_t XID; //事务ID 随机数

uint16\_t Sec; //秒数

uint16\_t Flags; //标志，只使用最高位bit

uint32\_t Ciaddr; //客户端IP地址

uint32\_t Yiaddr; //你的IP地址

uint32\_t Siaddr; //服务器IP地址

uint32\_t Giaddr; //网关IP地址

uchar8\_t Chaddr[16]; //客户端硬件地址

uchar8\_t Sname[64]; //服务器名

uchar8\_t File[128]; //引导文件名

}DHCPHeader\_t;

接下来介绍程序中存储整个pcap文件中数据的几个链表

typedef struct ICMP\_List{

struct ICMP\_List \*next;

int Seq\_num;

pcapPktHeader\_t pkthdr;

EtherHeader\_t etherhdr;

IPHeader\_t iphdr;

ICMPHeader\_t icmphdr;

uchar8\_t data[];

}ICMP\_List\_t;

typedef struct TCP\_List{

struct TCP\_List \*next;

int Seq\_num;

pcapPktHeader\_t pkthdr;

EtherHeader\_t etherhdr;

IPHeader\_t iphdr;

TCPHeader\_t tcphdr;

uchar8\_t data[];

}TCP\_List\_t;

typedef struct UDP\_List{

struct UDP\_List \*next;

int Seq\_num;

pcapPktHeader\_t pkthdr;

EtherHeader\_t etherhdr;

IPHeader\_t iphdr;

UDPHeader\_t udphdr;

uchar8\_t data[];

}UDP\_List\_t;

typedef struct ARP\_List{

struct ARP\_List \*next;

int Seq\_num;

pcapPktHeader\_t pkthdr;

EtherHeader\_t etherhdr;

ARPHeader\_t arphdr;

uchar8\_t data[];

}ARP\_List\_t;

//总表（包含所有表的表头指针）

typedef struct All\_list\_hdr{

int icmp\_num;

int tcp\_num;

int udp\_num;

int arp\_num;

ICMP\_List\_t \*icmp\_listhdr;

TCP\_List\_t \*tcp\_listhdr;

UDP\_List\_t \*udp\_listhdr;

ARP\_List\_t \*arp\_listhdr;

}All\_list\_hdr\_t;

## 主要函数

本节主要介绍主要函数。首先是pcap文件读取函数

pcap\_loop();

其中数据包处理函数为一个回调函数，每次循环读取数据包后都会调用该函数。

函数的处理流程如下：

在介绍

## 实现细节

实现细节，在程序中

## 本章小结

本章对网络数据包分析软件各个组成部分的设计与实现进行了详细的论述。首先介绍了界面模块，数据包分离模块，数据包分析统计模块三个模块的具体设计与实现。分析了在运行过程中可能遇到的种种情况并且综合各种情况选择最有利的设计方案。然后。本章对网络数据包分析软件各个组成部分的设计与实现进行了详细的论述。首先介绍了系统的总体结构，每个模块之间的合作与分工。然后介绍了整个程序用到的数据结构，包括数组、结构体和链表等。还介绍了程序的一些主要函数实现，最后介绍了一些具体实现细节与特点，如何合理运用Qt程序设计方法来实现程序预期要达到的效果。

# 性能测试与分析

## 测试环境

测试硬件环境如下：

系统：MacOS 10.13.3

处理器：2.3 GHz Intel Core i5

内存：8GB 2133 MHz LPDDR3

软件环境如下：

Qt：Qt 5.10.0（Clang 7.0（Apple）,64 bit）

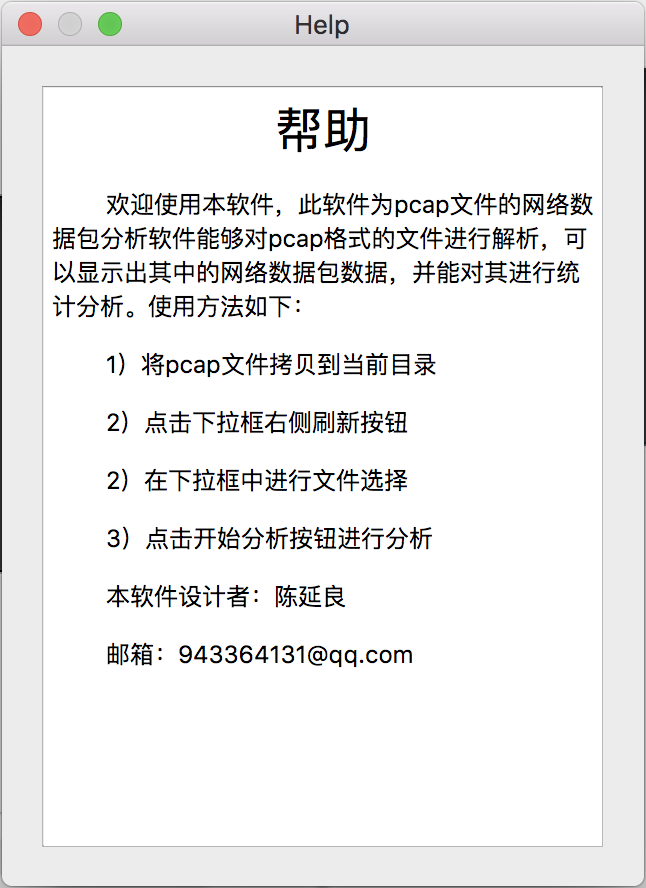
QtCreator：Qt Creator 4.5.0

WireShark：Wireshark Version 2.4.3

## 运行测试

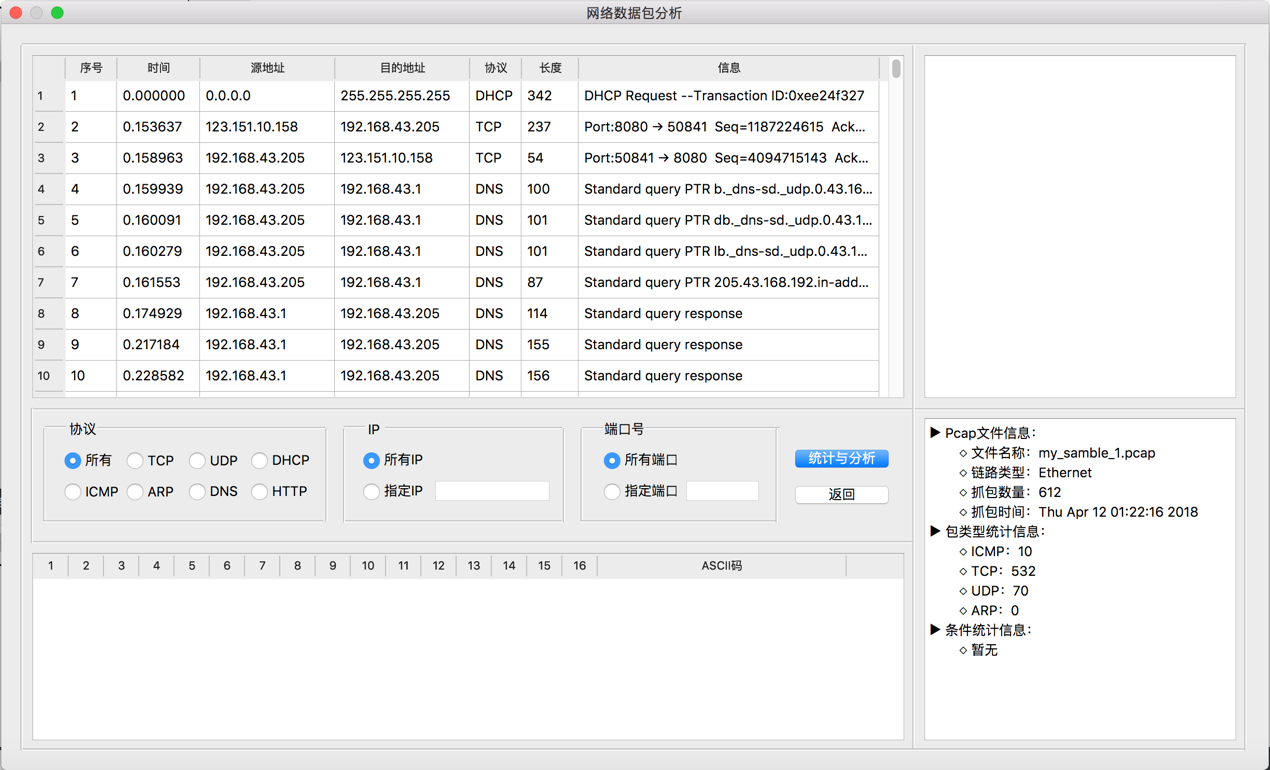
我们在通过测试一个包含有多种类型数据包的pcap文件来检验系统的功能。

首先在程序的初始界面选择该pcap文件，如图

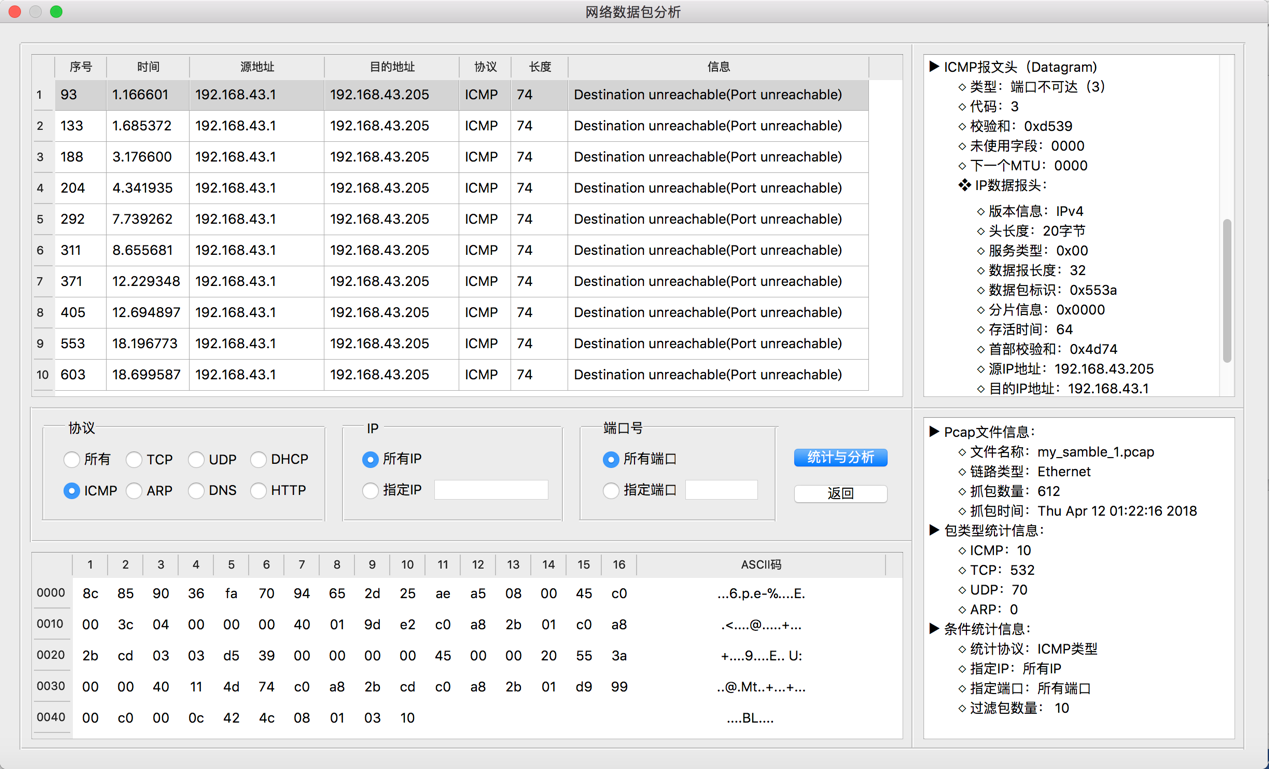


（图 初始文件选择界面）

然后打开该文件的总揽界面如下。

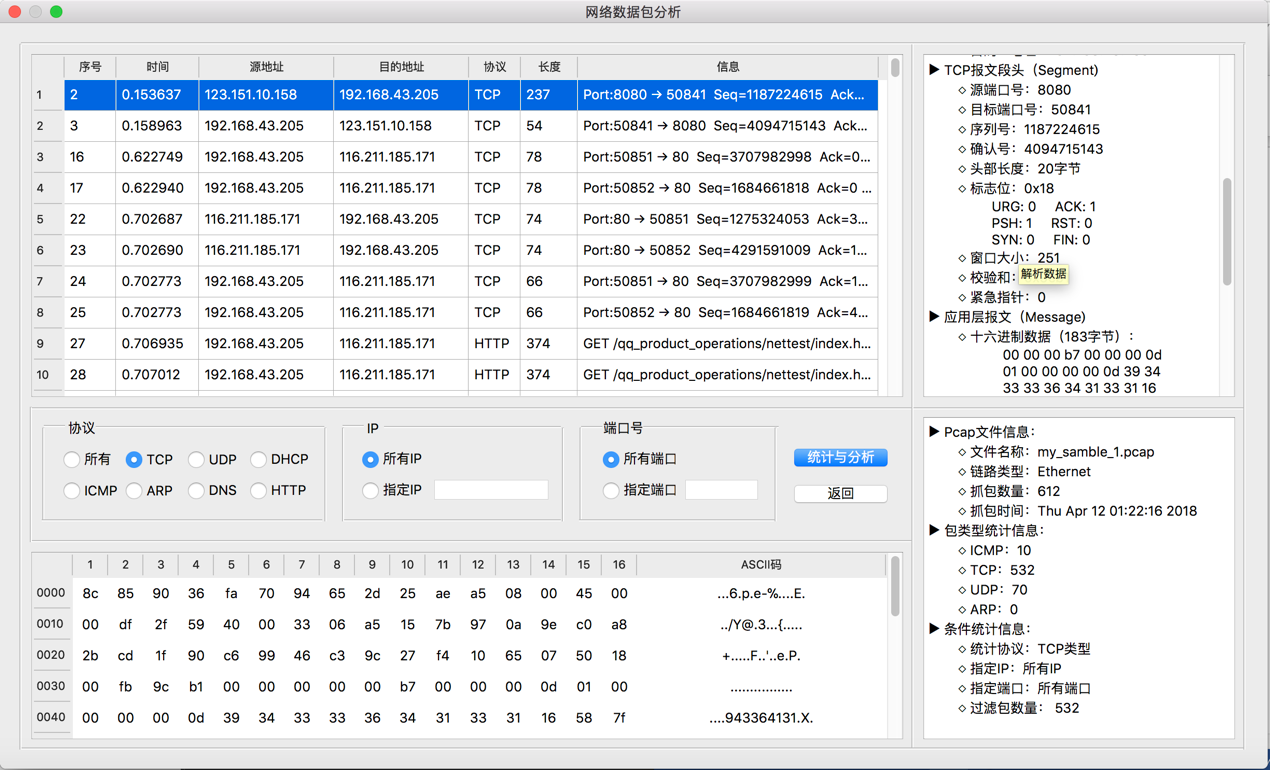


对ICMP类型数据包进行测试：



（图 ICMP）

对TCP包



（图 TCP）

对HTTP包

（图 HHTP）

对 UDP包

（图 UDP）

对 DNS包

（图 DNS）

对 DHCP包

（图 DHCP）

对ARP包

（图 ARP）

## 本章小结

本章通过对不同类型的数据包进行解析测试。展示了系统对多个协议的支持情况。目前系统主要支持一下协议：ICMP、TCP、UDP、ARP等，同时在应用层协议有HTTP、DNS、DHCP等。系统通过表格的形式将数据包显示出来，然后通过点击表格的数据包条目，在文本框中进一步显示详细包信息，同时在下方将当前数据包的二进制数据显示给用户。

# 总结与展望

在这几个月的设计与实现中，网络数据包分析软件基本实现了预期的要求：

1. 能够读取pcap文件类型的数据包文件，能够正常解析出文件中信息。
2. 实现对多个协议类型的解析，目前支持网络层支持对ARP、IP等协议解析，运输层支持对ICMP、TCP、UDP等协议解析，应用层支持对HTTP、DNS、DHCP等协议解析。
3. 能够将支持对pcap文件中所有数据包进行分类统计，并对统计分析结果呈现给用户。
4. 能够通过协议、IP地址、端口号对数据包进行过滤，得到特定的数据包。
5. 用户界面相对友好，用户能够方便地对数据包进行概览，选择特定的数据包类型进行分析查看，得到有用的数据。

当然，由于时间有限，接下来讨论一些本课题还有一些目前存在的问题和一些日后可进一步完善的地方。

1. 支持协议不完全，对于一个普通网络适配器正常情况下所捕捉流量尚不能完全解析出来。
2. 尚不具备对异常流量的分类以及分析功能。
3. 只能对协议、端口号、IP地址进行过滤，不具备对MAC地址等其他筛选条件的过滤功能。
4. 尚未实现以图的形式显示统计结果的功能。

本次设计的网络数据包分析软件在对HTTP包进行解析时可能会对错误地处理。这个功能我们也打算在以后的研究中实现。

致 谢

整个毕业设计从题目的选择到最后实现所有的功能用了大概四个月左右的时间，在这段时间里，我得到了很多的帮助，也从中学到了很多。

首先要感谢的是我的导师周正勇老师，在整个设计过程中，周正勇老师给予我无私的帮助，为我提供了种种专业知识上的指导，和一些富于创造性的建议，在我遇到问题，苦于无法解决时，老师总是能及时给予我细心而专业的回答，在后期答辩之前，周老师也仔细浏览了我的开题报告，文献翻译等内容并提出了许多珍贵的修改意见，周老师一丝不苟的作风，严谨求实的态度使我深受感动。正是在周老师的帮助下，我得以顺利的完成此次毕业设计。在此向周正勇老师表示深深的感谢和崇高的敬意！

其次，在临近毕业之际，我还要借此机会感谢这四年中给予我诸多教诲和帮助的各位老师们，是你们的教诲让我对计算机领域有了大致的了解并为以后进行更深入的研究打下了夯实的基础。感谢他们四年来的辛勤栽培，不积跬步何以致千里，各位任课老师认真负责，在他们悉心的帮助和支持下，我能够很好的掌握和运用专业知识，并在设计中得以体现，顺利完成毕业论文。

此外，我还要感谢各位同学和我的各位室友，在毕业时即这段时间里，你们给予了我很多启发，提出了许多宝贵的意见。当然，除了在学习上的互帮互助，还有在生活上的互相照顾。对于你们的帮助和支持，在此我表示深深的感谢！

最后，感谢母校，感谢学院。学院为我们提供了丰富的学习资源与优秀的学习环境，让我们在一个良好的氛围中度过了这充实的大学四年。

参考文献

1. Palkar S, Lan C, Han S, et al. E2: a framework for NFV applications[C]. Symposium on Operating Systems Principles. ACM, 2015:121-136.
2. Bremler-Barr A, Harchol Y, Hay D. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions[C]. Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:511-524.
3. Sekar V, Egi N, Ratnasamy S, et al. Design and implementation of a consolidated middlebox architecture[C]. Usenix Conference on Networked Systems Design and Implementation. 2011:24-24.
4. Anderson J W, Braud R, Kapoor R, et al. xOMB: extensible open middleboxes with commodity servers[C]. Eighth ACM/IEEE Symposium on Architectures for NETWORKING and Communications Systems. ACM, 2012:49-60.
5. A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual Middleboxes as First-class Entities[R]. Technical report, UW-Madison 2012.
6. Zhang W, Liu G, Zhang W, et al. OpenNetVM: A Platform for High Performance Network Service Chains[C]. The Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM, 2016:26-31.
7. Gember-Jacobson A, Viswanathan R, Prakash C, et al. OpenNF: enabling innovation in network function control[C]. ACM Conference on SIGCOMM. ACM, 2015:163-174.
8. Rajagopalan S, Dan W, Jamjoom H, et al. Split/merge: system support for elastic execution in virtual middleboxes[C]. Usenix Conference on Networked Systems Design and Implementation. 2013:227-240.
9. J. Khalid, A. Gember-Jacobson, R. Michael, A. Abhashkumar, and A. Akella. Paving the Way for NFV: Simplifying Middlebox Modifications Using StateAlyzr[C]. In Proc. of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI’16), 2016.
10. Ballani H, Costa P, Gkantsidis C, et al. Enabling End-Host Network Functions[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4):493-507.
11. Paxson V. Bro: a system for detecting network intruders in real-time[C]. Conference on Usenix Security Symposium. USENIX Association, 1998:3-3.
12. Yoan N. Iptables[M]. Miss Press, 2013.
13. Sherry J, Gao P X, Basu S, et al. Rollback-Recovery for Middleboxes[C]. ACM Conference on Special Interest Group on Data Communication. ACM, 2015:227-240.
14. Chrome G. HTTP persistent connection[J]. 2015.
15. Surhone L M, Tennoe M T, Henssonow S F, et al. Ffmpeg[M]. 2010.
16. Housley R, Hoffman P. Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP[J]. Ietf Rfc Sri Network Information, 1999, 11(3):82--89.
17. Maheshwari A, Sharma A, Ramamritham K, et al. TranSquid :Transcoding and Caching Proxy for Heterogenous E-Commerce Environments[C]. International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/e-Business Systems. IEEE Computer Society, 2002:50.
18. J. Khalid, A. Gember-Jacobson, R. Michael, A. Abhashkumar, and A. Akella. Paving the Way for NFV: Simplifying Middlebox Modifications Using StateAlyzr[C]. In Proc. of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI’16), 2016.
19. NFV Paper [EB/OL].https://portal.etsi.org/NFV/NFV White Paper2.pdf,2017-05-05.
20. Hwang J, Ramakrishnan K K, Wood T. NetVM: high performance and flexible networking using virtualization on commodity platforms[C]. Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014:445-458.
21. S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy. SoftNIC: A Software NIC to Augment Hardware[R]. Technical report, EECS Department, University of California, Berkeley, 2015..
22. Martins J, Ahmed M, Raiciu C, et al. ClickOS and the art of network function virtualization[C]. Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014:459-473.
23. Panda A, Han S, Jang K, et al. NetBricks: taking the V out of NFV[C]. Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2016:203-216.
24. A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual Middleboxes as First-class Entities[R]. Technical report, UW-Madison 2012.
25. Palkar S, Lan C, Han S, et al. E2: a framework for NFV applications[C]. Symposium on Operating Systems Principles. ACM, 2015:121-136.
26. Rajagopalan S, Dan W, Jamjoom H. Pico replication:a high availability framework for middleboxes[C]. Symposium on Cloud Computing. 2013:1-15.
27. Sherry J, Gao P X, Basu S, et al. Rollback-Recovery for Middleboxes[C]. ACM Conference on Special Interest Group on Data Communication. ACM, 2015:227-240.
28. Qazi Z A, Tu C C, Chiang L, et al. SIMPLE-fying middlebox policy enforcement using SDN[J]. Computer Communication Review, 2013, 43(4):27-38.
29. Zhang W, Hwang J, Rajagopalan S, et al. Flurries: Countless Fine-Grained NFs for Flexible Per-Flow Customization[C]. International on Conference on Emerging NETWORKING Experiments and Technologies. ACM, 2016:3-17.
30. Bremler-Barr A, Harchol Y, Hay D. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions[C]. Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:511-524.

附录：英文缩写

|  |  |  |
| --- | --- | --- |
| 英文缩写 | 英文全称 | 中文全称 |
| IP | Internet Protocol | 因特网互联协议 |
| MAC | Media Access Control | 媒体访问控制（硬件地址） |
| ICMP | Internet Control Message Protocol | 因特网控制报文协议 |
| TCP | Transport Control Protocol | 传输控制协议 |
| UDP | User Datagram Protocol | 用户数据报协议 |
| ARP | Address Resolution Protocol | 地址解析协议 |
| HTTP | HyperText Transfer Protocol | 超文本传输协议 |
| DNS | Domain Name System | 域名系统 |
| DHCP | Dynamic Host Configuration Protocol | 动态主机配置协议 |

**成 绩 评 定**

**指导教师评定意见**

一、对毕业设计（论文）的学术评语（应具体、确切、实事求是）

|  |
| --- |
| 陈延良同学学习态度评价…工作时间进度评价…论文内容与价值…论文结构评价…论文规范性评价…作者学术水平评价…，论文（基本）达到本科毕业设计论文的要求，同意参加本科毕业答辩。  导师评语建议学生先写个草稿，由导师审阅论文时一并修订成型，最终评语可以直接打印，导师签字即可，）  （答辩前导师需要给出具体成绩，下表评分中合计部分更新域可自动运算求和） |

二、对毕业设计评分

(1)理工医科评分表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 调研论证  (10分) | 外文翻译  (5分) | 设计(论文)撰写质量  (10分) | 学习态度  (10分) | 基本理论和基本技能  (50分) | 创 新  (15分) | 合 计  (100分) |
| 得分 | **8** | **4** | **8** | **8** | **40** | **12** | **80** |

(2)文科评分表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 文献阅读与文献综述  (10分) | 外文翻译  (10分) | 论文撰写质量  (10分) | 学习态度  (10分) | 学术水平、论证能力和创新  (60分) | 合 计  (100分) |
| 得分 |  |  |  |  |  |  |

指导教师签字：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日

**答辩小组评定意见**

一、评语（根据学生答辩情况及其设计（论文）质量综合评价）

|  |
| --- |
|  |

二、评分

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 答 辩 情 况 | | 论 文 质 量 | | 合 计  (100分) |
| 答辩情况  (15分) | 回答问题情况  (25分) | 规范要求与文字表达  (20分) | 学术水平  (40分) |
| 得分 |  |  |  |  |  |

答辩小组长签字：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日

**毕业答辩及成绩评定说明**

1. 毕业答辩
2. 答辩前，答辩小组应详细审阅每个学生的毕业设计（论文），为答辩做好准备。
3. 严肃认真组织答辩，开好答辩会。
4. 指导教师应参加所指导学生的答辩会，但评定其成绩时宜回避。
5. 答辩中要做好记录以供成绩评定时参考。
6. 成绩评定
7. 答辩前每个学生都要将自己的毕业设计（论文）在指定时间内交给指导教师，由指导教师审阅，写出评语并预评分。
8. 答辩工作结束后，答辩小组应举行专门会议进行讨论，在参考指导教师预评结果的基础上，结合学生毕业设计（论文）质量和学生答辩情况，综合评定每个学生的成绩。
9. 院（系）对专业答辩小组提出的优秀和不及格的毕业设计（论文），要组织院（系）级答辩，最终确定成绩，并向学生公布。
10. 各专业学生的最后成绩应符合正态分布规律。
11. 请用蓝、黑钢笔手写或五号宋体字编辑，签名须手写，A4纸双面打印。

**毕业设计（论文）成绩评定**

|  |
| --- |
| 班号：**计科201408班** 学生姓名：**陈延良**  综合成绩：\_\_\_\_\_\_**86**\_\_\_\_\_\_\_分（折合等级\_\_\_\_\_**良**\_\_\_\_\_\_\_）  答辩小组长（签名）：\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日 |