西安邮电大学
毕业设计（论文）

题目：基于安卓 VPN 技术的移动端抓包软件设计与开发

学院：计算机学院

专业：软件工程

班级：

学生姓名：

学号：

导师姓名：　　职称：

起止时间：2018 年 11 月 19 日 至 2019 年 6 月 7 日

毕业设计（论文）声明书

本人所提交的毕业论文《基于安卓VPN技术的移动端抓包软件设计与开发》是本人在指导教师指导下独立研究、写作的成果，论文中所引用他人的文献、数据、图件、资料均已明确标注；对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明并表示感谢。

本人完全理解《西安邮电大学本科毕业设计（论文）管理办法》的各项规定并自愿遵守。

本人深知本声明书的法律责任，违规后果由本人承担。

论文作者签名：

日期：　　年　月　日

西安邮电大学本科毕业设计(论文) 选题审批表

<table>
<tr><td>申 报 人</td><td></td><td>职 称</td><td></td><td>学　　院</td><td colspan="2">计算机学院</td></tr>
<tr><td>题目名称</td><td colspan="6">基于安卓 VPN 技术的移动端抓包软件设计与开发</td></tr>
<tr><td>题目来源</td><td>科研</td><td></td><td>教学</td><td></td><td>其它</td><td>✓</td></tr>
<tr><td>题目类型</td><td>软件系统研发</td><td>✓</td><td colspan="2">软件产品设计</td><td>软件技术研究</td><td></td></tr>
<tr><td>题目
简述</td><td colspan="6">在对涉及到网络通信的软件代码进行调试时，经常需要通过抓包的方式来得到网络发送或接收到的实际内容。目前对安卓平台上的软件进行抓包通常采用计算机抓包软件(Fiddler、Charles 等)配合同一局域网内的移动端设置系统代理来实现。此种方式进行抓包配置较为繁琐，只支持 HTTP/HTTPS 数据的抓取，而且要求移动设备和计算机必须在同一局域网内。如果能够在移动端不借助其他设备的情况下进行抓包，那么通信软件的调试将会极为便利。 本课题旨在实现一个在移动设备端独立运行、无需借助其他设备或软件产品的抓包软件，并且拥有传统 PC 端抓包软件的几乎所有的功能。用户只需开启软件，即可抓取移动设备上所有协议下的网络数据包(没有代理下的 HTTP 限制)，并能够对不同类型的数据进行分析以格式化显示。抓取到的数据包可以保存在本地，进行编辑或再次发送、批量发送。</td></tr>
<tr><td>对学
生知
识与
能力
要求</td><td colspan="6">1、掌握 Java 语言编程。
2、掌握安卓平台上的软件开发知识及常用第三方库。
3、熟练使用 Android Studio 进行开发调试。
4、掌握计算机网络，能够无误的对各种主流网络协议数据进行解析、转换。
5、掌握 Java 并发编程。
6、了解 Java NIO 的原理及使用。</td></tr>
<tr><td>预期
目标</td><td colspan="6">1、软件能够正确、稳定地抓取手机上所有的网络数据。
2、对抓取到的网络数据进行分析并格式化显示。
3、网络数据拦截、修改、批量重发。
4、毕业设计论文</td></tr>
<tr><td>时间
进度</td><td colspan="6">2018.11.19-2018.12.31 确定需求，调研并确定具体的实现方案
2019.1.1-2018.3.31 软件客户端编写并调试，能稳定运行
2019.4.1-2019.4.31 对软件进行性能优化，尽可能减少软件对手机网络的影响
2019.5.1-2019.5.31 软件细节优化，撰写毕业论文
2019.6.1-2019.6.7 准备答辩事宜</td></tr>
<tr><td colspan="2">系（教研室）主任
签字</td><td colspan="2">年　月　日</td><td>主管院长
签字</td><td colspan="2">年　月　日</td></tr>
</table>

西安邮电大学本科毕业设计（论文）开题报告

学生姓名		学号		专业班级	
指导教师		题目	基于安卓 VPN 技术的移动端抓包软件设计与开发		

选题目的

随着科学技术的一步步提高，现今的软件开发理论及技术都越来越成熟了，同时在人们的认知中，软件测试的重要性也愈发的提高了。通常软件开发整个流程中编码占比是很少的，而更多的资源都消耗在软件测试中[1]。随着互联网技术的发展，现在大多数软件都或多或少含有互联网的成分，此时的软件测试变得更加复杂[2]，对于测试工具的需求也更加迫切。

由于 TCP 协议具有着可靠传输的能力，使用方便等各种优点[3]，TCP 协议已经成为目前使用最为广泛的传输层协议，而随着万维网发展起来的应用层协议 HTTP[4]也是基于 TCP 进行发开出来的。随后为了通信安全的防范，又在现有 HTTP 技术上配合 SSL[6]加密协议衍生出了广为使用的 HTTPS 协议。互联网软件一般都会涉及到多种网络协议的使用，在测试中想要知晓软件在众多网络协议中传输的信息，就需要进行抓包处理，较为经典的抓包软件像 WireShark[6]，能够直接抓取电脑网卡上的所有网络信息,并且能够覆盖到各种主流的网络协议。如果需要对 Android 移动软件进行抓包则需要配合电脑端设置手机代理来执行，通过代理的方式将手机端的流量导向电脑端，再使用电脑端抓包软件来执行抓包操作[7]。传统移动端抓包的方式设置麻烦、需要第三方设备接入，且由于安全原因无法抓取 TCP 数据。

事实上，Android 自身就提供了有代理全局网络的功能，即 VPN。VPN 由于拥有代理全局网络的特性，通常用于网络安全加密，或是使外网访问到一般方式无法访问到的内网[8]。使用 Android 提供的 VpnService[9]接口，能够方便的创建一个 VPN 设备，从而捕获设备全局的包括 IP 协议层的所有数据(可能需要考虑到 IPV6 的使用)[10]，突破传统移动端代理抓包的限制。不过目前业界还没有成熟的 VPN 抓包技术，所以对这方面的研究以及技术积累非常具有意义。

参考文献：

[1] 易敏捷.软件测试国内外发展现状及趋势研究[J].电脑知识与技术,2013,9(26):6020-6022.

[2] 刘泓杉.关于软件测试在 Web 开发中的运用探讨[J].电脑迷,2018(12):68.

[3] Fall, Kevin R., and W. Richard Stevens. TCP/IP illustrated, volume 1: The protocols. addison-Wesley, 2011.

[4] Belshe, Mike, Roberto Peon, and Martin Thomson. Hypertext transfer protocol version 2 (http/2). No. RFC 7540. 2015.

[5] Freier, Alan, Philip Karlton, and Paul Kocher. The secure sockets layer (SSL) protocol version 3.0. No. RFC 6101. 2011.

[6] Sanders, Chris. Practical packet analysis: Using Wireshark to solve real-world network problems. No Starch Press, 2017.
[7] Shah K. Penetration Testing Android Applications[J]. online] Whitepaper, Foundstone, a division of McAfee, 2011.
[8] 蒋月华.VPN 技术的高级应用[J].现代信息科技,2018(12):176-181.
[9] 陈双宝. Android 系统的 VPN 客户端的研究与实现[D].华北电力大学,2013.
[10] Deering, Steve, and Robert Hinden. Internet protocol, version 6 (IPv6) specification[J]. No. RFC 8200. 2017.

前期基础（已学课程、掌握的工具，资料积累、软硬件条件等）

已学课程：《面向对象程序设计(JAVA)》、《数据结构与算法》、《数据库原理》、《操作系统》、《Android 应用开发》、《软件工程》、《软件测试》等

掌握的工具：Android Studio(Android 开发 ide)、Intellij Idea(Java、groovy 开发 ide)、git（版本控制管理工具）、Eclipse（Java 开发 ide）、Fiddler（Windows 端抓包工具）、Charles（mac 端抓包工具）、APKTool（android apk 反编译工具）等

资料积累：Google android 开发官方文档 developer.android.google.cn、《TCP/IP 详解》、《Java 开发规范（阿里）》、《单元测试之道-使用 JUnit(Java 版)》等

软件条件：Android Studio + Android SDK
硬件条件：Android4.0 以上的设备

要研究和解决的问题（做什么）

功能需求：

1 获取手机全局网络请求及响应
2 网络请求的解析，从链路层实体的 ip 数据解析为 tcp 实体数据
3 抓取的网络包存储到本地，并能从本地进行读取
4 网络包以列表形式展示在手机上，支持搜索、过滤
5 网络包详情查看器、编辑器
6 保存的网络包编辑后批量发送

制定网络拦截修改规则，自动将接受包、发送包替换为指定数据

非功能需求：

由于 VPN 拦截了手机设备全局的网络访问，软件应该尽可能的不影响原本的网络传输，以接近原本最高网速。

由于手机设备全局的网络数据量较大，软件应该将实时运行内存控制在一个阈值之内。

3 解析中转网络数据时应尽可能保证中途不出错

4　高效处理大量的异步数据并保证准确性

工作思路和方案（怎么做）

1. 工作思路

需求分析，对将要实现的功能进行细化，文档化。

由于 Android VpnService 接口太过冷门，所以首先对 Android 的 VPN 系统进行全面的了解研究。

详细研究 IP 层、TCP、UDP、HTTP 等协议，能够完全手动的去解析还原这些协议的内容，并能够进行协议之间实体的互相转换。

确定具体的实现方案，并设计整体的系统结构。

基本的编码以及必要的单元测试。

系统地对功能进行测试，并对性能进行评估

2.技术方案

整体的页面展示 + 数据抓取 + 数据处理采用 MVP 结构；页面展示中使用 RecyclerView 以及 ViewPager 等组件实现一个可滑动页面列表，并实时同步悬浮窗展示；数据获取使用 Android 系统提供的 VpnService 服务接口，创建并绑定一个 VPN 设备，从该设备上读取手机设备全局的网络数据；数据处理主要进行网络数据包的解析及转发，转发使用 Java NIO 进行高效地操作，解析逻辑完全自实现。

3.进度计划

2018.11.19 - 2018.12.1　确定需求，

2018.12.1 - 2018.12.31　调研并确定具体的实现方案

2019.1.1 - 2019.2.28　软件客户端编写，能达到大部分需求

2019.3.1 - 2019.3.31　软件客户端完整测试，保证稳定性

2019.4.1 - 2019.4.31　对软件进行性能优化，尽可能减少软件对手机网络的影响

2019.5.1 - 2019.5.31　软件细节优化，撰写毕业论文

2018.6.1 - 2018.6.7　准备答辩事宜

指导教师意见

同学对课题所需的网络制式有较好的了解，对课题有明确的认识，具有开题资格，同意开题。

签字:　　年　月　日

西安邮电大学毕业设计(论文)成绩评定表

<table>
<tr><td>姓名</td><td colspan="2"></td><td>性别</td><td></td><td>学号</td><td colspan="2"></td><td>专业
班级</td><td></td></tr>
<tr><td>课题
名称</td><td colspan="9">基于安卓VPN技术的移动端抓包软件设计与开发</td></tr>
<tr><td rowspan="2">前期
成绩</td><td colspan="2">背景与目标(30)</td><td colspan="2">参考文献(20)</td><td colspan="2">设计方案(30)</td><td colspan="2">撰写质量(20)</td><td>总分</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="2">中期
成绩</td><td colspan="2">完成情况(20)</td><td colspan="2">关键问题(30)</td><td colspan="2">前期问题改进
(20)</td><td colspan="2">方案创新性与合
理性(30)</td><td>总分</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="3">指导
教师
意见</td><td colspan="9">（从项目实现情况、创新性、毕设过程中学生的学习能力、翻译的质量等方面进行考核）

指导教师(签字)：_____ 年 月 日</td></tr>
<tr><td colspan="2">项目论证(50)</td><td colspan="2">创新性(10)</td><td colspan="2">自学能力(30)</td><td colspan="2">译文(10)</td><td>总分（百分制）</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="3">评阅
教师
意见</td><td colspan="9">（从设计方案的合理性、测试设计、论文质量和对社会的影响等方面进行考核）

评阅教师(签字)：_____ 年 月 日</td></tr>
<tr><td colspan="2">设计方案(20)</td><td colspan="2">测试方案(20)</td><td colspan="2">社会影响(10)</td><td colspan="2">撰写质量(50)</td><td>总分（百分制）</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td></tr>
</table>

<table>
<tr><td>验收小组意见</td><td colspan="4">（从设计方案的实现程度、创新性、项目代码完成情况等方面进行考核）

验收教师(签字)：_____ 年 月 日
<table><tr><td>设计方案(40)</td><td>创新(20)</td><td>完成情况(40)</td><td>总分(百分制)</td></tr><tr><td></td><td></td><td></td><td></td></tr></table></td></tr>
<tr><td>答辩小组意见</td><td colspan="4">（从答辩过程体现出的创新意识，项目对社会影响的论述、阐述的项目实现过程、回答问题等方面进行考核）

答辩小组组长(签字)：_____ 年 月 日
<table><tr><td>创新意识(20)</td><td>社会影响(20)</td><td>答辩质量(60)</td><td>总分(百分制)</td></tr><tr><td></td><td></td><td></td><td></td></tr></table></td></tr>
<tr><td>评分比例</td><td colspan="4">前期情况总分(10%) ____ 中期情况总分 (10%) _____ 指导教师评分 (20%) _____
评阅教师评分(25%) ____ 验收小组评分 (25%) _____ 答辩小组评分 (10%) _____</td></tr>
<tr><td>学生总评成绩</td><td>百分制成绩</td><td></td><td>等级制成绩</td><td></td></tr>
<tr><td>答辩委员会意见</td><td colspan="4">毕业论文(设计)最终成绩(等级)：_____

学院答辩委员会主任(签字、学院盖章)： 年 月 日</td></tr>
</table>

摘　要

随着移动互联网的发展，越来越多的应用都或多或少地使用着互联网，纯粹的单机应用已经很少见了，所以对于应用程序网络调试的需求越来越迫切。而 Android 系统的出现使得网络调试不再仅仅局限于 PC 端，此时的 Android 系统也需要像 PC 端一样的抓包程序。

目前常见的 Android 移动端抓包方式均采用了 PC 端抓包软件加上移动端设置代理的方式来进行，但由于此种抓包方式的兼容性存在有一定的问题，而且仅支持基于 HTTP 的网络数据，所以此种方式仍然不是一个最佳的方案。

针对于传统 Android 移动端抓包方存在的问题，本文提出了借助 Android 系统的 VPN 机制来进行抓包的解决方案，并从系统需求、系统设计、系统开发、测试等方面对整个实现过程进行了描述，同时记录了关键的难点问题以及对应的解决方案。对基于 Android VPN 机制的抓包程序的研究与实现，将对移动端的网络调试带来便捷，具有很强的现实意义。

关键词：Android；VPN；抓包；调试

ABSTRACT

With the development of the Internet, more and more applications use the Internet more or less. Applications without internet are rare, so the need for network debugging is becoming more and more urgent. The emergence of the Android makes network debugging no longer limited to the PC. At this time, the Android also needs the same packet capture program as the PC.

At present, the common Android packet capture method uses the PC packet capture software with the mobile devices to set the proxy mode. However, due to the compatibility of this capture method, there are some problems, and only HTTP-based network data is supported. So this approach is still not the best solution.

Aiming at the problem of traditional Android capture method still exists, this article proposes a solution for capturing packets by means of the VPN mechanism of the Android, and describes the whole implementation process which consists of system requirements, system design, system development and testing, and records difficult issues and corresponding solutions. The research and implementation of the packet capture program based on the Android VPN mechanism will bring convenience to the network debugging of the mobile devices and has a practical significance.

Key words： Android; VPN; packet capture; debug

目 录

第一章 绪论 ……1

1.1 课题背景 ……1
1.2 国内外研究现状 ……1
1.3 课题研究目标 ……2
1.4 本文内容安排 ……2

第二章 系统需求 ……3

2.1 业务需求 ……3
2.2 系统功能需求 ……3
2.2.1 数据包抓取 ……4
2.2.2 数据包聚合 ……4
2.2.3 数据包存储和读取 ……5
2.2.4 数据包过滤查看 ……5
2.2.5 查看单个数据包 ……5
2.2.6 数据包抓取历史查看 ……6
2.2.7 数据包单条保存 ……6
2.2.8 数据包转发 ……6
2.2.9 响应数据回写到网卡 ……6
2.3 系统非功能需求 ……7
2.3.1 界面需求 ……7
2.3.2 性能需求 ……7

第三章 系统设计 ……8

3.1 设计决策 ……8
3.2 体系结构设计 ……8
3.2.1 逻辑架构 ……8
3.2.2 开发架构 ……9
3.2.3 物理架构 ……10
3.3 界面接口设计 ……10
3.4 数据存储设计 ……10
3.4.1 数据模型设计 ……11
3.4.2 数据文件设计 ……11
3.5 业务模块设计 ……11
3.5.1 数据包抓取 ……11
3.5.1.1 软件单元构成 ……12
3.5.1.2 执行流程设计 ……12
3.5.2 数据包聚合 ……13

3.5.2.1 软件单元构成 ……13
3.5.2.2 执行流程设计 ……14
3.5.3 数据包存储和读取 ……16
3.5.3.1 软件单元构成 ……16
3.5.3.2 执行流程设计 ……16
3.5.4 数据包过滤查看 ……17
3.5.4.1 软件单元构成 ……17
3.5.4.2 执行流程设计 ……18
3.5.5 查看单个数据包 ……18
3.5.5.1 软件单元构成 ……18
3.5.5.2 执行流程设计 ……20
3.5.6 数据包抓取历史查看 ……21
3.5.6.1 软件单元构成 ……21
3.5.6.2 执行流程设计 ……21
3.5.7 数据包单条保存 ……22
3.5.7.1 软件单元构成 ……22
3.5.7.2 执行流程设计 ……23
3.5.8 数据包转发 ……24
3.5.8.1 软件单元构成 ……24
3.5.8.2 执行流程设计 ……24
3.5.9 响应数据回写到网卡 ……25
3.5.9.1 软件单元构成 ……25
3.5.9.2 执行流程设计 ……26

第四章　系统开发 ……27

4.1 开发环境 ……27
4.2 关键技术 ……27
4.2.1 网卡链路层数据的解析 ……27
4.2.2 数据包的转发 ……28
4.2.3 数据包的回写 ……29
4.3 开发成果 ……30

第五章　系统测试 ……33

5.1 测试设计 ……33
5.1.1 测试环境 ……33
5.1.2 测试范围 ……33
5.2 测试用例及测试记录 ……33
5.3 测试结果及结论 ……34
5.3.1 测试用例执行情况 ……34
5.3.2 软件缺陷分析 ……35
5.3.3 测试结论 ……35

第六章　总结与展望 ……36

6.1 本文工作总结 ……36
6.1 未来工作展望 ……36

结束语 ……37

致　谢 ……38

参考文献 ……39

第一章　绪论

1.1 课题背景

随着科学技术的一步步提高，现今的软件开发理论及技术都越来越成熟了，同时在人们的认知中，软件测试的重要性也愈发的提高了。通常软件开发整个流程中编码占比是很少的，而更多的资源都消耗在软件测试中[1]。随着互联网技术的发展，现在大多数软件都或多或少含有互联网的成分，此时的软件测试变得更加复杂[2]，对于测试工具的需求也更加迫切。

现在的互联网软件一般都会使用一种或多种网络协议来进行数据交互。比如具有着可靠传输的能力，使用方便等各种优点的 TCP 协议[3]，只要软件本身没有什么特殊要求，基本上都会选择使用 TCP 协议（或者基于 TCP 的协议），其已经成为目前使用最为广泛的传输层协议。以及随着万维网发展起来的应用层协议 HTTP[4]，它本身基于 TCP 协议，以其无连接、可靠、使用便捷的特点，成为了几乎各个互联网软件必用的协议。随后为了通信安全的防范，又在现有 HTTP 技术上配合 SSL[5]加密协议衍生出了 HTTPS 协议。

在测试这些互联网软件中想要知晓其在众多网络协议中传输的信息，就需要进行抓包处理，较为经典的抓包软件像 WireShark[6]，能够直接抓取电脑网卡上的所有网络信息，并且能够覆盖到各种主流的网络协议。如果需要对 Android 移动软件进行抓包则需要配合电脑端设置手机代理来执行，通过代理的方式将手机端的流量导向电脑端，再使用电脑端抓包软件来执行抓包操作[7]。传统移动端抓包的方式设置麻烦、需要第三方设备接入，且由于安全原因无法抓取 TCP 数据。

事实上，Android 自身还有另一种代理全局网络的功能，即 VPN。VPN 由于拥有代理全局网络的特性，通常用于网络安全加密，或是使外网访问到一般方式无法访问到的内网[8]。使用 Android 提供的 VpnService[9]接口，能够方便的创建一个 VPN 设备，从而捕获设备全局的包括 IP 协议层的所有数据(可能需要考虑到 IPV6 的使用)[10]，突破传统移动端代理抓包的限制。不过目前业界还没有成熟的 VPN 抓包技术，所以对这方面的研究以及技术积累非常具有意义

1.2 国内外研究现状

目前要对安卓移动端设备进行抓包基本上都是借助 PC 端和代理设置来实现。主流的 PC 端抓包软件如 Windows 上的 Fiddler、Mac 上的 Charles，它们除了可以抓取 PC 端自身的网络数据外，还实现了一个代理的服务器，支持从远程设备进行连接。一旦远程设备连接到了该代理服务器，那么这些抓包软件就能直接获取到远程设备的网络流量。

安卓系统本身提供有设置代理服务器的功能，安卓手机只要将自己的代理指向开启了具有代理功能抓包软件（Fiddler、Charles 等）的 PC 端地址，即可在 PC 端抓包软件上捕获到安卓手机端的网络数据。这种方法由于基于手机系统的代理机制，所以也受到了代理机制只能代理 HTTP 的限制，如果目标应用使用了非 HTTP 的协议，那么这些数据都是无法抓取到的。

1.3 课题研究目标

本课题主要是为打破传统的安卓移动端抓包方式，打造一种更加稳定、便捷、适应性更强的抓包软件。从功能上，软件会包含最基本的数据包抓取、存储功能，以及便捷的数据包查看、过滤，性能上也会做到尽可能小的网络影响。相比于传统的抓包方式，本项目基本上可以杜绝抓包抓不到的情况，只要手机能用 VPN，就能够在不借助任何其他设备的情况下抓取一切网络数据。

1.4 本文内容安排

本文主要内容安排如下：

第一章绪论，主要介绍目前常用的移动端抓包方式，并说明本课题的意义。

第二章系统需求，主要介绍本课题要打造的软件都包括哪些功能。

第三章系统设计，主要介绍本项目的具体设计方案，包括技术方案、体系结构、物理架构、存储方式等。

第四章系统开发，主要说明开发环境、开发中解决的难点问题，以及最终成果的展示。

第五章系统测试，说明本项目的测试相关问题，包括测试设计、测试过程及测试结果。

第六章总结与展望，对本文内容进行总结，并说明本项目需要完善的地方。

第二章　系统需求

2.1 业务需求

本项目旨在简单便捷的帮助用户完成抓包调试的流程，用户需要进行抓包时，只需按照图 2.1 流程进行操作：

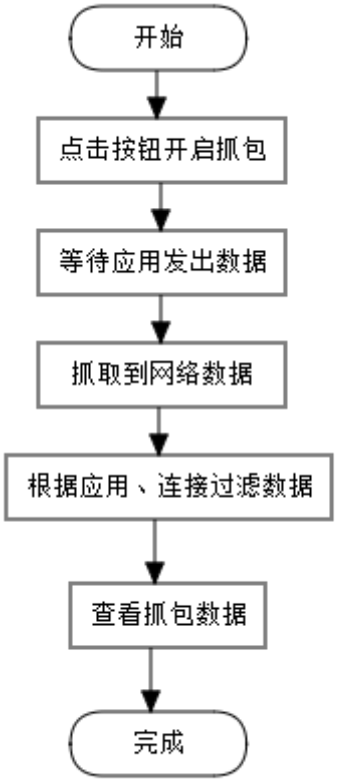

图 2.1　业务操作流程图

其中，用户只需按下应用中的“开始”按钮，等待数据到来即可获得所有数据包，网络数据的截获、解析都是在后台静默完成，并会根据应用、连接进行分批显示，用户随后可以方便的按照时间顺序查看所有应用发出的所有网络请求，并且支持按照应用进行过滤显示。

2.2 系统功能需求

系统功能需求如图 2.2 所示，其中对于用户有：数据包抓取、数据包聚合、数据包存储和读取、数据包过滤查看、查看单个数据包、数据包抓取历史查看、数据包单条保存；对于安卓系统有：数据包转发和响应数据回写到网卡。

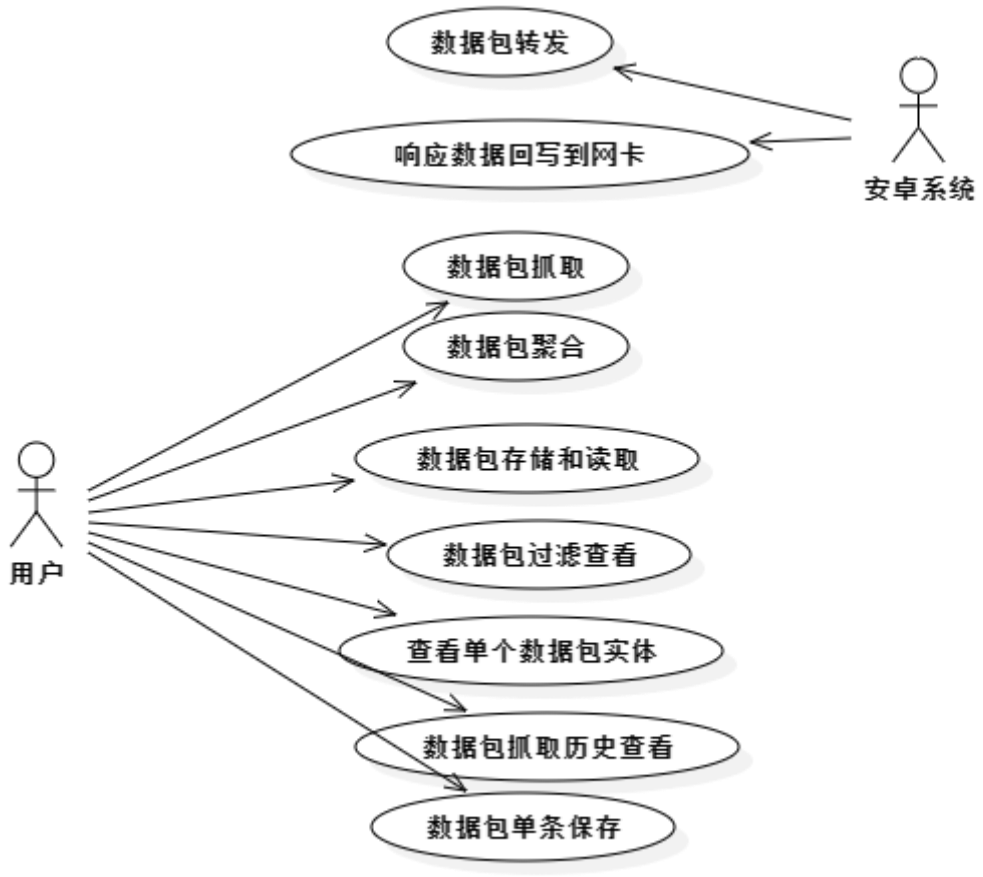

图 2.2　用例图

2.2.1 数据包抓取

- 用例名称：数据包抓取。
- 概述：应用程序可以抓取安卓系统与外界交互产生的所有网络数据。
- 主要角色：用户。
- 前置条件：用户打开应用程序。
- 后置条件：应用程序抓取到安卓系统与外界交互产生的所有网络数据。
- 主成功场景：

  1.用户点按应用程序的“开始”按钮。

  2.等待系统其它应用产生网络数据。

2.2.2 数据包聚合

- 用例名称：数据包聚合。
- 概述：应用程序可以按照应用名与连接 IP、端口进行聚合所有抓取到的数据包。
- 主要角色：用户。

- 前置条件：用户打开应用程序并开始抓包。
- 后置条件：抓取到的所有数据包会按照应用名与连接 IP、端口进行聚合。
- 主成功场景：

  数据包抓取成功时后台静默进行聚合。

2.2.3 数据包存储和读取

- 用例名称：数据包存储和读取。
- 概述：抓取到的所有数据包都可自由存储、读取。
- 主要角色：用户。
- 前置条件：成功抓取数据包。
- 后置条件：抓取到的所有数据包使用自定义的格式进行存储，可自由读取。
- 主成功场景：

  数据包抓取成功时后台静默进行存储。

2.2.4 数据包过滤查看

- 用例名称：数据包过滤查看。
- 概述：可以按照关键字过滤数据包列表中的内容。
- 主要角色：用户。
- 前置条件：成功抓取数据包。
- 后置条件：数据包列表过滤掉所有不包含用户输入关键字的内容。
- 主成功场景：

  1.用户在过滤输入框输入关键字。

  2.输入过程中或完成时应用自动过滤不包含关键字的内容

2.2.5 查看单个数据包

- 用例名称：查看单个数据包。
- 概述：在单独的界面查看单个数据包的内容
- 主要角色：用户。
- 前置条件：成功抓取数据包。
- 后置条件：应用显示单个数据包的内容。
- 主成功场景：

  1.用户在数据包列表中点按要查看的单个数据包。

  2.应用跳转到单数据包查看页面。

  3.应用显示用户选择的数据包的内容。

2.2.6 数据包抓取历史查看

- 用例名称：数据包历史查看。
- 概述：可以自由查看所有的抓包历史记录。
- 主要角色：用户。
- 前置条件：用户打开应用程序。
- 后置条件：数据包列表显示所有抓包历史数据。
- 主成功场景：

  应用启动时列表显示所有抓包历史数据。

2.2.7 数据包单条保存

- 用例名称：数据包单条保存。
- 概述：可以保存单个数据包。
- 主要角色：用户。
- 前置条件：用户正在查看单条数据包。
- 后置条件：单条数据包被保存，并可查看。
- 主成功场景：

  1.用户点按单数据包查看界面的保存按钮。

  2.应用程序弹窗提示是否保存。

  3.用户点击确认按钮。

  4.应用程序保存单条数据包。

2.2.8 数据包转发

- 用例名称：数据包转发。
- 概述：对捕获的数据包进行转发以保证系统其它应用正常运行。
- 主要角色：操作系统。
- 前置条件：用户打开应用程序并开始抓包。
- 后置条件：捕获拦截到的数据包都能正常发出。
- 主成功场景：

  应用程序后台静默进行转发数据包。

2.2.9 响应数据回写到网卡

- 用例名称：响应数据回写到网卡。
- 概述：对接收到的外界数据及 TCP 数据的确认进行回写。
- 主要角色：操作系统。
- 前置条件：用户打开应用程序并开始抓包。

- 后置条件：接收到的外界数据和 TCP 数据的确认回写到 VPN 设备中。
- 主成功场景：

  应用程序后台静默进行回写。

2.3 系统非功能需求

2.3.1 界面需求

应用程序外观采用了安卓 MaterialDesign 风格。

2.3.2 性能需求

1. 应用程序自身应运行流畅，操作时不应有感知到的卡顿出现。

2. 由于应用程序建立了 VPN 设备来接管系统全局网络，因此在做额外处理（保存、解析、转发等）时应该尽可能减少对系统网络环境的负面影响（如网速影响）。

第三章　系统设计

3.1 设计决策

技术方案：如图 3.1 所示，整体结构分为三个模块。其中本机 VPN 服务模块使用 Android 底层 VPN 接口进行拦截全局流量，只负责原始数据生产；VPN 数据转发回写处理服务用于接收原始数据，进行解析、聚合、存储、转发等处理，并提供全局数据模块供其它模块直接访问拿取数据；UI 列表界面通过访问全局数据模块来展示已抓取并处理的所有数据包。

工具：Android Studio、Android SDK、JDK

环境平台：Windows 10（开发）、安卓真机（调试）

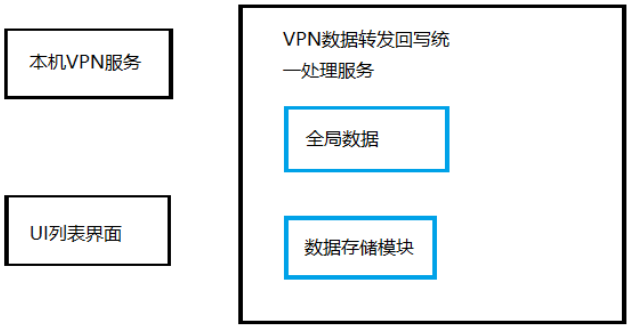

图 3.1　整体结构图

3.2 体系结构设计

3.2.1 逻辑架构

整体架构如图 3.2，其中 packet 模块负责数据包的抓取和回写处理，产生所有的数据，history 模块用于读取 packet 中的数据来展示历史抓包记录和正在抓包的记录，saved 模块读取 packet 中的数据来展示已保存的数据包，editor 展示单个数据包，view 模块包含了一些通用的界面控件，text 模块包含了查看器需要的一些文本处理工具类。

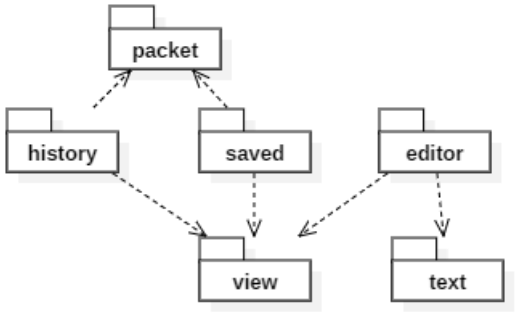

图 3.2　逻辑架构图

3.2.2 开发架构

工程结构如图 3.3、图 3.4 所示，为典型的安卓工程结构：src 文件夹放置 Java 源文件，res 文件夹放置工程所需资源文件，其中 layout 放置界面配置文件，drawable 放置界面所需背景、图片等资源，menu 放置菜单界面配置文件。

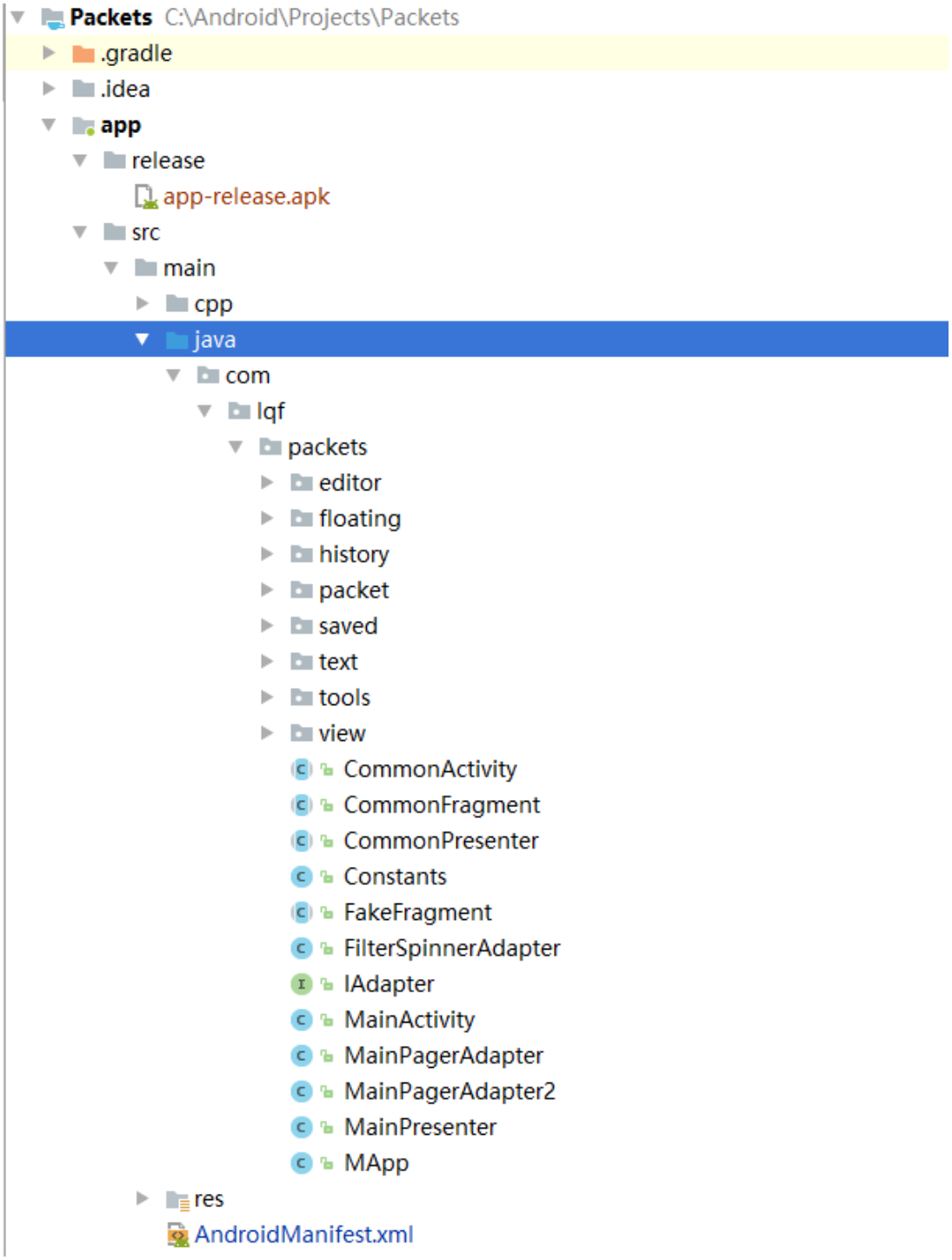

图 3.3　工程整体文件结构图

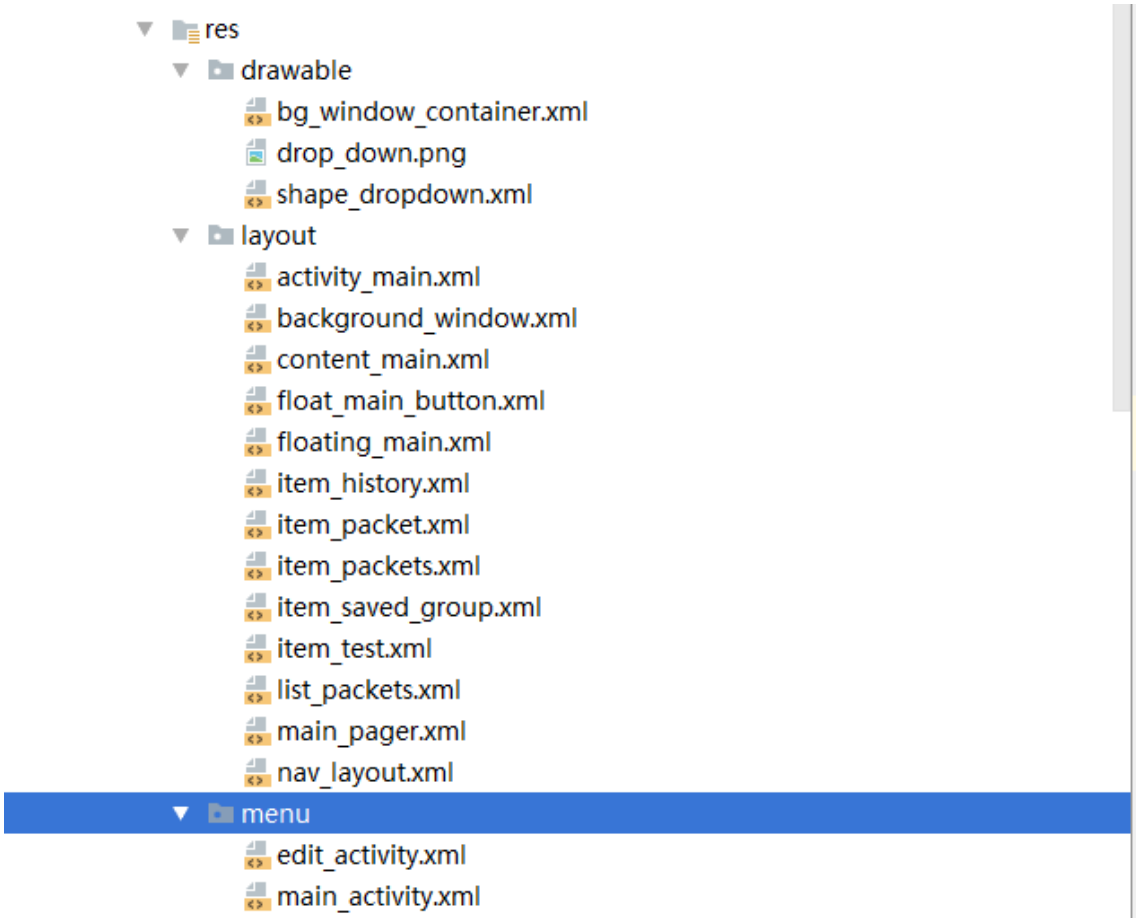

图 3.4　工程资源文件结构图

3.2.3 物理架构

软件只需在客户端安装，VPN 开启所需服务器实质上在客户端内进行了实现（数据包转发模块）。

3.3 界面接口设计

软件主界面包含一个左右滑动的页面，两个页面各包含一个列表，其中一个列表展示所有的数据包记录，另一个列表展示所有保存的数据包。数据包记录列表是一个多级列表，第一级是所有的抓取记录，只展示时间戳，选中点按可以打开下一级，展开选中时间下当次抓取的所有数据包记录。数据包记录列表展示应用名称、应用报名、应用图标、IP、端口，一条记录表示一个连接，点按以展开该连接下的所有数据包。

3.4 数据存储设计

3.4.1 数据模型设计

整体数据模型如图 3.5 所示。主要实体有四种，其中单次抓取表示每一次开启软件抓包获取的所有数据，其内包含多个数据包列表，并且记录抓取时间；每个数据包列表中包含多个原生数据包内容，并记录应用 UID 等信息；原生数据包即抓取到的原始数据，不做任何处理；保存的数据包与单次抓取以同样的方式进行存储，其内包含多个数据包列表。

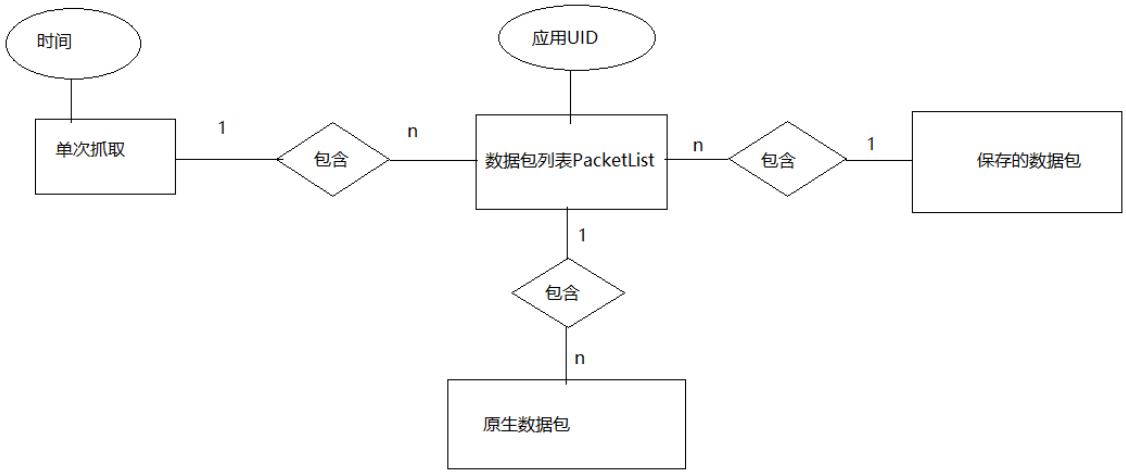

图 3.5　数据模型 ER 图

3.4.2 数据文件设计

数据主要使用文件直接进行存储，每一次抓取都会生成一个以时间戳命名的文件夹，表示一次抓取历史，其内存放当次抓取的所有数据。数据包实体则直接以原始内容进行存储，其中命名方式为“索引_应用 UID.pkts”，一个数据包实体文件包含单个连接的所有数据。实际需要读取使用时与从 VPN 设备中读取的流程是一致的（因为内容格式也是一致的），该种方式虽然格外简单，但是非常利于代码复用，且从效率上优于使用数据库进行存储。

3.5 业务模块设计

3.5.1 数据包抓取

抓取安卓系统与外界交互产生的所有网络数据。

3.5.1.1 软件单元构成

涉及主要类如图 3.6。其中 ClientService 继承于系统提供的 VpnService，用于建立 VPN 网卡设备，ServerService 用于对捕获网络数据的后续处理，Packet 为数据包通用封装，其子类 IPPacket、TCPPacket 分别封装了 IP 类型和 TCP 类型的网络数据。

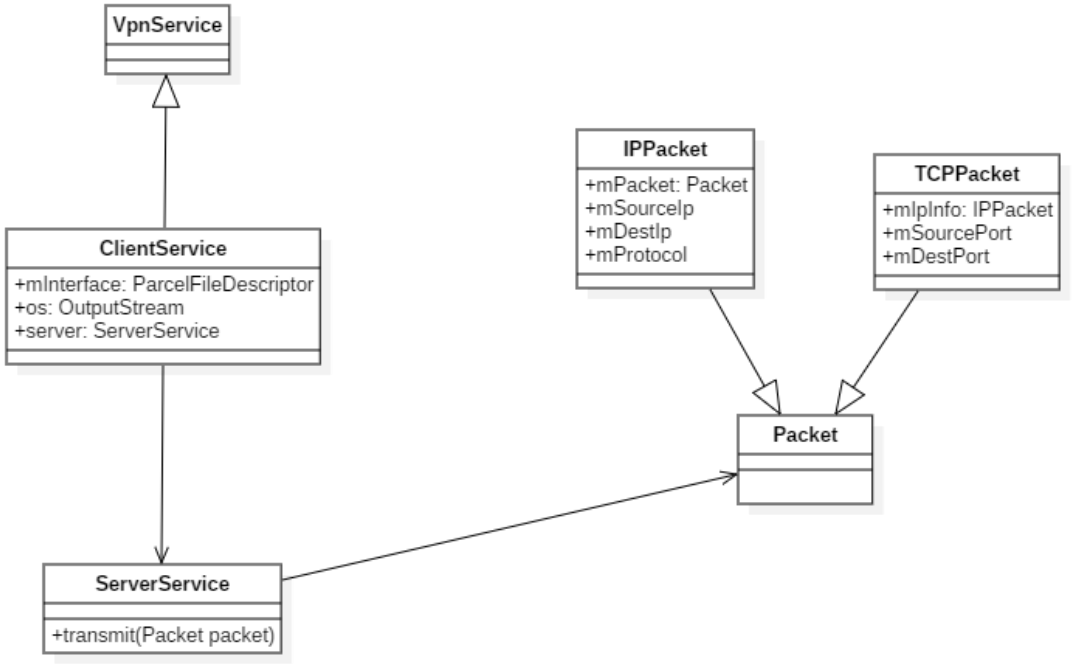

图 3.6　数据包抓取类图

3.5.1.2 执行流程设计

整体流程大致如图 3.7。其中从用户点按“开始”按钮起，首先应用启动了 ClientService，创建 VPN 网卡设备用于获取到所有的链路层网络数据（网卡上直接拿到的数据就是链路层数据流）。网卡设备创建完毕后拿到该设备的文件描述符对应的输入流，一旦该文件描述符上有数据可以读取，则说明安卓系统有其他应用产生了网络请求，此时读取该文件流即可拿到链路层原生数据。同时，将拿到的链路层原生数据封装为上述提到的 Packet 对象进行传递，以进行后续操作。

抓取到数据的处理使用了异步的另一个 ServerService 来进行处理。在 ClientService 创建时会立刻异步的创建 ServerService，ServerService 内维护了一个数据包队列，当 ClientService 拿到数据包并传送给该队列时，则认为有数据包被成功抓取到，此时进行相应的后续处理。

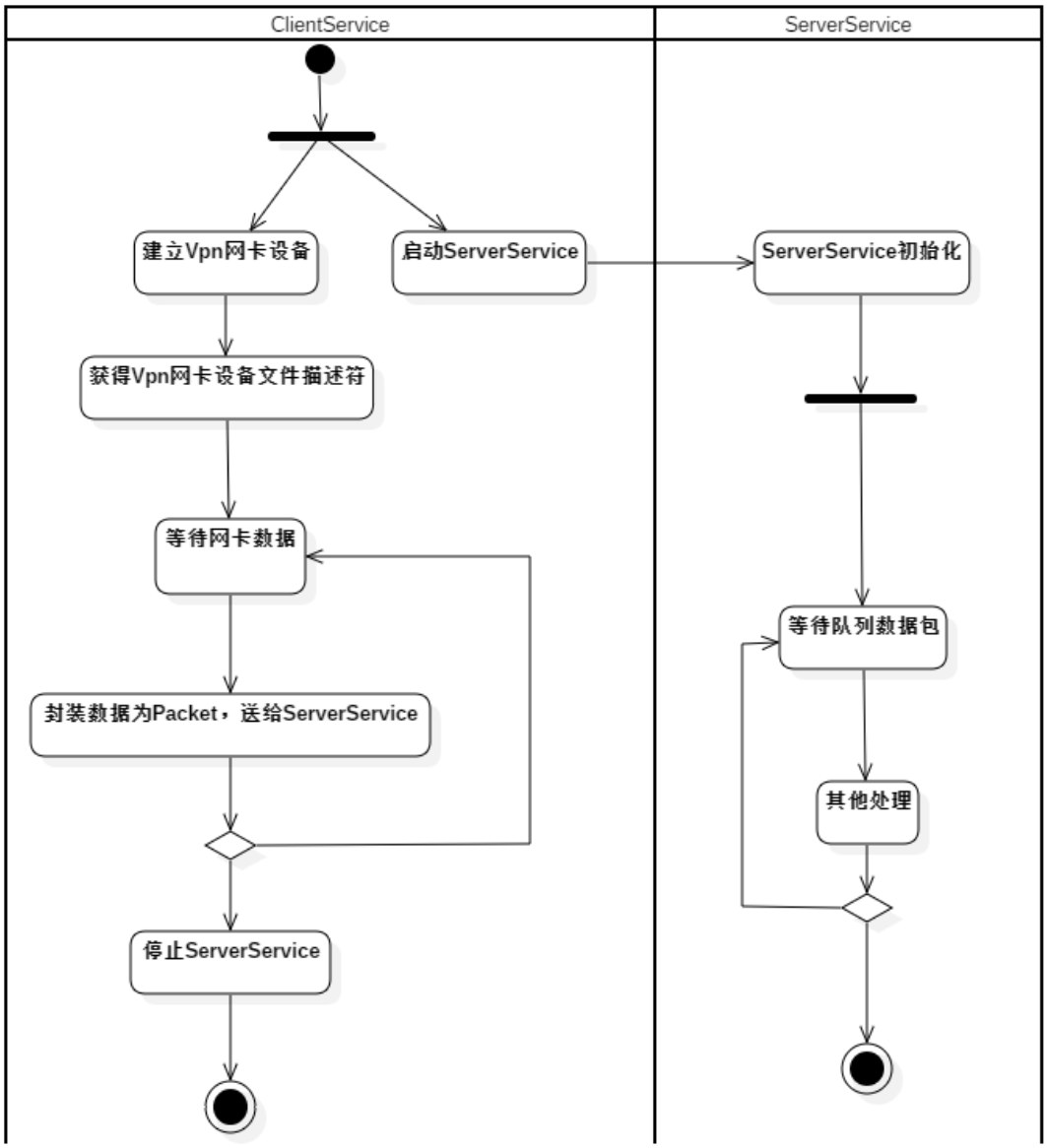

图 3.7　数据包抓取活动图

3.5.2 数据包聚合

应用程序可以按照应用名与连接 IP、端口进行聚合所有抓取到的数据包。

3.5.2.1 软件单元构成

涉及主要类如图 3.8。其中 ServerService 维护一个 CaptureInfo 的列表。CaptureInfo 封装应用使用时一次点击开始按钮到停止按钮之间抓取的所有数据包，其内包含了一个 PacketList 的列表，PacketList 封装了一个应用在一次 TCP 连接中产生的所有数据，其内包含了一个 PacketItem 列表以及该对应的应用信息，而一个 PacketItem 封装了一个 TCP 数据包。AppInfo 即为应用信息的封装，使用 AppPortList 来根据连接端口获取应用信息。

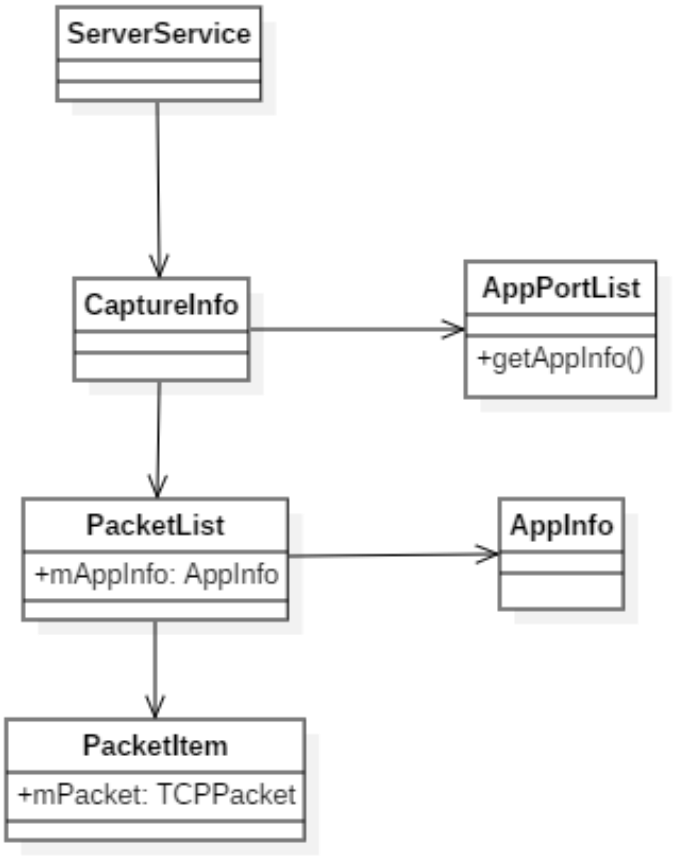

图 3.8　数据包聚合类图

3.5.2.2 执行流程设计

数据包后期处理主要由 ServerService 发起，如图 3.9，图中省略了除 ServerService 的其他地方。其中当 ServerService 的独立线程轮询拿到了抓取到的 TCP 数据后，判断其是否是一个新的连接（通过 TCP 内标志位可以进行判别，SYN 为新连接），如果是新的连接则使用它的端口号来查询应用信息（安卓系统内产生的所有 TCP 连接都会在“/proc/net/tcp”路径下记录源端口号和相关应用的信息，读取这个文件可以通过 TCP 连接的源端口号来得知该 TCP 连接到底是哪个应用发起的），建立新的 PacketList，否则拿到已有对应的 PacketList，向其内创建并添加 PacketItem。

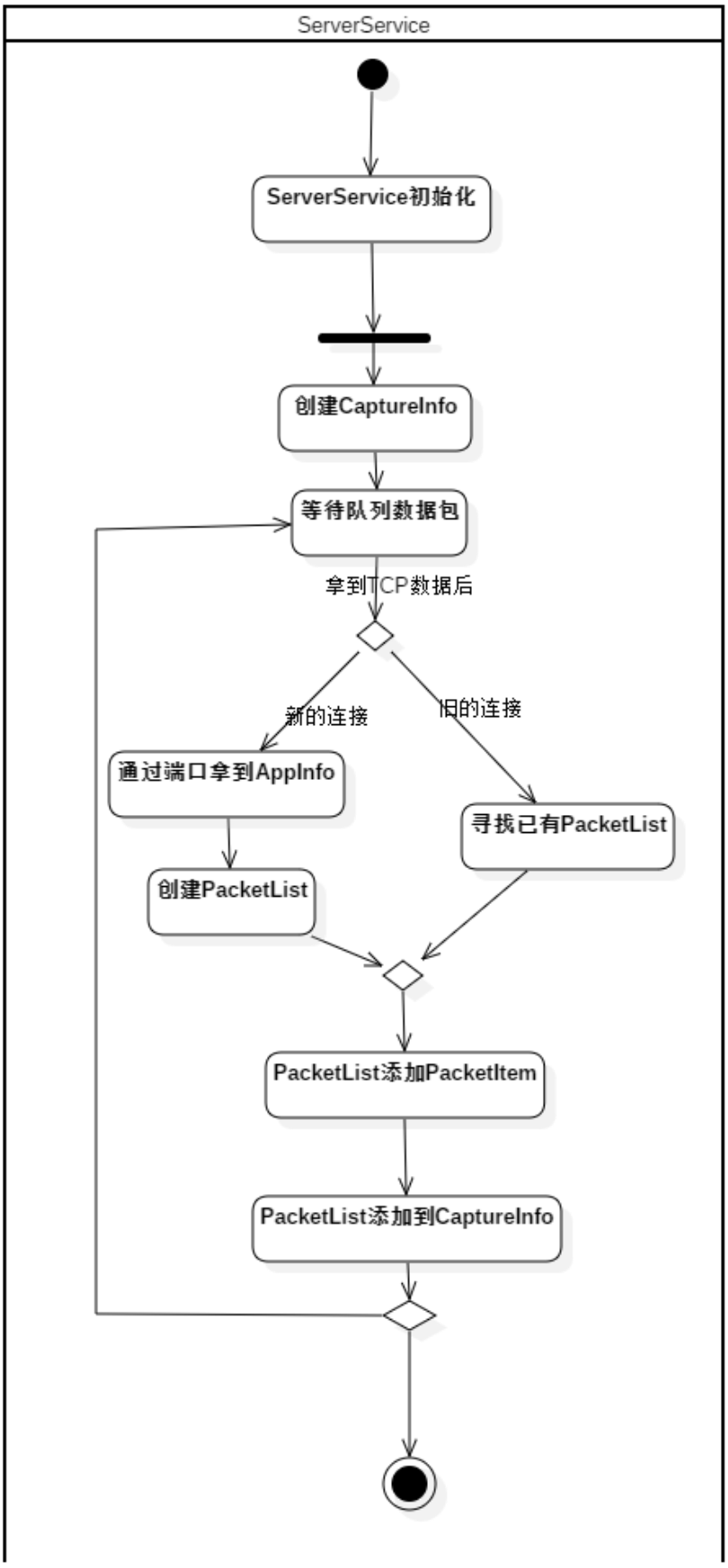

图 3.9　数据包聚合活动图

3.5.3 数据包存储和读取

抓取到的所有数据包自动进行存储，自由进行读取。

3.5.3.1 软件单元构成

涉及主要类如图 3.10。其中 LocalPackets 为一个单例类，引用全局所有的抓取数据，即 CaptureInfo。LocalPacketsMgr 用于单独管理 LocalPackets，PersistThread 用于统一处理所有的文件读写操作。PersistRequest 是封装了多种读写操作的辅助类，需要进行数据包的读写时直接使用 PersistRequest 的子类进行构建对应操作。LocalPackets 中支持设置回调，如 OnPacketChangeListener 用于监听数据包增添变化。

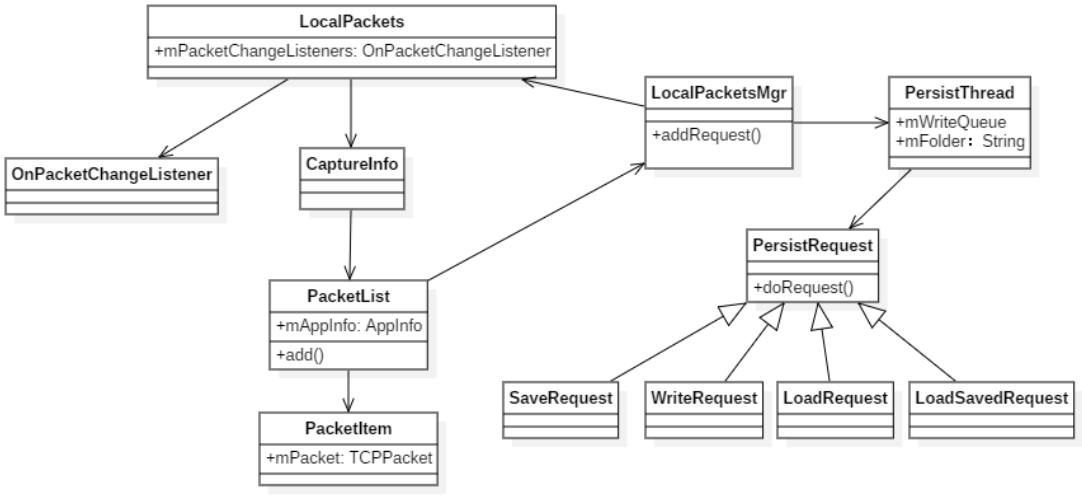

图 3.10　数据包存储和读取类图

3.5.3.2 执行流程设计

数据包存储写入流程如图 3.11。其中 ServerService 部分省略了获取数据及处理的部分，直接从添加数据包到列表开始。ServerService 在启动时，首先初始化 Localpackets，其实 LocalPackets 初始化时主要做的就是创建 PersistThread，该线程用于单独处理所有的文件读写操作，其维护了一个 PersistRequest 的队列，一旦该队列中有新的 PersistRequest，则进行执行。除此外，由于每次打开应用开启抓包的存储路径都是有差异的，该线程还会关联着正确的存储路径，PersistRequest 要做的其实就是在指定文件夹下进行读写操作。实际运作中，当 ServerService 内抓取到了新的 Packet 并作为 PacketItem 添加到 PacketList 中时，就创建一个 PersistRequest 的子类 WriteRequest，来进存储刚刚添加的数据包。

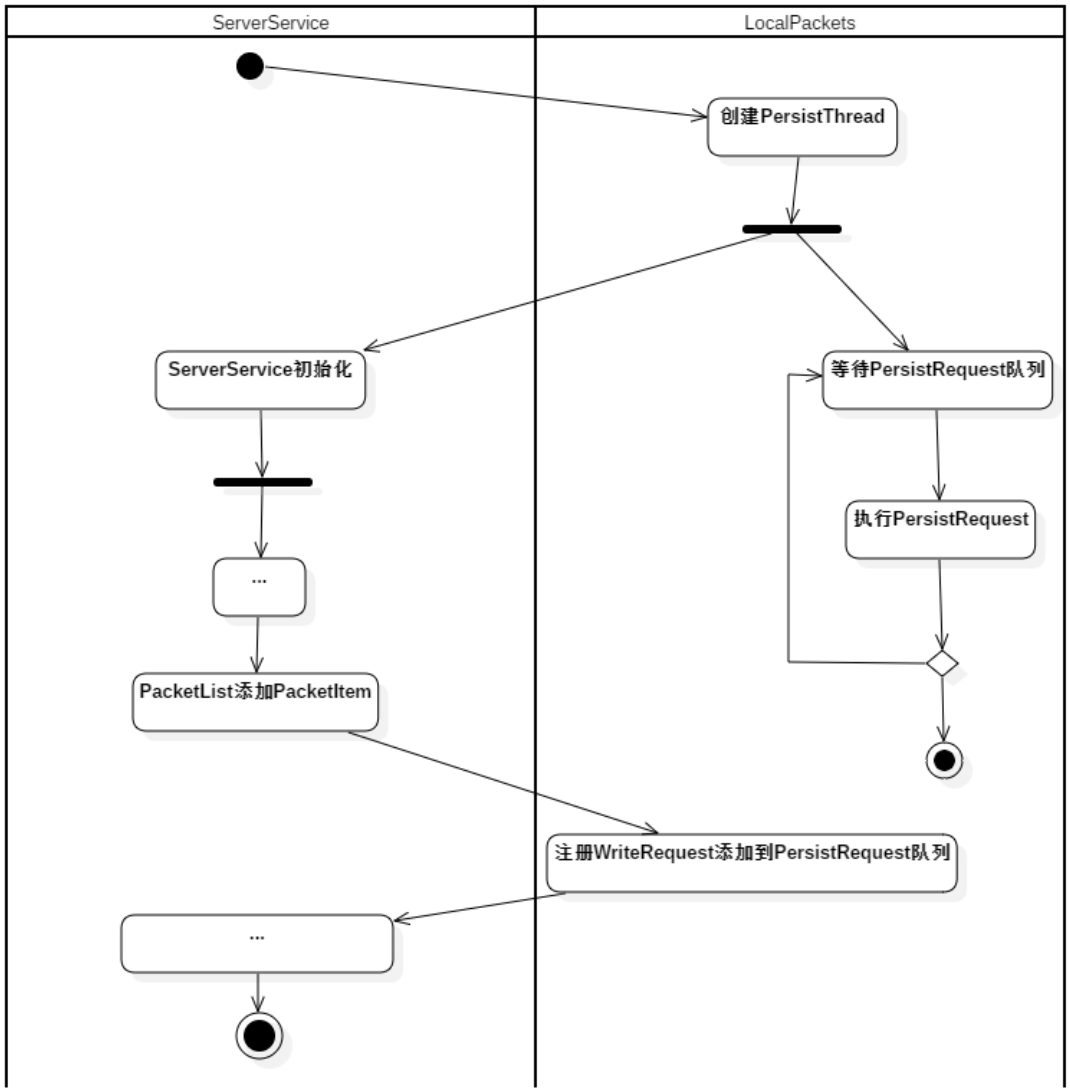

图 3.11　数据包存储和读取活动图

3.5.4 数据包过滤查看

可设置关键字过滤掉不包含关键字的所有数据包来进行查看。

3.5.4.1 软件单元构成

涉及主要类如图 3.12。其中 CaptureInfo 为所有抓包历史，其内包含一个 PacketsFilter，用于过滤其子列表 PacketList，PacketList 内包含所有的抓包数据及应用相关信息。SavedInfoFilter 为保存的数据包的应用信息，并可过滤其子列表 SavedFilter；SavedFilter 为对应应用下所有保存的数据包，可过滤其子列表 PacketList。Filter 类提供过滤辅助服务，只需子类设定过滤条件即可进行过滤，PacketList、PacketsFilter 等类均继承了它。

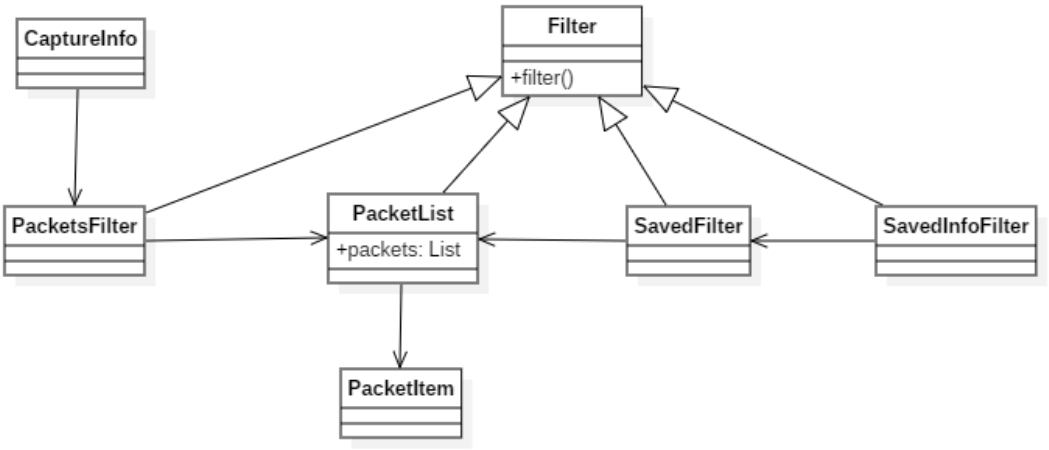

图 3.12　数据包过滤类图

3.5.4.2 执行流程设计

数据包过滤流程大致如图 3.13。其中主要过滤流程实现在 Filter 中，Filter 内默认存在一个列表，即未进行过滤的列表，此时 Filter 与普通 ArrayList 的使用和功能是一致的。当向 Filter 中添加一些数据后，并设置过滤类型和关键字时（过滤方法接受两个参数，一为类型，二为关键字），Filter 针对每一种设定的过滤类型创建一个新的列表（如果有了就直接复用），即为过滤列表。随后，Filter 根据子类实现的过滤条件来过滤出所有符合条件的数据加入过滤列表中。

3.5.5 查看单个数据包

可跳转新的页面来查看单个数据包的内容。

3.5.5.1 软件单元构成

涉及主要类如图 3.14。其中 EditActivity 为单独的页面，跳转来查看单个数据包内容，且为 MVP 模式，EditPresenter 用于对主要 View 和数据实体进行操作。EditPacketInfo 为数据实体封装，FixedWithTextView 为该页面最主要的 View，用于显示文本内容。SimpleFixedLayout 为批量字符显示的绘制辅助类，TextEditor 为文本内容缓存读取辅助类，并使用了独立线程 CacheThread 来进行缓存。

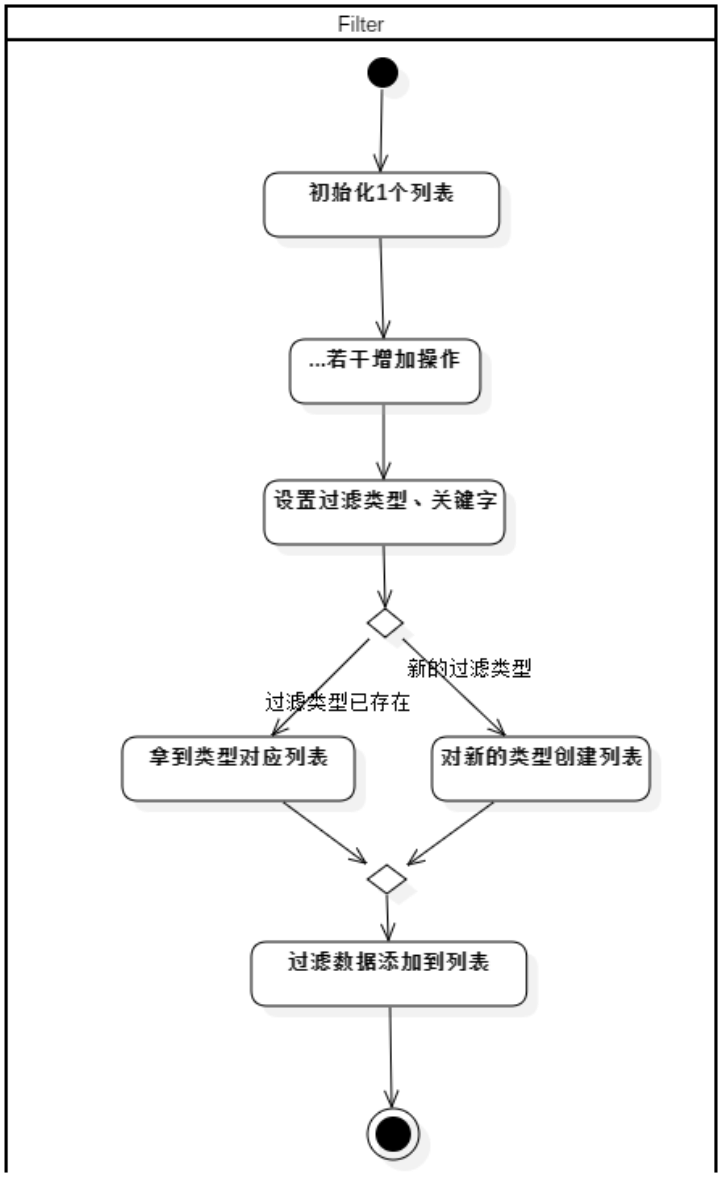

图 3.13　数据包过滤活动图

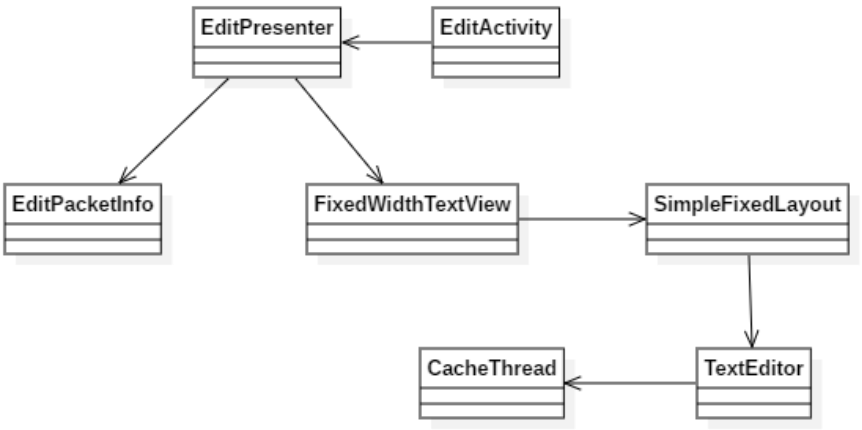

图 3.14　单个数据包查看类图

3.5.5.2 执行流程设计

查看单个数据包流程大致如图 3.15。首先 EditActivity 按照预定布局初始化所有 View 及 Presenter，EditPresenter 拿取到数据包实体封装，来对 FixedWithTextView 进行相关设定，并对 CacheThread 进行初始化。此时 View 绘制文本内容需要先拿到数据包的内容，针对一些从文件读取的情况，此时先判断目标文本内容是否已经读取并存在，如果不存在则向 CacheThread 发出缓存消息，并结束此次绘制；否则直接进行文本绘制。CacheThread 则在初始化完成后进入等待 Cache 消息的状态，一旦有 Cache 消息来，则进行文本内容读取，读取结束通知 UI 线程进行刷新。

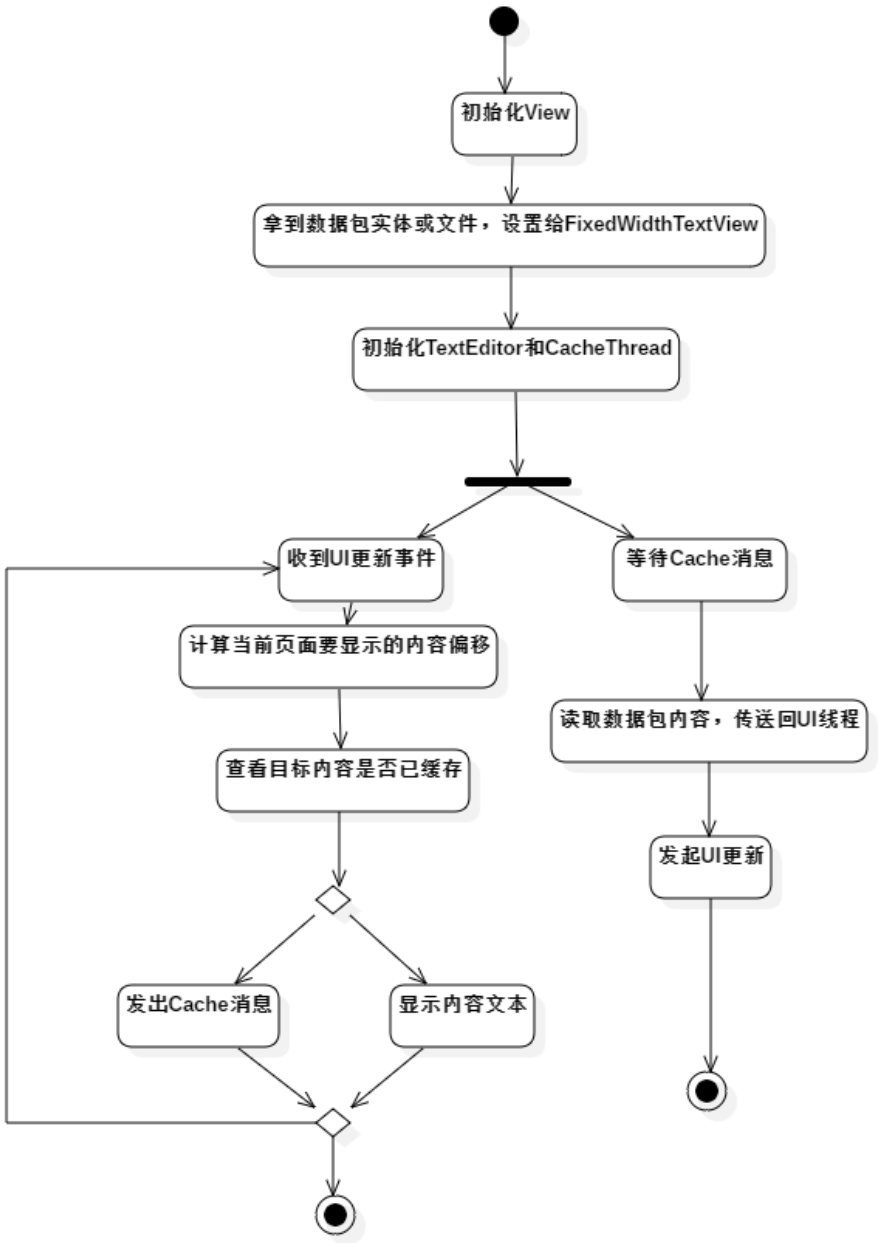

图 3.15　单个数据包查看活动图

3.5.6 数据包抓取历史查看

可查看包括当次抓包的所有抓包历史的数据。

3.5.6.1 软件单元构成

涉及主要类如图 3.16。这里不再介绍数据包读写机制相关类，其中 HistoryFragment 为抓取历史查看页面，同样以 MVP 模式进行构建，HistoryPresenter 用于控制所有的数据操作和 View 绑定。ExpandableRecyclerView 为自定义的一个多级展开的列表视图，对应的 HistoryAdapter 为历史数据绑定辅助类。ExpandableItem 为树形结构的节点操作封装，多级展开后每一个条目都是 ExpandableItem 的一个节点，树形结构比较适合。

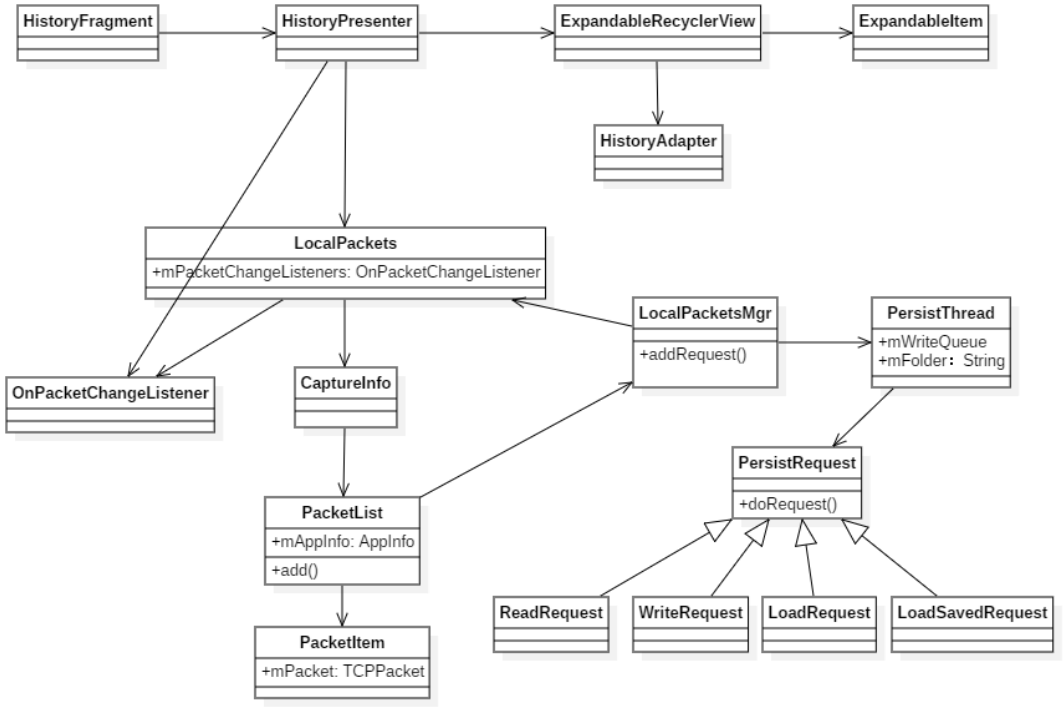

图 3.16　数据包抓取历史类图

3.5.6.2 执行流程设计

数据包抓取历史查看流程大致如图 3.17。首先 LocalPackets 在最开始进行最基本的初始化工作，创建 PersistThread。HistoryPresenter 初始化时会设置一个 OnPacketChangeListener，用于监听所有的数据更新并在数据更新同时更新 View 数据绑定，可在正在抓包时动态更新数据包列表或动态异步刷新数据包列表。当列表视图触发 UI 更新时，先查看需要展示的数据是否存在（如果是历史数据，则很可能还没有从文件读取），如果存在就直接进行绑定；否则发出 ReadRequest，

让 PerisistThread 进行文件读取。PersistThread 当收到 ReadRequest 时并处理完毕时，会直接操作 LocalPackets 中的数据，此时 LocalPackets 中的数据有更新时，会调用 OnPacketChangeListener 来通知监听。由于 HistoryPresenter 最开始已经设置好了监听，当监听触发时再次进行 UI 更新的流程即可完成异步更新流程。

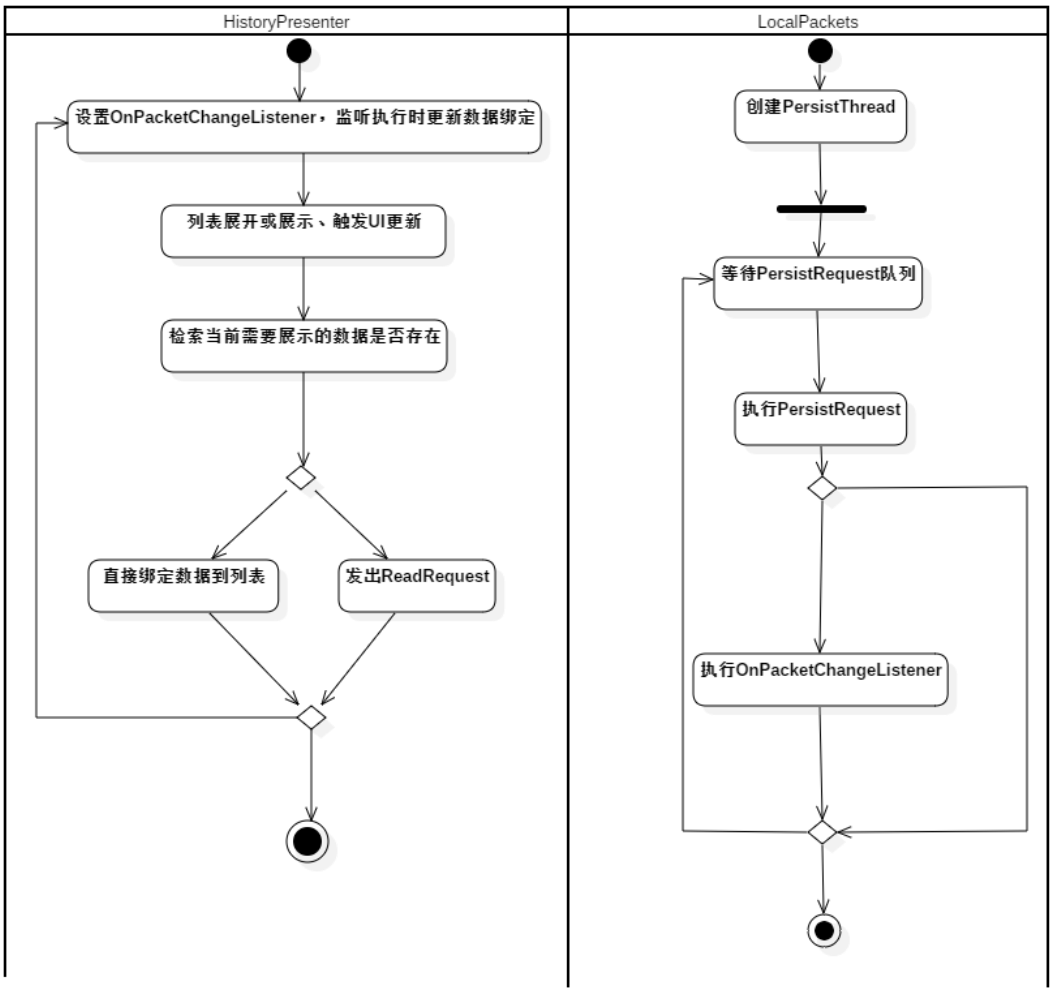

图 3.17　数据包抓取历史活动图

3.5.7 数据包单条保存

可在查看单个数据包时保存单条数据包。

3.5.7.1 软件单元构成

涉及主要类如图 3.18。这里不再介绍数据包读写机制相关类，整体结构类似于历史记录查看部分，其中 SavedFragment 为抓取历史查看页面，同样以 MVP 模式进行构建，SavedPresenter 用于控制所有的数据操作和 View 绑定。

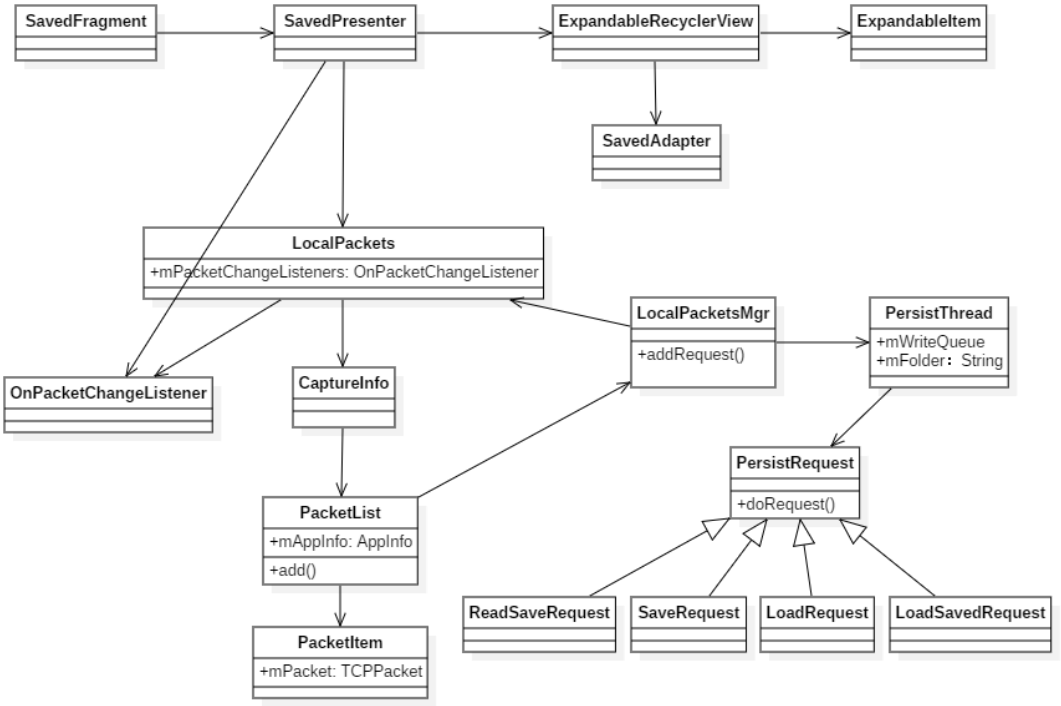

图 3.18　数据包单条保存类图

3. 5. 7. 2 执行流程设计

数据包抓取历史查看流程大致如图 3.19。整体流程类似于数据包存储，在单条数据包查看界面点击右上角保存按钮时，随即发出一个 SaveRequest 给 PersistThread 进行实际的保存处理。SavedFragment 展示部分随后会收到保存成功的回调，之后的展示逻辑与历史抓取查看部分是一致的。

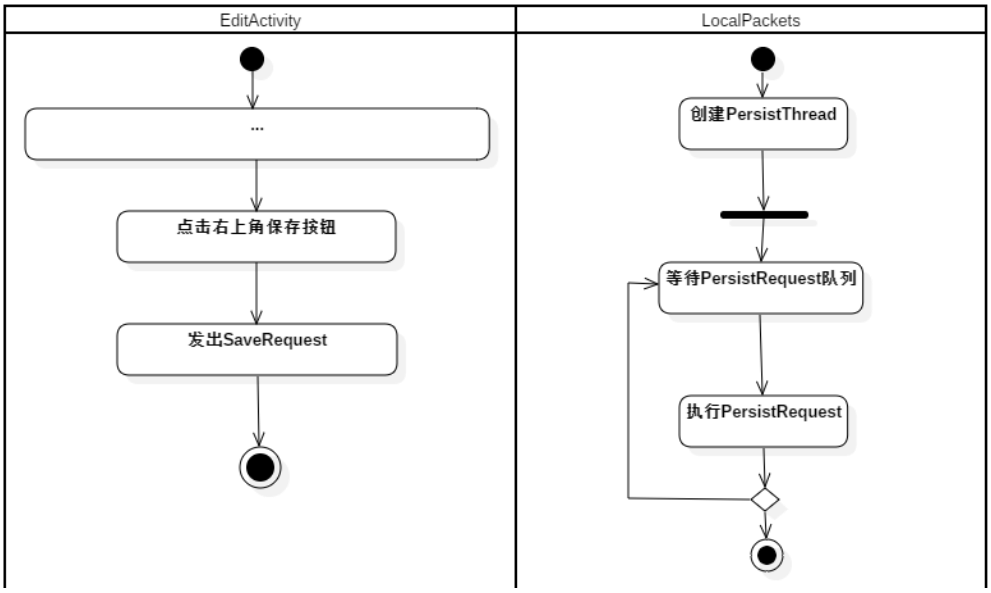

图 3.19　数据包单条保存活动图

3.5.8 数据包转发

应用程序转发所有拦截到的数据包。

3.5.8.1 软件单元构成

涉及主要类如图3.20。其中TransmitThread用于统一处理所有的数据包分发，TCPStatus 封装了连接状态，其内包含着连接实体 SocketChannel，以及预备要向外发送的一个列表，SendEntry 封装发送内容，其内包含着要发送的数据包实体。ReadThread 用于转发数据包以及负责本地数据与外界的交互（实际的网络请求全部由该线程处理），Key 是一个简单实体，用于 NIO 专用的事件注册。

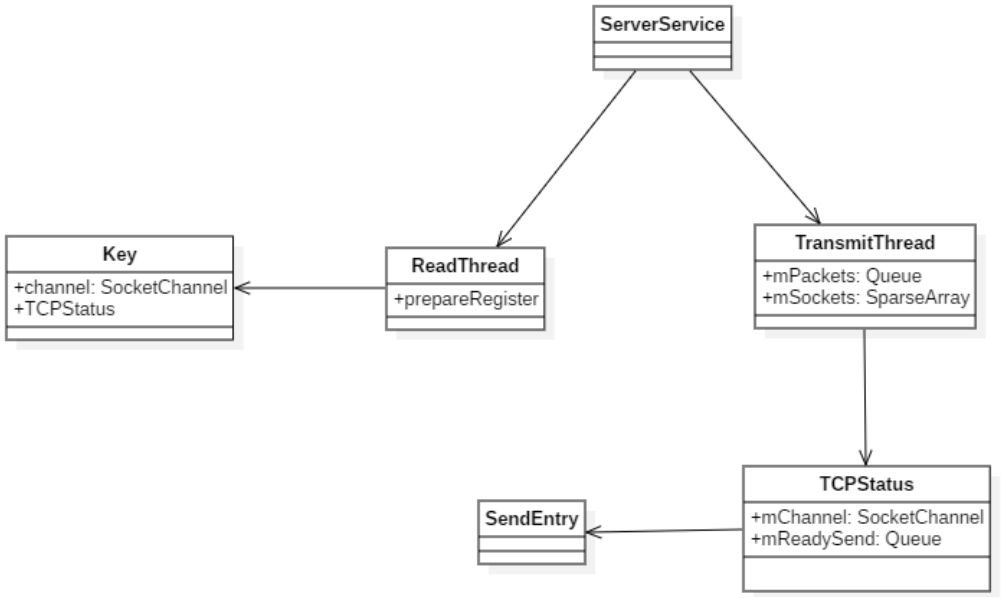

图 3.20　数据包转发类图

3.5.8.2 执行流程设计

数据包转发流程如图 3.21。从 ServerService 开始，初始化时先建立 TransmitThread 和 ReadThread，其中 TransmitThread 用于统一处理所有数据包。当发现新的数据包到来时，先寻找是否已经建立了这个连接，查找已经有的对应的 TCPStatus，如果没有则新建，并向 ReadThread 注册对应的操作。如果创建了 TCPStatus，则向 ReadThread 注册创建连接的操作；如果已有 TCPStatus，说明有数据要写入，则注册写入数据的操作。ReadThread 在执行时直接进入循环，先查看是否有来自 TransmitThread 注册来的附加操作，如果有则使用 NIO 进行注册响应操作（NIO 机制，先注册，然后监听等待资源可用响应，再进行对应的操作）。随后开始监听 NIO 是否有可以进行的操作，如果有则执行对应操作（创建连接或写入数据）。使用 NIO 相对于传统 Socket 来说，只用有限的线程就能处理多数的连接，具有极大地性能提升。

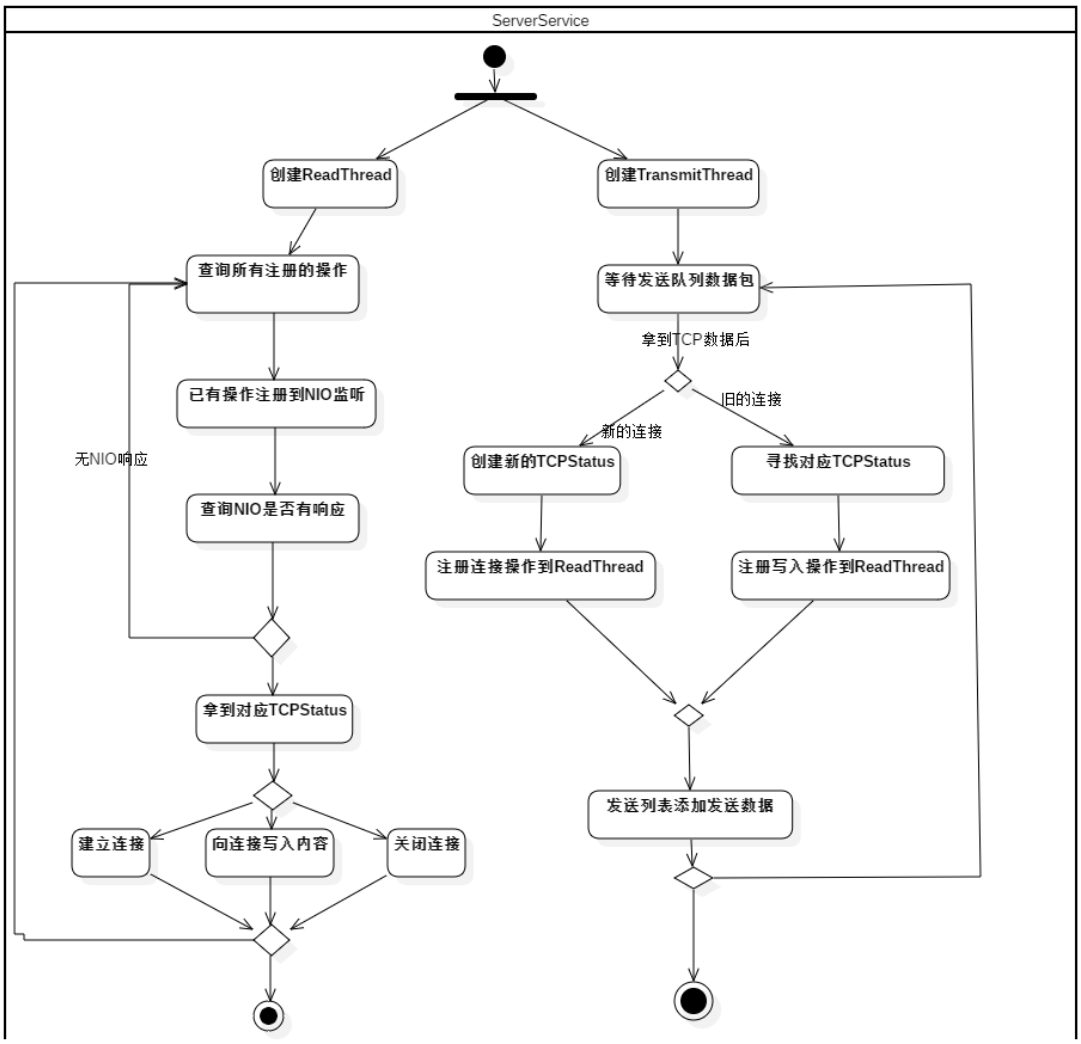

图 3.21　数据包转发活动图

3.5.9 响应数据回写到网卡

用于拦截数据包并转发后，进行必要的数据回写，保证操作系统运行正常。

3.5.9.1 软件单元构成

涉及主要类如图 3.22。其中 ReadThread 为处理数据包转发的线程，WriteThread 为处理数据包统一回写的线程，包括外界响应数据回写和 TCP 回应数据回写，回写使用 LocalService 提供的 write 方法进行操作。

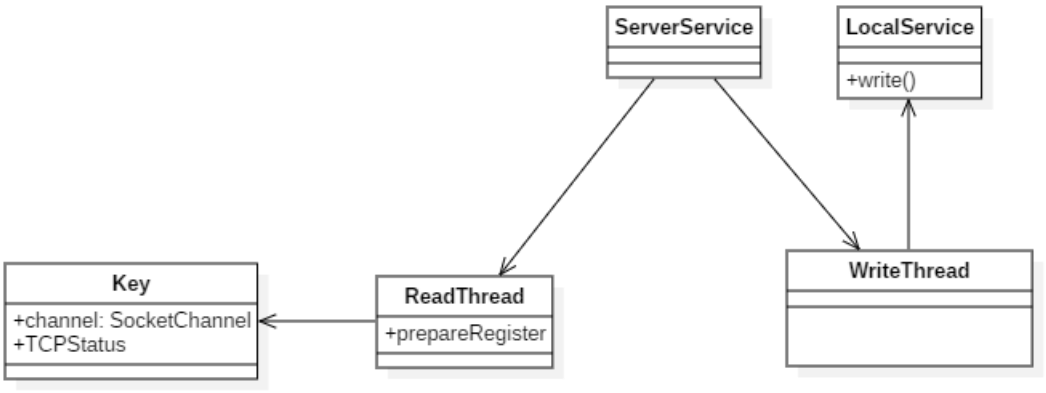

图 3.22　响应数据回写类图

3.5.9.2 执行流程设计

响应数据回写流程大致如图 3.23。从数据包转发部分开始，当 ReadThread 发现有 NIO 事件可用时，判断其事件类型，如果需要向外界写入数据，则手动构建对应 TCP 数据的确认响应；如果需要从外界读取数据时，则读取外界数据，并构造封装的 TCPPacket。不论哪种构造的 TCPPacket，统一交付给 WriteThread 的队列中，在 WriteThread 独立线程中进行统一写入到 LocalService 中，LocalService 只需将传递来的内容实体写入到网卡文件流中即可完成操作。

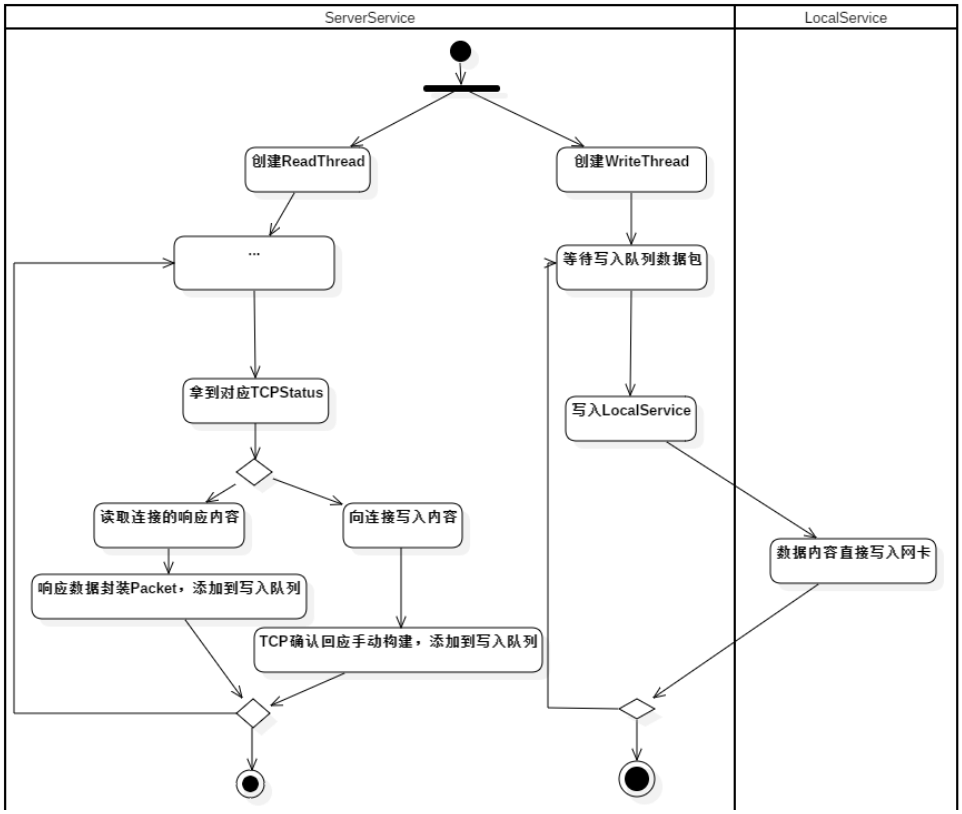

图 3.23　响应数据回写活动图

第四章　系统开发

4.1 开发环境

工具：Android Studio、Android SDK、JDK

环境平台：Windows 10（开发）、安卓真机（调试）

4.2 关键技术

4.2.1 网卡链路层数据的解析

虽然只要成功建立 VPN 网卡设备就能直接拿到系统全局的网络数据，但是网卡内获取到的是链路层原生数据，所以首先要对链路层数据进行解析。本应用主要对 IP 数据及 TCP 数据进行解析，其他协议由于使用极少甚至不再使用，所以暂时不进行处理。

（1） IP 数据解析

这里引用 TCP-IP 详解一书中的图：

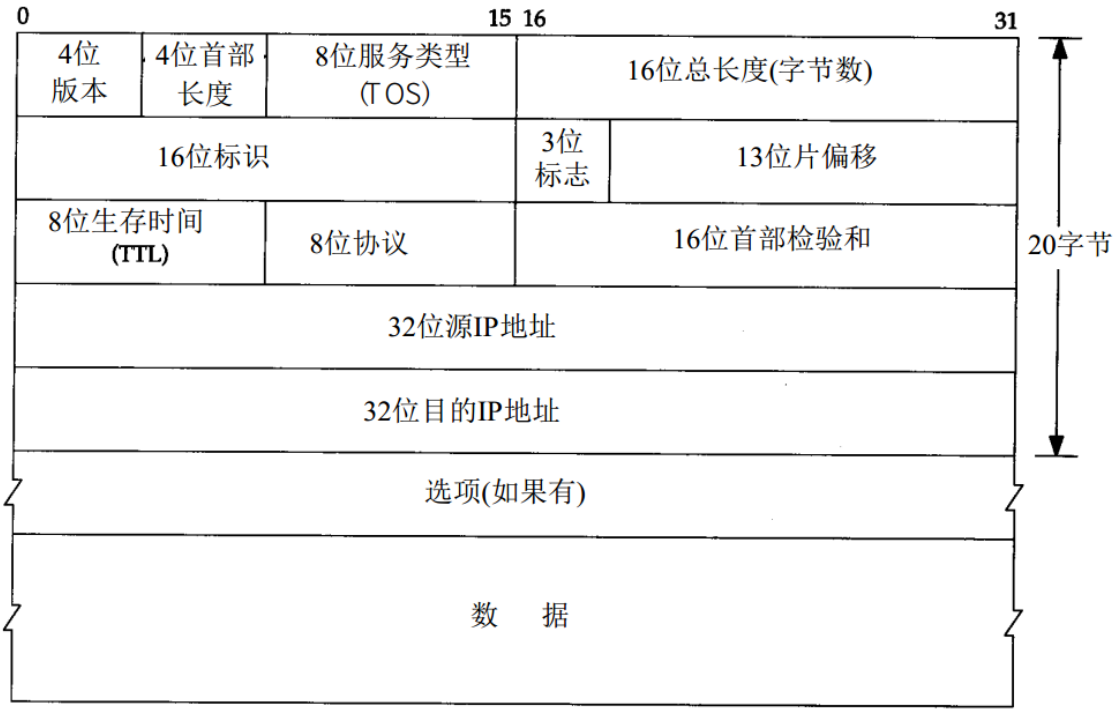

图 4.1　IP 协议图

由于 IP 协议属于无连接协议，所以 IP 协议头的解析实际上较为简单。首先从网卡文件流中读取到了一个字节数组，此时根据 IP 协议头部中的“16 位总长度偏移”，来得知该 IP 数据报的具体长度。如果已经读出的字节数组不足该长度，则要继续读取；如果超出该长度，则要进行截取处理。拿到完整的一个 IP 数据

报之后则可以进行其他信息的读取。随后按照 IP 协议的头部信息依次获取所有有用信息，即完成了 IP 数据的解析。

（2） TCP 数据解析

TCP 数据解析的前提是，遇到了一个 IP 数据报。当完成了一个 IP 数据报的解析后，此时根据已解析的 IP 数据报头部信息中的“8 位协议”标识来判断该 IP 数据报承载的数据实体类型，如果该标识值为 6 就说明它承载了一个 TCP 数据，此时可以继续后面的解析，这里同样引用 TCP-IP 详解中的图。此时可以按照 IP 数据解析的方式，将 TCP 数据提取出来，并得到所有必要的头部信息。

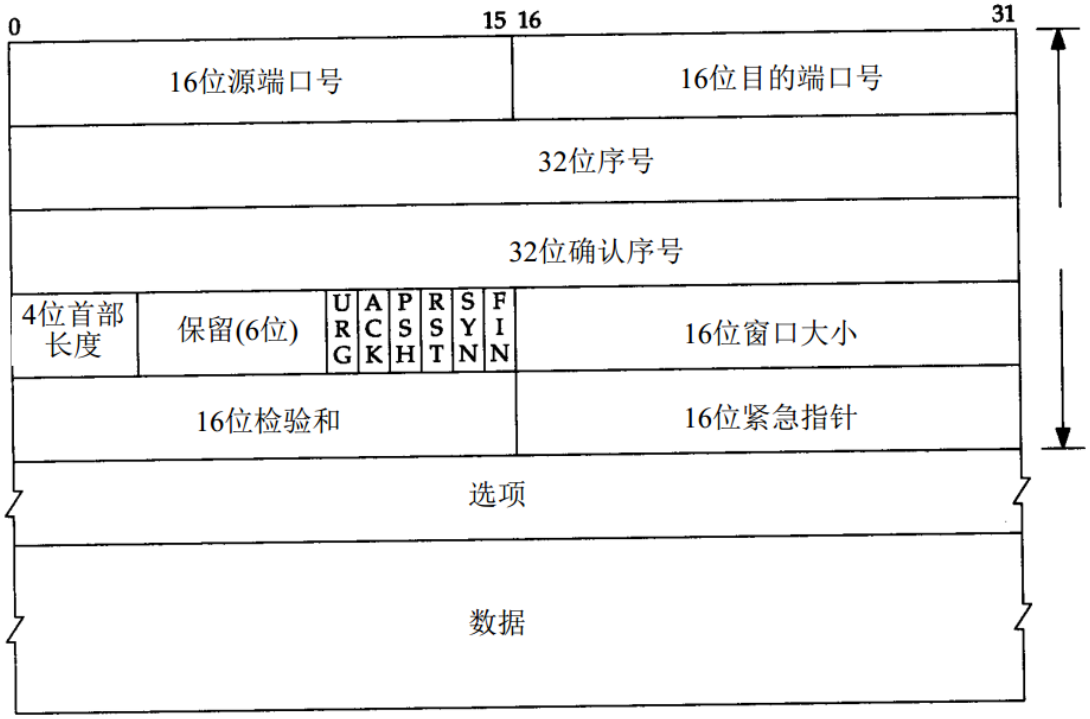

图 4.2 TCP 协议图

4. 2. 2 数据包的转发

在前面已经完成了全局数据的抓取，并成功的解析为可读的数据，但由于创建的 VPN 设备已经接管了真实网卡，系统内产生的流量到达 VPN 设备后如果不进行任何操作，那么数据是无法发送到外界的，这时就需要进行手动转发。

（1） 被淘汰的传统 Java Socket 转发

如果使用传统的 Java Socket 进行转发操作，那么流程应该大致如图 4.3。其中每捕获到一个系统的连接，那么就需要开启两个线程：一个用于新连接的写入操作，一个用于新连接的读取处理。这样一来，可能会产生大量的线程，极大地拖累手机运行速度和网络速度。实测在这种实现下，网路几乎变得不再可用。

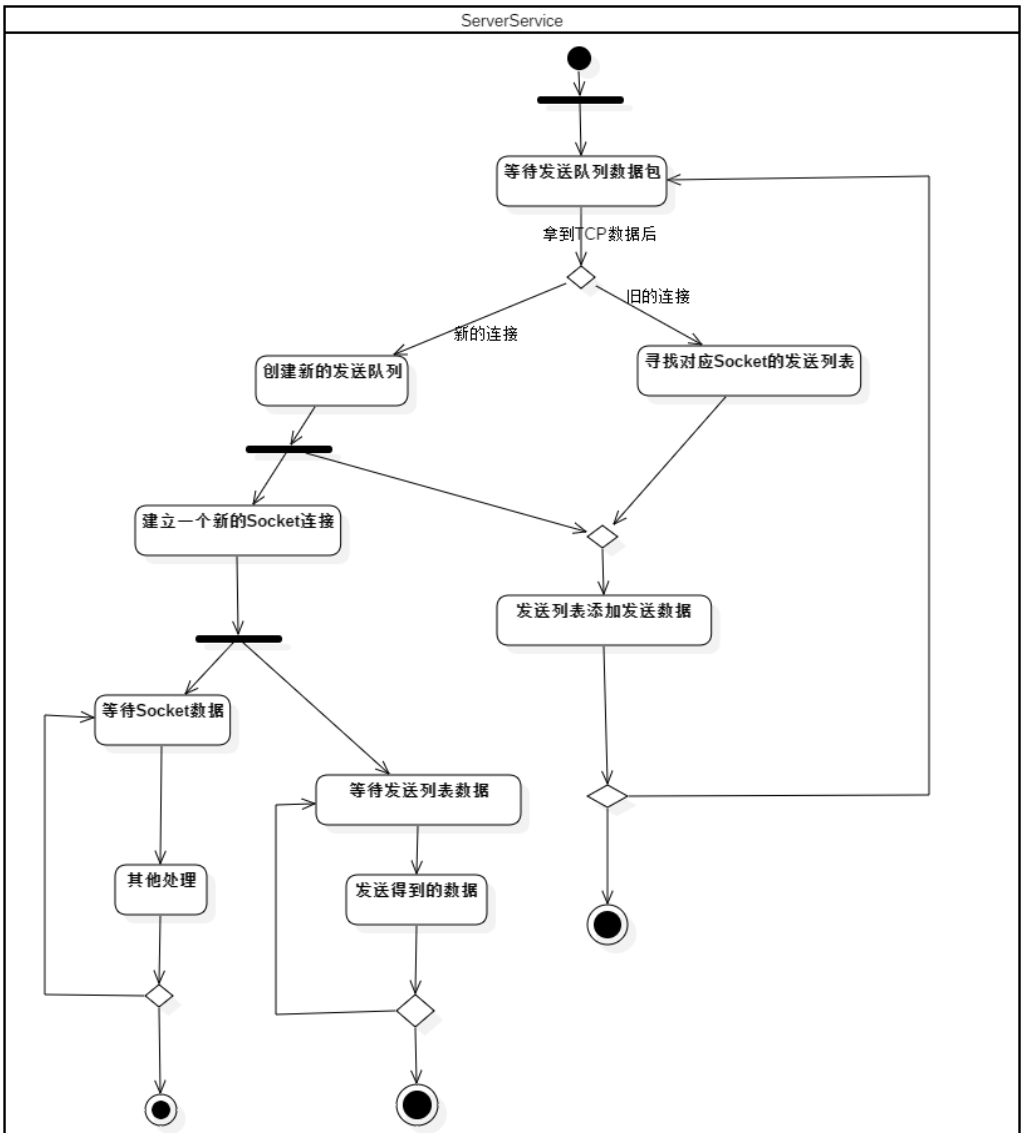

图 4.3　传统 Socket 转发活动图

（2） 优化后的 Java NIO 转发

为了优化传统 Socket 的每建立一个连接都要新开线程的方式，这里使用了 Java NIO。Java NIO 实质上是一种多路复用的处理方式，在单个线程中采用系统主动注册事件的方式来处理所有的事件，理论上处理多连接的效率可接近于单连接的效率。具体工作方式见图 3.21(数据包转发活动图)。改进之后基本上不会察觉到明显的网络卡顿。

4. 2. 3 数据包的回写

事实上即使在捕获了所有网络请求时进行了转发，对于系统内运行的应用来说，仍旧无法正常运作。主要是因为，目前安卓手机上的绝大多数应用使用的都是 TCP 协议进行网络请求（目前主流的 HTTP、HTTPS 也是基于 TCP 的），而 TCP 有连接、可靠的特性使得其每次网络请求都需要一次确认回应。虽然已经进

行了转发，但是对于第三方应用来说，它发出了网络请求后没有收到其请求的确认回应。因此当使用创建的 VPN 设备接管了真实网卡的工作后，必须对每条 TCP 数据进行一次确认回应后，其他应用才能够正常运作。另外，还需要以同样的方式将远程发送过来的数据进行回写。

（1） TCP 数据构建

要构建 TCP 数据正确的回应，首先要按照 TCP 协议格式来构造一个完整的 TCP 数据实体。除了如端口号、首部长度等普通数据，还需要：1. 确认序列号，拿到要回应的数据的序列号以及数据实体的长度（如果是 SYN 连接请求，数据实体长度应加一），二者相加来得到回应包的确认序列号，并写入回应数据包正确的位置。2. 序号，这里需要在回应该连接的第一个数据开始（SYN 连接请求），就记录下序列号（初始为 0 即可），并且在每次收到外界数据进行回写时将序列号更新为其加上该数据长度。

（2） IP 数据构建

大致同上，IP 数据中也有类似序列号的字段，即 16 位标识，需要以同样的方式进行更新。不同的是该字段是全局共用的，每当向 VPN 设备写入一次 IP 数据报时，该值就应该增加 1。

4. 3 开发成果

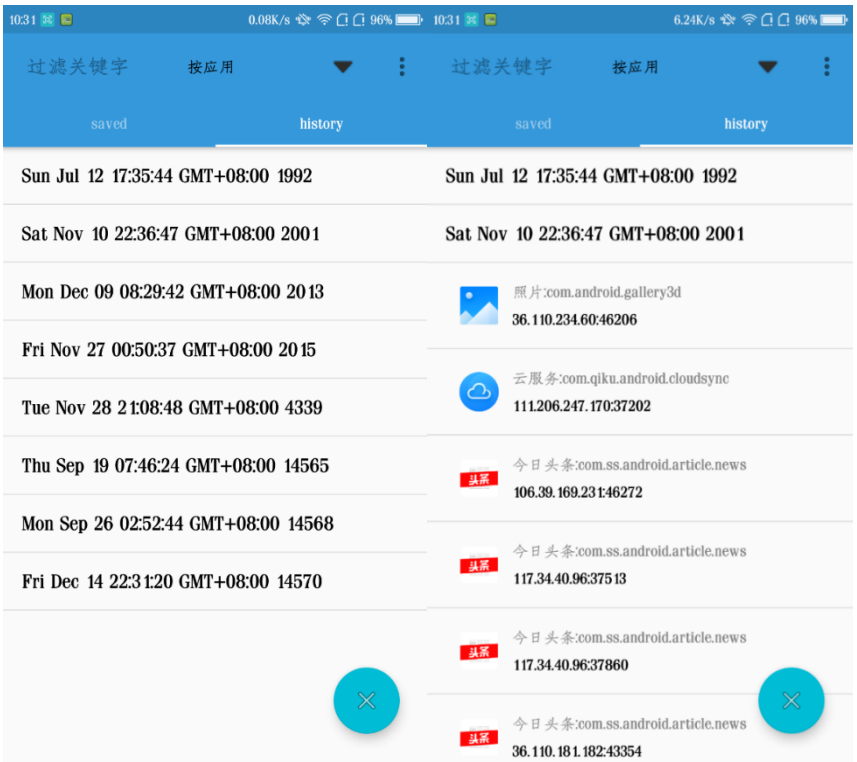

图 4.4　历史数据列表截图

历史展示列表是一个三级列表，如图 4.4、图 4.5 所示。其中第一级只显示

每次抓取的时间点，点按展开第二级，第二级按连接进行聚合数据包，并会展示每个连接对应的应用信息、IP、端口。点按连接可展开第三级列表，展示实际数据包的内容。

图 4.5　历史数据详情截图

图 4.6　数据过滤截图

如图 4.6 所示，在数据过滤输入框中输入“云服务”，过滤掉了不包含该关键字的内容。

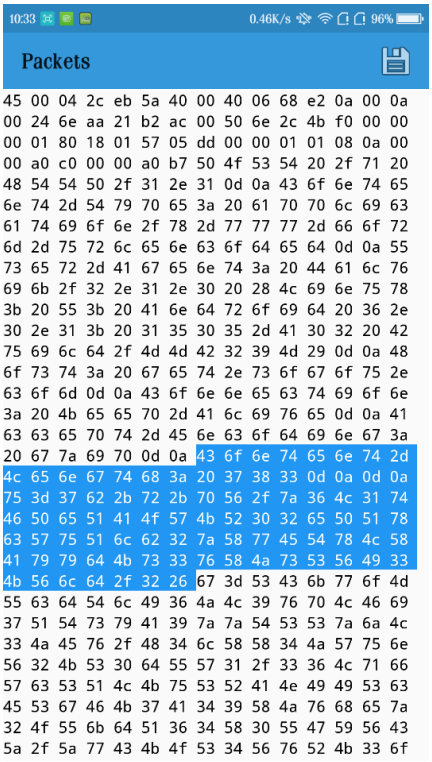

图 4.7　数据包查看截图

数据包查看如图 4.7，以 16 进制数据格式查看选中数据包的内容，右上角按钮点按可以进行保存单个数据包。

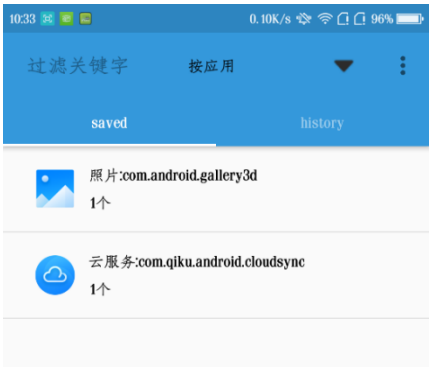

图 4.8　保存数据截图

保存的数据包如图 4.8，使用二级列表来展示，第一级只展示应用信息，点开后展示所有数据包。

第五章　系统测试

5.1 测试设计

5.1.1 测试环境

Windows 10，用于查看运行时 logcat 以及必要的断点调试。

腾讯云 CentOS，用于运行测试程序服务端。

安卓手机 360N4S，用于运行被测程序以及测试程序客户端。

5.1.2 测试范围

测试项整体可分为两类，其一为数据实时抓取的测试，需要在安卓手机上开启本软件并打开抓包，如表 5.1 所示。另一类为已有数据处理的测试，需要本软件已经有历史抓包数据，打开软件时无需再开启抓包即可进行测试，如表 5.2 所示。

表 5.1　数据实时抓取测试

测试项	测试目标
数据包抓取	确认可正确的抓取数据
数据包聚合	确认抓取数据与展示的应用信息正确对应
数据包存储和读取	确认产生的存储文件正确
数据包转发	开启抓包时，其他应用可正常发出数据
响应数据回写到网卡	开启抓包时，其他应用可正常进行网络交互
稳定性测试	开启抓包时，其他应用可正常运行

表 5.2　已有数据处理测试

测试项	测试目标
数据包过滤查看	确认列表可被正确的过滤显示
查看单个数据包	可正常跳转并展示单个数据包内容
数据包历史查看	可正确查看历史抓包记录
数据包单条保存	保存数据包文件正确且能够显示在保存列表中

5.2 测试用例及测试记录

表 5.3　测试记录

测试项	步骤	输入	预期结果	实际结果
数据包抓取	开启抓包、运行测试程序、运行其他第三方程序	点击开始抓包按钮	主界面列表刷新，出现抓包数据，且包含有测试程序指定的数据包	列表成功刷新，抓包数据正确显示，可以找到测试程序指定数据

数据包聚合	开启抓包、运行测试程序、运行其他第三方程序。	点击开始抓包按钮	数据包均正确的按照应用、连接进行聚合	数据包正确的按照连接、应用聚合
数据包存储和读取	开启抓包、运行测试程序、运行其他第三方程序	点击开始抓包按钮	存储目录下有文件，且文件内容正确	存储目录下的文件内容均可用，且可正确读取
数据包过滤查看	在过滤输入框中输入关键字	在过滤输入框中输入关键字	列表中过滤掉不包含关键字的数据包	列表可正确过滤
查看单个数据包	点按单个数据包	点击列表中单个数据包	跳转到数据包查看页面，并正确显示该数据包	成功跳转，数据包显示无误
数据包历史查看	查看列表中内容	点按展开各级列表各级历史	历史数据都可正常显示	历史数据与文件存储结果一致
数据包单条保存	查看单个数据包，点按保存	点按列表中单个数据包，跳转后点击右上角进行保存，返回到保存列表中	保存列表中出现保存的数据包，且内容无误	保存列表中成功添加，内容无误
数据包转发	开启抓包、运行测试程序	开启抓包，运行测试程序	测试程序服务端可收到数据	测试程序服务端成功收到数据
响应数据回写到网卡	开启抓包、运行测试程序	开启抓包，运行测试程序	测试程序客户端可与服务端正常交互	测试程序客户端成功与服务端交互
稳定性测试	开启抓包、运行测试程序、运行其他第三方程序，重复多次	开启抓包，重复运行其他程序	所有程序都可正常运行	有偶现崩溃，偶尔会有网络卡顿

5.3 测试结果及结论

5.3.1 测试用例执行情况

表 5.4　测试执行情况

测试项	测试结论
数据包抓取	正确
数据包聚合	正确
数据包存储和读取	正确
数据包过滤查看	开启抓包且过滤正在动态更新的数据时，较大可能出现崩溃
查看单个数据包	无
数据包历史查看	无
数据包单条保存	无

数据包转发	无
响应数据回写到网卡	无
稳定性测试	在其他应用每分钟内发起连接的个数大于 1000，且此时正在查看动态更新的数据列表时，有较小可能出现崩溃

5.3.2 软件缺陷分析

目前项目中存在的问题主要在数据包过滤和复杂环境下的稳定测试中。数据包过滤时的崩溃主要源于对界面列表的动态改变，由于列表是一个自定义的三级列表，过滤时可能暴露出该自定义控件的一些问题而崩溃，目前正在持续修正中。复杂环境下的稳定性测试中，主要是由于多线程原因，各个集合的增添异步访问处理不当。由于性能的要求，这种情况不能一味的加锁同步，需要对数据结构本身以及访问方式做调整，目前还在定位中。

5.3.3 测试结论

本项目已经可以满足测试环境下正常进行抓包、读写、查看等关键操作，但是稳定性仍有待提升，在复杂情况下偶尔会有崩溃发生。在实际使用中偶尔还会有网络异常的情况，这些问题还需进一步进行定位。

第六章　总结与展望

6.1 本文工作总结

本项目在探索打破传统移动端抓包方式的过程中，最终选择了借助安卓 VPN 机制来进行实现，不仅突破了只能抓取 HTTP 数据的限制，比起传统的代理方式也更加稳定，还能够减少在实际开发测试过程中传统抓包环境搭建的繁琐过程，大幅度减少了抓包失败的可能性。

不过本项目目前还略有粗糙，距离真正应用在实际开发中、完全替代传统抓包方式还有很大的距离，比如运行时仍旧有偶发的崩溃。而且当前来看，用户界面上还有很大的不足，在开发中，手机端比起电脑端的操作还是略有不便，关于如何展示数据包这方面还有非常大的改进空间。

6.1 未来工作展望

目前本项目只实现了最基本的网络抓包功能，实际上还有许多可以扩展的地方。

（1） 自定义规则响应。

不论是 Windows 上的 Fiddler，还是 Mac 上的 Charles，除了单纯的抓包功能外，还具有自定义设置网络响应的功能。它们可以通过一定的设置，让开启了抓包的机器在发现机器发出了某个网络请求时，拦截并回复一段事先指定的数据，甚至还可以通过高级配置，来对实际外界响应做一定的处理再回应给机器本身。这一功能在实际软件开发中还是有很大的用处的，利用该功能完全可以在不改变实际服务端代码或数据的情况下，来针对单个客户端改变服务端的逻辑。

（2） 自定义规则配置分享。

自定义响应的高级功能其实并不是人人都能够有兴趣去手动设置的，所以这一套设置应该具有分享机制，用户可以将自己构建的具有某个功能的自定义规则分享到线上，供其他人直接下载使用。

结束语

本文针对传统 Android 移动端抓包方式存在的不稳定、兼容性等问题，提出了直接在 Android 端上基于 VPN 实现抓包程序的解决方案，并进行了实现与实际应用，有效地便捷了 Android 移动端抓包的操作流程与环境搭建，提高了兼容性。从理论上，本文对于 VPN 的应用以及相关原理的探索，为今后该方面的研究提供了参考；且本项目在实际开发调试中也具有很强的现实意义。

今后本项目还会持续修改已有的问题，如偶现崩溃、导致的网络延迟等，持续优化，并预计加入更多功能（如自定义规则处理），进一步扩大应用范围。

致　谢

参考文献

[1] 易敏捷.软件测试国内外发展现状及趋势研究[J].电脑知识与技术,2013,9(26):6020-6022.

[2] 刘泓杉.关于软件测试在 Web 开发中的运用探讨[J].电脑迷,2018(12):68.

[3] Fall, Kevin R., and W. Richard Stevens. TCP/IP illustrated, volume 1: The protocols[M]. addison-Wesley, 2011:579-804.

[4] Belshe, Mike, Roberto Peon, and Martin Thomson. Hypertext transfer protocol version 2 (http/2)[J]. No. RFC 7540. 2015:4-17.

[5] Freier, Alan, Philip Karlton, and Paul Kocher. The secure sockets layer (SSL) protocol version 3.0[R]. No. RFC 6101. 2011.

[6] Sanders, Chris. Practical packet analysis: Using Wireshark to solve real-world network problems[M]. No Starch Press, 2017:53-63.

[7] Shah K. Penetration Testing Android Applications[J]. online] Whitepaper, Foundstone, a division of McAfee, 2011:10-11.

[8] 蒋月华.VPN 技术的高级应用[J].现代信息科技,2018(12):176-181.

[9] 陈双宝. Android 系统的 VPN 客户端的研究与实现[D].华北电力大学,2013.

[10] Deering, Steve, and Robert Hinden. Internet protocol, version 6 (IPv6) specification[J]. No. RFC 8200. 2017:4-7.