摘 要

作为网络嗅探的重要方式之一，数据包的捕获与还原技术一直以来都受到网络监管人员的广泛关注。针对不同的网络协议，还原的方式各不相同，其中超文本传输协议（HTTP）是一种能够在浏览器和服务器之间传输 HTML 等网络资源的应用层协议，是在 Web 上进行数据交换的基础。本文围绕网络数据包的捕获和HTTP协议的还原技术展开探讨，旨在为HTTP协议设计和实现一套特殊的还原系统。

本系统由数据包捕获、HTTP协议识别和HTTP协议数据还原三个部分组成，能较准确地识别、较高质量地还原HTTP协议，同时附加实现了浏览器和服务器之间的交互过程分析，数据包的基本信息展示，网络流量过滤器的设置等功能，向用户清晰地展现Web上的数据交换过程，增强用户体验感。

具体的实现方法是借助WinPcap库从链路层抓取数据包，自底向上解析该数据包中的各层协议的首部，提取出上层协议类型、IP地址、端口号、字节流序号等关键字段的信息。针对HTTP协议的特点，结合首部信息，设计算法完成识别功能。针对不同的网络情况，设计算法对数据数据包进行排序和重组，从而还原完整的HTTP协议实体部分，并用C++语言编程实现算法，图形用户界面基于MFC的对话框应用程序框架编写。

**关键词：**数据包捕获；HTTP协议识别；HTTP协议还原；C++；MFC

目 录

[摘 要 I](#_Toc102576538)

[目 录 II](#_Toc102576539)

[第一章 绪 论 1](#_Toc102576540)

[1.1 HTTP协议还原系统介绍 1](#_Toc102576541)

[1.1.1 历史与发展 1](#_Toc102576542)

[1.1.2 软件使用方式 1](#_Toc102576543)

[1.2 Winpcap库介绍 1](#_Toc102576544)

[第二章 系统分析 3](#_Toc102576545)

[2.1 功能分析 3](#_Toc102576546)

[2.1.1 基本功能 3](#_Toc102576547)

[2.1.2 拓展功能 3](#_Toc102576548)

[2.2 程序结构和类关系分析 3](#_Toc102576549)

[第三章 详细设计及实现 4](#_Toc102576550)

[3.1数据包捕获（Pkt\_capturer）类 4](#_Toc102576551)

[3.2 HTTP协议识别（Http\_identifier）类 4](#_Toc102576552)

[3.3 交互过程分析类 6](#_Toc102576553)

[3.4 数据包还原（Http\_reverter）类 6](#_Toc102576554)

[3.5 请求协议还原（Request\_reverter）类 12](#_Toc102576555)

[3.6 应答协议还原（Respond\_reverter）类 12](#_Toc102576556)

[3.7 图形用户界面 12](#_Toc102576557)

[第四章 测试 14](#_Toc102576558)

[4.1 测试环境 14](#_Toc102576559)

[4.1.1](#_Toc102576560)[测试的硬件环境 14](#_Toc102576560)

[4.1.2 测试的软件环境 14](#_Toc102576561)

[4.2 界面展示 14](#_Toc102576562)

[4.3 交互过程分析 15](#_Toc102576563)

[4.4 数据包还原效果 16](#_Toc102576564)

[4.4.1请求协议的还原效果 16](#_Toc102576565)

[4.4.2 应答协议的还原效果 18](#_Toc102576566)

[4.5错误测试 21](#_Toc102576567)

[4.5.1输入错误测试 21](#_Toc102576568)

[4.5.2网络或过滤器异常 22](#_Toc102576569)

[4.5.3文件读写异常 22](#_Toc102576570)

[4.6 小结 23](#_Toc102576571)

[参考文献 24](#_Toc102576572)

第一章 绪 论

* 1. HTTP协议还原系统介绍
     1. 历史与发展

随着互联网技术的普及，网络已成为人们日常生活中不可或缺的一部分。通过网络包还原出网络协议来进行网络内容管理可以有效减少网络中的负面信息的影响。网络包协议的还原包括包捕获、协议识别和软件级数据恢复三个部分。

目前，有许多相关产品用于还原网络数据包，例如Wireshark，Nmap，Snort，NTOP等。然而，随着网络传输容量和数据量的迅速增加，网络管理者对网络包协议还原系统的功能和性能提出了新的要求[1]。

* + 1. 软件使用方式

首先点击按钮获取设备列表，选择要监听的设备和是否使用过滤器，然后点击提交按钮开始抓取数据包，等待抓取完成后，通过按钮选择使用交互分析功能分析浏览器和服务器之间的交互过程，或使用协议还原功能进行HTTP协议的还原。

* 1. Winpcap库介绍

WinPcap是Windows环境中用于链路层网络访问的工具，其功能包括：绕过协议栈捕获和传输网络包，内核级包过滤，网络统计引擎和对远程包捕获。

WinPcap由三个模块组成，第一个模块NPF(内核级的数据报过滤器)是一个虚拟设备驱动程序文件，功能是过滤数据包，并把这些数据包传给用户态模块。第二个模块Packet.dll(数据包低级驱动程序库)为Win32平台提供了一个公共的接口，可执行如获取网卡名或者机器掩码等低级的操作。第三个模块 Wpcap.dll(数据包高级驱动程序库)是不依赖于操作系统的。它提供了更高层的函数调用，诸如产生过滤器、用户级缓冲以及包注入等高级功能[2]。

由于它的一系列优良特性，WinPcap已被广泛应用到各类开源和商业网络工具中，如协议分析器、网络监视器、网络入侵检测系统、嗅探器、流量生成器和网络测试器均有使用WinPcap作为包捕获和过滤引擎。

图1-1 WinPcap的组成架构图

第二章 系统分析

2.1 功能分析

2.1.1 基本功能

1.从链路层捕获数据包

2.识别HTTP协议

3.还原HTTP协议

2.1.2 拓展功能

1.设置网络流量过滤器

2.展示客户端和浏览器的交互过程

3.保存并展示数据包内容

4.查询数据包的相关信息

2.2 程序结构和类关系分析

整个项目大致分为main函数，Pkt\_capturer（数据包捕获）类，Http\_identifier（HTTP协议识别）类，Http\_reverter（数据包还原）类，Analyser交互分析类，数据包还原类有两个子类分别是请求协议和应答协议的还原类。

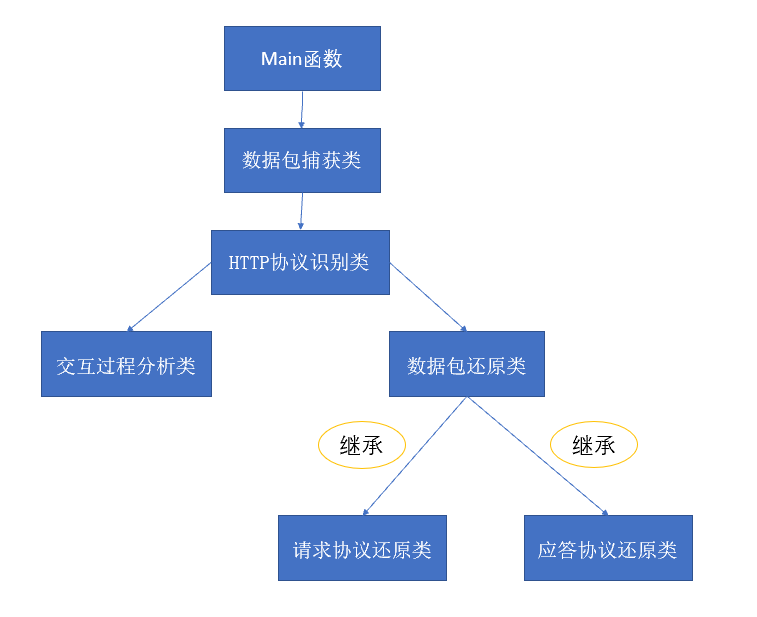
系统结构如下图（图2-2）所示，图中没有文字标注的箭头仅表示创建和使用实例的顺序，指示程序运行的流程，不表示类之间的继承关系。

图2-1 系统结构

第三章 详细设计及实现

3.1数据包捕获（Pkt\_capturer）类

该类负责完成数据包的捕获和保存，具有三个方法分别实现以下三种功能：

1）get\_device()选择要监听的设备。首先利用WinPcap的库函数 pcap\_findalldevs\_ex()获取网络适配器的列表，然后等待用户输入想要监听的设备编号，打开该适配器，释放列表。

2）set\_filter()设置流量过滤器。WinPcap有强大的过滤引擎，可通过筛选表达式对数据包的协议类型，端口号等信息进行筛选，该函数中使用"tcp port 80"来筛选出使用HTTP协议的数据包，如果用户想要减少干扰、节省时间，可以开启过滤器。

3）save\_packet()捕获并转存数据包。使用pcap\_loop()函数捕获数据包，回调函数packet\_handler()处理数据包，由于捕获数据包的速度很快，来不及立刻分析每个数据包，只能先将数据包转存到文件中，捕获结束后再依次读取分析。读取数据包时使用pcap\_next\_ex()函数循环读取，该函数比[pcap\_loop](https://www.winpcap.org/docs/docs_412/html/group__wpcapfunc.html" \l "ga6bcb7c5c59d76ec16b8a699da136b5de" \o "Collect a group of packets.)()的可读性好，且使用更灵活，不需要回调处理。

3.2 HTTP协议识别（Http\_identifier）类

该类负责识别HTTP协议。

HTTP协议使用80端口，TCP连接，即运输层使用TCP协议，网络层使用IP协议，链路层使用MAC协议，HTTP的首部有HTTP版本号字段，其中含有“HTTP”字符串。根据HTTP协议的这些特性设计识别算法，由于WinPcap是从链路层捕获数据包，因此需要从链路层开始向上，层层解析首部并检验关键字段，算法分为以下几个步骤：

1. 解析MAC协议，如果协议类型字段为“0x0800”，表示网络层使用IP协议，则进行下一步，否则说明不使用HTTP协议。
2. 跳过MAC首部，解析IP协议，如果协议类型字段为“6”，表示传输层使用TCP协议，则进行下一步，否则说明不使用HTTP协议。
3. 跳过MAC和IP首部，解析TCP协议，如果源端口号或者目的端口号字段为“80”，说明使用80端口，则进行下一步，否则说明不使用HTTP协议。
4. 跳过MAC、TCP、IP首部，匹配HTTP首部中的“HTTP”字符串，若能匹配则可确定为HTTP协议，否则说明该数据包中没有HTTP首部或者该数据包不属于使用HTTP协议会话的内容。算法流程如图3-1所示。

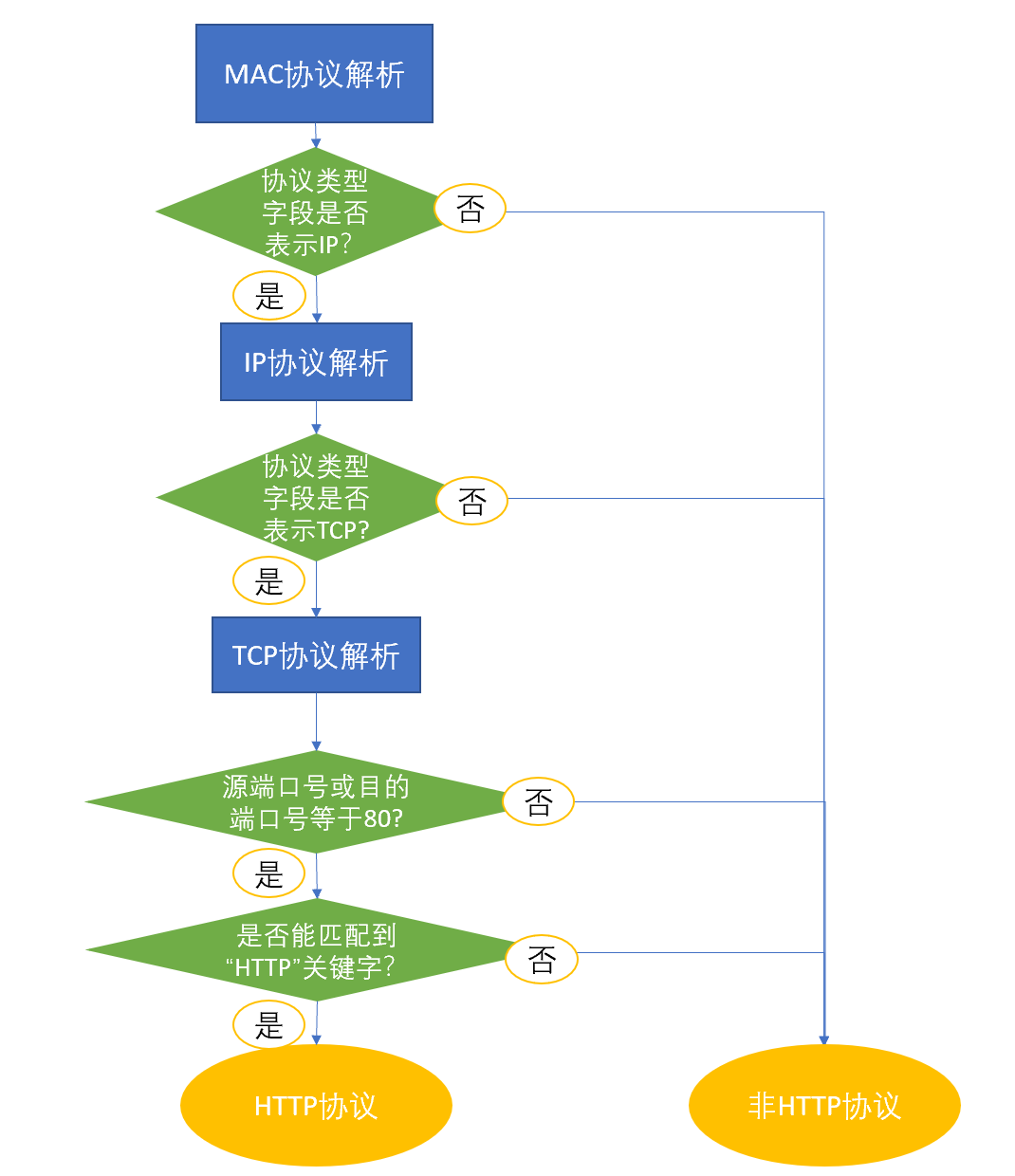
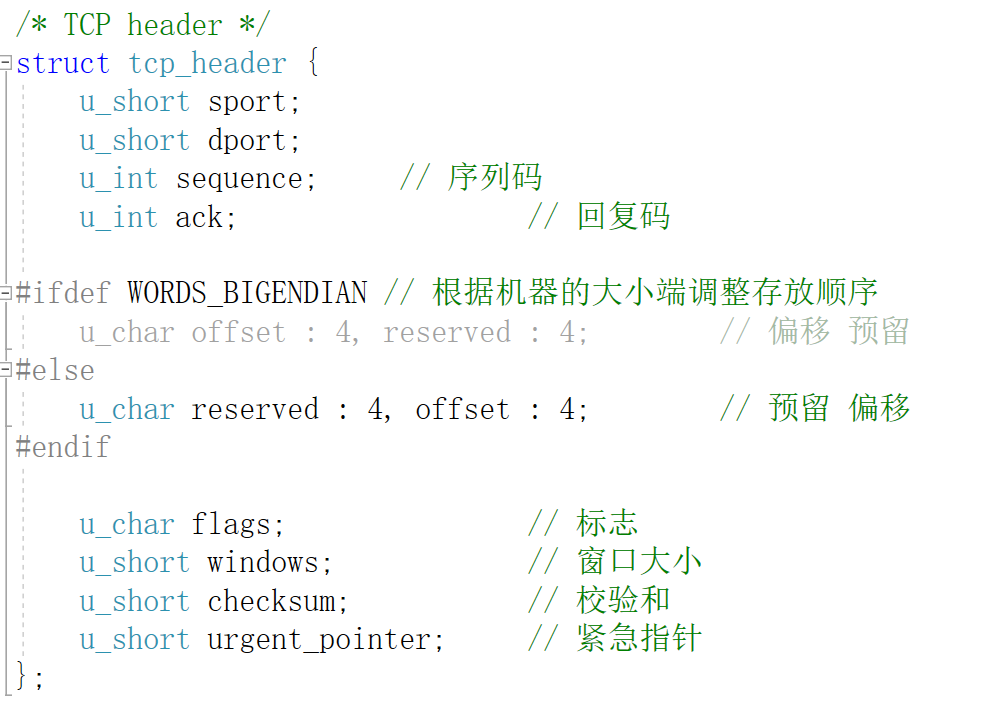
该类具有三个方法分别实现MAC协议，IP协议，TCP协议首部的解析并根据首部中的字段判断应用层是否使用HTTP协议，解析的具体方法就是用和字段长度相等的变量来保存该字段的内容，用结构体来集合这些变量，例如描述TCP协议首部的结构体即为代码3-1所示。解析的过程中还需要将TCP序列号、源IP地址、目的IP地址、TCP标志等关键信息保存在记录数据包信息的pkt\_node结构体中，便于后续还原协议时使用。

图3-1识别HTTP协议的算法流程图



代码3-1 保存TCP协议首部的结构体

3.3 交互过程分析类

该类通过分析连续捕获到的多个数据包的一系列信息，向用户展示浏览器和服务器之间的交互过程。

分析的内容包括：数据包的捕获时间，数据包的总长度，源IP地址和目的IP地址等网络层信息，TCP协议的首部长度、TCP字节流序号和TCP标志等运输层信息，HTTP版本，HTTP请求协议的方法、URL，HTTP应答协议的状态码和修饰符等应用层信息。

3.4 数据包还原（Http\_reverter）类

该类的主要功能是实现HTTP协议的还原，有请求HTTP还原和应答HTTP还原两个子类。该类包括六个关键的方法，分别实现类初始化，重组数据包，保存HTTP协议首部，解析HTTP首部，保存HTTP实体部分，展示还原出的关键信息六项功能，由于HTTP请求协议和应答协议的首部结构略有区别，因此需要在子类中用不同的方法实现两种协议首部的解析和保存，这两个方法在该类中被定义为虚方法，在子类中被重写实现。该类的定义如代码3-2所示。

代码3-2 HTTP协议还原类的定义

下面依次分析该类中关键方法的具体实现：

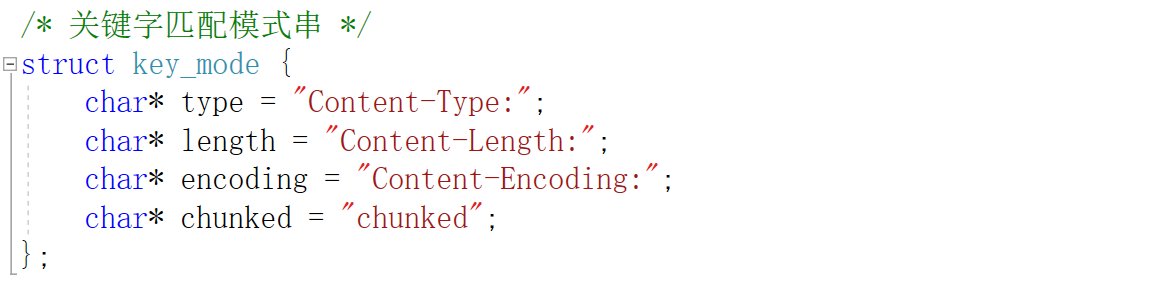
1. HTTP首部的解析和保存，save\_head和http\_head\_parse函数。

HTTP首部和实体之间有两个’\r\n’分隔，依次读取并保存数据包中的内容，直到出现’\r\n\r\n’就停止读取，并记录首部的长度。

HTTP首部采用ASCII编码，可读性好，不需要再做解码处理，HTTP首部各个字段的长度不固定，无法采用解析IP和TCP首部的方法（直接用长度匹配的变量保存各个字段的内容）来解析HTTP首部，只能采用匹配关键字定位关键信息，然后再保存关键信息的方式。由于匹配的同时还需定位，且首部内容一般不长，关键字占比较多，BM算法，Trie树等匹配算法的操作过于复杂，因此本文直接采用依次读取字符串并匹配的方式。

匹配算法如下：

1. 使用输入流读取保存HTTP协议首部的文件，依次读入字符串，
2. 检查是否和关键字匹配
3. 读到文件尾部的结束标志，匹配结束

代码3-3 HTTP首部关键字结构体图2-2 系统结构

1. 数据包的重组，http\_handling函数。由于WinPcap是从链路层捕获数据包，为了简化处理，设置pcap\_datalink(adhandle) != DLT\_EN10MB，仅捕获使用10MB带宽的链路传送的以太网帧，捕获到的实质上是去掉了前同步码和CRC校验位的帧，在本文中简称为数据包，由于帧能承载的数据大小有限制，运输层会对报文进行分段，因此需要重组捕获到的数据包才能还原出服务器和浏览器之间传送的完整对象。运输层使用TCP协议时，TCP协议为了提高效率，一般会控制分段的长度来避免在网络层再进行分片，本系统只针对HTTP协议，重组数据包时只考虑了分段的重组，不考虑分片的重组。

考虑以下三种网络环境下的传输情况[3]：

1. 在理想的网络环境下，捕获到的相邻两个数据包属于同一个HTTP协议，且字节流序号连续。即一个HTTP协议的数据包传输完后，再传输下一个数据包的内容，不同协议的数据包不会混杂传输，且同一个协议内的数据包是按顺序传输的。例如网络对HTTP协议A、B、C……的传输顺序是如下所示：

A1，A2，A3，……，An，B1，B2，B3，……，Bn ，C1，C2，C3,……,Cn

针对这种情况，我们只需按顺序读取数据包，然后依次保存，并利用之前从TCP协议首部中获得的字节流序号和从HTTP首部中获得的实体内容长度等信息判断重组是否完成即可。这种理想情况一般不会出现在常见现实环境中，只会出现在实验环境中，或网络清闲的局域网中，但可以先基于这种情况设计重组算法，再针对复杂情况进行改进。

1. 在正常的网络下，假设没有丢包和乱序的情况，可能有多个HTTP协议在同时进行传输，此时不同协议的数据包可能会交叉混杂传输，但是同一个协议内的数据包还是按顺序传输的。例如网络对HTTP协议A、B、C……的传输可能如下所示:

A1，A2，C1，A3,……An，C2，B1，B2，C3，……，Cn，B3，……，Bn

在这种数据包交叉的情况下，我们需要借助之前从IP协议首部中获得的源IP和目的IP地址，源端口号和目的端口号等信息判断数据包是否属于同一个HTTP协议。

1. 在复杂的网络下，会出现乱序、重传、丢包的情况，此时不同协议的数据包可能会交叉混杂传输，同一个协议内的数据包的传输顺序也可能发生混乱，例如网络对数据A、B、C……的传输可能如下所示:

A1，An，A2，……，C1，B1，B2,C2，……，Cn, B3，……，C3，Bn

这时要考虑使用链表来暂存失序的数据包，设计如图3-2结构的链表

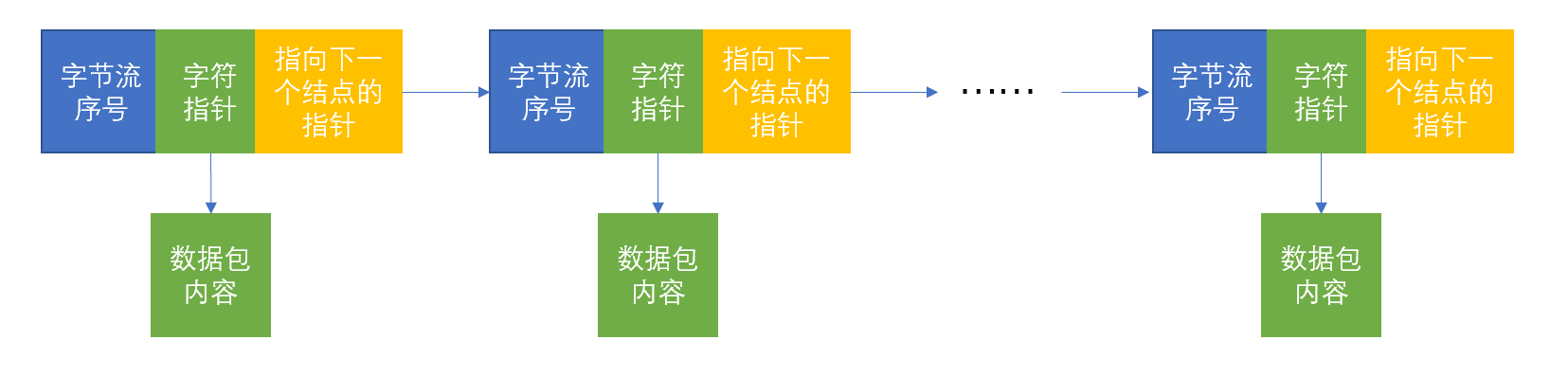
链表的结点由字节流序号（记录数据包的TCP字节流序号），指向数据包内容的指针和指向下一个结点的指针组成，插入结点时维护链表中结点的字节流序号递增，方便进行查找。

图3-2 暂存失序数据包的链表

综合考虑以上三种情况后设计出的重组算法的流程如图3-2所示，文字描述如下：

算法采用mark变量记录HTTP协议的还原是否完成，mark=0表示还未开始还原或上一次还原已完成，mark=1表示正处在还原的过程中。识别到HTTP首部后就开始协议还原。使用定义在循环外的结构体key\_now保存还原中需要用到的关键信息，如Content-length，首个数据包的TCP序列号，首个数据包的源IP和目的IP地址等。

此算法不能进行多个协议的并行还原，运行一次算法只针对一个HTTP协议进行还原。

1） 读取文件中转存的数据包。

2） 判断数据内容中是否包含关键字“HTTP”，再判断mark为0还是为1,，包含“HTTP”且mark=0则说明该数据包中有HTTP首部内容，且没有其他还原过程再进行中，可以启动还原过程，转移到第三步；如果包含“HTTP”且mark=1说明上一个协议还未完成还原时又出现了新的首部，有其他协议的数据包混入了正在还原的协议，因此跳过该包，读取下一个数据包；如果不包含“HTTP”且mark=0，没有读取到HTTP首部无法开始还原，则转移到第一步，读取下一个数据包；如果“HTTP”且mark=1，说明还原过程正在进行，该数据包的内容可能是某个HTTP协议的主体部分，转移到第四步。

3） 将HTTP首部内容保存到相应的路径中，利用上述匹配算法提取出首部中关键信息并保存在变量和key\_now结构体中，并将mark设置为1。然后判断该数据包中是否还有协议实体部分的内容，如果有则保存到指定路径中，并转移到第五步，如果没有则直接转移到第五步。

4） 判断该数据包是否属于正在还原的协议以及顺序是否正确，如果不属于或不正确则进行乱序处理，属于且正确就直接将该数据包的HTTP协议部分追加写入保存协议实体的文件中。

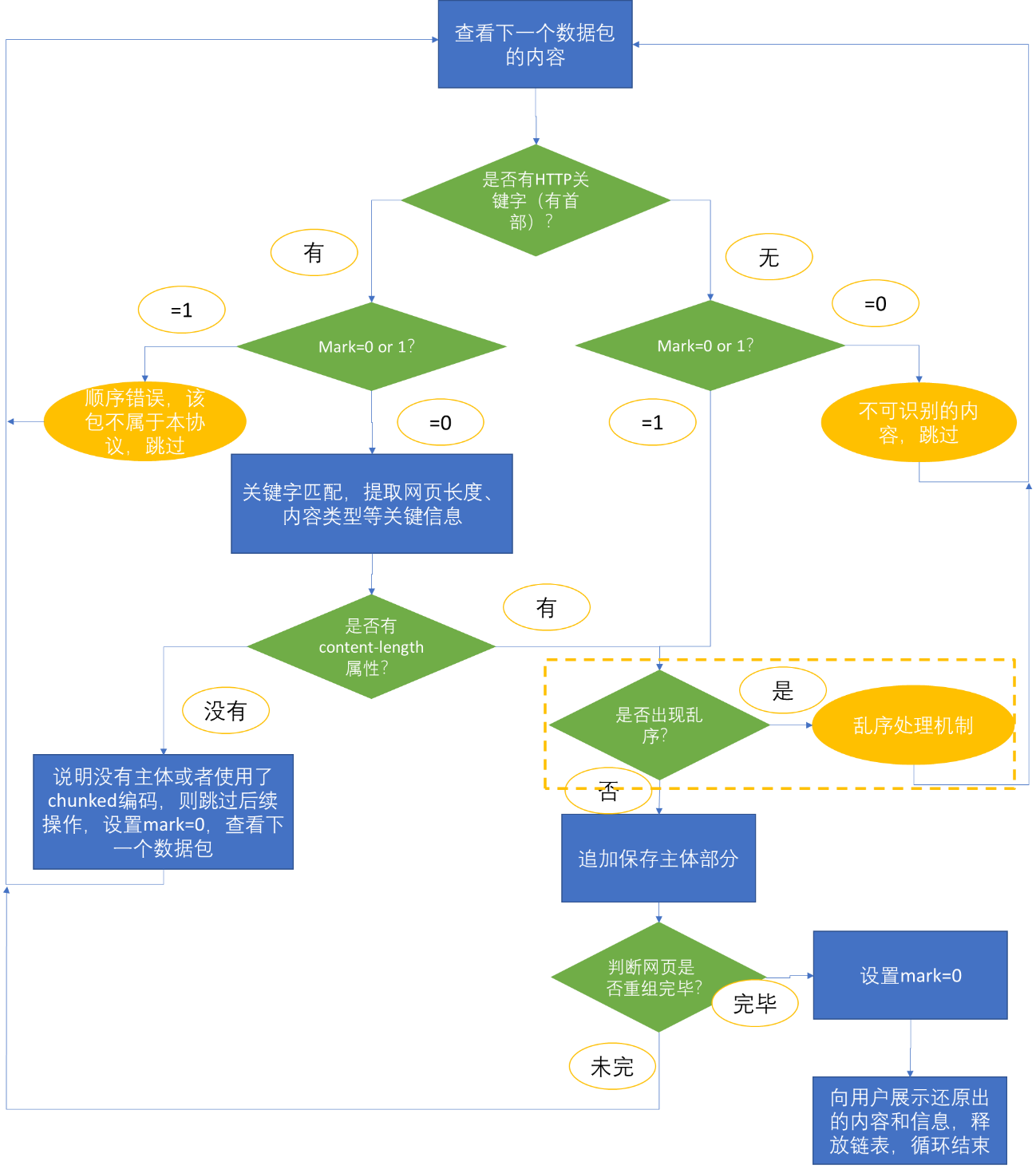
5） 判断协议是否还原完毕，如果完毕则展示出还原后的信息，算法结束。如果未完则转移到第一步，处理下一个数据包。

图3-3 数据包重组算法流程图

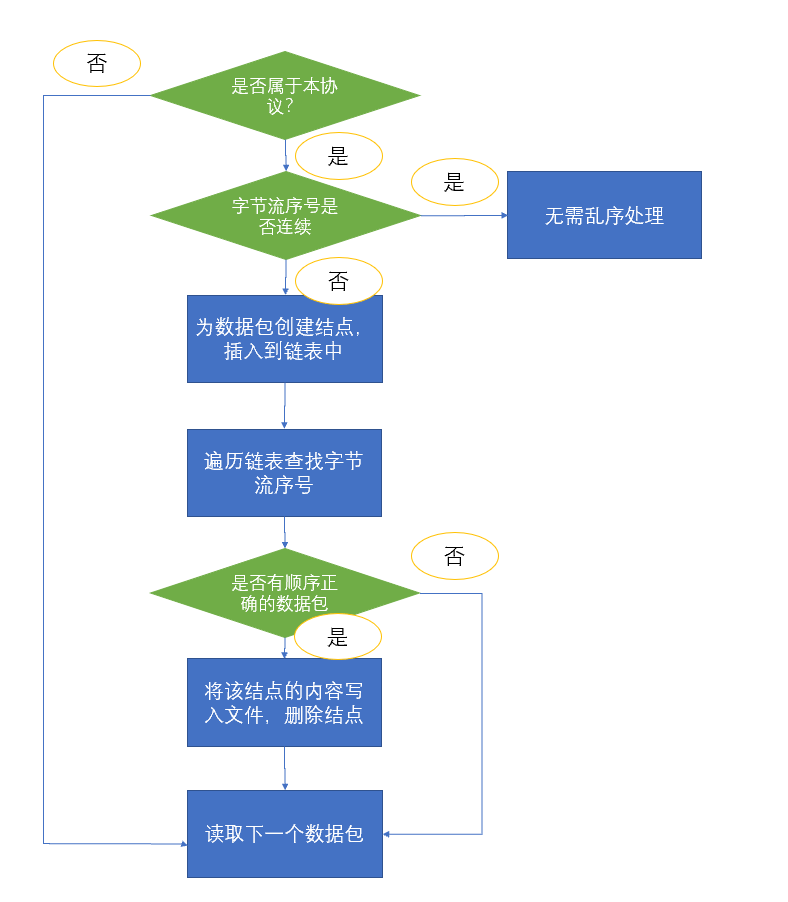
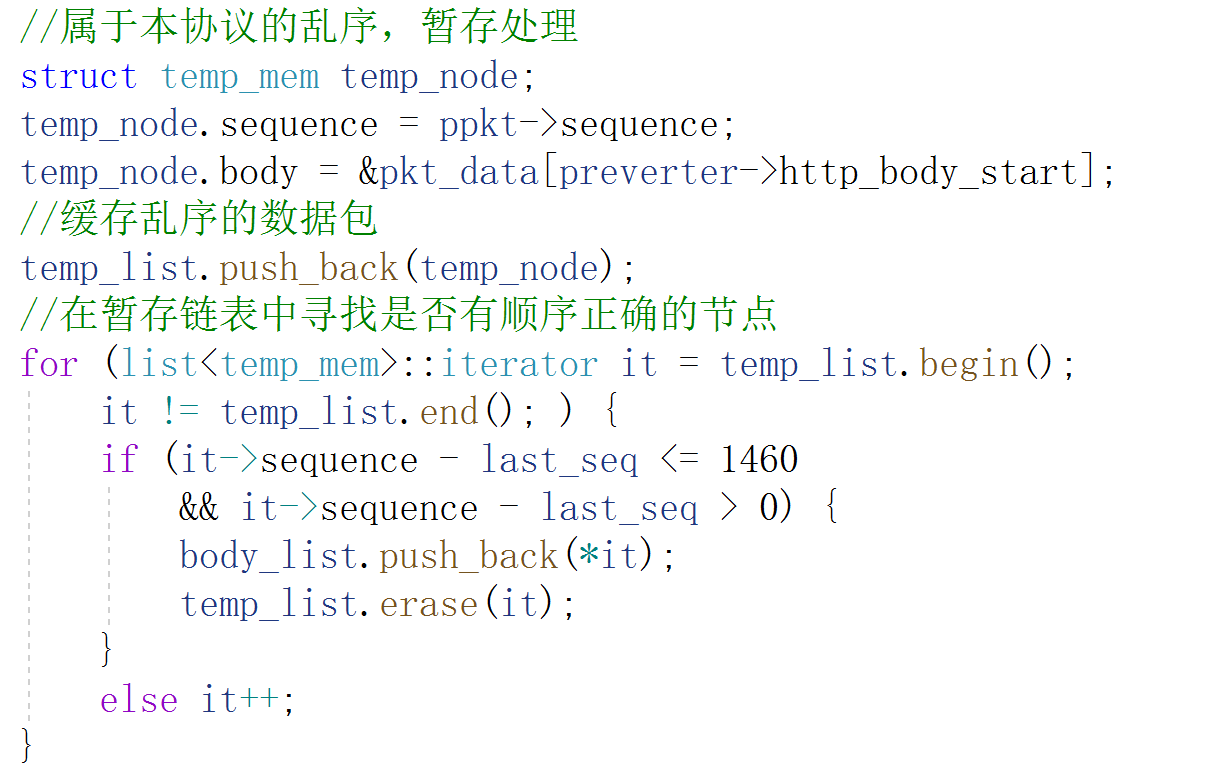
乱序判断和处理的具体方法：借助HTTP首部的Content-length字段可推断出属于该协议的数据包的TCP字节流序号的范围，然后判断该数据包和还原的首个数据包的源IP、目的IP，源端口号、目的端口号是否一致，以及字节流序号是否在指定范围内，如果不是，则说明该数据包不属于正在还原的协议，应当跳过该数据包，读取下一个；如果均是，则说明属于，然后判断该数据包的TCP字节流序号和上一个数据包的TCP字节流序号之差是否大于0，小于等于1460（MSS），若是，则说明顺序正确，可以将数据包的内容直接写入文件，若顺序不正确，则先为该数据包创建一个新的结点，插入到暂存链表(temp\_list)中，然后到链表中查找是否有之前暂存的结点的字节流序号符合当前的顺序，如果有则将该结点中的数据内容写入文件，并删除该结点。为了避免某部分的数据包丢失或时延过长导致整个还原无法进行，使用cnt变量记录连续处理的失序数据包的个数，如果连续5个以上的数据包时序则放弃该协议的还原，设置mark=0，进行下一个协议的还原。流程图如图3-4所示。

图3-4 乱序处理算法的流程图

代码3-4 处理失序的数据包

判断是否完成的具体方法：HTTP首部的Content-length字段会提供整个HTTP协议中实体部分的长度，HTTP首部的长度在保存首部时已经求出，TCP首部的序号字段记录了数据段（跳过首部）中第一个字节在数据流中的位置编号，当前数据包的序号与还原的第一个数据包的序号之差就表示已还原的数据长度（包括首部）。已还原的数据长度+该数据包的HTTP协议部分的数据长度-HTTP协议首部长度<实体部分的总长度，说明还原还未完成。

代码3-5 检测到HTTP首部后的数据包处理

1. 实体部分的保存，save\_body函数，实体部分的内容主要分为image、application 、text三大类，application 、text都是文本类型，直接使用输出流写入文件即可，为了不破坏图片格式，image要采用二进制的方式写入文件。实体部分可能使用传输编码，如chunked等，此时首部中不再出现Content-length字段，实体消息分为实体头和实体，实体头定义的是与实体有关的信息，通过这些信息判断是否重组完成。对系统不对使用了传输编码的数据包的实体部分进行还原，仅保存其首部，并向用户展示关键信息。实体内容可能被加密和压缩，许多类型文件如GIF、JPEG和 MIDI 等均是已经压缩过的文件，而如HTML、文本文件是没有压缩的。HTTP/1.0虽支持压缩，但是没有提供足够的压缩机制，HTTP/1.1对压缩技术提供了全面的支持，大大提高了传输的效率。HTTP/1.1协议支持的压缩算法有，gzip压缩算法，compress压缩算法和 deflate压缩算法[4]。针对压缩文件，本系统将记录压缩方式，并保存未解压的实体部分。

当协议的实体部分较长时，每次读取一个存放实体内容的数据包都要进行文件读写操作，程序运行的效率较低。为了优化程序，采用链表（body\_list）按顺序暂存实体内容，结点的结构如图3-2所示，和暂存失序数据包的结点结构一致，还原结束后，再遍历链表将内容全部写入文件中。程序中的链表均使用C++ STL容器list实现。

3.5 请求协议还原（Request\_reverter）类

请求协议还原类继承于Http还原类，增加了字符串类型的成员变量method，URL和http\_version，分别用于保存首部协议第一行中的请求方法，URL和HTTP协议的版本三个关键字段。

实现了虚方法save\_head和http\_head\_parse，针对HTTP请求协议的首部结构进行保存和解析。

3.6 应答协议还原（Respond\_reverter）类

应答协议还原类继承于Http还原类，增加了字符串类型的成员变量http\_version，status和modifier，分别用于保存首部协议第一行中的HTTP协议的版本，状态码和状态修饰符三个关键字段。

实现了虚方法save\_head和http\_head\_parse，针对HTTP应答协议的首部结构进行保存和解析。

3.7 图形用户界面

还原系统的图形用户界面使用MFC基于对话框的应用程序框架编写，使用对话框作为用户界面，按钮和可编辑文本框作为主要控件，通过单击按钮触发事件处理程序，推动程序运行。运行出错时通过设置消息弹窗向用户显示相应的报错信息，并终止程序。

打开程序首先进入设置界面，设置好监听设备和网络流量过滤器后即可开始抓取数据包，抓包完成后跳转到主菜单界面，执行交互分析或协议还原功能。

程序中MFCApplication2Dlg.h和MFCApplication2Dlg.cpp是设置界面对话框的头文件和源文件，menu.h和menu.cpp是主菜单对话框的头文件和源文件，reverter.h头文件主要存放与协议还原功能先关的类、结构体和函数定义。

第四章 测试

4.1 测试环境

4.1.1测试的硬件环境

笔记本电脑联想小新Pro13，配置i5-10210U处理器、16GB内存（板载双通道）、512GB SSD、MX250显卡

4.1.2 测试的软件环境

操作系统：Win10 64bit

网络环境：安装WinPcap库

开发环境： Microsoft Visual Studio 2019集成开发环境，Visual Studio 2019 DebugWin32测试环境，安装MFC图形库和插件，使用MFC基于对话框的应用程序框架。

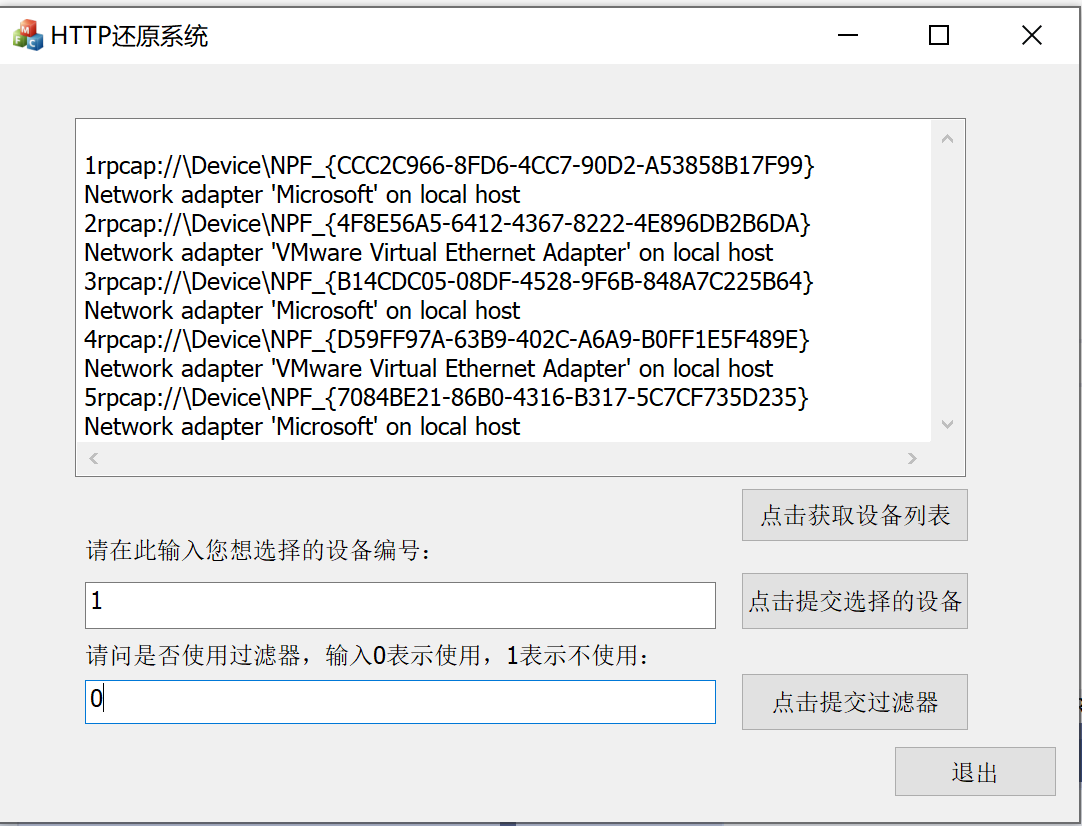
4.2 界面展示

图4-1 设置界面展示

4.3 交互过程分析

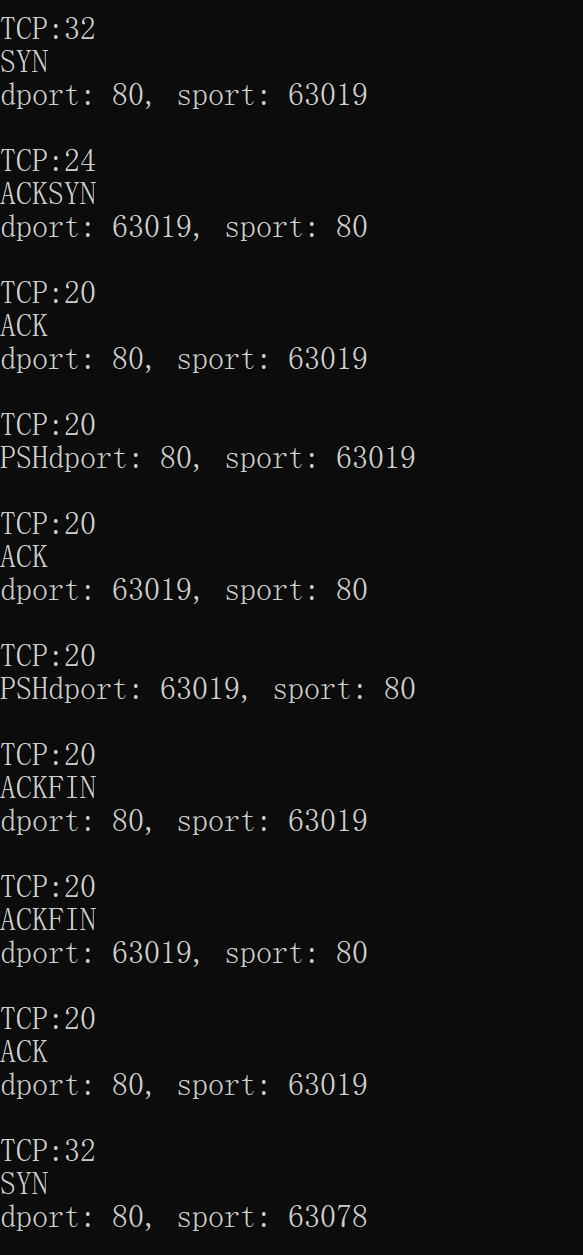
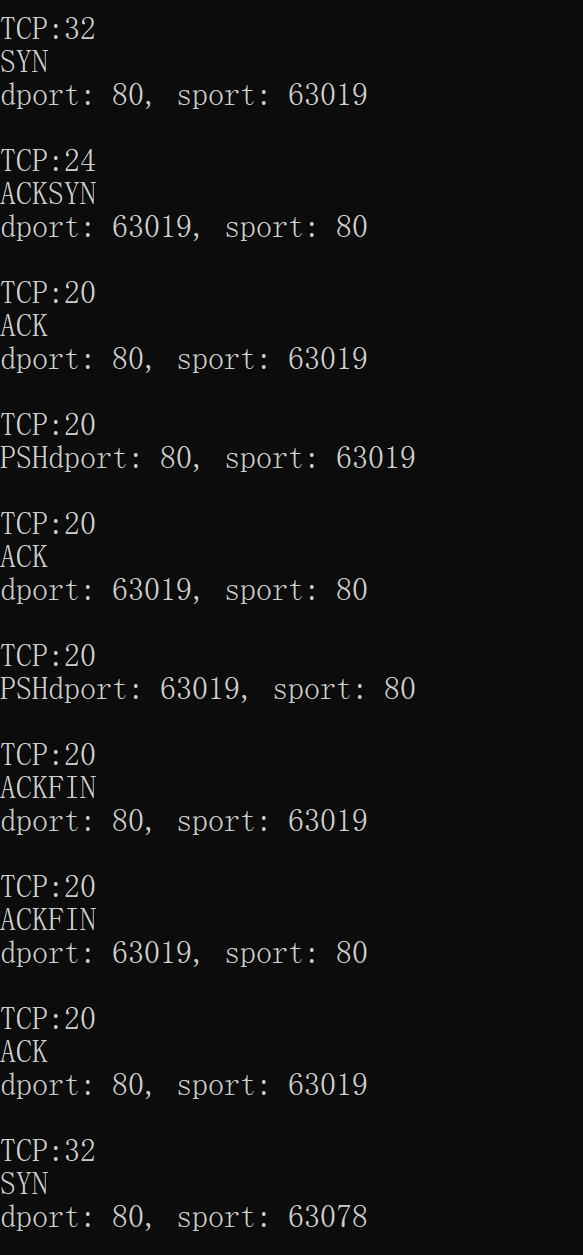
通过交互分析的可以清晰地展示服务器和浏览器交互创建TCP连接时的三次握手过程和结束传输时的四次分手过程。从图4-3的交互分析中可以看到在TCP协议中使用RST标志表示复位，用来异常的关闭连接。在发送RST包关闭连接时，不必等缓冲区的包都发出去，直接就丢弃缓冲区的包发送RST包。而接收端收到RST包后，也不必发送ACK包来确认。“异常的关闭连接”的情况在使用WinHTTP时Microsoft Windows HTTP Services时很常见。

图4-2 a)三次握手交互过程展示 b)四次分手交互过程展示

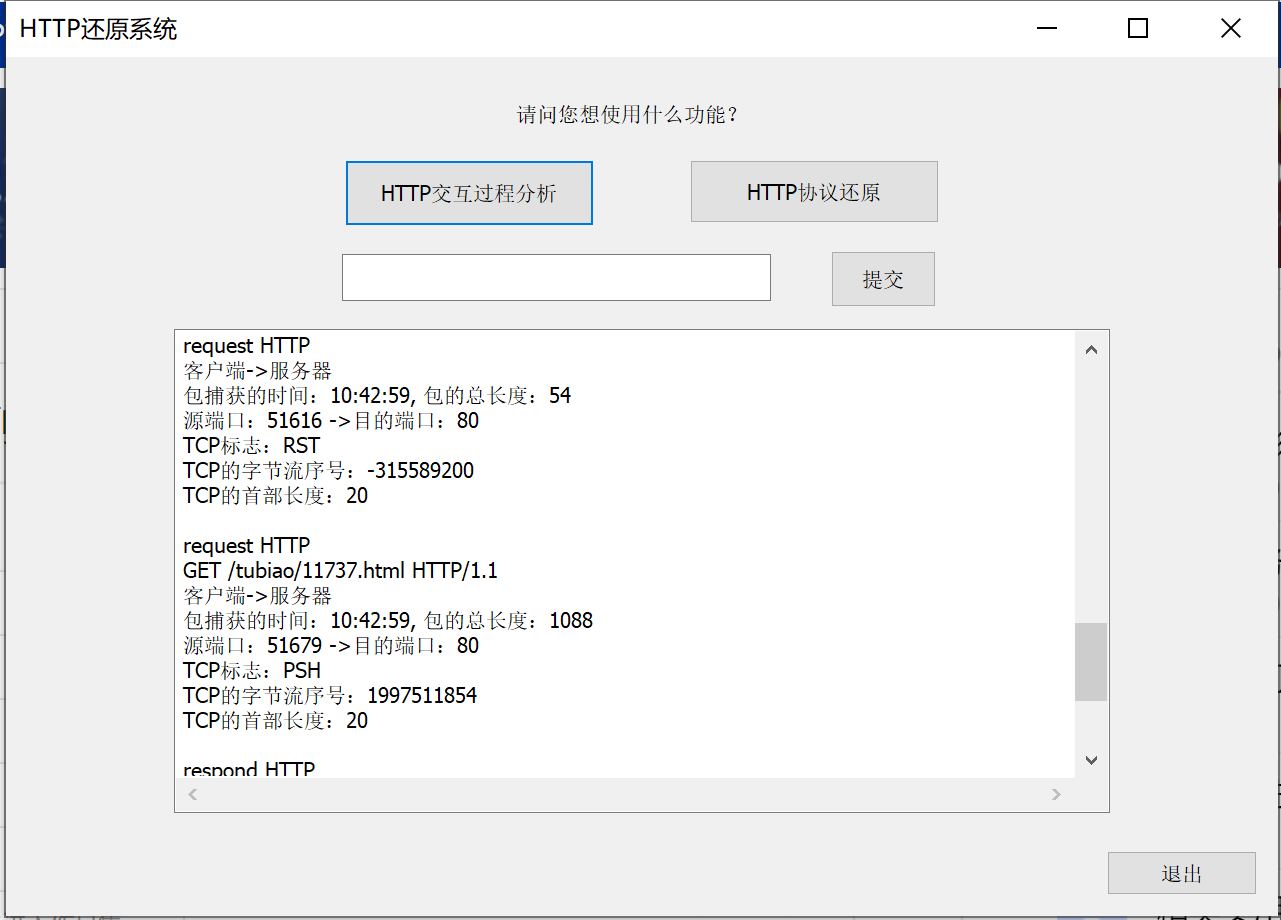


图4-3 WinHttp常用的“异常的关闭连接”

4.4 数据包还原效果

请求协议还原完成后会在文本框控件中向用户展示协议使用的HTTP版本、HTTP方法和请求的URL，应答协议还原完成后会展示HTTP版本、状态码和状态修饰符，如果有协议有实体部分则还会展示传输对象的长度、内容类型，是否使用了内容加密、压缩，是否进行了传输编码等信息。还原的协议首部将保存在request\_head.txt/respond\_head.txt中，实体部分保存在body.data（后续可根据内容类型更改拓展名）中。下面用还原出的文件和用户界面的截图来展示测试效果。

下面以[gobang-v2 (light7.cn)](http://gobang.light7.cn/" \l "/)网站为例展示测试效果，该网页是一个五子棋游戏的网页，网页界面展示如图4-4。

图4-4 测试网页的页面展示

4.4.1请求协议的还原效果

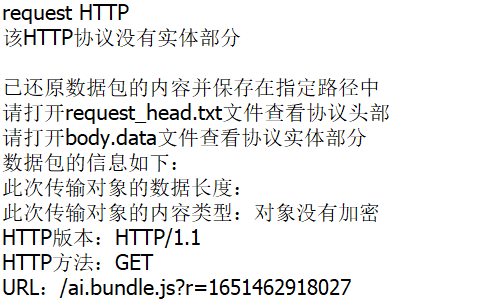
使用GET方法的请求协议还原完成后的用户界面如图4-5所示，由于使用GET方法的HTTP协议没有实体部分，因此传输的数据长度为空。保存在request\_head.txt文件中的请求协议首部如图4-6所示。

图4-5 HTTP请求协议还原完成后的相关信息展示（GET）

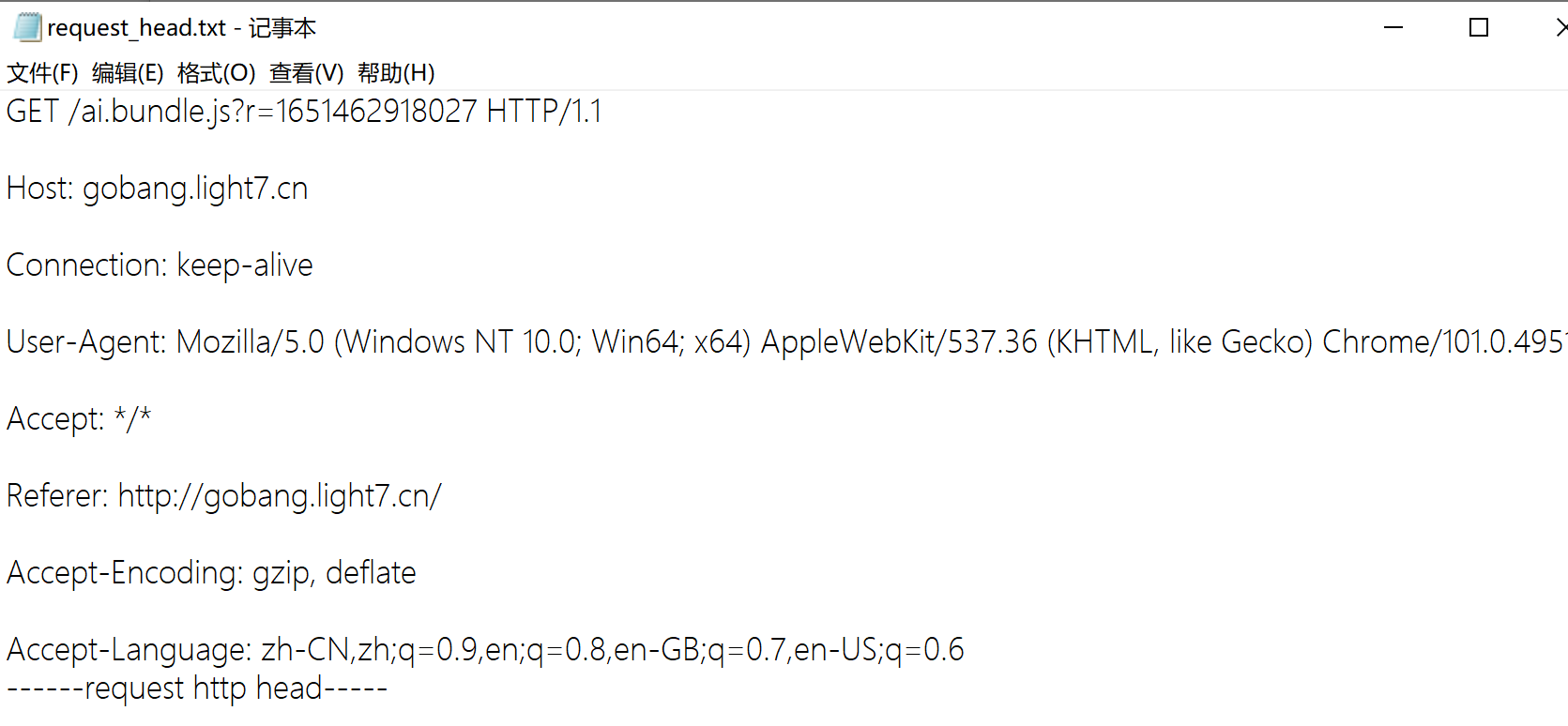


图4-6 保存在文件中的HTTP请求协议首部内容

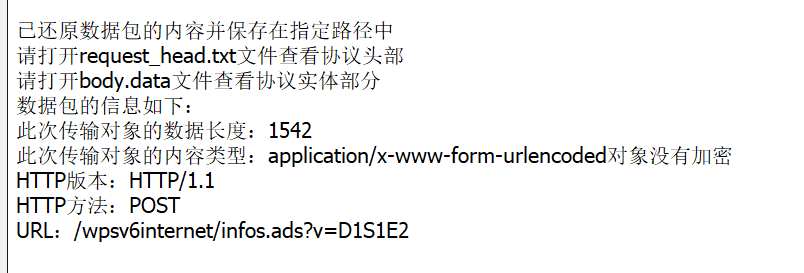
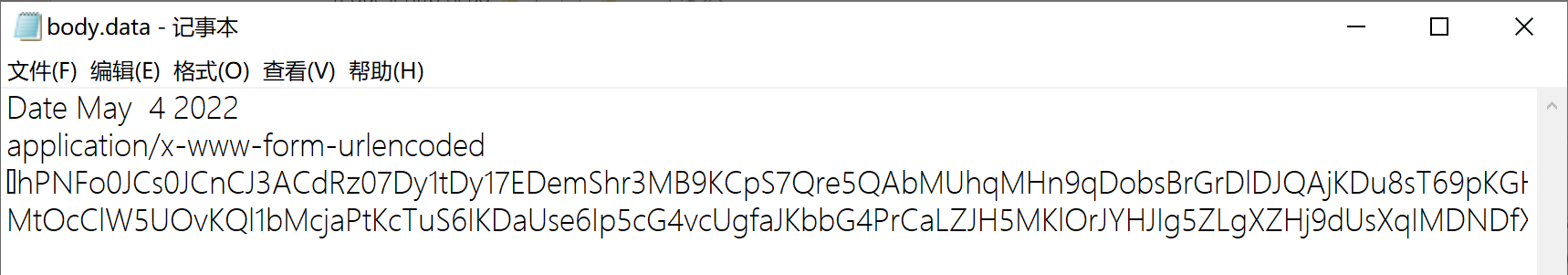
使用POST方法的请求协议还原完成后的用户界面如图4-7所示，协议实体部分的内容类型是application，保存在body.data文件中如图4-8所示。

图4-7 HTTP请求协议还原完成后的相关信息展示（POST）

图4-8 保存在文件中的请求协议实体内容（application）

4.4.2 应答协议的还原效果

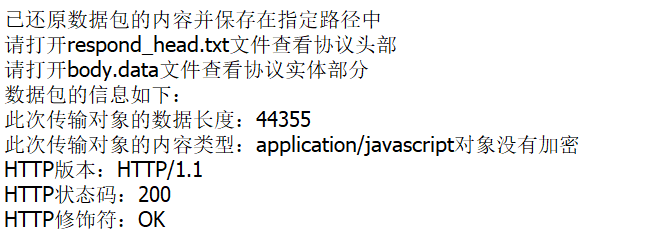
当应答协议实体部分的内容类型是application/JavaScript时，用户界面如图4-9，还原的首部内容如图4-10，实体内容如图4-11（截图展示的是没有修改过的还原结果，由于文件传输时没有包含换行符，阅读较困难）。测试网站是一个五子棋游戏网站，可以看出实体内容是五子棋游戏的程序代码，说明还原的内容与网页内容相符，还原出的信息正确。

图4-9 HTTP应答协议还原完成后的相关信息展示

图4-10 保存在文件中的应答协议首部内容

图4-11 保存在文件中的应答协议实体内容（application/JavaScript）

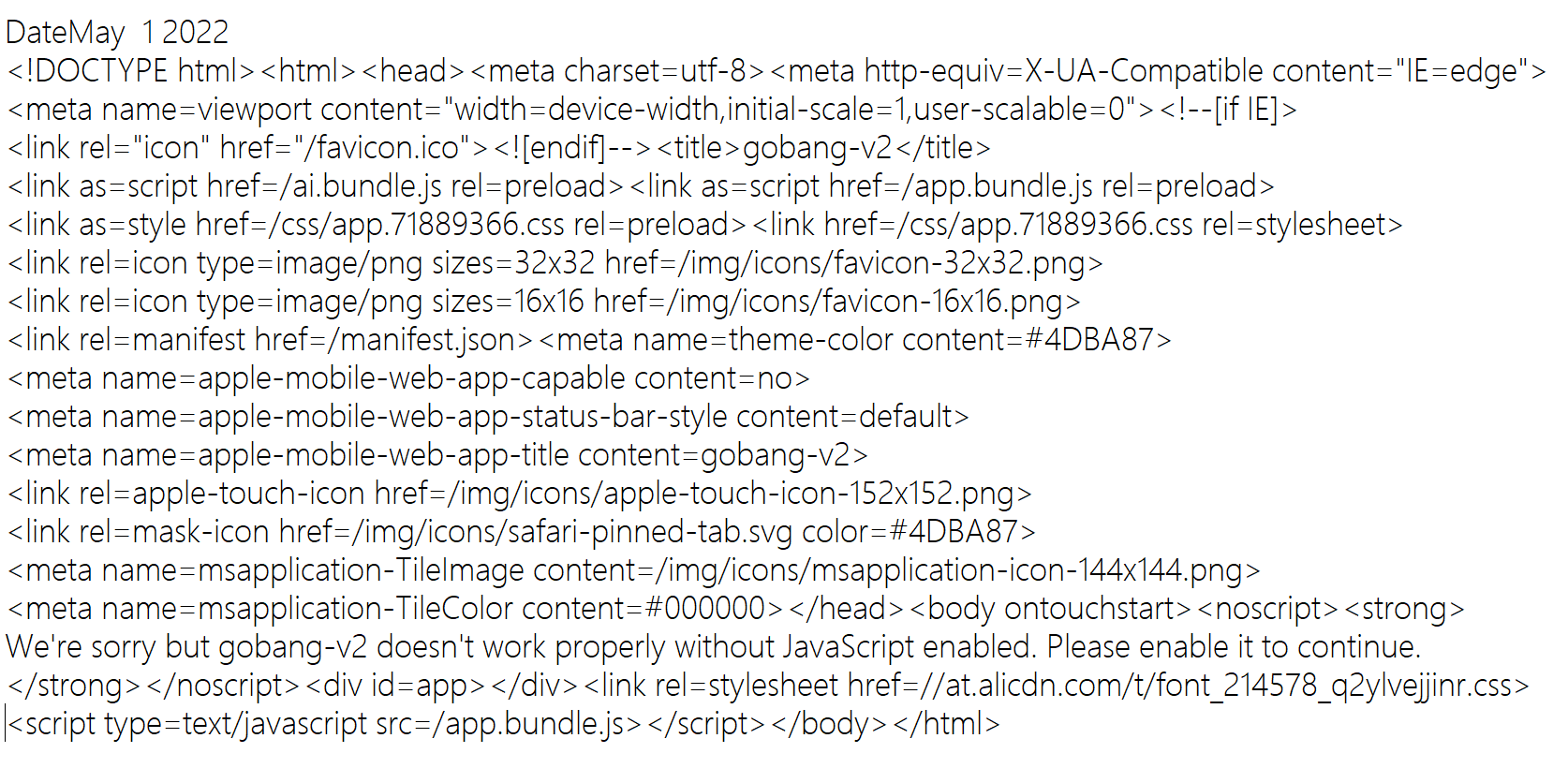
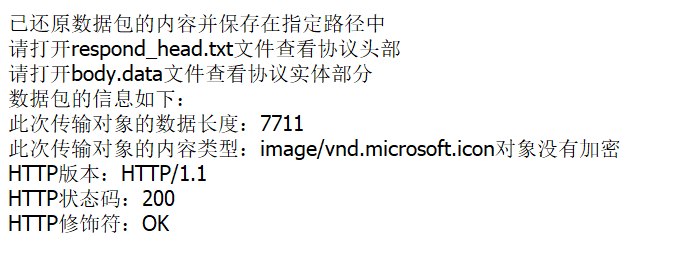
当应答协议实体部分的内容类型是text/HTML时，实体内容如图4-12（传输内容不含换行符，为了便于展示测试效果还原的原始文件进行了换行），打开测试网页的网页源代码如图4-13，可见还原的内容与网页源代码相同，说明本系统完整且正确地还原出了传输HTML对象的应答协议。

图4-12 保存在文件中的应答协议实体内容（text/HTML）



图4-13 测试网页的网页源代码

当应答协议实体部分的内容类型是image/png时，用户界面如图4-14，实体内容如图4-15（截图展示的是以文本形式打开图片类型的实体，图片对格式要求高，需要调整格式后才能用图片软件打开）。

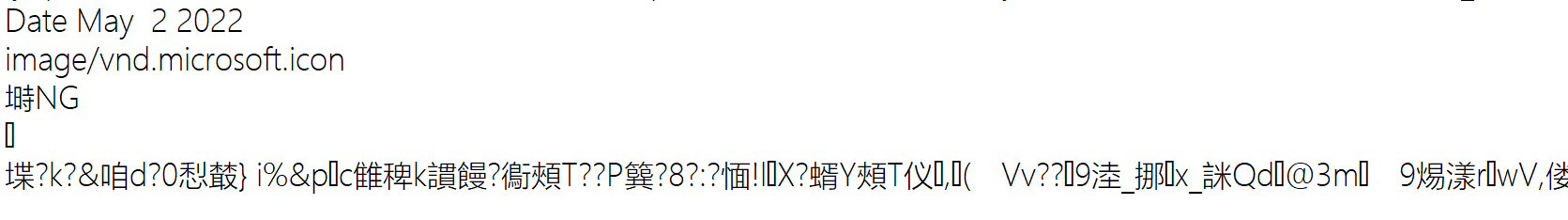
图 4-14 HTTP应答协议还原完成后的相关信息展示

图4-15 保存在文件中的应答协议实体内容（iamge）

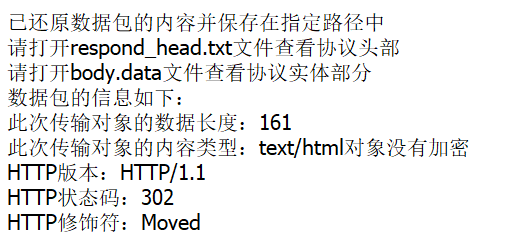
应答协议的状态不正确时，将向用户展示错误的状态码和修饰符，如图4-16。

图4-16 应答协议状态不正确时的相关信息展示

4.5错误测试

4.5.1输入错误测试

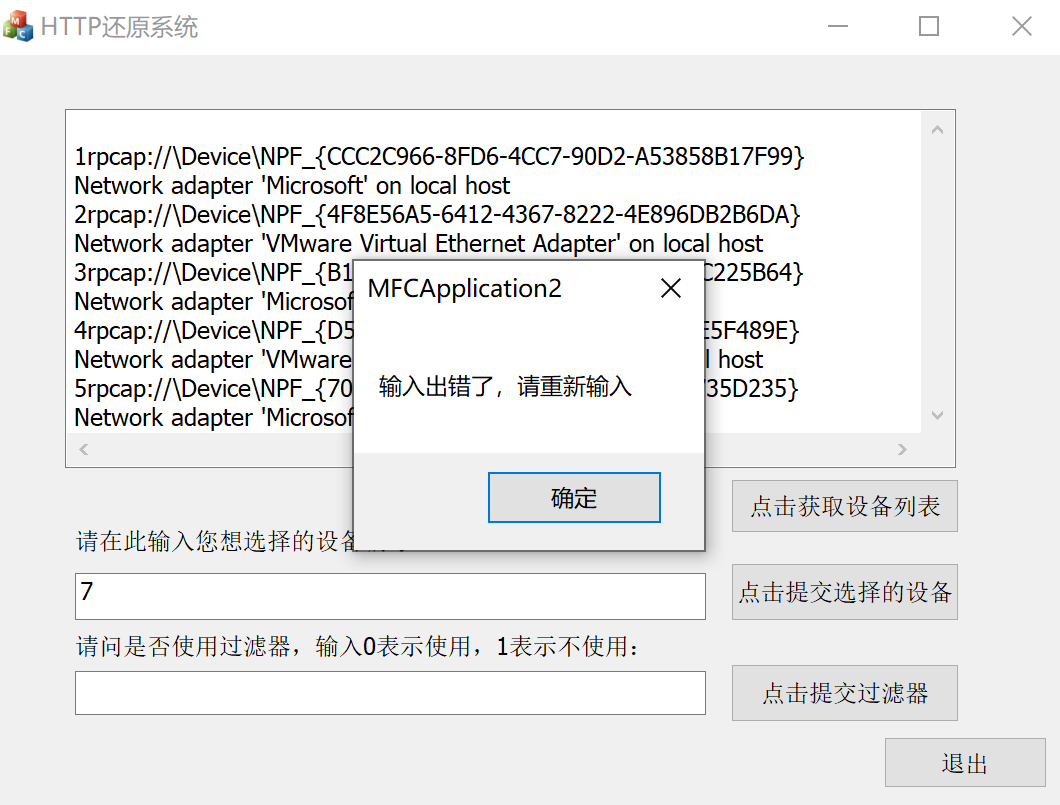
输入的数据超过允许范围时弹出的报错提示如图4-17所示。

图4-17

4.5.2网络或过滤器异常

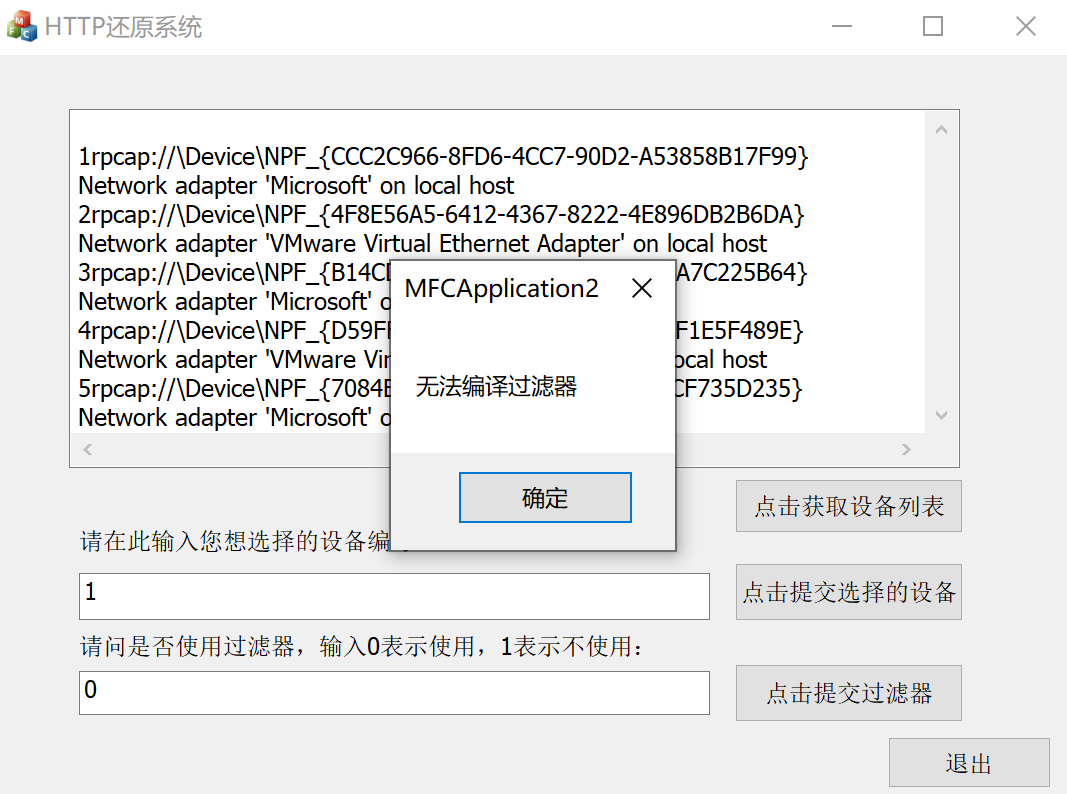
过滤条件设置错误导致过滤器无法编译时弹出的报错提示如图4-18所示。

图4-18

4.5.3文件读写异常

文件路径异常等错误导致文件无法打开时弹出的报错信息如图4-19。

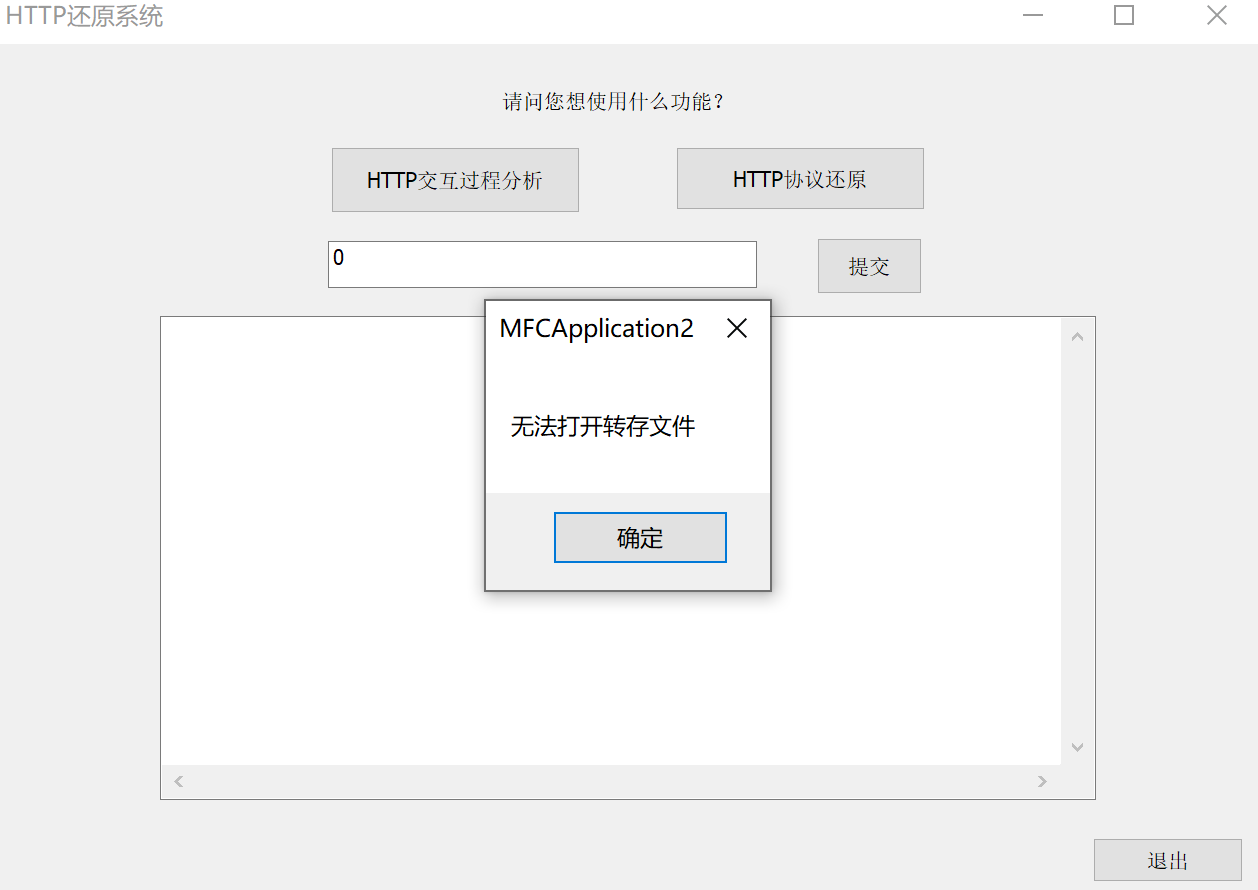


图4-19

4.6 小结

经过测试，本系统对请求和应答两种类型的HTTP协议都能较好的识别，对协议首部和内容类型为文本、程序、图片的实体部分都能较完整的还原，并且通过对比，与测试网页的内容重合度高，说明还原的准确性高。系统的图形界面简洁美观，操作方便，针对程序运行中可能遇到的各类错误设计了异常处理机制，并将报错和提示信息反馈给用户，给用户良好的使用体验。

由于本项目是在学习的过程中完成的，初期的设计还有可优化的空间，比如用文件来暂存捕获的数据包，之后只能顺序读取数据包，如果设计适当的结构体来存储数据包，按照捕获的顺序形成链表，操作会更灵活。

本次课程设计加深了我对计算机网络体系结构和各层协议的理解，让我学到了面向对象编程和制作图形用户界面的技巧，使我受益颇深。

参考文献

1. 向宇. HTTP协议还原系统的设计与实现[D].华中科技大学,2011.
2. WinPcap主页(https://www.winpcap.org/) introduction to WinPcap
3. 刘文敏. 基于数据包捕获的数据内容还原分析技术研究[D].北京邮电大学,2014.
4. 刘忠文. 基于Winpcap的网络信息监听系统研究与实现[D].华中科技大学,2007.
5. 钱松波,刘嘉勇.一种适于HTTP数据还原的QS改进算法[J].通信技术,2015,48(03):351-356.