数据包捕获与分析

实验目的

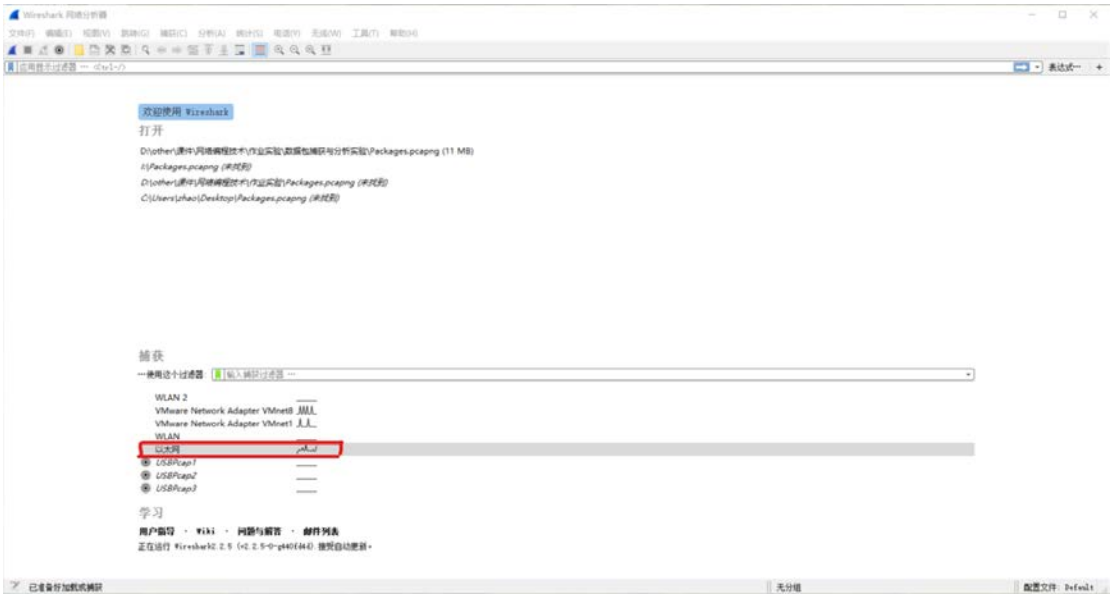
本实验学习通过 Wireshark 捕捉实时网络数据包，并根据网络协议分析流程对数据包在 TCP/IP 各层协议中进行实际抓包分析，为网络协议分析和还原提供技术手段。

实验要求

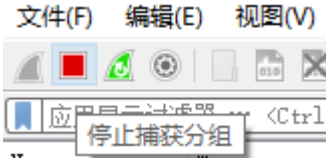
参考 Wireshark 的工作原理，用 Visual C＋＋或 Eclipse 或者 Qt 编写一个简单的数据包捕获(可选）与分析工具。

开始抓包

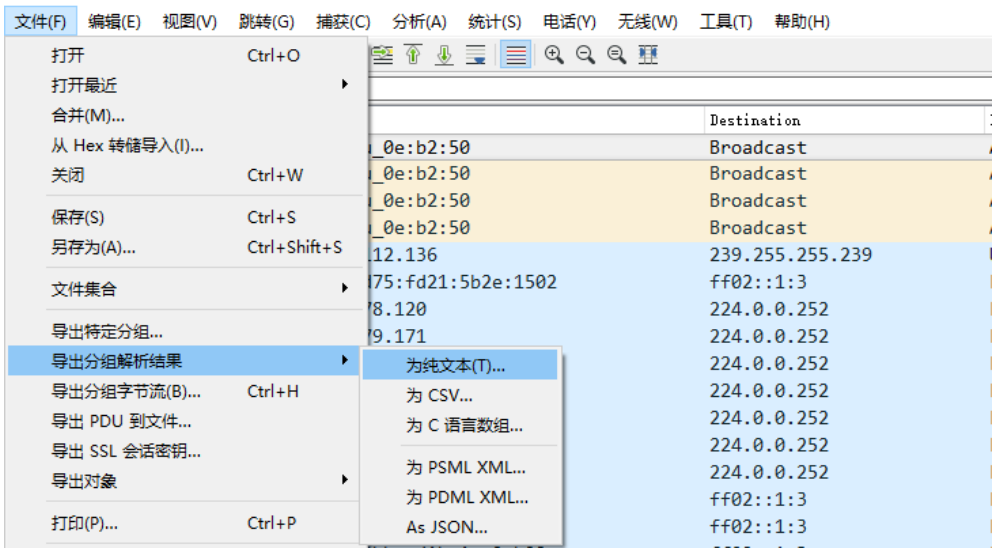
1. 启动 WireShark，选择要抓包的设备。这里选择以太网。

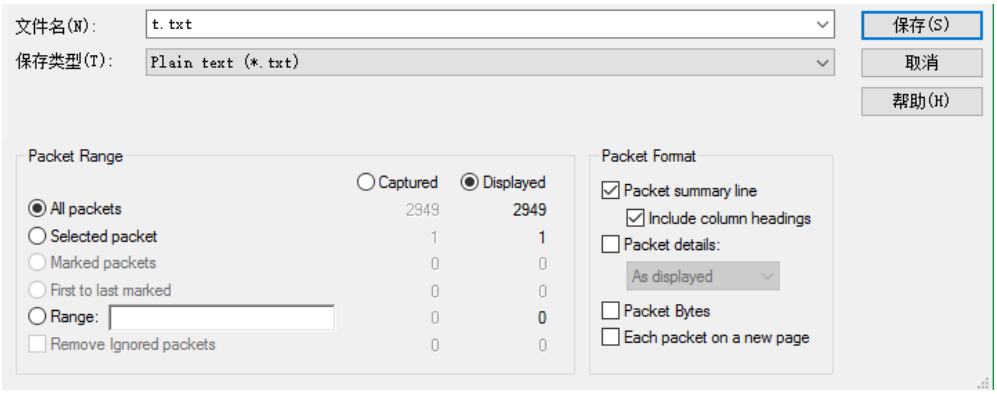


2.开始抓包，等待一段时间后结束，停止捕获分组。



3.将捕获的分组另存为文本，以便单独分析。注意另存为文本的时候勾选 Package summary line 和 Include column heading 选项。





开始编程

编程环境

操作系统：Windows10
开发环境：Qt5.8+MSVC2015
注意：需要安装 WinCap

编程思路

读取抓到的数据包，每组数据包格式一致。因而考虑一种数据结构：

```
class Package{
        int No;
        double Time;
        QString Source;
        QString Destination;
        QString Protocal;
        int Length;
        QString Info;
    }
```

表示一组数据包中的数据信息。然后用 QTableWidget 展示所有抓取到的数据分组，并且可以通过输入关键字筛选想要查看的分组信息。最终得到的程序效果如下图所示：

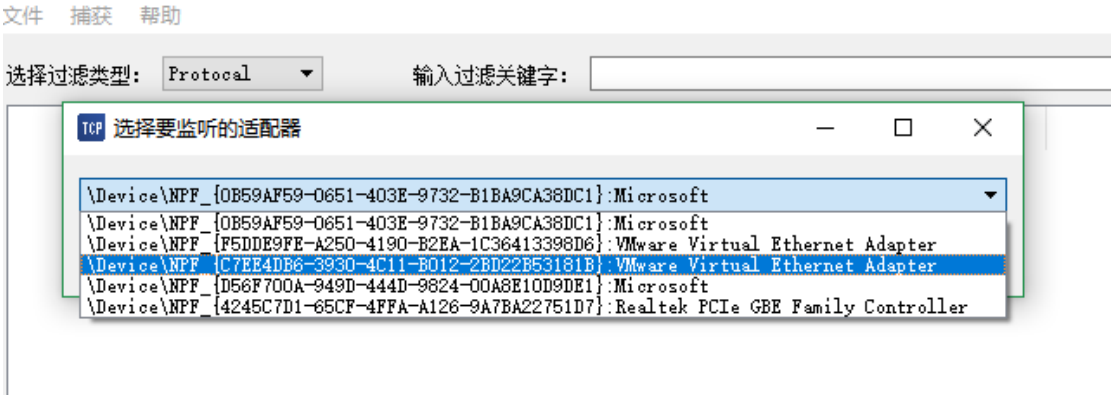
程序演示

1、抓包

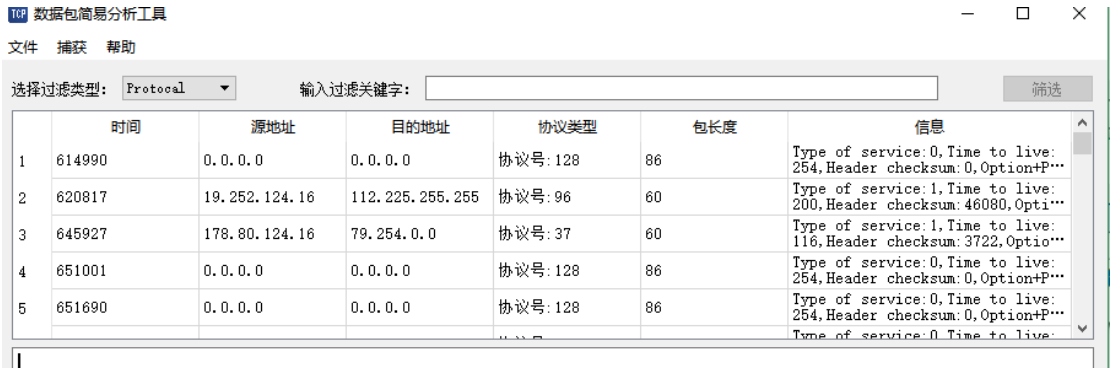
1.打开程序，数据包分析程序升级版

2.选择捕获选项

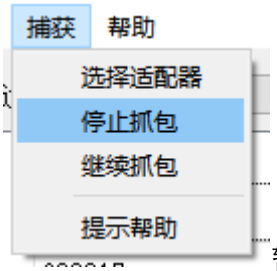
3.点击选择适配器，在弹出的窗口中选择要监听的适配器



4.点击开始监听后，开始抓包。并将抓包实时显示在表中。



5.抓包进行中可以选择



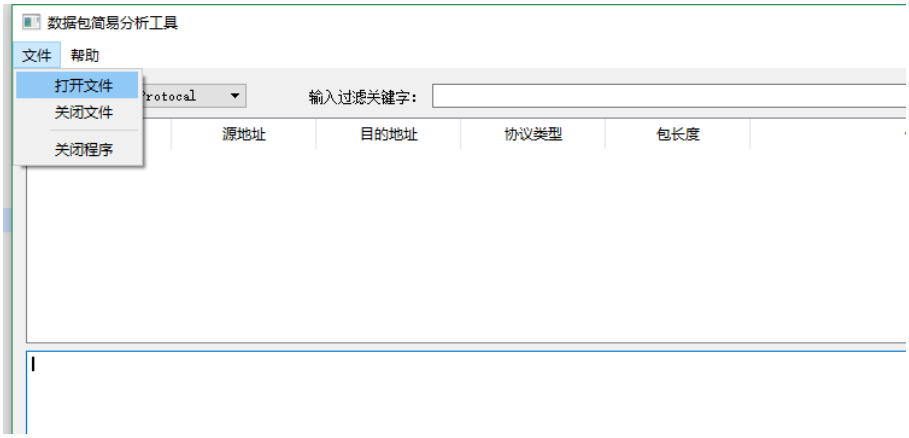
暂停抓包，暂停之后还可以选择继续抓包

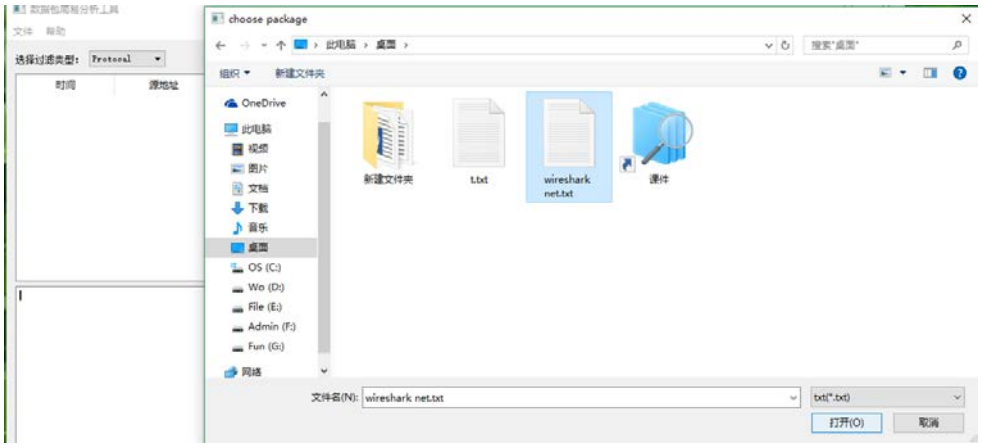
2、分析包

1. 打开程序，为了方便。分别编译了 64 位和 32 位两个版本的程序。

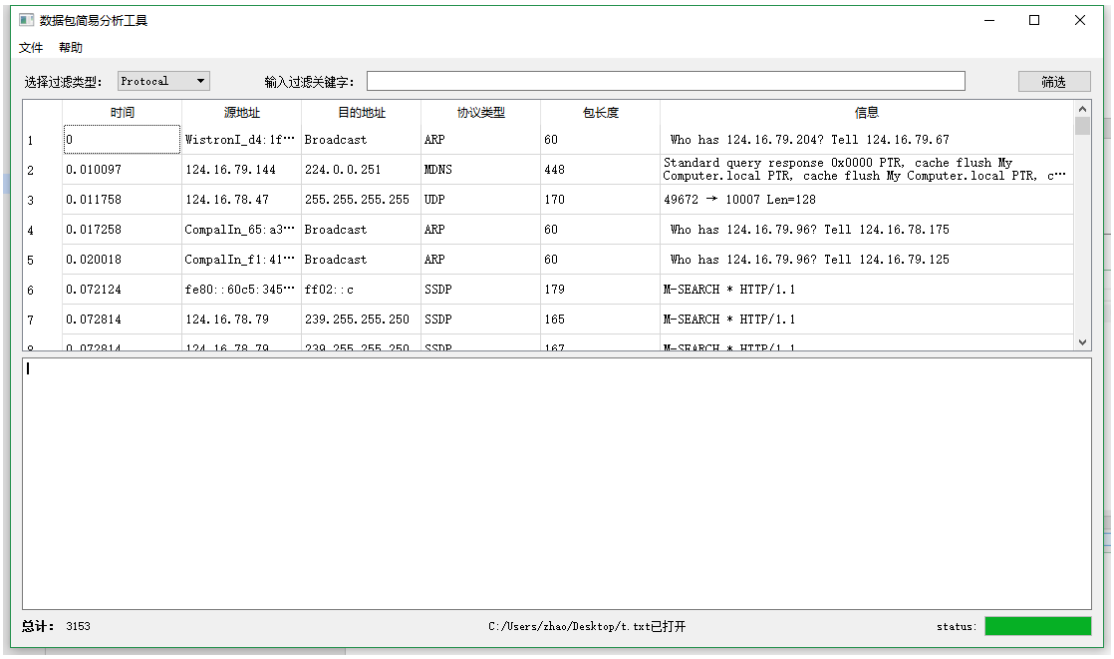
为了方便运行，后续又作了更改。编译打包成可以单独运行的 exe 文件。

2. 界面如下所示，点击打开文件。选择要分析的数据包



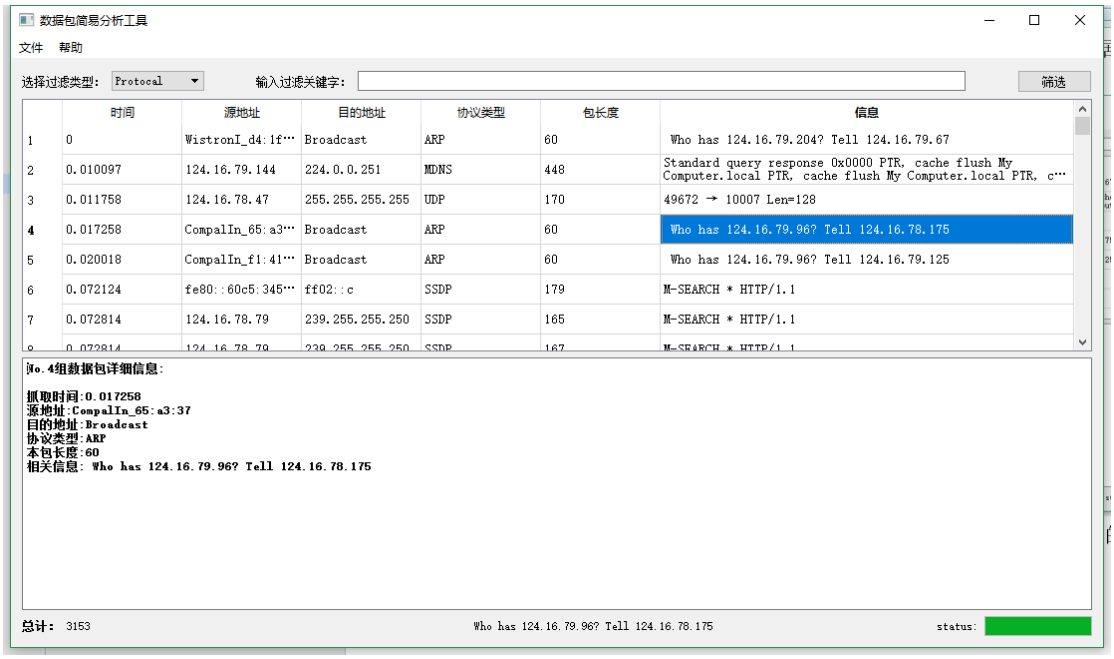


3. 打开之后，显示相应的状态信息：提示打开文件的路径，数据包总分组数等。

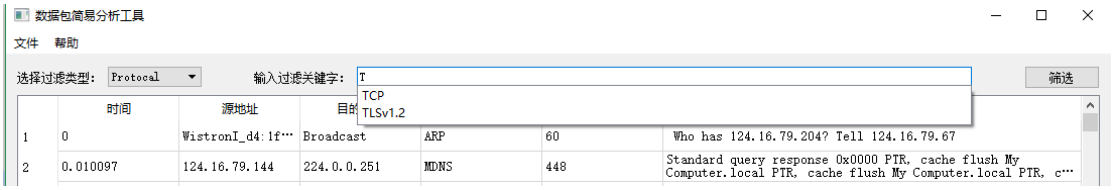


	时间	源地址	目的地址	协议类型	包长度	信息
1	0	WistronI_d4:1f…	Broadcast	ARP	60	Who has 124.16.79.204? Tell 124.16.79.67
2	0.010097	124.16.79.144	224.0.0.251	MDNS	448	Standard query response 0x0000 PTR, cache flush My Computer.local PTR, cache flush My Computer.local PTR, c…
3	0.011758	124.16.78.47	255.255.255.255	UDP	170	49672 → 10007 Len=128
4	0.017258	CompalIn_65:a3…	Broadcast	ARP	60	Who has 124.16.79.96? Tell 124.16.78.175
5	0.020018	CompalIn_f1:41…	Broadcast	ARP	60	Who has 124.16.79.96? Tell 124.16.79.125
6	0.072124	fe80::60c5:345…	ff02::c	SSDP	179	M-SEARCH * HTTP/1.1
7	0.072814	124.16.78.79	239.255.255.250	SSDP	165	M-SEARCH * HTTP/1.1

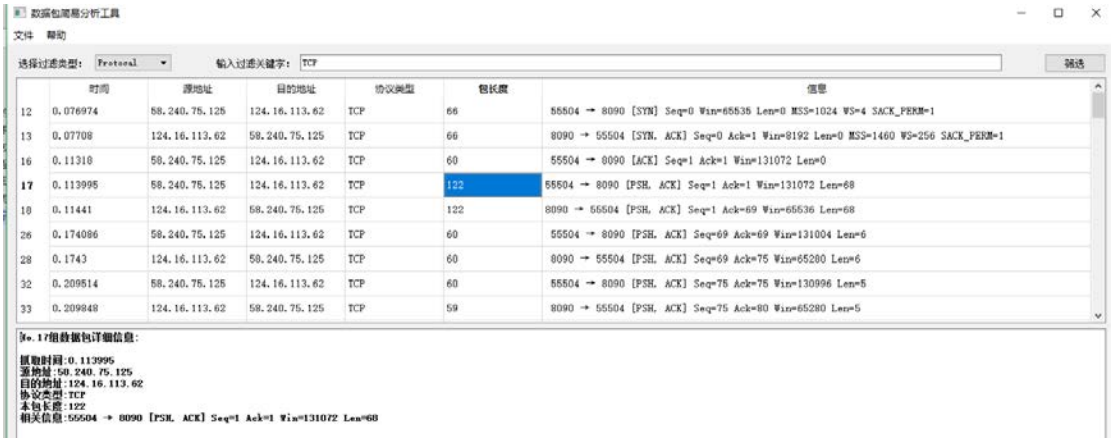
4. 点击任意单元格，即在下方的文本框里显示该单元格所在分组的具体信息。并在状态栏显示单元格的内容。



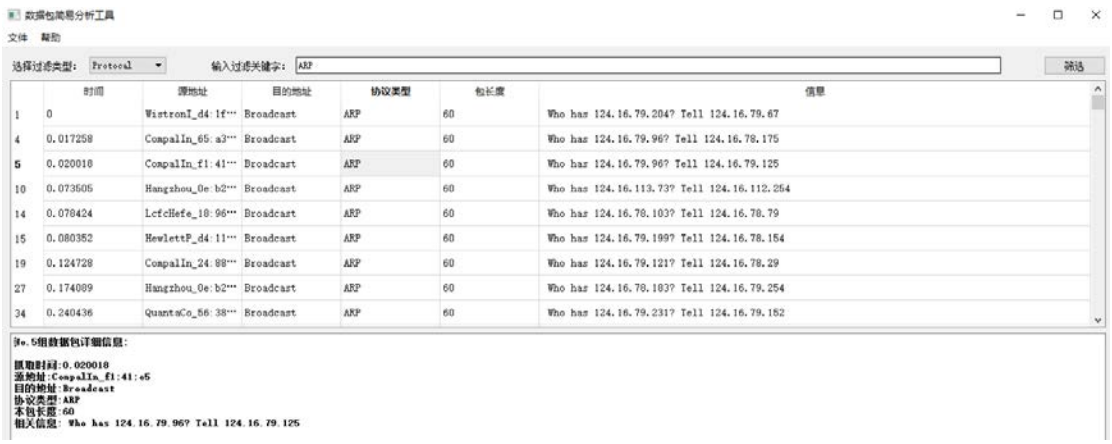
5. 筛选功能：可以按照不同的类型对数据包进行筛选。默认勾选“Protocol”类型，在后面的输入框内输入想要查看某种协议， 输入框加入了人性化的提示功能。



6. 回车(Enter)或者点击“筛选”按钮都会自动将相应的分组筛选出来。如输入“TCP”,筛选出全部 TCP 协议的分组。



7. 要想再次查看或者筛选分组，则重新输入，此时回重置表格，并且显示新的筛选分组。演示：再次筛选 ARP 分组：



8. 要想重新打开或查看新的数据文件，可以直接“打开文件”。也可以“关闭文件”，关闭当前打开的数据文件。初始化程序：

9. 关于，帮助。显示我们组成员相关信息。

10. 退出并关闭程序。

程序源码

文件：TCP_Package_Analysis.pro

```
#-------------------------------------------------
#
# Project created by QtCreator 2017-04-19T23:26:28
#
#-------------------------------------------------

QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = TCP_Pakage_Analysis
TEMPLATE = app
# The following define makes your compiler emit warnings if you use
# any feature of Qt which as been marked as deprecated (the exact
warnings
```

```
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated
APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain
version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all
the APIs deprecated before Qt 6.0.0
SOURCES += main.cpp\
        mainwindow.cpp \
    package.cpp \
HEADERS  += mainwindow.h \
    package.h \
FORMS    += mainwindow.ui
CONFIG +=C++11
```

文件：mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QFile>
#include <QTableWidgetItem>
#include <filterwidget.h>
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void filterExecution(QString keyWord);
public slots:
    void itemSelectedAndShowDetail(QTableWidgetItem *item);
private slots:
    void on_filterButton_clicked();
private:
    Ui::MainWindow *ui;
    filterWidget *filterWindow;
    QFile file;
```

```
    int TotalLineOfPackage=0;
    //QString allProtocal[9] =
{"TCP","WSP","ULP","SSL","SSDP","IPv4","HTTP","DNS","Ethertype"};
};
#endif // MAINWINDOW_H
```

文件 mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "package.h"
#include <QDebug>
#include <QFileDialog>
#include <QDialog>
#include <QMessageBox>
#include <QCompleter>
#include <QStringList>
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->openFile->setText(QString("Open File"));
    ui->closeFile->setText(QString("Close File"));
    ui->exit->setText(QString("Exit Program"));
    ui->about->setText(QString("TCP Team"));
    ui->progressBar->setValue(100);
    //设置过滤输入的提示器
    QStringList listOfProtocal;

listOfProtocal<<"TCP"<<"WSP"<<"ULP"<<"UDP"<<"TLSv1.2"<<"SSDP"<<"MDNS"
<<"LLMNR"<<"IPv6"<<"IGMPv2"<<"ICMPv6"<<
                    "HTTP"<<"DNS"<<"DHCPv6"<<"ARP";
    QCompleter *completer = new QCompleter(listOfProtocal,this);
    completer->setCaseSensitivity(Qt::CaseInsensitive);
    ui->keyWord->setCompleter(completer);
    /**
    Open
    打开文件并读取数据包
    **/
    connect(ui->openFile,&QAction::triggered,
            [=]()
    {
        qDebug()<<"Open Files.";
```

```
        QString path = QFileDialog::getOpenFileName(this,"choose
package","/home","txt(*.txt)");
        //成功打开数据包则进行后面的分析工作
        if(path.isEmpty()!=true)
        {
            ui->progressBar->setValue(0);
            file.setFileName(path);
            //文件成功打开，继续
            if(file.open(QIODevice::ReadOnly))
            {
                ui->selectedText->setText(QString(path+" opened"));
                char buf[1024];
                qint64 lineLength;
                do{

                    lineLength = file.readLine(buf, sizeof(buf));
                if (lineLength != -1)
                {
                    //qDebug()<<buf;//已经过测试
                    QString line(buf);
                    Package *package = new Package(line);
                    /**
                      TEST CODE
                    **/
                    //表中插入一行
                    ui->table->insertRow(TotalLineOfPackage);
                    ui->table->setItem(TotalLineOfPackage,0,new
QTableWidgetItem(QString("%1").arg(package->Time)));
                    ui->table->setItem(TotalLineOfPackage,1,new
QTableWidgetItem(package->Source));
                    ui->table->setItem(TotalLineOfPackage,2,new
QTableWidgetItem(package->Destination));
                    ui->table->setItem(TotalLineOfPackage,3,new
QTableWidgetItem(package->Protocal));
                    ui->table->setItem(TotalLineOfPackage,4,new
QTableWidgetItem(QString("%1").arg(package->Length)));
                    ui->table->setItem(TotalLineOfPackage,5,new
QTableWidgetItem(package->Info));
                    ++TotalLineOfPackage;
                    ui->progressBar->setValue(TotalLineOfPackage);
                }
                }while(lineLength != -1);
                ui->progressBar->setValue(100);
```

```
ui->totalPackage->setText(QString("%1").arg(TotalLineOfPackage));
                ui->filterButton->setEnabled(true);
            }
            else
            {
                ui->selectedText->setText(QString("Open file
failed!"));
            }
        }
    });
    /**
     * 效果等用于按下“筛选按钮”，回车快捷键更方便
     * */

connect(ui->keyWord,&QLineEdit::returnPressed,this,&MainWindow::on_fi
lterButton_clicked);
    /**
     * 重设表格，恢复上一次过滤的改变,下一次过滤继续使用
     * */
    connect(ui->keyWord,&QLineEdit::textEdited,
            [=]()
    {
        for(int i=0;i<TotalLineOfPackage;++i)
        {
            ui->table->setRowHidden(i,false);
        }
    });
    /**
      About
      关于程序
      **/
    connect(ui->about,&QAction::triggered,
            [=]()
    {
       //
        qDebug()<<"Show help message.";
        QMessageBox::about(this,QString::fromLocal8Bit("TCP
Team"),QString::fromLocal8Bit("ZhaoPeng  TengFei.Wang  LuGan"));
    });
    /**
      Close
      关闭当前打开的数据包文件，并初始化
    **/
```

```
    connect(ui->closeFile,&QAction::triggered,
            [=]()
    {
        qDebug()<<"Close current files and reset window.";
        file.close();
        ui->keyWord->setText("");
        ui->detail->clear();
        ui->totalPackage->setText(" ");
        ui->selectedText->setText(" ");
        ui->filterButton->setEnabled(false);
        ui->table->reset();
        ui->progressBar->setValue(0);
        for(int i=TotalLineOfPackage;i!=-1;--i)
        {
           ui->table->removeRow(i);
        }
        //包计数器重置
        TotalLineOfPackage=0;
    });
    /**
      Exit
      关闭并退出程序
    **/
    connect(ui->exit,&QAction::triggered,
            [=]()
    {
        this->close();
    });
    /**
      单元格选中触发事件
      **/

connect(ui->table,&QTableWidget::itemClicked,this,&MainWindow::itemSe
lectedAndShowDetail);

}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::itemSelectedAndShowDetail(QTableWidgetItem *item)
{
```

```
    QString text = item->text();
    //qDebug()<<text;
    ui->selectedText->setText(text);
    int row = item->row();
    ui->detail->setText(QString("No.%1Package
Detail:").arg(row+1).toUtf8());
    ui->detail->append(" ");

ui->detail->append(QString("Time:"+ui->table->item(row,0)->text()));

ui->detail->append(QString("Source:"+ui->table->item(row,1)->text()))
;

ui->detail->append(QString("Destination:"+ui->table->item(row,2)->tex
t()));

ui->detail->append(QString("Protocal:"+ui->table->item(row,3)->text()
));

ui->detail->append(QString("Length:"+ui->table->item(row,4)->text()))
;

ui->detail->append(QString("Info:"+ui->table->item(row,5)->text()));
}

void MainWindow::on_filterButton_clicked()
{

    if(ui->filterBox->currentIndex()==0)
    {
        //选中按“协议”筛选
        QString keyWord = ui->keyWord->text();
        if(keyWord=="")
        {

QMessageBox::about(this,QString("Warming"),QString("Please Input
Keyword"));
        }
        else
        {

//"TCP","WSP","ULP","SSL","SSDP","IPv4","HTTP","DNS","Ethertype"
            if(keyWord=="TCP")
            {
```

```
    filterExecution(keyWord);
}
else if(keyWord=="WSP")
{
    filterExecution(keyWord);
}else if(keyWord=="ULP")
{
    filterExecution(keyWord);
}else if(keyWord=="SSL")
{
    filterExecution(keyWord);
}else if(keyWord=="SSDP")
{
    filterExecution(keyWord);
}else if(keyWord=="IPv4")
{
    filterExecution(keyWord);
}else if(keyWord=="HTTP")
{
    filterExecution(keyWord);
}else if(keyWord=="DNS")
{
    filterExecution(keyWord);
}else if(keyWord=="Ethertype")
{
    filterExecution(keyWord);
}else if(keyWord=="ARP")
{
    filterExecution(keyWord);
}else if(keyWord=="UDP")
{
    filterExecution(keyWord);
}else if(keyWord=="MDNS")
{
    filterExecution(keyWord);
}else if(keyWord=="TLSv1.2")
{
    filterExecution(keyWord);
}else if(keyWord=="LLMNR")
{
    filterExecution(keyWord);
}else if(keyWord=="IGMPv2")
{
    filterExecution(keyWord);
```

```
            }else if(keyWord=="ICMPv6")
            {
                filterExecution(keyWord);
            }else if(keyWord=="DHCPv6")
            {
                filterExecution(keyWord);
            }else
            {

QMessageBox::about(this,QString::fromLocal8Bit("Nothing
Found"),QString::fromLocal8Bit("No Such Package"));
            }

        }
    }
    else
    {
        QMessageBox::question(this,QString("Sorry"),QString("Only
Support 'Protocal'"),QMessageBox::Close,QMessageBox::Yes);
    }
}

void MainWindow::filterExecution(QString keyWord)
{
    //qDebug()<<"过滤 Ethertype";
    for(int i=0;i<TotalLineOfPackage;i++)
    {
        QString temp = ui->table->item(i,3)->text();
        qDebug()<<temp;
        if(temp!=keyWord)
        {
            ui->table->setRowHidden(i,true);
        }

    }
}
```