LTE-M Monitor2000

服务器部分详细设计说明书

[1引言 2](#_Toc485810250)

[1.1编写目的 2](#_Toc485810251)

[1.2背景 2](#_Toc485810252)

[2程序系统的结构 2](#_Toc485810253)

[3 原始套接字（Raw Socket） 5](#_Toc485810254)

[3.1程序描述 5](#_Toc485810255)

[3.2功能 5](#_Toc485810256)

[3.3流程逻辑 5](#_Toc485810257)

[3.4具体实现函数说明 6](#_Toc485810258)

[4 数据包处理 8](#_Toc485810259)

[4.1程序描述 8](#_Toc485810260)

[4.2功能 8](#_Toc485810261)

[4.3流程逻辑 9](#_Toc485810262)

[4.4具体实现函数说明 9](#_Toc485810263)

[5 用户筛选条件的配置与读取 12](#_Toc485810264)

[5.1 程序描述 12](#_Toc485810265)

[5.2功能 13](#_Toc485810266)

[5.3流程逻辑 13](#_Toc485810267)

[5.4具体实现函数说明 14](#_Toc485810268)

[6 Pcap文件保存 16](#_Toc485810269)

[6.1 程序描述 16](#_Toc485810270)

[6.2 Pcap文件结构 16](#_Toc485810271)

[6.3 Pcap文件保存程序设计 17](#_Toc485810272)

[7 离线查找CBTC历史数据包 19](#_Toc485810273)

[7.1程序描述 19](#_Toc485810274)

[7.2功能 19](#_Toc485810275)

[7.3流程逻辑 19](#_Toc485810276)

[7.4具体实现函数说明 21](#_Toc485810277)

**详细设计说明书**

# 1引言

## 1.1编写目的

本设计概要的目的如下：

1. 解释该软件的用途及使用方法；
2. 解析该软件的编写流程；
3. 记录该软件的修改内容、时间、修改人等信息

## 1.2背景

说明：

1. 软件名称：LTE-M Monitor2000服务器部分
2. 任务提出者：地铁运营商
3. 开发者：北京交通大学

# 2程序系统的结构

程序的主体流程如下：



图2-1 整体设计流程图

其中，在线监测部分的结构图如下：



图2-2 在线监测流程图

文件传输模块的结构图如下：



图2-3 文件传输模块流程图

# 3 控制命令通信（TCP）

## 3.1程序描述

软件中客户端与服务器端命令交互采用的是TCP协议，通过Socket封装的TCP实现数据包的发送与接收。

在程序入口MainForm类中，开启线程mainThread用来不断的监听客户端发来的控制命令（开线程防止堵塞主线程），然后，将服务器端监听命令的Socket置为侦听状态。当有客户端连接时，记录为该客户端分配的Socket和Thread。

之后，通过CommunitationToClient类中的communicate方法开始循环读取与客户端之间的数据流，通过ServerToClientSend类中的sendServerCmd方法发送服务器的回复信息。

## 3.2功能

实现对连入的客户端Accept操作，并为对应的客户端分配socket和thread，循环监听客户端命令。

## 3.3流程逻辑

TCP命令端口操作的流程如下：



图3-1 控制命令流程图

## 3.4具体实现函数说明

**3.4.1 CommunitationToClient类**

成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| eRecData | event | 绑定处理数据包的方法 |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| listenClient(int port) | 为TCP控制命令分配的端口号 | 无 | 为控制命令的Socket绑定EndPoint，将该Socket置为侦听模式，调用listenPort方法处理连入的客户端。 |
| listenPort(Socket TCPCommandWatchSocket) | 为TCP控制命令分配的Socket | 无 | 对连入的客户端执行Accept操作，并记录返回的socket，为该客户端开放线程。由于服务器端只需要与唯一一个客户端对应，所以当已经存在对应的Socket和Thread时，要做释放操作。 |
| communicate(object Socket) | 为TCP控制命令分配的Socket | 无 | 循环接收Socket传来的数据流，由于TCP通信是按照流传送数据，所以每次读取时先读取四个字节的数据头，从中解出数据总长度，再按照该长度从Socket上读取数据流。将完整的数据流放到total数组中，通过eRecData事件调用处理方法。 |

**3.4.2 ServerToClientSend类**

为用户提供选择绑定网卡IP的窗体。

成员变量：无

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| sendSeverCmd(SocketTypes conName, byte[] cmd, int lenth) | Socket、要发送的数组、发送的长度 | 无 | 向客户端发送相应数据。 |

# 4 命令处理

## 4.1程序描述

将数据流的头部（4个字节）解出来之后，确定该数据包的命令类型。本软件根据需求共定义六种命令，通过枚举CToSCommandType进行描述，分别为：

(1) 客户端发来的连接请求，加载用户筛选信息命令ConnectAndFilterInfo

(2) 客户端请求开始监测命令RealTimeDataReq

(3) 客户端请求暂停或停止命令RealTimeDataPause

(4) 客户端请求文件目录命令GetFileDirectory

(5) 客户端请求下载文件（下载指定文件）命令DownSeletedFile

(6) 客户端请求文件下载（根据指定时间段下载）命令DownTimeFile

根据命令的不同所采取的操作在以下方法介绍中介绍。

## 4.2功能

对客户端的操作进行相应

## 4.3流程逻辑

处理命令的流程图如下：



图4-1 处理命令流程图

## 4.4具体实现函数说明

**4.4.1 ServerManagement类**

成员变量：无

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| recClientCmd(byte[] receiveData, int lenth) | 完整的数据流、数据流长度 | 无 | 与事件eRecData绑定，实现对命令包的处理。首先调用GetHeader方法解出数据头，确定命令类型，根据命令类型执行不同的方法。 |
| ConnectAndFilter(Unpack unpack) | 数据流 | 无 | 处理客户端的ConnectAndFilterInfo命令，根据数据流的内容，调用GetFilterInfo()方法解出用户配置的筛选信息。然后调用establishRealTimeConnection()方法建立与客户端实时转发数据包的Socket，调用establishCatchSocket()方法建立捕捉数据包的Socket。最后将两个Socket建立成功与否的消息打包在回应数据包中。将数据包回复给客户端。 |
| StartMonitor() | 无 | 无 | 该方法是对开始监测命令的相应，首先判断当前时候有正在抓包的线程，若果有，释放该线程，然后开辟新线程通过CatchData类实现数据包的捕获，然后将该线程记录在字典中。 |
| PauseOrStop() | 无 | 无 | 该方法是对暂停和停止监测命令的相应，释放保存和转发的Timer线程，将捕获数据包的线程释放，从字典中移除。 |
| FilDirectory() | 无 | 无 | 该方法是对目录请求命令的相应。将安装目录下的“pcap”文件夹的目录发送到客户端。这里FileDirectory类是目录的对象类，最后将该类的实体对象序列化，转换成字节后发送。解析目录通过递归调用analysisDirectory()方法实现。  注：由于序列化与反序列化要对同一类型的对象进行操作，故将FileDirectory类和序列化与反序列化方法制作成Serialize.dll程序集，这里需要添加对其的引用。 |
| DownLoadFileByChoosed(Unpack unpack) | 数据流 | 无 | 该方法是对下载选中文件命令的相应，将用户选中的文件路径反序列化之后，通过UploadNew类进行上传。 |
| DownLoadFileByTime(Unpack unpack) | 数据流 | 无 | 该方法是对下载指定时间段文件命令的相应，将用户设置的时间段反序列化之后，调用FindFileList()方法找到符合条件的文件list，通过UploadNew类进行上传。 |

# 5 数据包捕获

## 5.1 程序描述

首先将Socket获取的数据包的网络层和数据链路层解出来，获得数据包的源IP和端口与目的IP和端口，通过对比用户设置的筛选条件，确定是否需要进一步处理。

当数据包符合用户设置的筛选条件时，为该数据包打入时间戳，缓存在内存中。通过Timer定时将缓存输出到Pcap文件中，并转发给客户端。

## 5.2功能

捕获CBTC数据包。

## 5.3流程逻辑

抓包流程如图：



图5-1 抓包流程

## 5.4具体实现函数说明

**5.4.1 CatchData类**

该类是开始监测后的核心类，包括如下功能模块：

(1)利用RawSocket抓包

(2)在合适的时间创建Pcap文件

(3)将缓存输出到Pcap文件，并转发给客户端

成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| KeepCatching | Bool | 是否开始监测的标志 |
| receiveBufferLength | Int | 存放数据包的数组的长度 |
| receiveBufferBytes | Byte[] | 存放数据包的数组 |
| genePcapFile | GeneratePcapFile | 创建Pcap文件的类 |
| \_fileSavePath | String | 当前正在写入的Pcap文件的路径 |
| \_fileName | String | 当前正在写入的Pcap文件的文件名 |
| CreatPcapPerDay | System.Threading.Timer | 每天凌晨12点后建立新的Pcap文件的Timer |
| OutputDataTimer | System.Threading.Timer | 周期性的输出缓存的Timer |
| TemporaryStorage | ConcurrentQueue<PacketProperties> | 存放数据包的缓存集合 |
| RealTimeClientEndPoint | IPEndPoint | 客户端的EndPoint |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| CatchData() | 无 | 无（构造函数） | 该类的构造函数，如果是第一次开始监测时，调用CreatPcapFile()函数创建Pcap文件，初始化存放数据包的数组，打开输出数据包和每天创建Pcap文件的Timer。 |
| StartCatch(object o) | 客户端的EndPoint | 无 | 与抓包线程绑定的方法，执行开始抓包的操作。如果是打开软件后第一次开始监测时，首先初始化缓存集合，然后将开始监测的标志KeepCatching置为true，最后调用BeginReceive方法开始抓包。 |
| BeginReceive() | 无 | 无 | 当监测的标志KeepCatching为True时，采用异步的方式从网卡抓包，当有数据包到达时，将数据包放在receiveBufferBytes中，并调用回调函数CallReceive |
| CallReceive(IAsyncResult | 异步操作状态 | 无 | 停止异步抓包，获取数据包长度receivedLength，将数据包信息传递给HandlePacket类对数据包进行处理，处理之后清空数组，继续异步抓包。 |
| CreatPcapFile() | 无 | 无 | 调用GeneratePcapFile类创建Pcap文件，文件的名字以创建时刻的时间命名，路径按照年、月、日以此创建文件夹。 |
| CreatPcapFilePerDay(object o) | 无 | 无 | 与每天创建Pcap文件的Timer绑定，创建完之后，继续打开Timer开始计时，计时周期为第二天0点与当前时间的时间差。 |
| Output(ConcurrentQueue<PacketProperties> PropertiesList) | 存放数据包的缓存集合 | 无 | 将缓存集合中的数据包输出到Pcap文件中，如果当前写入的Pcap文件大小超过用户设置，则创建新的Pcap文件。通过集合的TryDequeue方法，将每一个数据包取出后输出，并转发到客户端。 |

**5.4.2 HandlePacket类**

该类的工作流程：

1. 定义了一个IP头结构体IPHeader，这样每当有一个数据包到达，可以用强制类型转化把包中的数据流转化为一个个IPHeader对象；
2. 每当有一个数据包到达，实例化一个PacketProperties类（后边将介绍）；
3. 将对网络层和数据链路层的信息解析后的结果传递到PacketProperties类中的属性，方便后边使用；
4. 根据地址和端口信息判断是否符合筛选条件；
5. 若符合条件，将数据包保存到缓存中；
6. 周期性的对缓存进行输出，输出到Pcap文件，并转发到客户端。

成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| PacketProperties | PacketProperties | 用来存储数据包信息的对象（类） |
| \_packetDateTimeTicksBuff | byte[] | 存放数据包时间戳的数组 |
| \_realTimePacketBuff | Byte[] | 存放转发的数据包的数组 |
| RealTimeEP | IPEndPoint | 客户端的EndPoint |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| HandlePacket(IPEndPoint RealTimeEP, byte[] buf,int len,byte[] packetDateTimeTicksBuff,byte[] realTimePacketBuff) | 客户端的EndPoint，抓取的数据包，数据包长度 | 无（构造函数） | 该类的构造函数，为数据包实例化PacketProperties对象，调用Unpack解包函数 |
| Unpack(byte[] buf, int len) | buf、len | 无 | 利用指针、IPHeader结构体、Lambda表达式解数据包的先关属性，复制给PacketProperties中的属性成员，判断当protocol == 17（UDP协议）时，调用过滤方法DataFilterByIPAndPort，当protocol == 17（IPV6）时，截去IPV6层的40个字节，再调用DataFilterByIPAndPort  注：①该方法中使用了指针（不安全代码），所以方法前要加unsafe关键字；  ②要在项目属性中勾选“允许不安全代码”（项目—属性—生成—勾选（允许不安全代码）） |
| DataFilterByIPAndPort(  PacketProperties Properties) | Properties（数据包属性） | 无 | 根据数据包的源和目的端口及IP，判断是否符合用户的筛选条件，若符合，保存到缓存TemporaryStorage中。 |
| SendRealTimePacket(byte[] packetBuff, int packetLen, DateTime packetDateTime) | 数据包、数据包长度、数据包的捕获时间 | 无 | 转发给客户端的函数，在数据包前加入8位的时间戳，组合后转发。 |

**5.4.2 FilterForm类**

该类为筛选的界面窗体，该窗体为用户提供筛选的交互。

成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| PcapLengthNum | double | Pcap文件最大保存大小 |
| MaxVOBCToZC、MaxVOBCToCI、MaxVOBCToATS、MaxATSToVOBC、MaxCIToVOBC、MaxZCToVOBC | double | 不同通信实体的数据包的时间间隔的告警门限 |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| FilterForm() | 无 | 无（构造函数） | 初始化，调用LoadIpConfiguration和LoadHistoryConfig方法 |
| LoadIpConfiguration() | 无 | 无 | 将IPList中的所有通信实体的名字添加到界面的两个ListBox中 |
| LoadHistoryConfig()、  LoadHistory(List<string> backUps) | 无 | 无 | 读取BackUps.txt中的历史记录，自动填充界面有关告警门限以及Pcap保存大小的相关控件 |
| SaveToTxt() | 无 | 无 | 将筛选信息保存到BackUps.txt中 |
| GetAlarmInterval() | 无 | 无 | 为告警门限的六个变量赋值 |
| SaveSetting() | 无 | 无 | 将用户选择的源地址和目的地址保存下来（对ConfigProperties中的每一个ConfigurationProperties对象的IsSourceChoose和IsDestChoose赋值） |
| SelectAll(CheckedListBox ListBox, CheckState state) | 两个ListBox和应该的状态 | 无 | 对两个ListBox中的所有项全选或取消全选 |

# 6 Pcap文件保存

## 6.1 程序描述

该模块主要实现Pcap文件的创建、写入与保存。

## 6.2 Pcap文件结构

Pcap文件的基本格式如下图：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Pcap文件头 | 数据包头 | 数据报 | 数据包头 | 数据报 | 数据包头 | 数据报 | ··· |

以下将详细介绍Pcap文件头和数据包头。

**6.2.1 Pcap文件头**

Pcap文件头共有24个字节，其内容如下：

|  |  |  |
| --- | --- | --- |
| 字节 | 内容 | 作用 |
| 0-3（4字节） | 0x1A2B3C4D | 标示文件的开始 |
| 4-5（2字节） | 0x02 | 当前文件主要的版本号 |
| 6-7（2字节） | 0x04 | 当前文件次要的版本号 |
| 8-11（4字节） |  | 当地的标准时间 |
| 12-15（4字节） |  | 时间戳的精度 |
| 16-19（4字节） | 0x0000ffff | 最大存储长度 |
| 20-23（4字节） | 0x00000001 | 链路类型（1表示Ethernet以太网链路） |

**6.2.2 数据包头**

数据包头共有16个字节，其内容如下：

|  |  |  |
| --- | --- | --- |
| 字节 | 名字 | 意义 |
| 0-3（4字节） | Timestamp | 数据包时间戳高位，精确到seconds |
| 4-7（4字节） | Timestamp | 时间戳低位，精确到microseconds |
| 8-11（4字节） | Caplen | 当前数据区的长度，即抓到的数据帧长度，由此可以得到下一个数据帧的位置 |
| 12-15（4字节） | Len | 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 |

**6.2.3 数据报**

即Pcaket（链路层的数据帧去掉前面用于同步和标识帧开始的8字节和最后用于CRC校验的4字节）具体内容，长度就是Caplen，这个长度的后边，就是当前Pcap文件中存放的下一个Pcaket数据报，也就是说，Pcap文件里没有规定捕获的Packet数据报之间有什么间隔字符串，我们需要靠一个Packet包确定下一组数据在文件中的位置，向后以此类推。

## 6.3 Pcap文件保存程序设计

**6.3.1 流程逻辑**



图6-1 Pcap文件保存流程

**6.3.2 具体实现函数说明**

**6.3.2.1 PacapPacketHeader结构体**

成员变量:

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| Ts | Timestamp | 8字节 抓包时间 4字节表示秒数，4字节表示微秒数 |
| CapLen | uint | 当前数据区的长度，即抓取到的数据帧长度 |
| Len | uint | 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 |

**6.3.2.2 PcapFileHeader结构体**

成员变量:

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| Magic | uint | pcap文件标识 目前为0xa1b2c3d4 |
| MajorVersion | ushort | 2字节主版本号 |
| MinorvVersion | ushort | 2字节次版本号 |
| ThisZone | uint | 4字节时区修正 |
| SigFigs | uint | 4字节精确时间戳 |
| SnapLen | uint | 4字节抓包最大长度 |
| LinkType | uint | 4字节链路类型，一般都是1：ethernet |

**6.3.2.3 Timestamp结构体**

成员变量:

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| Timestamp\_S | uint | 时间戳高位，精确到seconds |
| Timestamp\_MS | uint | 时间戳低位，精确到microseconds |

**6.3.2.4 GeneratePcapFile类**

成员变量:

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| TCPDUMP\_MAGIC | const uint | 常量，Pcap文件标识，目前为0xa1b2c3d4 |
| PCAP\_VERSION\_MAJOR | const ushort | 常量，2字节主版本号，目前为2 |
| PCAP\_VERSION\_MINOR | const ushort | 常量，2字节次版本号，目前为4 |
| fs | FileStream | 文件流，用该流写入Pcap文件 |
| \_pcapFileSavePath | string | Pcap文件保存路径，作为PcapFileSavePath属性的返回值 |
| \_pcapFileName | string | Pcap文件名，作为PcapFileName属性的返回值 |
| m\_pcapPktHead | byte[] | 数据包头16位 |
| fileHeadBuff | byte[] | 文件头24位 |
| ethernetHeader | byte[] | 以太网头部8位 |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| GeneratePcapFile() | 无 | 无（构造函数） | 调用生成Pcap文件头部函数 |
| GetFileHeadBuff() | 无 | 无 | 生成Pcap文件头部，以此将Pcap文件头部的各个属性赋值给结构体PcapFileHeader的各个成员变量，然后使用指针将结构体中的数据填充到数组fileHeadBuff中 |
| CreatPcap(string fileSavePath, string FileName) | string fileSavePath, string FileName | 无 | 创建Pcap文件函数。根据入口参数的地址和文件名判断是否存在该Pcap文件，若不存在，创建文件。然后将pcap文件头通过文件流fs写入Pcap文件。 |
| WritePacketData(DateTime dt, byte[] packetData, int packetLenth) | 数据包捕获的时间dt，捕获到数据包，捕获到数据包长度 | 无 | 将捕获的数据包保存到Pcap文件中。先根据入口参数中的DateTime（数据包捕获时间）生成时间戳，赋值给Timestamp结构体，然后将Timestamp结构体赋值给PacapPacketHeader结构体的对应变量。之后，根据入口参数中的数据包长度设置PacapPacketHeader结构体中的CapLen和Len，注意，这里需要加14位的链路层长度。然后，然后使用指针将结构体中的数据填充到数组pcapPktHead中，最后将pcapPktHead、ethernetHeader和入口参数中的数据包数组写入文件流fs并输出到Pcap文件。 |
| DateTimeToTimestamp(DateTime dt) | 需要转换为时间戳的DateTime类型的时间 | 转换后的Timestamp | 将DateTime类型的时间转换成时间戳Timestamp结构体 |

# 7 文件上传

## 7.1 程序描述

服务器程序通过UploadNew类实现文件的上传，将需要下载的文件传入UploadNew类，然后服务器使用TCPClient连接客户端的8500端口，以便通知客户端将要传送的文件以及使用的端口。

随后，服务器端随机开放一个端口，绑定TCPListener，并置于侦听模式。接下来，服务器通过TCPClient通知客户端将要发送的文件名称及长度以及使用的端口，以便客户端连入TCPListener，通知的载体为FileProtocol对象类。

最后，Accept连入TCPListener的客户端，并开始向该通道写入文件流。发送完毕后，释放该链接。

## 7.2 功能

从服务器端向客户端上传Pcap文件。

## 7.3 流程逻辑

文件上传的流程如下：



图7-1 文件上传流程图

## 7.4 具体实现函数说明

成员变量：

|  |  |  |
| --- | --- | --- |
| 名称 | 数据类型 | 说明 |
| FileLists | List<string> | 需要传送的文件路径的集合 |
| client | TcpClient | 用于通知客户端发送文件信息的TcpClient |
| Sendlistener | TcpListener | 用于监听客户端的连接，连接后传送文件流的TcpListener |
| streamToClient | NetworkStream | 绑定TcpClient的网络流 |
| fileBuffer | byte[] | 文件流的缓存 |

具体方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 输入参数 | 函数返回值 | 说明 |
| UploadNew(List<string> fileLists) | 要传送的文件路径的集合 | 无（构造函数） | 实例化TcpClient，连接客户端的8500端口 |
| StartSend() | 无 | 无 | 随机开放TCP端口，用于文件流的传输，对于每一个需要传送的文件，调用SendFileName方法发送文件，当所有文件发送成功后，释放端口资源。 |
| SendFileName(string file) | 文件路径 | 无 | 通过TCPClient连接向客户端发送文件的信息以及开放的端口，通知客户端可以连入接收文件 |
| SendFile(byte[] isSendBuffer, string fileName, FileProtocol protocol) | 文件名、文件信息 | 无 | 接收客户端的连接，在该链接上写入文件流，写完后释放该连接。 |