密级：**普通 秘密 机密 绝密**

VANET车载终端固件概要设计说明书

文件编号：xxxx-xx-xx

北京车网互联科技股份有限公司

2014年11月



版权说明

|  |
| --- |
| 本文件中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属北京车网互联科技股份有限公司所有，受到有关产权及版权法保护。  本文件及其附录的知识产权归北京车网互联科技股份有限公司所有。如无授权，请勿复制或传播。 |

用词规范

|  |
| --- |
| 1.必须（must）：优先实现，用P1表示  2. 应该（should）：建议优先，用P2表示  3. 有更好（nice-to-have）：最好能够实现，用P3表示  4-5 被过滤，根据实际需求情况重新予以考虑：预留，远期实现，用P4-P5表示 |

版本变更说明

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修改内容** | **修改人** | **修改日期** |
| V0.1 | 创建文档，确定目录结构 | 王义锋 | 2014-11-05 |
| V0.2 | 第3章总体设计部分基本完成；  第4.3节内部接口部分基本完成；  第7.1节收发帧处理模块设计基本完成； | 王义锋 | 2014-11-28 |
| V0.3 | 新增部分包括：  第4.1节用户接口  第6.6节无线MAC驱动模块设计  第7.2节LLC协议模块设计  第8.5节多跳转发模块设计（合并广播算法模块内容至此部分）  对下列部分进行了更新完善：  第2.2节设计需求定义  第3章总体设计  第4.3节内部接口  第7.1节收发帧处理模块设计 | 王义锋 | 2014-12-12 |
| V0.4 | 新增部分包括：  第7.3节DSMP协议模块设计；  第7.4节DSA协议模块设计；  第8章VANET中间件部分各模块设计；  第9章VSA部分各模块设计；  对下列部分进行了更新完善：  第8.5.3节广播路由算法设计； | 王义锋  王磊  葛学元  王文杰 | 2014-12-23 |
| V0.5 | 增加第8.5节多跳转发模块中的接口及实现原理部分 | 王义锋 | 2014-12-30 |
| V0.51 | 完善第8.5节多跳转发模块设计  在第8.4节中增加与多跳转发模块的接口定义 | 王义锋 | 2015-01-12 |
| V0.52 | 在第5.6节系统调试支持模块中增加osal\_log接口定义 | 王义锋 | 2015-02-16 |
| V0.53 | 更新第9.2节，9.3节，9.4节告警参数和描述 | 葛学元 | 2015-08-25 |
| V0.54 | 更新第6.6.2节，增加帧类型4 | 王磊 | 2015-09-06 |
|  |  |  |  |
|  |  |  |  |

目录

[1 引言 12](#_Toc429325026)

[1.1 编写目的 12](#_Toc429325027)

[1.2 适用范围 12](#_Toc429325028)

[1.3 术语定义 12](#_Toc429325029)

[1.4 参考资料 12](#_Toc429325030)

[2 任务概述 13](#_Toc429325031)

[2.1 系统概述 13](#_Toc429325032)

[2.2 设计需求规定 14](#_Toc429325033)

[2.3 运行环境规定 20](#_Toc429325034)

[3 总体设计 20](#_Toc429325035)

[3.1 设计综述 20](#_Toc429325036)

[3.1.1 硬件资源概况 20](#_Toc429325037)

[3.1.2 软件概况 22](#_Toc429325038)

[3.1.3 软件设计规范 24](#_Toc429325039)

[3.2 软件总体架构 27](#_Toc429325040)

[3.3 软件流程描述 30](#_Toc429325041)

[3.3.1 主要功能实现流程 30](#_Toc429325042)

[3.3.2 外部事件响应流程 31](#_Toc429325043)

[3.3.3 数据流处理流程 34](#_Toc429325044)

[3.4 数据结构设计 36](#_Toc429325045)

[3.5 出错处理设计 37](#_Toc429325046)

[3.6 软件设计要点 37](#_Toc429325047)

[4 接口设计 37](#_Toc429325048)

[4.1 用户接口 37](#_Toc429325049)

[4.1.1 基本界面 37](#_Toc429325050)

[4.1.2 预警提示界面 38](#_Toc429325051)

[4.1.3 预警提示类别 38](#_Toc429325052)

[4.2 外部接口 39](#_Toc429325053)

[4.3 内部接口 39](#_Toc429325054)

[4.3.1 接口关系 40](#_Toc429325055)

[4.3.2 接口1：系统接口 40](#_Toc429325056)

[4.3.3 接口2：VANET中间件接口 41](#_Toc429325057)

[4.3.4 接口3：网络传输层接口 43](#_Toc429325058)

[4.3.5 接口4：无线MAC层驱动接口 43](#_Toc429325059)

[4.3.6 接口5：数据加解密驱动接口 43](#_Toc429325060)

[4.3.7 接口6：其它外设驱动接口 44](#_Toc429325061)

[5 系统底层及公共部分 45](#_Toc429325062)

[5.1 系统初始化模块设计 45](#_Toc429325063)

[5.2 Rt-thread操作系统移植 45](#_Toc429325064)

[5.3 操作系统抽象层设计 45](#_Toc429325065)

[5.4 中断管理模块设计 45](#_Toc429325066)

[5.5 动态内存管理模块设计 45](#_Toc429325067)

[5.6 系统调试支持模块设计 45](#_Toc429325068)

[5.6.1 功能描述 45](#_Toc429325069)

[5.6.2 接口设计 45](#_Toc429325070)

[6 接口及外设驱动部分 47](#_Toc429325071)

[6.1 UART驱动模块设计 47](#_Toc429325072)

[6.2 SPI驱动模块设计 47](#_Toc429325073)

[6.3 GPIO驱动模块设计 47](#_Toc429325074)

[6.4 GPS驱动模块设计 47](#_Toc429325075)

[6.5 GSensor驱动模块设计 47](#_Toc429325076)

[6.6 无线MAC驱动模块设计 48](#_Toc429325077)

[6.6.1 功能描述 48](#_Toc429325078)

[6.6.2 接口设计 48](#_Toc429325079)

[6.6.3 实现原理 49](#_Toc429325080)

[6.7 Flash驱动模块设计 54](#_Toc429325081)

[6.8 按键驱动模块设计 54](#_Toc429325082)

[6.9 声音播放驱动模块设计 54](#_Toc429325083)

[6.10 LCD显示驱动模块设计 54](#_Toc429325084)

[6.11 数据加密驱动模块设计 54](#_Toc429325085)

[7 无线网络通信部分 55](#_Toc429325086)

[7.1 收发帧处理模块设计 55](#_Toc429325087)

[7.1.1 功能描述 55](#_Toc429325088)

[7.1.2 接口设计 55](#_Toc429325089)

[7.1.3 实现原理 60](#_Toc429325090)

[7.2 LLC协议模块设计 64](#_Toc429325091)

[7.2.1 功能描述 64](#_Toc429325092)

[7.2.2 接口设计 64](#_Toc429325093)

[7.2.3 实现原理 66](#_Toc429325094)

[7.3 DSMP协议模块设计 67](#_Toc429325095)

[7.3.1 功能描述 67](#_Toc429325096)

[7.3.2 接口设计 72](#_Toc429325097)

[7.3.3 加密的DSM结构 74](#_Toc429325098)

[7.3.4 实现原理 74](#_Toc429325099)

[7.4 DSA协议模块设计 \* 74](#_Toc429325100)

[7.4.1 功能描述 74](#_Toc429325101)

[7.4.2 接口设计 78](#_Toc429325102)

[7.4.3 实现原理： 79](#_Toc429325103)

[8 VANET中间件部分 79](#_Toc429325104)

[8.1 应用编程接口（API）模块设计 79](#_Toc429325105)

[8.2 车辆状态管理（VSM）模块设计 87](#_Toc429325106)

[8.3 车辆本地信息处理（LIP）模块设计 87](#_Toc429325107)

[8.4 DSRC消息协议（RCP）模块设计 87](#_Toc429325108)

[8.4.1 协议功能分析 88](#_Toc429325109)

[8.4.2 消息格式 89](#_Toc429325110)

[8.4.3 接口设计 104](#_Toc429325111)

[8.5 多跳转发模块设计 105](#_Toc429325112)

[8.5.1 功能描述 105](#_Toc429325113)

[8.5.2 广播路由算法设计 105](#_Toc429325114)

[8.5.3 接口设计 113](#_Toc429325115)

[8.5.4 实现原理 115](#_Toc429325116)

[9 车辆安全应用部分 118](#_Toc429325117)

[9.1 总体设计 118](#_Toc429325118)

[9.1.1 车辆安全应用在VANET软件架构中的层次 118](#_Toc429325119)

[9.1.2 车辆安全应用危险场景 119](#_Toc429325120)

[9.1.3 车辆相对位置分类 120](#_Toc429325121)

[9.1.4 车辆位置预处理模块 121](#_Toc429325122)

[9.1.5 相关数据结构 122](#_Toc429325123)

[9.2 近距离危险（CCW）模块设计 123](#_Toc429325124)

[9.2.1 功能描述 123](#_Toc429325125)

[9.2.2 接口设计 124](#_Toc429325126)

[9.2.3 实现原理 124](#_Toc429325127)

[9.3 紧急制动危险（EEBL）模块设计 126](#_Toc429325128)

[9.3.1 功能描述 126](#_Toc429325129)

[9.3.2 接口设计 127](#_Toc429325130)

[9.3.3 实现原理 128](#_Toc429325131)

[9.4 车辆故障危险（VBD）模块设计 129](#_Toc429325132)

[9.4.1 功能描述 129](#_Toc429325133)

[9.4.2 接口设计 129](#_Toc429325134)

[9.4.3 实现原理 130](#_Toc429325135)

[9.5 交叉路口碰撞预警模块(CICAS) 131](#_Toc429325136)

[9.5.1 功能描述 131](#_Toc429325137)

[10 系统部分 133](#_Toc429325138)

[10.1 系统控制模块设计 133](#_Toc429325139)

[10.2 人机交互模块设计 133](#_Toc429325140)

[10.3 参数管理模块设计 133](#_Toc429325141)

[10.4 固件更新模块设计 \* 133](#_Toc429325142)

[10.5 生产测试模块设计 \* 133](#_Toc429325143)

图目录

[图 2‑1 VANET系统拓扑图 13](#_Toc429325976)

[图 3‑1固件功能框图 22](#_Toc429325977)

[图 3‑2 目录结构 23](#_Toc429325978)

[图 3‑3 软件总体架构 27](#_Toc429325979)

[图 3‑4 功能实现流程 31](#_Toc429325980)

[图 3‑5 网络接收事件处理流程 32](#_Toc429325981)

[图 3‑6 本地车辆状态改变处理流程 33](#_Toc429325982)

[图 3‑7 用户事件处理流程 34](#_Toc429325983)

[图 3‑8 数据流处理流程 35](#_Toc429325984)

[图 3‑9 传输帧处理示意图 36](#_Toc429325985)

[图 3‑10 系统数据结构 37](#_Toc429325986)

[图 4‑1 基本界面 38](#_Toc429325987)

[图 4‑2 预警提示界面 38](#_Toc429325988)

[图 4‑3 预警提示类别 39](#_Toc429325989)

[图 4‑4 系统内部接口关系 40](#_Toc429325990)

[图 6‑1 无线MAC驱动模块接口关系 48](#_Toc429325991)

[图 6‑2 无线MAC驱动模块内部结构图 49](#_Toc429325992)

[图 6‑3 MAC帧类型1格式定义 50](#_Toc429325993)

[图 6‑4 MAC帧类型2格式定义 51](#_Toc429325994)

[图 6‑5 MAC帧类型3格式定义 52](#_Toc429325995)

[图 6‑6 MAC帧类型4格式定义 53](#_Toc429325996)

[图 6‑7 RT3070传输帧格式 54](#_Toc429325997)

[图 7‑1 收发帧处理模块接口关系 55](#_Toc429325998)

[图 7‑2 收发帧处理模块内部结构图 60](#_Toc429325999)

[图 7‑3 收发帧处理模块数据结构 60](#_Toc429326000)

[图 7‑4 上行数据流处理流程 63](#_Toc429326001)

[图 7‑5 下行数据流处理流程 64](#_Toc429326002)

[图 7‑6 LLC协议模块接口关系 65](#_Toc429326003)

[图 7‑7 LLC帧格式 67](#_Toc429326004)

[图 7‑8 DSMP封装的帧格式 68](#_Toc429326005)

[图 7‑9 DSMP/DSA扩展域格式定义 69](#_Toc429326006)

[图 7‑10 DSMP的数据流程 72](#_Toc429326007)

[图 7‑11 DSMP接口关系图 72](#_Toc429326008)

[图 7‑12 1609.2帧格式 74](#_Toc429326009)

[图 7‑13 VSIE帧承载WSA的定义 75](#_Toc429326010)

[图 7‑14 1609.2承载DSA的帧定义 75](#_Toc429326011)

[图 7‑15 DSA帧格式定义 76](#_Toc429326012)

[图 7‑16 Service setup & WSM delivery 77](#_Toc429326013)

[图 7‑17 DSA服务通道建立及数据交换流程 78](#_Toc429326014)

[图 7‑18 DSA接口关系图 78](#_Toc429326015)

[图 8‑1 MsgID定义 89](#_Toc429326016)

[图 8‑2 BSM消息帧格式 90](#_Toc429326017)

[图 8‑3 BrakeSystemStatus 定义 91](#_Toc429326018)

[图 8‑4 EventFlags 定义 92](#_Toc429326019)

[图 8‑5 BSM消息处理流程 93](#_Toc429326020)

[图 8‑6 EVAM消息帧格式 95](#_Toc429326021)

[图 8‑7 EVAM消息处理流程 97](#_Toc429326022)

[图 8‑8 EVA消息格式 98](#_Toc429326023)

[图 8‑9 EVAM消息处理流程 99](#_Toc429326024)

[图 8‑10 PVD消息格式 101](#_Toc429326025)

[图 8‑11 snapshot的发送流程图 102](#_Toc429326026)

[图 8‑12 PDM消息格式 103](#_Toc429326027)

[图 8‑13 rcp接口 105](#_Toc429326028)

[图 8‑14安全应用场景之一 107](#_Toc429326029)

[图 8‑15大量节点转发示意图 107](#_Toc429326030)

[图 8‑16数据包广播转发示意图 111](#_Toc429326031)

[图 8‑17 路由算法流程图 112](#_Toc429326032)

[图 8‑18 多跳转发模块接口关系 113](#_Toc429326033)

[图 8‑19 多跳转发模块内部结构图 115](#_Toc429326034)

[图 8‑20 多跳转发模块数据结构 116](#_Toc429326035)

[图 8‑21 多跳转发处理流程 117](#_Toc429326036)

[图 8‑22 超时处理流程 118](#_Toc429326037)

[图 9‑1车辆安全应用的软件架构层次 118](#_Toc429326038)

[图 9‑2车辆安全应用危险场景软件处理结构 120](#_Toc429326039)

[图 9‑3车辆相对位置分类 121](#_Toc429326040)

[图 9‑4位置预处理模块示意图 122](#_Toc429326041)

[图 9‑5位置数据结构组成 122](#_Toc429326042)

[图 9‑6 VSA部分数据结构 123](#_Toc429326043)

[图 9‑7前车近距离碰撞预警示意图 123](#_Toc429326044)

[图 9‑8后车近距离碰撞预警示意图 124](#_Toc429326045)

[图 9‑9近距离危险预警处理流程图 126](#_Toc429326046)

[图 9‑10紧急制动预警示意图 127](#_Toc429326047)

[图 9‑11紧急制动危险预警处理流程 129](#_Toc429326048)

[图 9‑12车辆故障报警流程图 131](#_Toc429326049)

[图 9‑13交叉路口碰撞危险示意图 132](#_Toc429326050)

[图 9‑14交叉路口碰撞示意图 132](#_Toc429326051)

表目录

[表 2‑1 固件设计需求 14](#_Toc429326052)

[表 3‑1 模块缩写 25](#_Toc429326053)

[表 3‑2 系统线程分配 29](#_Toc429326054)

[表 4‑1 系统消息定义 41](#_Toc429326055)

[表 4‑2 VANET中间件接口—上层应用 41](#_Toc429326056)

[表 4‑3 VANET中间件接口—获取车辆状态信息 42](#_Toc429326057)

[表 4‑4 VANET中间件接口—更新本地车辆状态 42](#_Toc429326058)

[表 4‑5 VANET中间件接口—应用层协议接口 43](#_Toc429326059)

[表 4‑6 网络传输层接口 43](#_Toc429326060)

[表 4‑7 无线网络MAC层接口 43](#_Toc429326061)

[表 4‑8 数据加解密接口 44](#_Toc429326062)

[表 4‑9 外设驱动接口函数 44](#_Toc429326063)

[表 4‑10 外设驱动接口事件 44](#_Toc429326064)

[表 5‑1 打印级别定义 45](#_Toc429326065)

[表 5‑2 打印控制接口 46](#_Toc429326066)

[表 5‑3 打印输出接口 47](#_Toc429326067)

[表 6‑1 drv\_wifi\_send原型定义 48](#_Toc429326068)

[表 6‑2 drv\_wifi\_mac\_header\_len原型定义 49](#_Toc429326069)

[表 6‑3 MAC帧类型1定制字段定义 50](#_Toc429326070)

[表 6‑4 MAC帧类型2定制字段定义 51](#_Toc429326071)

[表 6‑5 MAC帧定制字段定义 52](#_Toc429326072)

[表 6‑6 MAC帧类型2定制字段定义 53](#_Toc429326073)

[表 7‑1 wnet\_get\_txbuf原型定义 56](#_Toc429326074)

[表 7‑2 wnet\_release\_rxbuf原型定义 56](#_Toc429326075)

[表 7‑3 WNET\_TXBUF\_DATA\_PTR原型定义 56](#_Toc429326076)

[表 7‑4 WNET\_TXBUF\_INFO\_PTR原型定义 57](#_Toc429326077)

[表 7‑5 WNET\_RXBUF\_PTR原型定义 57](#_Toc429326078)

[表 7‑6 wnet\_send原型定义 57](#_Toc429326079)

[表 7‑7 wnet\_recv原型定义 58](#_Toc429326080)

[表 7‑8 wnet\_send\_complete原型定义 58](#_Toc429326081)

[表 7‑9 fp\_send原型定义 59](#_Toc429326082)

[表 7‑10 llc\_send原型定义 65](#_Toc429326083)

[表 7‑11 llc\_recv原型定义 65](#_Toc429326084)

[表 7‑12 LLC帧EtherType值 67](#_Toc429326085)

[表 7‑13 DSMP帧头AID定义 68](#_Toc429326086)

[表 8‑1 mda\_handle原型定义 113](#_Toc429326087)

[表 8‑2 rcp\_create\_forward\_msg原型定义 114](#_Toc429326088)

[表 8‑3 rcp\_send\_forward\_msg原型定义 114](#_Toc429326089)

# 引言

## 编写目的

本文档的编写目的是根据《软件需求说明书》的规定，说明软件总体设计方案和架构，软件模块的划分和接口设计，并对模块具体功能的设计思想进行概要说明，从而指导软件的编码实现。

## 适用范围

本文档的预期读者为软件开发人员、软件测试人员项目管理人员。

## 术语定义

VANET：Vehicular Ad-hoc NETwork车辆自组织网络

V2V：Vehicle To Vehicle，即车和车

V2I：Vehicle To Infrastructure，即车和设施

OBU：On board unit，即车载单元

RSU：Road Side unit，即路侧单元

CMS：车载终端

PMS：路政移动站

BTS：路侧固定站

## 参考资料

1. 《车车通信标准项目立项建议书》
2. 《CMS原型机设计需求说明书》
3. 《CMS原型机总体设计方案》

# 任务概述

## 系统概述

本产品是一款应用于车载环境的无线报警终端产品，通过该产品能提供车辆与车辆之间超视距条件下的无线通信，并对影响驾驶安全的潜在危险进行报警，实现车车的协同安全。如下图所示VANET系统中的CMS终端。

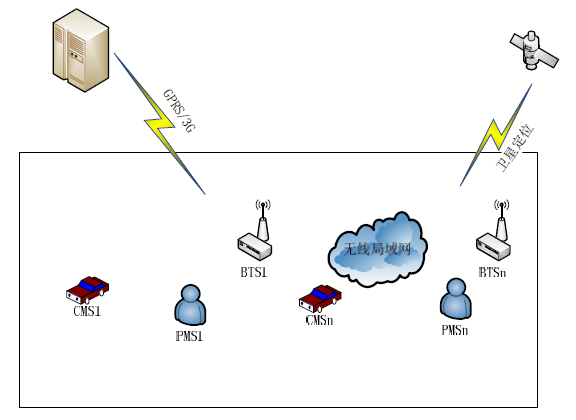


图 2‑1 VANET系统拓扑图

上述系统旨在在解决高速公路、山区道路或恶劣环境行车状况下超出行车视距范围的车辆故障、事故等因素导致的有碍正常行车因素的预先告警。系统采用短距离无线通信终端完成一定区域范围内的自组网，该区域网某一终端或某些终端的车辆出现妨碍后续车辆正常行车的因素或发现行驶路径上出现妨碍后续同向车辆正常行车的因素后将该警情发送给区域网内其他终端。整个系统由以下四部分组成，包括：车载终端（CMS）、路侧固定站（BTS）和路政移动站（PMS）和系统后台。

本文所述为VANET系统中的车载终端CMS产品，主要功能特点如下：

* 支持VANET车辆自组织网络，
* 支持车与车间直接通信及多跳转发，有效保证通信距离
* 支持车与设施间通信，提供多种安全信息服务
* 支持无线数据传输加密（AES），充分保证通信安全
* 支持多种场景的行车危险检测与告警提示
  + 近距离危险(前方/后方)
  + 紧急制动危险
  + 故障车辆危险（手动/碰撞/翻车）
  + 弯道/交叉口会车危险提示（需路侧终端配合）
  + 特殊道路（急弯/隧道/桥梁/十字路口/...）提示（需路侧终端配合）
  + 电子交通标识（信号灯/限速/...）提示（需路侧终端配合）

## 设计需求规定

本软件总体功能需求如下表所示：

表 2‑1 固件设计需求

|  |  |
| --- | --- |
| 项 目 | 要 求 |
| 操作系统 | 操作系统适配层接口 |
| RTOS内核，支持包括：   * 支持抢占式任务调度 * 支持任务同步机制（信号量、消息） * 支持中断管理 * 支持定时器 * 支持动态内存管理 |
| 驱动 | 接口   * UART接口读写 * SPI接口读写 * USB接口读写 * I2S接口读写 * GPIO读写 |
| 设备   * WiFi网卡 * G-Sensor * Audio Codec * LCD * Flash |
| 无线网络 | 支持自组网 |
| 支持多跳网络传输，最大3跳 |
| 支持广播路由算法 |
| 无线MAC帧的组帧和解帧 |
| 数据收发队列管理 |
|  |
| VANET中间件 | 支持车辆远程通信协议，包括：   * 支持的消息类型   MSG\_BasicSafetyMessage：基本安全消息  MSG\_EmergencyVehicleAlert：紧急车辆警告消息   * 消息编解码 |
| 支持车辆本地信息处理，包括：   * GPS（NMEA）消息解析，获取车辆位置、速度、方向和标准时间信息 * G-Sensor数据解析，获取车辆加速度数据 |
| 支持车辆信息管理，支持如下：   * 本车状态维护 * 邻车状态维护 * 本车状态消息自动发送，发送周期P=d\*1000/r，其中：   P表示发送周期，单位ms，最小精度10ms，强制取值范围100ms~3000ms  d表示距离因子，单位m，默认取值为5  r表示车速，单位为m/s |
| 上层应用接口，包括：   * 读取本车信息接口 * 读取邻车信息接口 * 发送报警消息接口 * 工作模式设置接口（包括：启动/停止本车状态自动发送） * 本车状态更新事件处理回调 * 邻车状态更新事件处理回调 * 邻车报警事件处理回调 |
| 安全应用 | 紧急制动危险报警：  当本车检测到速度急剧降低时，向周围车辆发出紧急制动危险报警。   * 输入参考量： * 本车速度、加速度、行驶方向 * 触发条件： * 前向行驶 * 时速大于30Km/h * 加速度为负（即减速）且绝对值大于0.3g（3m/s2） * 解除条件： * 持续时间到（5s） * 其它要求： * 报警消息发送间隔50ms * 报警消息支持多跳传输 |
| 车辆事故（碰撞/翻车）报警：  当本车高速行驶时，遇突发事故导致车辆急停或倾覆时，自动向周围车辆发出危险报警。   * 输入参考量： * 本车速度、加速度、行驶方向 * 触发条件： * 符合紧急制动危险触发条件 * 或车辆侧倾角度超过90度 * 车速降至并保持在0 * 解除条件： * 以下任一条件时解除 * 车辆再次行驶，时速大于10Km/h * 用户手动取消 * 其它要求： * 报警消息发送间隔50ms * 报警消息支持多跳传输 |
| 手动故障报警：  当本车因故需要非正常停车或低速行驶时，可由用户设置手动故障模式，向周围车辆发出危险报警。   * 输入参考量：   无   * 触发条件： * 用户手动设置 * 解除条件： * 用户手动解除 * 其它要求： * 报警消息发送间隔50ms * 报警消息支持多跳传输 |
| 紧急制动危险检测预警提示：  当本车接收到前车发出的紧急制动危险报警消息时，向本车发出危险预警提示。   * 输入参考量： * 本车速度、行驶方向 * 邻车加速度、行驶方向、相对位置 * 邻车紧急制动报警消息 * 触发条件： * 本车时速大于30Km/h时，启动紧急制动危险检测 * 本车前向行驶 * 邻车前向行驶 * 本车位于邻车后方 * 接收到邻车紧急制动报警消息 * 解除条件： * 上述任一条件不满足时 * 一定时间内（5s）不再接收到同一报警消息 * 用户手动关闭预警时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..危险请注意” * 其它要求：   无 |
|  |
| 车辆故障危险检测预警提示：  当本车接收到前车发出的车辆危险报警消息时，向本车发出危险预警提示。   * 输入参考量： * 本车速度、行驶方向 * 邻车相对位置 * 邻车车辆故障报警消息 * 触发条件： * 本车时速大于30Km/h时，启动紧急制动危险检测 * 本车前向行驶 * 本车位于邻车后方 * 接收到邻车车辆故障报警消息 * 解除条件： * 上述任一条件不满足时 * 户手动关闭预警时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..危险请注意” * 其它要求：   无 |
| 近距离危险检测预警提示：  当本车检测到与前车逐渐接近时，并小于安全距离时，向本车发出危险预警提示。   * 输入参考量： * 本车速度、行驶方向 * 邻车速度、行驶方向、相对位置 * 两车距离 * 触发条件： * 本车时速大于30Km/h时，启动与前车的距离检测 * 本车前向行驶 * 邻车前向行驶或静止 * 本车位于邻车后方 * 本车速度大于邻车速度 * 两车距离小于D=（2\*ra-rb）\* k，其中ra表示本车速度,rb表示邻车速度，单位m/s，k表示安全系数，默认取值为3 * 解除条件： * 上述任一条件不满足时 * 用户手动关闭预警时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率2~5Hz * 声音：仅在初次触发预警时播放一次，“哔哔” * 其它要求：   无 |
| 会车危险预警提示  当本车接收到路侧设备发出的会车危险预警消息时，向本车发出危险预警提示。   * 输入参考量： * 本车速度、行驶方向 * 路侧设备相对位置 * 会车危险报警消息 * 触发条件： * 本车时速大于30Km/h时 * 本车前向行驶 * 接收到会车危险报警消息 * 解除条件： * 上述任一条件不满足时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..请注意对面车辆” * 其它要求：   无 |
| 特殊道路预警提示  当本车接收到路侧设备发出的特殊道路危险预警消息时，向本车发出危险预警提示。   * 输入参考量： * 本车速度、行驶方向 * 路侧设备相对位置 * 会车危险报警消息 * 触发条件： * 本车时速大于30Km/h时 * 本车前向行驶 * 接收到会车危险报警消息 * 解除条件： * 上述任一条件不满足时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..请注意前方XX道路” * 其它要求：   无 |
| 交通标识预警提示  当本车接收到路侧设备发出的交通标识预警消息时，向本车发出预警提示。   * 输入参考量： * 本车速度、行驶方向 * 路侧设备相对位置 * 会车危险报警消息 * 触发条件： * 本车时速大于30Km/h时 * 本车前向行驶 * 接收到会车危险报警消息 * 解除条件： * 上述任一条件不满足时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..请注意前方XX” * 其它要求：   无 |
| 紧急车辆预警提示  当本车接收到紧急车辆发出的警告消息时，向本车发出预警提示。   * 输入参考量： * 本车速度、行驶方向 * 紧急车辆的速度、行驶方向 * 紧急车辆的相对位置 * 紧急车辆警告消息 * 触发条件： * 紧急车辆位于本车后方 * 接收到紧急车辆警告消息 * 解除条件： * 上述任一条件不满足时 * 预警提示要求： * 灯光：在预警状态下持续闪烁，频率5Hz * 声音：循环播放，“哔哔..后方有XXX，请注意避让” * 其它要求：   无 |
| 人机接口 | 工作状态指示（绿色LED灯）   * 闪烁 – GPS定位中 * 常亮 – GPS已定位 * 常灭 – 未定义 |
| 危险预警指示（红色LED灯）   * 闪烁 – 危险预警中（频率随危险程度变化） * 常亮 – 未定义 * 常灭 – 待机中 |
| 手动报警状态指示（黄色LED灯）   * 闪烁 – 手动故障报警状态 * 常亮 – 未定义 * 常灭 – 待机中 |
| 预警提示语音   * 提示音1：“哔哔” * 提示音2：“请注意前方车辆” * 待补充。。。 |
| 预警提示图形显示   * 根据预警提示类型显示对应的图标及文字 |
| 手动故障报警按键   * 单击 – 启动/停止 |
| 功能按键1   * 单击 – 未定义 |
| 功能按键2   * 单击 – 未定义 |
| 系统管理 | 工作模式管理 |
| 配置参数管理 |
| 人机交互管理 |
| 其它功能 |
|  |
| 测试维护 | 打印系统调试信息 |
| 配置系统工作参数 |
|  |

## 运行环境规定

无

# 总体设计

## 设计综述

### 硬件资源概况

#### CPU

* CPU资源

**CPU：**

STM32F405RGT6，主频168MHz，其内核采用的是Arm Cortex-M4，支持浮点运算。

#### 存储器

**Flash：**

1Mbyte FLASH（内置）

内部flash分配（暂定，可根据实际应用修改），暂存区域保存升级临时文件，地址范围要大于app。

|  |  |  |  |
| --- | --- | --- | --- |
| 名 称 | 大 小 | 地址范围 | 说明 |
| BOOT | 64KB | 0x8000000~0x8010000 | 存放Bootloader程序 |
| QC | 64KB | 0x8010000~0x8020000 | 存放生产测试程序 |
| APP | 384KB | 0x8020000~0x8080000 | 存放应用主程序和升级备份程序 |
| PARAMETER | 1KB | 0x80E0000~0x80E0400 | 存放系统工作参数 |
| RESOURCE | 127KB | 0x80E0400~0x8100000 | 存放声音数据 |

**RAM：**

112+16+64Kbyte （内置）

|  |  |  |  |
| --- | --- | --- | --- |
| 名 称 | 大 小 | 地址范围 | 说明 |
|  | 128KB | 0x20000000~0x20020000 |  |
|  | 64KB | 0x10000000~0x10010000 | 零等待RAM,不能用DMA，可用作高速数据Buffer |
|  |  |  |  |

### 软件概况

#### 功能划分



图 3‑1固件功能框图

本软件从从功能上可以划分为下面的几大部分：

* 系统底层及公共功能部分：

实现系统软件的底层支持，包括操作系统、中断管理、动态内存管理及调试支持等功能。

* 接口及外设驱动部分：

实现系统中的设备驱动，包括了BSP层和设备驱动层两个部分，实现了系统中用到的所有硬件接口及设备的驱动程序。

* 无线网络通信部分：

主要实现无线网络管理、网络层协议、传输层协议、数据收发管理等功能。

* VANET中间件部分：

主要实现针对基于VANET车联网V2V、V2I应用的协议及功能，为应用层提供统一的接口。

* 车辆安全应用部分：

主要实现基于车辆行车安全领域的应用，包括各种危险检测与报警功能。

* 系统部分：

主要实现做为一个独立终端设备所必需的相关系统功能，包括系统控制、人机接口、参数管理等。

#### 开发环境

* 编译环境：Keil-MDK 版本4.54以上
* 仿真工具：Jlink
* 操作系统：RT-thread

#### 软件别名

【Cygnoides】

鸿雁，是大雁的一种。大雁是出色的空中旅行家，在长途旅行中，大雁通常以6只或6只的倍数组成雁群，队伍组织得十分严密，它们常常排成人字或一字形，它们一边飞着，还不断发出“嘎、嘎”的叫声。大雁的这种叫声起到互相照顾、呼唤、起飞和停歇等的信号作用。与大雁的叫声类似，V2V的作用就是在车辆的不断的发出“声音”，与周围的车辆进行相互提醒，从而保证行车的安全。

#### 目录结构



图 3‑2 目录结构

* Cygnoides：

项目根目录。

* app：

应用程序子目录，用于存放系统的应用层程序源文件，主要为C源文件，包括VANET中间件模块、无线网络通信模块、车辆安全应用模块、系统模块等。

* bin：

二进制文件子目录，用于存放生成的固件二进制映像文件。

* bsp：

设备驱动层中板级支持包子目录，用于存放于板级硬件相关的驱动程序，主要为C源文件，包括接口与外设硬件配置功能等。

* drivers：

设备驱动层子目录，用于存放系统的接口与设备驱动源文件，主要为C源文件，包括Mac收发驱动、gps驱动、gsensor驱动等。

* os：

系统底层及公共功能子目录，用于存放操作系统及相关公共功能模块（如动态内存管理、调试支持等），包括C源文件和汇编源文件，本程序采用开源的RTT实时内核。

* include：

头文件子目录，用于存放系统的公用头文件，主要为H源文件。

* library：

库子目录，用于系统提供的库及相应头文件，主要为.lib或.a文件。

* prj\_keil：

程序项目文件子目录，用于存放Keil环境下的主程序工程文件，编译文件等。

* prj\_si：

程序项目文件子目录，用于存放source insight工程文件等。

* startup：

启动文件子目录，用于存放与系统初始化相关的程序源文件，包括C源文件河汇编源文件。

* tools：

系统工具子目录，用于存放系统所需的软件工具，如编译控制、BIN文件生成、格式转换等，主要为可执行程序和脚本。

### 软件设计规范

#### 编程规范

参见《软件编程格式规范》。

#### 模块名称规范

本项目前缀为cv，各功能模块名称定义采用2~4个小写英文字符的单词缩写，如下表所示，

表 3‑1 模块缩写

|  |  |  |  |
| --- | --- | --- | --- |
| 主模块名称 | 次模块名称 | 缩 写 | 备注 |
| 系统底层及公共部分 |  | os | OS Adapter Layer |
|  | 操作系统 | rtt | RT-thread |
| 动态内存管理 | dmem |  |
| 调试模块 | dbg |  |
|  |  |  |
| 接口及外设驱动部分 |  | drv |  |
|  | 串口 | uart |  |
| SPI接口 | spi |  |
| I2C接口 | iic |  |
| I2S接口 | iis |  |
| GPIO | gpio |  |
| GPS | gps |  |
| 加速度传感器 | gsnr |  |
| Audio Codec | auc |  |
| LCD控制器 | lcd |  |
| LED指示灯 | led |  |
| 按键输入 | key |  |
| 语音播放 | voc |  |
| 图形显示 | disp |  |
| WiFi MAC | wifi |  |
| Flash | fls |  |
| 加密模块 | cryp |  |
| 无线网络通信部分 |  | wnet | Wireless Network Transport |
|  | 数据帧收发管理 | fp | Frame Process |
| DSMP协议 | dsmp |  |
| DSA协议 | dsa |  |
|  |  |  |
|  |  |  |
| VANET中间件部分 |  | vam | VANET Middleware |
|  | 应用编程接口 | api |  |
| 车辆状态管理 | vsm | Vehicle Status Management |
| DSRC消息协议 | rcp | DSRC-Message Protocol |
| 车辆本地信息处理 | lip | Local Information Process |
| 多跳转发算法 | mda | Multihop Distributed Algorithm |
| 车辆安全应用部分 |  | vsa | Vehicle Safty Application |
|  | 近距离危险检测 | ccw | Cooperative Collision Warning |
| 紧急制动危险检测 | eebl | Emergency Electronic Brake Lights |
| 车辆故障报警 | vbd | Vehicle Break Down |
| 会车危险 | vmd | Vehicle Meeting Danger |
| 特殊道路提示 | drn | Dangerious road Notice |
| 交通标示提示 | tsn | Traffic Sign Notice |
|  |  |  |
| 系统部分 |  | sys |  |
|  | 系统控制 | ctrl |  |
| 人机交互 | ui |  |
| 参数管理 | pm |  |
| ….. |  |  |

#### 软件版本规则

软件版本格式为“VX.Y.ZZZ[M,Build\_time]”，其中：

* V：软件版本标识，取固定字符’V’
* X：主版本号，初始值为1，用于表示固件重大更新，每次递加1
* Y：次版本号，初始值为0，用于表示升级更新的版本，每次递加1，当主版本号更新时归零
* ZZZ：内部版本号，仅用于内部软件开发和测试，取值范围000~999，每次软件更新递加1，在软件生命周期内单向递加,不归零
* M：版本类别，可取标识dbg（测试版本）或rel（发布版本）
* Build\_time：版本生成时间（采用Keil编译器预定义宏\_\_TIME\_\_ \_\_DATE\_\_）
* 示例：V1.0.123[dbg,15:32:26 Nov 10 2014]

## 软件总体架构



图 3‑3 软件总体架构

上图所示为固件程序的总体架构，从软件功能上主要分为以下六个部分：系统底层及公共功能、接口及外设驱动、无线网络通信、VANET中间件、车辆安全应用和系统。下面分别对各部分功能及相互关系进行说明。

* 系统底层及公共功能

此部分提供的功能包括：系统初始化、嵌入式操作系统、动态内存管理、调试支持模块及操作系统适配层（OSAL）。

系统初始化功能用来完成CPU的引导及基本工作环境配置，如中断、时钟配置等，此模块采用的是Arm公司为Cortex-M系列内核提供的参考代码及CMSIS框架，只需针对所使用的处理器和RTOS进行少量配置即可。

本软件采用的是RT-thread嵌入式操作系统。RT-Thread是包含了一个支持抢占的实时操作系统内核，除了提供基本的线程、同步、通信等机制外，同时也提供了一些轻量级的嵌入式软件平台组件，如动态内存管理、设备驱动框架、TCP/IP协议栈、GUI等。为了保证本固件中核心功能部分代码的可移植性，本系统中仅使用了RT-thread最基本的线程管理及相关的同步通信机制。如下面所介绍的动态内存管理功能。

动态内存管理，本系统采用了RT-Thread内部的动态内存管理功能。

调试支持模块实现的是为整个系统提供统一的打印输出控制功能，支持模块级的打印级别和打印开关控制。

操作系统适配层（OSAL），主要是为了提高上层软件代码的可移植性，对系统底层（包括操作系统、动态内存等公共功能）提供统一的抽象接口API。

* 接口及外设驱动

此部分实现的是系统中所有硬件接口与设备的管理与操作功能。本部分由下到上可以分成三个层次（参见图中灰色区域）：HAL层、BSP层和DRIVER层。

HAL层：主要是面向主处理器芯片，提供的是寄存器访问和底层操作接口，此部分采用的是由主芯片原厂（即ST）提供的官方程序库。本系统使用到的库包括ST标准外设库和USB Host（Otg）驱动库。

BSP层：主要是面向板级，实现硬件接口和其它外接硬件设备的寄存器级配置和操作接口。在本系统中，UART和SPI接口使用了RT-thread提供的虚拟文件访问机制的驱动框架来实现，而其它部分则直接基于HAL层实现。本系统中需要支持如下接口UART、SPI、USB、I2S、GPIO和外设硬件（GPS、Gsensor、LCD、AudioCodec）。

DRIVER层：主要实现抽象的设备管理和统一的应用接口，通常只区分设备类型，而与具体所使用的硬件型号无关。本系统需要实现的设备驱动包括：GPS、Gsensor、Flash、文本/图形显示、声音播放、按键输入以及*数据加密*。

* 无线网络通信

此部分主要实现系统中数据通信部分的收发队列管理机制、网络传输层协议和广播路由算法。

收发帧处理，实现的是与具体MAC层实现无关的数据收发队列管理机制。

LLC，实现的是数据链路层封装功能。

网络传输层协议，在VANET网络层正在制定中的国家标准中，以美国版的WAVE（Wireless Access in Vehicular Environments）标准中的传输层协议WSMP/WSA为基础对进行了部分修正，称之为DSMP/DSA。本系统的传输层将同时参照上述两种标准协议来制定，详见相关模块的设计章节。的主要包括无线数据帧的解帧和组帧，无线自组网的管理、多跳路由算法以及网络传输层协议的实现。本部分向下与MAC驱动接口，向上为应用层提供数据传输接口，同时为系统部分提供网络管理与控制接口。

* VANET中间件

此部分主要实现针对车辆联网中V2V和V2I通信的所需的应用层中间件，包括上层应用接口、车辆状态管理、车辆远程通信协议、车辆本地信息处理共四部分。本部分向下与网络通信部分以及GPS、加速度传感器等驱动接口，向上为应用层提供统一的应用接口。

上层应用接口：用来实现一层适用于VANET领域的如车辆安全、交通信息共享等应用的统一接口，所有上层应用逻辑的实现都可基于应用接口完成。

车辆状态管理：用来实现对车辆（包括本车及所有邻车）状态信息的更新维护和本车状态信息发送的管理。

车辆远程通信协议：用来实现VANET通信中各种信息（如车辆状态、交通信息）交换所需的消息格式定义与编码和传输机制。

车辆本地信息处理：用来实现VANET通信中车辆自身状态信息的处理，如GPS信息的解析、OBD协议的解析等。

多跳转发算法，主要实现的是VANET网络中的广播多跳转发功能，此模块功能相对比较独立，在软件实现中要基于统一的接口，实现算法模块的可替换性。

* 车辆安全应用

此部分主要实现基于VANET的具体应用，在本产品中现阶段实现如下六个应用，其中：近距离危险预警、紧急制动预警和手动故障报警为基于V2V的行车危险自动检测与报警功能，而会车危险提示、特殊道路提示和交通标识提示功能则基于V2I，需要与路侧设备配合来实现。在本软件中，对于后三个功能只需实现能接收相应信息并向用户发出预警提示即可。

本部分的实现全部基于VANET中间件的应用接口，且不同的应用的具体逻辑实现相互独立，从而保证应用的可移植性和可扩展性。

* 系统功能

此部分主要实现作为一个独立设备的固件程序所必需的系统管理等相关功能。在本产品中包括系统控制、人机交互控制、参数管理等功能模块。

系统控制，在本软件中，此模块仅实现为系统中其它部分（主要是车辆安全应用和人机接口）之间的消息分配和转发功能。

人机交互，本系统中如下方式的人机交互接口：按键输入、LED指示灯输出、语音输出和LCD显示输出。本模块主要实现对人机交互事件的整体协调，保证各个部分的输出状态一致性，同时为系统其它部分提供抽象的输入输入接口，屏蔽具体的物理实现。

参数管理，用来实现对系统中所有配置参数的维护和管理功能，并为系统中其它部分提供统一的参数管理接口。

* 系统线程分配

（注：此表中不含操作系统内部保留的线程，如Idle、Timer管理等线程）

表 3‑2 系统线程分配

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 | 名 称 | 优先级 | 说 明 |
| 接口及外设驱动部分 | GPS主线程 | Highest-6 | 通过Uart接口GPS接收（Uart接口） |
| Gsnr主线程 | Highest-8 | Gsensor接收（SPI接口） |
| Key主线程 | Highest-22 | 按键检测 |
| USB主线程 | Highest-2 | USB Host事件处理等 |
| USB枚举线程 | Highest-4 | USB 设备枚举（仅初始化过程运行一次） |
| 数据加解密线程 | Highest-11 | 数据加解密事件处理 |
| 无线网络通信部分 | 网络接收线程 | Highest-12 | 网络上行接收数据处理 |
| 网络发送线程 | Highest-10 | 网络下行发送数据处理 |
| VANET中间件部分 | VANET主线程 | Highest-14 | VANET事件处理 |
| 多跳转发线程 | Highest-9 | 多跳转发消息处理 |
| 车辆安全应用部分 | VSA主线程 | Highest-16 | VSA事件处理 |
| 系统部分 | 系统控制主线程 | Highest-18 | 系统控制事件，包括系统消息分发、系统级功能实现等 |
| 人机交互处理线程 | Highest-20 | 负责人机接口事件的处理，包括语音输出、LED指示灯输出、LCD显示输出 |
| 初始化线程 | Highest | 系统各部分初始化（仅初始化过程运行一次） |

## 软件流程描述

### 主要功能实现流程

本软件实现的主要功能是通过无线通信手段实现车辆之间实时信息交换，并对影响驾驶安全的潜在危险进行判断和报警，主要功能实现流程如下图所示：



图 3‑4 功能实现流程

功能实现流程要点如下：

* 系统通过GPS和Gsensor来周期性采集车辆的位置、行驶方向、行驶速度及加速度等信息，并提交给VANET中间件部分处理
* VANET中间件部分根据上述信息生成本车车辆状态消息，并通过无线传输协议进行周期性的广播发送
* 同时，无线传输部分也会将从无线网路中接收到的其它车辆状态消息提交给VANET中间件部分
* VANET中间件部分对接收到车辆状态消息进行解析，获得其位置、行驶速度等信息，并维护一个本车的邻居车辆列表
* 车辆安全应用部分周期性从VANET中间件部分获取本车与邻车的行驶状态信息（对于紧急事件，采用VANET主动触发方式），并判断如果存在危险状况，则通过人机接口部分发出报警提示
* 除了基本的车辆状态消息，VANET中间件部分还支持路侧设备公告、紧急车辆等其它类型消息的接收和解析，并通过车辆安全应用及人机接口部分根据实际场景向用户发出提示

### 外部事件响应流程

本系统的运转主要依靠外部（及周期定时）事件的触发，主要包括网络接收到的消息、本地车辆状态改变和用户输入三类事件，下面分别进行说明。

* **网络接收事件处理流程**



图 3‑5 网络接收事件处理流程

1. 网络接收驱动部分将从无线网络接收到的存放到网络数据接收缓冲区，然后发送消息通知网络接收处理线程；
2. 网络接收线程对数据包进行网络传输协议解析，如果是应用层数据则发送消息通知VANET中间件处理线程；
3. VANET中间件处理线程对接收到的应用数据进行解析与处理，并根据需要通知车辆安全应用处理线程；
4. 车辆安全应用处理线程对接收到的事件进行处理，如果需要生成报警提示，则通知系统控制处理线程；
5. 系统控制线程统一通知人机接口处理线程输出相应的报警提示；

* **本地车辆状态改变事件处理流程**



图 3‑6 本地车辆状态改变处理流程

1. 中断处理程序从GPS或GSensor接收到事件或数据后，封装成消息通知VANET中间件处理线程；
2. VANET中间件处理线程对接收到的事件或数据进行解析，获取本地车辆状态改变的信息，然后生成相应的事件，并根据需要通知车辆安全应用线程；
3. 车辆安全应用线程对接收到的事件进行处理，如果需要生成报警消息则通知VANET中间件处理线程，如果需要生成本地报警提示，则通知系统控制线程并转到步骤6；
4. VANET中间件处理线程对输入的消息进行封装后发送给网络发送处理线程；
5. 网络发送处理线程负责将该报警消息发送到网络中；
6. 系统控制控制线程统一通知人机接口处理线程输出相应的报警提示；

* **用户事件处理流程**



图 3‑7 用户事件处理流程

1. 人机接口处理线程接收并封装用户输入事件，然后通知系统控制线程；
2. 根据输入事件类型，如果是启动手动报警，则系统控制线程通知车辆安全应用线程；
3. 车辆安全应用线程对接收到的事件进行处理，如果条件满足则生成报警消息并将其发送VANET中间件处理线程；
4. VANET中间件处理线程对输入的消息进行封装后发送给网络发送处理线程；
5. 网络发送处理线程负责将该报警消息进行传输层协议封装并发送到网络中；

### 数据流处理流程

本系统中的数据流主要指的是无线网络通信部分队列管理机制为核心的数据收发处理流程，分为上行（即接收数据）和下行（即发送数据）两个方向，如下图所示。



图 3‑8 数据流处理流程

以下所示为数据传输帧在进行组帧发送流程中，系统各个部分的处理示意图（接收帧格式相同）：



图 3‑9 传输帧处理示意图

## 数据结构设计

本系统以数据结构“cms\_global\_t”为系统的核心数据结构，管理完成系统主要功能的系统管理模块数据结构、车辆安全应用模块数据结构、VANET中间件模块数据结构、无线通信模块数据结构以及接口与设备驱动模块数据结构等系统核心数据结构，系统核心数据结构间的关系如下图所示：



图 3‑10 系统数据结构

## 出错处理设计

无。

## 软件设计要点

本软件主要实现了基于VANET的车载危险报警终端功能，其中在软件设计中需要重点关注的是在VANET应用中的公共功能部分，包括：VANET中间件、车辆安全应用、无线网络通信中的传输层协议和广播路由协议。为了实现这些公共功能模块的可复用，需在软件设计编码中充分考虑代码的可移植性。

# 接口设计

## 用户接口

本产品提供三种形式的用户接口，语音播放、LED指示灯和LCD图形显示。本节内容重点描述图形显示接口的用户界面定义（注：本系统中的LCD仅用于显示，不支持用户输入功能）。

### 基本界面



图 4‑1 基本界面

### 预警提示界面



图 4‑2 预警提示界面

### 预警提示类别



图 4‑3 预警提示类别

## 外部接口

本软件是一个可独立运行的产品固件程序，无外部软件接口。

## 内部接口

此处的内部接口指的是整个系统软件中主要功能部分（即第3.1.2.1节中所述的软件六大部分）之间的接口，其余功能部分内部子模块之间的接口详见各模块设计章节。此外，对于如操作系统接口、初始化等系统中公共功能接口，这里不再赘述。

### 接口关系



图 4‑4 系统内部接口关系

本系统内部接口主要分为三个类型：函数接口、消息接口和数据接口，函数接口主要是指系统各模块对外提供的API接口，供系统其它模块调用，消息接口主要用来实现系统模块间的事件交互，可用于中断->线程、线程<->线程之间，数据接口主要指网络传输的批量数据传递。本节主要说明固件程序内部主要部分间接口关系，包括：

### 接口1：系统接口

系统部分与其它部分之间的接口，主要采用消息的方式，使用统一的消息结构，定义如下（需要说明的是，本系统中其它部分及子模块间的消息同样采用此消息结构）：

|  |
| --- |
| typedef struct \_sys\_msg {  uint16\_t id;  uint16\_t len;  uint32\_t argc;  voidt\* argv;  } sys\_msg\_t; |

其中：

* id：消息ID，由两个部分组成，高8位表示消息所属模块（通常为处理该消息的线程）id，低8位表示消息类型
* len：参数长度，表示的是参数的长度，具体含义根据消息类型定义
* argc：参数1
* argv：参数2，此部分可指向动态申请的可变长度消息的参数区

本系统中支持的消息包括：

表 4‑1 系统消息定义

|  |  |  |
| --- | --- | --- |
| 消息名称 |  | 说明 |
|  | | |
| SYS\_MSG\_BASE | 0x0000 | 所有由系统控制模块处理的消息的基址 |
| SYS\_MSG\_INITED |  |  |
| SYS\_MSG\_KEY\_PRESSED |  |  |
| SYS\_MSG\_KEY\_RELEASED |  |  |
|  |  |  |
|  |  |  |
| …… |  |  |

### 接口2：VANET中间件接口

VANET中间件模块提供给系统中其它部分的接口，为了保证中间件部分的模块化设计，其所有接口全部采用API形式，即函数接口。这些接口包括：

* 提供给应用程序即车辆安全应用部分的应用API接口

表 4‑2 VANET中间件接口—上层应用

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| vam\_init | VANET中间件初始化 |
| vam\_deinit | VANET中间件反初始化（卸载） |
| vam\_start | VANET中间件启动服务 |
| vam\_stop | VANET中间件停止服务 |
| vam\_get\_config | 获取工作参数 |
| vam\_set\_config | 设置工作参数 |
| vam\_set\_event\_handler | 设置事件回调函数接口 |
| …… |  |
| vam\_get\_local\_status | 获取本车的状态信息 |
| vam\_set\_local\_status | 更新本车的状态信息（仅用于当VANET不支持内部解析本地GPS、加速度传感器等功能时） |
| vam\_get\_peerlist | 获取邻车节点列表 |
| vam\_get\_peer\_status | 获取指定邻车节点的车辆状态信息（全部原始传感器数据） |
| vam\_get\_peer\_relative\_pos | 获取指定邻车节点的相对于本车的位置（距离） |
| vam\_get\_peer\_relative\_dir | 获取指定邻车节点的相对于本车的行驶方向 |
| vam\_get\_peer\_relative\_speed | 获取指定邻车节点的相对于本车的相对行驶速度 |
| vam\_get\_peer\_absolute\_speed | 获取指定的邻车节点的绝对行驶速度 |
| …… |  |
| vam\_active\_alert | 激活告警消息发送 |
| vam\_cancel\_alert | 取消告警消息发送 |
| …… |  |

* 提供给无线网络通信部分用于获取车辆状态信息的API接口

表 4‑3 VANET中间件接口—获取车辆状态信息

|  |  |
| --- | --- |
| 接口函数 | 说明 |
|  |  |
|  |  |
|  |  |
|  |  |

* 提供给外设驱动部分用于更新本地车辆状态的API接口

表 4‑4 VANET中间件接口—更新本地车辆状态

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| lip\_update\_local\_gps | 更新本地车辆gps信息 |
| lip\_update\_local\_acc | 更新本地车辆加速度信息 |
|  |  |
|  |  |

* 提供给无线网络通信部分用于解析应用层协议的API接口

表 4‑5 VANET中间件接口—应用层协议接口

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| rcp\_recv | 接收RCP消息数据桢 |

### 接口3：网络传输层接口

无线网络通信部分提供给系统中其它部分的接口，为函数接口形式。此部分主要提供的是供VANET中间件部分使用的传输层协议帧发送接口。

表 4‑6 网络传输层接口

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| wnet\_get\_txbuf | 获取发送buffer（注：由wnet内部自动释放） |
| wnet\_release\_rxbuf | 释放接收buffer（注：由wnet内部自动获取） |
| wnet\_recv | 接收无线数据帧 |
| wnet\_send | 发送无线数据帧 |
| wnet\_send\_complete | 发送完成 |

### 接口4：无线MAC层驱动接口

无线网络MAC部分提供给无线网络通信及加解密驱动部分的数据收发接口，为函数接口形式。此部分主要用于网络数据帧在MAC层与协议处理层之间传递。为保证大量数据的高效交换，此接口内部实现采用共享数据缓冲区（队列）并辅以线程间的信号量同步机制的方式。

表 4‑7 无线网络MAC层接口

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| drv\_wifi\_send | 发送无线数据帧 |

### 接口5：数据加解密驱动接口

数据加解密部分提供给无线网络通信部分和网络传输层的接口，为函数接口形式。此部分主要用于网络数据帧的加解密操作。为保证大量数据的高效交换，此接口内部实现采用共享数据缓冲区（队列）并辅以线程间的信号量同步机制的方式。

表 4‑8 数据加解密接口

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| drv\_cryp\_encrypt | 加密数据桢 |
| drv\_cryp\_decrypt | 解密数据帧 |

### 接口6：其它外设驱动接口

其它外设设备驱动为上层提供操作外设的API接口，主要为函数接口形式。此部分包括了图像显示、语音播放、LED显示，flash读写操作接口。。

表 4‑9 外设驱动接口函数

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| drv\_display\_text | 显示文本 |
| drv\_display\_bmp | 显示图形 |
| drv\_display\_xxx | 显示。。。 |
| 。。。 | 显示。。。 |
| drv\_voc\_play | 开始播放 |
| drv\_voc\_stop | 停止播放 |
| drv\_led\_on | LED亮 |
| drv\_led\_off | LED灭 |
| drv\_led\_blink | LED闪烁 |
| drv\_fls\_read | 读flash |
| drv\_fls\_write | 写flash |
| drv\_fls\_erase | 擦除flash |

表 4‑10 外设驱动接口事件

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| DRV\_VOC\_PLAY\_COMPLETE | 语音播放完成事件 |
|  |  |

# 系统底层及公共部分

## 系统初始化模块设计

## Rt-thread操作系统移植

## 操作系统抽象层设计

## 中断管理模块设计

## 动态内存管理模块设计

## 系统调试支持模块设计

### 功能描述

在嵌入式平台上调试固件，或者系统在运行期间检测到错误并输出错误信息，或者想查看各个模块运行情况，往往需要通过输出软件调试信息。本模块即完成此功能，本质上是调用系统提供的printf或类似函数来实现。通过本模块，可以提供系统一级的调试打印输出控制接口，包括：打印级别控制、子模块分别控制。

为了兼顾调试功能和运行效率，本模块提供了两种调试信息输出方式，一种是直接打印到物理串口，通常用于软件开发过程的调试，在软件release以后，此方式的打印应当关闭；另一种是调试信息首先输出到系统的RAM空间中，在系统空闲时再自动输出到物理设备（包括串口或SD卡等存储设备），此类调试信息可在软件release后保留，用于设备维护。

### 接口设计

#### 打印级别

表 5‑1 打印级别定义

|  |  |
| --- | --- |
| 级别名称 | 说明 |
| OSAL\_DEBUG\_OFF | 关闭所有打印输出 |
| OSAL\_DEBUG\_ERROR | 输出错误级别信息 |
| OSAL\_DEBUG\_WARN | 输出警告及以上级别信息 |
| OSAL\_DEBUG\_INFO | 输出信息及以上级别信息 |
| OSAL\_DEBUG\_TRACE | 输出程序追踪及以上级别信息 |
| OSAL\_DEBUG\_LOUD | 输出及以上级别信息 |

#### 打印控制

表 5‑2 打印控制接口

|  |  |
| --- | --- |
| 接口名称 | 说明 |
| OSAL\_GLOBAL\_DEBUG | 全局调试输出开关控制 |
| OSAL\_DEBUG\_ERROR | 全局调试输出级别控制 |
| OSAL\_MODULE\_DEBUG | 模块级调试输出开关控制 |
| OSAL\_MODULE\_DEBUG\_LEVEL | 模块级调试输出级别控制 |
| OSAL\_DEBUG\_ENTRY\_DECLARE | 支持动态设置打印级别的模块声明 |
| OSAL\_DEBUG\_ENTRY\_DEFINE | 支持动态设置打印级别的模块定义 |
| osal\_dbg\_set\_level | 函数接口，动态模块打印级别设置 |
| debug | 功能同osal\_dbg\_set\_level，仅用于Rtt的FINSH控制台 |

* 动态设置模块打印级别功能说明：

与采用宏定义来控制预编译的方式不同，本系统提供了动态设置模块打印级别功能，可以通过函数接口osal\_dbg\_set\_level()或FINSH控制台的debug命令来实时修改指定模块的打印输出级别。要使用此功能，需按照如下流程：

1、在需要启用动态打印设置功能的模块（即.c文件）开始处添加下列代码：

|  |
| --- |
| #define OSAL\_MODULE\_DEBUG /\* 使能本模块的打印输出 \*/  #define OSAL\_MODULE\_DEBUG\_LEVEL OSAL\_DEBUG\_TRACE /\* 使能本模块的默认打印输出级别 \*/  #define MODULE\_NAME "module\_name" /\* 设置本模块名称（可打印字符） \*/  #include "cv\_osal\_dbg.h"  OSAL\_DEBUG\_ENTRY\_DEFINE(module\_name) /\* 设置本模块支持动态设置打印级别 \*/ |

其中：module\_name为模块名称

2、在系统源文件cv\_osal\_debug.c中指定处增加下列代码（仅粗体部分）：

|  |
| --- |
| /\*\*  \* [block begin] Debug level configuration  \* You can add your module here and DONOT modify any other code in the block.  \*/  const debug\_entry\_t debug\_entry\_table\_start = {NULL, NULL};  OSAL\_DEBUG\_ENTRY\_DECLARE(sysc)  **OSAL\_DEBUG\_ENTRY\_DECLARE(*module\_name*)**  // add your module here...  const debug\_entry\_t debug\_entry\_table\_end = {NULL, NULL}; |

#### 打印输出接口

表 5‑3 打印输出接口

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| osal\_printf | 函数接口，直接打印到物理串口 |
| osal\_log | 函数接口，打印内容到RAM空间 |
| OSAL\_DBGPRT | 通用打印（不含模块名称） |
| OSAL\_MODULE\_DBGPRT | 模块打印（打印信息中可带模块名称） |
| osal\_dbg\_dump\_data | 函数接口，打印buffer数据内容 |
| dump | 功能同osal\_dbg\_dump\_data，仅用于Rtt的FINSH控制台 |

#### 其它

# 接口及外设驱动部分

## UART驱动模块设计

## SPI驱动模块设计

## GPIO驱动模块设计

## GPS驱动模块设计

## GSensor驱动模块设计

## 无线MAC驱动模块设计

### 功能描述

本模块主要负责实现基于WiFi的底层无线通信功能，包括网卡驱动及MAC层的实现。本系统采用的WiFi基带处理器是应用与USB接口无线网卡的芯片RT3070，因此本模块主要基于Ralink的RT3070 Linux驱动移植而来，保留了原驱动中芯片硬件配置相关的代码，去除了802.11协议部分，同时将USB设备驱动部分重新基于ST官方的USB Host库来实现。

### 接口设计

#### 接口关系



图 6‑1 无线MAC驱动模块接口关系

#### 接口I1

* drv\_wifi\_send

表 6‑1 drv\_wifi\_send原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int drv\_wifi\_send(wnet\_txinfo\_t \*txinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_txinfo\_t \*txinfo | 发送信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 无线数据发送，通过USB WiFi无线网卡将数据发送出去。 | |

* drv\_wifi\_mac\_header\_len

表 6‑2 drv\_wifi\_mac\_header\_len原型定义

|  |  |
| --- | --- |
| 项 目 | 描 述 |
| 原型定义 | int drv\_wifi\_mac\_header\_len(void); |
| 输入项 | 无 |
| 输出项 | 无 |
| 返回值 | 返回WiFi MAC帧头保留长度 |
| 功能描述 | 获取WiFi MAC帧头长度。（用于在txbuffer中预留长度） |

#### 接口X1

由无线网络通信部分的收发帧处理模块提供给本模块调用的接口，详见相关模块设计。

* + wnet\_recv
  + wnet\_send\_complete

### 实现原理

#### 内部结构



图 6‑2 无线MAC驱动模块内部结构图

#### MAC帧格式定义

本系统的通信协议并不支持802.11，但是受芯片硬件限制，通过无线传输的帧格式必须符合802.11协议的规定。因此，从本系统通信特点出发，采用对特定类型的802.11标准帧格式的部分字段进行重定义或扩展来承载本系统中的实际数据，从而实现在不会正常的WiFi网络进行干扰的前提下传输特定数据的目的。本系统定义了几种不同的帧类型及格式，分别使用在不同的应用环境中。

* 帧类型1：



图 6‑3 MAC帧类型1格式定义

此方式使用的802.11协议的Beacon帧的标准格式定义，对部分字段（图中蓝色字段）进行了专门的规定，包括：

表 6‑3 MAC帧类型1定制字段定义

|  |  |  |
| --- | --- | --- |
| 字段 | 802.11协议中含义 | 本系统中含义 |
| FC | 表示帧控制域，包括帧类型定义等 | 采用beacon类型帧，取固定值：  0x80, 0x00 |
| BSSID | 表示无线网络ID，通常由无线路由器指定 | 做为VANET网络标识，取固定值：  0x00, 0x63, 0x73, 0x76, 0x32, 0x76 |
| TimeStamp/LLC | 表示网络时间戳 | LLC帧头 |
| Reserved | 表示网络参数 | 取固定值：  0x64, 0x00, 0x22, 0x04, 0x00, 0x00 |
| ID | 信息域ID，取固定值：0xDD （表示厂商自定义） | |
| Len | 厂商自定义信息长度 | DSRC消息帧长度，取值范围：0~255 |
| Payload | 厂商自定义信息内容 | DSRC消息帧内容 |

* 帧类型2：



图 6‑4 MAC帧类型2格式定义

此方式使用的802.11协议的data帧的标准格式定义，对部分字段（图中蓝色字段）进行了专门的规定，包括：

表 6‑4 MAC帧类型2定制字段定义

|  |  |  |
| --- | --- | --- |
| 字段 | 802.11协议中含义 | 本系统中含义 |
| FC | 表示帧控制域，包括帧类型定义等 | 采用data类型帧，取固定值：  0x08, 0x00 |
| BSSID | 表示无线网络ID，通常由无线路由器指定 | 做为VANET网络标识，取固定值：  0x00, 0x63, 0x73, 0x76, 0x32, 0x76 |
| LLC | LLC帧头 | LLC帧头 |
| Payload | 厂商自定义信息内容 | DSRC消息帧内容 |

* 帧类型3：



图 6‑5 MAC帧类型3格式定义

此方式使用的802.11协议的Beacon帧的标准格式定义，对部分字段（图中蓝色字段）进行了专门的规定，包括：

表 6‑5 MAC帧定制字段定义

|  |  |  |
| --- | --- | --- |
| 字段 | 802.11协议中含义 | 本系统中含义 |
| FC | 表示帧控制域，包括帧类型定义等 | 采用beacon类型帧，取固定值：  0x80, 0x00 |
| BSSID | 表示无线网络ID，通常由无线路由器指定 | 做为VANET网络标识，取固定值：  0x00, 0x63, 0x73, 0x76, 0x32, 0x76 |
| ID | 信息域ID，取固定值：0x00 （表示网络SSID） | |
| Len | SSID长度 | 位置信息长度，取固定值：0x18 |
| SSID/POSITION INFO | 网络SSID | 位置信息 |

* 帧类型4：



图 6‑6 MAC帧类型4格式定义

此方式使用的802.11协议的Beacon帧的标准格式定义，对部分字段（图中蓝色字段）进行了专门的规定，包括：

(注：RSU路测设备发送帧使用此格式)

表 6‑6 MAC帧类型2定制字段定义

|  |  |  |
| --- | --- | --- |
| 字段 | 802.11协议中含义 | 本系统中含义 |
| FC | 表示帧控制域，包括帧类型定义等 | 采用beacon类型帧，取固定值：  0x80, 0x00 |
| BSSID | 表示无线网络ID，通常由无线路由器指定 | 做为VANET网络标识，取固定值：  0x00, 0x63, 0x73, 0x76, 0x32, 0x76 |
| TimeStamp | 表示网络时间戳 | 区别与帧类型1 |
| Reserved | 表示网络参数 | 取固定值：  0x64, 0x00, 0x22, 0x04, 0x00, 0x00 |
| Payload | 厂商自定义信息内容 | DSRC消息帧内容 |

#### RT3070帧传输描述符定义

上节中定义的MAC帧内容，要实现通过RT3070网卡发射到空中，还需要首先通过USB接口传输到RT3070基带处理器，下图定义的分别是下行发送方向（由MCU发送到RT3070基带）和上行接收方向（从RT3070基带到MCU）的传输帧格式：



图 6‑7 RT3070传输帧格式

其中：

TXINFO：USB帧发送描述符，主要存放USB传输相关参数

TXWI：MAC帧发送描述符，主要存放无线物理层发射参数

RXDMALEN：USB接收帧长度

RXINFO：USB帧接收描述符，主要存放USB传输相关参数

RXWI：MAC帧接收描述符，主要存放无线物理层接收参数

#### 处理流程

由于此部分涉及较多的USB及WiFi硬件底层操作，流程复杂且主要是RT3070开源Linux驱动内部流程，本文中不再详述。

## Flash驱动模块设计

## 按键驱动模块设计

## 声音播放驱动模块设计

## LCD显示驱动模块设计

## 数据加密驱动模块设计

# 无线网络通信部分

## 收发帧处理模块设计

### 功能描述

本模块主要负责网络帧的数据收发管理及链路层（LLC）功能。在本系统中，传输的数据流分为上行和下行两个方向。在上行数据，即从网络接收数据方向，实现对从MAC驱动部分上传的数据帧进行管理，通过接收队列与信号量机制的方式，实现MAC驱动部分与网络通信部分上行数据接收的同步。在下行数据，即向网络发送数据方向，实现对从DSMP协议模块下发的数据帧进行管理，通过发送队列与信号量机制方式，实现与VANET中间层部分的数据的同步。

### 接口设计

#### 接口关系

本模块分别与VANET中间件部分的RCP协议模块、无线网络通信部分的DSMP协议模块、LLC协议模块及外设驱动部分的MAC驱动模块之间存在接口关系，如下图所示：



图 7‑1 收发帧处理模块接口关系

#### 接口O1

* wnet\_get\_txbuf

表 7‑1 wnet\_get\_txbuf原型定义

|  |  |
| --- | --- |
| 项 目 | 描 述 |
| 原型定义 | wnet\_txbuf\_t \*wnet\_get\_txbuf(void); |
| 输入项 | 无 |
| 输出项 | 无 |
| 返回值 | Buffer结构指针 |
| 功能描述 | 获取发送buffer |

* wnet\_release\_rxbuf

表 7‑2 wnet\_release\_rxbuf原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | void wnet\_release\_rxbuf(wnet\_rxbuf\_t \*rxbuf); | |
| 输入项 | wnet\_rxbuf\_t \*rxbuf | Buffer结构指针 |
| 输出项 | 无 | |
| 返回值 | 无 | |
| 功能描述 | 释放接收buffer | |

* WNET\_TXBUF\_DATA\_PTR(txbuf)

表 7‑3 WNET\_TXBUF\_DATA\_PTR原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | #define WNET\_TXBUF\_DATA\_PTR(txbuf) txbuf->data\_ptr | |
| 输入项 | wnet\_txbuf\_t \*txbuf | Buffer结构指针 |
| 输出项 | 无 | |
| 返回值 | Txbuffer中的数据指针 | |
| 功能描述 | 获取Txbuffer中的数据指针 | |

* WNET\_TXBUF\_INFO\_PTR(txbuf)

表 7‑4 WNET\_TXBUF\_INFO\_PTR原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | #define WNET\_TXBUF\_INFO\_PTR(txbuf) &txbuf->info | |
| 输入项 | wnet\_rxbuf\_t \*rxbuf | Buffer结构指针 |
| 输出项 | 无 | |
| 返回值 | Txbuffer中的信息结构指针 | |
| 功能描述 | 获取Txbuffer中的信息结构指针 | |

* WNET\_RXBUF\_PTR(info)

表 7‑5 WNET\_RXBUF\_PTR原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | #define WNET\_RXBUF\_PTR(info) (struct \_wnet\_rxbuf \*)((uint32\_t)info - sizeof(list\_head\_t)) | |
| 输入项 | wnet\_rxinfo\_t \*info | 接收信息结构指针 |
| 输出项 | 无 | |
| 返回值 | Txbuffer结构体指针 | |
| 功能描述 | 通过接收信息指针获取Rxbuffer的结构体指针 | |

* wnet\_send

表 7‑6 wnet\_send原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int wnet\_send(wnet\_txinfo\_t \*txinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_txinfo\_t \*txinfo | 接收帧信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 网络数据发送，将封装网络层数据格式并加入发送队列。 | |

#### 接口O2

* wnet\_recv

表 7‑7 wnet\_recv原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int wnet\_recv(wnet\_rxinfo\_t \*rxinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_rxinfo\_t \*rxinfo | 接收帧信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 网络数据接收，将接收数据复制并加入到接收队列。 | |

* wnet\_send\_complete

表 7‑8 wnet\_send\_complete原型定义

|  |  |
| --- | --- |
| 项 目 | 描 述 |
| 原型定义 | void wnet\_send\_complete (void); |
| 输入项 | 无 |
| 输出项 | 无 |
| 返回值 | 无 |
| 功能描述 | 无线数据发送完成，将发送buffer移出发送队列。 |

#### 接口I1

* fp\_send

表 7‑9 fp\_send原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int fp\_send(wnet\_envar\_t \*p\_wnet , wnet\_txinfo\_t \*txinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_envar\_t \*p\_wnet | wnet环境变量指针 |
| wnet\_txinfo\_t \*txinfo | 发送信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 无线数据发送，将发送buffer加入到发送等待队列。 | |

#### 接口X1

由MAC驱动模块提供给本模块调用的接口，详见相关模块设计。

* + drv\_wifi\_send
  + drv\_wifi\_mac\_header\_len

#### 接口X2

由LLC模块提供给本模块调用的接口，详见相关模块设计。

* + llc\_recv

#### 接口X3

由DSMP模块提供给本模块调用的接口，详见相关模块设计。

* + dsmp\_send

### 实现原理

#### 内部结构



图 7‑2 收发帧处理模块内部结构图

#### 数据结构



图 7‑3 收发帧处理模块数据结构

本模块数据结构做为无线网络通信部分数据结构“wnet\_envat\_t”的一部分，在整个系统中的位置及相互关系如上图所示。主要结构体定义如下：

* wnet\_envar\_t：

|  |
| --- |
| typedef struct \_wnet\_envar {  void \*global;  /\* working\_param \*/  wnet\_config\_t working\_param;  **/\*\***  **\* Frame process**  **\*/**  **osal\_task\_t \*task\_wnet\_tx;**  **osal\_task\_t \*task\_wnet\_rx;**  **osal\_sem\_t \*sem\_wnet\_tx;**  **osal\_sem\_t \*sem\_wnet\_rx;**  **list\_head\_t txbuf\_free\_list;**  **list\_head\_t txbuf\_waiting\_list;**  **list\_head\_t rxbuf\_free\_list;**  **list\_head\_t rxbuf\_waiting\_list;**  **wnet\_txbuf\_t txbuf[TXBUF\_NUM];**  **wnet\_rxbuf\_t rxbuf[RXBUF\_NUM];**  **//to be added …**  } wnet\_envar\_t; |

其中：

* task\_wnet\_tx：网络发送任务
* task\_wnet\_rx：网络接收任务
* task\_wnet\_tx：网络发送任务同步信号量
* task\_wnet\_rx：网络接收任务同步信号量
* txbuf\_free\_list：发送buffer池
* txbuf\_waiting\_list：发送等待队列
* rxbuf\_free\_list：接收buffer池
* rxbuf\_waiting\_list：接收等待队列
* txbuf[]:发送buffer空间
* rxbuf[]：接收buffer空间
* wnet\_txbuf\_t：

|  |
| --- |
| typedef struct \_wnet\_txbuf {  list\_head\_t list;  wnet\_txinfo\_t info;  uint32\_t flag; /\* buffer's status \*/  uint8\_t \*data\_ptr;  int32\_t data\_len;  uint8\_t buffer[TXBUF\_LENGTH];  }wnet\_txbuf\_t; |

* wnet\_rxbuf\_t：

|  |
| --- |
| typedef struct \_wnet\_rxbuf {  list\_head\_t list;  wnet\_rxinfo\_t info;  uint8\_t \*data\_ptr;  int32\_t data\_len;  uint8\_t buffer[RXBUF\_LENGTH];  }wnet\_txbuf\_t; |

* wnet\_txinfo\_t：

|  |
| --- |
| typedef struct \_wnet\_txinfo {  wnet\_addr\_t dest;  uint8\_t protocol; /\* 0 - DSMP, 1 - UDP \*/  uint8\_t encryption; /\* 0 - none, 1 - encpypted \*/  uint8\_t prority; /\* 0 - NORMAL, 1 - EMERGENCY \*/  uint32\_t timestamp; /\* time of generated message \*/  void \*extension; /\* reserved for future \*/  }wnet\_txinfo\_t; |

* wnet\_rxinfo\_t：

|  |
| --- |
| typedef struct \_wnet\_rxinfo {  wnet\_addr\_t src;  uint8\_t protocol; /\* 0 - DSMP, 1 - UDP \*/  uint8\_t encryption; /\* 0 - none, 1 - encpypted \*/  uint8\_t rssi;  uint32\_t timestamp; /\* time of received message \*/  void \*extension; /\* reserved for future \*/  }wnet\_rxinfo\_t; |

#### 处理流程

* 数据接收（上行）处理流程：



图 7‑4 上行数据流处理流程

1. 当系统MAC驱动部分从网络中接收到数据时触发数据接收处理流程；
2. 首先调用wnet\_send接口来接收数据，此接口内部将从空闲buffer池中获取接收buffer，并将接收数据帧内容复制到buffer中；
3. 将数据桢加入到接收队列；
4. 向网络接收处理线程发送信号量；
5. 网络接收处理线程接收到信号量后唤醒，并从接收队列提取数据桢（注：从接收队列中删除）；
6. 调用传输层处理部分解析传输层协议；
7. 调用应用协议处理接口（向应用处理线程发出消息）；
8. 在应用层部分的相应线程中对数据进行后续处理；
9. 处理完成后释放数据buffer；

* 数据发送（下行）处理流程：



图 7‑5 下行数据流处理流程

1. 由系统中其它部分（如VANET应用中间件）触发数据发送流程；
2. 首先调用wnet\_get\_tx\_buf接口从空闲buffer池中获取发送buffer；
3. 在buffer空间中进行组帧操作（须按照由上至下依次进行不同层的协议封装）；
4. 调用wnet\_send接口来发送数据，此接口内部将把数据帧加入发送队列，同时向网络发送线程发出信号量；
5. 网络发送线程接收到信号量后唤醒，并从发送队列读取数据帧（注：此时不从队列中删除）；
6. 调用底层发送接口，此接口内部将数据帧内容进行复制，并进行后续底层发送操作；
7. 底层发送完成后，释放数据buffer，同时从发送等待队列中删除；

## LLC协议模块设计

### 功能描述

本模块主要负责网络传输中的链路层协议。

### 接口设计

#### 接口关系

本模块分别与无线网络通信部分的DSMP协议模块和收发帧处理模块之间存在接口关系，如下图所示：



图 7‑6 LLC协议模块接口关系

#### 接口I1

* llc\_send

表 7‑10 llc\_send原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int llc\_send(wnet\_envar\_t \*p\_wnet , wnet\_txinfo\_t \*txinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_envar\_t \*p\_wnet | wnet环境变量指针 |
| wnet\_txinfo\_t \*txinfo | 发送信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 无线数据发送，将发送buffer加入到发送等待队列。 | |

#### 接口I2

* llc\_recv

表 7‑11 llc\_recv原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int llc\_recv(wnet\_envar\_t \*p\_wnet , wnet\_rxinfo\_t \*rxinfo, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | wnet\_envar\_t \*p\_wnet | wnet环境变量指针 |
| wnet\_txinfo\_t \*txinfo | 接收信息结构指针 |
| uint8\_t \*pdata | 数据首地址 |
| uint32\_t length | 数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | 无线数据接收，将接收数据复制并加入到接收队列。 | |

#### 接口X1

由DSMP协议模块提供给本模块调用的接口，详见相关模块设计。

* + dsmp\_recv

#### 接口X2

由收发帧处理模块提供给本模块调用的接口，详见相关模块设计。

* + fp\_send

### 实现原理

#### LLC帧格式

本系统中的LLC帧格式定义如下：



图 7‑7 LLC帧格式

本系统中的LLC层在传统的IEEE 802.2格式在EtherType字段的可选值中增加表示WSA消息的类型定义，如下：

表 7‑12 LLC帧EtherType值

|  |  |
| --- | --- |
| EtherType值 | 描 述 |
| 0x0800 | Internet Protocol version 4 (IPv4) |
| 0x86DD | Internet Protocol version 4 (IPv6) |
| 0x88DC | WSMP Protocol Version 1 |
| **0x88DD** | **WSMP Protocol Version 1（with security header）** |
| **0x88DD** | **WSA Protocol Version 1** |
| **0x88DD** | **WSA Protocol Version 1（with security header）** |
| 0x88E2 | DSMP Protocol Version 1 |

## DSMP协议模块设计

DSMP与DSA协议基于1609.3协议涉及，参考正在制定中的专用短程通信国标，简化了WME（管理实体）模块。

### 功能描述

主要包括DSM帧的解帧和组帧。本部分向下与数据收发驱动接口、加密模块接口，实现rcp消息的dsmp封装，并传递信道、速率、安全加密等发送参数给下层，然后调用LLC层接口发送消息（或者调用加密接口，由加密模块加密后发送）。向上解帧传递应用短消息给rcp模块。

DSMP数据帧封装在LLC帧中，EtherType=0x88E2，见表 7.2.3.1定义。

#### DSMP数据帧格式



图 7‑8 DSMP封装的帧格式

其中：

* + 版本Version：区分不同的版本号，当前值为1。
  + 应用标识AID：应用服务商的应用标识，区分不同的应用，类似于TCP、UDP端口号。

目前我们的应用：20

通用AID：

表 7‑13 DSMP帧头AID定义

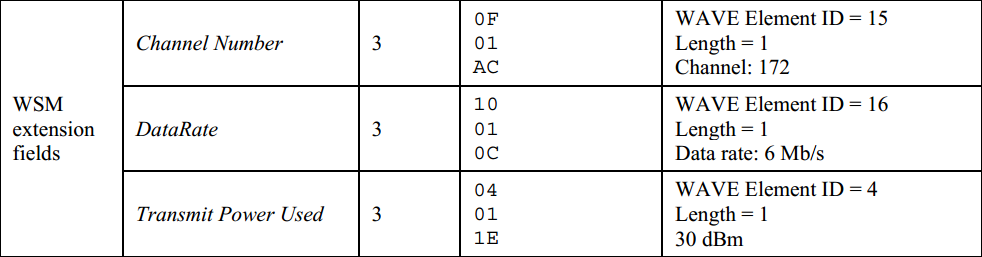
| **AID** | **Service** |
| --- | --- |
| 0 | system |
| 1 | automatic-fee-collection |
| 2 | freight-fleet-management |
| 3 | public-transport |
| 4 | traffic-traveler-information |
| 5 | traffic-control |
| 6 | parking-management |
| 7 | geographic-road-database |
| 8 | medium-range-preinformation |
| 9 | man-machine-interface |
| 10 | intersystem-interface |
| 11 | automatic-vehicle-identification |
| 12 | emergency-warning |
| 13 | private |
| 14 | multi-purpose-payment |
| 15 | dsrc-resource-manager |
| 16 | after-theft-systems |
| 17 | cruise-assist- highway-system |
| 18 | multi-purpose-information system |
| 19 | public-safety |
| **20** | **vehicle-safety** |
| 21 | general-purpose-internet-access |
| 22 | onboard diagnostics |
| 23 | security manager |
| 24 | signed *WSA* |

* + 扩展域Extension：可选，可用于指定信道编号、传输功率等级和数据传输速率等信息。扩展域的数据结构是如下的TLV结构，Type即ElementID，见下图：



图 ‑9 DSMP/DSA扩展域格式定义

例如:



* + 数据标识Element ID：通过DSMP协议发送的数据类型编号，用于表示数据信息的不同作用，0x80表示短消息协议。

表7.3-2 ElementID定义

|  |  |  |  |
| --- | --- | --- | --- |
| **Element ID** | **编号** | **应用域** | **描述** |
| WSA Elements | | | |
|  | 0 | Reserved |  |
| Service Info | 1 | 应用公告服务信息 | 多字节。包含服务提供者编号，服务优先级，信道检索号以及扩展域 |
| Channel Info | 2 | 应用信道信息 | 基本信息为6字节。包含操作级别、信道号、调节度，数据率，传输功率等级以及扩展域 |
| WRA | 3 | 应用路由信息 | 基本信息为58 字节。WAVE路由：包含路由时限，IpPrefix，IpPrefix长度，默认网关，网关MAC地址，Primary DNS以及扩展域 |
| Extension Elements | | | |
| Transmit power used | 4 | 应用公告帧头 | 8bit，表示传输DSMP数据帧的发送功率。单位为dBm。 |
| 2D Location | 5 | 应用公告帧头 | 8字节。2D位置。4个字节表示纬度，4个字节表示经度。精确至0.1微度。 |
| 3DLocationAndConfidence | 6 | 应用公告帧头 | 15字节。3D位置，4个字节表示纬度，4个字节表示经度。经纬度精确至0.1微度。2个字节表示海拔，精确至0.1米。4bit表示位置置信度。4bit表示海拔置信度。4字节表示位置精确度。 |
| Advertiser identifier | 7 | 应用公告帧头 | 1-32字节。字符串，应用提供者的识别号，应用服务者标识字符串 |
| Provider Service Context | 8 | 应用公告服务信息 | 1-31字节。字符串，提供与高层服务关联的补充信息。 |
| IPv6 address | 9 | 应用公告服务信息 | 128bit。IPv6地址 |
| Service port | 10 | 应用公告服务信息 | 16 bit。为应用服务提供的高层实体的端口号，当服务采用IP地址的时候48 bit。 |
| Provider MAC address | 11 | 应用公告服务信息 | 48 bit。提供服务的主机的MAC地址。 |
| EDCA Parameter Set | 12 | 应用公告信道信息 | 1字节。被推荐的最小的接收WSA信号值，单位为dBm，范围为0至-110. |
| Secondary DNS | 13 | 应用公告路由信息 | 128bit IPv6地址，表示备选DNS服务器地址 |
| Gateway MAC Address | 14 | 应用公告路由信息 | 48bit网卡物理地址，表示默认的网络物理地址 |
| Channel Number | 15 | DSMP帧头 | 1字节，表示信道号 |
| DataRate | 16 | DSMP帧头 | 1字节，表示数据发射速率，取值0x02~0x7F，单位500kbit/s |
| Repeat Rate | 17 | 应用公告帧头 | 1字节，表示含义为在5秒内服务公告重复发送的次数。此参数可以用来评估链路质量。 |
| Country String | 18 | 应用公告帧头 | 3字节，表示设备所在的国家和区域。 |
| RCPI Threshold | 19 | 应用公告服务信息 | 1字节，接收信道功率指示（RCPI）阈值，表示推荐可用的服务公告消息（WSA）的最小信号值，低于此阈值以下时可以忽略。单位dBm，取值范围0~-110。具体定义参见IEEE Std 802.11k-2008 |
| WSA Count Threshold | 20 | 应用公告服务信息 | 1字节，表示接收服务公告消息（WSA）最小个数的阈值，当接收到的消息数目低于此值时，接收端可以忽略此服务 |
| Channel Access | 21 | 应用公告信道信息 | 信道访问控制，含义如下：  0 - 当前服务可在SCH和CCH信道同时访问  1-当前服务仅在SCH信道可访问 |
| WSA Count Threshold Interval | 22 | 应用公告服务信息 | 1字节，表示使用服务公告消息（WSA）最小个数阈值时对应的时间间隔，单位100ms，有效取值1~255.如果不指定时，默认取值1s |
|  | 23 to 127 | 保留 |  |
| DedicatedShort Message | 128 | DSMP帧头 | 表示DSMP消息 |
| DSMP-S | 129 | DSMP header | 表示DSMP-S消息（安全补充协议，用于交互了本端的信道切换信息） |
|  | 130 to 255 | 保留 |  |

* + 数据长度Length：表示应用层数据实体的字节长度。
  + 数据Data：是承载的应用层数据实体，如rcp消息

#### DSM发送接收流程



图 7‑10 DSMP的数据流程

### 接口设计

设计原则：基于IEEE1609.3标准规定的原语设计上下层API接口。

#### 接口关系：



图 7‑11 DSMP接口关系图

Dsmp提供接口：

* D1： dsmp\_send()
* D2： dsmp\_recv()

调用接口：

* R2：rcp\_recv()，收到的数据上送给DSRC消息协议(rcp)层
* L1：如果DSM不需要加密，调用L1接口LLC\_send()，发送数据到LLC层，参见7.1.2.2。
* E1：如果DSM需要加密，则调用加密模块接口E1，将数据发给Encry模块加密

#### dsmp\_send

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int dsmp\_send(wnet\_envar\_t \* pwnet, wnet\_txinfo\_t \*txinfo, uint8\_t \*pdata, uint32\_t length)； | |
| 输入项 | pwnet | 网络层p\_wnet\_envar环境变量指针 |
| txinfo | 发送参数 |
| pdata | 应用层数据指针 |
| length | 应用层数据长度 |
| 输出项 | 无 |  |
| 返回值 | 1. 成功 2. 失败 |  |
| 功能描述 | 应用层调用，DSM组帧并发送 | |

#### dsmp\_recv

|  |  |  |
| --- | --- | --- |
| 条目 | 内容 | |
| 原型定义 | int dsmp\_recv (wnet\_envar\_t \* pwnet, wnet\_rxinfo\_t \*rxinfo, uint8\_t \*pdata, uint32\_t length )； | |
| 输入项 | wnet\_rxinfo\_t \*rxinfo | 接收帧信息结构指针 |
| pwnet | 网络层p\_wnet\_envar环境变量指针 |
| uint8\_t \*pdata | 接收数据首地址 |
| uint32\_t length | 接收数据长度 |
| 输出项 | 无 |  |
| 返回值 | 0 | 成功 |
| 1 | 失败 |
| 功能描述 | LLC调用，接收DSM消息后解帧并分发DSM中的应用数据到上层应用 | |

### 加密的DSM结构

DSM加密帧的结构参照1609.3规定的WSA加密帧的机构进行定义。



图 7‑12 1609.2帧格式

如系统数据流图所示，对于上层调用dsmp\_send，需要加密时，组帧后调用加密模块接口后直接返回，

drv\_enc(wnet\_rxinfo\_t \*rxinfo, uint8\_t \*pdata, uint32\_t length, uint32\_t enc\_offset)，

加密模块完成加密后，加密模块调用LLC层发送接口。需要注意的是，enc\_offset参数，表示需要对pdata数据偏移enc\_offset后的数据进行加密，即只加密DSM，加密模块加密的密文直接覆盖替换DSM原明文。

SecurityHeader：针对不同算法而定。

### 实现原理

主要完成上层DSRC消息的DSMP封装与解帧，给其他模块提供接口使用。不再详述。

## DSA协议模块设计 \*

### 功能描述

IEEE 1609.3 定义了两种服务的角色：提供者（Provider）及使用者（User）。提供者负责发送控制用的 WAVE Service Advertisement（WSA）封包，当一个设备想要提供服务时就会广播 WSA。而使用者则是负责接收 WSA，WSA包含了许多建立服务需要的重要资讯：信道、功率、速率、路由、网关、DNS等信息，类似于建立连接。

#### WSA的数据帧的承载：

IEEE 802.11p新增控制帧——时间公告 (timing advertisement, TA) 帧包含一个或多个厂商信息单元（VSIE：vendor-specific information elements），1609.3规定WSA由TA帧的VSIE携带进行透明传输；如下图所示：



图 7‑13 VSIE帧承载WSA的定义

然而我们目前MAC层只采用beacon帧进行传输，所以LLC层无法区分DSA，故而在LLC帧的Ethertype扩展一种类型0x88E1来承载DSA帧，见表7.3-1。

我们定义DSA如下：



图 7‑14 1609.2承载DSA的帧定义

其中：type值为unsecured (0), signed(1), encrypted (2)，表示不可靠，签名，加密。

#### DSA帧格式：



图 7‑15 DSA帧格式定义

#### WSA流程1：Service setup & WSM delivery



图 7‑16 Service setup & WSM delivery

以上是标准规定的流程，我们目前直接使用固定信道发生WSM，不需要WSA公告来建立连接。

#### WSA流程2：Service setup & other data delivery



图 7‑17 DSA服务通道建立及数据交换流程

### 接口设计

设计原则：基于IEEE1609.3规范原语设计上下层API接口。

#### 接口关系



图 7‑18 DSA接口关系图

DSA包含的DME子模块负责维护管理ME实体（信道，服务等），与上层接口，实现服务请求、应答、通知等功能。

DSA内部完成服务公告的DSA组帧与解帧，提供向下的接口完成DSA帧的发送以及接收。

* **提供接口：TBD**

dsa\_service\_request()、dsa\_service\_cfm()、dsa\_service\_inform()、dsa\_recv()

* **调用接口：**

I1：LLC发送接口、服务连接的底层控制接口（信道、IP、路由等）

E1：加密接口

#### dsa\_service\_request()

TBD

#### dsa\_service\_cfm

TBD

#### dsa\_service\_inform

TBD

#### dsa\_recv

TBD

### 实现原理：

暂不实现。

# VANET中间件部分

## 应用编程接口（API）模块设计

*（这是之前原型机总体设计方案里面的内容，可以在此基础上补充和修改）*

即上图中接口2，是由VANET中间件模块提供给上层应用的编程接口，采用下表所列的接口API的形式：

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| vam\_init | VANET中间件初始化 |
| vam\_deinit | VANET中间件反初始化（卸载） |
| vam\_start | VANET中间件启动服务 |
| vam\_stop | VANET中间件停止服务 |
| vam\_get\_config | 获取工作参数 |
| vam\_set\_config | 设置工作参数 |
| vam\_set\_event\_handler | 设置事件回调函数接口 |
| …… |  |
| vam\_get\_local\_status | 获取本车的状态信息 |
| vam\_set\_local\_status | 更新本车的状态信息（仅用于当VANET不支持内部解析本地GPS、加速度传感器等功能时） |
| vam\_get\_peerlist | 获取邻车节点列表 |
| vam\_get\_peer\_status | 获取指定邻车节点的车辆状态信息（全部原始传感器数据） |
| vam\_get\_peer\_relative\_pos | 获取指定邻车节点的相对于本车的位置（距离） |
| vam\_get\_peer\_relative\_dir | 获取指定邻车节点的相对于本车的行驶方向 |
| vam\_get\_peer\_relative\_speed | 获取指定邻车节点的相对于本车的相对行驶速度 |
| vam\_get\_peer\_absolute\_speed | 获取指定的邻车节点的绝对行驶速度 |
| …… |  |
| vam\_active\_alert | 激活告警消息发送 |
| vam\_cancel\_alert | 取消告警消息发送 |
| …… |  |

接口函数详细定义如下：

|  |  |
| --- | --- |
| vam\_init | |
| 函数功能 | VANET中间件初始化 |
| 函数原型 | void vam\_init(void) |
| 输入参数 | 无 |
| 返回结果 | 无 |

|  |  |
| --- | --- |
| vam\_deinit | |
| 函数功能 | VANET中间件反初始化（卸载） |
| 函数原型 | void vam\_deinit(void) |
| 输入参数 | 无 |
| 返回结果 | 无 |

|  |  |
| --- | --- |
| vam\_start | |
| 函数功能 | VANET中间件启动服务功能 |
| 函数原型 | int32\_t vam\_start(void) |
| 输入参数 | 无 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_stop | |
| 函数功能 | VANET中间件停止服务功能 |
| 函数原型 | int32\_t vam\_stop(void) |
| 输入参数 | 无 |
| 返回结果 | 无 |

|  |  |
| --- | --- |
| vam\_get\_config | |
| 函数功能 | 获取工作参数 |
| 函数原型 | int32\_t vam\_get\_config(vam\_config\_t \*config) |
| 输入参数 | vam\_config\_t \*config  工作参数结构指针 |
| 返回结果 | 0：成功  <0：错误代码 |
| 数据结构 | *typedef struct \_vam\_config {*  *uint8\_t bsm\_hops; //BSM消息最大跳数；*  *uint8\_t bsm\_broadcast\_type; //BSM消息广播方式，0 – 随速自适应， 1- 固定周期，由bsm\_broadcast\_peroid指定；*  *uint16\_t bsm\_broadcast\_peroid; //BSM消息广播周期*  *…*  *uint8\_t evam\_hops; //EVAM消息最大跳数；*  *uint8\_t evam\_broadcast\_type; //EVAM消息广播方式，0 – 随速自适应， 1- 固定周期，由evam\_broadcast\_peroid指定；*  *uint16\_t evam\_broadcast\_peroid; //EVAM消息广播周期*  *…*  *}vam\_config\_t;* |

|  |  |
| --- | --- |
| vam\_set\_config | |
| 函数功能 | 设置工作参数 |
| 函数原型 | int32\_t vam\_set\_config(vam\_config\_t \*config) |
| 输入参数 | vam\_config\_t \*config  工作参数结构指针 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_set\_event\_handler | |
| 函数功能 | 设置工作参数 |
| 函数原型 | int32\_t vam\_set\_event\_handler(uint32\_t evt, vam\_evt\_handler \*callback) |
| 输入参数 | uint32\_t evt  事件ID，支持的事件包括：  *VAM\_EVT\_LOCAL\_UPDATE*  *VAM\_EVT\_PEER\_UPDATE*  *VAM\_EVT\_PEER\_ALARM*  *VAM\_EVT\_LOCAL\_UPDATE*  vam\_evt\_handler \*callback  事件处理回调函数指针 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_get\_local\_status | |
| 函数功能 | 获取本车的车辆状态信息，包括所有的位置、速度、方向等原始数据。 |
| 函数原型 | int32\_t vam\_get\_local\_status(vam\_stastatus\_t \*local) |
| 输入参数 | vam\_stastatus\_t \*local  本车状态结构体指针 |
| 返回结果 | 0：成功  <0：错误代码 |
| 数据结构 | *typedef struct \_vam\_stastatus{*  *uint8\_t pid[4]; //temporary ID*  *uint16\_t timestamp;*  *vam\_position\_t pos ;*  *vam\_dir\_t dir;*  *vam\_speed\_t speed;*  *vam\_acce\_t acce;*  *…*  *} vam\_stastatus\_t;* |

|  |  |
| --- | --- |
| vam\_set\_local\_status | |
| 函数功能 | 更新本车的车辆状态信息，本接口仅用在当VANET中间件不提供GPS、GSensor等传感器功能时。 |
| 函数原型 | int32\_t vam\_set\_local\_status(vam\_stastatus\_t \*local) |
| 输入参数 | vam\_stastatus\_t \*local  本车状态结构体指针 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_get\_peerlist | |
| 函数功能 | 获取周围所有可见邻车节点列表 |
| 函数原型 | int32\_t vam\_get\_peerlist(vam\_stastatus\_t \*\*local,  uint32\_t maxitem,  uint32\_t \*actual) |
| 输入参数 | vam\_stastatus\_t \*local  车辆状态结构数组指针  uint32\_t maxitem  最大节点个数  uint32\_t actual  实际返回节点个数 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_get\_peer\_status | |
| 函数功能 | 获取指定邻车节点的车辆状态信息，包括所有原始的传感器数据 |
| 函数原型 | int32\_t vam\_get\_peer\_status(uint8\_t pid[4], vam\_stastatus\_t \*local) |
| 输入参数 | uint8\_t pid[4]  邻车的temporaryID  vam\_stastatus\_t \*local  本车状态结构体指针 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_get\_peer\_relative\_pos | |
| 函数功能 | 获取指定的邻车节点的相对于本车的位置（距离） |
| 函数原型 | int32\_t vam\_get\_peer\_relative\_pos(uint8\_t pid[4]) |
| 输入参数 | uint8\_t pid[4]  邻车的temporaryID |
| 返回结果 | 绝对值表示两车距离，单位m  >0 ：位于本车行驶方向前方  < 0 ：位于本车行驶方向后方 |

|  |  |
| --- | --- |
| vam\_get\_peer\_relative\_dir | |
| 函数功能 | 获取指定的邻车节点的相对于本车的行驶方向 |
| 函数原型 | int32\_t vam\_get\_peer\_relative\_dir(uint8\_t pid[4]) |
| 输入参数 | uint8\_t pid[4]  邻车的temporaryID |
| 返回结果 | >0 ：与本车同向  < 0 ：与本车反向 |

|  |  |
| --- | --- |
| vam\_get\_peer\_relative\_speed | |
| 函数功能 | 获取指定的邻车节点的相对于本车的行驶速度 |
| 函数原型 | int32\_t vam\_get\_peer\_relative\_speed(uint8\_t pid[4]) |
| 输入参数 | uint8\_t pid[4]  邻车的temporaryID |
| 返回结果 | 绝对值表示两车相对车速，单位m/s  >0 ：大于本车时速  < 0 ：小于本车时速 |

|  |  |
| --- | --- |
| vam\_get\_peer\_absolute\_speed | |
| 函数功能 | 获取指定的邻车节点的绝对行驶速度 |
| 函数原型 | int32\_t vam\_get\_peer\_absolute\_speed(uint8\_t pid[4]) |
| 输入参数 | uint8\_t pid[4]  邻车的temporaryID |
| 返回结果 | 绝对车速，单位m/s |

|  |  |
| --- | --- |
| vam\_active\_alert | |
| 函数功能 | 激活告警消息发送 |
| 函数原型 | int32\_t vam\_active \_alert(uint32\_t alerttype) |
| 输入参数 | uint32\_t alerttype  报警类型  0 – 车辆故障  1 – 紧急制动 |
| 返回结果 | 0：成功  <0：错误代码 |

|  |  |
| --- | --- |
| vam\_cancel\_alert | |
| 函数功能 | 取消告警消息发送 |
| 函数原型 | int32\_t vam\_cancel\_alert(uint32\_t alerttype) |
| 输入参数 | uint32\_t alerttype  报警类型  0 – 车辆故障  1 – 紧急制动 |
| 返回结果 | 0：成功  <0：错误代码 |

* 接口3：数据通信接口

即上图中接口3，是由VANET中间件模块和无线通信模块之间的数据通信接口，采用接口API的形式，定义如下：

|  |  |
| --- | --- |
| 接口函数 | 说明 |
| vam\_rcp\_recv | RCP协议模块数据接收 |
|  |  |

接口函数详细定义如下：

|  |  |
| --- | --- |
| vam\_rcp\_recv | |
| 函数功能 | 由VANET中间件中的车辆远程通信协议子模块提供的数据接收接口，供无线通信模块调用，从而向上层传递应用数据帧 |
| 函数原型 | int32\_t vam\_rcp\_recv(rcp\_rxinfo\_t \*rxinfo,  uint8\_t \*databuf,  uint32\_t datalen) |
| 输入参数 | rcp\_rxinfo\_t \*rxinfo  应用数据帧信息  uint8\_t \*databuf  应用数据帧buffer首地址  uint32\_t datalen  应用数据帧长度 |
| 返回结果 | 0：成功  <0：错误代码 |
| 数据结构 | *typedef struct \_rcp\_rxinfo {*  *uint8\_t src[N]; //源地址*  *uint8\_t hops; //实际传输转发跳数*  *uint8\_t prority; //发送优先级*  *uint8\_t channel; //发送信道*  *uint8\_t datarate; //发送速率*  *uint8\_t rssi; //接收信号强度*  *}rcp\_rxinfo\_t;* |

## 车辆状态管理（VSM）模块设计

## 车辆本地信息处理（LIP）模块设计

## DSRC消息协议（RCP）模块设计

### 协议功能分析

DSRC消息协议（Dedicated Short Range Communications Message Protocol），作为DSRC协议栈应用层的通信消息格式，实现车辆安全应用的通信。本模块基于SAE J2735标准设计，但是为了简化原型机设计，在满足基本的车辆应用通信的基础上，仅支持部分消息类型，并且对消息结构的可选项进行了适当裁减。

J2735消息集定义了以下15种消息，如下：

表8.4-1 DSRC消息简介

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Message | Typical Use | Notes |
| 0 | Reserved | N/A |  |
| 1 | MSG\_A\_la\_Carte(ACM) | V2X |  |
| 2 | MSG\_BasicSafetyMessage (BSM) | V2V | 基本安全消息：目前应用场景都基于此消息，消息携带位置信息、车辆状态及车辆告警信息； |
| 3 | MSG\_CommonSafety(CSR) | V2V | 车辆单播发送到目标车辆请求相关BSM扩展参数，目标车辆收到后在广播的BSM中带上请求的参数发送 |
| 4 | MSG\_EmergencyVehicleAlert(EVA) | V2X | 紧急车辆警告消息：特殊车辆向外广播自身紧急告警及其位置和状态信息便于其他车辆进行规避等 |
| 5 | MSG\_IntersectionCollisionAvoidance(ICA) | V2X |  |
| 6 | MSG\_MapData(MAP) | I2V | 地图数据(GID Layer)：需要电子地图支持 |
| 7 | MSG\_NMEA\_Corrections(NMEA) | I2V |  |
| 8 | MSG\_ProbeDataManagement (PDM) | I2V | 探测数据管理消息：路侧设备通过此消息控制车辆发送PVD消息中snapshot生成的频度及宽度 |
| 9 | MSG\_ProbeVehicleData(PVD) | V2I | 探测车辆数据信息：车辆发送自身状态、位置等消息给路侧设施 |
| 10 | MSG\_RoadSideAlert(RSA) | I2V | 路侧警告消息：路测设备发送警告消息给附近车辆 |
| 11 | MSG\_RTCM\_Corrections(RTCM) | I2V |  |
| 12 | MSG\_SignalPhaseAndTiming(SPAT) | I2V | 需要地图支持， |
| 13 | MSG\_SignalRequestMessage(SRM) | V2I |  |
| 14 | MSG\_SignalStatusMessage(SSM) | I2V | 信号灯状态信息供车辆使用 |
| 15 | MSG\_TravelerInformation Message(TIM) | I2V |  |

目前，本系统暂仅支持BSM、RSA、EVA、PVD、PDM及MAP消息，这些消息体均在RCP模块中给出定义，但具体业务逻辑及应用暂只实现与项目需求相关的BSM、RSA及EVA消息，而PVD/PDM及MAP消息的后续实现。

### 消息格式

本协议的消息直接采用字节流进行传输，下面对其语法格式和处理流程进行描述。

其中，各消息中共有数据项定义如下：

* MsgID：bit4-0表示消息ID（按照J2735的规定） ， bit7-5 表示此消息的跳数(hops)，跳数给路由转发模块用来判断收到此消息后是否转发。



图 ‑1 MsgID定义

* MsgCnt：消息序号，根据不同的类型，取值范围0~127，表示本消息类型的发送计数。
* TemporaryID：设备ID，由参数指定。
* DSecond：时钟信号，单位为ms，有效值取值0~ 65535，由Tick得出。

#### MSG\_BasicSafetyMessage (BSM)

车辆基本安全消息，V2V，按WAVE标准100ms间隔发送，实现上发送间隔可设。此消息分两部分，PartI是必选；PartII可选，只在必要（有车辆警告）时发送。目前使用PartII的VehicleSafetyExtension项的EventFlag完成本车故障告警、紧急制动告警。

* **J2735的ASN.1格式定义：**

BasicSafetyMessage ::= SEQUENCE {

***-- Header items***

msgID DSRCmsgID, -- 1 byte

***-- Part I, sent as a single octet blob***

blob1 BSMblob,

***--***

***-- The blob consists of the following 37 packed bytes:***

***--***

***-- msgCnt MsgCount, -x- 1 byte***

***-- id TemporaryID, -x- 4 bytes***

***-- srcid TemporaryID, -x- 4 bytes***

***-- secMark DSecond, -x- 2 bytes***

***-- pos PositionLocal3D,***

-- lat Latitude, -x- 4 bytes

-- long Longitude, -x- 4 bytes

-- elev Elevation, -x- 2 bytes

-- accuracy PositionalAccuracy, -x- 4 bytes

***-- motion Motion,***

-- speed TransmissionAndSpeed, -x- 2 bytes

-- heading Heading, -x- 2 byte

-- accelSet AccelerationSet4Way, -x- 7 bytes

***-- control Control,***

***-- brakes BrakeSystemStatus, -x- 2 bytes***

BrakeSystemStatus ::= OCTET STRING (SIZE(2))

-- Encoded with the packed content of:

-- SEQUENCE {

-- wheelBrakes BrakeAppliedStatus, 4 bits

-- wheelBrakesUnavailable BOOL 1 bit (1=true)

-- spareBit 1 bit, set to zero

-- traction TractionControlState, 2 bits

-- abs AntiLockBrakeStatus, 2 bits

-- scs StabilityControlStatus,2 bits

-- brakeBoost BrakeBoostApplied, 2 bits

-- auxBrakes AuxiliaryBrakeStatus, 2 bits

-- }

***-- basic VehicleBasic,***

***-- size VehicleSize, -x- 3 bytes***

*-- Part II, sent as required*

safetyExt VehicleSafetyExtension OPTIONAL,

VehicleSafetyExtension ::= SEQUENCE {

events EventFlags OPTIONAL,

pathHistory PathHistory OPTIONAL,

pathPrediction PathPrediction OPTIONAL,

theRTCM RTCMPackage OPTIONAL,

}

EventFlags ::= INTEGER (0..8192) {

-- With bits as defined:

eventHazardLights EventFlags ::= 1

eventStopLineViolation EventFlags ::= 2 -- Intersection Violation

eventABSactivated EventFlags ::= 4

eventTractionControlLoss EventFlags ::= 8

eventStabilityControlactivated EventFlags ::= 16

eventHazardousMaterials EventFlags ::= 32

eventEmergencyResponse EventFlags ::= 64

eventHardBraking EventFlags ::= 128

eventLightsChanged EventFlags ::= 256

eventWipersChanged EventFlags ::= 512

eventFlatTire EventFlags ::= 1024

eventDisabledVehicle EventFlags ::= 2048

eventAirBagDeployment EventFlags ::= 4096

}

status VehicleStatus OPTIONAL,

}

* **帧格式：**

保留PathHistory和PathPrediction用于后续扩展。RTCMPackage主要用来修正GPS不使用，VehicleStatus和BrakeSystemStatus要通过CAN获取，但是BrakeSystemStatus是必选项因此保留（值填0），VehicleStatus删除。



图 8‑2 BSM消息帧格式

|  |  |
| --- | --- |
| 值 | 含 义 |
| TemporaryID | 设备的PID |
| SrcTID | 上一跳转发节点的设备PID（新增，用于多跳转发，如果非多跳此为发送设备PID或0） |

POSITION 3D：3D位置信息，定义如下，

|  |  |
| --- | --- |
| 值 | 含 义 |
| Latitude | 纬度，单位是1/10微度 |
| Longtitude | 经度，单位是1/10微度 |
| Elevation | 海拔高度，单位是0.1米 |
| Accuracy | 表示经纬度的精确度 |

MOTION：运动信息，定义如下，

|  |  |
| --- | --- |
| 值 | 含 义 |
| TransmissionAnd  Speed | Bits 14~16：TransmissionState  0-Neutral; 1-Park; 2-Forward; 3-Reverse; 7-unavailable |
| Bits 1~13： Speed：0-8191单位0.02m/s |
| Heading | 方向，单位0.0125度，范围(0..28800)  -- A range of 0 to 359.9875 degrees |
| AccelerationSet4Way | 加速度，包括  纵向: -2000~2000, 单位0.01m/s2  横向: -2000~2000, 单位0.01m/s2  垂直: -127~+127, 单位0.08m/s2  偏航率:-32768~+32768, 单位0.01deg/s |

BrakeSystemStause：车辆制动系统状态信息，定义如下，



图 8‑3 BrakeSystemStatus 定义

|  |  |
| --- | --- |
| 值 | 含 义 |
| TactionControlState | Bit9:8  00 - Not Equippted  01 - off  10 - on  11 - Engerd |
| AntiLockBrakeStatus | 同上 |
| StabilityControlStatus | 同上 |
| BrakeBoostApplied | 同上 |
| AuxiliaryBrakeStatu | 同上 |

VehicleSize：车辆尺寸信息，定义如下，

|  |  |
| --- | --- |
| 值 | 含 义 |
| VehicleWidth | 车宽，单位cm; 1- |
| VehicleLength | 车长，单位cm |

PART II

VehicleSafetyExtension：扩展安全信息，可选，必要时发送，定义如下，



图 8‑4 EventFlags 定义

|  |  |
| --- | --- |
| 值 | 含 义 |
| EventFlags | 车辆事件：目前使用两种  车辆故障:0x01  紧急制动:0x80  车辆翻车:0x800: |
| PathHistory | 可选,历史路径包括  FullPositionVector:位置  GPSstatus：gps状态  Count：HistorypointSets的点数：1-23  pathHistoryPointSets:PathHistoryPointType01-10中的1种 |
| PathPrediction | 可选，包括：  radiusOfCurve:单位0.1m,-32767~32767, 32767表示直线  confidence: 置信度0.5%, 0~200 |

* **处理流程：**

（注：采用本协议的通信节点在实际工作过程中需要同时处理接收和发送的任务，但是为了表示方便，下面所示的流程将节点1作为发送端，节点2作为接收端。下同）



图 8‑5 BSM消息处理流程

Step1：节点1和2的应用层通过start请求启动RCP协议工作；

Step2：节点1的RCP协议层自动周期广播BSM消息数据包；

Step3：节点2接收到节点1的BSM数据包后通知其应用层；

Step4：如果节点1的应用层发送stop请求，那么RCP协议层停止BSM数据广播；

注：bsm消息作为heartbeat报文，启动后一般不停止，若停止了，在有本车告警时也会开启。

另外：当车辆需要发送本车的告警时，处理流程与之前的EVAM相同，应用层依然调用active\_alert()/cancel\_alert()两个API，rcp层修降alert事件类型转换成BSM消息PartII中定义的事件值，增加到BSM信息中，并按事件发送的频率更新原BSM发送定时器timeout时间，当节点收到BSM解析出带Events字段后通知上层。

Step5：节点1应用层发出激活紧急制动告警（Alert 1）请求；

Step6：节点1的RCP协议层自动周期广播EVAM消息数据包（包含Alert 1）

Step7：节点2接收到节点1的EVAM数据包后通知其应用层；

Step8：节点1应用层发出激活车辆故障告警（Alert 0）请求；

Step9：节点1的RCP协议层更新BSM消息数据包内容（同时包含Alert 0和Alert1）；

Step10：节点1应用层发出取消紧急制动告警（Alert 1）请求；

Step11：节点1的RCP协议层更新BSM消息数据包内容（包含Alert 0）；

Step12：节点1应用层发出取消车辆故障告警（Alert 0）请求；

Step13：节点1的RCP协议层更新EVAM消息数据包内容（不包含任何Alert），并发送一次；

Alert：VAM层定义供上次调用, RCP层做alert到BSM中Event的互相转换。

#define VAM\_ALERT\_MASK\_VBD (0x1)

#define VAM\_ALERT\_MASK\_EBD (0x2)

#define VAM\_ALERT\_MASK\_VTO (0x3): VehicleTurnedOver

#### MSG\_RoadSideAlert (RSA)

路侧告警信息，目前需求支持

RoadSideAlert ::= SEQUENCE {

msgID DSRCmsgID,

-- the message type.

msgCnt MsgCount,

typeEvent **ITIS.ITIScodes**, -- 2 bytes in length

-- a category and an item from that category

-- all ITS stds use the same types here

-- to explain the type of the

-- alert / danger / hazard involved

description SEQUENCE (SIZE(1..8)) OF **ITIS.ITIScodes** **OPTIONAL**, -- up to 16 bytes

-- up to eight ITIS code entries to further

-- describe the event, give advice, or any

-- other ITIS codes

priority Priority **OPTIONAL**, -- 1 byte in length

-- the urgency of this message, a relative

-- degree of merit compared with other

-- similar messages for this type (not other

-- message being sent by the device), nor a

-- priority of display urgency

(Level0 – Level 7)

RawValue： bit[7:4] 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0

Routine(0), LeastImportant(1), 2, 3, 4, 5, 6, Most Urgent(7)

heading HeadingSlice **OPTIONAL**, -- 2 byte

-- Applicable headings/direction

HeadingSlice ::= OCTET STRING (SIZE(2))

***-- Each bit 22.5 degree starting from***

***-- North and moving Eastward (clockwise)***

***-- Define global enums for this entry***

extent Extent **OPTIONAL**, -- 1 byte in length

-- the spatial distance over which this

-- message applies and should be presented

-- to the driver

positon FullPositionVector **OPTIONAL**,

-- a compact summary of the position,

-- heading, rate of speed, etc of the

-- event in question. Including stationary

-- and wide area events.

furtherInfoID FurtherInfoID **OPTIONAL**,

-- a link to any other incident

-- information data that may be available

-- in the normal ATIS incident description

-- or other messages

-- 1~2 bytes in length

crc MsgCRC

}

**帧格式：**



图 8‑6 EVAM消息帧格式

其中，

|  |  |
| --- | --- |
| 值 | 含 义 |
| TypeEvent | 警告事件：由**ITIS.ITIScodes**定义,  bit31-bit16：类别:交通状况,事故,障碍等  bit15-bit00：每类的细化条目  需求规格所定的：  弯道/交叉口会车危险：0x1C07  急弯：0x1F5A(curve)  隧道：0x1F02， 0x2029  桥梁：0x1F01, 0x2025  十字路口：0x1F5D  信号灯：signal：0x2936, Observe-signals:0x1D07  限速： 0x0A04  其他：  Overturned-vehicle：0x022A  Ice: 0x1712  Emergency-vehicles-on-roadway:0x0704 |
| MsgCRC | CRC校验码 |
| 以下为可选项： | |
| Description | 由1-8个ITIScodes组成 |
| Priority | 优先级[bit7-bit5], b000, 001, 010, 011, 100, 101, 110, 111，表示这个Event的优先级 |
| HeadingSlice | 2Bytes, 360度方位，从正北方向顺时针每22.5度一个区间，依次bit0-15,每一位代表1个方位区间  noHeading = '0000'H  allHeadings = 'FFFF'H  000.0~022.5 = '0001'H  022.5~045.0 = '0002'H  045.0~067.5 = '0004'H  067.5~090.0 = '0008'H    090.0~112.5 = '0010'H  112.5~135.0 = '0020'H  135.0~157.5 = '0040'H  157.5~180.0 = '0080'H    180.0~202.5 = '0100'H  202.5~225.0 = '0200'H  225.0~247.5 = '0400'H  247.5~270.0 = '0800'H  270.0~292.5 = '1000'H  292.5~315.0 = '2000'H  315.0~337.5 = '4000'H  337.5~360.0 = '8000'H |
| Extent | 事件影响的范围：枚举值  useInstantlyOnly (0), useFor3meters (1), useFor10meters (2),  useFor50meters (3), useFor100meters (4), useFor500meters (5),  useFor1000meters (6), useFor5000meters (7), useFor10000meters (8),  useFor50000meters (9), useFor100000meters (10),forever (127) |
| FullPositionVector | 全位置信息 |
| FurtherInfoID | 相关事故消息2Byte 也由ITIScode表示 |

**处理流程：**



图 8‑7 EVAM消息处理流程

Step1：节点1和2的应用层通过start请求启动RCP协议工作；

Step2：节点1应用层发出激活路侧告警x（Alter rsa x）请求；

Step3：节点1的RCP协议层组包并自动周期广播RSA消息数据包（包含rsa x）

Step4：节点2接收到节点1的RSA数据包后通知其应用层；

Step5：节点1应用层发出取消路测告警x（Alert x）请求；

Step6：节点1的RCP协议层更新RSA消息数据包内容（不包含任何rsa x），并发送一次；

Step7：节点1的RCP协议层自动发送N次后停止发送RSA消息数据包；

Step8：节点2接收到节点1的RSA数据包后通知其应用层取消告警；

#### MSG\_EmergencyVehicleAlert (EVA)

紧急车辆告警信息，应用在救护车、警车或消防车等紧急车辆上，消息由这些车辆发出，用来提醒周边收到信息的车辆。此消息主体是RSA消息，除包含RSA外，还包含了本车辆类型、车重及受影响车辆

EmergencyVehicleAlert ::= SEQUENCE {

msgID DSRCmsgID,

id TemporaryID OPTIONAL,

rsaMsg RoadSideAlert,

-- the DSRCmsgID inside this

-- data frame is set as per the

-- RoadSideAlert. The CRC is

-- set to a value of zero.

responseType [ResponseType](%20#ASN_ResponseType) OPTIONAL,

details [EmergencyDetails](%20#ASN_EmergencyDetails) OPTIONAL,

-- Combines these 3 items:

-- SirenInUse,

-- LightbarInUse,

-- MultiVehicleReponse,

-- combine above three into one byte!

Mass VehicleMass OPTIONAL,

basicType VehicleType OPTIONAL,

VehicleType ::= ENUMERATED {

none (0), -- Not Equipped, Not known or unavailable

unknown (1), -- Does not fit any other category

special (2), -- Special use

moto (3), -- Motorcycle

car (4), -- Passenger car

carOther (5), -- Four tire single units

bus (6), -- Buses

axleCnt2 (7), -- Two axle, six tire single units

axleCnt3 (8), -- Three axle, single units

axleCnt4 (9), -- Four or more axle, single unit

axleCnt4Trailer (10), -- Four or less axle, single trailer

axleCnt5Trailer (11), -- Five or less axle, single trailer

axleCnt6Trailer (12), -- Six or more axle, single trailer

axleCnt5MultiTrailer (13), -- Five or less axle, multi-trailer

axleCnt6MultiTrailer (14), -- Six axle, multi-trailer

axleCnt7MultiTrailer (15), -- Seven or more axle, multi-trailer

... -- # LOCAL\_CONTENT

}

-- values to 127 reserved for std use

-- values 128 to 255 reserved for local use

***-- type of vehicle and agency when known***

vehicleType IT IS.VehicleGroupAffected OPTIONAL,

responseEquip IT IS.IncidentResponseEquipment OPTIONAL,

responderType IT IS.ResponderGroupAffected OPTIONAL,

crc MsgCRC,

}

帧格式：



图 8‑8 EVA消息格式

处理流程：



图 8‑9 EVAM消息处理流程

Step1：节点1和2的应用层通过start请求启动RCP协议工作；

Step2：节点1应用层发出激活x告警（Alert x）请求；

Step3：节点1的RCP协议层自动周期广播EVA消息数据包（包含Alert x）

Step4：节点2接收到节点1的EVA数据包后通知其应用层；

Step5：节点1应用层发出激活y告警（Alert y）请求；

Step6：节点1的RCP协议层更新EVA消息数据包内容（同时包含Alert y和Alert x）；

Step7：节点1应用层发出取消x告警（Alert x）请求；

Step8：节点1的RCP协议层更新EVA消息数据包内容（包含Alert y）；

Step9：节点1应用层发出取消y告警（Alert y）请求；

Step10：节点1的RCP协议层更新EVA消息数据包内容（不包含任何Alert），并发送一次；

注：同BSM的本车alert，VAM定义上层调用的AlertID，rcp层完成AlertID到EVA消息中告警ID的互转。

#### MSG\_ProbeVehicleData (PVD)\*

J2735规定PVD是由车辆发送给可接收探测的RSU，单播发送。此消息包括车辆当前和过去的位置及状态信息，历史信息由1-32个Snapshots（快照）组成，每个Snapshot包含车辆位置、车辆安全扩展及车辆状态信息。

PVD何时发送？DSRC协议栈规定，当OBU收到一个RSU的由PSID为5和PSC为3标识的应用服务公告（公告由固定信道的TA帧承载传输）时， OBU开始通过RSU公告指定的服务信道发送自身PVDs给RSU。

但是由于我们目前系统仅用1个信道1种帧，所以设计为收到RSU的PDM帧后，开始发送PVD。

ProbeVehicleData ::= SEQUENCE {

msgID DSRCmsgID, -- App ID value, 1 byte

segNum ProbeSegmentNumber OPTIONAL,

-- a short term Ident value

-- not used when ident is used

probeID VehicleIdent OPTIONAL,

-- ident data for selected

-- types of vehicles

VehicleIdent ::= SEQUENCE {

name DescriptiveName OPTIONAL,

-- a human readable name for debugging use

vin VINstring OPTIONAL,

-- vehicle VIN value

ownerCode IA5String(SIZE(1..32)) OPTIONAL,

-- vehicle owner code

id TemporaryID OPTIONAL,

-- same value used in the BSM

vehicleType VehicleType OPTIONAL,

vehicleClass CHOICE

{

vGroup ITIS.VehicleGroupAffected,

rGroup ITIS.ResponderGroupAffected,

rEquip ITIS.IncidentResponseEquipment

} OPTIONAL,

... -- # LOCAL\_CONTENT

}

-- Roy: above two items could be in a CHIOCE statement?

startVector FullPositionVector, -- the space and time of

-- transmission to the RSU

vehicleType VehicleType, -- type of vehicle, 1 byte

cntSnapshoots INTEGER (1..32) OPTIONAL,

-- a count of how many snaphots

-- type entires will follow

snapshots SEQUENCE (SIZE(1..32)) OF Snapshot,

-- a seq of name-value pairs

-- along with the space and time

-- of the first measurement set

} -- Est size about 64 bytes plus snapshot sizes (about 12 per)

帧格式：



图 8‑10 PVD消息格式

**Probe Snapshot采样（生成）：**

每个 Probe Data Message由一系列的 Probe Data Snapshots 组成，在没有相关PVD 的情况下，车辆有以下三种方式自动生成snapshot：

* Periodically – at intervals based on vehicle movement between RSUs
* Event Triggered – these occur when the state of certain vehicle status elements change:发动机on/off等
* Starts and Stops – these occur when a vehicle starts moving and stops moving

周期性生成snapshot时的时间间隔T：跟车速V相关

|  |  |
| --- | --- |
| 时间间隔T | 速度V |
| T=4s | V<=20 mph(8.9m/s) |
| T=aV+b， T与V线性相关 | 20mph < V < 60mph |
| T=20s | V>=60mph(26.8 m/s) |

PVD消息中snapshot的发送流程：



图 8‑11 snapshot的发送流程图

#### MSG\_ProbeDataManagement (PDM)\*

PDM消息是RSU广播发送给所以附近的OBU，用来控制OBU发送PVD消息中snapshot生成的频率、宽度等参数。

ProbeDataManagement ::= SEQUENCE {

msgID [DSRCmsgID](%20#ASN_DSRCmsgID), -- This is a unique message

-- identifier, NOT related to

-- the PSID\PSC

sample [Sample](%20#ASN_Sample), -- identifies vehicle

-- population affected

directions [HeadingSlice](%20#ASN_HeadingSlice),

-- Applicable headings/directions

term CHOICE {

termtime [TermTime](%20#ASN_TermTime), -- Terminate management process

-- based on Time-to-Live

termDistance [TermDistance](%20#ASN_TermDistance) -- Terminate management process

-- based on Distance-to-Live

},

snapshot CHOICE {

snapshotTime [SnapshotTime](%20#ASN_SnapshotTime), -- Collect snapshots based on time

snapshotDistance [SnapshotDistance](%20#ASN_SnapshotDistance) -- Collect snapshots based on Distance

},

txInterval [TxTime](%20#ASN_TxTime), -- Time Interval at which to send snapshots

cntTthreshold INTEGER (1..32), -- number of thresholds that will be changed

dataElements SEQUENCE (SIZE(1..32)) OF

[VehicleStatusRequest](%20#ASN_VehicleStatusRequest),

-- a data frame and its assoc thresholds

}

帧格式：



图 8‑12 PDM消息格式

|  |  |
| --- | --- |
| 值 | 含 义 |
| Sample |  |
| HeadingSlice | 方位信息 |
| Term | |
| TermPR | typedef enum term\_PR {  term\_PR\_NOTHING, /\* No components present \*/  term\_PR\_termtime,  term\_PR\_termDistance  } term\_PR; |
| TermTime | PVD消息停止时间 |
| TermDistance | PVD消息停止的距离 |
| Snapshot | |
| SnapshotPR | typedef enum snapshot\_PR {  snapshot\_PR\_NOTHING, /\* No components present \*/  snapshot\_PR\_snapshotTime,  snapshot\_PR\_snapshotDistance  } snapshot\_PR; |
| [SnapshotTime](file:///C:\Users\073\Desktop\vanet_rcp\0设计\#ASN_SnapshotTime) | Snapshot生成的时间间隔 |
| [SnapshotDistance](file:///C:\Users\073\Desktop\vanet_rcp\0设计\#ASN_SnapshotDistance) | Snapshot生成的距离间隔 |
| [TxTime](file:///C:\Users\073\Desktop\vanet_rcp\0设计\#ASN_TxTime) | PVD发送的时间间隔 |
| cntTthreshold |  |
| [VehicleStatusRequest](file:///C:\Users\073\Desktop\vanet_rcp\0设计\#ASN_VehicleStatusRequest) | 请求的车辆状态 |

#### MSG\_MapData (MAP)\*

RSU发送给Vehicle的消息，指示GID地图数据。

MapData ::= SEQUENCE {

msgID DSRCmsgID,

msgCnt MsgCount,

-- updated when message content changes

name DescriptiveName OPTIONAL,

layerType LayerType OPTIONAL,

LayerType ::= ENUMERATED {

none (0),

mixedContent (1), -- two or more of the below types

generalMapData (2),

intersectionData (3),

curveData (4),

roadwaySectionData (5),

parkingAreaData (6),

sharedLaneData (7),

}

layerID LayerID OPTIONAL,

intersections SEQUENCE (SIZE(1..32)) OF

Intersection OPTIONAL,

***-- other objects may be added at this layer, tbd,***

***-- this might become a nested CHOICE statement***

***-- roadSegments SEQUENCE (SIZE(1..32)) OF***

***-- RoadSegments OPTIONAL,***

***-- curveSegments SEQUENCE (SIZE(1..32)) OF***

***-- curveSegments OPTIONAL,***

***-- wanted: some type of data frame describing how***

***-- the data was determined/processed to go here***

dataParameters [DataParameters](%20#ASN_DataParameters) OPTIONAL,

crc [MsgCRC](%20#ASN_MsgCRC),

}

### 接口设计



图 8‑13 rcp接口

接口R1：rcp\_send()：供上层调用发送rcp消息到对端

接口R2：rcp\_recv()：提供给dsmp模块调用

接口R3：rcp\_create\_forward\_msg()和rcp\_send\_forward\_msg ()：提供给多跳转发模块调用，用于创建和发送转发帧

接口I1：见7.1.2.1，主要是获取、释放发送buffer

## 多跳转发模块设计

### 功能描述

本模块主要负责实现对于紧急类的RCP告警消息的多跳转发功能。在VANET网络中，通信节点间的多跳转发是一个热点和难点，可以有效扩大单个节点的通信范围。本系统中目前采用的是简单的基于地理位置进行优化的的广播转发算法，并且将转发算法的实现放在应用层的方案，主要是出于以下考虑：

* 摆脱对通信底层的依赖，使其具有广泛的应用性和良好的可移植性
* 算法主要的输入因素是GPS提供的位置、速度以及进一步计算得出的节点间相对距离等条件，因此从逻辑上更贴近应用层而不是通信层

将转发算法上移到应用层的方式也会带来一些缺点，主要是处理环节增多而导致的转发延迟加大。

本模块的设计原则为尽可能保持转发算法的独立性，便于后期算法的升级更新。

### 广播路由算法设计

#### VANET网络路由特点

车载网络具有许多特殊的性质，这些特性对 VANET路由协议的设计至关重要。它们产生的影响主要有如下几点：

（1） 车辆节点运动速度快，城市交叉路口众多，道路拓扑结构复杂，两个节点的相对位置往往在极短的时间内就会发生变化，导致已经建立起的路由路径平均寿命很短。

（2） 移动节点的分布受到道路的限制，往往呈现一条纵列，导致空间复用率低，网络容量非常有限。加上车辆数目众多，主干道或高峰期的车流密度很大，因此不适宜使用洪泛广播。

（3） 路边建筑物和绿化带的遮挡效应导致位于不同道路上的两辆车之间，即使处在彼此的通信范围之内，也难以进行直接通信。而两条相隔较远的道路之间若没有其它道路的存在，也会因缺乏中继节点而无法进行多跳。

（4） 车辆节点的运动路径具有规律性。若综合考虑驾驶员的行为、主次干道的分布、周边的生活环境和地理因素以及具体的出行时间等，则可对节点的运动趋势进行预测。

（5） 通过使用GPS 、导航、下一代网络（Next Generation Network；NGN）和电子地图等技术成果，车辆节点可以获得极为丰富的地理位置、道路拓扑、网络节点分布等信息的支持。

（6） 车载设备的续航能力和运算能力相对一般移动节点来说很强，因此设计协议时，无需在功率控制、复杂度等方面耗费大量思路。

综合考虑上述因素，VANET的网络拓扑具有高度动态特性，很难寻找并维持一条持续稳定和高效的路径。而VANET所提供的安全服务应用对通信质量要求较高，要求路由协议具有高度的自适应性和灵活性。考虑到无限频谱资源的限制，用于建立以及维护路由的控制开销也是衡量其路由技术性能的重要因素。此外，VANET中的终端节点分布很不均匀，主要受到道路结构限制，交通时段和路段，建筑物等的遮挡，交通灯和驾驶员主观行为等的影响，这些都造成了VANET网络的割裂现象，给VANET中路由技术的研究带来了巨大的挑战。但同时VANET中的节点是电子设备丰富的车辆终端，能量、计算和存储能力都不再是约束其路由技术的因素，车上的GPS、电子地图等设备都可作为路由技术研究中的辅助条件。另外，节点运动又是有规律可循的，车辆通常都是沿着道路运行，且其速度和方向具有一定的连贯性，根据历史数据可以预测其未来一段时间内的运动轨迹，以此作为路由发现时的参考因素，可以有效的提高路径的持续时间等。

因此，在 VA N E T路由协议的设计中，应该趋利避害，充分利用车载设备信息充足、功能强大的优点，以克服其在信道质量和通信容量上的不足。同时，从需求出发，应该主要从路由建立所需的时间、数据传输的时延和可靠性、对快速动态变化的网络拓扑结构的适应性等方面来衡量车载自组织网络路由协议的性能。

#### 路由实现方案分析

在VANET的V2V通信方式中，由于收到道路和车载设备发射功率的限制，车辆的一跳通信范围往往有限，通常为250m-300m，而道路车辆又处于高速运动中，尤其是高速公路上的车辆，一旦出现危害交通安全的事故，如果安全消息不被及时扩散出去，往往会引发连环车祸，造成不可估量的后果。因此，安全消息需要经过多跳传输，快速地扩散到事故后方更远的车辆，以提醒驾驶员采取相应规避措施，避免事故或者减轻事故的危害。

本文所设计的车辆安全应用中，主要安全消息包括三类，1.车辆紧急制动报警。2.车辆故障事故报警。3.车辆安全距离报警。以下以其中一个场景为例，分析说明安全消息按照多跳路由协议扩散的过程。



图 8‑14安全应用场景之一

车辆A由于紧急制动、或车辆故障等原因造成突然减速，车载通信设备发起安全信号，并迅速广播，由于道路车辆行驶速度快，后方车辆如果不及时收到警告消息并采取相应规避措施，很可能造成连环车祸，又由于受到发射功率和无限信道不稳定等因素影响，车辆通信范围有限，需要进行多跳才能将消息转发到较远车辆，确保整个道路上车辆都能采取一定措施避免交通事故。车辆B在A的一跳范围收到广播消息后报警告知驾驶员，并转发该广播消息给后方更远的车辆C，C经过相同的操作处理广播消息，这样循环下去，直到该消息扩散到一定距离后过期被丢弃。

以上描述了一个安全消息从产生到广播、转发的简单过程，实际上就是一个简单的消息路由过程，其功能就是将安全消息扩散给周围及更远处的更多车辆。而在实际应用场景中，道路车辆众多，而且运动速度快，无线信道传输质量不稳定，等等众多因素影响安全消息的快速可靠传输。因此，VANET中广播路由协议实现难度较大，需要解决节点之间传输多跳的问题。广播的成功率、通信开销和传输延迟、道路车辆的覆盖率、数据包的碰撞等都是VANET中广播路由协议所面临的挑战。

最常见的多跳广播路由算法是泛洪（flooding），基本思想是节点收到一个非重复的数据包马上转发，在稀疏网络拓扑中，泛洪方式能获得优良的路由性能，能以很短的传输时延将安全消息成功覆盖到较远区域的车辆。而在大规模的场景如城市上下班密集交通环境中，大量节点的广播帧转发很快消耗掉有限的网络资源，很容易造成网络拥塞甚至瘫痪。如下图所示场景为例加以说明：



图 8‑15大量节点转发示意图

从上图可以可以看出，随着网络中参与通信的节点数目和转发次数增大，整个网络中的数据通信量会成指数级的增长，这会导致广播风暴，主要表现为：

* 同一时刻大量节点同时竞争信道，使得碰撞几率大大增加，从而降低信道物理带宽的利用率
* 转发跳数增多将会导致传输时延增大，降低传输的实时性
* 相同的数据被大量、频繁的转发，使得有效数据传输时间并挤占，导致信道的有效使用率降低

因此，在V2V无线广播通信路由机制中，需要解决最根本的问题是如何减少无效的数据发送，提高转发的效率。要实现此目标，可以从以下三个方面考虑：

1. 减少网络跳数，这点比较容易做到，我们只需根据实际应用（如通信距离）的需求，在网络通信中对广播帧设定有限的转发次数上限即可；
2. 减少转发的节点数，与单纯的减少网络跳数不同，要在保证一定的通信质量的前提下，如何选择有效的转发节点是一个复杂的问题，结合V2V通信网络结构的特点，可以采用的转发节点选择策略，包括：
   1. 最远节点法，即在网络中，仅选择与源节点最远的一个或多个节点参与下一跳转发；
   2. 邻居节点法：按照这种方式，只有当接收到数据的节点有比源节点联系不上的邻居节点时才进行转发
   3. 反向节点法：由于V2V通信特点所致，对向行驶的车辆可以作为一个天然的理想的数据转发节点，将源节点的数据快速的传递到后方；
   4. 其它。。。
3. 减少网络负载，在保障信息传输可靠性的同时，尽可能降低网络中控制数据量，减少负载对有限宽带资源的占用，避免网络拥塞。

这也是本文中所述的路由协议所要解决的重点问题，具体到本产品的实际应用场景，根据已有的研究成果，下面提出了几种改进广播转发效率的路由协议实现方案，其基本思想都是基于尽量减少参与转发的节点数的原则，以最小的网络开销、最短的传输时、将安全消息广播到更远距离的车辆，以提高道路的行车的安全性和道路的使用效率。按照针对层面和实现难易程度分类如下：

（一）最远节点转发准则

VANET应用场景中，车辆节点通过周期性的BSM消息交互自己的位置、速度等基本安全消息，节点收到BSM消息后可以建立自己周围邻居节点的邻居列表，成员就是数据一跳广播对象，也是多跳传输的中继转发对象。为了避免泛洪算法大量冗余数据造成广播风暴的问题，在所有邻居节点中选择距离源节点最远的节点作为中继转发节点，其他节点收到转发包后纷纷停止转发，从而达到减少重复数据包大量转发的目的，大大减少网络冗余。此为最易实现的方案之一，下面几种方案均是从不同角度在此方案上的改进。

（1）基于竞争转发的后验路由

后验路由中继转发的判断在数据包的接收端。源节点产生安全消息并广播之后，其一跳范围的邻居节点均收到数据包，它们分别竞争转发该数据包，竞争准则是等待一定时延，该时延根据源节点与自身的距离决定，距离越远，等待时间越短，越先竞争得到转发权，其他节点收到转发包后放弃竞争而丢包。此方案的缺点是传输时延长，因为需要经过等待一定的时延用来竞争转发权。

（2）基于位置预测的先验路由

为了让消息尽快被转发，提出了基于位置的先验路由，即路由决策在发送端完成，源节点在发送数据包之前查询邻居节点中距离自身最远的邻居选为转发节点，写入数据包，邻居节点收到该数据包后查询转发节点ID，匹配上了则转发，否则直接丢包，无需等待大大提高了数据包转发效率，而且无需缓存，减轻硬件压力。然而，邻居列表中被选中的最远节点可能不是最佳的，因为邻居列表的建立是通过周期性的BSM消息，此消息有一定的延迟，由于车辆一直处于高速运动中，在下一个BSM消息到来之前，原本互为邻居的节点此时已经离开了对方的通信范围，造成邻居误判，从而没有节点完成转发造成丢包，而最远邻居出现误判的几率最大。考虑到获取的邻居节点信息可能会存在一定程度的滞后性，为了减轻这种影响，可以引入位置估计机制，根据车辆的速度，加速度，时间戳，经纬度，和历史轨迹预判出现邻居收到自身发出的BSM消息时刻的位置写入对方的邻居列表，使得所有节点的邻居列表的位置准确性更高。

（3）基于ACK反馈的可靠路由

为了保障数据包转发的可靠性，引入ACK确认机制，确保有效邻居节点转发数据包。首先定义一种消息交互机制，使车辆能获取当前其周围节点的距离或传输质量信息，同时，在报警消息中指定需要进行转发该消息的节点。当车辆发出报警时，只有与之相邻的车辆节点能接收到该广播消息，这些邻居节点接收到该消息后会向源节点反馈ACK，这样源节点就可以通过这些消息选择距离自己最远的N个节点，后续的广播消息包含指定的这N个节点的地址信息，进而由这些节点进行下一步转发。以此类推。最远节点的信息设置一个时效，一段时间（如10秒）后需重新启动选择过程。这种方式可以凭借较少的外部条件就有效筛选出最佳转发节点，但其缺点是增加了消息交换机制，人为的增加了传输延迟。

（4）基于RTS/CTS的链路自适应路由

利用CSMA/CA中的RTS/CTS握手机制，广播节点发送RTS包，邻居节点收到后回复CTS包，自适应地计算回复等待时间，与相对距离、相对速度、信道质量、传输速率、发射功率等因素有关， 链路质量高的节点优先转发数据包，其他节点收到转发数据包后停止计时，这类路由协议可以避免隐藏终端、传输可靠性的影响，很大程度上提高了网络效率，但是其面临信道利用率低，传输时延增加。

（5）**（单播逐跳确认机制）**

在选择最远的节点时候，同样基于有邻居表的状态，可以以单播的方式按照最远到最近的顺序分别给所有邻居点发送广播数据，单播是有ack的，所以可以知道最远的一个点有没有收到自己的广播，没有就选择第二远的点，依次类推，直到单播成功，一旦单播成功，就不再选择后续的中继点，所以最后只会有一个且一定有一个成为中继点，从而保证自己有可靠的中继节点，避免了邻居表滞后的不可靠影响。同时无线的敞开性，不是目的地址的也可以监听到这个广播数据，从而得到告警信息，效果和广播一样。缺点是节点需要等待ACK才转发增加了传输时延。

当最远的节点成为中继节点后，要考虑是不是它要成为独立的广播源，从而不依赖与最原始的广播源，因为最远的节点通讯很脆弱，那应该成为独立源是更好的选择。这样最多网络中会有4个独立广播源。但我们还没考虑，有几辆车同样达到触发条件和连锁情况的出现，这时网络的广播源会非常多，那么广播风暴怎么来克服呢？

（二）邻居差异转发准则

基于邻居节点转发原则，每一个接收到的广播消息的节点都根据自身的邻居节点情况与源节点的邻居节点情况进行对比，从而决定是否需要进行转发。在这种方式下，网络中的相邻节点需要交换各自的邻居表信息，这可以基于网络中节点的周期性状态包的交互来实现。这是一种分布式的、按需转发的方式，在特定的网络模式下，如高速公路的节点线性分布时，可以有效的进行覆盖。这种方法的缺点是转发策略都是基于单个节点来实现的，没有进行全网优化，在一些复杂的网络结构中，可能会导致性能较差。

（三）反向节点转发准则

由于车辆运动收到道路和交通规则的限制，道路上的车辆一般都是双向车道行驶，当安全消息产生后，主要是警告后方车辆及时采取措施，因此提出基于反向节点转发原则，选择对向车辆作为广播源，让它持续广播消息，将前方车辆的报警信息迅速传递到后方。这种方式的优点非常明显，消息传递快速、高效、准确。但是其缺点也同样显著，即实际场景下局限性比较大，而且会造成完全不同的网络模型，因此这种方法不适合单独使用，而且与其它方式配合使用可能会获得比较好效果。

（四）跨层设计路由算法

传统的层级协议构架已渐渐不再适用VANET网络。例如在严格的层级协议构架中，由于传输层无法获知物理层的信息，无线信道的衰落等因素而产生的丢包常被误判成网络拥塞，而这会导致系统采取错误的措施来应对拥塞，浪费了系统资源，增大了开销，而且不能解决实际问题。因此，跨层技术可用来实现网络性能优化的目标。在本文的实际应用中，要实现安全消息的可靠快速传输，可以结合物理层节点链路的通信条件和MAC层接入碰撞状况等信息，随着数据帧逐层上传到传输层来跨层设计路由，提高全局的网络传输效率。

借助跨层技术，VANET网络的协议栈具有了动态适应环境的能力，并能够通过联合协议栈多层的参数来高效的进行端到端重配置，实现网络的自主适变，改善、提升全局网络的端到端性能。在路由设计中，结合路由选择和频谱选择的跨层设计方法，同二者单独研究相比，会有更好的性能表现。因此，我们后续将对此进行深入研究来优化路由算法。

将以上积累方案分类整理如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 方案 | 名称 | Beacon  交互 | ACK  确认 | 优点 | 缺点 |
| 基于最远  邻居中继  转发策略 | 基于竞争转发的后验路由 | × | × | 设计简单 | 可靠性低时延大 |
| 基于位置预测的先验路由 | √ | × | 邻居实时性高 | 依赖于预测算法 |
| 基于ACK反馈的可靠路由 | × | √ | 可靠性高 | 时延、负载大 |
| 基于RTS/CTS的链路自适应路由 | × | √ | 可靠性高 | 时延大信道利用率低 |
| 单播逐跳确认机制 | × | √ | 可靠性高 | 时延大 |
| 邻居差异转发准则 |  | √ | × | 实现简单 | 扩展性低 |
| 反向节点转发准则 |  | √ | × | 时延低 | 扩展性低 |
| 跨层设计路由算法 |  | √ | × | 性能优化 | 实现复杂 |
|  |  |  |  |  |  |

#### 路由算法设计

从上节分析可以看出，不同的路由实现方案都有各自的优势和不足，因此一个理想的解决方案是根据不同场景，灵活的选择最适合的策略，充分发挥各自的优点，实现优势互补，从而在各种场景下都能获得较好的效果。

在本方案的开发中，本着从实际出发的原则，首先在以**方案一中最远节点转发准则**为基础进行实现最基本的基于竞争转发的后验路由算法，结合后续的实际测试表现，同时吸取其它方案的思想和优势，不断的对算法进行完善和优化。

下面以一个实际的场景来例，来具体说明本方案的算法设计思想：



图 8‑16数据包广播转发示意图

举例说明：节点A由于交通事故产生安全消息，车载模块监测到事故马上广播安全消息，其一跳范围的邻居节点（BCDEFG）收到广播包之后按照路由算法转发该安全消息给其他车辆，直到道路上的其他车辆都收到该安全消息，作出相应的驾驶控制行为，确保行车安全。该路由算法需要尽可能地减少网络负载，即该安全消息的重复包数量以及对信道的占用时间，尽可能地快速扩散该安全消息，尽可能地最大范围覆盖道路区域车辆。本方案的路由算法流程图如下所示：





图 8‑17 路由算法流程图

路由算法具体步骤如下：

1、节点收到数据包，先检查队列是否包含此数据包，判断依据是网络层源地址和网络层包序号。

2、如果不重复，则说明收到新数据包，则根据公式（1-1）计算转发等待时延，竞争转发权，距离上一跳节点距离越远的节点优先竞争到转发权转发数据包。

（1-1）

其中为单位时隙，dij为相对距离，R为车车通信范围。

3、如果重复数据包list重复，再查询转发list，如果有说明还没转发，（其他节点已经转发数据包）则直接丢包。如果没有，不做任何处理（待数据包超时删除）。

本方案的路由算法中，主要思想是对flooding算法在转发节点的选择和转发跳数的限制上优化，采用最远邻居节点转发准则优选中继转发节点，此种算法虽然能大大降低网络中冗余数据量，在节点密集时众多安全消息并发依然会耗尽网络资源，造成信道拥塞。因此，需要进一步优化，后续优化方案主要包括在保证广播可靠性的同时优选最佳中继转发节点，结合传输时延、链路状况等因素进行跨物理层和MAC层的跨层路由协议设计，最大限度地提高路由整体性能，算法具有良好的扩展性，能广泛适应各类路况。

### 接口设计

#### 接口关系

本模块分别与DSRC消息协议模块和应用编程接口模块之间存在接口关系，如下图所示：



图 8‑18 多跳转发模块接口关系

#### 接口I1

* mda\_handle

表 8‑1 mda\_handle原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | int mda\_handle(mda\_envar\_t \*p\_mda , mda\_msg\_info\_t \*src\_sta, mda\_msg\_info\_t \*pre\_sta, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | mda\_envar\_t \*p\_mda | Mda结构变量指针 |
| mda\_msg\_info\_t \*src\_sta | 源节点的信息 |
| mda\_msg\_info\_t \*pre\_sta | 上一转发节点的信息 |
| uint8\_t \*pdata | 原始消息数据首地址 |
| uint32\_t length | 原始消息数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：0 – 成功， <0 – 失败，错误代码 | |
| 功能描述 | RCP消息多跳转发处理。 | |

#### 接口X1

由应用编程接口模块提供的API接口，定义参见第8.1节。

#### 接口X2

由RCP模块提供给本模块调用的接口，定义如下。

* rcp\_create\_forward\_msg

表 8‑2 rcp\_create\_forward\_msg原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | wnet\_txbuf\_t \*rcp\_create\_forward\_msg(uint8\_t left\_hops, uint8\_t \*pdata, uint32\_t length); | |
| 输入项 | uint8\_t left\_hops | 剩余转发跳数 |
| uint8\_t \*pdata | 原始消息数据首地址 |
| uint32\_t length | 原始消息数据长度 |
| 输出项 | 无 | |
| 返回值 | 返回处理结果：网络发送buffer首地址 | |
| 功能描述 | 创建转发消息，内部申请网络发送buffer，并完成消息体的复制和更新。 | |

* rcp\_send\_forward\_msg

表 8‑3 rcp\_send\_forward\_msg原型定义

|  |  |  |
| --- | --- | --- |
| 项 目 | 描 述 | |
| 原型定义 | void rcp\_send\_forward\_msg(wnet\_txbuf\_t \*txbuf); | |
| 输入项 | wnet\_txbuf\_t \*txbuf | 网络发送buffer首地址 |
| 输出项 | 无 | |
| 返回值 | 无 | |
| 功能描述 | 发送转发消息。 | |

### 实现原理

#### 内部结构



图 8‑19 多跳转发模块内部结构图

#### 数据结构



图 8‑20 多跳转发模块数据结构

本模块数据结构做为VANET中间件部分数据结构“vam\_envat\_t”的一个子结构，在系统中的位置及相互关系如上图所示。主要结构体定义如下：

* mda\_envar\_t：

|  |
| --- |
| typedef struct \_mda\_envar {  osal\_task\_t \*task\_mda;  osal\_timer\_t \*timer\_forward;  osal\_timer\_t \*timer\_rx\_history;  osal\_sem\_t \*sem\_forward;  osal\_sem\_t \*sem\_rx\_history;  list\_head\_t forward\_waiting\_list;  mda\_history\_t rx\_history\_table[RX\_HISTORY\_NUM];  } mda\_envar\_t; |

其中：

* task\_mda：转发处理线程
* timer\_forward：转发定时器
* timer\_rx\_history：接收历史帧超时定时器
* sem\_forward：转发帧队列同步信号量
* sem\_rx\_history：接收历史帧队列同步信号量
* forward\_waiting\_list：转发帧等待队列
* rx\_history\_table:接收历史记录表

#### 处理流程



图 8‑21 多跳转发处理流程



图 8‑22 超时处理流程

# 车辆安全应用部分

## 总体设计

### 车辆安全应用在VANET软件架构中的层次

车辆安全应用模块(简称VSA)，位于VANET软件架构里面中间件(VAM)的上层，VANET中间件通过回调函数发送相关车辆信息到VSA模块，在VSA模块中，通过对自身和邻近车辆信息的处理，来判断是否对本车发出预警或者取消预警，也同时决定是否向外广播报警或者取消发送报警。VSA模块通过消息队列的方式将信息传送给系统模块，最后再由系统模块统一进行人机的交互处理。



图 9‑1车辆安全应用的软件架构层次

### 车辆安全应用危险场景

根据目前的模块硬件配置和实际道路情况，现将危险场景分为四类，分别是**近距离危险**，**紧急制动危险**和**车辆障危险**，交叉路口碰撞预警。前三类使用的是V-V模块，第四种使用在I-V的场景中。其中，近距离危险还分为前车近距离和后车近距离，用于处理前车和后车的近距离危险接近。其中紧急制和车辆故障都包括对本车的危险预警和对邻车的报警发送。

车辆安全应用使用VAM中间件提供的路径预测功能，每100ms进行一次数据获取，将所得到的数据送入位置信息预处理模块进行加工，得到VSA安全应用可以直接使用的数据。其中，EEBL和VBD还需要BSM提供的报警消息。

下面是一些专用名词的简写：

CICAS：Cooperative Intersection Collision Avoidance System

CCW：Cooperative Collision Warning

CFCW：Cooperative Forward Collision Warning

CRCW： Cooperative Rear Collision Warning

EEBL：Emergency Electronic Brake Lights

VBD：Vehicle Broken Danger



图 9‑2车辆安全应用危险场景软件处理结构

### 车辆相对位置分类

根据车辆的相对GPS数据可以把车辆位置分为8类，见下图。由于GPS精度的限制，目前模块能够识别的位置分成6类，见下图表示的相对位置，分别是左前，前方，右前，右后，后方，左后方。

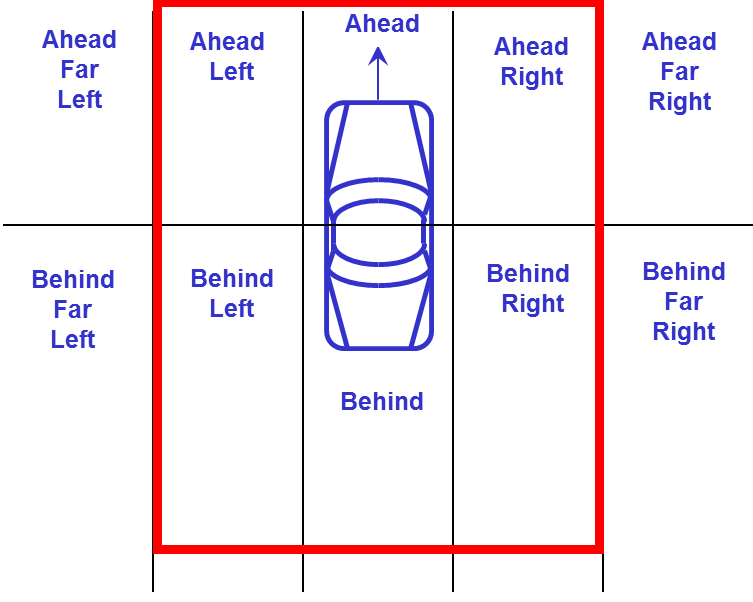


图 9‑3车辆相对位置分类

### 车辆位置预处理模块

根据VANET软件架构的特点，需要一个数据预处理模块来进行共同数据的获取和处理，以便减少运算数量，降低处理器的负担。位置信息模块会将原始的车辆位置信息，进行运算，来获得上层安全应用直接使用的数据。

具体需要从vam模块中的邻结点列表来获取数据，经过加工，生成新的VSA可用数据结构链表。运行周期为100ms。

新生成的数据结构包括PID，相对位置，相对速度，横向偏移，纵向偏移，直线距离，加速度，方向角和跳数。



图 9‑4位置预处理模块示意图



图 9‑5位置数据结构组成

### 相关数据结构



图 9‑6 VSA部分数据结构

## 近距离危险（CCW）模块设计

### 功能描述

近距离危险(CRD)模块是用来检测当邻车与本车距离过近时，是否会造成危险的一种预判。判断基于本车速度和行驶方向，邻车的速度和行驶方向，以及两车车的相对距离。当满足报警条件时，会触发近距离危险报警，分为前车报警(CFCW)和后车报警(CRCW)，此模块只有本车报警功能，不具备向邻车发送的功能。

CFCW：



图 9‑7前车近距离碰撞预警示意图

CRCW：



图 9‑8后车近距离碰撞预警示意图

### 接口设计

#### ccw\_proc

|  |  |  |
| --- | --- | --- |
| 条目 | 内容 | |
| 原型定义 | static int ccw\_proc(vsa\_envar\_t \*p\_vsa, void \*arg) | |
| 输入项 | p\_vsa | vsa\_envar\_t数据结构指针 |
| arg | 空指针，传递消息队列消息 |
| 输出项 | 无 | |
| 返回值 | 0 | 判定无近距离危险 |
| 1 | 判定有近距离危险 |
| 功能描述 | 回调函数，通过传入的vsa节点信息，判断是否有近距离危险 | |

### 实现原理

#### 判断条件

**近距离前方预警(CFCW):**

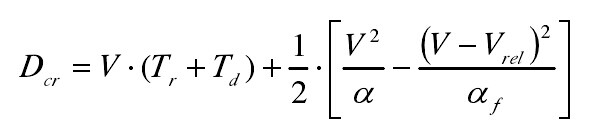
|  |  |
| --- | --- |
| 近距离前方危险检测预警(CFCW) |  |
| 输入参考量 | 两车行驶方向 ，相对速度，相对距离，本车速度，相对位置信息，邻车速度 |
| 触发条件 | 1. 本车车速大于**安全速度**(可设置)时，启动CFCW预警 2. 本车和邻车**同向**行驶 3. 本车位于邻车**后方** 4. 本车车速减去邻车车速大于**相对安全速度**(可设置) 5. 两车距离小于**安全距离**(根据公式计算得出) |
| 解除条件 | 1.上述任一触发条件不满足时 |
| 其他要求 |  |

**近距离后方预警(CRCW):**

|  |  |
| --- | --- |
| 近距离后方危险检测预警(CRCW) |  |
| 输入参考量 | 两车行驶方向 ，相对速度，相对距离，相对位置信息，本车速度，邻车速度 |
| 触发条件 | 1. 本车车速大于**安全速度**(可设置)时，启动CRCW预警 2. 本车和邻车**同向**行驶 3. 本车位于邻车**前方** 4. 邻车车车速减去本车车速大于**相对安全速度**(可设置) 5. 两车距离小于**安全距离**(可设置) |
| 解除条件 | 1.上述任一触发条件不满足时 |
| 其他要求 |  |

注：

**安全距离计算公式1(验证使用)**： D=（2\*ra-rb）\* k，其中ra表示本车速度,rb表示邻车速度，单位m/s，k表示安全系数，默认取值为4，k可设置

**安全距离计算公式2(可选)**：

式中：Tr为驾驶员反应时间（s），一般为 0.5—2s；Td 为制动协调时间（s），一般为 0.2s；V 为自车速度（m/s）；

Vrel为两车相对速度（m/s）；通常取α =αf；

#### 处理流程

位置信息模块100ms会更新一次邻结点的位置相关信息，CCW模块同步进行筛选，满足判断条件的邻近车辆会被分别添加到CFCW和CRCW的报警邻列表，向系统模块发送报警消息。



图 9‑9近距离危险预警处理流程图

## 紧急制动危险（EEBL）模块设计

### 功能描述

紧急制动危险模块的功能主要是，当车辆进行紧急制动时，模块对本车状态进行检测，符合判断标准将对邻车进行紧急制动报警，同时模块还将处理收到的邻车紧急制动报警信息，即模块同时具有接受和发送的功能。

EEBL：

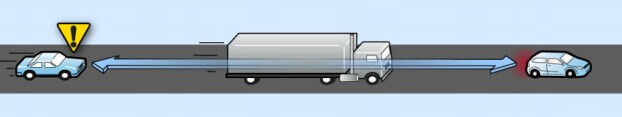


图 9‑10紧急制动预警示意图

### 接口设计

#### alarm\_update\_proc

|  |  |  |
| --- | --- | --- |
| 条目 | 内容 | |
| 原型定义 | static int alarm\_update\_proc(vsa\_envar\_t \*p\_vsa, void \*arg) | |
| 输入项 | p\_vsa | 转型为vsa\_envar\_t类型的指针 |
| arg | 空指针，传递消息 |
| 输出项 | 无 | |
| 返回值 | 返回err信息,err =1 消息未处理，err =0 消息已经处理 | |
| 功能描述 | 可以注册的回调函数，用于处理收到紧急报警消息 | |

#### 9.3.2.2 gsnr\_detect\_proc

|  |  |  |
| --- | --- | --- |
| 条目 | 内容 | |
| 原型定义 | static int gsnr\_detect\_proc(vsa\_envar\_t \*p\_vsa, void \*arg) | |
| 输入项 | p\_vsa | 转型为vsa\_envar\_t类型的指针 |
| arg | 空指针，传递消息 |
| 输出项 | 无 | |
| 返回值 | 返回err信息,err =1 消息未处理，err =0 消息已经处理 | |
| 功能描述 | 可以注册的回调函数，用于处理加速度模块发送的报警信息 | |

### 实现原理

#### 判断条件

本车接收报警信息：

|  |  |
| --- | --- |
| 前方车辆紧急制动预警(EEBL) |  |
| 输入参考量 | 两车行驶方向 ，相对速度，相对距离，相对位置信息，本车速度，邻车速度，邻车紧急制动报警消息(BSM) |
| 触发条件 | 1. 本车接收到邻车发送的紧急制动报警消息，启动EEBL检测 2. 本车车速大于**安全速度**(可设置) 3. 本车和邻车**同向**行驶 4. 本车位于邻车**后方** |
| 解除条件 | 1.上述任一触发条件不满足时 2.一定时间内(5S)不再接收到同一报警消息 |
| 其他要求 |  |

|  |  |
| --- | --- |
| 本车发出报警信息： |  |
| 输入参考量 | 本车加速度，本车速度 |
| 触发条件 | 1. 本车车速大于**安全速度** 2. 加速度为负（即减速）且绝对值大于0.3g（3m/s2） |
| 解除条件 | 1.上述任一触发条件不满足时 2.一定时间内(5S)不再接收到同一报警消息 |
| 其他要求 |  |

#### 处理流程

接受预警流程：

本车接收到邻车发送的EEBL报警消息，然后根据发送消息的邻车pid，从车辆位置信息数据链表的模块中提取所需要的位置信息，经过判断，如果满足报警条件，就向系统模块发送报警消息(**紧急消息**)。不满足，则终止此次判断。



图 9‑11紧急制动危险预警处理流程

发送预警流程：

安全应用模块，接收从VAM中间件发送来的内部紧急制动消息，结合本车自身信息，满足条件就向外广播紧急制动消息。不满足条件，则舍弃本次内部消息。

## 车辆故障危险（VBD）模块设计

### 功能描述

车辆故障危险模块的功能主要是，在前车出现故障，不能继续行驶的时候，向周围邻近车辆发送故障信息。接收到故障车辆信息后，本车经过模块判断决定是否发送报警信息。

### 接口设计

#### key\_update\_proc

|  |  |  |
| --- | --- | --- |
| 条目 | 内容 | |
| 原型定义 | static int key\_update\_proc(vsa\_envar\_t \*p\_vsa, void \*arg) | |
| 输入项 | p\_vsa | 转型为vsa\_envar\_t类型的指针 |
| arg | 空指针，指向报警消息 |
| 输出项 | 无 | |
| 返回值 | 无 | |
| 功能描述 | Vam中间件的回调函数，注册故障信息报警函数，发送报警信息 | |

### 实现原理

#### 判断条件

本车接收报警信息：

|  |  |
| --- | --- |
| 前方车辆故障预警 |  |
| 输入参考量 | 两车行驶方向 ，本车速度，相对位置信息，邻车车辆故障报警消息(BSM) |
| 触发条件 | 1. 本车接收到邻车发送的紧急制动报警消息，启动VBD检测 2. 本车车速大于**安全速度** 3. 本车**前向**行驶 4. 本车位于邻车**后方** |
| 解除条件 | 1.上述任一条件不满足 |
| 其他要求 |  |

本车发出故障报警信息：

|  |  |
| --- | --- |
| 本车发送车辆故障预警信息 |  |
| 输入参考量 | 无 |
| 触发条件 | 手动触发送故障报警信息 |
| 解除条件 | 手动解除 |
| 其他要求 |  |

#### 处理流程

接受报警处理流程：当本车接收到邻车发送来的车辆故障信息后，根据车辆的pid，去位置信息模块中获取相应的数据信息，判断是否符合报警条件，符合条件则发送报警消息(**消息种类为紧急消息**)至系统模块。不符合，则终止本次判断。



图 9‑12车辆故障报警流程图

触发报警：当本车车辆发生故障不能继续行驶时，使用相应开关，手动触发报警。

## 交叉路口碰撞预警模块(CICAS)

### 功能描述

车辆交叉路口碰撞预警系统包括V-V模块和路测站。通过路测设备，本车可以获取交叉路口其他道路的车辆信息，通过系统的计算判断和路径预估，可以对交叉路口的碰撞进行预警。通过路测设备发送给车辆模块的预警信息分为通知和预警两类。在交叉路口其他道路有车辆接近路口并且判断相撞几率小的情况下，通过人机设备发出通知信息。当判断很大几率会发生碰撞时，发出报警信息。

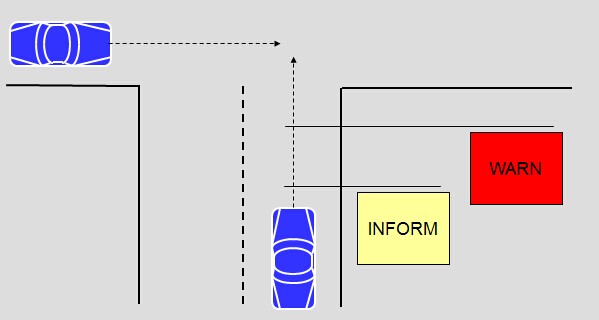


图 9‑13交叉路口碰撞危险示意图

使用路测设备应用于交叉路口的优势在于，可以解决交叉路口的障碍物遮挡产生的通讯中断问题，而且可以将大数据量的计算放置于路测设备中。

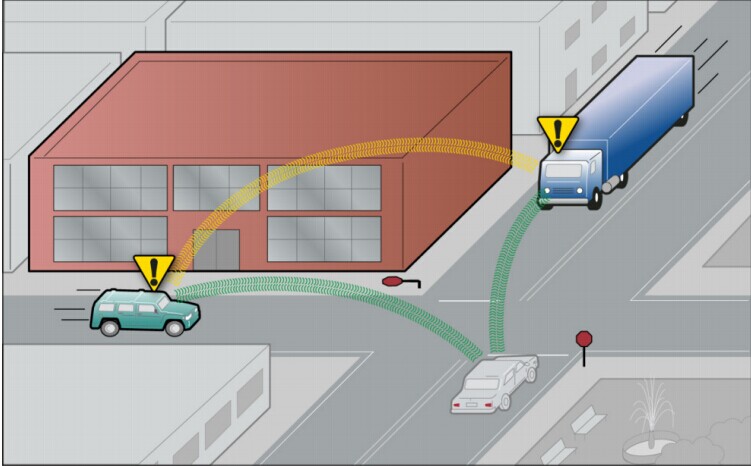


图 9‑14交叉路口碰撞示意图

# 系统部分

## 系统控制模块设计

## 人机交互模块设计

## 参数管理模块设计

## 固件更新模块设计 \*

## 生产测试模块设计 \*