密级：**普通 秘密 机密 绝密**

VANET路侧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 | 王磊 | 2015-7-30 |
|  |  |  |  |

目录

[1 引言 9](#_Toc426620402)

[1.1 编写目的 9](#_Toc426620403)

[1.2 适用范围 9](#_Toc426620404)

[1.3 术语定义 9](#_Toc426620405)

[1.4 参考资料 9](#_Toc426620406)

[2 任务概述 10](#_Toc426620407)

[2.1 系统概述 10](#_Toc426620408)

[2.2 设计需求规定 11](#_Toc426620409)

[2.3 运行环境规定 12](#_Toc426620410)

[3 总体设计 12](#_Toc426620411)

[3.1 设计综述 12](#_Toc426620412)

[3.1.1 软件概况 13](#_Toc426620413)

[3.1.2 软件设计规范 15](#_Toc426620414)

[3.2 软件总体架构 18](#_Toc426620415)

[3.3 软件流程描述 20](#_Toc426620416)

[3.3.1 数据流处理流程 20](#_Toc426620417)

[3.4 数据结构设计 22](#_Toc426620418)

[3.5 出错处理设计 22](#_Toc426620419)

[3.6 软件设计要点 22](#_Toc426620420)

[4 接口设计 23](#_Toc426620421)

[4.1 用户接口 23](#_Toc426620422)

[4.1.1 CLI 23](#_Toc426620423)

[4.1.2 Web 23](#_Toc426620424)

[4.2 外部接口 23](#_Toc426620425)

[5 接口及外设驱动部分 23](#_Toc426620426)

[5.1 无线通信驱动模块设计 23](#_Toc426620427)

[5.1.1 功能描述 23](#_Toc426620428)

[5.1.2 接口设计 24](#_Toc426620429)

[6 无线网络通信部分 28](#_Toc426620430)

[6.1 收发帧处理模块设计 28](#_Toc426620431)

[6.1.1 功能描述 28](#_Toc426620432)

[6.1.2 接口设计 28](#_Toc426620433)

[6.1.3 实现原理 34](#_Toc426620434)

[6.2 LLC协议模块设计 38](#_Toc426620435)

[6.2.1 功能描述 38](#_Toc426620436)

[6.2.2 接口设计 38](#_Toc426620437)

[6.2.3 实现原理 40](#_Toc426620438)

[6.3 DSMP协议模块设计 41](#_Toc426620439)

[6.3.1 功能描述 41](#_Toc426620440)

[6.3.2 接口设计 46](#_Toc426620441)

[6.3.3 加密的DSM结构 48](#_Toc426620442)

[6.3.4 实现原理 48](#_Toc426620443)

[6.4 DSA协议模块设计 \* 48](#_Toc426620444)

[6.4.1 功能描述 48](#_Toc426620445)

[6.4.2 接口设计 52](#_Toc426620446)

[6.4.3 实现原理： 53](#_Toc426620447)

[7 VANET中间件部分 53](#_Toc426620448)

[7.1 应用编程接口（API）模块设计 53](#_Toc426620449)

[7.2 车辆状态管理（VSM）模块设计 61](#_Toc426620450)

[7.3 车辆本地信息处理（LIP）模块设计 61](#_Toc426620451)

[7.4 DSRC消息协议（RCP）模块设计 61](#_Toc426620452)

[7.4.1 协议功能分析 62](#_Toc426620453)

[7.4.2 消息格式 63](#_Toc426620454)

[7.4.3 接口设计 78](#_Toc426620455)

[8 车辆安全应用部分 79](#_Toc426620456)

[8.1 交叉路口碰撞预警模块(CICAS) 79](#_Toc426620457)

[8.1.1 功能描述 79](#_Toc426620458)

[9 系统部分 81](#_Toc426620459)

[9.1 系统控制模块设计 81](#_Toc426620460)

[9.2 OAM模块设计 81](#_Toc426620461)

[9.3 参数管理模块设计 81](#_Toc426620462)

[9.4 固件更新模块设计 \* 81](#_Toc426620463)

[9.5 生产测试模块设计 \* 81](#_Toc426620464)

图目录

[图 2‑1 VANET系统拓扑图 12](#_Toc407723175)

[图 3‑1固件功能框图 21](#_Toc407723176)

[图 3‑2 目录结构 22](#_Toc407723177)

[图 3‑3 软件总体架构 26](#_Toc407723178)

[图 3‑4 功能实现流程 30](#_Toc407723179)

[图 3‑5 网络接收事件处理流程 31](#_Toc407723180)

[图 3‑6 本地车辆状态改变处理流程 32](#_Toc407723181)

[图 3‑7 用户事件处理流程 33](#_Toc407723182)

[图 3‑8 数据流处理流程 34](#_Toc407723183)

[图 3‑9 传输帧处理示意图 35](#_Toc407723184)

[图 3‑10 系统数据结构 36](#_Toc407723185)

[图 4‑1 基本界面 37](#_Toc407723186)

[图 4‑2 预警提示界面 37](#_Toc407723187)

[图 4‑3 预警提示类别 38](#_Toc407723188)

[图 4‑4 系统内部接口关系 39](#_Toc407723189)

[图 6‑1 无线MAC驱动模块接口关系 47](#_Toc407723190)

[图 6‑2 无线MAC驱动模块内部结构图 48](#_Toc407723191)

[图 6‑3 MAC帧类型1格式定义 49](#_Toc407723192)

[图 6‑4 MAC帧类型2格式定义 50](#_Toc407723193)

[图 6‑5 MAC帧格式定义 51](#_Toc407723194)

[图 6‑6 RT3070传输帧格式 52](#_Toc407723195)

[图 7‑1 收发帧处理模块接口关系 54](#_Toc407723196)

[图 7‑2 收发帧处理模块内部结构图 59](#_Toc407723197)

[图 7‑3 收发帧处理模块数据结构 59](#_Toc407723198)

[图 7‑4 上行数据流处理流程 62](#_Toc407723199)

[图 7‑5 下行数据流处理流程 63](#_Toc407723200)

[图 7‑6 LLC协议模块接口关系 64](#_Toc407723201)

[图 7‑7 LLC帧格式 66](#_Toc407723202)

[图 7‑8 DSMP封装的帧格式 67](#_Toc407723203)

[图 7‑9 DSMP/DSA扩展域格式定义 68](#_Toc407723204)

[图 7‑10 DSMP的数据流程 71](#_Toc407723205)

[图 7‑11 DSMP接口关系图 71](#_Toc407723206)

[图 7‑12 1609.2帧格式 73](#_Toc407723207)

[图 7‑13 VSIE帧承载WSA的定义 74](#_Toc407723208)

[图 7‑14 1609.2承载DSA的帧定义 74](#_Toc407723209)

[图 7‑15 DSA帧格式定义 75](#_Toc407723210)

[图 7‑16 Service setup & WSM delivery 76](#_Toc407723211)

[图 7‑17 DSA服务通道建立及数据交换流程 77](#_Toc407723212)

[图 7‑18 DSA接口关系图 77](#_Toc407723213)

[图 8‑1 MsgID定义 88](#_Toc407723214)

[图 8‑2 BSM消息帧格式 89](#_Toc407723215)

[图 8‑3 BrakeSystemStatus 定义 90](#_Toc407723216)

[图 8‑4 EventFlags 定义 91](#_Toc407723217)

[图 8‑5 BSM消息处理流程 92](#_Toc407723218)

[图 8‑6 EVAM消息帧格式 94](#_Toc407723219)

[图 8‑7 EVAM消息处理流程 96](#_Toc407723220)

[图 8‑8 EVA消息格式 97](#_Toc407723221)

[图 8‑9 EVAM消息处理流程 98](#_Toc407723222)

[图 8‑10 PVD消息格式 100](#_Toc407723223)

[图 8‑11 snapshot的发送流程图 101](#_Toc407723224)

[图 8‑12 PDM消息格式 102](#_Toc407723225)

[图 8‑13 rcp接口 104](#_Toc407723226)

[图 8‑14安全应用场景之一 106](#_Toc407723227)

[图 8‑15大量节点转发示意图 106](#_Toc407723228)

[图 8‑16数据包广播转发示意图 110](#_Toc407723229)

[图 8‑17 路由算法流程图 111](#_Toc407723230)

[图 8‑18 多跳转发模块接口关系 112](#_Toc407723231)

[图 8‑19 多跳转发模块内部结构图 113](#_Toc407723232)

[图 8‑20 多跳转发模块数据结构 114](#_Toc407723233)

[图 8‑21 多跳转发处理流程 115](#_Toc407723234)

[图 8‑22 超时处理流程 116](#_Toc407723235)

[图 9‑1车辆安全应用的软件架构层次 116](#_Toc407723236)

[图 9‑2车辆安全应用危险场景软件处理结构 118](#_Toc407723237)

[图 9‑3车辆相对位置分类 119](#_Toc407723238)

[图 9‑4位置预处理模块示意图 120](#_Toc407723239)

[图 9‑5位置数据结构组成 120](#_Toc407723240)

[图 9‑6 VSA部分数据结构 121](#_Toc407723241)

[图 9‑7前车近距离碰撞预警示意图 121](#_Toc407723242)

[图 9‑8后车近距离碰撞预警示意图 122](#_Toc407723243)

[图 9‑9近距离危险预警处理流程图 124](#_Toc407723244)

[图 9‑10紧急制动预警示意图 125](#_Toc407723245)

[图 9‑11紧急制动危险预警处理流程 127](#_Toc407723246)

[图 9‑12车辆故障报警流程图 129](#_Toc407723247)

[图 9‑13交叉路口碰撞危险示意图 130](#_Toc407723248)

[图 9‑14交叉路口碰撞示意图 130](#_Toc407723249)

表目录

[表 2‑1 固件设计需求 14](#_Toc426616766)

[表 3‑1 模块缩写 19](#_Toc426616767)

[表 3‑2 系统线程分配 23](#_Toc426616768)

[表 4‑1 系统消息定义 29](#_Toc426616769)

[表 4‑2 VANET中间件接口—上层应用 29](#_Toc426616770)

[表 4‑3 VANET中间件接口—获取车辆状态信息 30](#_Toc426616771)

[表 4‑4 VANET中间件接口—更新本地车辆状态 30](#_Toc426616772)

[表 4‑5 VANET中间件接口—应用层协议接口 31](#_Toc426616773)

[表 4‑6 网络传输层接口 31](#_Toc426616774)

[表 4‑7 无线网络MAC层接口 31](#_Toc426616775)

[表 4‑8 数据加解密接口 32](#_Toc426616776)

[表 4‑9 外设驱动接口函数 32](#_Toc426616777)

[表 4‑10 外设驱动接口事件 32](#_Toc426616778)

[表 5‑1 打印级别定义 33](#_Toc426616779)

[表 5‑2 打印控制接口 33](#_Toc426616780)

[表 5‑3 打印输出接口 35](#_Toc426616781)

[表 6‑1 drv\_wifi\_send原型定义 36](#_Toc426616782)

[表 6‑2 drv\_wifi\_mac\_header\_len原型定义 36](#_Toc426616783)

[表 6‑3 MAC帧类型1定制字段定义 38](#_Toc426616784)

[表 6‑4 MAC帧类型2定制字段定义 39](#_Toc426616785)

[表 6‑5 MAC帧定制字段定义 39](#_Toc426616786)

[表 7‑1 wnet\_get\_txbuf原型定义 42](#_Toc426616787)

[表 7‑2 wnet\_release\_rxbuf原型定义 42](#_Toc426616788)

[表 7‑3 WNET\_TXBUF\_DATA\_PTR原型定义 43](#_Toc426616789)

[表 7‑4 WNET\_TXBUF\_INFO\_PTR原型定义 43](#_Toc426616790)

[表 7‑5 WNET\_RXBUF\_PTR原型定义 44](#_Toc426616791)

[表 7‑6 wnet\_send原型定义 44](#_Toc426616792)

[表 7‑7 wnet\_recv原型定义 44](#_Toc426616793)

[表 7‑8 wnet\_send\_complete原型定义 45](#_Toc426616794)

[表 7‑9 fp\_send原型定义 45](#_Toc426616795)

[表 7‑10 llc\_send原型定义 52](#_Toc426616796)

[表 7‑11 llc\_recv原型定义 52](#_Toc426616797)

[表 7‑12 LLC帧EtherType值 54](#_Toc426616798)

[表 7‑13 DSMP帧头AID定义 55](#_Toc426616799)

[表 8‑1 mda\_handle原型定义 100](#_Toc426616800)

[表 8‑2 rcp\_create\_forward\_msg原型定义 101](#_Toc426616801)

[表 8‑3 rcp\_send\_forward\_msg原型定义 101](#_Toc426616802)

# 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

OAM ：操作、管理和维护（ Operation Administration and Maintenance）

## 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BTS终端。

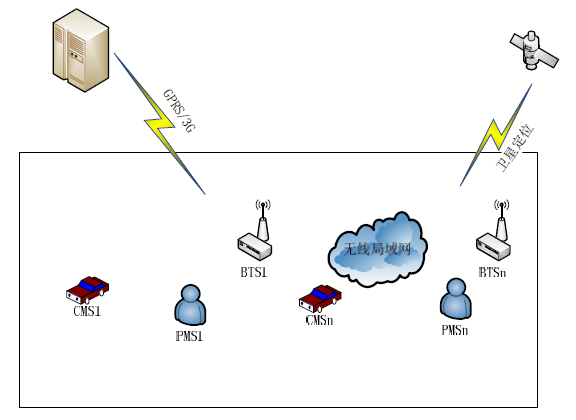


图 2‑1 VANET系统拓扑图

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系统中的路侧终端BTS产品，主要功能特点如下：

* 支持VANET车辆自组织网络，
* 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
  + 弯道/交叉口会车危险提示）
  + 特殊道路（急弯/隧道/桥梁/十字路口/...）提示
  + 电子交通标识（信号灯/限速/...）提示

## 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 |
|  |
| 驱动 | 接口   * uart/spi/pcie等linux内核层完成。 |
| 设备驱动：   * WiFi网卡收发、管理 * LTE芯片收发、管理 |
| 无线网络 | 支持自组网 |
| 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  MSG\_RoadsideAlert: 路侧警告消息   * 消息编解码 |
| 支持本地信息处理，包括：   * GPS（NMEA）消息解析，获取路侧设备位置和标准时间信息   ？ 导入地图信息，包含本地GPS。 |
| 支持车辆信息管理，支持如下：   * 邻车状态维护 |
| 上层应用接口，包括：   * 读取邻车信息接口 * 发送报警消息接口 * 邻车状态更新事件处理回调 |
| 安全应用 | 交通标识预警提示  发出的交通标识预警消息。 |
| 会车危险预警提示  根据邻居车辆状态，判断会车危险，发出会车危险预警消息。 |
| 特殊道路预警提示  路侧设备发出特殊道路危险预警消息。 |
|  |
| 系统管理 | 工作模式管理 |
| 配置参数管理 |
| OAM（CLI、Web） |
|  |
| 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

### 软件概况

#### 功能划分



图 3‑1固件功能框图

系统目前有两个进程，分别为rsu，cli\_agent；

rsu进程参照车载终端软件架构，从功能上可以划分为下面的几大部分：

* 内核：

操作系统等。

* 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，包括危险检测与告警、道路提示消息功能。

* 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、参数管理、OAM管理等。

cli\_agent进程： 实现命令行管理控制，与rsu通过socket通信，通信协议参照《V2V模块设计》的管理配置帧格式。

#### 开发环境

* 编译环境：arm-linux-gcc
* 操作系统：linux

#### 目录结构

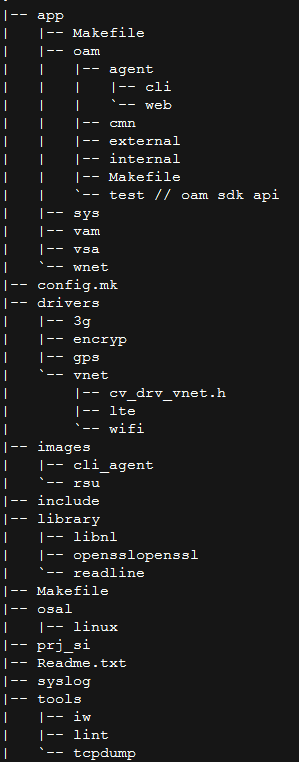


图 3‑2 目录结构

* Rsu：

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、系统及管理模块等。

* oam：

agent：管理控制客户端代码，是独立的进程，包括cli进程，和后续要开发的web进程。

* images：

应用程序镜像文件子目录，用于存放生成的应用程序文件和相关启动脚本。

* 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收发驱动等。

这部分驱动指的是用户态设备驱动，接口驱动linux内核态完成。

* drivers/vnet：

分两部分：WiFi、LTE等应用层驱动。

* osal：

操作系统及公共功能子目录，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。

* 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.a 和.so文件。

* prj\_si：

程序项目文件子目录，用于存放source insight工程文件等。

* 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、代码走查。

### 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 |  | osal | OS Adapter Layer |
|  | 动态内存管理 | osal\_mem\* |  |
| 调试模块 | dbg |  |
|  |  |  |
| 接口及外设驱动部分 |  | drv |  |
|  | 串口 | uart |  |
| GPS | gps |  |
| 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 |
| 本地信息处理 | lip | Local Information Process |
| 车辆安全应用部分 |  | vsa | Vehicle Safty Application |
|  | 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 |  |
| 管理 | oam |  |
| 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，系统软件在用户态实现，内核态不描述。从软件功能上主要分为以下六个部分：操作系统适配层（包括Netlink库、shell库等）、外设驱动、无线网络通信、VANET中间件、车辆安全应用、系统控制、OAM管理模块。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：操作系统适配层（OSAL）。

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。

* 外设驱动

用户态驱动，基于Linux I/O系统调用实现外设的访问控制，及功能逻辑。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、硬件接口无关。本系统需要实现的设备驱动包括：WiFi、GPS？。

* 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、本地道路信息处理共四部分。本部分向下与网络通信部分以及GPS、文件操作等驱动接口，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。

本地信息处理：用来实现VANET通信中道路信息的加载和处理等。

* 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三个应用，其中：会车危险提示、特殊道路提示和交通标识提示功能，需与车载终端配合。RSU需实现能发送相应消息到车载终端。

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、OAM管理模块。

OAM模块与CLI、web管理进程交互，提供统一的管理配置接口供CLI、Web等管理方式调用。

CLI：方案一：rtthread的shell命令行；方案二：基于zebra类Cisco命令行。

Web： thttpd、Goahead

管理端为多进程，通过socket通信。

功能模块为多线程，采用POSIX线程接口，消息队列使用POSIX Queue

主要线程：vam、sys、cli、wnet、vsa线程等；

## 软件流程描述

### 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。

注：

1、使用POSIX队列不需要维护发送接收buffer

2、发送接收都是用户态socket操作，硬件收发linux内核网卡驱动完成不再描述



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

## 出错处理设计

无。

## 软件设计要点

本软件主要实现了基于VANET的路侧终端功能，其中在软件设计中需要重点关注的是达到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的独立性，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。同时兼容不同的通信方式（WIFI、LTE等）。

# 接口设计

## 用户接口

本产品提供两种形式的用户接口：CLI（支持telnet）、Web。。

### CLI

通过CLI完成参数配置管理、设备。CLI作为外部管理控制进程与rsu的OAM模块进行socket通信，通信帧格式如下：



* FrameType: 0x0002
* MsgType定义如下：

|  |  |  |
| --- | --- | --- |
| MsgType | 描述 | 备注 |
| 0x0001 | 命令请求 |  |
| 0x0002 | 命令回复 |  |

* MsgData结构设计：

|  |  |  |
| --- | --- | --- |
| MsgData | 描述 | 备注 |
| ret | 命令执行结果 |  |
| req\_id | 命令请求计数的id |  |
| reservd | 保留字段（备用） |  |
| sdk\_id | 管理sdk的索引 |  |
| len | 管理参数数据的长度 |  |
| data | 管理参数值 |  |

typedef struct

{

uint32\_t ret;

uint32\_t req\_id;

uint32\_t reservd;

uint16\_t sdk\_id;

uint16\_t len;

uint8\_t data[1];

} \_\_attribute\_\_((packed))cv\_sdk\_msg\_t;

### Web

## 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。

# 接口及外设驱动部分

## 无线通信驱动模块设计

。

### 功能描述

本模块主要负责实现基于WiFi的无线通信功能，包括网卡驱动及MAC层的实现。本系统采用的WiFI网卡ath9005，内核态底层驱动采用compat开源代码，编译成内核模块作为内核态底层驱动。本模块主要实现用户态无线帧的收发，封装与解析等功能

用户态实现方案：网卡使用Monitor模式， 基于Libnl Netlink库，通过NL80211接口，实现报文的收发，网卡的配置管理。

同时，向上提供设备无关的收发帧接口wnet\_send，wnet\_recv。

### 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

#### 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 |

# 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

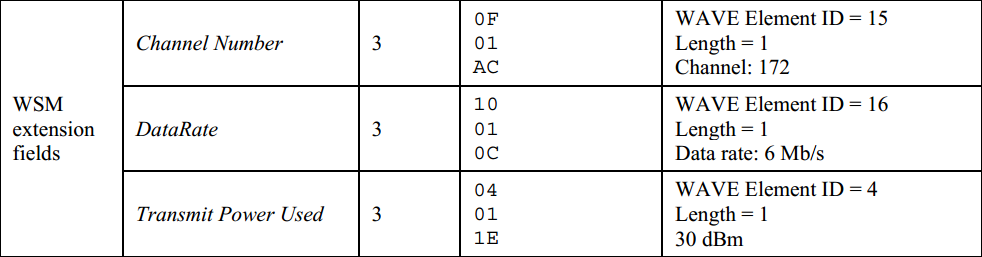
| **AID** | **Service** |
| --- | --- |
| 0 | system |
| 1 | automatic-fee-collection |
| 2 | freight-fleet-management |
| 3 | public-transport |
| 4 | traffic-traveler-information |
| 5 | traffic-control |
| 6 | parking-management |
| 7 | geographic-road-database |
| 8 | medium-range-preinformation |
| 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 7‑9 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 8‑1 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***

***-- 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

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" \l "ASN_SnapshotDistance" \o "To allow Network Users to change the snapshot collection policy based on speed and distance.  Two di...) -- 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

# 车辆安全应用部分

## 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。

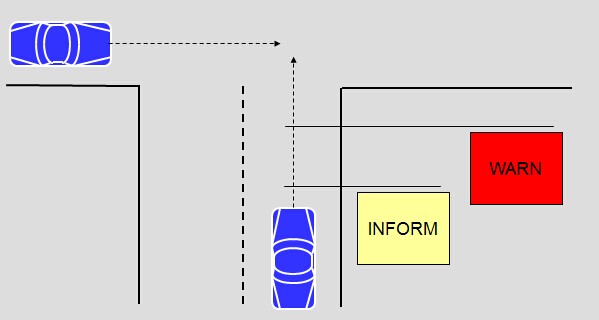


图 9‑13交叉路口碰撞危险示意图

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。

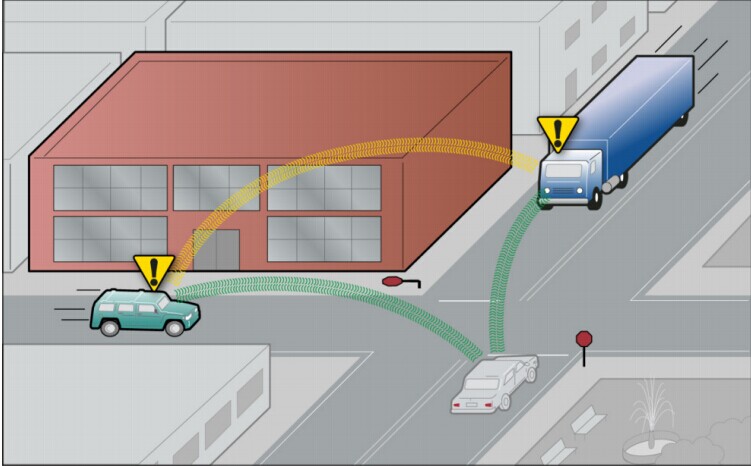


图 9‑14交叉路口碰撞示意图

# 系统部分

## 系统控制模块设计

## OAM模块设计

CLI

Web

## 参数管理模块设计

## 固件更新模块设计 \*

## 生产测试模块设计 \*