第34卷 第24期 计 算 机 工 程 2008年12月
Vol.34 No.24 Computer Engineering December 2008
·网络与通信· 文章编号：1000—3428(2008)24—0100—03 文献标识码：A 中图分类号：TP336

基于IPv6的CAN应用层协议实现

董劲男[1]，秦贵和[1,2]，于 赫[1]，苗丽颖[3]，刘文静[1]
(1. 吉林大学计算机科学与技术学院，长春 130022；2. 吉林大学汽车动态模拟国家重点实验室，长春 130022；
3. 吉林大学汽车工程学院，长春 130025)

摘 要：作为车载网络与移动通信网络互联的相关软硬件产品的核心技术，车载网络与IPv6网络的网关设计是互联互通的基础，也是开发的难点。该文论述一种支持IPv6网络连接的控制器局域网络(CAN)应用层协议的设计过程，采用命令故障码表作为IPv6协议和CAN协议应用层级功能的映射工具，描述网关的软件处理流程。在车身开发中进行了具体实现，验证了网关的有效性和正确性。
关键词：IPv6协议；控制器局域网络；网关；应用层协议

Implementation of IPv6-based CAN Application Layer Protocol

DONG Jin-nan[1], QIN Gui-he[1,2], YU He[1], MIAO Li-ying[3], LIU Wen-jing[1]
(1. College of Computer Science and Technology, Jilin University, Changchun 130022; 2. State Key Laboratory of Automobile Dynamic Simulation, Jilin University, Changchun 130022; 3. School of Automobile Engineering, Jilin University, Changchun 130025)

【Abstract】As a key technique of software and hardware development related with interconnection between Universal Vehicle Network(UVN) and IPv6, the gateway between UVN and IPv6 is the basis of interconnection and the hard point of development. This paper presents the design process of a kind of Controller Area Network(CAN) application layer protocol which supports the connection with IPv6 backbone. As a structure of mapping the function between the IPv6 protocol and CAN application layer protocol, the Order/Fault Code Table(OFCT) is constructed and the software operation flow of gateway is described. The effectiveness and accuracy of this CAN application layer protocol are validated in the self-development automotive body network.
【Key words】IPv6 protocol; Controller Area Network(CAN); gateway; application layer protocol

1 概述

IPv6是国际公认的开放网络通信标准。与IPv4相比，IPv6具有地址空间更大、网络整体吞吐量更高、服务质量和多播功能更好、安全性更强、即插即用和移动应用更方便等诸多优点[1]。IPv6是互联网产业化的必然要求，尤其是IPv6大大扩展了地址空间，恢复了原来因地址受限而推动的端到端连接功能，保证了端到端的服务质量和安全性，为互联网的进一步发展和缩小数字鸿沟提供了基本条件[2]。

目前在国际上，车载网络等嵌入式网络出现了快速发展的势头，其中，控制器局域网络(Controller Area Network, CAN)应用最广泛[3]。CAN属于总线式串行通信网络，由于采用了许多新技术及独特的设计，因此其数据通信具有突出的可靠性、实时性和灵活性[4]。

IPv6的发展对嵌入式网络产生了巨大的影响，尤其是车载网络[5]。各种新型汽车服务的出现使数据流量呈爆发性增长，随之产生对IP地址的大量需求。IPv6协议提供的海量地址空间使每个嵌入式节点都能分配独立的地址，使端到端的通信成为可能，大大减少路由带来的成本开销，同时通信质量和安全性都能得到极大的保障。车载网络的普及、IPv6主干网的开通以及巨大的市场需求必然带来车载网络与移动通信网络互联的相关软硬件产品的开发热潮，从而促进汽车电子产业的发展。

作为不同类型网络连接的核心技术，网关承担了协议间的转换功能。目前对IPv6与CAN协议的转换研究有2种思路：(1)进行完全的协议转换，在CAN协议中加入缺失的协议层；(2)直接将CAN帧作为IPv6帧的数据段。前一种方法由于CAN协议只采用了ISO/OSI模型中的物理层、数据链路层与应用层[6]，如果加入缺失的协议层，会增加协议的处理时间，影响协议的实时性，这与现场总线协议的设计出发点是背离的。后一种方法将处理工作都放到服务器端，底端节点只是被动接受指令，大大降低了系统灵活性，不利于系统功能的扩展。基于以上分析，本系统采用自主设计的应用层协议实现IPv6协议与CAN协议的转换，并最大程度地保证了系统的可扩展性。

2 系统总体结构

IPv6协议满足开放系统互连模型(OSI)7层结构，而CAN协议只采用了ISO/OSI模型中的物理层、数据链路层与应用层。因此，CAN通信接口设计需要在CAN协议中自定义应用层协议，实现2种协议的转换。笔者设计的基于IPv6的车身网络控制系统的结构以及网关所在位置和功能见图1。系统流程如下：客户端所发出的控制或检测命令经由有线或者无线的IPv6网络传送至服务器端，服务器根据命令故障码表的映射关系，将客户端指令转换为标准的网络通信数据包，

基金项目：国家发改委下一代互联网示范工程2005年研究开发产业化及应用试验基金资助项目“基于IPv6的车载信息系统研究与产业化”(3H006l421421)
作者简介：董劲男(1980－)，男，博士研究生，主研方向：嵌入式与实时系统；秦贵和，教授、博士生导师；于 赫，硕士研究生；苗丽颖，助理工程师、硕士；刘文静，硕士研究生
收稿日期：2008-06-25 **E-mail：**magicalstick@163.com

再经由无线网络传送至车内网关。网关接收遵循 TCP/IP 协议的数据包，提取其数据，并根据命令类型的不同，由命令故障码表的映射关系将数据分别转成相应的总线网络数据格式，发送至底层网络的相应节点 ECU。然后由底层的车内网络节点控制各个执行机构，完成客户端所要求的各种动作，并将结果反馈给客户端。

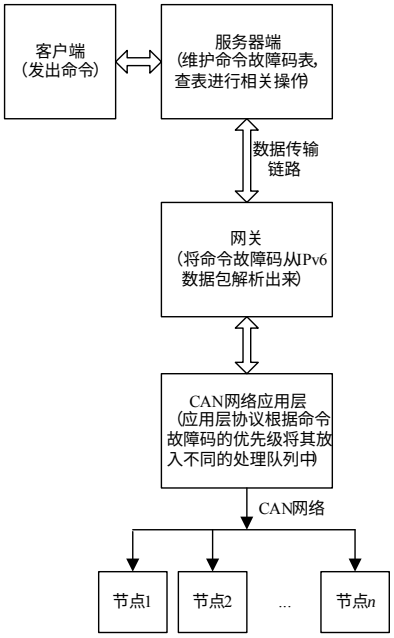

图 1　基于 IPv6 的车身网络控制系统结构

网关的主要作用是在不同架构、不同电气特性网络之间转发数据(本系统中主要是 IPv6 数据与 CAN 网络数据之间的转发)。协议设计的主要工作是在 Linux 系统之上开发 CAN 网络数据的收发程序、IPv6 网络数据的收发程序和网关程序。

3　协议设计

为了简化查询命令故障码表的算法，将命令故障码表的数据结构定义为二维数组，并且预留了未使用的故障码，以便今后系统功能扩展，在协议转换时通过查表直接使用传输的数据(即 CAN 帧的 ID 和数据段)作为码表的数组下标。

命令故障码表数据结构如图 2 所示，其中，第 1 部分为系统控制；第 2 部分为车窗控制；第 3 部分为车灯控制；第 4 部分为车门控制；第 5 部分为实时数据采集部分。

命令	0x00	0x01	0x02	0x03	0x04
0x00 ↓ 0x09	开机并建立连接 …	关机 …	重启 …	系统更新 …	错误 重发 …
0x0A ↓ 0x0F	前左车窗升 …	前左车窗降 …	命令 检测 …		
0x10 ↓ 0x1F	刹车灯开 …	刹车灯关 …	命令 检测 …	ECU 失效 …	线路 故障 …
0x20 ↓ 0x2F	前左车门开 …	前左车门关 …	命令 检测 …	ECU 失效 …	线路 故障 …
0x40 ↓ 0x4F	测试车速 …	测试水温 …			

图 2　命令故障码表数据结构

CAN2.0B 协议给出了标准帧和扩展帧 2 种报文格式，两者的主要区别在于报文所含标识符的位数不同，标准帧包含 11 位标识符，扩展帧包含 29 位标识符。考虑到系统的可扩展性和与其他系统的兼容性，本系统使用扩展帧格式。CAN 收发线程 can_thread()调用接口函数 write()发送 6 Byte 的数据。网关分析接到的前面 2 Byte，确定节点(本系统中为车灯)的种类和开关命令，每种车灯定义了一种 CAN 帧，即一种灯对应一种帧 ID(底层定义 CAN 网中使用扩展帧 ID.20-ID.13)；控制命令则放入 CAN 帧中的数据段部分(规定数据段长度为 2 Byte，这里只使用了 1 Byte)。

4　协议实现

4.1　CAN 线程

网关首先对接收到的 IPv6 数据包进行解析，提取出数据段，并将其送入 can_in 队列中。

CAN 线程 can_thread()函数的主要功能是实现网关到 CAN 网络和 CAN 网络到网关的双向数据传输。CAN 线程维护 2 个队列：一个发送队列和一个接收队列。CAN 线程从 can_in 队列里取得数据并解析，然后打成 CAN 帧发送到 CAN 网络上；CAN 线程收到从底层网络发来的 2 Byte 的命令故障码，然后填充成 carbuff 结构写进接收队列 can_out 中。其主要工作流程如图 3 所示。

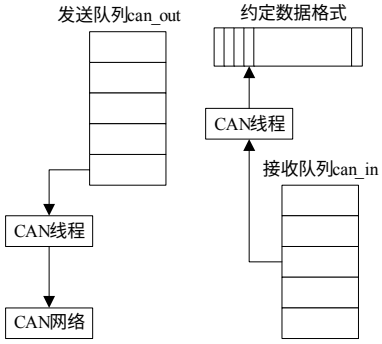

图 3　CAN 线程工作流程

与 can_thread 相关的程序接口有 CAN 驱动程序 mcp2510.c 和 carbuff 队列接口程序 carbuff.c。

CAN驱动程序mcp2510.c中定义了CAN网络的读写接口：

```
write(struct file *file, char *buf, size_t count, loff_t *offset)
read(struct file *file, char *buf, size_t count, loff_t *offset)
open(struct inode *inode, struct file *file)
close(struct inode *inode, struct file *file)
```

carbuff 队列接口程序中使用以下接口调用：

```
//检查 can_in 队列
carbuff_big_check(struct carbuff_big_head * list)
//从队列中读取一个 carbuff 数据
carbuff_big_read(struct carbuff_big_head *list)
//释放一个 carbuff 数据
carbuff_del(struct carbuff * buff)
//向 can_out 队列尾部添加
carbuff_big_add(struct carbuff_big_head * list, struct carinfo * buff)
```

carbuff 结构定义(carbuff.h)

```
struct carinfo
{
   unsigned char     type;    //sign car bus 0:can
   unsigned short    len;     //length of data，16 bytes
   char *            data;
};
```

```
struct carbuff
{
  struct carbuff *        prev;
  struct carbuff *        next;
  struct carbuff_head *   list;
  struct carinfo *        data;};
```

其中，carinfo->data 数据遵台下的约定格式：

信息头 0x7F	保留位 0x0000	命令故障码	信息长度	信息内容	校验和	结束标志 0xFF00
1 Byte	2 Byte	2 Byte	1 Byte	≤255 Byte	1 Byte	1 Byte

并将其定义其为 can_buff 结构：

```
typedef struct
{
  unsigned char head;
  unsigned short reserved;
  unsigned short ins_err;
  unsigned char info_len;
  unsigned char* info;
  unsigned char check;
  unsigned short info_end;
}can_buff;
```

接收的 data 中信息长度项 info_len 为 0，信息内容项 info 为 NULL，总长 16 Byte。下层网络的反馈信息放在命令故障码 ins_err 项中(can_buff 只在向网关返回反馈信息时使用)。

4.2 网关程序流程

网关的处理可分为 3 个步骤。

步骤 1 从 can_in 队列取数据，初始化及校验，见图 4。

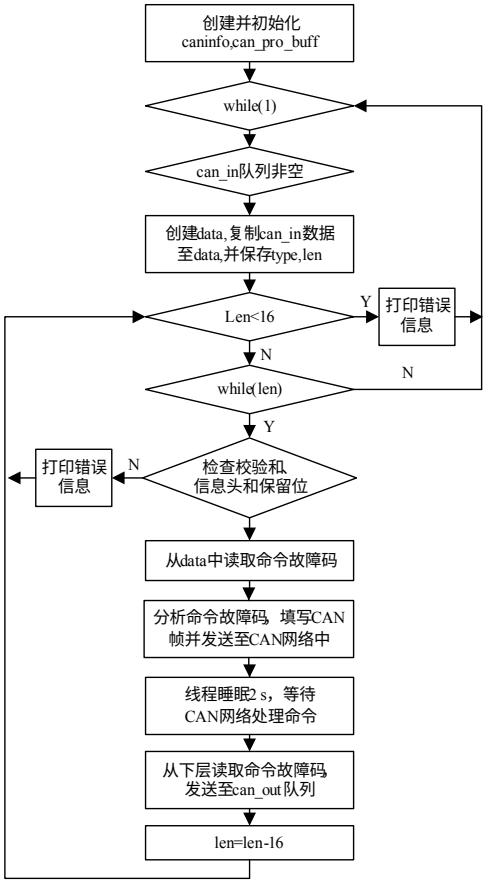

图 4 步骤 1 具体流程

步骤 2 分析命令故障码，填写 CAN 帧并发送至 CAN 网络中，见图 5。

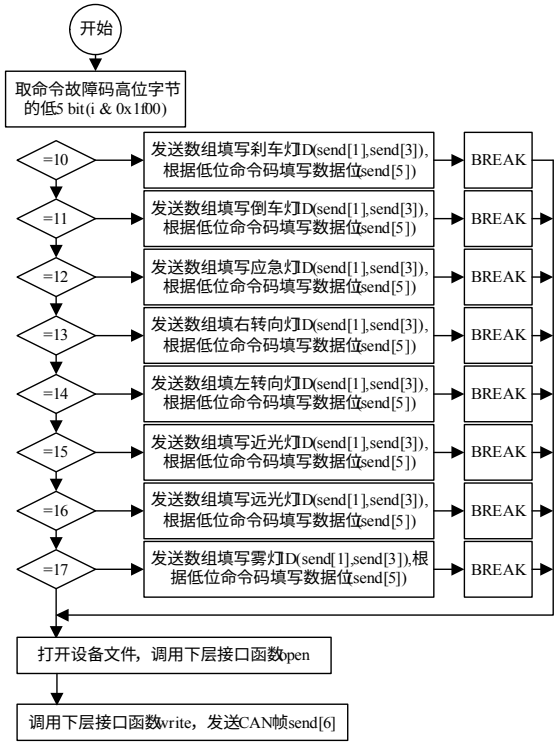

图 5 步骤 2 具体流程

步骤 3 从下层读取命令故障码，发送至 can_out 队列，见图 6。

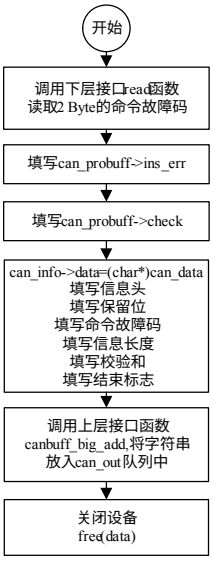

图 6 步骤 3 具体流程

4.3 硬件实现

网关的硬件结构如图 7 所示。

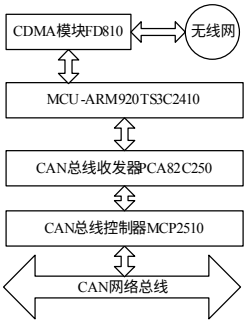

图 7 网关的硬件结构 (下转第 105 页)