文章编号:1001 - 9081(2008)06 - 1490 - 04

DSRC 协议 ASN. 1 模块的设计与实现

唐 波,王 能
(华东师范大学 信息科学技术学院,上海 200241)
(tbwises@163.com)

摘 要:通过研究专用短程通信(DSRC)协议栈中的应用层部分,针对其中的 ASN. 1 PER 编解码模块,设计并实现了一种编解码框架,框架包括 ASN. 1 映射规则、通用编解码流程以及符合小设备应用的内存管理机制。良好的模块划分使得此框架具有较好的通用性。测试结果表明,此实现框架具有良好的性能,满足协议的实时性要求。

关键词:专用短程通信; ASN. 1; 紧缩编码规则; 内存管理
中图分类号: TP393.04 **文献标志码**:A

Design and implementation of ASN. 1 module in DSRC

TANG Bo, WANG Neng
(*Department of Computer Science and Technology, East China Normal University, Shanghai* 200241, *china*)

Abstract: Based on the study of the application layer of Dedicated Short Range Communication (DSRC) protocol stack, as for the ASN. 1 PER encoding and decoding module, a framework for encoding and decoding was designed and implemented, which included ASN. 1 mapping rules, a general procedure for encoding and decoding, and a memory management method specifically designed for small devices. The framework is of good generality because of well-organized modules. The test results show that the framework has good performance, and it meets the real-time requirement of the protocol.

Key words: Dedicated Short Range Communication (DSRC); Abstract Syntax Notation One (ASN. 1); Packed Encoding Rules (PER); memory management

0 引言

专用短程通信(Dedicated Short Range Communication, DSRC)目前主要应用于智能交通,如:不停车收费、道路交通流量控制、停车位控制等,现有中国、欧洲、日本、美国等标准,中国标准于 2007 年 5 月颁布,本文适用于当前所有 DSRC 标准。一般情况下,DSRC 据 OSI 模型分为物理层(L1)、链路层(L2)、应用层(L7)等。由于 DSRC 应用存在多样性,且设备大多是异构的,为提高设备间互操作性,DSRC 协议在设计时设定 L7 及其上层应用数据用 ASN. 1 表示。

ASN. 1 使用一种抽象语法来表示协议数据,处于 OSI 七层参考模型中的表示层。它解决了由于端系统的异构(如大小端不同)和编程语言等不同而导致的各种数据交换问题,广泛应用于无线网络、VoIP 等领域。在实际应用中需由 ASN. 1 compiler 将 Abstract Syntax 转换为基于具体语言数据结构(如 C)的 Concrete Syntax,然后由相应的 Encoder 转换为能在媒介上传输的基于比特流的 Transfer Syntax。

ASN. 1 标准所制定的 Encoding rule 有基本编码规则(Basic Encoding Rules, BER)、区分编码规则(Distinguished Encoding Rules, DER)、紧缩编码规则(Packed Encoding Rules, PER)和 XML 编码规则(XML Encoding Rules, XER)等。BER 采取 TLV(Tag, Length, Value)嵌套表示格式,每个字段必须显式按字节编码,存在较多冗余信息。基于减少冗余、节约带宽的考虑,紧凑编码规则 PER 由此产生。PER 按位编码,采取的是 PLV(Optional Preamble, Optional Length, Optional Value)嵌套表示格式,相对于 BER,主要通过以下方式减少冗余:

1)省略类型字段:由于通信双方有着互相兼容的 ASN. 1 描述,接收方总是可以通过参考 ASN. 1 描述来获知编码域中各个字段是哪种类型的内容;

2)精简长度字段:对于定值的长度字段,PER 可以省略编码;并且长度字段的单位根据编码对象的不同可以是位、字节等,而 BER 只能是字节;

3)利用约束:PER 利用约束对数值范围的限制,减少了对 Value 字段的冗余编码。如,有约束的整型数据中,只需编码偏移量。ASN. 1 标准对多种数据类型定义了约束,对减少编码冗余有重要作用。

由于省却了某些字段的处理,PER 的编解码速度一般比 BER 快。PER 又分为 aligned 和 unaligned,aligned 方式要求某些字段按字节对齐,显然,unaligned 更加紧凑;PER 也分为 Basic 和 Canonical,Canonical 要求更严格,如需对 SET OF 排序,适用于安全相关应用(如协议要求含数字签名等)。现有 DSRC 协议采用的均是 Basic unaligned PER,即紧凑而快的编码方式,从而能获得更高的通信效率。本文针对 DSRC 设计了一套 PER 编解码框架并在小设备上加以了实现。

1 编解码框架的设计

1.1 基本思想

ASN. 1 PER 编解码功能位于 L7。本文所设计的 DSRC 协议的编解码模块由两部分构成:

1) ASN. 1 抽象语法描述的数据结构(abstract syntax)到 C 语言(concrete syntax)数据结构的映射(简称 C 结构)之上的

收稿日期:2007 - 12 - 27;**修回日期**:2008 - 02 - 23。

作者简介:唐波(1982 -),男,四川成都人,硕士研究生,主要研究方向:计算机网络; 王能(1942 -),男,湖南醴陵人,教授,博士生导师,主要研究方向:计算机网络协议、计算机网络安全、数字移动通信系统的用户设备测试。

一个符合 ASN.1 所描述协议的编解码框架;

2)编解码框架所需要的 PER 基本数据类型的各个编解码函数以及一些辅助函数等。

这样设计的优点是:第一部分的框架需要随着协议的 ASN.1描述的改变而改变,但是其总体结构具有良好的通用性,适用于 PER aligned、BER 等。第二部分为支撑框架所需的 PER 库函数以及辅助函数,这部分不需要随着 ASN.1 描述改变而改变,具有良好的移植性。

由于 ASN.1 描述是嵌套的,映射后的 C 结构也是嵌套表示的,基于此 C 结构,第一部分所设计的编解码框架也是嵌套式的,即:编码过程按照 C 结构内的成员顺序编码,每编码一个字段,则将其结果存放到编码结果缓冲区;如编码时遇到 C 结构内嵌套另外一个 C 结构,则先编码另外的 C 结构,完成后,再回到原来的 C 结构编码函数以继续之前的编码流程。为维护存放嵌套编码过程中所产生的编码结果的缓冲区并提供一致的错误处理机制等,设计了 ASN1Context 结构,每个编解码函数参数都应传入此结构以保证编解码过程的正确性。

1.2　Abstract Syntax 到 Concrete Syntax

这一步实际上完成的是 ASN.1 compiler 的工作,即将协议的抽象语法表示映射成用具体编程语言表示的数据结构,可以通过第三方工具或者只针对协议要求的 ASN.1 类型手动完成。基于空间和 DSRC 实时性的考虑,我们使用效率较高的 C 语言。ASN.1 数据类型分为复合类型和基本类型,复合类型由基本类型构成,复合类型一般映射为 C 语言中的 struct,基本类型有的需要映射到 struct,有的可以直接映射为 C 语言中的内建类型(如 INTEGER 可以映射为 int,由于应用无特别要求,我们的实现未考虑大整数)。对于嵌套的 ASN.1 结构,在 C 结构使用指针指出所嵌套的结构;对于标记为 OPTIONAL 的字段,在 C 结构开始处添加含位域的结构体以表示各成员是否被设置;对于标记为 DEFAULT 的字段,据标准,可采取忽略策略。主要映射规则如表 1 所示。

表 1　主要映射规则

ASN.1 类型	C 结构	C 结构成员说明
SEQUENCE	struct	各成员顺序排列,每个成员用相应类型的指针指向
SEQUENCE OF	double linked list	链表中的每一个组成元素用 void * 指向
CHOICE	struct	第一个成员用 int 类型来表示当前选择的 union 内的特定成员,第二个成员为 union 用以列举各个成员,各成员用指针指向
INTEGER	int	C 结构内建类型
BIT STRING	struct	第一个成员为 bit 数,第二个成员为数据指针
OCTET STRING	struct	第一个成员为字节数,第二个成员为数据指针

表 2 为 DSRC 中应用层 ASN.1 描述中 Action-Response 和 Action_Response_fill 映射为 C 结构的对照。

1.3　嵌套的编解码流程

基于映射后的嵌套 C 结构,所设计的实际编码流程也是嵌套的。任意一个映射后的 C 结构定义都应有其相应的编码函数,其任务为顺序编码该 C 结构体内的各个成员,如果 C 结构内某个成员为复合类型的指针,即嵌套了另一个 C 结构,则函数调用此 C 结构的编码函数;否则,该成员为 PER 基本类型,则直接调用库函数内的相应基本类型的编码函数(见 2.1 节)。每调用一种类型的编码函数,都会存储编码结果到编码结果缓冲区内。当所有函数返回时,编码结束。

仍然以 Action_Response 为例,其编码函数 asn1PE_Action_Response 调用其所属成员的编码函数流程如图 1 所示。

表 2　一个映射的示例

ASN.1 表示	C 结构表示
Action-Response :: = SEQUENCE{ fill BIT STRING (SIZE(2)), did Dsrc-DID, responseParameter Container	typedef struct extern Action_Response {struct {unsigned respPPresent: 1; unsigned iidPresent: 1;} m; Action_Response_fill fill; Dsrc_DID did;
OPTIONAL, iid Dsrc-DID OPTIONAL, ret ReturnStatus}	struct Container * respPa; Dsrc_DID iid; ReturnStatus ret; } Action_Response;
Action_Response_fill	typedef struct extern Action_Response_fill { numbits; char data[1]; } Action_Response_fill;

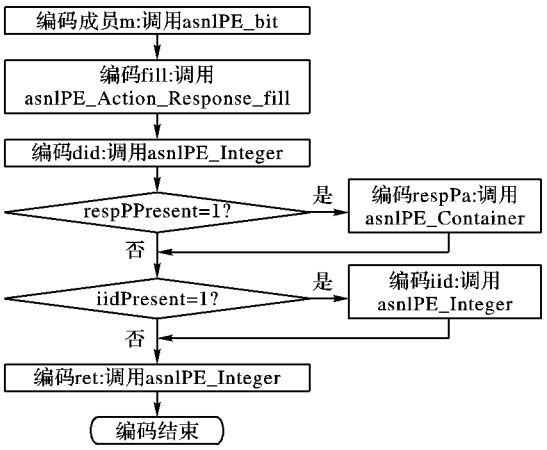

图 1　编码流程示例

图 1 中,asn1PE_Action_Response 等成员体的编码函数根据其自身结构成员还会调用其他结构体的编码函数,这里并没有完整画出。

需要注意的是,如果嵌套调用的层次过深,可能会造成堆栈溢出,对于资源紧张的小设备来说尤其应慎重考虑。基于 RISC 平台的参数的传递以及局部变量的分配多在寄存器内完成,但寄存器数量有限,仍然需要足够的栈空间。根据 DSRC 协议中 ASN.1 的定义以及所映射的 C 结构来看,嵌套层次不会超过 15 层,因此,本文实现时划分 1 KB 为堆栈大小。

解码即编码的逆过程,其流程也是嵌套的,这里不再赘述。

1.4　ASN1Context 结构

显然,嵌套编解码过程中需要维护编解码的执行环境。在编码时,需要存储各嵌套编码函数的编码结果,维护编码结果缓冲区的当前状态和编码过程中所需内存等;在解码时,需要维护当前解码缓冲区的当前位置和解码过程中所需内存等。因此,设计了 ASN1Context 结构,此结构是可扩展的,可按需加入任何 Context specific 的控制信息。同时,为了方便

处理约束,其中加入了一个约束结构以将所编码的ASN.1语法中的所有约束链接为单链表,从而可得到最小约束。ASN1Context结构定义如下:

```
typedef struct _ASN1Context
{ void * pHeap;
  ASN1BUFFER buffer;
  Asn1SizeCnst * pSizeConstraint;
} ASN1Context;
```

其中,pHeap用于支持编解码过程中的内存管理,我们将在2.2节中详述。buffer用于维护编解码缓冲区。pSizeConstraint用于取得最小约束,如需要编码的数据不在最小约束范围内,则为非法数据,那么编码失败;否则根据约束对数据进行处理。

在嵌套编码过程中,各编码函数需要知道当前编码结果缓冲区的位置,以便本编码函数可以在当前缓冲区之后加入其编码结果。同时,由于PER的编码是按位进行的,编码缓冲区结构需要维护一个指示当前缓冲区的位的成员。同样,在嵌套解码过程中,需要指出当前已解码的位位置。基于以上考虑,ASN1Context结构中的缓冲区结构ASN1BUFFER的定义如下:

```
typedef struct
{ unsigned char * pData;
  unsigned int byteIndex;
  unsigned int size;
  unsigned short bitOffset;
} ASN1BUFFER;
```

在编码时,pData指向编码结果缓冲区的起始位置,byteIndex指向当前编码结果字节位置,size指示编码缓冲区的大小,方便进行缓冲区溢出检测,bitOffset指示当前编码字节(byteIndex)内的bit偏移量,因此当前已编码bit位可以通过byteIndex × 8 + bitOffset算得。解码时,pData指向需要解码的数据缓冲区,其他数据成员与编码时的含义类似。

在1.3节流程进行之前,此结构需要被适当初始化。编码流程结束后,可以根据此结构获得编码结果,如bitOffset未字节对齐,那么根据协议要求需要进行位填充以字节进行对齐;解码流程结束,此结构内部偏移量应为源解码数据的末尾,解码结果应由使用者指定的指针指向。

2 库函数的设计与实现

库函数包括基本类型的编解码函数和一些辅助功能函数(如运行时缓冲区检查、内存管理等)。这些函数实现X.691标准所定义的一些基本过程(procedure)以及支撑框架所需的必要功能。库函数不应随着相应协议的ASN.1变化而变化,它实现的是编解码底层的支撑函数,符合标准C,具有较好的可移植性。

2.1 基本类型的编解码库函数

这里的基本类型包括OCTET STRING、BOOLEAN、INTEGER等。这些基本类型大多可以和C语言的内建类型相对应,比如INTEGER对应int;有的则需要映射,如OCTET STRING需要映射为struct,第一个成员指示长度,第二成员指针指示数据缓冲区首地址。

在X.691中,定义了一些基本过程,其中一些本身即基本类型的编解码流程,它们在其他基本类型的编解码函数以及1.3节中的编解码流程中经常会被调用,如:1)non-negative-binary-integer; 2) 2's-complement-binary-integer; 3)constrained whole number;4)semi-constrained whole number;5)unconstrained whole number; 6) small non-negative whole number; 7) length determinant(即长度字段)。各流程含义以及应该在何时使用在X.691中有详细描述。我们将每个流程定义为独立的函数,以供编码基本类型的函数调用。在DSRC实现中,考虑到无需大整数的支持,所以把整数限制在了32位。

基于以上基本流程,编解码更加方便。对于每一个可能产生编解码结果的函数,都需要传入ASN1Context环境变量。同时,在编码过程中对缓冲区的使用尽量做检查,防止缓冲区溢出。解码时,由于无法实现一次性校验所有字段的有效性,所以需要在解码过程中的各函数内部的适当位置检查当前字段,比如在解码获得长度字段L大小后,可以向前探测L个字节,从而验证是否越界。

2.2 内存管理

引入独立的内存管理主要基于以下原因:

1)在DSRC中,协议要求编解码在L7完成,但除了在L7有ASN.1定义外,在L7之上的设备应用层也有ASN.1定义,根据协议标准这部分数据的编解码也应由L7完成。如果使用标准的内存管理,上层申请的C结构需要L7编码完后一一释放,冗繁而且容易造成内存泄露。独立的内存管理可以方便L7和设备应用层间的数据共享;

2)解码时,各字段长度无法准确预知,比如不定长的OCTET STRING,需要进行特殊处理;

3)编解码内存申请频繁,易造成碎片;鉴于DSRC有较高实时性要求,需要提高其内存管理效率。

我们设计的内存管理基于"多次申请,一次释放"原则。首先从RAM专门划出大小合适的内存块,由ASN1Context的pHeap指针指出。所有编解码相关的内存申请都在此进行。申请时采取顺序按需分配,无需串联各块所申请的内存;释放时只需对整个独立内存区一次释放,开销仅为$O(1)$。以下分别从编码和解码两个方面说明此机制。

2.2.1 编码

设备应用层需要将填充好的C结构传给L7进行编解码,L7在编解码完后必须将设备应用层传来的C结构所使用的内存释放以防止内存泄露,因此L7需对设备应用层所申请的每个C结构单独释放,而C结构嵌套的深度和广度都可能比较大,L7要逐个释放每一个C结构显然是不现实的。因而我们专门分出一块内存用于设备应用层和L7对C结构和其他实际数据(如C结构内指针所指向的字节串)的内存申请,L7编码完后将此内存区一次释放。如果C结构1嵌套了C结构2,C结构2中嵌套了C结构3以及C结构4。内存申请时使用函数PerMemAlloc(sizeof(C结构N))以申请"C结构N"大小的内存块,并返回所申请块的头指针。各个C结构申请内存的结果如图2所示。

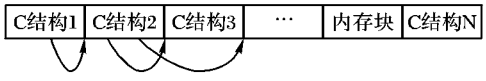

图2 编码内存申请结果

释放时,只需要调用函数PerFree(pHeap),pHeap指向以上整个内存块。释放后,以上整块内存可供下次使用。在实现中,我们将此内存区的大小设置为1 KB,根据各个C结构的定义及其嵌套深度,已足以应付一次编码各个C结构所需空间。

2.2.2 解码

解码时的管理方式和编码类似,在L7收到需要解码的数

据后,所有相关的 C 结构以及其他实际数据(如 C 结构内指针所指向的字节串)都在独立内存区申请,L7 解码完成之后所有 C 结构已经填充好值,L7 可以进行进一步处理。如果 C 结构需要传递给设备应用层,设备应用层需要将此独立内存区释放;否则应由 L7 进行释放。

2.2.3　说明

由于编解码所占用的时间在 DSRC 中较多,"多次申请,一次释放"的方式提高了编解码的效率,解决了编解码内存使用的诸多问题。由于我们的应用是单任务的,所以内存为静态划分。在多任务环境下,编解码所需内存应动态分配,申请释放都在此动态分配的内存区上进行。另外,如果编解码的数据量较大且内存容量充足,难以指定一次为编解码分配多大内存区,可以考虑使用链表将各内存块串联,进行动态管理。

在小设备中,资源相对紧张,内存使用相对比较频繁,在可以预知模块内存使用上限的情况下,使用单独的内存管理方式往往对整体性能的提高有较大作用。以上所述的内存管理机制比较通用,适用于多种应用场景。

2.3　其他辅助函数

其他辅助函数功能包括:根据约束的上界和下界计算表示此字段的最小位数(这在 PER 编码中随处可见);提供编码运行时的缓冲区溢出检查功能;提供解码时向前探测所解码字段的长度以防止解码时超出所解码数据的缓冲区等。这些辅助函数能够简化编解码的错误处理,方便库函数的编写。

2.4　其他

由于各函数为嵌套调用,在设计各函数(如基本类型的编解码函数、辅助函数等)时应该从空间和时间两方面权衡。如果从节约代码空间考虑,应尽量将各个可能多处调用的功能设计为函数;若从时间考虑,应尽量设计为宏。某基本类型的函数可能被多处调用,如果将其设计为宏,产生的代码量将非常可观,小设备上实现时应加以注意。

3　运行结果

本文的实现已稳定运行于实际环境中,在此以 BST 和 VST 的编解码为例简单说明其运行性能。

运行环境:采用 STR712F 为 CPU,其自带 64 KB SRAM,设置其主频为 48 MHz;无操作系统支持,即单线程独占 CPU 运行。通过 Multi ICE 及 AXD 调试器运行编解码程序,通过计算编解码开始与结束时的时钟差值来计算编解码所需要的时间。

运行结果:解码 20 Byte 符合标准格式的 BST 平均开销为 150 μs,编码 90 Byte 符合标准格式的 VST 时平均开销为 350 μs,很好地符合 DSRC 中实时性要求。

4　结语

本文给出了 DSRC 中 ASN.1 模块的实现框架,提出了适合小设备应用的编解码内存管理机制,对其做出了实现。经验证,具有较理想的编解码效率。同时,本文所提出的 ASN.1 PER 编解码方法以及内存管理机制比较通用,适用于其他类型的编解码规则以及应用场景。

参考文献:

[1] 刘富强 项雪琴,邱冬. 车载通信 DSRC 技术和通信机制研究[J]. 上海汽车, 2007(8): 35 - 39.

[2] PANAYAPPAN R, TRIVEDI J M, STUDER A, *et al*. VANET-based approach for parking space availability [C]// International Conference on Mobile Computing and Networking: Proceedings of the fourth ACM International Workshop on Vehicular Ad Hoc Networks. New York: ACM Press, 2007: 75 - 76.

[3] GB/T 16263.2-2006, 信息技术 ASN.1 编码规则, 第2 部分: 紧缩编码规则(PER) 规范[S]. 2006.

[4] GB/T 20851.3-2007, 电子收费专用短程通信——应用层[S]. 2007.

[5] GB/T 20851.4-2007, 电子收费专用短程通信——设备应用[S]. 2007.

[6] ISO/IEC 8825 - 2, ITU-T-1997, Recommendation X.691 ASN.1 encoding rules specification of packed encoding rules [S]. 1997.

[7] ISO/IEC 8824 - 1, ITU-T-1997, Recommendation X.680 Abstract Syntax Notation (ASN.1) specification of basic notation [S]. 1997.

[8] DUBUISSON O. ASN.1 communication between heterogeneous systems [M]. San Francisco: Morgan Kaufmann, 2000.

(上接第 1443 页)

但相似性矩阵中 $s(j,j)$ 的确定是个有待解决的问题,怎样方便设定 $s(j,j)$,同时又能得到恰当的聚类数目,而且一开始各个数据点成为中心的可能性不一样时,又应如何处理才能不产生错误的中心,以及如何自动选择防震荡因子 λ,这些都将是我们接下来要研究的工作。

参考文献:

[1] RUI XU, WUNSCH D II. Survey of clustering algorithms [J]. IEEE Transactions on Neural Networks, 2005, 16(3): 645 - 678.

[2] de CARVALHO F A T. A fuzzy clustering algorithm for symbolic interval data based on a single adaptive Euclidean distance [C]// ICONIP 2006, Part III, LNCS 4234. Berlin: Springer-Verlag, 2006: 1012 - 1021.

[3] GOWDA K C, DIDAY E. Symbolic clustering using a new dissimilarity measure [J]. Pattern Recognition, 1991, 24(6): 567 - 578.

[4] GOWDA K C, DIDAY E. Symbolic clustering using a new similarity measure [J]. IEEE Transactions on Systems, Man and Cybernetics, 1992, 22(2): 368 - 378.

[5] ICHINO M, YAGUCHI H. Generalized Minkowski metrics for mixed feature type data analysis [J]. IEEE Transactions on Systems, Man and Cybernetics, 1994, 24(4): 698 - 708.

[6] de CARVALHO F A T, de SOUZA R M C R, CHAVENT M, *et al*. Adaptive Hausdorff distances and dynamic clustering of symbolic interval data [J]. Pattern Recognition Letters, 2006, 27(3): 167 - 179.

[7] FREY B J, DUECK D. Clustering by passing messages between data points [J]. Science, 2007, 315(5814): 972 - 976.

[8] 洪志令. 模糊聚类中判别聚类有效性的新指标[J]. 计算机科学, 2004, 31(10): 121 - 125.

[9] HUBERT L, ARABIE P. Comparing partitions [J]. Classification, 1985, 2(1): 193 - 218.

[10] EL-SONBARY Y, ISMAIL M A. Fuzzy clustering for symbolic data [J]. IEEE Transactions on Fuzzy System, 1998, 6(2): 195 - 204.

[11] GURU D S, KIRANAGI B B. Multivalued type dissimilarity measure and concept of mutual dissimilarity value for clustering symbolic [J]. Patterns Recognition, 2005, 38(1): 151 - 156.