嵌入式系统读书报告

2016 级软件工程卓越二班 刘瑞康 2016302580242

1. 基本概况

题目:

《基于动态规划的嵌入式系统代码参数存储算法》(A Storage Algorithm of Code Parameters in Embedded System Based on Dynamic Programming)

作者:

Xu Na ; Zhang Xiaotong ; Zhang Yan ; Yuan Lingling ; Zhang Lei ; Hu Guolin

出处:

2008 International Symposium on Computer Science and Computational Technology &

IEEE，20-22 Dec. 2008

时间：2008

2. 详细摘要

2.1 目的

在嵌入式系统中，如何利用有限的物理存储容量来实现代码参数是一个具有挑战性的问题。论文在 DOCSIS 标准下提出了一种基于动态规划的算法来解决这一问题。该算法能够根据总线的宽度、优先级和这些参数的最小存储空间生成参数的最佳存储结构。该算法已在网络设备中得到实际应用，评估结果表明，该算

法能够显著提高物理存储器的利用率，降低硬件成本。

2.2 原理及方法

文章的算法实现基于混合光纤同轴和动态规划，以下为其概念的介绍。

2.2.1 混合光纤同轴（Hybrid Fiber-Coax）介绍

混合光纤同轴（HFC）是广泛使用的宽带接入技术。DOCSIS（数据电缆服务接口规范）的 MAC 层传输是必须实现的重要技术。MAC 层的分组包括以类型/长度/值 (TLV) 形式编码的大量参数。这些由软件分析的参数存储在物理存储器中，供嵌入式系统使用。DOCSIS 标准仅指定代码格式，但是没有说明如何在嵌入式系统中存储这些参数。

HFC 网络是树形分支结构。包括有三个部分：HFC 传输介质，几个电缆调制解调器 (CM) 和一个电缆调制解调器终端系统 (CMTS)。CMTS 是中央管理设备。HFC 最明显的特征是上行信道和下行信道之间的独立性。

A. HFC 网络的拓扑结构

如图 1，在下行信道上，CMTS 是唯一的发送方，顺序发送广播数据包。 每个 CM 都分配一个特殊地址，并且只接收具有自己地址的数据包。

在上行信道上，来自不同 CM 的数据包经过相同的传输介质发送到 CMTS。 任何上行信道都由几个 CM 共享，因此需要管理传输顺序。

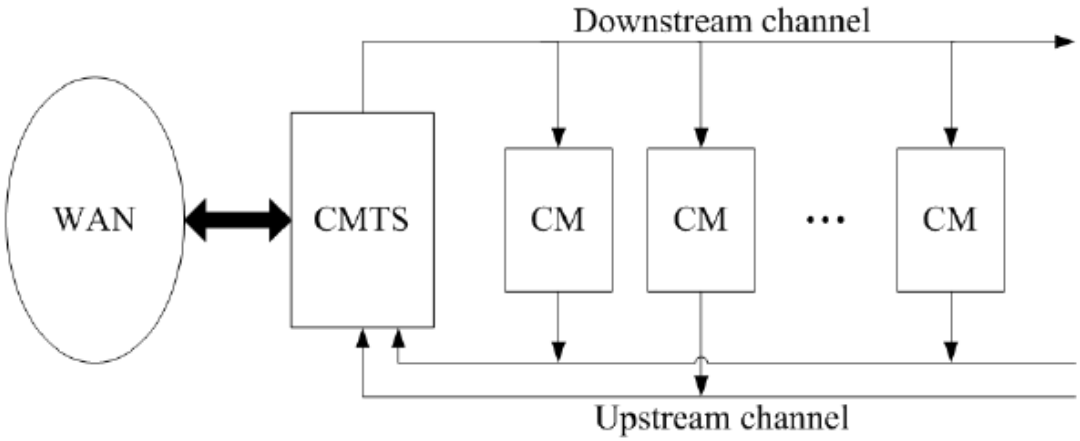


Figure 1. Topology of HFC network

B.上游的传输

CMTS 定义了微时隙，以便能够控制上行信道上的传输。传输机会被定义为可以允许 CM 开始传输的任何微时隙。CMTS 通过带宽分配控制上行信道的分配，并确定哪些微时隙容易发生冲突。CMTS 允许在请求或数据上发生冲突。

C.参数的代码

CMTS 必须以周期性间隔发送上行信道描述符 (UCD) 以定义上行信道的特性 (见图 2)。必须为每个活动的上行信道传输单独的消息。信道 ID 之后的消息参数必须以类型/长度/值（TLV）形式编码，其中类型和长度字段均为 1 个八位字节长。

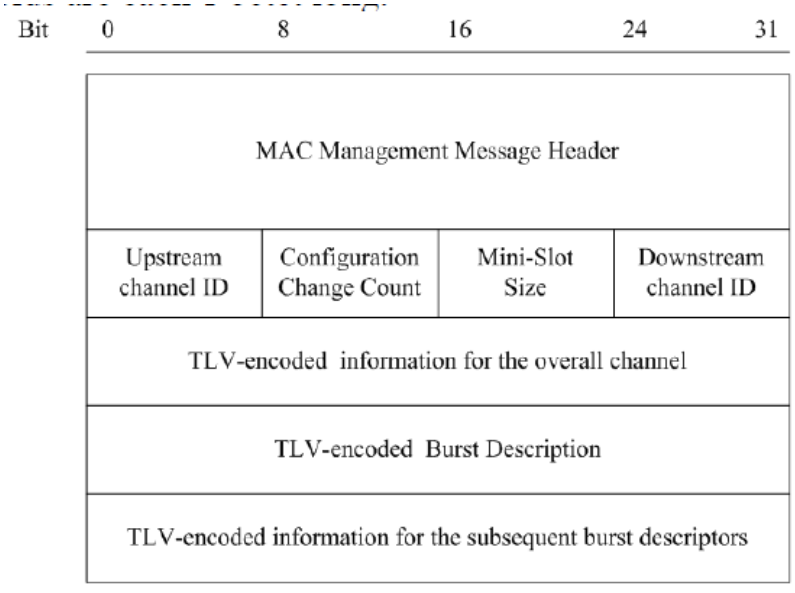


Figure 2. Upstream Channel Descriptor

在带宽分配的每个间隔中使用的分配代码，必须包括描述符。在每个描述符内是无序的物理层属性列表，编码为 TLV 值。这些属性如下表所示。CMTS 必须确保 UCD 中所有描述符的属性集允许该上游的任何 CM 能够请求足够的微时隙以便能够发送最大大小的数据包。根据每个值的有效大小，可以将存储参数的最小大小计算为其硬件位空间。

TABLE I. UPSTREAM PHYSICAL-LAYER BURST ATTRIBUTE

Space (bit)	TLV parameters		
	Type	*Length*	*Value*
3	1	1	1~6
1	2	1	1 or 2
11	3	2	Max(1536)
11	4	2	0~1536
5	5	1	0~16
8	6	1	16~253
16	7	2	0~15
8	8	1	0~255
8	9	1	0~255
1	10	1	1 or 2
1	11	1	1 or 2
8	12	1	8
12	13	2	2×Nr~2048
1	14	1	1 or 2
1	15	1	1 or 2
8	16	1	1~128
5	17	1	1~31
1	18	1	1 or 2

2.2.2 最优的存储策略

为了优化参数的存储，有两个目标:

•使存储空间更小，降低硬件成本并提高物理存储的利用率。

•使存储结构合理，确保在更快的内存中频繁访问参数，以便缩短平均访问时间。

可以一次访问的物理块的大小由系统总线的宽度决定。如果总线是 32 位，则该块为 4 字节。通过属性统计来计算每个参数的访问频率，可以得到其优先级。

存储的最简单方法是将内存分配为参数的长度字段。

图 4 展示了不同存储策略带来的影响。如果系统总线是 16 位，则在访问它们时需要移动 10 个参数。如果系统总线是 32 位，则需要移动的参数数量为 14. 如果系统总线是 8 位，则有 5 个参数必须寻址 2 次，且会分为数据片段。但是，这些操作在任何情况下都是不可避免的。图 4 显示了六种存储策略: 字母 a 后缀表示无掩码操作，并存储最低位的一个参数; 字母 b 后缀表示访问某些参数时存在掩码操作。

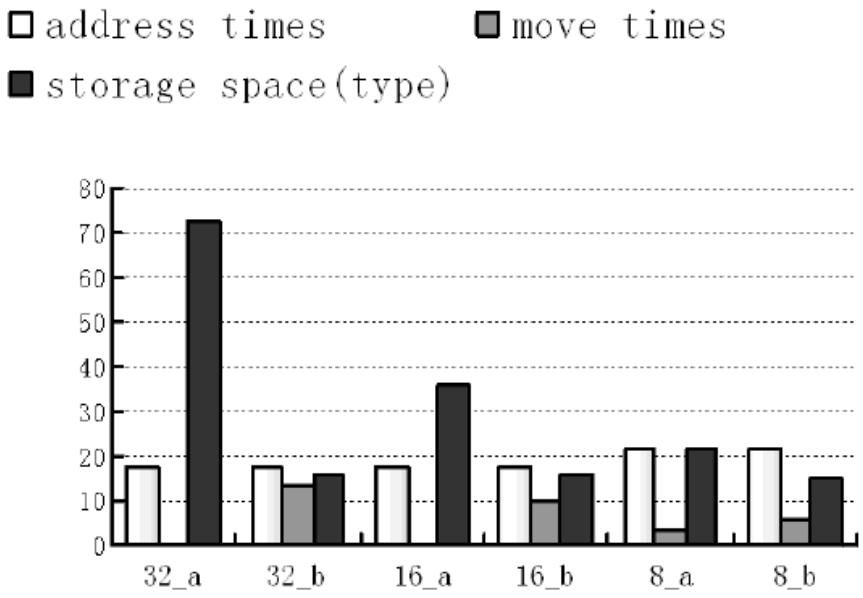


Figure 4. Comparing the storage strategies

通过分析系统的体系结构和统计数据，可以得出以下结论：

•使用掩码操作，可以减少空间的总大小。

•如果没有掩码操作，减小总线宽度，存储空间将减少；而存在掩码操作时，存储空间与总线宽度无关。

•随着总线宽度的减小，访问这些参数的地址的时间也会增加。

•存在掩码操作时，移动操作是不可避免的；在没有掩码操作的情况下，随着总线宽度的增加，移动操作的概率降低。

2.2.3 动态规划

在数学和计算机科学领域，动态规划是一种解决问题的方法，这些问题表现出重叠子问题和最优子结构的特性，利用这些特性来解决问题比原始方法花费的时间少得多。该术语最初由贝尔曼在 20 世纪 40 年代用于描述解决问题的过程，贝尔曼方程是动态编程的核心结果，它以递归形式重述优化问题。

最经典的动态规划问题为背包问题。背包问题是关于组合优化的一个问题。其名称来源于最大化问题，得到放入旅行中携带的一个袋子中的必需品的最优选择组合。其本质为给定一组具有成本和值的物品，确定要包括在集合中的物品和每个物品的数量，以使总成本小于给定限制并且总值尽可能大。

假设有 n 种物品，从 1 到 n。 每个物品 i 有一个值 ci 和一个权重 wi。 可以携带的最大重量是 M. 0-1 背包是原始背包问题的一个特例，其每个物品都不能拆分。0-1 背包问题将每种物品的数量 xi 限制为 0 或 1。

在数学上，0-1 背包问题可以表述为：

$$\text{Maximize } Z = \sum_{i=1}^{n} c_i \times x_i \qquad (1)$$

$$\text{Subject to } \sum_{i=1}^{n} w_i \times x_i \le M(x_i = 0 \wedge x_i = 1) \qquad (2)$$

类似的问题经常出现在商业，组合，复杂性理论，密码学和应用数学中。

2.3 关键技术

A.算法设计

在搜索存储参数的最优策略的过程中，问题可以像背包问题一样被抽象出来。

将存储问题放入 0-1 背包问题的模型中，那么可以一次访问的块是背包的有限最大空间，参数即为待放入的物品，而参数的优先级则为物品的值。

B.算法的实现

算法 1：找出一个块中的最佳存储结构。

```
Function Z(n,M);
    If m<0 then z=Minnum exit
        Else Z1=Z(n-1,m);
    Z2=Z(n-1,m-Wn)+Cn;
    Z=max(Z1,Z2);
End;
```

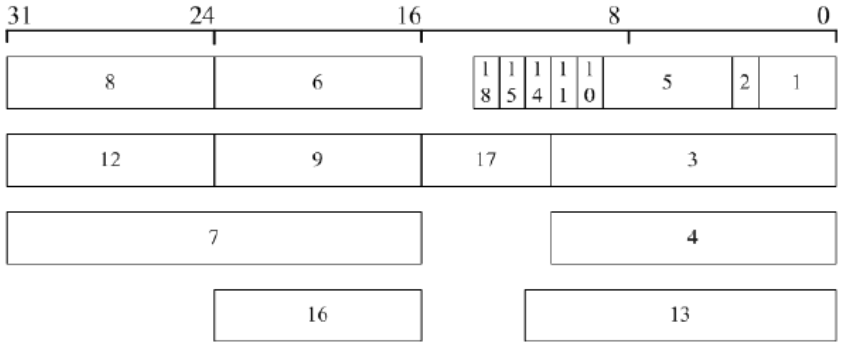
算法 2：用于实现所有参数的最佳存储的多调用。

```
While (n>0)
   Z(n,M);
   Put out the result of Z(n,M);
   j=0;
   For i=1 to n step 1
      If (parameter[i] is not in the block) then
         j=j+1;
         Insert the parameter[i] into queue as parameter[j];
      End if;
   Next i;
   n=j;
End while;
```

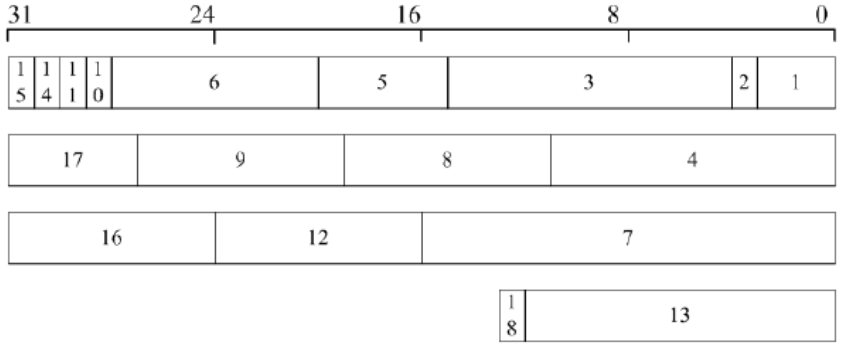
C.测试和评估

根据嵌入式系统的架构，我们可以根据总线的宽度获得一个块的大小。在初始阶段，优先级是随机输入的。经过长时间的系统运行后可以得到访问频率的统计信息，这些统计信息用来决定参数的优先级。分别用 8 位，16 位和 32 位的总线进行测试。

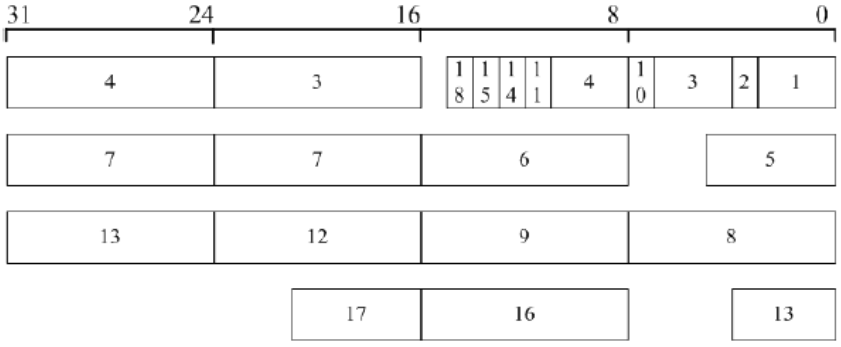
图 6 显示了不同宽度总线的存储结构。



(a) The storage structure of 16-bit bus



(b) The storage structure of 32-bit bus



(c) The storage structure of 8-bit bus

文章的结果数据表明，使用其最佳存储策略可以将物理存储的利用率提高到50%。

2.4 解决了什么问题

节省内存资源和降低硬件成本是嵌入式系统设计的主要原则。文章提出了一种基于动态规划的嵌入式系统代码参数存储算法。该算法很好的体现了嵌入式系统设计的主要原则，可以自动为设计人员生成最优的存储策略。其不仅实现了设计目

标，而且提高了嵌入式系统的开发效率。通过在 HFC 等网络设备中实现，该算法被证明是可靠和令人满意的。

3. 读书体会

3.1 学到的方法和思路

3.1.1 基于树形分支结构系统的设计方案

文章中用于实现算法的混合光纤同轴（HFC）为典型的树形分支结构系统，其上行信道和下行信道又分别体现了典型的传输类型。

对于类似下行信道一对多的传输类型，作为唯一的发送方，可以按顺序或优先级广播待发送的数据包。为每个接收方都分配一个特殊地址，使其只接收具有自己地址的数据包，即可实现在同一信道上的一对多传输。

对于类似上行信道多对一的传输类型，由于发送方在请求或数据上可能发生冲突，需要通过带宽分配或频分复用技术控制上行信道的分配，从而保证接收方接收到完整准确的传输信息。

3.1.2 基于实验结果进行分析并完善理论

文章在实现算法前，先根据是否存在掩码操作以及总线宽度对各种储存策略进行实验，基于实验的结果进行分析，得出掩码操作、总线宽度对于储存空间和效率的影响，在此基础上进行算法的实现。采用这种方法，可以减少不必要的理论推导，并能提高理论的可靠性。

3.1.3 基于统计结果进行设计

文章中，在测试的初始阶段，优先级是随机输入的。经过长时间的系统运行

后可以得到访问频率的统计信息，这些统计信息用来决定参数的优先级。对于不同应用情景，系统的参数优先级可能发生很大改变，而这种改变是无法通过理论推导获得最优设定的。文章采用针对具体系统，根据其统计信息来决定其参数优先级的方式，既简单的规避了可能无解的参数优先级理论设计，也确保了系统在特定应用中的良好表现。

3.1.3 动态规划/编程思想的应用

动态规划作为程序开发乃至数学和计算机科学领域的重要方法，在文章中得到了充分的应用。通过动态规划，利用重叠子问题和最优子结构的特性，可以保证获得最优存储结构。这是我所了解到的第一个应用于实际系统而不是理论算法的动态规划的应用，此前在我的认识中，各种编程思想与实际系统之间存在着多重抽象，基本是完全割裂的，在文章中，由于嵌入式系统的简单系统结构和层级，实现了系统和编程思想的融会贯通，使我进一步体会到了动态规划及其他编程思想的实际应用意义并不止步于理论。

3.2 推广与应用

文章提出的的基于动态规划的嵌入式系统代码参数存储算法应用性极强，并不仅限于 HFC，而是适用于所有需要在有限空间进行结构化储存的情景。通过人为的设计同样可以实现储存结构的最优实现，但采用该算法可以大大提高设计效率，降低设计成本，因此适用于各类小规模应用的嵌入式系统的存储设计以节约开发成本，提高经济效益。

3.3 文献缺陷及解决方案

3.3.1 文献可能存在的缺陷

文章算法的参数优先级是由一段长时间的系统运行得到的访问频率的统计信息来决定的，这种实现方式对于单一应用情景的嵌入式系统来说实现简单、效率较高。但实际生活中，存在应用情景随时间发生周期性阶段性改变的情形，在此种情形下，如果仍然采用该实现机制，可能仅能在其中较少情景下实现高效运作，在情景改变导致参数访问模式发生巨大变动时，固定的优先级实现甚至会造成效率的负面降低。

3.3.2 设想的解决方案

A.进行人工设计

人为的划分使用情景，针对其中每一单一情景，仍采用在初始阶段，随机输入优先级，使用长时间的系统运行后得到的访问频率的统计信息来决定参数的优先级的方式。设定程序，当情景切换时，应用不同情景下获得的优先级运行。

B.运用模式识别

采用附加的设备或系统，定义较小的模式周期，在每一周期的初始阶段随机输入优先级，经过一定时间后，根据当前的统计信息与历史获得的统计信息进行模式识别与分类，根据匹配结果产生新的情景优先级信息或应用匹配的情景产生的优先级信息。