

文章编号:1001-9081(2006)06-1301-03

基于 CAN 总线实时应用的可靠调度性研究

邓竹莎,雷航,罗淳,康涌泉

(电子科技大学 计算机科学与工程学院,四川 成都 610054)

(snoopyshy82@yahoo.com.cn)

摘要: CAN 总线是一种高级的串行通信协议,适用于各种分布式控制系统。在实时应用中,标准的 CAN 协议使用静态优先级算法,对传输信道的利用率比较低。对基于 CAN 总线通信的动态优先级调度算法进行研究后,提出了一种基于指数分配方式的 MTS 算法,在保证强实时性消息的同时兼顾了低优先级消息的公平性。

关键词: 标识符;单调时间限算法;单调速率算法;混合通行算法;最早时间限优先算法

中图分类号: TP393.04 **文献标识码:** A

Research on guaranteed task scheduling of real-time applications based on CAN-bus

DEND Zhu-sha, LEI Hang, LUO Chun, KANG Yong-quan

(College of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu Sichuan 610054, China)

Abstract: CAN is an advanced serial communication protocol and adapts to kinds of distributed control systems. In real-time applications, standard CAN protocol uses static priority algorithm. It gives a low schedulable utilization. To study the dynamic priority algorithm on CAN-bus, a mixed traffic scheduler (MTS) based on exponential assignment was proposed. It gives attention to both hard and soft real-time messages.

Key words: identifier(ID); DM(Deadline Monotonic); RM(Rate Monotonic); MTS(Mixed Traffic Scheduler); EDF(Earliest Deadline First)

0 引言

分布式控制系统是由多个微处理器(或微控制器)分别控制系统中多个控制回路,同时又可集中获取数据、集中管理和集中控制的自动控制系统。各回路之间和上下级之间通过高速数据通道交换信息。分布式控制系统具有数据获取、直接数字控制、人机交互以及监控和管理等功能,使用广泛的汽车电子就是一例采用一种基于 CAN 总线的分布式现场总线控制网络结构^[1],如图 1 所示。

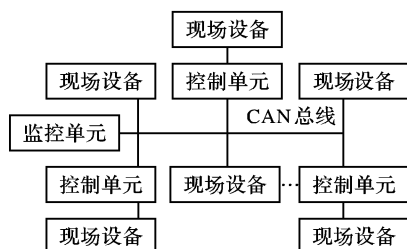


图 1 分布式现场总线控制网络系统结构

CAN 协议允许两种格式的消息同时存在:标准格式与扩展格式。这两种格式的主要区别在于标识符的长度。标准格式使用 11 位的标识符,扩展格式使用 29 位的标识符。

标识符是 CAN 总线提供实时性保证的基本措施,它主要有两个功能:为消息分配一个优先级以及判断消息是否被某

个节点接收。

标准的 CAN 协议在为消息分配优先级时,使用静态优先级算法,如:DM 和 RM。使用静态优先级算法很简单,便于管理和可调度性的判断。一旦为消息分配了优先级,直到消息结束优先级也不会变化。可是如同单机上一样,静态优先级算法不如动态优先级算法灵活,并且传输信道的利用率也不高。因此,一些调度算法将动态优先级思想引入 CAN 总线协议中,使其更适合于实时系统,具有更好的性能。目前在 CAN 总线上的动态优先级算法主要是 MTS。

本文在介绍 MTS 算法原理及其优缺点的基础上,提出了一种基于指数的 MTS 改进算法。该算法使截止时间近的消息发生优先级反转的可能性非常小。此外,还提出了一种简单有效的方法来计算整个系统的 ID 更新时间,避免了由于单点损坏带来的问题。最后,算法还在保证高优先级消息时间限制的前提下,考虑了低优先级消息的公平性,提高了系统的整体性能。

1 MTS 算法

MTS 是一种介于静态优先级算法(DM)和动态优先级算法(EDF)之间的折中算法。它的主要思想是将消息的相对截止时间编入标识符中,对高优先级的消息使用 EDF 算法,对低优先级的消息使用 DM 算法。

收稿日期:2005-12-09;修订日期:2006-02-20

作者简介: 邓竹莎(1982-),女,四川康定人,硕士研究生,主要研究方向:嵌入式实时系统、现场总线;雷航(1961-),男,四川富顺人,副教授,博士,主要研究方向:嵌入式实时系统可靠性测试及评价、系统性能分析、实时软件工程;罗淳(1981-),女,四川德阳人,硕士研究生,主要研究方向:嵌入式实时系统、现场总线;康涌泉(1978-),男,四川安岳人,硕士研究生,主要研究方向:嵌入式 Linux 技术、Linux BSP 技术。

因为绝对截止时间会不断增大,而用于表示时间的位数有限,因而 MTS 对截止时间的描述采用相对截止时间^[3,4]。相对截止时间用绝对截止时间与一个周期性增加的变量 SOE (Start Of Epoch) 的差值来表示。相邻两个 SOE 之间的长度称为时间段长度,用 l 来表示。 l 的长度由 CPU 分配的用来更新 ID 的时间段 x 和更新 ID 所需要的指令条数 n 决定: $l = n/(xM \times 10^6)$, 其中 M 表示 CPU 的运算速度,以 MIPS 为单位。如果用 t 表示当前时间, d_i 表示消息 i 的绝对截止时间,那么相对截止时间表示为: $D_i = d_i - \text{SOE} = d_i - \lfloor t/l \rfloor \times l$ 。在每个 SOE, CPU 将根据相对截止时间重新计算消息优先级,并将其填入标识符中。

MTS 中消息的标识符分配如图 2 所示。在本文中, CAN 总线使用“与”逻辑。



图 2 MTS 中各种消息的 ID 结构

首先,通过对最高位的设置,使硬实时消息具有比较软实时消息以及非实时消息更高的优先级。然后,在硬实时消息中将消息的相对截止时间填入 ID 中,以便对高优先级的消息使用 EDF 等动态优先级算法。由于相对截止时间的范围也很广,如果每一个截止时间都要对应一个单独的优先级,即目前的标识符长度是不能满足的,也是很很不合理的。因此使用了域(region)的概念。域的长度 l_r 表示为 $l_r = (l + D_{\max})/2^m$, 其中 m 表示在 ID 中用来表示相对截止时间的位数, D_{\max} 表示在 l 时间段产生的实时消息的最大生存时间,即 $\max(d_i - r_i)$, 其中 r_i 为消息 i 的产生时间。各消息根据其相对截止时间所在的域不同而获得不同的优先级,落在相同域中的消息具有相同的优先级。在每次 SOE,本地微处理器或微控制器将重新计算消息的优先级。若 m 为 3,则相对截止优先级分配如图 3 所示。

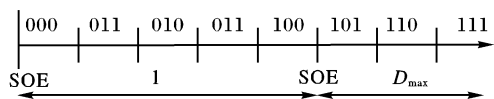


图 3 当 $m=3$ 时优先级分配示意图

具有相同优先级的硬实时消息通过后面 DM 优先级进行区分,以适应 CAN 总线对所有消息分配各不相同的标识符的要求。

而对于软实时消息以及非实时消息,则事先为其分配相应的 DM 优先级和固定优先级。它们的优先级在消息生存期间不再改变。

MTS 由于考虑了消息的截止时间,因此有更高的使用效率。实验结果表明使用 MTS 的效果远远好于仅使用静态优先级算法 DM;除了在负载非常重的情况下,MTS 的表现与 EDF 很接近。

MTS 最大的缺陷在于未解决优先级反转。假设两个消息 M_1 和 M_2 都属于硬实时消息,且相对截止时间落于相同的域,因此为其分配相同的优先级。若 $D_1 < D_2$,而 M_1 实际上没有能力阻止 M_2 先于自己执行。此外,如何让所有节点同时更

新 ID,避免节点 ID 的不一致性,以及缺乏对低优先级的消息的公平性考虑等都是制约 MTS 的因素。

2 基于指数的 MTS 改进算法

该算法对域的划分不再使用等分的方式,而采用指数划分的思想,因而在很大程度上缓解了优先级反转。此外在优先级的设置中加入对等待时间的考虑,以解决长期被阻塞的低优先级消息的公平性问题。

2.1 域的划分

在通常的 MTS 中,对域的划分使用平均的思想,每个域的长度相同。在本文中则使用指数的方式分配域。假设 ID 中使用 m 位来表示动态优先级,则有 $n = 2^m$ 个不同的优先级等级,即 n 个不同的域。其中第 k 个域的长度表示为 $\delta_k = t_{k+1} - t_k = 2^k \times l_0$ ($k = 0, 1, \dots, n-1$), 其中 l_0 是一个标量, $n = 2^m = \lceil \log_2((l + D_{\max})/l_0 + 1) \rceil$ 。所有相对截止时间落在区间 $[t_k, t_{k+1})$ 中的消息,为其分配优先级 k 。最后一个域比较特殊,它可能不是 l_0 的整数倍。具体分配方式如图 4 所示。

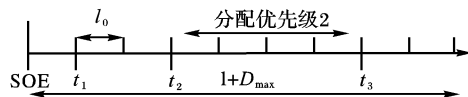


图 4 基于指数的域分配方式

2.2 ID 更新时间的计算

处理器在每个 SOE 开始的时候,根据当前待发送消息的绝对截止时间重新计算其相对截止时间以及动态优先级。

可是在一个分布式系统上要采用统一的时钟是比较困难的,因此如何保证总线上所有的节点都能够同时更新 ID,而不会出现新旧 ID 同时存在的情况是本算法中的一个难点。本文将采用以下方式更新各节点 ID。步骤如下:

1) 首先为各节点分配一个周期性的计时器,每隔 1 s 产生一次中断;

2) 当某个节点上产生该中断,则改写该节点上待发送消息队列中优先级最高消息的发送格式,将其中 CAN 协议帧格式中控制字段保留的 r_0 和 r_1 置为“01”,表示该节点计时器已到时;

3) 任何节点接收到 CAN 协议帧以后首先判断 r_0 和 r_1 是否为“01”。若为“01”,则接收该消息,将本节点计时器置 0,并重新计算该节点所有待发消息的 ID。在所有消息的 ID 都重新计算以后才开始使用新的 ID 进行消息发送。对接收到的消息除了对 r_0 和 r_1 进行判断,还将对其 ID 进行判断:若原本就是本节点应该接收的消息则将该消息送往应用层;否则丢弃该消息。

该方式计算量不大,却避免了使用单一的节点发送消息来统一各节点时钟^[5]所带来的单节点损坏问题。当单节点故障发生时,则必须有单独的算法来重新选举新的主节点,否则系统将不能正常工作。

2.3 消息的公平性

在 CAN 总线上,若多个节点同时发送消息则会产生冲突。解决冲突的方式是对标识符(ID)从高位开始进行与操作。节点首先将标识符最高位在总线上进行广播,间隔一段时间后(确定消息已经送达每个节点),该节点再从总线上读取数据,判断自己是否能够继续竞争发送权。若某节点发送“1”,而读回的数据为“0”,说明有优先级更高的消息等待发

送,则退回到发送前的状态,取消发送。

这种机制保证了高优先级的消息能够在最短的时间内发送出去。当高优先级的消息频繁发送时,低优先级消息就会被长期阻塞,这就是公平性问题。在通常的CAN协议中并没有考虑这一点。其实在保证高优先级消息的前提下,适当的考虑低优先级消息的公平性有助于提高系统性能。

修改了的MTS算法对于公平性问题的考虑采用了如下的方法。通常ID的结构如图2所示,修改以后的结构如图5所示。正常情况下将硬实时消息最高两位置为“00”,软实时消息最高两位置为“10”,非实时消息最高两位置为“11”,以确保高优先级消息的正常发送。若某条软实时消息反复发送*i*次,仍未能获取总线,则将其最高两位置为“01”,提高其优先级,使其在下次总线竞争中,能有更大的概率获取总线发送数据。同样,若非实时消息反复试验*j*次仍不能获得总线发送数据,则将该消息最高两位置为“10”。由于系统的具体环境以及系统的目标不同,*i*和*j*的值不同,根据实际情况确定。

由于硬实时消息最高两位始终为“00”,所以即使其他消息优先级别有所提高,也不会影响其响应时间。

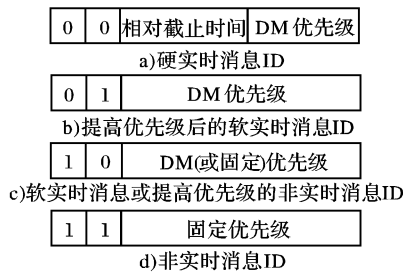


图5 改造后MTS中各种消息的ID结构

3 算法性能比较

下面对通常的MTS算法与改进后的算法造成优先级反转的可能性进行比较。假设系统仅有两个消息 M_1 和 M_2 ,其中 $D_1 < D_2$,当两个消息落入同一个域时就可能导致优先级反转。下面仅对最坏情况进行计算。假设 D_1 和 D_2 落入相同区间 $[t_k, t_{k+1})$ 中,分配相同的优先级 k_0 。区间长度 $\delta_k = t_{k+1} - t_k$,因此 $t_k \leq D_1 < t_{k+1}$ 且 $D_1 < D_2 < t_{k+1}$ 。 D_1 和 D_2 分配如图6所示。

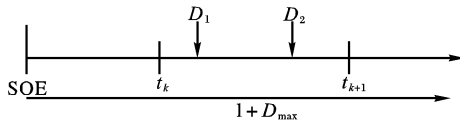


图6 D_1 和 D_2 分配

优先级反转的可能性是:

$$P = (t_{k+1} - t_k) / L \times (t_{k+1} - D_1) / L = \delta_k / L \times (t_{k+1} - D_1) / L \leq \delta_k / L \times \delta_k / L = \delta_k^2 / L^2, \text{ 其中 } L = l + D_{\max}.$$

在通常的MTS中 $\delta_k = t_{k+1} - t_k = L/2^m$,因此 $P_1 \leq (L/2^m)^2 / L^2 = 1/2^{2m}$ 。

修改后的MTS中 $\delta_k = t_{k+1} - t_k = 2^k \times l_0$,其中 $k = 0, 1, 2, \dots, n = 2^m = \lceil \log_2(L/l_0 + 1) \rceil$,所以 $2^{n+1} - 1 = L/l_0$ 或者 $2^n - 1 = L/l_0$ 。因此 $P_2 \leq (2^k \times l_0)^2 / L^2 = 2^{2k} \times (l_0/L)^2 = 2^{2k} \times (1/(2^{n+1} - 1))^2$,或者 $P'_2 \leq (2^k \times l_0)^2 / L^2 = 2^{2k} \times (l_0/L)^2 = 2^{2k} \times (1/(2^n - 1))^2$,其中 n 为一个常数,表示动态优先级的级数。当域越接近SOE端点,域的长度越短,两个消

息落在同一个优先级范围内的可能性越小,造成优先级反转的可能性也越小;相反则可能性越大。该机制的合理性在于相对截止时间越接近于SOE的消息在发生优先级反转时,受到的危害越大。在本例中取 $k = 0$ 计算两个消息同时落在第一个域中的可能性为 $P_2 \leq (1/(2^{n+1} - 1))^2$ 或 $P'_2 \leq (1/(2^n - 1))^2$,其中 $n = 2^m$ 。若在ID中取5位表示动态优先级,即 $m = 5$,此时出现优先级反转的最大概率低于 $1/(2^{32} - 1)^2$,远远低于使用普通MTS算法的 $1/2^{10}$ 。

4 结语

在CAN总线上通常使用的动态优先级算法MTS,它对高优先级消息使用EDF,对低优先级消息使用DM算法。在多数情况下这种算法的效率比较高,接近于EDF算法,远远高于使用静态优先级算法(如DM和RM)。可是MTS也有很多不足的地方,比如对总线上所有节点更新标识符的时间的确定,对低优先级消息公平性的考虑等,其中最主要的问题是优先级反转。

本文提出了一种MTS的改进算法,它基于指数方式分配动态优先级,使相对截止时间接近SOE的消息发生优先级反转的可能性非常小。该算法还对整个系统更新ID的时间提出了一种简单有效的方法,避免了由于单点损坏带来的问题。最后,算法还在保证高优先级消息时间限制的前提下,考虑了低优先级消息的公平性,提高了系统的整体性能。

文中提出的算法仍然有待完善,主要问题有:

1) 优先级反转问题并没有从根本上解决,只是在很大程度上得到了缓解;

2) 很多潜在的优先级反转问题并没有解决,如CAN控制器如果为待发送的消息提供一个缓存,则可能出现先产生的低优先级消息已经占满了空间,导致后产生的高优先级消息无法发送;若允许高优先级消息抢占低优先级消息占据的空间,当缓冲区的内容正在更新的时候,其他节点可能有一个低优先级消息抢先发送,造成优先级反转;

3) 本算法中使用的用于统一系统中各节点时钟的算法的精确度不够高。

CAN总线作为汽车电子领域中事实上的标准,必须为各类电子设备(ECU)提供实时性保障,因此为其提供一种有效的算法在将来的很长一段时间仍然非常重要。

参考文献:

- [1] NAVET N. Controller area network [CANs use within automobiles] [J]. Potentials, IEEE. 1998, 17(4): 12-14.
- [2] ZUBERI KM. Design and implementation of efficient message scheduling for controller area network [J]. IEEE Transactions on Computers. 2000, 49(2): 182-188.
- [3] NATALE MD. Scheduling the CAN bus with earliest deadline techniques [A]. Real-Time Systems Symposium. Proceedings[C]. The 21st IEEE, 2000. 259-268.
- [4] ZUBERI KM, SHIN KG. Non-preemptive scheduling of messages on controller area network for real-time control applications[A]. Real-Time Technology and Applications Symposium. Proceedings[C]. IEEE, 1995. 240-249.
- [5] PEDREIRAS P, ALMEIDA L. EDF message scheduling on controller area network [J]. Computing & Control Engineering Journal. 2002, 13(4): 163-170.