doi:10.3969/j.issn.1000-436x.2013.12.003

异构网络化汽车电子系统中多 DAG 离线任务调度

谢国琪，李仁发，杨帆，黄卫红
(湖南大学 嵌入式与网络计算湖南省重点实验室，湖南 长沙 410082)

摘　要：公平性和通信开销已成为影响调度性能的主要瓶颈，首先在任务优先级排序阶段，提出基于通信开销权值的轮转调度的公平排序标准 ;在处理器选择阶段,提出在插入法的基础上将任务分配到具有最小选择值的选择标准；综合上述 2 个阶段，提出以降低调度长度和减少通信开销为目标的多 DAG 离线公平任务调度(MDOFTS, multiple DAGs off-line and fairness task scheduling)算法。异构网络化汽车电子系统是一个典型的混合关键级嵌入式系统，既要确保实时性又要降低调度长度，提出了以满足安全关键 DAG 的多 DAG 离线优先级任务调度(MDOPTS, multiple DAGs off-line and Priority task scheduling)算法。综合 MDOFTS 和 MDOPTS，提出多 DAG 离线自适应任务调度(MDOATS, multiple DAGs off-line and adaptive task scheduling)算法，在满足实时性的基础上提高调度性能。实例分析和实验结果表明，提出的算法在调度长度、通信开销、不公平性、最差响应时间和实时性上都优于其他算法。

关键词：异构网络化汽车电子系统；多 DAG；通信开销；调度长度；实时性
中图分类号：TP393　　**文献标识码：**A　　**文章编号：**1000-436X(2013)12-0020-13

Multiple DAG off-line task scheduling for heterogeneous networked automobile electronic systems

XIE Guo-qi, LI Ren-fa, YANG Fan, HUANG Wei-hong
(Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, Changsha 410082, China)

Abstract: Fairness and communication overhead have become the major bottleneck in performance of scheduling, a fair sorting criteria based on round-robin with communication overhead weight was proposed and a selection criteria based on assigning the task to the minimum selection value considering insertion was proposed in processor selection phase. The multiple DAG off-line and fairness task scheduling (MDOFTS) algorithm was proposed combining the above two phases to reduce the schedule length and communication overhead. Heterogeneous networked automobile electronic systems are typical mixed-criticality embedded systems, which must make sure to be real-time and to reduce scheduling length. The multiple DAG off-line and priority task scheduling (MDOPTS) algorithm was proposed to make sure the safety-critical DAG. The multiple DAG off-line and adaptive task scheduling (MDOATS) algorithm was proposed to improve the system performance on the basis of real-time based on MDOFTS and MDOPTS. Example analysis and experimental results show that the MDOFTS algorithm is better than other algorithms in schedule length, communication overh unfairness, worst-case response time and real-time.

Key words: heterogeneous networked automotive electronic systems; multiple DAG; communication overhead; schedule length; real-time

1 引言

近年来，为了满足人们在安全性、驾驶性能等方面提出的更高要求，汽车电子系统规模和复杂性骤增。它由动力控制子系统、底盘控制子系统、安全控制子系统和车身控制子系统等多个子系统组

收稿日期：2013-06-18；**修回日期：**2013-10-09
基金项目：国家自然科学基金资助项目(61173036, 61070057, 61272061)；国家高技术研究发展计划(“863”计划)基金资助项目(2012AA01A301-01)
Foundation Items: The National Natural Science Foundation of China(61173036, 61070057, 61272061); The National High Technology Research and Development Program of China (863 Program) (2012AA01A301-01)

成，每个子系统又包括多个功能任务(如安全控制子系统包括防抱死制动、线控刹车等)[1,2]。这些子系统由遍布车身的上百个的异构电子控制单元（ECU, electronic control unit）通过各类异构总线互联和协作以完成各种智能控制功能[2]。如宝马 7 系，其 6 个子系统共享 70 多个 ECU，ECU 之间通过 LIN、CAN、FlexRay、MOST 和 Ethernet 5 种类型的网络和网关实现互连[3]；奥迪 A8 则包含 95 个 ECU 遍布 7 个子系统，由 CAN、FlexRay、LIN 和 MOST 4 种总线类型互联[4]。因此新一代汽车电子系统体系结构演变为异构化、网络化和复杂化的分布式网络体系结构。

然而，异构网络化汽车电子系统是一个典型的混合关键级嵌入式系统[5]，对性能与安全关键功能子系统的调度要求极其苛刻，例如防抱死制动功能与线控刹车功能，即使调度结果正确，但只要有一个无法在截止期限内完成，执行结果也就毫无意义，甚至造成车毁人亡的灾难性后果；与此同时，网络化使系统产生的通信数据量剧增且结构多样化，使调度极易产生大量的通信开销，从而造成系统调度结果延迟，通信开销已成为影响调度性能的主要瓶颈[1]。因此如何同时保证多个功能子系统都能在截止期限内完成并降低调度长度和减少通信开销已成为异构网络化汽车电子系统调度问题所面临的主要挑战。

2　调度模型

汽车电子系统的异构化、网络化使其调度问题变得复杂，因此需要一种形式化的调度模型加以描述。由于汽车电子系统由多个异构 ECU（不同供应商提供的 ECU 在设计方法或硬件实现上不同）、传感器和执行器等处理设备组成，本文将这些处理单元统称为处理器，并用集合 $P=\{p_1, p_2, \cdots, p_k\}$ 描述这些异构处理器的集合。以安全功能子系统为例，它的实现是由多个相互具有数据依赖的任务组成。首先通过摄像头采集车道偏离、红外传感器感知夜视、雷达自适应巡航控制，超声波辅助停车等；然后通过传感器融合，目标对象监测和识别技术交由刹车片、方向盘和节流阀等执行设备处理；再由线控转向、线控刹车、线控加速等功能做出行驶决策。上述功能任务在不同的处理器上执行，并产生大量的通信开销[6]。

因此，可以用一个有向无环图 DAG 描述一个功能子系统[7,8]。$G=\{N,W,E,C\}$ 表示一个 DAG，其中，$N=\{n_1,n_2,\cdots,n_i\}$ 表示任务集合；由于处理器处理能力的异构性，使不同任务在不同的处理器计算时间不尽相同，因此描述 W 是一个 $|N|\times|P|$ 的矩阵，$w_{i,k}$ 表示任务 n_i 在 p_k 上的计算执行时间开销；$E=\{e_{1,2}, e_{2,3},\cdots,e_{i,j}\}$ 描述为任务间的优先级约束与数据依赖关系集合；由于处理器之间通过 CAN、FlexRay、LIN 和 MOST 等多种类型的网络和网关实现互连，不同总线的带宽不尽相同，因此任务之间的通信开销也不同，描述 $C=\{c_{1,2},c_{2,3},\cdots,c_{i,j}\}$ 具有数据依赖的任务之间的通信开销集合，如果将这 2 个任务分配到同一个处理器上，则通信开销为 0。$pred(n_i)$ 表示任务 n_i 的直接前驱任务集合，$ind(n_i)$ 表示 n_i 的入度，也就是其直接前驱任务集合的个数，当前任务只有在它全部前驱任务的数据到达后才能执行。$succ(n_i)$ 表示任务 n_i 的直接后继任务集合，$outd(n_i)$ 表示 n_i 的出度，也就是直接后继任务集合的个数。没有前驱任务的任务为入口任务，记为 n_{entry}。没有后继任务的任务为出口任务，记为 n_{exit}。DAG 任务模型不仅考虑了考虑任务的优先级，还考虑了任务和处理器之间的相关性以及任务之间的通信开销，因此适合于汽车电子系统的调度问题研究。

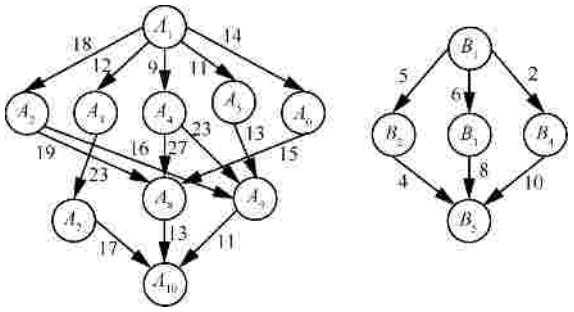
异构网络化汽车电子系统包括动力控制子系统、底盘控制子系统、安全子系统和车身子系统等多个网络子系统，因此需以多 DAG[9~13]来表示汽车电子系统调度模型，每个子系统表示一个 DAG，DAG 之间共享处理器和通信总线，由于各 DAG 都是同时发生，因此属于离线模型。本文用 $GS=\{G_1,G_2,\cdots,G_m\}$ 表示多 DAG 离线任务调度模型。因此可将异构网络化汽车电子系统的任务调度问题形式化为多 DAG 离线任务调度问题并进行研究。以下给出在多 DAG 中与本文密切相关的几个概念。

定义 1　DAG 调度长度(schedule length)。$makerspan(G_m)$ 表示 G_m 的所有任务调度完毕后，其出口任务的完成时间。

定义 2　多 DAG 调度长度(multiple DAG schedule length)。$makerspan(GS)$ 就是 GS 中具有最大调度长度 DAG 的调度长度。

如图 1 为一个多 DAG 任务模型，包含 DAG-A 和 DAG-B，表 1 为相应的多 DAG 计算开销矩阵，本文采用与文献[11]相同的模型实例，如图 1 和表 1 所示，例如图 1 中，A_1 与 A_2 之间的数字 18 表示任务 A_1 与任务 A_2 若分配不在同一处理器上执行的通

信开销，若 A_1 与 A_2 在分配在同一处理器上，则通信为 0；表 1 中 A_1 与 p_1 所结合表示的数字 18 表示为任务 A_1 在处理器 p_1 上的计算执行开销为 14。



(a) DAG-A (b) DAG-B

图 1 2 个 DAG 任务模型实例

3 相关研究与理论知识

谢勇[8]等研究时间触发网络总线技术 FlexRay 的静态段配置消息调度，KLOBEDANZ[14,15]等研究 FlexRay 的静态段配置容错调度，然而上述算法仅针对单一网络总线 FlexRay，且限于同一功能子系统内调度，不适应于汽车电子系统的异构化、网络化和复杂化需求。

3.1 多 DAG 任务调度的公平性

异构系统多 DAG 也包含任务优先级排序和处理器分配 2 个步骤。HEFT[7]算法提出了向上排序值的概念，其计算公式为

$$rank_u(n_i)=\overline{w}_i+\max_{n_j\in succ(n_i)}\{c_{i,j}+rank_u(n_j)\} \quad (1)$$

向上排序值已成为事实的任务优先级排序标准而被多种多 DAG 任务调度算法采用，但采用平均值 $\overline{w_i}$ 作为计算开销，对于大规模的多 DAG 任务调度，计算结果则不够精确。

HÖNIG[9]等提出了将多个 DAG 合成一个复合 DAG 后，再利用诸如 HEFT 等经典的单 DAG 启发式调度算法调度。但是此方法对于某些短 DAG 有明显的不公平性，例如在采用 HEFT 算法的情况下，尽管它们是同时调度，但是由于短 DAG 的向上排序值相比长 DAG 明显要小，因此短 DAG 中的任务始终得不到执行，最终造成短 DAG 调度长度和整个多 DAG 的调度长度都过长，因此需要在多 DAG 调度中保证一定的公平性，以降低调度长度。

ZHAO[10]首次指出了在多 DAG 调度中的公平性问题，并提出了 *slowdown* 驱动的多 DAG 公平任务调度算法 Fairness。其思想是首先基于 HEFT 算法对每个 DAG 调度一次且记下每个 DAG 的调度结果，并把各自的调度长度作为此 DAG 的值，然后将所有 DAG 按降序排列放入列表，选择具有最大 *slowdown* 值的就绪任务进行调度，并更新 *slowdown* 值(如果有多个 DAG 的 *slowdown* 值相等，则选择向上排序值最大的任务调度)。*slowdown* 计算公式为

$$slowdown(n_i)=\frac{ft_{\text{own}}(n_i)}{ft_{\text{multi}}(n_i)} \quad (2)$$

其中，$ft_{\text{own}}(n_i)$表示 n_i 单独调度的完成时间，$ft_{\text{multi}}(n_i)$表示 n_i 在多 DAG 中调度的完成时间。如此重复，直到列表中没有未调度完成的 DAG 为止。同时文献[10]也提出了评价 DAG 不公平性的具体方法，首先计算某个 DAG 的 *slowdown* 值，其计算公式为

$$slowdown(G_m)=\frac{makespan_{\text{own}}(G_m)}{makespan_{\text{multi}}(G_m)} \quad (3)$$

其中，$makespan_{\text{own}}(G_m)$表示 G_m 单独调度的调度长度，$makespan_{\text{multi}}(G_m)$表示 G_m 在多 DAG 中调度的调度长度。基于 $slowdown(G_m)$计算多 DAG 系统调度的不公平性，其计算公式为

$$unfariness(GS)=\sum_{G_m\in GS}|slowdown(G_m)-\frac{1}{|GS|}\sum_{G_m\in GS}slowdown(G_m)| \quad (4)$$

由于选择任务调度是 *slowdown* 驱动的，因此算

表 1 DAG-A 和 DAG-B 中各任务在处理器上的计算开销

处理器	DAG														
	DAG-A										DAG-B				
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	B_1	B_2	B_3	B_4	B_5
p_1	14	13	11	13	12	13	7	5	18	21	4	9	18	21	7
p_2	8	19	13	8	13	16	15	11	12	7	5	10	17	15	6
p_3	16	18	19	17	10	9	11	14	20	16	6	11	16	19	5

法的关键思想是 *slowdown* 的更新。此算法虽然在每分配一个任务后通过更新 *slowdown* 值来保证 DAG 之间的公平性，但还是会产生不公平性的问题。例如，在 DAG-A 和 DAG-B 中，它们各自的 *slowdown* 值相等并且值都是 1，但是 DAG-A 中就绪任务的向上排序值大于 DAG-B，根据策略，则会选择 DAG-A 中的就绪任务调度，调度完后，DAG-A 的 *slowdown* 还是 1，则会继续选择 DAG-A 中的就绪任务进行调度，从而造成 DAG-B 中的就绪任务得不到分配处理器的机会，这样在调度开始处，就出现对后调度的 DAG 的不公平问题。田国忠[11]等将 Fairness 算法改成动态调度算法 E-Fairness，除了对新提交的 DAG 优先调度一次外，没有其他改进方法。但由于还是 *slowdown* 值驱动选择任务，因此同样会出现较高的不公平性。而且 Fairness 算法和 E-Fairness 算法在每次分配一个任务后需要更新 *slowdown* 值并对各 DAG 按升序排列，造成算法的复杂度较高，特别是在 DAG 个数很多的情况下，算法的效率更低。

3.2　多 DAG 任务调度中的轮转调度

轮转调度(round-robin)[17,18]由于具有较好的公平性，在数据通信网络和多处理器调度中得到了广泛的应用。LI[17]等提出的 DWRR(distributed weighted round-robin)算法则让大规模多处理器系统的调度具有更好的调度效率和更少的延迟；MOHANTY[18]等提出的 FBDRR(priority based dynamic round robin)等算法则在减少等待时间和响应时间上具有更好的性能。轮转调度在实现公平调度上具有优势，并且具有较高的调度效率。

HSU 等[12]提出了基于轮转调度的在线公平调度 OWM(online workflow management)算法，在 OWM 中，其策略是从每一个 DAG 中选择一个向上排序值 $rank_u(n_i)$ 最大的就绪任务放入 DAG 就绪队列。然后从 DAG 就绪队列中轮转选择最大向上排序值的任务并选择具有最小完成时间的处理器准备调度。此算法的轮转调度策略是优先调度 $rank_u(n_i)$ 最大的就绪任务，而短 DAG 就绪任务的 $rank_u(n_i)$ 相比长 DAG 就绪任务的 $rank_u(n_i)$ 总是要短，因此对短 DAG 造成了明显的不公平性，策略标准过于单调。

ARABNEJAD 等[13]提出了基于轮转调度的 FDWS(fairness dynamic workflow scheduling)算法，该算法也是在线调度，其策略也是从每一个 DAG 中选择一个向上排序值最大的就绪任务放入就绪队列，然后从就绪队列中不再选择具有向上排序值的任务，而是根据各个 DAG 剩余的未调度任务的比例及其关键路径长度定义了一个 $rank_r$ 值，其计算公式为

$$rank_r(n_i, G_m) = \frac{1}{PRT_{G_m}} + \frac{1}{CPL_{G_m}} \tag{5}$$

其中，*PRT* 表示剩余任务数的百分比，*CPL* 表示关键路径长度。此算法在考虑插入的基础上将任务分配具有最小完成时间的处理器，但由于短 DAG 的 $rank_r$ 相比长 DAG 总是要长，策略标准同样较单调，因而此算法对长 DAG 造成了明显的不公平性。虽然 OWM 和 FDWS 都是在线调度，但只要多个 DAG 同时发生，也就简化成了离线调度。

与单 DAG 任务调度的最大不同之处在于多 DAG 任务调度必须要保证每个 DAG 中的每个任务都能够及时被调度。表 2 总结出了目前多 DAG 任务调度算法的特点，其中，M_HEFT 算法即将多个 DAG 合成一个 DAG 后，再使用 HEFT 算法调度。

综上所述，现有多 DAG 任务调度算法并不能适应于当今汽车电子系统的异构化、网络化和复杂化特征，主要体现在：*slowdown* 值驱动的多 DAG 公平任务调度算法复杂度高，且容易在开始调度时就出现较高的不公平性；轮转调度的多 DAG 公平任务调度算法的策略标准过于单调而易导致不公平性；通信开销已成为影响调度性能的主要瓶颈，但缺乏相应的解决方案。汽车电子系统是一个典型的混合关键级嵌入式系统，既需要确保实时性又要降低调度长度。为此，本文旨在通过采用减少通信开

表 2　　4 种多 DAG 任务调度算法比较

算法特点	算法			
	M_HEFT	Fairness(E-Fairness)	OWM	FDWS
插入法	是	是	否	是
公平性	对短 DAG 不公平	对后调度 DAG 不公平	对短 DAG 不公平	对长 DAG 不公平
算法复杂度	$O(n^2\times d\times p)$	$O(n^2\times d\times p+ d^3\times p)$	$O(n^2\times d\times p)$	$O(n^2\times d\times p)$

销的轮转调度的任务优先级公平排序标准以及综合考虑通信开销、完成时间和拓扑结构的处理器选择标准。提出相应的减少通信开销、降低调度长度和满足安全关键与实时性的多 DAG 离线任务调度算法，以适应于汽车电子系统的异构化、网络化和复杂化需求。

4 调度算法

下面通过图 1 和表 1 所示的多 DAG 任务调度模型来说明本文所提出的调度方法。这个多 DAG 调度模型的复杂度、结构及各项参数与文献[11]使用的实例完全一样。

4.1 任务优先级排序

1)DAG 内的任务优先级排序

由于 E-Fairness[11]、OWN[12]和 FDWS[13]等多 DAG 任务调度算法使用经典的任务优先级排序 $rank_u(n_i)$中采用平均值 $\overline{w_i}$ 作为计算开销，实际上将处理器的异构特性消除，演变成同构计算。本文考虑异构计算特性，认为每个任务在每个处理器上都要有相应的向上排序值 $rank_{\mathrm{upd}}(n_i, p_k)$，计算公式为

$$\begin{cases} rank_{\mathrm{upd}}(n_{\mathrm{eixt}}, p_k) = w_{\mathrm{exit},k} \\ rank_{\mathrm{upd}}(n_i, p_k) = \max\limits_{n_j \in succ(n_i)} \{rank_{\mathrm{upd}}(n_j, p_k) + w_{i,k} + c_{i,j}\} \end{cases} \tag{6}$$

任务的出度也是影响任务优先级排序的因素。直接后继节点更多的任务如果不优先执行，则其所有后继节点都不能就绪，直接或间接地加大了 DAG 的调度长度。所以把任务的出度当作计算向上排序值的因子。

$$rank_{\mathrm{upd}}(n_i) = outd(n_i) \cdot \frac{1}{|P|} \cdot \sum_{k \in P} rank_{\mathrm{upd}}(n_i, p_k) \tag{7}$$

这样得出的 $rank_{\mathrm{upd}}(n_i)$ 更符合异构化特征。依据图 1 和表 1 提供的多 DAG 实例，可计算出每个任务的向上排序值 $rank_{\mathrm{upd}}(n_i,p_k)$和 $rank_{\mathrm{upd}}(n_i)$，如表 3 所示。

2) 多 DAG 任务公平性优先级排序

定义 3 通信开销权值(COW, communication overhead weight)。任务 n_i 的通信开销权值 $cow(n_i)$表示此任务与其所有直接前驱可能产生的最大通信开销之和，其计算公式为

$$\begin{cases} cow(n_{\mathrm{entry}}) = 0 \\ cow(n_i) = \sum\limits_{n_j \in pred(n_i)} c_{j,i} \end{cases} \tag{8}$$

slowdown 值驱动的多 DAG 公平任务调度算法复杂度高，不公平性也高，特别是在多 DAG 规模很大的情况下，算法的效率更低，因此不适应于复杂化的汽车电子系统。本文也采用公平性较好的轮转调度，针对多 DAG 系统 $GS=\{G_1,G_1,\cdots,G_m\}$，首先分别从各 DAG 就绪任务中取出一个 $rank_{\mathrm{upd}}(n_i)$最大的就绪任务放入 *REDAY_QUEUE*。对于 *REDAY_QUEUE* 队列中的就绪任务，如果其通信开销权值较大，说明其容易与其直接前驱节点产生更大的通信开销；如果此任务优先执行，则处理器空闲槽出现的几率大增，从而造成调度长度过长。

汽车电子系统的异构化和网络化使通信数据剧增，各总线的带宽又受限，再加上调度产生的大量开销，则使网络负载不堪重负，因此减少通信开销成为调度的主要目标之一，因此，本文优先调度通信开销权值较小的任务，则会尽可能地减少空闲槽出现的几率。本文提出基于通信开销权值的轮转调度的公平排序标准，具体策略为：

(a) 计算每个 DAG 中每个任务的向上排序值 $rank_{\mathrm{upd}}(n_i)$；

(b) 分别从各 DAG 中取出一个 $rank_{\mathrm{upd}}(n_i)$最大的就绪任务，放入到 DAG 的就绪队列 *REDAY_QUEUE*；

(c) 基于通信开销权值，对 *REDAY_QUEUE* 升序排序；

表 3 DAG-A 和 DAG-B 中的向上排序值

向上排序值	DAG														
	DAG-A										DAG-B				
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	B_1	B_2	B_3	B_4	B_5
$rank_{\mathrm{upd}}(n_i, p_1)$	105	79	79	86	75	67	45	39	50	21	44	20	33	38	7
$rank_{\mathrm{upd}}(n_i, p_2)$	95	69	75	66	56	62	39	31	30	7	42	20	31	31	6
$rank_{\mathrm{upd}}(n_i, p_3)$	114	81	86	87	70	67	44	43	47	16	55	20	30	34	5
$rank_{\mathrm{upd}}(n_i)$	523.5	150.6	80	159.4	67	65.3	42.7	27.7	42.3	0	141	20	30	34	0

(d) 从 *REDAY_QUEUE* 中轮转选择具有最小通信开销权值的任务分配处理器，直到 *REDAY_QUEUE* 为空再重复步骤（b）。

4.2 处理器选择

相比任务优先级排序，处理器选择则更加复杂。例如，DAG-B 中，如果将所有任务分配到同一个处理器，那么其总通信开销为 0，但使调度长度最大串行化；反之，如果将任务负载均衡的分配到相应处理器，不但不一定使调度长度最小化，而且可能消耗较大的通信开销。

1) 处理器选择值

定义 4　最早开始时间（EST, earliest start time）。$est(n_i, p_k)$表示在处理器 p_k 上，从入口任务 n_{entry} 到任务 n_i 的最长路径长度(不包含 n_i 本身的计算时间)，也就是在处理器 p_k 上，n_i 最早可能开始执行的时间，计算公式为

$$\begin{cases} est(n_{\text{entry}}, p_k) = 0 \\ est(n_i, p_k) = \max\{avail[k], \max\limits_{n_j \in pred(n_i)} \{aft(n_j^k) + c_{j,i} \cdot x_{j,i}\}\} \end{cases} \tag{9}$$

其中，$c_{j,i}$ 为任务 n_j 和任务 n_i 的通信开销，$x_{j,i}$ 取值为 0 或 1，$n_j \in pred(n_i)$，若任务 n_j 和任务 n_i 分配在同一处理器上则 $x_{j,i}$ =0，否则 $x_{j,i}$ =1；$avail[k]$ 表示处理器 p_k 获得的最早可用时间；$aft(n_i^k)$ 表示任务 n_i 的实际完成时间且 n_i 被分配在处理器 p_k 执行，由此公式可知当 n_i 为出口任务时，$aft(n_{\text{exit}}^k)$ 为 DAG 的调度长度，因此 $aft(n_j^k)$ 影响 DAG 的调度长度。任务 n_i 在处理器 p_k 的最早完成时间 $eft(n_i, p_k)$ 表示为

$$eft(n_i, p_k) = est(n_i, p_k) + w_{i,k} \tag{10}$$

E-Fairness、OWN 和 FDWS 等多 DAG 任务调度算法都以最早完成时间 $eft(n_i, p_k)$ 作为分配处理器的策略，一方面以“向下看”的角度综合考虑了调度长度和通信开销；但另一方面，忽略了 DAG 的拓扑结构对调度长度的影响，即还需要以“向上看”的角度考虑调度完相应任务后，此任务距离出口的时间和通信开销。用处理器选择值 $select(n_i, p_k)$ 代替 $eft(n_i, p_k)$，计算公式为

$$select(n_i, p_k) = eft(n_i, p_k) \cdot (rank_{\text{upd}}(n_i, p_k) - w_{i,k}) \tag{11}$$

在同构计算环境下，$rank_{\text{upd}}(n_i, p_k)$ 和 $w_{i,k}$ 对每个处理器而言都是相等的，不需要考虑，这也说明了 E-Fairness、OWN 和 FDWS 算法在处理器分配上也是在同构计算环境下进行的。而 $select(n_i, p_k)$ 的计算则从“向下看”和“向上看”的角度，综合考虑调度长度和通信开销对处理器分配的影响。

2) 插入分配法

正如图 2(c)所示，多 DAG 任务调度中，可将符合插入条件的任务插入到所有因优先级约束和通信开销造成的处理器空闲槽中，以提高处理器利用率，并能降低调度长度。插入分配过程为：

(a) 记录每个处理器的空闲时间段，用 *SLOT_SET*(p_k)表示 p_i 的空闲槽集合，每个处理器都有一个空闲槽集合；

(b) 在对任务 n_i 进行处理器分配时，遍历所有处理器，搜寻满足[$est(n_i, p_k)$，$eft(n_i, p_k)$] 属于 *SLOT_SET* (p_k)的处理器空闲段；

(c) 如果只存在唯一的空闲段，则直接插入的；如果存在多个处理器的空闲段，则选择选择值最小处理器插入，并更新相应的 *SLOT_SET*。

为此，得出本文的多 DAG 处理器选择标准：将从 DAG 的就绪队列 *REDAY_QUEUE* 中选择的任务，在考虑插入法的基础上分配到具有最小选择值的处理器上调度，直到 *REDAY_QUEUE* 中的任务全部分配到相应的处理器。

4.3 公平调度算法

定义 5　DAG 通信开销 (DOC, DAG communication overhead)。G_m 中具有直接优先级约束的所有任务因没有分配在同一个处理器而消耗的通信开销之和，即

$$dco(G_m) = \sum_{n_i \in N} \sum_{n_j \in pred(n_i)} (c_{j,i} \cdot x_{j,i}) \tag{12}$$

DAG 本身可能产生的最大通信开销则为

$$dcom(G_m) = \sum_{n_i \in N} \sum_{n_j \in pred(n_i)} c_{j,i} \tag{13}$$

定义 6　多 DAG 通信开销率(MDCOR, multiple DAGs communication overhead ratio)。指多 DAG 中，所有 DAG 的通信开销之和与其可能产生的最大通信开销之和的比值。那么，由 M 个 DAG 组成的多 DAG 系统 $GS=\{G_1, G_1, \cdots, G_m\}$ 调度完成所付出的通信开销率表示为

$$mdcor(GS) = \frac{\sum_{m=1}^{|M|} (dco\ (G_m))}{\sum_{m=1}^{|M|} (dcom(G_m))} \tag{14}$$

基于 4.1 节提出的多 DAG 任务优先级排序标准和 4.2 节提出的多 DAG 处理器选择标准，本节提出以降低调度长度和减少通信开销为目标的，适用于异构网络化汽车电子系统中的多 DAG 离线公平任务调度算法 MDOFTS(multiple DAG off-line and fairness task scheduling)。

算法 1 MDOFTS 算法

for(DAG 个数)

计算每个 DAG 中任务的向上排序值 $rank_{upd}(n_i, p_k)$，$rank_{upd}(n_i)$ 与通信开销权值 $cow(n_i)$；

end for

计算出拥有的最大任务数的 DAG，并将个数设置为 n

for(var i=1: n)

for(DAG 个数)

分别从各 DAG 中取出一个向上排序值最大的就绪任务，放入到任务的就绪队列 *REDAY_QUEUE*

end for

for(DAG 个数)

基于公平轮转调度，从 *REDAY_QUEUE* 中选择具有最小通信开销权值 $cow(n_i)$的任务 n_i

获取任务 n_i 所在 DAG 为 G_m

计算 n_i 在每个处理器上的选择值 *select* (n_i, p_k)

考虑插入法的基础上将 n_i 分配到选择值 *select* (n_i, p_k)最小的处理器上

更新 G_m 的通信开销 $dco(G_m)$

更新 GS 通信开销率 $mdcor(GS)$

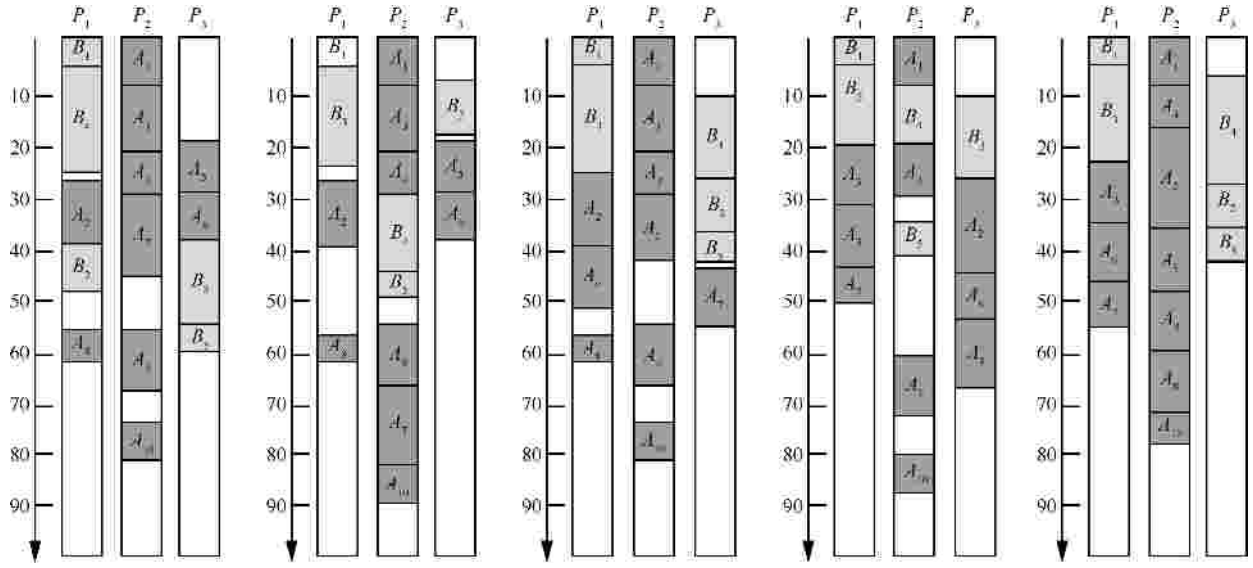
end for

end for

MDOFTS 算法的时间复杂度为 $O(n^2 \times d \times p)$，其中 n 为拥有最多任务数的 DAG 所包含的任务数，d 为 DAG 个数，p 为处理器个数。

证明 调度完所有 DAG，遍历任务次数的时间复杂度为 $O(n)$，遍历所有 DAG 的时间复杂度为 $O(d)$；计算 n_i 的 *select* (n_i, p_k)需要遍历它的前驱和各个处理器的时间复杂度为 $O(n \times p)$。所以 MDOFTS 总的算法时间复杂度为 $O(n^2 \times d \times p)$，证毕。

图 2 为 5 种算法的调度 Gannt 图，表 4 为相应计算结果。DAG-A 和 DAG-B 的总的通信开销为 241 和 35。从结果可以看出，MDOFTS 算法调度长度为 78，其他算法的调度长度都在 81 以上，MDOFTS 算法优势比较明显；在通信开销率方面，MDOFTS 算法仅为 0.2645，而其他算法的通信开销都在 0.529 0 以上，MDOFTS 算法在减少通信开销方面的优势相当明显；公平性方面，尽管 MDOFTS 算法的不公平性相比 OWN(off-line)和 FDWS (off-line)要高一点，但是任务优先级排序结果相比这 2 个算法更具有公平性。因此，实例结果显示，MDOFTS 算法在没有提高算法时间复杂度的情况



(a) M_HEFT (b) E-Fairness(off-line) (c) OWN((off-line) (d) FDWS(off-line) (e) MDOFTS

图 2 5 种算法公平调度 DAG-A 和 DAG-B 的 Gannt 图

表 4　5 种算法公平调度 DAG 结果比较

算法结果	算法				
	M_HEFT	E-Fairness(off-line)	OWN(off-line)	FDWS(off-line)	MDOFTS
任务优先级	$A_1,A_3,A_4,A_2,A_5,A_6,A_9,A_7,B_1,A_8,B_4,B_3,B_2,A_{10},B_5$	$A_1,B_1,A_3,A_4,A_2,A_5,A_6,B_4,A_9,A_7,A_8,A_{10},B_3,B_2,B_5$	$A_1,B_1,A_3,B_4,A_4,B_3,A_2,B_2,A_5,B_5,A_6,A_9,A_7,A_8,A_{10}$	$B_1,A_1,B_4,A_3,B_3,A_4,B_2,A_2,B_5,A_5,A_6,A_9,A_7,A_8,A_{10}$	$A_1,B_1,B_4,A_4,B_3,A_2,B_2,A_3,A_5,B_4,A_6,A_7,A_9,A_8,A_{10}$
调度长度 *DAG*-A	81	89	81	87	78
调度长度 *DAG*-B	59	49	42	40	41
不公平性	0.389 831	0.175 418	0.118 17	0.008 046	0.121 951
通信开销 *DAG*-A	127	127	128	141	58
通信开销 *DAG*-B	20	19	21	22	15
通信开销率	0.532 6	0.529 0	0.539 9	0.605 1	0.264 5

下，不仅保证了调度长度更短，而且能够显著减少通信开销，还具有很好的公平性。

4.4　优先级调度算法

尽管公平性是需要关注的重点，但是对于混合关键级嵌入式系统，每个子系统都有相应的关键级划分，例如底盘控制等安全关键子系统，有严格的截止期限要求，如果不能在规定的时间内完成，执行结果也就毫无意义。因此需要优先执行安全关键的 DAG 应用，这就是 DAG 之间存在的混合关键优先级。针对上述情况，本节提出多 DAG 离线优先级任务调度 MDOPTS(multiple DAGs off-line and priority task scheduling)算法，调度策略如下。

1) 对于优先级高的 DAG 中的所有任务都要优先于优先级低的 DAG 中的所有任务。只有当优先级高的 DAG 中的所有任务分配处理器后，优先级低的 DAG 中的任务才能分配处理器。

2) 优先级低的 DAG 中的所有任务在考虑插入法的基础上按最小选择值分配处理器。其选择值的计算公式需调整为式(15)。因为低优先级 DAG 的入口节点 n_{entry} 的开始时间得到了一定的推迟，其所有后继节点的时间计算都要做相应调整，即

$$select(n_i, p_k) = (eft(n_i, p_k) - est(n_{\text{entry}})) \cdot (rank_{\text{upd}}(n_i, p_k) - w_{i,k}) \quad (15)$$

算法 2　MDOPTS 算法

for(DAG 个数) do

　　计算每个 DAG 中任务的向上排序值 $rank_{\text{upd}}(n_i,p_k)$, $rank_{\text{upd}}(n_i)$与通信开销权值 $cow(n_i)$及任务个数，并将任务放入各 DAG 的任务队列

end for

对所有 DAG 按给定的关键优先级降序放入关键级 DAG 队列

while 有 DAG 可以调度 do

　　从关键级 DAG 队列中取出未调度完成且具有最大优先级的 DAG

　　while 当前 DAG 的任务队列有任务可以调度 do

　　选择队列中具有最大 $rank_{\text{upd}}(n_i)$的就绪节点 n_i

　　计算 n_i 在每个处理器上的最早完成时间 $eft(n_i,p_k)$和 $select(n_i,p_k)$

　　考虑插入法的基础上将 n_i 分配到选择值 $select\ (n_i,p_k)$最小的处理器上

　　更新 G_m 的通信开销 $dco(G_m)$

　　更新 GS 通信开销率 $mdcor(GS)$

　　标记 n_i 为已调度任务

　　end while

　　标记当前 DAG 已调度

end while

MDOPTS 算法的时间复杂度为 $O(n^2\times d \times p)$

证明　调度完所有 DAG，遍历所有 DAG 的时间复杂度为 $O(d)$; 调度就绪队列中的任务，需要遍历一次的时间复杂度为 $O(n)$；遍历处理器并计算任务在每个处理器上的最早完成时间，其时间复杂度为 $O(n\times p)$。所以 MDOPTS 总的算法时间复杂度为 $O(n^2\times d \times p)$，证毕。

4.5　自适应调度算法

MDOPTS 算法虽然能够满足实时性，但却使低优先级的 DAG 没能及时分配到处理器，从而加大了整个多 DAG 的调度长度，造成性能明显下降。然而实时系统并不是必须要求优先级高的 DAG 中的所有任务都要优先于优先级低的 DAG 中的所有

任务，而是只要在截止期限内调度完成优先级高的 DAG 中的所有任务即可。本节提出多 DAG 离线任务自适应调度 MDOATS (multiple DAG off-line and priority task scheduling) 算法，在确保实时性的前提下，降低整个多 DAG 的调度长度和减少通信开销。调度策略如下。

1) 通过 MDOPTS 算法保证某些 DAG 在截止期限内完成，通过 MDOFTS 算法降低多 DAG 的调度长度和减少通信开销。

2) 用 MDOFTS 预调度多 DAG 系统，并判断高优先级系统是否满足截止期限，如果满足，则直接用 MDOFTS 调度；否则，用 MDOPTS 调度高优先级 DAG 中的第一个就绪任务。如此重复，直到所有 DAG 全部调度完毕。即依据截止期限和公平调度结果自适应的采用 MDOPTS 算法和 MDOFTS 算法。

算法 3 MDOATS 调度算法

for(DAG 个数) do

计算每个 DAG 中任务的向上排序值 $rank_{upd}(n_i, p_k)$，$rank_{upd}(n_i)$ 与通信开销权值 $cow(n_i)$，并放入各 DAG 的任务队列

end for

对所有 DAG 按关键优先级降序并放入 DAG 队列

while 有 DAG 可以调度 do

从 DAG 队列中取出未调度完成且具有最大优先级的 DAG 为 G_x，其截止期限为 $deadline(G_x)$

用 MDOFTS 算法单调调度 G_x，计算出其调度长度为 $makerspan(G_x)$

if $makerspan(G_x) \le deadline(G_x)$

while(G_x 中有任务可以调度) do

用 MDOFTS 算法预调度多 DAG 系统，并更新 G_x 的调度长度 $makerspan(G_x)$

从 G_x 中取出向上排序值最大的就绪任务

if $makerspan(G_x) \le deadline(G_x)$

用 MDOFTS 调度算法调度此任务

else

用 MDOPTS 调度算法调度此任务

end if

end while

else

该多 DAG 系统不可调度，调度失败

end if

end while

MDOATS 算法的时间复杂度为 $O(n^3 \times d^2 \times p)$

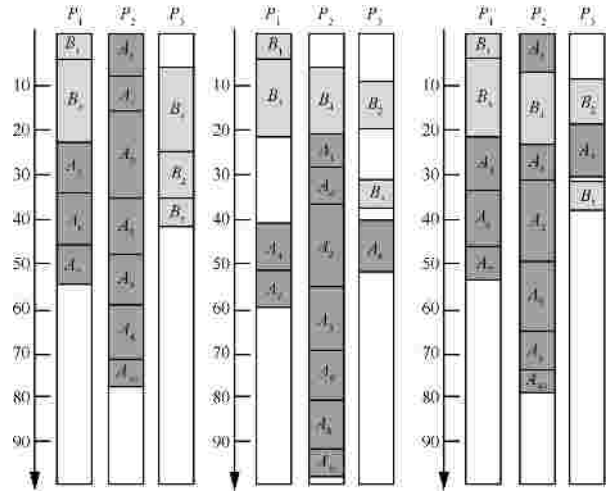
证明 调度完所有 DAG，遍历所有 DAG 的时间复杂度为 $O(d)$；调度就绪队列中的任务，需要遍历一次的时间复杂度为 $O(n)$；用 MDOFTS 和 MDOPTS 调度的算法复杂度都为 $O(n^2 \times d \times p)$。所以 MDOATS 总的算法时间复杂度为 $O(n^3 \times d^2 \times p)$，证毕。

图 3 为 DAG-B 截止期限为 40 的情况下，MDOFTS、MDOPTS 和 MDOATS 算法调度的 Gannt 图，表 4 为相应计算结果。从结果可以看出，MDOFTS 由于采用了公平轮转调度，其多 DAG 的调度长度最短，但是 DAG-B 的截止期限要求是 40，而 MDOFTS 调度的结果为 41，因此不能满足 DAG-B 的实时性，用此算法调度多 DAG 将会失败；MDOPTS 针对 DAG-B 的截止期限要求，优先调度 DAG-B，尽管满足了 DAG-B 的实时性，但由于调度公平性差，因而造成多 DAG 的调度长度过长，使 DAG 的运行时间过长，造成性能下降；MDOATS 综合考虑 MDOFTS 算法和 MDOPTS 算

表 5 3 种算法调度 DAG 结果比较($deadline_b$=40)

算法结果	算法		
	MDOFTS	MDOPTS	MDOATS
优先级	$A_1,B_1,B_4,A_4,B_3,A_2,B_2,A_3,A_5,B_4,A_6,A_7,A_9,A_8,A_{10}$	$B_1,B_4,B_3,B_2,B_5,A_1,A_3,A_4,A_2,A_5,A_6,A_9,A_7,A_8,A_{10}$	$B_1,A_1,B_4,A_4,B_3,A_2,B_2,A_3,B_5,A_5,A_6,A_7,A_9,A_8,A_{10}$
调度长度 DAG-A	78	99	80
调度长度 DAG-B	41	36	38
不公平性	0.121 951	0.212 121	0.034 868
通信开销 DAG-A	58	58	82
通信开销 DAG-B	15	25	25
通信开销率	0.264 5	0.300 7	0.387 9
时间复杂度	$O(n^2 \times d \times p)$	$O(n^2 \times d \times p)$	$O(n^3 \times d^2 \times p)$

法的特点，采用自适应调度策略，既能降低调度长度，又满足截止期限，因此很适合实时系统应用。尽管 MDOATS 算法的时间复杂度较高，但对于同时发生的多 DAG 调度属于编译时调度，因此并不会影响运行时性能。而且当优先级 DAG 数量不多时，MDOATS 非常接近于 MDOFTS，因为如果采用 MDOFTS 能满足所有多 DAG 的实时性，那么 MDOATS 的调度结果就是 MDOFTS 的调度结果。



(a) MDOFTS　(b) MDOPTS　(c) MDOATS

图 3　3 种算法调度 DAG-A 和 DAG-B 的 Gannt 图($deadline_b$=40)

5　实验

5.1　评价指标

本文采用加速比(speedup)[7,11]、通信开销率(MDCOR)、不公平性(unfairness)以及端到端最差响应时间(WCRT, worst-case response time) [19]作为评价指标。

加速比即在一个处理器上串行执行多 DAG 中的调度长度最大的 DAG 的所有任务使用的最少时间与其实际调度长度的比值。调度算法产生的加速比越大，说明算法越高效，加速比计算公式为

$$Speedup=\frac{\min\{\sum_{n_i\in V_{G\max}} w_{i,k}\}}{makespan(G_{\max})} \quad (16)$$

多 DAG 的通信开销率采用式(14)计算，通信开销率越低，说明算法越高效。不公平性采用式(4)计算，不公平性越低，算法越高效。

在汽车电子系统中，消息集的 WCRT 定义为消息集的第一个消息所分配的 ECU 触发就绪到传输完毕到达最后一个消息所在 ECU 节点之间的时间段，WCRT 越短，算法越高效。

实验的硬件环境为一台具有奔腾双核处理器(3.2 GHz/2.0 GB RAM)的 Windows XP 机器上，所使用的软件工具有 Java 和 Highcharts，DAG 任务图生成工具 TGFF 3.5 (task graphs for free)[20]。根据不同参数生成各种特性不同的加权 DAG。多 DAG 系统的 DAG 个数为 gs={2,4,10,20,40,60,80,100}，关键级 sc={1,2,3,4}，G_m 的截止期限长度统一为 $makerspan(G_m)$的 90%。产生随机 DAG 的参数设置为任务个数 n={30,40,50,60,70,80,90,100}，最大出度 ß={1,2,3,4,5}，最大入度 ?={1, 2, 3, 4, 5}，任务在不同处理器上执行时间的差异度 ?={0.1, 0.5, 1.0}。假设 w_i 表示任务 n_i 的平均计算开销，那么 n_i 在处理器 p_k 上的计算开销可以通过公式产生而得，即

$$w_i(1-h/2)\le w_{i,k}\le w_i(1+h/2) \quad (17)$$

通过生成具有不同特征的大量随机 DAG，并在一个由 15 个异构多处理器芯片组成的网络计算系统中运行。

5.2　实验结果及分析

实验 1　针对多个 DAG 样本，探究加速比随任务数和 DAG 数变化的情况，每个数据点值是由 2 000 个实验次数得出的平均值。从图 4 和图 5 得知：1)随着系统规模增长，加速比都提高；2) MDOFTS 的平均 Speedup 都优于其他算法，分别达 20%以上，说明 MDOFTS 采用的公平轮转策略能够明显地缩短调度长度，而且选择处理器时综合“向上看”和“向下看”的原则，相比其他算法仅“向下看”的原则，优势明显。

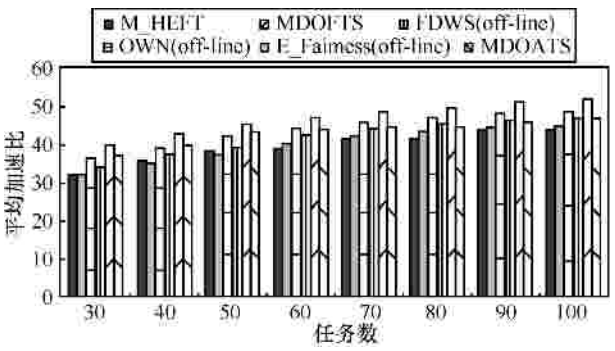


图 4　平均加速比随任务数变化

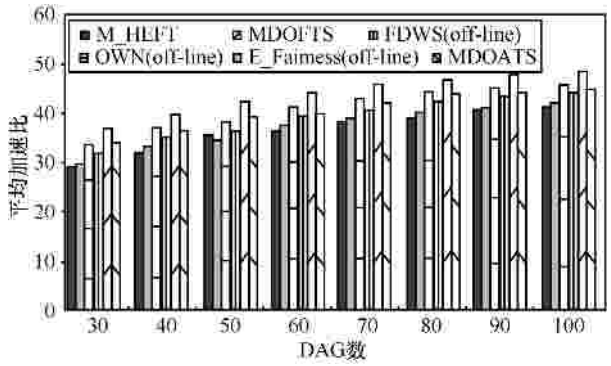


图5 平均加速比随DAG数变化

实验2 针对多个DAG样本，分析通信开销率随任务数和DAG数变化情况，如图6和图7所示：1）通信开销率随任务数和DAG数而提高，说明随着系统规模和复杂性增长，通信任务大增，造成通信开销加大；2）MDOFTS和MDOATS算法相比其他算法优势比较明显，通信开销率始终保持在12%~44%之间，在最好情况超过了50%，说明基于通信开销权值的轮转调度策略能够显著减少通信开销，相比其他算法调度目标更加明确，性能更加优越。

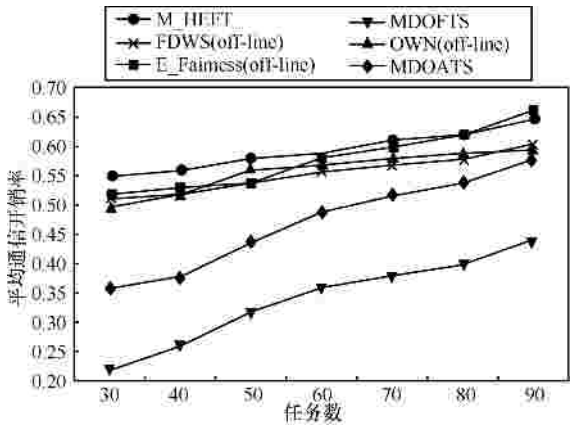


图6 平均通信开销率比随任务数变化

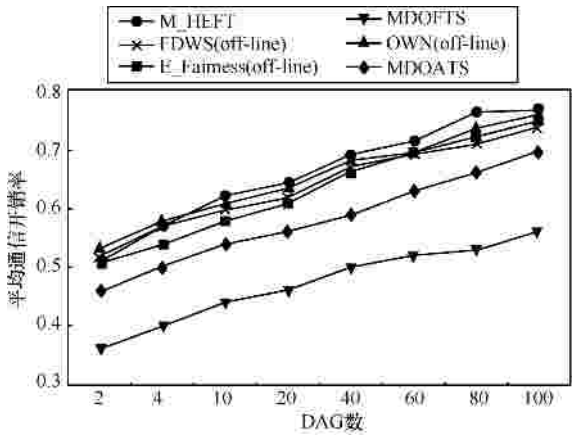


图7 平均通信开销率比随DAG数变化

实验3 分析不公平性随任务数和DAG数变化的情况，实验结果如图8和图9所示。可以看出在任务数或DAG数目较小时，MDOFTS和MDOATS算法优势并不明显；但随着DAG数目的增多，相比其他算法的优势分别达到20%以上，说明随着系统规模和复杂性增大，各DAG包含的任务数和属性不尽相同，使有些公平调度算法对长DAG或短DAG等造成一定的不公平性，而MDOATS的轮转调度策略不仅满足实时性，还具有较好的公平性。

实验4 在真实消息集环境下分析WRCT随消息任务数变化的情况，采用日本名古屋大学高田研究室提供的单个CAN网络的真实消息集[19]，该实验集包括65个消息任务，并且被分配到14个ECU之中，由于目前关于汽车电子网络的研究都是基于单个网络的情况，故本文将上述消息集分解为2个CAN网络消息子集，其中，DAG_1为33个，DAG_2为32个。实验结果如图10~图12所示，从结果可以看出，MDOFTS的WRCT算法最短，优于其他算法20%以上。MDOFTS和MDOATS算法的通信开销和不公平性也都优于其他算法。

以上4次实验的结果表明，本文所提出的相关多DAG离线任务调度算法在调度长度、通信开销和不公平性相比E-Fairness(off-line)、OWN(off-line)和FDWS(off-line)等算法都要优；在真实消息集环境下，具有更好的结果，因而能够很好地适应于异构网络化汽车电子系统的多DAG离线任务调度。

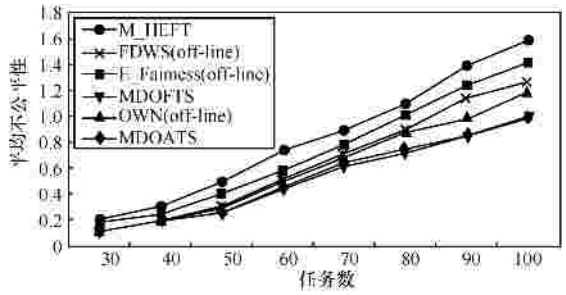


图8 平均不公平性比随任务数变化

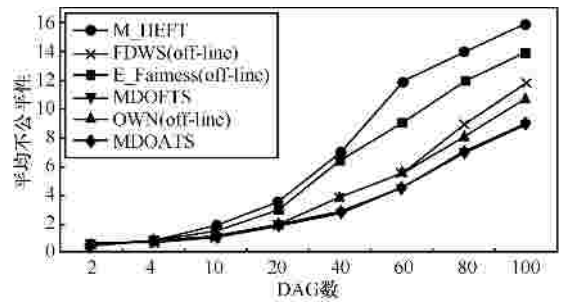


图9 平均不公平性比随DAG数变化

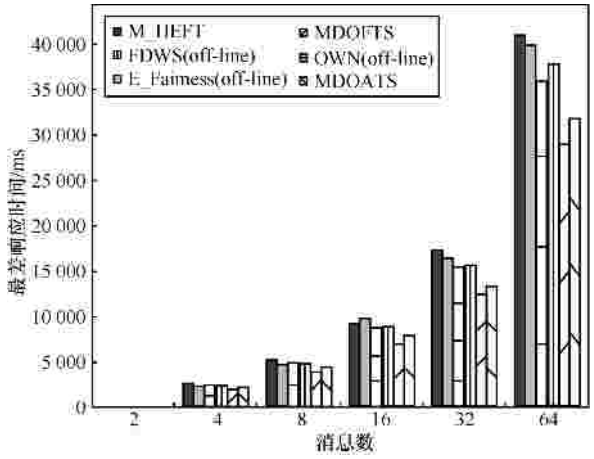


图 10　WCRT 随消息数变化

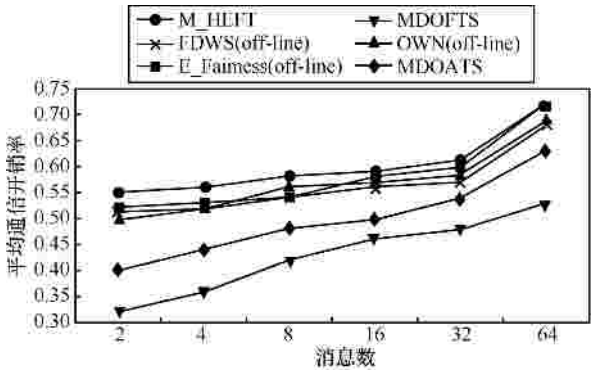


图 11　平均通信开销率随消息数变化

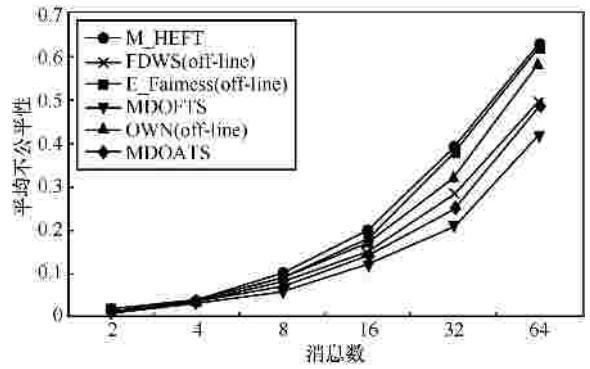


图 12　平均不公平性随消息数变化

6　结束语

本文面向异构网络化汽车电子系统领域进行调度问题研究，以多 DAG 离线任务模型为基础，分析了现有多 DAG 任务调度算法；然后提出基于通信开销权值的轮转调度公平任务排序标准和在考虑插入法的基础上将任务分配到具有最小选择值的处理器上作为处理器选择标准。综合这 2 个标准提出了面向异构网络化汽车电子系统中的多 DAG 离线公平任务调度 MDOFTS 算法；接着提出了以满足安全关键 DAG 的多 DAG 离线优先级任务调度算法 MDOPTS，并综合 MDOFTS 和 MDOPTS，提出多 DAG 离线自适应任务调度算法 MDOATS，在满足实时性的基础上提高调度性能。最后利用仿真实验将本文提出的算法与相关算法进行比较，得到本文所提出的算法在保证公平性的条件下，能够有效降低调度长度，极大地减少通信开销，产生更低的最差响应时间，具有更好的调度性能。

参考文献：

[1] BUCKL C, CAMEK A, KAINZ G, *et al*. The software car: building ICT architectures for future electric vehicles[A]. 2012 IEEE International Electric Vehicle Conference(IEVC)[C]. Kuching, Malaysia, 2012.1-8.

[2] FURST S. Challenges in the design of automotive software[A]. Proceedings of the Conference on Design, Automation and T in Europe[C]. Dresden, Germany, 2010. 256-258

[3] KONIK D. Development of the dynamic drive for the new 7series of the BMW group[J]. International Journal of Vehicle Design, 2002, 28(1):131-149

[4] Audi A8'10 electrical and network systems[EB/OL]. http://www.audionlinetraining.com, 2010.

[5] BARUAH S K, BURNS A, DAVIS R I. Response-time analysis for mixed criticality systems[A]. The 32nd IEEE Real-Time Systems Symposium[C]. Vienna, Austria, 2011.34-33.

[6] ALDERISI G, CALTABIANO A, VASTA G, *et al*. Simulative assessments of IEEE 802.1 Ethernet AVB and time-triggered Ethernet for advanced driver assistance systems and in-car infotainment[A]. Vehicular Networking Conference(VNC)[C]. Seoul, Republic of Korea, 2012.187-194.

[7] TOPCUOGLU H, HARIRI S, WU M. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 260-274.

[8] 谢勇, 李仁发, 阮华斌等. 最优的 FlexRay 静态段配置算法[J]. 通信学报, 2012, 33(11):33-40.
XIE Y, LI R F, RUAN H B, *et al*. Optimal Configuration algorithm for static segment of FlexRay[J]. Journal on Communications, 2012, 33(11):33-40.

[9] HÖNIG U, SCHIFFMANN W. A meta-algorithm for scheduling multiple DAGs in homogeneous system environments[A]. The 18th International Conference on Parallel and Distributed Computing Systems[C]. Dallas, USA, 2006.147-152.

[10] ZHAO H N, SAKELLARIOU R. Scheduling multiple DAGs onto heterogeneous systems[A]. The 20th International Parallel and Distributed Processing Symposium[C]. Rhodes Island, Greece, 2006. 14.

[11] 田国忠, 肖创柏, 徐竹胜等. 异构分布式环境下多 DAG 工作流的

混合调度策略[J]. **软件学报**, 2012, 23(10):2720-2734.
TIAN G Z, XIAO C B, XU Z S, *et al*. Hybrid scheduling strategy for multiple DAGs workflow in heterogeneous system[J]. Journal of Software, 2012, 23(10):2720 -2734.

[12] HSU C C, HUANG K C, WANG F J. On line scheduling of workflow applications in grid environments[J]. Future Generation Computer Systems, 2011, 27(6):860-870.

[13] ARABNEJAD H, BARBOSA J. Fairness resource sharing for dynamic workflow scheduling on Heterogeneous Systems[A]. 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)[C]. Madrid, Spain, 2012. 633-639.

[14] KLOBEDANZ K, KOENIG A, MUELLER W. A reconfiguration approach for fault-tolerant flexray networks[A]. Design, Automation & Test in Europe Conference & Exhibition (DATE)[C]. Grenoble, France, 2011.1-6.

[15] KLOBEDANZ K, KOENIG A, MUELLER W, *et al*. Self-reconfiguration for fault-tolerant flexRay networks[A]. 2011 14th IEEE International Symposium on Distributed Computing Workshops(ISORCW)[C]. Newport Beach, CA, 2011.207-216.

[16] HAGRAS T, JANECEK J. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems[J]. Parallel Computing, 2005, 31(7):653-670.

[17] LI T, BAUMBERGER D, HAHN S. Efficient and scalable mul iprocessor fair scheduling using distributed weighted round- robin[J]. ACM Sigplan Notices, 2009, 44(4):65.

[18] MOHANTY R, BEHERA H S, PATWARI K, *et al*. Priority based dynamic round robin (PBDRR) algorithm with intelligent ti slice for soft real time systems[J]. International Journal of Advanced Computer Seience and Applications, 2011, 2(2):46-50.

[19] CHEN Y, ZENG G, RYO K, *et al*. Effects of queueing jitter on worst-case response times of CAN messages with offsets[A]. In Proc of the Embedded System Symposium in Japan[C]. Tokyo, Japan, 2012.119-126.

[20] DICK R P, RHODES D L, WOLF W. TGFF: task graphs for free[A]. Proceedings of the 6th International Workshop on Hardware/Software Codesign[C]. Seattle, USA, 1998.97-101.

作者简介：

谢国琪（1983-），男，湖南宁乡人，湖南大学博士生，主要研究方向为嵌入式与网络计算、汽车电子网络体系结构、物联网。

李仁发（1957-），男，湖南宜章人，湖南大学教授、博士生导师，主要研究方向为嵌入式与网络计算、汽车电子、无线网络、物联网、CPS。

杨帆（1985-），男，湖南益阳人，湖南大学博士生，主要研究方向为嵌入式与网络计算、汽车电子、嵌入式建模。

黄卫红（1971-），男，湖南湘阴人，湖南大学博士生，主要研究方向为嵌入式与网络计算、汽车电子、物联网。

（上接第 19 页）

[12] **石华**, **李建东**, **李钊**. **认知异构网络中基于克隆选择算法的动态频谱分配**[J]. **通信学报**, 2012, 33(7):59-66.
SHI H, LI J D, LI Z. Dynamic spectrum allocation based on clone selection algorithm in cognitive heterogeneous wireless networks[J]. Journal on Communications, 2012, 33(7):59-66.

作者简介：

金顺福（1966-），女，朝鲜族，内蒙古满洲里人，燕山大学教授、博士生导师，主要研究方向为网络资源分配与优化，排队论研究与应用等。

解洪亭（1987-），男，山东泰安人，山东中医药大学附属医院网络中心助理工程师，主要研究方向为无线网络中的频谱分配策略。

赵媛（1985-），女，河北秦皇岛人，燕山大学博士生，主要研究方向为认知无线网络技术、排队论研究与应用。