5J-04

物理時間駆動と論理時間駆動に対応したリソース管理機能を有するリアルタイム OS

高堂 航希† 横山 孝典† 兪 明連†
東京都市大学‡

1. はじめに

複数の組み込みコンピュータをネットワーク接続した分散型組み込み制御システムは，実時間に基づいた処理が要求される．我々は，通信時間の変動するネットワークを用いたシステムでも時間駆動アーキテクチャ[1]と同等の動作を実現することを目的として，物理時間と論理時間を用いた分散処理環境を提案した[2]．本分散処理環境は，入出力処理を行うタスクを現実の物理時間によって管理し，それ以外のタスクを仮想的な論理時間によって管理することで，通信時間の変動により生じる遅延やジッタを許容可能とした．そして，固定優先度スケジューリングと EDF(Earliest Deadline First)スケジューリング[3]が共存したリアルタイム OS（RTOS）を導入したが，リソース管理機能は固定優先度スケジューリングにしか対応していなかったため，共有リソースを扱えないという問題があった．

本論文では，固定優先度スケジューリングを用いる物理時間駆動と EDF スケジューリングを用いる論理時間駆動に対応したリソース管理機能を有する RTOS を提案する．

2. 分散処理環境

2.1 論理時間を導入した時間駆動分散処理環境

本分散処理環境では，入出力タスクは物理時間に同期したジッタの少ない時間駆動アーキテクチャに基づいて管理する．一方，算出タスクはメッセージ受信イベントで起動し論理時間に基づいて管理することで，通信時間の変動を許容する．

論理時間は非同期動作する算出タスクを時間駆動アーキテクチャと等価的に扱うための仮想的な時間である．論理時間における起動時刻を論理起動時刻と呼ぶ．入出力タスクの論理起動時刻は物理時間の起動時刻と等しいものとする．各タスクのジョブがメッセージを送信するときに，そのジョブの論理起動時刻をタイムスタンプとして付加する．受信側のタスクはタイムスタンプの論理遅延時間を加えることで論理起動時刻を決定する．

図 1 に本分散処理環境を用いたシステムの動作例を示す．この例では 4 つのノード上に入力タスク，2 つの算出タスク，出力タスクが分散配置されている．タスクの周期は全て 10 である．通信時間の変動によって算出タスクの起動時刻にジッタが発生しているが，論理時間上ではジッタが発生していないものとして扱う．これにより時間駆動アーキテクチャと同等の動作と見なすことができる．

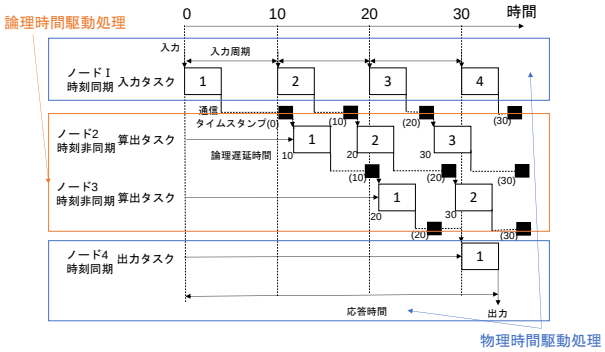

図 1：論理時間を導入した時間駆動分散処理の動作例

2.2 リアルタイム OS

本分散処理環境の RTOS は，OSEK OS 仕様[4]に基づく TOPPERS/ATK1[5]をベースに拡張したもので，入出力タスクは固定優先度で，算出タスクは EDF でスケジューリングする．固定優先度で扱うタスクを固定優先度タスク，EDF で扱うタスクを EDF タスクと呼ぶ．

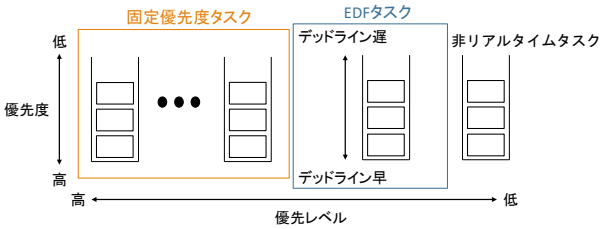

図 2：タスク管理用レディキュー

図 2 にタスク管理のためのレディキューを示す．実行待ち状態のタスクは優先レベル毎にレディキューに格納される．EDF タスクは固定優先度タスクより優先レベルが低いレディキューに格納され，デッドラインが早い順にソートされる．タスクは優先レベル順に実行され，同じ優先レベルのタスクはレディキューの先頭から順に実行される．

3. リソース管理方式

3.1 方針

OSEK OS のリソース管理用のシステムサービス GetResource()と ReleaseResource()を EDF タスクにも対応できるよう拡張する．排他区間を開始するには GetResource()を呼んでリソースを獲得し，排

A Real-Time Operating System with Resource Management Functions for Time-Triggered Systems Based on Physical Time and Logical Time
†Koki Takado , Takanori Yokoyama and Myungryun Yoo
‡Tokyo City University

他区間を終了するときには ReleaseResource()を呼んでリソースを解放する．固定優先度タスクを含むタスク間で共有するリソースについては OSEK 優先度上限プロトコルを，EDF タスク間で共有するリソースについては論理デッドラインを用いた EDF 向け優先度上限プロトコル[6]を使用する．

3.2 固定優先度タスク向け優先度上限プロトコル

固定優先度タスク向けのリソース管理機構である OSEK 優先度上限プロトコルは，リソースを獲得したタスクの優先レベルを一時的に上昇させることで共有リソースの排他制御を行う．その動作例を図 3 を用いて説明する．図の横軸は時間で，縦軸はタスクの優先度を表している．タスク A とタスク B は同じリソースを共有する．タスク A の優先レベルよりもタスク B の優先レベルの方が高いため，共有リソースの上限優先度はタスク B の優先レベルとなる．まず，タスク A が起動され実行を開始する．タスク A が共有リソースを獲得すると，その優先レベルを共有リソースの上限優先度レベルまで一時的に上昇させる．その後タスク B が起動されるが，プリエンプションは発生しない． タスク A がリソースを解放すると優先レベルが元に戻りタスク B にプリエンプトされる．

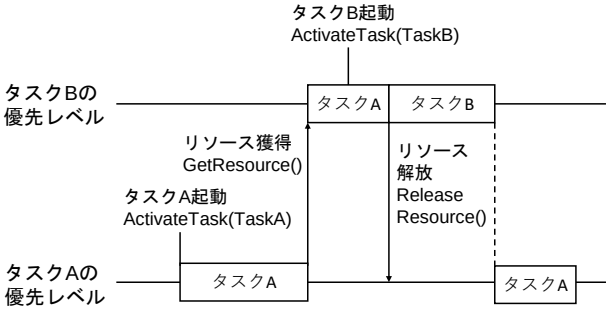

図 3：OSEK 優先度上限プロトコルの動作

3.3 EDF タスク向け優先度上限プロトコル

EDF タスク向け優先度上限プロトコルは，リソース獲得中のタスクがそのリソースを共有するより高い優先度のタスクの優先度を継承することで，共有リソースの排他制御を行う．その動作例を図 4 を用いて説明する．図の横軸は時間を，縦軸はタスクの優先度（デッドライン）を表す．タスク C，タスク D 共に EDF タスクであり，同じリソースを共有している．まず，タスク C が起動され，共有リソースを獲得する．同じリソースを共有するタスク D が起動されると，タスク D のデッドラインがタスク C のデッドラインより近いため，タスク C はタスク D のデッドラインを継承する．このためプリエンプションは発生しない．その後タスク C がリソースを解放すると，タスク C のデッドラインが元に戻り，よりデッドラインの近いタスク D にプリエンプトされる．優先度継承を行うため，タスク起動用のシステムコール ActivateTask()も拡張する．

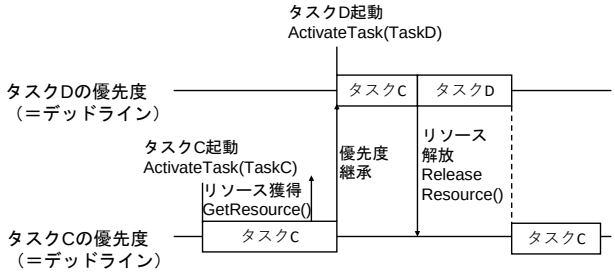

図 4：EDF タスク向け優先度上限プロトコルの動作

4. 実装と評価

実装にはマイクロコントローラ H8S/2638F を搭載した評価ボードを用いた．H8S/2638F は 256KB の ROM，16KB の RAM を内蔵しており，CPU のクロック周波数は 20MHz である．

拡張した 3 つのシステムコール ActivateTask(), GetResource()，ReleaseResource()の実行時間を表 1 に示す．処理の追加により実行時間が増加しているが，従来の RTOS と比較して最大で 3 割程度の増加であり，実用上問題ないと考える．

表 1：拡張した個所の実行時間

システムコール	実行時間[μsec]			
	従来の RTOS		本 RTOS	
	固定優先度	EDF	固定優先度	EDF
ActivateTask()	22.4	28.0	23.2	29.0
GetResource()	10.1	～	12.3	13.4
ReleaseResource()	11.0	～	12.3	14.9

5. おわりに

固定優先度スケジューリングを用いる物理時間駆動と EDF スケジューリングを用いる論理時間駆動に対応したリソース管理機能を提案し，提案した機能を有する RTOS を実装した．今後は，本 RTOS のスケジューラを非周期タスクに対応させるよう拡張する予定である．

謝辞

本研究で使用した TOPPERS/ATK1 の開発者に感謝する．本研究は JSPS 科研費 JP18K11225, JP21K11815 の助成を受けたものである．

参考文献

[1]. Kopetz, H., Should Responsive Systems be Event-Triggered or Time-Triggered?, IEICE Trans. Inf. & Syst., Vol.E76-D, No.11, pp.1325-1332(1993).
[2]. Ichimura, A., Yokoyama, T. and Yoo, M., A Time-Triggered Distributed Computing Environment for Cyber-Physical Systems Based on Physical Time and Logical Time, IEEE TENCON2018, pp.1516-15210(2018).
[3]. Liu C.and Layland J.W., Scheduling Algorithms for Multiprogramming in a Hard-RealTime Environment, Journal of the ACM, Vol.20, No.1, pp.46-61(1973)
[4]. OSEK/VDX, OSEK/VDX Operating System Version 2.2.3(2005)
[5]. TOPPERS/ATK1, https://www.toppers.jp/atk1.html
[6]. 原田祐軸，阿部一樹，兪明連，横山孝典，アスペクト指向プログラミングによるリアルタイム OS スケジューラのカスタマイズ，情報処理学会論文誌，Vol.57，No.8，pp.1752-1764(2016).