7K-04

Linux上のリアルタイムOSエミュレータ

川田 陸† 竹原 永也† 横山 孝典† 兪 明連†
東京都市大学‡

1 はじめに

制御処理における遅延は制御性能低下の原因となる [1] ため，デッドラインを守るとともに遅延の少ない分散処理環境が必要である．遅延の少ない分散処理環境に，ノード間で同期して周期的タスクを実行する時間駆動アーキテクチャ (Time-Triggered Architecture)[2] があるが通信時間の変動がなく時刻同期も可能な時間駆動ネットワークが必要である．今後は，無線通信を利用したシステムが増えると予測されており，無線通信は通信時間が変動しやすいため，通信時間の変動を許容できるリアルタイム分散処理環境が求められる．

我々は通信時間の変動を許容するため論理時間を導入した分散処理環境を開発した [3]．しかしながら，制御システム向けの OSEK OS[4] ベースのリアルタイム OS(RTOS) を使用していたため，対象は制御システムに限られていた．制御系と情報系が融合されたシステムが今後増えると予測されるので，制御系向けの RTOS に限らず情報系で広く使用されている組込み Linux への対応が求められる．

そこで本論文では，制御系コンピュータと情報系コンピュータが混在するシステムで論理時間を導入した分散処理ミドルウェアを動作可能とすることを目的に，Linux 上で動作する RTOS エミュレータを提案する．

2 RTOS エミュレータ

本研究が対象とする分散処理環境のソフトウェア構成を図 1 に示す．本 RTOS エミュレータは組込み Linux 上で動作し，本 RTOS エミュレータ上にて動作する分散処理ミドルウェア上でアプリケーションが動作する．

RTOS エミュレータは OSEK OS のタスク管理機能を Linux のスレッドを用いてエミュレートする．OSEK OS のカウンタとアラーム機能を実装し，カウンタを用いて論理時間を表現する．タスクは論理時間におけるデッドライン (論理デッドライン) を用いて EDF(Earliest Deadline First)[5] によりスケジューリングする．RTOS エミュレータが提供する API を表 1 に示す．周期的にカウンタを加算する SignalCounter() を呼び出して，タスクの起動時刻に達するとアラームが満了し，ActivateTask() を呼び出してタスクを起動する．タスクの処理が終了したら TerminateTask() によりタスクは休止状態へと遷移する．

RTOS エミュレータの動作を図 2 に示す．RTOS エミュレータは予めタスク数分のスレッドを生成し，休止状態にして同期待ちする．カウンタ加算処理機能によりアラームが満了すると，指定されたスレッドを再開させる準備をする．指定されたスレッドのみ動作する場合は，論理デッドラインを設定してスレッドを再開させタスク処理を行う．他のスレッドが動作している場合は，各スレッドの優先度を論理デッドラインが近い順番になるように設定して，スレッドを再開させタスク処理を行う．各スレッドは以降，優先度順に動作する．タスク処理が終了したスレッドは再び RTOS エミュレータの同期を待つため休止状態となる．以上がタスク 1 周期分のスレッドの動作である．

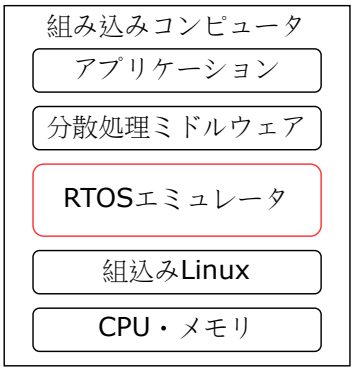

図 1 ソフトウェア構成

表 1 RTOS エミュレータの API

API 名	動作内容
ActivateTask()	タスクを起動させる
TerminateTask()	タスクを休止状態とする
SignalCounter()	カウンタ加算処理 アラーム満了時にタスクの起動

3 非周期タスクへの対応

EDF をベースに非周期タスクに対応するため TBS(Total Bandwidth Server)[6] を導入する．TBS は，非周期タスクに対して疑似的にデッドラインを与えることで，非周期タスクの応答時間を短縮できる方式である．

k 番目の非周期タスクのジョブには式 (1) で求める疑似デッドライン d_k を与える．

$$d_k = max(r_k,\ d_{k-1}) + \frac{C_k}{U_s} \quad (1)$$

r_k は k 番目の非周期タスクのジョブの起動要求時刻，C_k は k 番目の非周期タスクのジョブの実行時間，U_s は非周期タスクに与えるバンド幅（CPU 使用率）を表し，$max()$

Real-Time OS emulator on Linux
Riku Kawata, Eiya Takehara, Takanori Yokoyama, Yoo Myungryun, ‡ Tokyo City University

Copyright ©2023 Information Processing Society of Japan.
All Rights Reserved.

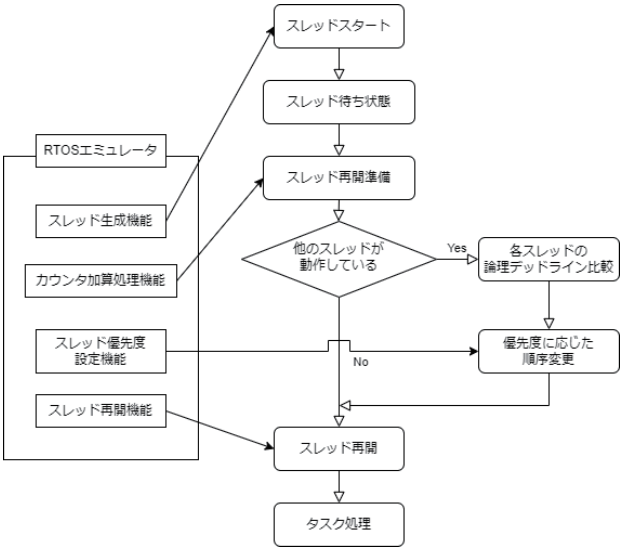

図 2　RTOS エミュレータの動作

表 2　OS からタスクを起動した場合の測定結果

測定対象	実行時間 [μsec]		
	平均実行時間	最大実行時間	最小実行時間
OS から周期タスク起動	41.1μs	96.2μs	21.0μs
OS から非周期タスク起動	72.6μs	112.2μs	66.4μs

で k 番目の起動要求時刻とそのひとつ 前の非周期タスクのデッドラインの遅い方をとる．そして周期タスクと非周期タスクをまとめて EDF でスケジューリングする．複数の非周期タスクは起動要求に対して逐次的に実行される．

4　評価

開発した RTOS エミュレータが実用上問題ない性能であるか評価するため，組込み Linux を使用している Raspberry Pi 3 model B (動作クロック 1．2GHz) を用いてタスクの起動にかかる時間を測定した．

(1)OS からタスクを起動した場合の実行時間

OS から周期タスク及び非周期を起動した場合の測定結果を表 2 に示す．OS から周期タスクを起動した場合の平均実行時間は 41.1μsec，最大実行時間は 96.2μsec である．非周期タスクを起動した場合の平均実行時間は 72.6μsec，最大実行時間は 112.2μsec である．非周期タスクを起動した場合の時間が長くなっているが，これは TBS によるデッドラインの算出を行っているためである．

(2) 周期タスクから他タスクを起動した場合の実行時間

周期タスクから他タスクを起動させた場合の測定結果を表 3 に示す．高優先度な周期タスク及び非周期タスクを起動した場合の平均実行時間はそれぞれ 45.6μsec，82.3μsec である．

実行中の他のタスクから呼び出したタスクに実行権を切り替えるための優先度変更操作などにより，実行時間が増大している．低優先度な周期タスク及び非周期タスクを起動した場合の平均実行時間はそれぞれ 11.4μsec，30.3μsec である．高優先度なタスクを起動した場合の平均実行

表 3　周期タスクから他タスクを起動した場合の測定結果

測定対象	実行時間 [μsec]		
	平均実行時間	最大実行時間	最小実行時間
高優先度周期タスク起動	43.6μs	48.7μs	26.0μs
低優先度周期タスク起動	11.4μs	69.2μs	8.4μs
高優先度非周期タスク起動	82.3μs	233.6μs	53.4μs
低優先度非周期タスク起動	30.3μs	42.0μs	19.3μs

表 4　非周期タスクから周期タスクを起動した場合の測定結果

測定対象	実行時間 μsec		
	平均実行時間	最大実行時間	最小実行時間
高優先度周期タスク起動	52.3 μs	81.7μs	48.8μs
低優先度周期タスク起動	32.4μs	68.8μs	24.8μs

時間 45.6μsec，82.3μsec と比較するとそれぞれ 34.2μsec，52.0μsec 減少している．これは，タスクの実行を開始する前に ActivateTask() 関数を終了するためである．

(3) 非周期タスクから周期タスクを起動した場合の実行時間

非周期タスクを起動させた場合の測定結果を表 4 に示す．高優先度周期タスクを起動した場合の平均実行時間は 52.3μsec，最大実行時間は 81.7μsec である．周期タスクの測定結果と同様に，実行中の他のタスクから呼び出したタスクに実行権を切り替えるために優先度変更操作などにより，実行時間が増大している．

本 RTOS エミュレータが対象としているのはネットワーク通信を伴うタスクであり，その最短周期は 10msec 以上と想定している．したがって (1)(2)(3) で示した実行時間は，実用上問題ない範囲に収まっているものと考える．

5　おわりに

論理時間を導入した分散処理ミドルウェアを組込み Linux 上で動作させるための RTOS エミュレータを開発した．また，TBS を導入することで，非周期タスクの応答時間の短縮を可能とした．今後の課題として RTOS エミュレータを拡張し，イベント制御機能やリソース管理機能を実現することが挙げられる．

謝辞

本研究は JSPS 科研費 JP18K11225，JP21K1815 の助成を受けたものである．

参考文献

[1] Cervin,A., Henriksson,D.,Lincoln,B.,Eker J. and Arzen,K.: How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime,IEEE Control Systems,Vol.23,No.3,pp.16-30 (2003).
[2] Kopetz H.,The Time-Triggered Architecture,proceedings of the IEEE,Vol.91,No.1, pp.112-126,(2003).
[3] Ayumu I, Takanori Y and Myungryun Y :A Time-Triggered Distributed Computing Envirnment for Cyber-Physical Systems Based on Physical Time and Logical Time, Proceedings of TENCON 2018-2018 IEEE Region 10 Conference, pp.1510-1515(2018)
[4] OSEK/VDX, OSEK/VDX Operating System Version 2.2.3, (2005).
[5] Liu C.and Layland J.W.:Scheduling Algorithms for Multiprogramming in a Hard realTime Environment,Journal of the ACMVol.20,No.1,pp.46-61 (1973).
[6] Spuri, M. and Buttazzo, G. C.: Efficient Aperiodic Service under Earliest Deadline Scheduling, Proc. 15th IEEE Real-Time Systems Symposium, pp.2-11(1994).

Copyright ©2023 Information Processing Society of Japan.
All Rights Reserved.