6G-05

# タイムスタンプを用いた 組込みシステム向け時間駆動分散処理環境の提案

市村 歩† 兪 明連† 横山 孝典†
東京都市大学†

## 1. はじめに

近年，実世界の情報取得や制御を行うサイバーフィジカルシステムが増加している [1]．例えば，自動運転システムは外部の情報や車両の情報をセンサで取得し，ECU（Electronic Control Unit）で処理をして，エンジンやモータなどのアクチュエータを制御する．自動車内の ECU 間の通信の他に車車間通信や路車間通信も使用されつつあり，有線及び無線ネットワークを用いた分散型組込み制御システムが必要である．

制御システムにおけるリアルタイムタスクはデッドラインを守ることとジッタの少ないことが要求される．リアルタイム性の優れた分散処理環境として FlexRay などの時間駆動（Time-Triggered）ネットワークを用いた分散処理環境がある [2]．全てのノードで時刻同期をし，周期的にタスクを起動することでジッタの少ない分散処理を実現している．しかし，無線ネットワークなどの通信時間が変動するネットワークには対応していない．

通信時間が変動するネットワークでもジッタの少ない処理をする研究として PTIDES[3] がある．全てのノードで時刻同期をするとともに，入力時のタイムスタンプをベースに，出力までに通過する各ノード上のアプリケーションの最悪実行時間やノード間の最悪通信時間を加算してタイムスタンプを更新し，更新後のタイムスタンプの時刻で出力することで，デッドラインを守るとともにジッタの少ない処理を実現している．しかし，アプリケーションを含めた専用の開発環境を使用し，詳細な時間制約を与える必要があるため，導入は容易ではない．

そこで本論文では，無線ネットワークなどの通信時間が変動するネットワーク環境を対象に，リアルタイム性のある分散処理を，ミドルウェアで実現できる分散処理環境を提案する．

## 2. 分散処理環境の構成と機能

本研究では，詳細な時間制約を与えることなく，入力処理から出力処理までの相対デッドラインと，タイムスタンプで与える入力処理時刻とを用いて，デッドラインを守るとともにジッタの少ない分散処理環境を実現する．現時点で対象としているネットワークは CAN と ZigBee である．

本分散処理環境の構成を図 1 に示す．リアルタイム OS（RTOS）には OSEK OS を用いる．入力処理タスクと出力処理タスクは時間駆動，それ以外の算出処理を行うタスクはイベント駆動で起動する．ミドルウェアは，入力処理タスクや出力処理タスクの起動周期，入力から出力までの相対デッドラインなどを記憶したコンフィギュレーションデータを参照して処理を実行する．コンフィギュレーションデータは，開発者が入力した情報に基づいてコンフィギュレーションツールにより自動生成する．ただし現時点では，コンフィギュレーションツールは未開発のため，手作業でコンフィギュレーションデータを記述している．

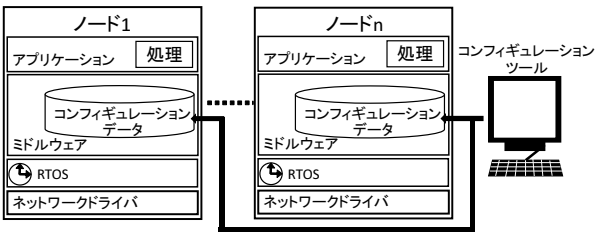


図 1: 分散処理環境の構成

本ミドルウェアは，OSEK COM 仕様に基づくメッセージ送受信用の API として SendMessage(mesID, DataRef) 及び ReceiveMessage(mesID, DataRef) を提供する．mesID はメッセージ ID，DataRef はアプリケーションとミドルウェア間でデータを参照するためのポインタである．

メッセージは入力処理時刻を表すタイムスタンプとアプリケーションの処理に用いるデータから構成される．タイムスタンプの初期値は最初に入力処理タスクを起動した時刻とし，入力処理タスクを起動するたびにその起動周期を加算する．異なる入力処理タスクで取得した複数の入力データを扱う場合には，どの入力データのタイムスタンプを用いるかをコンフィギュレーションデータで指定する．

## 3. ミドルウェアの動作

ミドルウェアの処理は，アプリケーションからの API 呼び出し，メッセージ受信割込み，及びタスク起動時に呼び出すコールバックルーチンにより実行される．分散処理環境の動作を図 2 を用いて説明する．この例は入力処理をする入力ノード，算出処理をする算出ノード，出力処理をする出力ノードから成り，入力データ及び出力データはそれぞれひとつのみの場合である．入力処理タスクと出力処理タスクは同期した時刻に基づいて周期的に起動される．算出処理タスクはメッセージを受信したイベントで起動する．この例ではコンフィギュレーションデータとして，入力処理タスクの最初の起動時刻 50 と起動周期 10 を入力ノードと算出ノードに与え，出力処理タスクの起動周期 10 と相対デッドライン 30 を出力ノードに与える．

A Time-Triggered Distributed Computing Environment using Timestamps for Embedded Systems
†Ayumu Ichimura, Myungryun Yoo, Takanori Yokoyama Tokyo City University

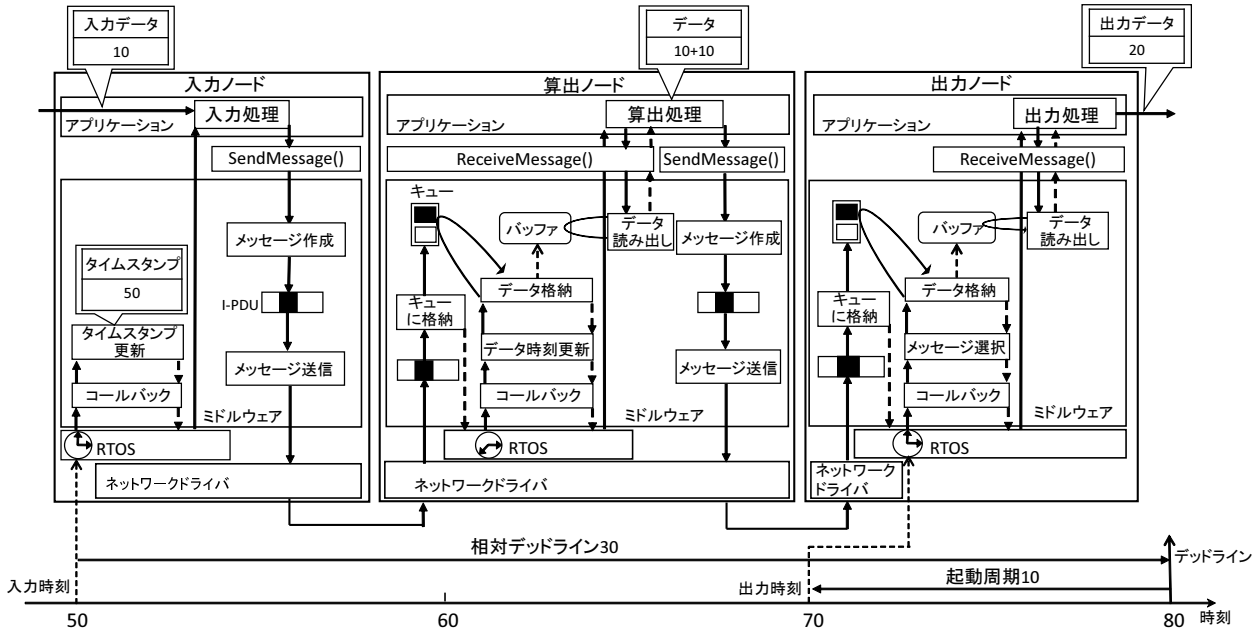


図 2: 分散処理環境の動作

入力ノードでは入力処理タスク起動前のコールバックルーチンにより，入力時刻のタイムスタンプを更新する．この例では，最初にタイムスタンプを 50 に設定し，以降，タスク起動前のコールバックルーチンで周期 10 をタイムスタンプに加算する．入力処理タスクは入力データ（この例では 10）を読み出し，SendMessage() を呼び出している．ミドルウェアはタイムスタンプと引数で与えられた入力データから成るメッセージを作成し，I-PDU（Interaction Layer Protocol Data Unit）にパックして，ネットワークドライバに送信要求を出す．

メッセージを受信した算出ノードでは，受信割込み処理により I-PDU からメッセージをアンパックし，mesID ごとのキューに，タイムスタンプの古い順に並ぶように格納する．また，受信イベントによる算出タスク起動前のコールバックルーチンで，どのタイムスタンプのデータを使用するか判定するためのデータ時刻を更新した後，キューの中からデータ時刻と一致するタイムスタンプをもつデータをバッファに格納する．データ時刻の初期値は入力処理タスクの最初の起動時刻（この例では 50）で，データ時刻と一致するタイムスタンプを見つけるたびに入力処理タスクの起動周期（この例では 10）を加算する．

算出処理タスクが ReceiveMessage() を呼び出すと，バッファからタイムスタンプ順にデータが読み出される．そしてこの例では，算出処理がデータに 10 を加算して SendMessage() を呼び出す．ミドルウェアはデータ（この例では 20）とタイムスタンプ（この例では 50）から成るメッセージを作成し，I-PDU にパックして，メッセージ送信要求を出す．

メッセージを受信した出力ノードの受信割込み処理は，I-PDU からメッセージをアンパックし，mesID ごとのキューに格納する．そして，出力処理タスク（この例では最初の起動時刻は 70，起動周期は 10）起動前のコールバックルーチンにより，キューの中から出力処理タスクで使用するメッセージを選択し，そのデータをバッファに格納する．具体的には，デッドラインの 1 周期前に出力処理タスクを起動するため，「タイムスタンプ＋相対デッドライン-出力タスク起動周期」の値が現在時刻と一致するタイムスタンプをもつメッセージを選択する．この例では，現在時刻 70 に対応するタイムスタンプ 50 をもつメッセージを選択して，データを格納する．そして，出力処理タスクは ReceiveMessage() を呼び出してデータを取得し，出力処理を行う．

## 4. おわりに

本論文では，通信時間が変動するネットワーク環境でもデッドラインを守るとともにジッタの少ない分散処理を実現するためのミドルウェアを提案した．現在は本ミドルウェアの実装及び性能評価を行っており，今後はコンフィギュレーションツールを開発するとともに，より多くのネットワークをサポートできるよう拡張を行う予定である．

## 謝辞

本研究は JSPS 科研費 JP15K00084 の助成をうけたものである．

## 参考文献

[1] E.A.Lee, Cyber Physical Systems: Design Challenges, Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp.363-369, 2008.

[2] H.Kopetz, Should Responsive Systems be Event-Triggered or Time-Triggered?, IEICE Transactions on Information and Systems, Vol.E76-D,No.11, pp.1325-1332, 1993.

[3] J.C.Eidson, E.A.Lee, S.Matic, S.A.Seshia, J.Zou, Distributed Real-Time Software for Cyber-Physical Systems, Proceedings of the IEEE (special issue on CPS), Vol.100, No.1, pp.45-59, 2012.