応答時間を短縮した時間駆動分散処理ミドルウェア

冨田 有紀　横山 孝典　志田 晃一郎　兪 明連
武蔵工業大学

1. はじめに

自動車制御システムや産業システムなど組込み制御システムの多くが，複数の組込みコンピュータによって構成される分散制御システムになってきた．これらのシステムは分散化され，リアルタイムに処理を行っている．リアルタイムシステムの構築方法には，周期に応じて処理を行う時間駆動と，イベントに応じて処理を行うイベント駆動がある [1].

自動車制御のような，厳しい時間制約が課せられているハードリアルタイムシステムでは，時間駆動が使われるのが普通である．また，ディジタル制御は一定の制御周期（サンプリング周期）で処理を行う必要があり，時間駆動が適している．時間駆動を利用することで，ジッタ（周期のずれ）が少ない周期的処理を実現できる．しかし，各コンピュータにおいて同期のための待機時間が発生し，システム全体の応答時間が長くなる可能性がある．

一方，イベント駆動ではイベントに応じて処理を行うため，同期のための待機時間が発生せず，応答時間の短い実行が可能である．しかし，最悪の応答時間の予測が困難であり，ジッタも発生しやすい．

我々は時間駆動とイベント駆動を組み合わせることで，ジッタの発生を抑えるとともに，可能な限り応答時間を短縮する方法を提案し，そのための分散処理ミドルウェアを開発する．

2. 時間駆動とイベント駆動

2.1 分散制御システム

複数のコンピュータがネットワークを介して通信を行っている分散制御システムの例を図1に示す．このようなシステムでは各々のコンピュータの役割が異なり，複数のコンピュータで分散して一連の動作を行う．この例では5台のコンピュータを使用しており，コンピュータ1がセンサ等からの入力処理（サンプリング処理）を行い，コンピュータ2，3，4とネットワークを介してメッセージ通信を行い，コンピュータ5がアクチュエータ等への出力処理を行っている．入力から出力までが，システム全体の応答時間になる．

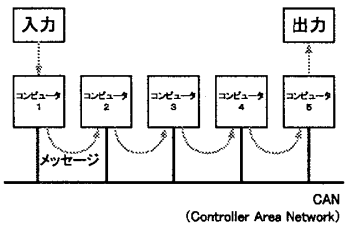

図1　分散制御システム

2.2 時間駆動

図1に示したシステムに時間駆動を適用した場合の処理の流れと，メッセージによるデータの受け渡しを表したタイムチャートの例を図2に示す．縦軸に各コンピュータの処理を配置し，横軸が時間の流れを表している．時間駆動周期はTである．

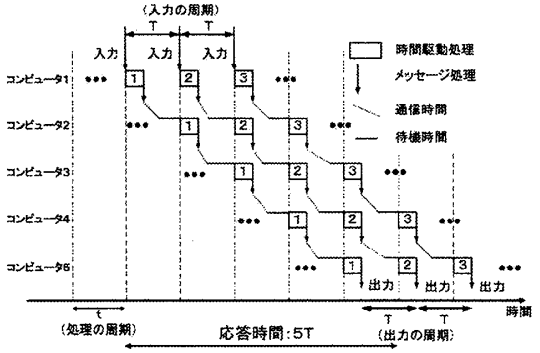

図2　時間駆動のタイムチャートの例

コンピュータ1が処理を終えるとコンピュータ2にメッセージを送る．時間駆動では，コンピュータ2はメッセージを受信した後，次の周期の開始を待って処理を開始する．全コンピュータが同期して周期Tで処理を行うため，ジッタは発生しにくい．

しかし，時間駆動では，上記のようにメッセージ受信から次の周期の開始までの待機時間が発生する．応答時間には処理のみでなく，同期のための待機時間も含まれ，この図の例では最大応答時間は5 Tとなる．

2.3 イベント駆動

イベント駆動システムの処理の流れと，メッセージによるデータの受け渡しを表したタイムチャートの例を図3に示す．

コンピュータ1は，一定周期Tで入力（サンプリング）処理を行っているが，それ以降のコンピュータではメッセージ受信のイベントで処理を開始する．

コンピュータ1が処理を終えるとコンピュータ2にメッ

A Time-Triggered Distributed Computing Middleware to Improve End-to-End Response Time
Yuki Tomita, Takanori Yokoyama, Koichiro Shida and Myungryun Yoo
Musashi Institute of Techonology

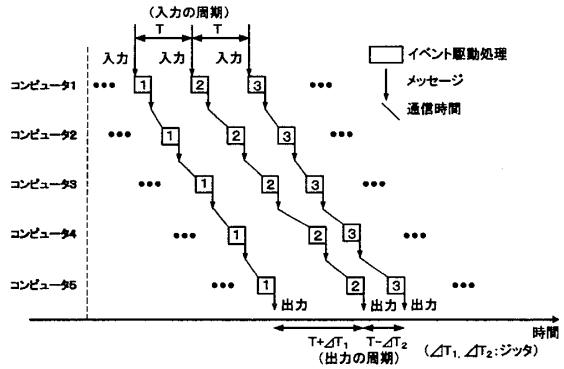

図 3 イベント駆動のタイムチャートの例

セージを送る．イベント駆動では，コンピュータ2はメッセージを受信すると直ちに処理を開始するので，時間駆動のような待機時間は発生しない．したがって，時間駆動と比較し，応答時間を短縮できる．

しかし，いくつものコンピュータを介するとき，ネットワークの通信時間の変動によりジッタが発生しやすい．

3. 提案手法

我々は，ジッタが発生しにくいという利点を持つ時間駆動と，応答時間の短い処理が可能なイベント駆動との，両者の利点を活かせる実行方式を提案する．具体的には，制御性能に影響する入力処理と出力処理にはジッタの発生しにくい時間駆動を，その間の処理には同期のための待機時間が発生しないイベント駆動を適用する．図4に時間駆動とイベント駆動を利用した実行方式のタイムチャートの例を示す．

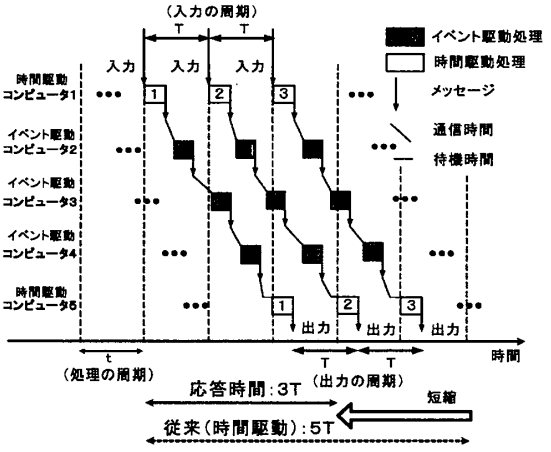

図 4 提案する実行方式のタイムチャートの例

入出力処理以外の処理をイベント駆動で行うため，時間駆動で発生した待機時間は発生しない．入出力処理を時間駆動で行うため，出力処理にジッタが発生しにくい．よって，ジッタの発生が少なく，応答時間の短い処理が可能となる．図2の時間駆動の応答時間は5 T であったが，本手法を適用した図4の例では3 T に短縮されている．

ただし，実際のシステムでどの程度応答時間を短縮できるかはアプリケーションに依存する．また，そのためのスケジューリングも必要である．

4. 分散処理ミドルウェア

前節で提案した実行方式を実現するための分散処理ミドルウェアを開発した．図5にミドルウェアの構成を示す．

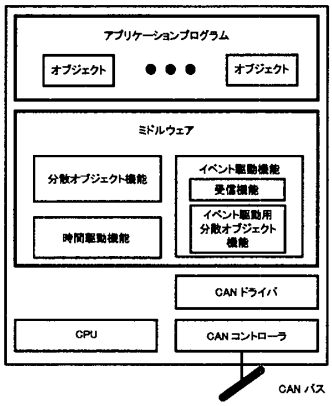

図 5 提案する分散処理ミドルウェア

本ミドルウェアは，以前我々が開発した時間駆動分散処理ミドルウェア [2] に，イベント駆動機能を追加することで実現した．メッセージ受信割り込みにより処理を起動することで，イベント駆動処理を行わせる．本ミドルウェアは自動車制御を主な対象とし，ネットワークには自動車制御でよく用いられる優先度制御可能なネットワークである，CAN（ Controller Area Network ）を使用した．

5. おわりに

本研究では，入出力以外の処理を行うコンピュータにイベント駆動を適用することで，同期のための待機時間をなくして応答時間を短縮できる時間駆動分散処理方式を提案し，そのためのミドルウェアを開発した．

今後の課題は，本方式を実現するためのスケジューリング方法を検討することである．

参考文献

1) Kopetz, H.：Should Responsive Systems be Event-Triggered o0r Time-Triggered?, *IEICE Transaction on Information & Systems*, Vol. E76-D, No. 11, pp. 1325–1332, 1993.
2) 石郷岡祐，横山孝典：組み込み制御システム向け時間駆動分散オブジェクト環境,情報処理学会論文誌, Vol.48, No.9, pp.2936–2945, 2007.