1J-8

時間駆動オブジェクト指向モデルに基づく組み込み制御システム向け分散処理環境

石郷岡　祐　　横山　孝典　　志田晃一郎
武蔵工業大学

1　はじめに

CORBA[1]に代表される分散オブジェクト技術が普及し，リアルタイムシステム分野への適用も進みつつある．

リアルタイムシステムの構成法にはイベントに応じて処理を行なうイベント駆動アーキテクチャと時間の周期に応じて処理を行なう時間駆動アーキテクチャがある[2]．自動車制御のような時間制限が厳しい組み込みシステムでは，ディジタル制御が一定のサンプル周期による処理を基本としていることや，最悪の反応時間を予測しやすく，時間保証が容易なため，処理を周期的に実行する時間駆動が望ましい．イベント駆動アーキテクチャはCORBAのような分散オブジェクトを用いて容易に実現できる．しかし，時間駆動に基づくシステム構成法として，時間駆動オブジェクト指向モデル[3]が提案されているが，時間駆動に基づく分散オブジェクト環境が実現されるには至っていない．

2　時間駆動オブジェクト指向モデル

時間駆動オブジェクト指向モデルの基本モデルを図1に示す．このモデルは，処理に必要なデータ値を算出するメソッドupdate()を持つオブジェクトの集合から成る．オブジェクト間のやりとりは，算出結果を参照するメソッドget()の呼び出しのみなので，メソッド呼び出しのネストを制限することができ，時間予測が容易になる．

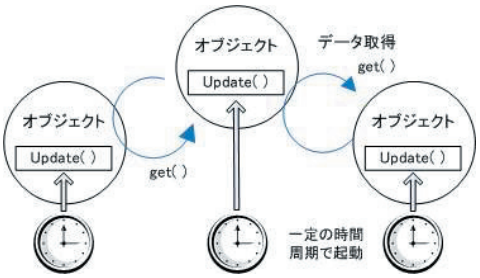

図1：時間駆動オブジェクト指向モデル

A Distributed Computing Enviroment based on a Time-Triggered Object-Oriented Model for Embedded Control Systems
Tasuku Ishigouoka, Takanori Yokoyama and Koichiro Shida, Musashi Institute of Techonology

分散処理環境では，オブジェクトが他のコンピュータからも参照されるとき，そのレプリカを参照するコンピュータ上に設け，周期的にオブジェクトのデータをレプリカにコピーする．これにより，メソッドget()の呼び出しにネットワークを介さないデータ取得が可能となり，時間駆動に基づいた分散処理を実現できる．

そこで本研究の目的は，時間駆動に基づいた分散処理を行なうための時間駆動分散オブジェクトアーキテクチャを提案し，そのミドルウェアを開発することである．

3　時間駆動分散オブジェクトアーキテクチャ

本研究で提案する時間駆動分散オブジェクトアーキテクチャを図2に示す．

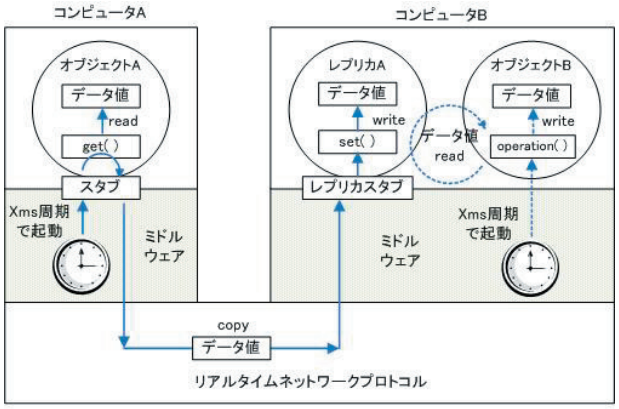

図2:時間駆動分散オブジェクトアーキテクチャ

オブジェクトAとBは，制御に必要なデータ値を算出するメソッドを持ち，周期的に起動することで時間駆動に基づく動作を実現する．get()はデータを取得する処理を行ない，set()はデータを保存する処理を行なう．update()は他のオブジェクトのデータ値を必要とする演算であり処理結果をデータ値に保存する．

時間駆動に基づく分散処理を行なうためにレプリカの設置，レプリカの更新を行なう機能が必要となる．レプリカの更新を行なうため，オブジェクト本体側にスタブ，レプリカ側にレプリカスタブを設ける．

スタブはオブジェクトからレプリカへ転送する複数のデータを読み出し送信メッセージにまとめる機能，

レプリカスタブは受信メッセージ中のデータを複数のデータに戻し，レプリカに保存する機能を持つ．

4　ミドルウェア

ミドルウェアは以下の機能を持つ．

(1) ネットワーク全体で処理の同期を合わせる．

(2) オブジェクトの変数をレプリカの変数に周期的にコピーする．

(3) オブジェクトがレプリカの変数を参照する処理と，(2) がレプリカの変数へデータを書き込む処理が衝突するのを防ぐ．

(4) 異なる時間周期を混在させることができる．

各々の機能は次のように実装する．(1) は，ネットワークにハードウェアタイマを合わせる信号を流し同期を合わせる．(2) は，各々のスタブを呼び出してデータを対象のコンピュータごとにまとめ，ネットワークで送信する．そして受信後，レプリカスタブによりレプリカへコピーされる（図 3）．これを一定の時間周期毎に行なう．(3) は，タスクに優先度を付け，処理の衝突に起こらないようにする．(4) は，各々の周期で行なうべき処理をまとめ，それらを呼び出すことで複数の異なる時間周期の混在を可能にする．

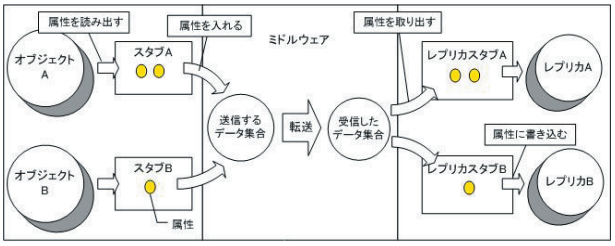

図 3：コピー処理の流れ

以上の機能を実装し，自動車制御に良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）に対応したミドルウェアのプロトタイプを作成した．

5　スタブ生成とコンフィギュレーション

オブジェクト毎に生成する必要があるスタブ，レプリカ，レプリカスタブは，IDL（Interface Definition Language）[1] により記述したオブジェクトのインタフェース定義から，IDL コンパイラによって自動生成する．例えば，IDL を次のように記述する．

```
interface objA{
        attribute short value;
        .........
}
```

これを IDL コンパイラでコンパイルすると，C 言語マッピングの場合はオブジェクトを構造体で書くという規則 [3] に基づき以下を含むコードを生成する．

```
レプリカの構造体 ･･･ struct _objA{
                          short value;
                     };
レプリカの宣言 ･･･  struct _objA _objA_replica;
アクセス関数（get 関数）･･･ short objA_get_value();
アクセス関数（set 関数）･･･ void objA_set_value(short);
スタブ ･･･ unsigned char objA_pack(unsigned char);
レプリカスタブ ･･･ void objA_unpack(unsigned char*);
```

objA_pack() 関数は，get 関数で得たデータをミドルウェアの送信用バッファに格納するスタブである．objA_unpack 関数は，set 関数を呼び出し受信用データバッファのデータをレプリカに書き込むレプリカスタブである．

またシステム構築時まで明確にならない設定はコンフィギュレータによって生成する．例えば，ハードウェア依存する部分や通信環境に関する部分が挙げられる．

なお，現時点では IDL コンパイラとコンフィギュレータは未実装であり，ハンドコンパイルによりスタブ，レプリカスタブの生成，システム構築の設定を行なっている．

6　おわりに

時間駆動分散オブジェクトアーキテクチャを提案し，その実行環境であるミドルウェアのプロトタイプを作成した．

今後の課題として，開発したミドルウェアの評価や，IDL コンパイラとコンフィギュレータの実装が挙げられる．また，イベント駆動にも対応することによって，時間駆動とイベント駆動が混在したシステムに対応するため，既存の CORBA との統合化を行なうことを検討している．

参考文献

[1] Object Management Group, Common Object Request Broker Architecture:Core Specification, Version 3.0.3, 2004

[2] H.Kopetz, Should Responsive Systems be Event-Triggered or Time-Triggered? IEICE Trans. Inf. & Syst., Vol. E76-D, no.11 pp.1325-1332, 1993.

[3] 横山 孝典ほか，組込み制御システムのための時間駆動オブジェクト指向ソフトウェア開発法，電子情報通信学会論文誌，D-Ⅰ，Vol.J84-D-Ⅰ，No.4 pp.338-349，2001年