第10回(社)計測自動制御学会
システムインテグレーション部門講演会

10th SICE System Integration Division
Annual Conference

SICE SI 2009

講演概要集

SICE®

SICE SI
SYSTEM INTEGRATION DIVISION

2009.12.24-26
芝浦工業大学　豊洲キャンパス

コンポーネント間の多様なポリシに基づく送受信方式を実現するデータポートの実装

– OpenRTM-aist-1.0 の新しいデータポート –

産業技術総合研究所 ○安藤慶昭, 栗原眞二, 神徳徹雄,
静岡大学 清水昌幸,
高エネルギー加速器研究機構 仲吉一男, 安芳治

Implementation of Data Port which Supports Data Transfer with Various Policies

– New data port in the OpenRTM-aist-1.0 –

*Noriaki ANDO, Shinji KURIHARA, Tetsuo KOTOKU, AIST
Masayuki SHIMIZU, Shizuoka Uuniversity,
Kazuo NAKAYOSHI, Yoshiji YASU, KEK

Abstract— The data exchange method between RT-Components can take various policy, and the policy can be decided according to both ends' RT-Component feature and their relations. Since the connection between RT-Components is established dynamically, these policies have to be given in run-time. In this paper, data exchange functionality for Data-Port which supports various policy is realized on OpenRTM-aist-1.0.

Key Words: RT(Robot Technology), middleware, network

1. はじめに

RT ミドルウエアの著者らによる実装である OpenRTM-aist では、RT コンポーネント間のデータ指向の通信方法としてデータポートを、サービス指向の通信方法としてサービスポートと呼ばれる機構を提供している。RT コンポーネント間のデータ通信方法は、接続されている両端の RT コンポーネントの種類、実行方法、データ生成・処理の周期や速度、RT コンポーネント間の関係に応じて、様々な方法が考えられる [1]。本稿では、先の検討結果を踏まえて実装された OpenRTM-aist-1.0 用の、設計および各種機能を示す。

2. InPort/OutPort

RT コンポーネントはデータ指向型のコンポーネント間通信の枠組みとしてデータポートと呼ばれるポートを持つ。他のコンポーネントからデータを受け取るポートを InPort、他のコンポーネントにデータを送るポートを OutPort と呼ぶ。アクティビティと呼ばれる主たる処理を行う部分では、InPort で受信されたデータを処理し、アクティビティで処理または生成されたデータは OutPort から他の RT コンポーネントに送信される。

RT システムにおいては、モジュール間の関係により、同期または非同期通信が行われたり、すべてのデータの到達を保証するか、逆にベストエフォート型でデータを伝送するといったケースが存在する。多様な通信特性が混在するため、データポートにはこれらに対応できるデータ交換方法が求められる。

3. データポートの設計

[1] での検討に基づき、図 1 に示すコネクタという概念を導入した。

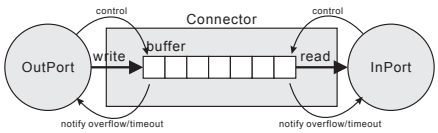

Fig.1 The relation among data ports and connector.

コネクタはバッファおよび通信路を抽象化したオブジェクトである。OutPort とバッファ間では書き込み、制御およびバッファフル状態の通知およびタイムアウト、InPort とバッファ間では読み出しと、制御およびバッファエンプティ状態の通知およびタイムアウトの機能が必要となる。また、これらの機能のために、OutPort/InPort 一対に対して、それぞれ一つコネクタが存在する必要があることがわかる。

さらに、コネクタをサブスクリプション型に対応した実装レベルでモデル化するにあたり、パブリッシャと呼ばれる非同期通信のためのオブジェクトを導入した。これを図 2 に示す [1]。

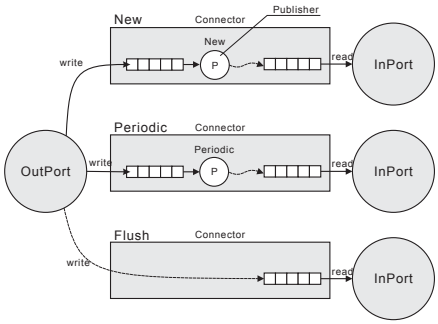

Fig.2 The data port architecture.

図 1 に示すように、OutPort–バッファ間、バッファ–InPort 間で制御メッセージがやりとりされるため、図 2 に示すようにコネクタ毎にバッファを持たせた方が設計の見通しが良い。

4. データポートの実装

4·1 コネクタ

上述したように、コネクタは OutPort-InPort 間の通信路の論理表現したオブジェクトであり、接続の確立によりコネクタオブジェクトとそれに付随するバッファオブジェクトや必要に応じてパブリッシャオブジェクトが生成されコネクタオブジェクトに関連付けられる [1]。コネクタは以下 (表 1) のインターフェースを持つ基底クラスから派生させる。

Table 1 コネクタのインターフェース

`name`	接続名の取得
`profile`	ConnectorProfile の取得
`disconnect`	接続の解除
`activate`	接続のアクティブ化
`deactivate`	接続の非アクティブ化
`getBuffer`	バッファの取得

今回の実装においては、間に CORBA による通信を利用するため、InPort と OutPort それぞれの側に対応するコネクタを定義した。一つは、Push 型のデータ通信をサポートする `InPortPushConnector` / `OutPortPushConnector` の対であり、他方 Pull 型の通信を実現する `InPortPullConnector` / `OutPortPullConnector` である。

実際の通信部分はコネクタオブジェクトに隠蔽され、OutPort からは単にバッファへの書き込みとバッファ状態の通知のみが行われる。

4·2 バッファインターフェース

先の検討 [1] では、接続ごとにバッファを設ける必要性、また生産者・消費者問題への対応として、バッファ状態検出機能、タイムアウト機能、バッファ操作機能が必要であることが示された。この指針に従い、新たなバッファの実装を行った。新たなバッファは、バッファ状態の検出やタイムアウト機能を実現するため、バッファへの読み書き操作手順を細分化したインターフェース (表 2) を持つ。

Table 2 バッファのインターフェース

`advanceRptr()`	読み出しポインタを進める (戻す)
`advanceWptr()`	書き込みポインタを進める (戻す)
`empty()`	バッファが空かどうかを調べる
`full()`	バッファが一杯かどうかを調べる
`get()`	現在のポインタのデータを読み出す
`put()`	現在のポインタにデータを書き込む
`read()`	ポインタ操作を含めた読み出し
`write()`	ポインタ操作を含めた書き込み

また、バッファの状態を呼び出し側に通知する必要性から、以下の表 3 ようなリターンコードを定義した。

なお、C++言語での実装においては、バッファオブジェクトを外部からロードしたモジュールから動的に生成することを想定しているため、例外的な状態通知に関しても、例外機構ではなく戻り値を利用することとした。

Table 3 バッファの状態を通知するリターンコード

`BUFFER_OK`	正常終了
`BUFFER_ERROR`	エラー終了
`BUFFER_FULL`	バッファフル状態
`BUFFER_EMPTY`	バッファ空状態
`NOT_SUPPORTED`	サポート外の要求
`TIMEOUT`	タイムアウト
`PRECONDITION_NOT_MET`	事前状態を満たさない

OutPort・InPort 間の生産者・消費者問題として見た場合、バッファは両者の生産量、消費量の差を吸収するために利用される。[1] で示したように、バッファの境界条件において、生産者では上書きまたはタイムアウト、消費者では無視もしくはタイムアウトするといった振る舞いを定義できる。これらの振る舞いを実行時に選択する設定機能を設けた。

4·3 Push ポリシー

データフロー型が Push、かつサブスクリプション型が New または Periodic の場合、すなわちデータ転送に別スレッドを用いて非同期通信を行う場合、バッファに保持されている複数のデータをどのように送るかを指定する必要がある。

データ送信のポリシーとして、以下の 4 つのタイプを実装した。

`All`	バッファ内データを一括送信
`FIFO`	バッファ内データを FIFO で 1 個ずつ送信
`NEW`	バッファ内データを LIFO で 1 個ずつ送信
`SKIP`	バッファ内データを間引いて送信

これらのポリシーの選択は、接続時に行う必要があるため、設定ファイルもしくは接続時のプロファイルにて指定するように実装した。

4·4 シリアライズ機構

データを通信路経由で送受信する場合、かつ異なる言語間で送受信する場合、一般にマーシャリングと呼ばれるデータの直列化 (シリアライズ) 処理が行われる。CORBA を利用する場合は、マーシャリングは暗黙的に行われるが、本実装ではデータポート内で直接データのシリアライズを行い、バッファにはシリアライズされたデータを保持する構造とした。これにより、CORBA 以外の共有メモリや TCP ソケットを直接利用する場合でも、単なるバイナリデータ列を送受信すればよい。なお、シリアライズの方法には CDR (Common Data Representation) 方式を用いた。

5. おわりに

本稿では、RT コンポーネント間のデータの送受信を行うデータポートに関して、[1] の考察結果を踏まえ、新たなデータポートを設計し、コネクタ、バッファ、パブリッシャの実装を行った。これらの実装により、目的とする種々の通信方法が実現できることが確かめられた。今後は、共有メモリ型を利用したコネクタや、TCP ソケットを直接利用したコネクタを実装し、通信方式の拡張を行う予定である。

参考文献

[1] RT コンポーネント間のデータ送受信方法に関する考察, 安藤 慶昭, 清水 昌幸, 神徳 徹雄, 日本機械学会 ロボティクス・メカトロニクス講演会 2008, p.1P1-E08, 2008/06