2008 JSME Conference on Robotics and Mechatronics
主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

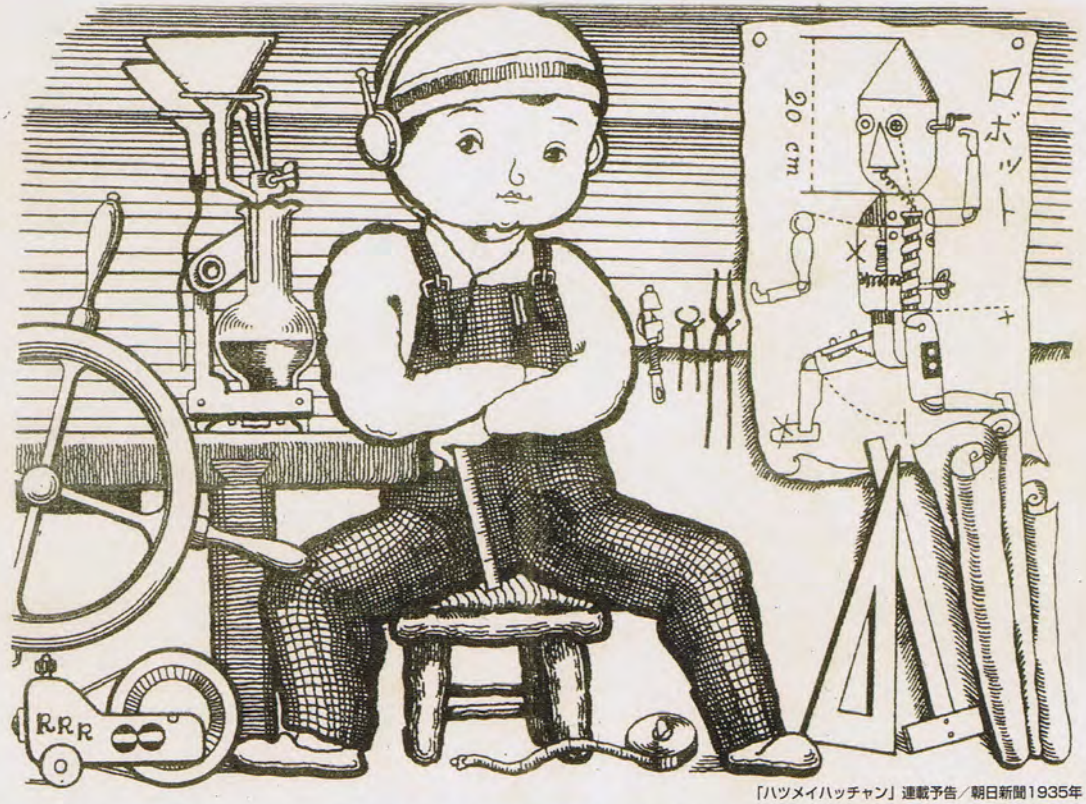
ROBOMEC 2008 in NAGANO

CONFERENCE DIGEST

ROBOTICS AND MECHATRONICS FOR SUSTAINABLE INDUSTRIAL DEVELOPMENT

持続的な産業発展を支えるロボティクス・メカトロニクス

Thu. 5th ~ Sat. 7th June, 2008　ビッグハット（長野市若里多目的スポーツアリーナ）



「ハツメイハッチャン」連載予告／朝日新聞1935年

長野県が生んだ世界的童画家の武井武雄は1935年にロボットが主人公として登場する「ハツメイハッチャン」という新聞マンガの連載をはじめました。カレル・チャペックが戯曲「R.U.R.」ではじめてロボットという言葉を発表したわずか15年後のことでした。

武井武雄の世界 イルフ童画館　長野県岡谷市 http://www.ilf.jp/

社団法人 日本機械学会
〒160-0016 東京都新宿区信濃町35番地 信濃町煉瓦館5階　TEL(03)5360-3500　FAX(03)5360-3508

RT コンポーネント間のデータ送受信方法に関する考察

A study of data exchange scheme between RT-Components

○正 安藤 慶昭 (産総研)　正 清水 昌幸 (産総研)　正 神徳 徹雄 (産総研)

Noriaki ANDO, National Institute of Advanced Industrial Science and Technology, n-ando@aist.go.jp
Masayuki SHIMIZU, National Institute of Advanced Industrial Science and Technology
Tetsuo KOTOKU, National Institute of Advanced Industrial Science and Technology

In the RT-Middleware, a data-centric communication method between RT-Components(RTCs) called "Data Port" is provided. Many types of the data-centric communication schemes, which depend on both ends RT-Component properties such as execution method, data producer's/consumer's processing throughput and period and relationship between them, can be possible. In this paper, we discuss about the data-centric communication scheme between RTCs and its new architecture is shown.

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

1. はじめに

著者らは、RT のソフトウエアモジュール化を促進し、再利用性を向上させるためのプラットフォームとして RT ミドルウェアおよび RT コンポーネントを提案してきた [1, 2]。

RT ミドルウエアでは、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 (図 1)。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。本稿では、データポートを様々なシステムに対応させるため、これらデータ通信方式に関して詳細に検討を加え、RT コンポーネントの新たなデータポートの設計に関して議論する。

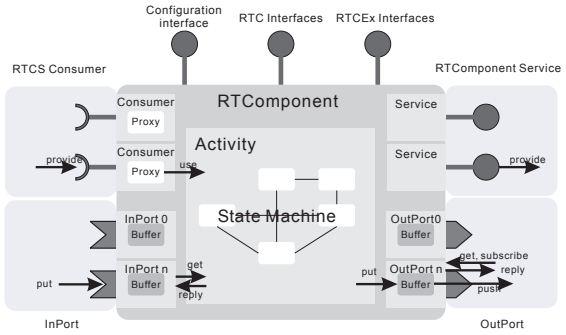


Fig.1 The architecture of RT-Component

2. InPort/OutPort

著者らはこれまで、RT コンポーネントのオブジェクトモデルとして、図 1 に示モデルを提案してきた。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。アクティビティは 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 システムの一般的な傾向として、モジュール間の通信頻度は上位系になるほど散発的になり、下位系では通信頻度が高い。一方、通信 1 回あたりのデータ量は、上位系では比較的大きく、下位系では小さい傾向がある。しかしながら、画像データ通信のように、通信頻度、データ量ともに大きい通信も存在する。このように、RT システムでは、多様な通信特性が混在していることがわかる。

3. 接続・通信の多様性

ここでは、データポートの接続および通信方式について整理するとともに、現在著者らが開発している RT ミドルウエア:OpenRTM[2] のデータポートにおいて問題となる点を議論する。

3.1 接続トポロジ

データポートは OutPort から InPort へ一方通行のデータチャネルとしてモデル化されている。したがって、多対多の接続は考慮しない。また、InPort は送信元を区別しないため単一の InPort が複数の OutPort からデータを受け取るモデルも除外する。すなわち、接続形態として

- 「単一 OutPort」 対 「単一 InPort」
- 「単一 OutPort」 対 「複数 InPort」

の関係が存在する。現在、OpenRTM では、一部の例外を除き「単一 OutPort」 対 「複数 InPort」の通信をサポートしていない。

3.2 データ送受信方法

OutPort から InPort へデータを伝送する際には、OutPort から一方的にデータを送信する Push 型通信と、InPort 側からのリクエストに応じて OutPort がデータを送信する Pull 型通信が考えられる (図 3)。

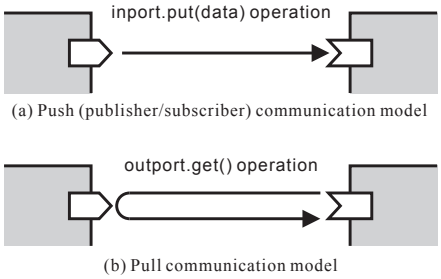


Fig.2 Push/Pull communication model between data ports.

Push 型はデータが生成されたタイミングでそれを送ることができるため、データが定期的・連続的に生成・処理されるシステムにおいては都合がよい。一方、Pull 型は処理を行う側が任意のタイミングでデータを取得できるため、不要なデータの伝送を避けることができる。

RT システムにおいては、Push 型、Pull 型ともに必要であり、どちらの方式を使用するかはモジュール間の関係に依存するため、モジュール作成時には決定できない。

3.3 データ送信タイミング

Pull 型通信では、データ送信タイミングは InPort 側のアクティビティが決定するため、ロジックに依存する。一方、Push 型通信では、OutPort 側のアクティビティのデータ生成タイミングとデータ送信タイミングは必ずしも一致するとは限らず、様々なデータ送信タイミングが考えられる。そこで、OpenRTM ではこれをサブスクリプション型と呼び、New 型、Periodic 型、Flush 型の三種類を定義している。

New 型 OutPort 側のデータ生成と非同期で生成後可能な限り早く InPort にデータを送信する。

Periodic 型 OutPort 側のデータ生成と非同期で、一定周期でデータを送信する。

Flush 型 OutPort 側のデータ生成と同期して InPort にデータを送信する

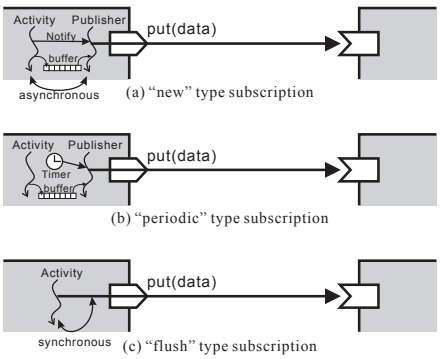


Fig.3 Subscription types between data ports.

3.4 バッファ

上記 Push 型のサブスクリプション型のうち、非同期型の New 型や Periodic 型では、送信側と受信側の処理速度の違いを許容している。これはいわゆる OutPort・InPort 間の生産者・消費者問題に帰着することができる。一般にこの問題は、生産者と消費者間にバッファを設け、バッファオーバーフロー時、生産者はバッファが一杯なら待ち、消費者はバッファがいっぱいでなくなれば生産者に通知する。逆にバッファアンダーフロー時には、消費者はバッファが空なら待ち、生産者はバッファが空でなくなったら消費者に通知することにより、協調的に動作する。

さらに RT システムにおいては、多くの生産者・消費者が固定周期で処理を行い、データ到着を待たずに処理を続行したり、その処理においても最新値のみが必要である場合などがあるため、上記に加えて以下の場合が考えられる。

- バッファオーバーフロー時の生産者の振舞い
  - 無視して上書きする
  - タイムアウトする
- バッファアンダーフロー時の消費者の振舞い
  - 無視する
  - タイムアウトする

以上の考察により、バッファには以下の機能が必要であると考えられる。

- オーバーフロー検出機能
- アンダーフロー検出機能
- タイムアウト機能
- バッファ操作機能

3.5 通信路の特性

RT コンポーネントのデータポート間は、通常 TCP ソケットや CORBA 等を用いて、ネットワーク経由でデータの送受信が行われる [2]。現在の OpenRTM のデータポートでは通信が断絶した際には自動的に接続が解除される。しかしながら、アクティビティ側でこれを検知する方法がないため、送信できなかったデータを退避・再送するなどの処置をとることができない。また、同一プロセス内での通信においても、ポート間のデータコピーが発生するため、画像等の大容量データの通信には向かないといった問題がある。これを解決する方法として、

- 通信路断絶の通知機能
- ポート間のバッファ領域共有機能

等が考えられる。なお、バッファ領域共有機能とは、ポート間でバッファメモリ領域を共有することでデータコピーのオーバーヘッドを避ける機能である。

4. データポートの設計

以上の議論を考慮しデータポートの設計を行った。単一 OutPort に対して複数 InPort が接続され、かつそれぞれについて、異なるデータ送信タイミングが設定可能であるとする。この仕様を満たすために、図 4 に示すコネクタという概念を導入する。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。上述の議論から、OutPort とバッファ間では書き込み、制御およびオーバーフロー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、制御およびオーバーフローの通知およびタイムアウトの機能が必要であることがわかる。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。

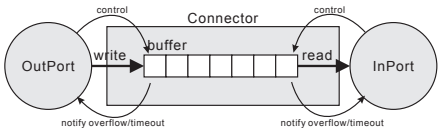


Fig.4 The relation among data ports and connector.

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 5 に示す。パブリッシャオブジェクトは既存の OpenRTM においても既に導入されているが、本モデルではこれをコネクタに統合した。

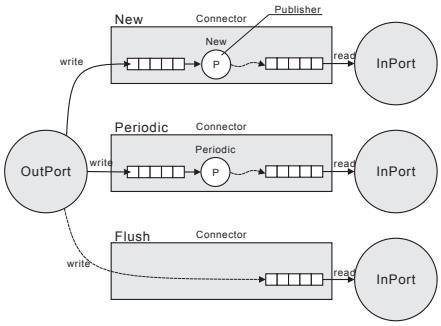


Fig.5 The data port architecture.

図 4 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 5 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。また、上述したバッファの共有に関して、図 5 から、バッファ共有データポートには、Flush 型のサブスクリプション型しか存在しないこともわかる。

5. おわりに

本稿では、データポートを様々なシステムに対応させるため、接続トポロジ、送受信方法、送信タイミング、バッファ、通信路の特性といった観点から、RT コンポーネント間のデータ通信方法について議論し、既存の RT ミドルウエアの問題点を明らかにした。議論を元に、データポートにコネクタおよびパブリッシャという概念を導入し設計を行った。なお、バッファ共有機能は 2 個以上の RT コンポーネント間の関係も考慮する必要があるため本稿では割愛した。今後、本稿での設計に基づき新たなデータポートを設計し、多様なシステムに適用しパフォーマンス等の評価を行う。

参考文献

[1] 安藤, 末廣, 北垣, 神徳, 尹, 「RT 要素のモジュール化および RT コンポーネントの実装」, ロボティックシンポジア予稿集, pp.288-293, 2004.

[2] 安藤, 神徳他, "OMG RTC 標準仕様に準拠した RT ミドルウエアの実装 OpenRTM-aist-0.4.0 新機能の紹介", 日本機械学会 ロボティクス・メカトロニクス講演会 2007, p.1P1-A02, 2007.05